GE Fanuc Automation

Programovatelné řídicí systémy

Motion Mate™ DSM314 pro PLC Series 90™-30

Příručka pro uživatele

GFK-1742B-CZ *březen 2003*

Výstrahy, upozornění a poznámky tak jak jsou používané v této publikaci

Výstraha

Výstražná upozornění se v této publikaci používají ke zdůraznění nebezpečného napětí, proudu, teploty nebo jiných stavů vyskytujících se na tomto zařízení nebo jiných stavů, které by mohly být spojené s jeho používáním a které by mohly způsobit zranění osob.

V situacích, kde by nepozornost mohla způsobit buď zranění osob nebo poškození zařízení, se používá Výstražné upozornění.

Upozornění

Upozornění se používají tam, kde by mohlo dojít k poškození zařízení, pokud by obsluha nedávala pozor.

Poznámka: Poznámky pouze upozorňují na informace, které jsou zejména důležité pro pochopení a obsluhu zařízení.

Tento dokument obsahuje informace, které byly k dispozici v době jeho publikování. I když byla věnována maximální snaha přesnosti, cílem zde obsažených informací není zahrnout všechny podrobnosti nebo odchylky v hardwaru nebo softwaru ani postihnout všechny možné souvislosti ve spojitosti s instalací, obsluhou nebo údržbou. Mohou zde být popisované vlastnosti, které se u hardwarových a softwarových systémů nevyskytují. GE Fanuc Automation nepřijímá žádné závazky upozornit majitele této dokumentace na změny provedené později.

GE Fanuc Automation nepřijímá žádné stížnosti ani záruky, přímé nebo zákonné, a nepřebírá žádnou zodpovědnost za přesnost, úplnost, dostatečnost nebo užitečnost zde obsažených informací. Nejsou poskytovány žádné záruky obchodovatelnosti nebo vhodnosti.

Obsah tohoto manuálu

Tento manuál popisuje Motion Mate DSM314, úplný integrovaný systém řízení pohybu ve formě inteligentního programovatelného přídavného modulu pro programovatelné automaty (PLC) Series 90-30.

Související publikace

Související informace najdete v následujících publikacích:

Manuál pro instalaci programovatelného automatu Series 90™-30, GFK-0356

Požadavky na instalaci v souladu s normami, GFK-1179

Manuál popisu servozesilovačů α Series (SVU), GFZ-65192EN

Řídicí zesilovač motorů α Series (SVM), GFZ-65162E

Manuály servomotoru α Series, GFZ-65165E, GFZ-65150E, GFZ-65142E

Průvodce specifikací serva β Series, GFH-001

Uživatelský manuál serva SL Series; GFK-1581

Uživatelský manuál softwaru VersaPro™, GFK-1670

Krátký úvod pro CIMPLICITY® Machine Edition Logic Developer-PLC , GFK-1918

Kapitola 1

Přehled výrobku

Motion Mate DSM314 je vysoce výkonný modul řízení pohybu více os, který se snadno používá, má vysoký stupeň integrace s logikou a komunikačními funkcemi PLC Series 90-30.

DSM314 podporuje dvě konfigurace primární smyčky řízení:

- Standardní režim (smyčka řízení vlečené osy zakázána)
- Vlečný režim (smyčka řízení vlečené osy povolena)

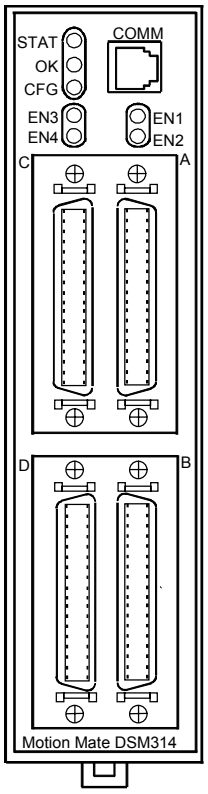
Podporované typy serva

- Digitální – jsou podporovány digitální servozesilovače a motory GE Fanuc α Series a β Series. Popis těchto výrobků je uvedený v publikaci GFK-001.
- Analogová – jsou podporována analogová serva GE Fanuc SL Series a serva jiných výrobců s analogovým rozhraním rychlostních povelů a analogovým rozhraním povelů krouticího momentu. Popis serv GE Fanuc SL-Series je uvedený v Uživatelském manuálu serva SL Series, GFK-1581.

Vlastnosti Motion Mate DSM314

Vysoký výkon

- Řízení serv GE Fanuc pomocí digitálního signálního procesoru (DSP)
- Doba zpracování bloku menší než 5 milisekund
- Integrátor rychlosti posuvu a polohové odchylky zvyšuje přesnost sledování
- Vysoké rozlišení programovacích jednotek
  - Poloha: -536 870 912 ... +536 870 911 uživatelských jednotek
  - Rychlost: 1 ... 8 388 607 uživatelských jednotek/sec
  - Zrychlení: 1 … 1 073 741 823 uživatelských jednotek/sec/sec



Snadné použití

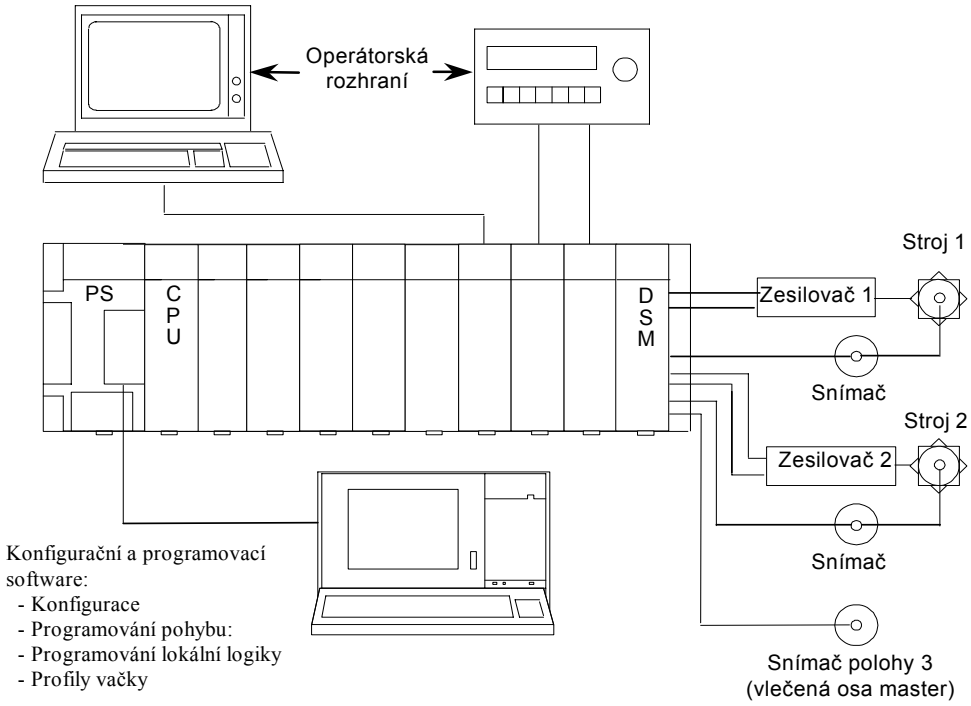
- Jednoduchá a výkonná instrukční sada Motion Program
- Jednoduché programy pohybu pro 1 až 4 osy. Programy pro více os používající osu 1 a 2 mohou využít synchronizovaného startu bloku.
- Energeticky nezávislé uložení 10 programů a 40 podprogramů vytvořených pomocí programovacího softwaru.
- Kompatibilní s CPU Series 90-30 vybavenými firmwarem verze 10.0 nebo pozdější (nepracuje s CPU 311 – 341 a 351). Podrobnosti k CPU najdete v Manuálu pro instalaci a hardware PLC Series 90-30 GFK-0356P nebo pozdější.
- Jeden bod připojení pro všechny úlohy programování a konfigurování včetně vytváření programu pohybu (Programy pohybu 1 - 10) a programování lokální logiky. Veškeré programování a konfigurace se načítá přes komunikační port programování PLC. CPU pak načte konfiguraci, programy pohybu a programy lokální logiky do DSM314 přes základní desku PLC.
- Uživatelská změna měřítka programovacích jednotek (uživatelské jednotky) ve standardním režimu a vlečném režimu.
- DSM314 firmware uložený v paměti Flash se aktualizuje přes COMM port na předním panelu. Sady pro aktualizaci firmwaru obsahují firmware a software Loader na disketě. Firmware je také k dispozici pro stažení na webových stránkách GE Fanuc (http://www.gefanuc.com/support).
- Programování receptů pomocí povelových parametrů jako operandy pro Povely zrychlení, rychlosti, pohybu a prodlevy.
- Automatický přenos dat mezi PLC tabulkami a DSM314 bez uživatelského programování
- Jednoduché propojení I/O pomocí prefabrikovaných kabelů a svorkovnic.
- Funkce elektronické vačky počínaje firmwarem verze 2.0

Univerzální I/O

- Řízení digitálních serv GE Fanuc α Series a β Series, serv SL-Series nebo serv jiných výrobců s rozhraním analogového rychlostního povelu nebo analogového povelu krouticího momentu.
- Vstupy spínačů výchozí polohy a přejetí pro každou servoosu
- Dva vzorkovací vstupy zachycení polohy pro každou osu mohou zachytit polohu osy a/nebo master polohu s přesností +/-2 jednotky plus rozptyl 10 mikrosekund.
- 5 V, 24 V a analogové I/O pro použití v PLC
- Inkrementální vstup snímače polohy s fázovým posuvem na každé ose pro režim snímače polohy/analogový režim
- Vstup snímače polohy s fázovým posuvem pro master osu vlečené osy
- 13-bitový analogový výstup je možno řídit pomocí PLC nebo použít jako monitor ladění digitálního serva
- Vysokorychlostní digitální výstup (čtyři 24 V a čtyři 5 V) přes interní řízení lokální logiky

Část 1: Přehled pohybového systému

DSM314 je inteligentní, plně programovatelný přídavný modul řízení pohybu pro programovatelný automat (PLC) Series 90-30. DSM314 umožňuje uživateli PLC kombinovat vysoce výkonné řízení a funkce lokální logiky s logikou PLC řešící funkce v jednom integrovaném systému. Následující obrázek znázorňuje hardware a software používaný pro nastavení a provozování servosystému. Tato část krátce popisuje jednotlivé součásti systému s cílem poskytnout celkový jeho přehled činnosti.



Obrázek 1-1. Hardware a software používaný ke konfigurování, programování a provozu servosystému DSM314

PLC Series 90-30 a DSM314

DSM314 a PLC Series 90-30 pracují společně jako jeden integrovaný soubor pro řízení pohybu. DSM314 komunikuje s PLC přes rozhraní základní desky. Každý cyklus PLC se data, například *Zadaná rychlost* a *Okamžitá poloha* v DSM314 přenesou do PLC v datech %I a %AI. Každý cyklus PLC se také přenesou data %Q a %AQ z PLC do DSM314. Data %Q a %AQ se používají pro řízení DSM314. Bity %Q vykonávají funkce jako spuštění pohybu, zrušení pohybu a vynulování příznaků vzorkování. Povely %AQ vykonávají funkce jako inicializace polohy a načtení registrů parametrů.

Kromě použití adres %I, %AI, %Q a %AQ ještě existuje další možnost poslání parametrů z PLC do DSM314 pomocí instrukce žebříkového programu COMM_REQ. Podrobnosti o použití instrukce COMM_REQ s DSM najdete v Dodatku B, *Instrukce COMM_REQ DSM314.*

Latence dat PLC a latence DSM314

DSM314 je inteligentní modul pracující asynchronně s modulem CPU Series 90-30. Výměna dat mezi CPU a DSM314 probíhá automaticky. Více informací o činnosti cyklus Series 90-30 najdete v *Referenční příručce instrukční sady pro CPU PLC Series 90-30/90/Micro*, GFK-0467. Následující informace uvádějí úvahy o časování používaném v modulu DSM314.

Přenos dat z PLC do DSM

- Funkce na bázi PLC mohou asynchronně načítat stavové informace DSM (%I a %AI) z datové paměti DSM. DSM interně obnovuje všechna stavová data kromě Okamžité rychlosti rychlostí polohové smyčky (každých 0.5 až 2 ms). Okamžitá rychlost se v datové paměti DSM aktualizuje každých 128 milisekund. DSM vytváří průměr a tím generuje přesný údaj Okamžité rychlosti; proto údaj Okamžité rychlosti není určený pro účely řízení vysokou rychlostí.
- Pokud DSM je umístěno v sestavě CPU, PLC vyžaduje při čtení dat (%I a %AI) z a zápisu dat (%Q a %AQ) do interní paměti DSM přibližně další 2-4 milisekundy na interní komunikaci. PLC normálně čte data z a zapisuje výstupní data do DSM jednou během cyklu PLC. V nejhorším případě se aktualizace interních dat DSM (která trvá 0.5 až 2 ms) vykoná hned po aktualizaci čtení vstupu PLC. V takovém případě PLC nenačte DSM data znovu až do dalšího čtení a veškeré změny dat DSM budou připravené pro PLC buď o 4-6 ms později nebo přibližně o jeden cyklus PLC později, podle toho, která hodnota je větší.
- Konfigurační software automaticky zvolí délku dat %AI a %AQ podle počtu nakonfigurovaných os. CPU PLC vyžaduje čas na čtení a zápis dat v interní komunikaci s DSM314. Referenční příručka instrukční sady PLC Series 90-30, GFK-0467 verze M nebo pozdější, uvádí vliv cyklu PLC podle skupiny modelu CPU, když se zvolí různé konfigurace os. Viz také list s Důležitými informacemi k produktu, který se dodává zabalený s modulem DSM.
- Povely PLC do DSM (%Q, %AQ) se přenesou na výstup DSM na konci cyklu PLC řešícího logiku. DSM zpracuje povely během 4 milisekund po příchodu.

Program pohybu / CTL vstupy na čelní desce

- Prodlevy související s řízením programu pohybu nebo větvením přes vstupy CTL na čelní desce jsou shodné s intervaly aktualizace polohové smyčky (0.5 až 2 ms) plus prodleva vstupního filtru (typicky 5 ms při 24-voltových CTL vstupech nebo 10 µs při 5-voltových CTL vstupech). Časy aktualizace polohové smyčky viz tabulky 1-1, 1-2 a 1-3.

Lokální logika

- Prodlevy související s aktualizacemi dat lokální logiky vycházejí z intervalů aktualizace polohové smyčky (viz následující odstavec "Doby aktualizace servosmyčky DSM314") a nesouvisí se čtením PLC. Proto programy lokální logiky je možno použít rychle změnou interních dat DSM, které nemůže použít CPU PLC z důvodu času přenosu dat z PLC do DSM a delší doby čtení PLC.

Doby aktualizace servosmyčky DSM314

DSM314 při řízení digitálního střídavého serva GE Fanuc používá časy aktualizace smyčky uvedené v Tabulka 1-1.

Tabulka 1-1. Časy aktualizace digitální servosmyčky GE Fanuc

Proudová/momentová smyčka motoru:	250 mikrosekund
Rychlostní smyčka motoru:	1 milisekunda
Polohová smyčka motoru:	2 milisekundy

DSM314 při řízení analogového serva **bez** lokální logiky používá časy aktualizace smyčky uvedené v Tabulka 1-2.

Tabulka 1-2. Časy aktualizace analogové servosmyčky bez lokální logiky

Polohová smyčka s 1 osou bez lokální logiky:	0.5 milisekundy
Polohová smyčka se 2 osami bez lokální logiky:	1 milisekunda
Polohová smyčka se 3-4 osami bez lokální logiky:	2 milisekundy

DSM314 při řízení analogového serva **s** lokální logikou používá následující časy aktualizace smyčky uvedené v Tabulka 1-3. Časy aktualizace smyčky s lokální logikou jsou delší, protože časový interval osy č. 4 se používá k výpočtu funkce lokální logiky.

Tabulka 1-3. Časy aktualizace analogové servosmyčky s lokální logikou

Polohová smyčka s 1 osou s lokální logikou:	1 milisekunda
Polohová smyčka s 2-3 osami s lokální logikou:	2 milisekundy

Režim analogového krouticího momentu kromě výše uvedených polohových regulátorů zahrnuje regulátor rychlosti. Pro osu v režimu analogového krouticího momentu se regulátor rychlosti spustí každých 0,5 milisekundy.

Polohové vzorkovací impulsy DSM314

Každý konektor osy na čelní desce DSM314 má dva vstupy pro Polohové vzorkovací impulsy. Puls s náběžnou hranou na vstupu Strobe způsobí, že se načte *Okamžitá poloha* osy. Rozlišení pro načtení polohy je +/- 2 pulsy s dalšími 10 mikrosekundami rozptylu na prodlevu vstupního filtru vzorkovacích impulsů. Skutečná zobrazená odchylka závisí na zrychlení serva a filtraci/vzorkování vstupu vzorkovacích impulsů. Přesné vztahy používané pro výpočet přesnosti vzorkování najdete v Dodatku G.

Vzorkovací data se aktualizují během jednoho intervalu aktualizace polohové smyčky (0.5 - 2 ms) v odpovídajícím %AI datovém registru *Strobe Position*. Data *Strobe Position* jsou také uložena v registru parametrů DSM, který je možno použít jako operand pro povely programu pohybu PMOVE a CMOVE a v lokální logice. Aktualizace dat *Strobe Position* do PLC závisí na délce cyklu PLC a může trvat déle než 2 ms.

V digitálním režimu tyto vzorkovací impulsy jsou 5 V jednobodové/diferenciální vstupy (IN1-IN2).

V analogovém režimu tyto vzorkovací impulsy jsou pouze 5 V jednobodové (IO5-IO6). Pokud nejsou fyzicky připojené k nějakému zařízení, jsou tyto vzorkovací impulsy přitažené na jedničku v pouze analogovém režimu (jak je vidět ve stavových bitech PLC %I Strobe).

Podíl času na čtení DSM314

Následující tabulka uvádí, kolik milisekund DSM314 přidá k době čtení PLC. Velikost podílu času čtení se vztahuje k (1) počtu nakonfigurovaných os v DSM314 a (2) typu sestavy (hlavní, expanzní nebo vzdálená), ve které je modul DSM314 namontovaný.

Počet nakonfigurovaných os	Podíl času na čtení DSM314 (v milisekundách)		
	Hlavní sestava	Expanzní sestava	Vzdálená sestava
1	1.6	2.6	6.9
2	2.2	3.8	9.9
3	2.8	4.3	13.0
4	3.3	5.2	15.9

Poznámky:

1. Nezapomeňte na to, že interní nástroj lokální logiky DSM314 má maximální dobu čtení 2 ms, které nezávisí na čtení PLC. To umožňuje uživateli pružně řídit pohybové úlohy kritické na čas pomocí programu lokální logiky. Podrobnosti k programování lokální logiky viz příslušné kapitoly v tomto manuálu.

2. U aplikací, kde výše uvedené zvýšení doby čtení má vliv na činnost stroje, může být k přenosu potřebných dat do a z DSM314 nutné použít funkce "Suspend I/O", "DOIO" a "SNAP". Tyto funkce umožní zabránit přenosu všech dat %I, %Q, %AI, %AQ při každém čtení, pokud to nevyžadujete tak často, což může snížit velikost podílu na času čtení.

Software

DSM314 vyžaduje následující konfigurační/programovací softwarové balíky:

- CIMPLICITY Machine Edition Logic Developer – PLC verze 2.1 nebo pozdější
- VersaPro verze 1.1 nebo pozdější.

Programovací/konfigurační softwarový balík se používá pro následující úlohy. Informace vytvořené těmito úlohami se posílají do DSM314 přes interní komunikaci PLC při každém zapnutí PLC.

- Konfigurace. Umožňuje uživateli zvolit nastavení a výchozí provozní parametry modulu.
- Vytvoření pohybového programu. Je přípustných až 10 pohybových programů a 40 podprogramů.
- Vytvoření programu lokální logiky. Program lokální logiky běží synchronně s pohybovým programem, ale je nezávislý na čtení CPU PLC. To umožňuje, aby DSM314 rychle reagoval na I/O signály pohybu na konektorech čelní desky. Tento interní čas odezvy na I/O signály pohybu je mnohem rychlejší než by bylo možné, když by logika pro tyto signály byla ovládána hlavním žebříkovým programem běžícím v PLC. Je to v důsledku (1) prodlevy při předávání signálů po propojovací rovině a (2) delší doby cyklu PLC.
- Vytvoření profilu vačky (CAM). Profil vačky určuje odezvu serva vlečené osy na index master polohy. Profily vačky jsou v příslušném pohybovém programu vyvolávané podle názvu.

  Poznámka: Editor CAM je plně integrovaný do Logic Developeru – PLC. VersaPro verze 1.5 nebo pozdější s přídavným softwarem editoru CAM je nutný pro programování funkce elektronické CAM.

Rozhraní obsluhy

Rozhraní obsluhy dávají obsluze způsob jak řídit a monitorovat servosystém pomocí řídicího panelu nebo CRT obrazovky. Tato rozhraní komunikují s PLC přes diskrétní I/O moduly nebo přes inteligentní moduly sériové nebo síťové komunikace.

Data obsluhy se automaticky přenášejí mezi PLC a DSM314 přes adresy %I, %AI, %Q a %AQ, které jsou zadané při konfiguraci modulu. Tento automatický přenos dat představuje flexibilní a jednoduché rozhraní s několika rozhraními obsluhy, které mohou být rozhraním s PLC Series 90-30.

Rozhraní servopohonu a stroje

Rozhraní servopohonu a stroje je tvořeno 36-pinovým konektorem pro každou osu. Toto rozhraní přenáší signály, které řídí polohu osy, například signály s pulsně-šířkovou modulací (PWM) do zesilovače, zpětnovazební signály digitálního sériového snímače polohy nebo analogový povel serva a zpětnou vazbu snímače polohy s fázovým posuvem. Také jsou zajištěné vstupy *Spínače výchozí polohy* a *Přejezdu osy* i univerzální vstupy a výstupy PLC.

Standardní kabely, které se připojují k uživatelské svorkovnici montované na lištu DIN nebo na panel, zjednodušují zapojování a je možno je zakoupit u GE Fanuc. Svorkovnice mají šroubové svorky pro polní zapojení modulu DSM314. Více informací ohledně kabelů a svorkovnic používaných s modulem DSM314 najdete v kapitole 3.

Část 2: Přehled činnosti DSM314

Každá osa DSM314 může být provozovaná s povolenou nebo zakázanou smyčkou řízení vlečené osy:

Standardní režim (konfigurace *řídicí smyčky vlečené osy* = zakázána)

- V digitálním standardním režimu modul umožňuje řízení polohy, rychlosti a krouticího momentu v uzavřené smyčce až pro dva servomotory α nebo β Series GE Fanuc v ose 1 a ose 2. Osu 3 je možno použít jako servoosu analogového rychlostního povelu rozhraní nebo jako pomocnou master osu.
- V analogovém standardním režimu modul umožňuje řízení polohy v uzavřené smyčce až pro čtyři servomotory. V závislosti na konfiguraci os DSM také umožňuje řízení v rychlostní smyčce pro režim analogového krouticího momentu. Když se DSM bude používat se servy s analogovým rychlostním rozhraním, řídicí a momentová řídicí smyčka se uzavře v servozesilovači, zatímco DSM uzavře polohovou smyčku. Když se DSM bude používat se servy s analogovým momentovým rozhraním, momentová řídicí smyčka se uzavře v servozesilovači, zatímco DSM uzavře polohovou a rychlostní smyčku.
- V případě digitálních a analogových aplikací je možno uživatelské programovací jednotky upravit nastavením poměru konfiguračních parametrů Uživatelské jednotky a Počet. Povely Jog, Pohyb rychlostí a Vykonat pohybový program umožňují používání standardního režimu v širokém rozsahu aplikací.

Vlečný režim (konfigurace *řídicí smyčky vlečené osy* = povolená)

- V režimu digitální vlečené osy modul umožňuje řízení polohy, rychlosti a krouticího momentu v uzavřené smyčce až pro dva servomotory α nebo β Series GE Fanuc v ose 1 a ose 2. Osu 3 je možno použít jako servoosu povelu analogového rychlostního rozhraní nebo jako pomocnou master osu.
- V režimu analogové vlečené osy modul umožňuje řízení polohy v uzavřené smyčce až pro čtyři servomotory (jednu nebo dvě ze čtyř možných os je možno místo toho použít jako pomocnou master osu). Kromě toho v závislosti na konfiguraci os modul umožňuje řízení v rychlostní smyčce pro režim analogového krouticího momentu. Když se DSM bude používat se servy s analogovým rychlostním rozhraním, řídicí a momentová řídicí smyčka se uzavře v servozesilovači, zatímco DSM uzavře polohovou smyčku. Když se DSM bude používat se servy s analogovým momentovým rozhraním, momentová řídicí smyčka se uzavře v servozesilovači, zatímco DSM uzavře polohovou a rychlostní smyčku.
- V případě digitálních a analogových aplikací modul umožňuje stejné funkce jako standardní režim včetně nastavitelného poměru Uživatelských jednotek a Počet.
- Kromě toho je možno nakonfigurovat vstup polohy master osy. Každá vlečená osa sleduje vstup master osy v naprogramovaném poměru (A:B). Pohyb vyvolaný povely Jog, Pohyb rychlostí a Vykonání pohybového programu je možno zkombinovat s pohybem vlečené osy vygenerovaným master osou.
- Možnosti vlečené osy jsou následující:
  - Zdroj master osy konfigurovatelný jako okamžitá nebo zadaná poloha od jiné osy
  - Bit %Q Master Source Select přepíná mezi dvěma předem nakonfigurovanými zdroji master osy
  - Rampa zrychlování umožňuje hladké zrychlení podřízené osy až do synchronizace polohy a rychlosti s master osou
  - Samostatné spouštěcí zdroje pro povolení a zakázání vlečené osy

Všimněte si, že režim Winder v DSM314 není podporovaný. Je podporovaný v DSM302.

Činnost standardního režimu

Obrázek 1-2 je zjednodušený diagram polohové smyčky standardního režimu. Interní generátor povelů pohybu vytváří zadanou polohu a zadanou rychlost pro polohovou smyčku. Polohová smyčka odečte okamžitou polohu (polohová zpětná vazba) od zadané polohy a vytvoří tak polohovou odchylku. Hodnota polohové odchylky se násobí konstantou zisku polohové smyčky a vznikne tak rychlostní povel serva. Aby se snížila polohová odchylka během pohybu serva, zadaná rychlost z generátoru povelů se sečte jako hodnota rychlostní zpětné vazby na rychlostní povelový výstup serva.

V datech, které DSM314 posílá do PLC, jsou následující položky:

Zadaná rychlost – okamžitá rychlost generovaná interním generátorem dráhy DSM314.

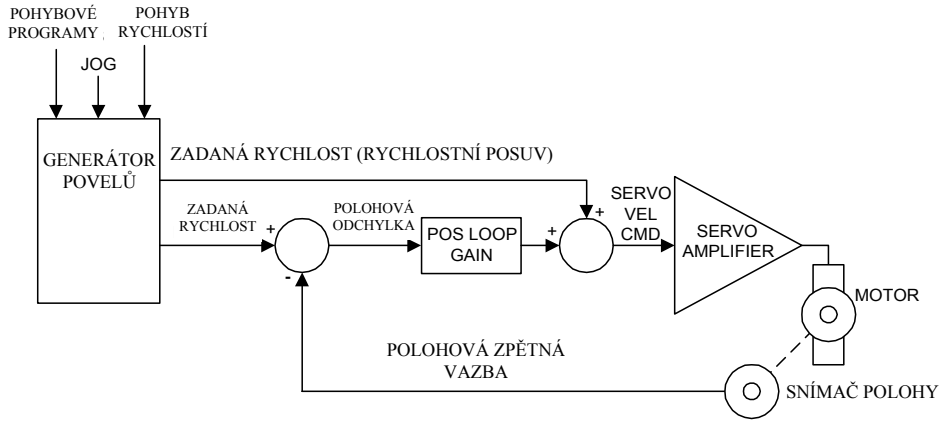
Zadaná poloha – okamžitá poloha generovaná interním generátorem dráhy DSM314.

Okamžitá rychlost – rychlost osy indikovaná zpětnou vazbou.

Okamžitá poloha – poloha osy indikovaná zpětnou vazbou.

Polohová odchylka – rozdíl mezi *Zadanou polohou* a *Okamžitou polohou*.

DSM314 umožňuje naprogramovat ***časovou konstantu polohové odchylky*** (v jednotkách 0.1 ms) a ***rychlostní zpětnou vazbu*** (v jednotkách 0.01 procent). ***Časová konstanta polohové smyčky*** nastaví zisk polohové smyčky a určí rychlost odezvy uzavřené polohové smyčky. Procento ***rychlostní zpětné vazby*** určuje velikost *zadané rychlosti*, která se připočítá k rychlostnímu povelu serva.



Obrázek 1-2. Zjednodušená polohová smyčka standardního režimu s rychlostní zpětnou vazbou (analogové rychlostní rozhraní)

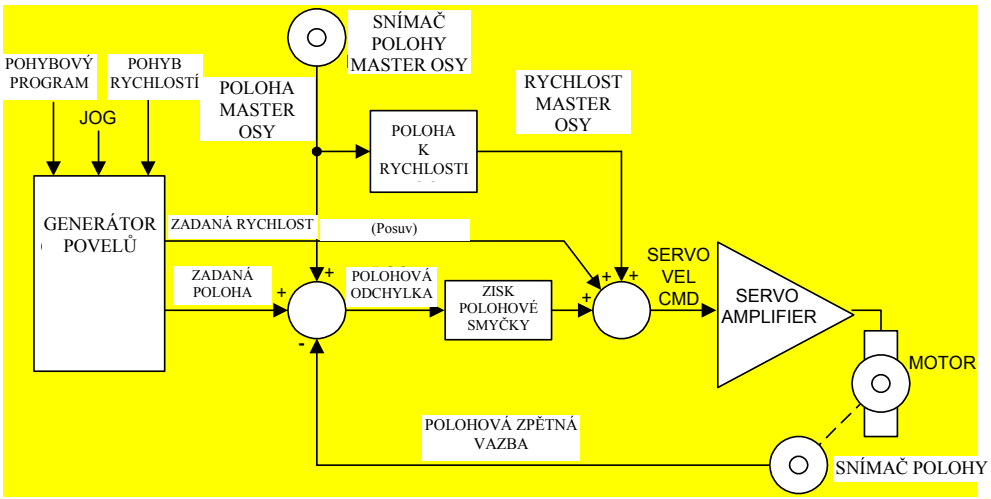
Činnost v režimu vlečené osy

Obrázek 1-3 je zjednodušený diagram polohové smyčky režimu ***vlečené osy***. Je podobný ***standardnímu*** režimu polohové smyčky (viz předchozí stránka) s přidáním vstupu master osy. Vstup master osy je přídavný povelový zdroj, který vytváří polohu master osy a rychlost master osy. Poloha master osy se sčítá se *zadanou polohou* z generátoru povelů osy. Rychlost master osy se sčítá s výstupem *zadané rychlosti* (rychlostní zpětná vazba) generátoru povelů osy. **Proto poloha a rychlost servomotoru jsou určené součtem výstupu generátoru povelů a vstupu master osy.** Generátor povelů a vstup master osy mohou při vytváření pohybu servoosy pracovat současně nebo nezávisle.

DSM314 umožňuje pro vstup master osy několik zdrojů:

- Zadaná poloha osy 1
- Okamžitá poloha osy 1 (snímač polohy osy 1)
- Zadaná poloha osy 2
- Okamžitá poloha osy 2 (snímač polohy osy 2)
- Zadaná poloha osy 3
- Okamžitá poloha osy 3 (snímač polohy osy 3)
- Zadaná poloha osy 4
- Okamžitá poloha osy 4 (snímač polohy osy 4)

Poměr, kterým servoosa sleduje master osu, je programovatelný jako poměr dvou celých čísel. Například servoosa může být naprogramovaná tak, aby se posunula o 125 jednotek polohové zpětné vazby na každých 25 polohových jednotek master osy. Při každé změně polohy master osy o 1 polohovou jednotku se servoosa posune o (125 / 25) = 5 jednotek polohové zpětné vazby.



Obrázek 1-3. Zjednodušená polohová smyčka režimu vlečené osy se vstupem master osy (analogové rychlostní rozhraní)

Část 3: Serva α Series (digitální režim)

Mezi vlastnosti digitálního serva α Series GE Fanuc (vyslovované "Alfa") patří:

- Světová špičková spolehlivost
- Malé nároky na údržbu, žádný *drift* součástek, žádné komutátorové kartáče
- Všechny parametry se nastavují digitálně, není nutné žádné další ladění
- Absolutní snímač polohy eliminuje nájezd do výchozí polohy (vyžaduje přídavnou baterii)
- Možnost přídavné brzdy motoru
- Pro většinu motorů je k dispozici přídavné krytí pro prostředí IP67
- Vysoké rozlišení 64K jednotek na otáčku zpětnovazebního snímače polohy (inkrementální nebo absolutní)

Servomotory GE Fanuc ověřené na více než třech milionech osách nainstalovaných na celém světě nabízejí nejvyšší spolehlivost a výkon. Nejnovější technologie, jako například vysokorychlostní sériové snímače polohy a vysoce účinné integrované výkonové moduly (IPM), dále zvyšují výhody zákazníků.

Servosystém GE Fanuc je jedinečný v tom, že všechny řídicí smyčky – proudové, rychlostní a polohové – jsou uzavřené v pohybovém kontroléru. Tento přístup snižuje dobu nastavování a dává významné výhody výrobní kapacity i v nejnáročnějších aplikacích.

Servopohony jsou levnější na integraci a údržbu. Teplotní změny nemají na řídicí obvody vliv. Neexistují žádné *zákaznické* moduly. Serva mají široký rozsah aplikací, to znamená široký rozsah setrvačnosti zátěže, pružnost při zrychlování/zpomalování a konfiguraci zpětné vazby, atd.

Pro optimalizaci výkonu a k překonání omezení stroje existují rozsáhlé funkce pro úpravy. Servozesilovače na bázi IPM vyžadují o 60% méně prostoru na panelu než konvenční spínané zesilovače a vytvářejí o 30% méně tepla.

Integrovaný digitální zesilovač α Series (SVU)

Integrovaný servozesilovač α Series (SVU) obsahuje zesilovač s integrovaným napájecím zdrojem v samostatné jednotce. Tato jednotka má stejné fyzické rozměry a plochu jako předchozí servozesilovače "C" Series GE Fanuc.

Integrované zesilovače SVU α Series používají stejné připojení jako zesilovače "C" Series s výjimkou toho, že obvod nouzového zastavení používá interní zdroj 24 V a tak už zde není požadavek na napájecí zdroj 100 V.

Teplo se odvádí do montážní části SVU přes panel mimo kryt.

Protože SVU zesilovače α nemají funkci rekuperace do sítě, můžou být nutné vybíjecí odpory. Ty se dodávají v několika velikostech.

Servozesilovač typu SVU α Series se dodávají v 5 velikostech se špičkovým proudem od 12 do 130 A. (Poznámka: GE Fanuc NA v současné době nabízí pouze modely 80 A a 130 A.)

Kabely pro připojení SVU zesilovačů k DSM314 a k motorům se dodávají v různých délkách.

V publikaci GFH-001, *Průvodce specifikacemi výrobků serva* najdete více informací o servech α Series.

Servomotory α Series

Řada servomotorů α Series zahrnuje vylepšený návrh k zajištění nejlepšího možného výkonu. Jsou k dispozici jmenovité hodnoty až do 56 Nm. Tyto motory jsou až o 15% kratší a lehčí než předchozí servomotory *S* Series. Nová izolace vinutí a celkový povlak tmelem pomáhají chránit motor před vlivem prostředí.

S motorem se dodává standardní snímač polohy 64K absolutních jednotek. Přídržné brzdy (90 V ss) a krytí IP67 jsou volitelné. Servomotory α Series vyhovují mezinárodním normám pro CE (EMC a nízké napětí), IEC a UL/CUL. Následující tabulka uvádí příklad použitelných motorů α Series (jsou k dispozici také některé αL, αC, αHV a αM).

Více informací najdete v kapitole 4 tohoto manuálu , "Konfigurace DSM314" v části s nadpisem "Typ motorů". Viz také následující publikace:

- GFH-001, *Průvodce specifikací výrobků serva*
- GFZ-65142E, *α Series Manuál popisu střídavého motoru*

Tabulka 1-1. Zvolené modely servomotoru α Series

Číslo α modelu	Krouticí moment Nm	Výkon KW	Maximální rychlost (ot./min)
α1	1	0.3	3000
α2	2	0.4	2000
α2	2	0.5	3000
α3	3	0.9	3000
α6	6	1.0	2000
α6	6	1.4	3000
α12	12	2.1	2000
α12	12	2.8	3000
α22	22	3.8	2000
α22	22	4.4	3000
α30	30	3.3	1200
α30	30	4.5	2000
α30	30	4.8	3000
α40	38	5.9	2000
α40/Fan	56	7.3	2000

Část 4: Serva β Series (digitální režim)

Mezi vlastnosti digitálního serva β Series GE Fanuc (vyslovované "Beta") patří:

- Světová špičková spolehlivost
- Malé nároky na údržbu, žádný *drift* součástek, žádné komutátorové kartáče
- Všechny parametry se nastavují digitálně, není nutné žádné další ladění
- Absolutní snímač polohy eliminuje nájezd do výchozí polohy (vyžaduje přídavnou baterii)
- Přídavná brzda motoru
- Vysoké rozlišení (32K – Beta) (64K – Beta M) jednotek na otáčku snímače polohy

Serva GE Fanuc β Series nabízejí nejvyšší spolehlivost a výkon. Nejnovější technologie, jako například vysokorychlostní sériové snímače polohy a vysoce účinné integrované výkonové moduly, dále zvyšují výkon servosystému. Protože servopohony β Series jsou navržené s ohledem na trh řízení pohybu, jsou ideálně vhodné pro průmysl balicí techniky, manipulace s materiálem, a zpracování kovů.

Servosystém GE Fanuc je jedinečný v tom, že všechny řídicí smyčky – proudové, rychlostní a polohové – jsou uzavřené v pohybovém kontroléru. Tento přístup snižuje dobu nastavování a dává významné výhody výrobní kapacity i v nejnáročnějších aplikacích.

Servopohony jsou levnější na integraci a údržbu. Teplotní změny nemají na řídicí obvody vliv. Neexistují žádné *zákaznické* moduly. Serva mají široký rozsah aplikací včetně širokého rozsahu setrvačnosti zátěže, pružnost při zrychlování/zpomalování a konfiguraci zpětné vazby, atd. Pro optimalizaci výkonu a k překonání omezení stroje existují rozsáhlé softwarové funkce pro úpravy.

Digitální zesilovače β Series

Servozesilovače β Series obsahují napájecí zdroj se spínacím obvodem. Proto GE Fanuc je schopný dodat kompaktní zesilovač, který je o 60% menší než konvenční modely. Ve skutečnosti zesilovač β Series má stejnou výšku a hloubku jako GE Fanuc modul PLC Series 90-30. To umožňuje při používání kontroléru pohybu DSM314 efektivní uspořádání panelu.

Zesilovač je navržený tak, aby vyhovoval mezinárodním normám.

GE Fanuc nabízí tři komunikační rozhraní pro zesilovače β Series: s pulsně-šířkovou modulací (PWM), Sériová sběrnice serva Fanuc (FSSB) a rozhraní I/O Link. S modulem DSM314 je možno použít pouze rozhraní s pulsně-šířkovou modulací (PWM). Rozhraní PWM používá standardní komunikační protokol serva GE Fanuc. Polohová zpětná vazba se přenáší sériově mezi kontrolérem DSM a sériovým snímačem polohy namontovaným na motoru.

Servomotory β Series

Servomotory β Series jsou postavené na vynikající technologii serv α Series. Zahrnují několik inovací návrhu, které dávají nejlepší kombinace vysokého výkonu, nízkých nákladů a kompaktních rozměrů. Jsou k dispozici jmenovité hodnoty 0,5 až 12 Nm.

Tyto motory jsou až o 15% kratší a lehčí než kompaktní serva. Nová izolace vinutí a celkový povlak tmelem pomáhají chránit motor před vlivem prostředí.

Motory β Series vyhovují mezinárodním normám (IEC). Motory mají ochranu úrovně IP65 (IP67 se dodává na zvláštní objednávku).

Absolutní snímač polohy (32K – Beta) (64 K – Beta M) se dodává standardně s každým servem β Series. S každým modelem je možno dodat přídavnou přídržnou brzdu 90 V ss.

Více informací najdete v kapitole 4 tohoto manuálu , "Konfigurace DSM314" v části s nadpisem "Typ motorů". Viz také následující publikace:

- GFH-001, *Průvodce specifikací výrobků serva*
- GFZ-65232E, *β Series Manuál popisu střídavého motoru*

Tabulka 1-2. Zvolené modely servomotoru β Series

Číslo β modelu	Krouticí moment[1] Nm	Výkon KW	Maximální rychlost ot./min
β0.5	0.5	0.2	3000
βM0.5	0.65	0.2	5000
β1	1	0.3	3000
βM1	1.2	0.4	5000
β2	2	0.5	3000
β3	3	0.5	3000
β6	6	0.9	2000
αC12	12	1.4	2000

[1] Označuje souvislý 100% činitel využití

Poznámka: Motor αC12 je uvedený s motory β z důvodu podobných vlastností a řady zesilovačů

Část 5: Serva SL Series (analogový rychlostní režim)

DSM314 podporuje všechny modely serv SL Series GE Fanuc. Podrobnosti o servozesilovačích SL Series, motorech a příslušenství najdete v *Uživatelském manuálu serva SL Series,* GFK-1581.

Kapitola 2 Krátký úvod

Tato kapitola se skládá ze sedmi částí, které uvádějí přehled nastavení systému Motion Mate DSM314 a ověření připojení a funkcí systému.

Tato kapitola slouží jako úvod pro ty, kteří nejsou seznámeni s pohybovým systémem Motion Mate DSM314. Vykonáním následujících kroků, které jsou zde popsané, byste měli být schopni během krátké doby ovládat svůj pohybový kontrolér a servo nebo vykonat funkci Jog. Vyžaduje se minimální úroveň obeznámenosti s některým z následujících konfiguračních/programovacích softwarových balíků. :

- CIMPLICITY Machine Edition Logic Developer – PLC verze 2.1 nebo pozdější
- VersaPro verze 1.1 nebo pozdější.

Výstraha

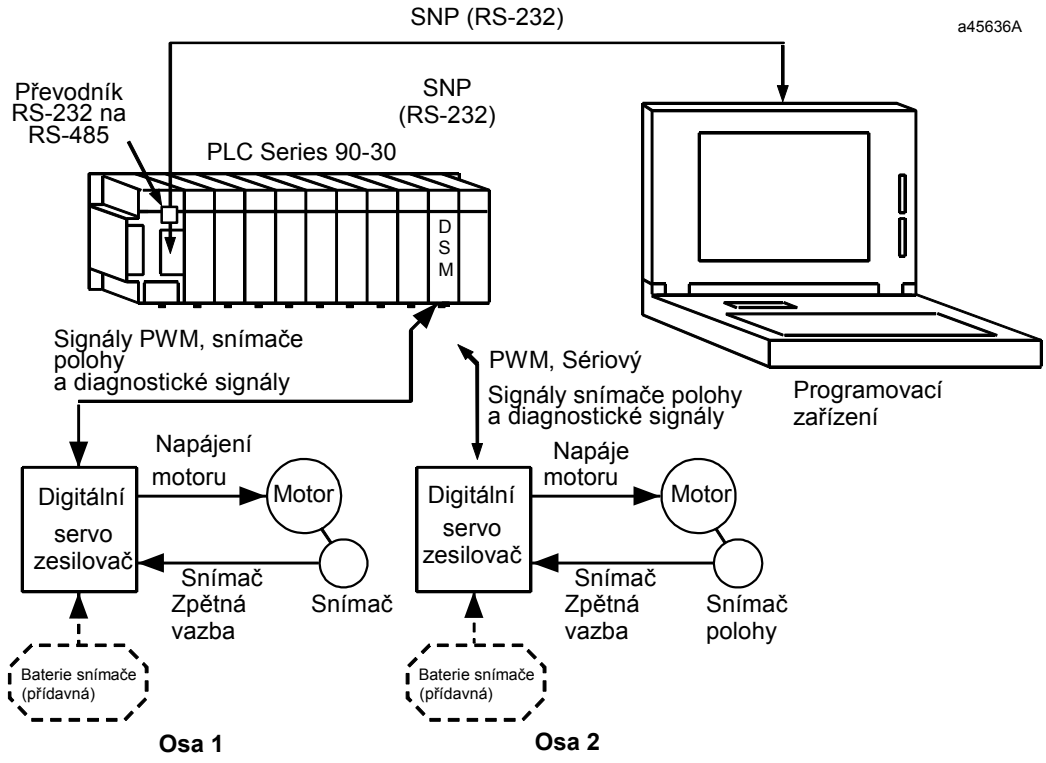
Nespojujte hřídel motoru s mechanickými zařízeními pro účely tohoto výkladu. Servomotor musí být pevně připojený ke stabilnímu povrchu.

Typický pohybový systém DSM314 obsahuje kontrolér pohybu DSM314, programovatelný automat Series 90-30 (PLC), motory, servozesilovače, I/O a rozhraní člověk-stroj (HMI).

Řídicí systém DSM314 se skládá ze dvou částí: řízení serva a řízení stroje.

Řízení serva převádí pohybové povely na signály, které se posílají do servozesilovače. Také v něm běží programy lokální logiky a pohybu. Servozesilovač přijímá řídicí signály z řízení serva a zesiluje je na požadovanou úroveň výkonu motoru. DSM314 zajišťuje řízení serva.

Řízení stroje (PLC Series 90-30) obsahuje modul DSM314 a I/O moduly. Řízení stroje vykonává uživatelem definovanou logiku řízení (ale ne lokální logiku). Řízení stroje (PLC) a řízení serva (DSM314) si vyměňují data přes propojovací rovinu Series 90-30.



Obrázek 2-1. Typický digitální systém řízení pohybu Motion Mate DSM314 pro dvě osy

Část 1: Vybalení systému

DSM314, digitální servozesilovače a motory jsou zabalené samostatně. Tato část popisuje, jak vybalit hardware a provést předběžnou kontrolu komponentů.

Vybalení DSM314

Opatrně vybalte systémové komponenty DSM314 a PLC. Přesvědčte se, že jste dostali všechny položky uvedené na seznamu materiálu. Uložte veškerou dokumentaci a průvodní doklady, které jste dostali s pohybovým systémem DSM314.

Vybalení digitálního servozesilovače

Pro použití s DSM314 se dodávají dva soubory digitálního zesilovače a podsystému serva, α Series nebo β Series.

Digitální servozesilovač se dodává ve dvouvrstvé krabici. Po odstranění horní vrstvy balicího materiálu se odkryje zesilovač. Pak opatrně vyndejte vnitřní krabici z vnějšího obalu. Pak vytáhněte zesilovač z vnitřní krabice. Ponechejte si volné díly nebo výplňový materiál zabalený se zesilovačem. Vizuálně zkontrolujte zesilovač, jestli během přepravy nedošlo k jeho poškození.

Poznámka: Nesnažte se v tomto okamžiku měnit nastavení zkratovacích propojek nebo přepínačů na zesilovači.

Vybalení motoru FANUC

Motory FANUC jsou balené dvěma různými způsoby v závislosti na jejich velikosti. Největší motory se odesílají na dřevěných paletách a jsou zakryté lepenkou. Většina motorů je však zabalená v kartónových krabicích.

1. Pokyny pro vybalení:
   - V případě motorů FANUC balených v krabicích otevírejte krabice shora. Motory jsou zabalené ve dvou kusech tvarovaného materiálu. Opatrně vytáhněte horní díl z krabice. Tím vznikne dostatečné místo pro vyndání motoru.
   - Pokud motor bude připevněný k paletě, vytáhněte lepenkový kryt. Tím vznikne přístup ke šroubům, které drží motor na paletě. Vytáhněte šrouby a uvolněte motor s palety.
2. Zkontrolujte, jestli motor není poškozený.
3. Přesvědčte se, že hřídel motoru je možno otáčet rukou.

   Poznámka: Pokud byl motor objednaný s přídavnou přídržnou brzdou, hřídel se nebude otáčet, dokud brzda nebude pod napětím.

Další krok. . . . . ***Sestavení systému Motion Mate DSM314***

Část 2: Sestavení systému Motion Mate DSM314

Všeobecné směrnice

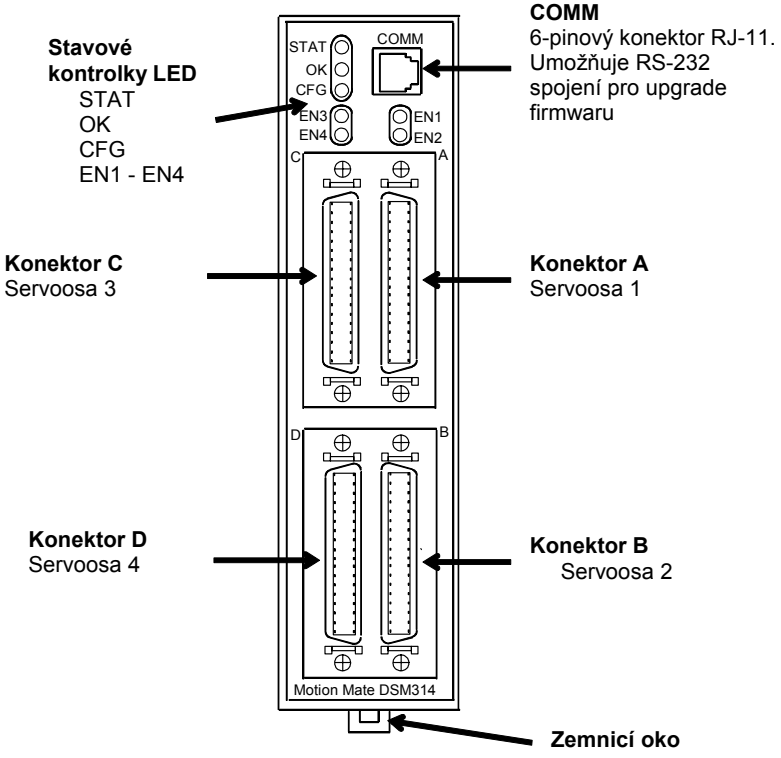
- Vždy se přesvědčte, že konektory lze zasunout do zásuvek. Konektory jsou provedené tak, že je možno je zastrčit pouze jedním směrem. Nesnažte se je zasunovat násilím.
- Nepřehlédněte důležitost řádného uzemnění systémových komponentů DSM314 **včetně zemnícího vodiče stínění čelní desky DSM314**. Informace k uzemnění jsou uvedené v této části.

Všechna uživatelská připojení kromě zemnícího oka jsou umístěna na přední straně modulu DSM314. Zemnící oko se nachází na spodku modulu. Viz následující obrázek.

Instrukce o instalaci DSM314, když je nutno dodržovat IEC a další normy, najdete v *Požadavcích na instalaci v souladu s normami*, GFK-1179.

Připojení Motion Mate DSM314

Obrázek 2-2 uvádí přehled čelní desky a štítků na modulu DSM314. Další informace a úplné schéma připojení najdete v kapitole 3, *Instalace a zapojení Motion Mate DSM314*.



Obrázek 2-2. Připojení na čelní desce systému pro řízení pohybu Motion Mate DSM314

Připojení digitálního servozesilovače SVU α Series

Pokud budete připojovat zesilovač β Series, přejděte na následující část.

Digitální servozesilovač α Series během prvního spuštění nebo po výměně součástí nevyžaduje žádné seřizování. Seřizování také není nutné při změně podmínek prostředí.

Chcete-li připojit digitální servozesilovač α Series, postupujte podle následujících kroků.

1. Připojte servozesilovač α Series k DSM314.

A. Než budete připojovat povelový kabel serva, přesvědčte se, že je připojený zemnicí vodič stínění čelní desky DSM314. Tento vodič se dodává s modulem DSM314 a musí být připojený ze 1/4 palcové nožové svorky ve spodní části modulu ke vhodnému zemnícímu bodu panelu.

B. Kabel povelů serva obsahuje výstupní signál pulsně-šířkové modulace (PWM) z DSM, sériová data ze snímače polohy motoru a diagnostické signály ze zesilovače. Signály přenášené tímto kabelem mají napěťovou úroveň datové komunikace a je nutno je vést mimo vnější vodiče, zejména vodiče s vysokým napětím.

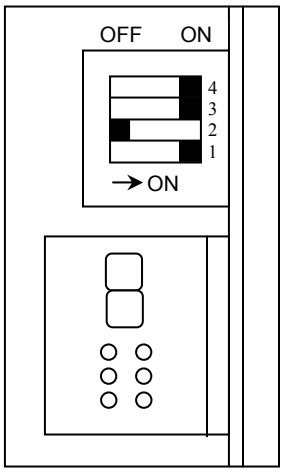
C. Opatřete si kabel povelů serva **IC800CBL001** (1 metr) nebo **IC800CBL002** (3 metry). Zastrčte protilehlý konec tohoto kabelu do konektoru **JS1B**, který je ve spodní části servozesilovače (viz obrázek 2-4).

D. Pokud nebudete používat svorkovnici osy IC693ACC335 k přerušení uživatelských I/O, například vstupy mezí přejetí nebo nájezdu do výchozí polohy, druhý konec kabelu zastrčte do konektoru označeného **A** pro osu 1 nebo **B** pro osu 2 v přední části DSM314. Pokud budete používat svorkovnici, zastrčte druhý konec kabelu do konektoru svorkovnice označeného **SERVO**. Dále si opatřete připojovací kabel se svorkovnicí **IC693CBL324** (1 metr) nebo **IC693CBL325** (3 metry). Jeden konec tohoto kabelu zastrčte do konektoru svorkovnice s označením **DSM**. Druhý konec kabelu zastrčte do konektoru s označením **A** v případě serva pro osu 1 nebo **B** v případě serva pro osu 2 v přední části modulu DSM314.

Poznámka: Informace týkající se uživatelských I/O použitelných pro připojení svorkovnice IC693ACC335 najdete v “Spoje I/O” v kapitole 3.

Přepínače kanálů zesilovače SVU

Přesvědčte se, že přepínače kanálů (přepínače DIP) umístěné za dvířkami zesilovače SVU jsou nastavené podle následujících tabulek. Všimněte si, že poloha **OFF** je doleva a poloha **ON** je doprava. Všimněte si také, že přepínače jsou číslované zdola nahoru (přepínač 1 je spodní přepínač). Například přepínače 1, 3 a 4 na obrázku 2-3 jsou zobrazené v poloze ON a přepínač 2 je zobrazený v poloze OFF.



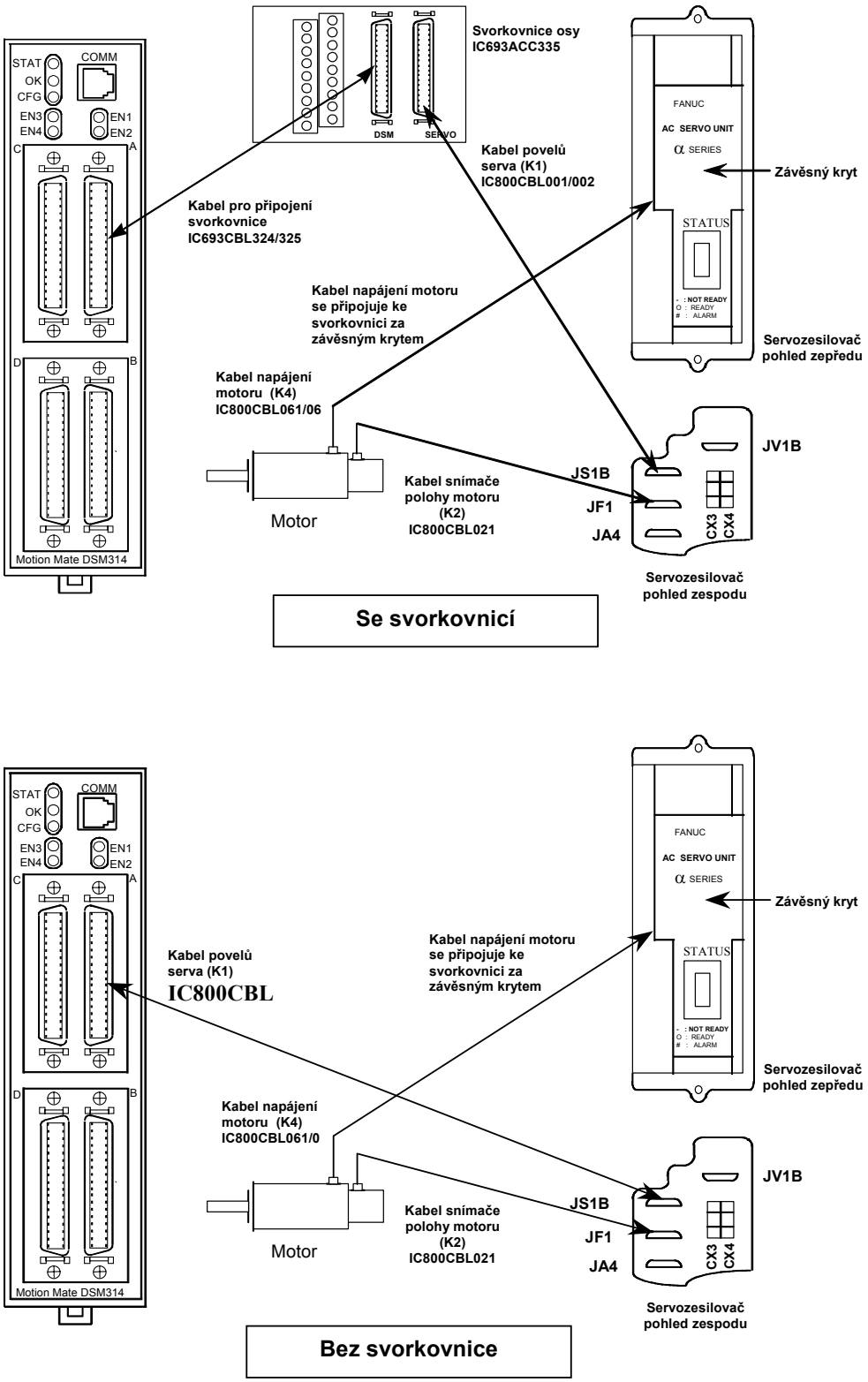
Obrázek 2-3. Přepínače kanálů zesilovače SVU

Tabulka 2-1. Nastavení přepínačů kanálů zesilovače SVU

Zesilovač SVU1-80				
Regenerační vybíjecí jednotka	**SW1**	**SW2**	**SW3**	**SW4**
Vestavěná (100 W)	ON	OFF	ON	ON
Samostatná A06B-6089-H500 (200 W)	ON	OFF	ON	OFF
Samostatná A06B-6089-H713 (800 W)	ON	OFF	OFF	OFF

Zesilovač SVU1-130				
Regenerační vybíjecí jednotka	**SW1**	**SW2**	**SW3**	**SW4**
Vestavěná (400 W)	ON	OFF	ON	ON
Samostatná A06B-6089-H711 (800 W)	ON	OFF	ON	OFF

(Chcete-li připojit další zesilovač, pro každý další zesilovač zopakujte kroky B, C a D výše.)



Obrázek 2-4. Připojení digitálního servozesilovače SVU α Series k Motion Mate DSM314

2. Připojte kabel napájení motoru k digitálnímu servozesilovači α Series.

A. Velikost motoru objednaného pro váš systém určuje kabel pro napájení motoru K4, který použijete, když pro svůj systém objednáte předem zkompletované kabely. Motory v následující tabulce jsou seskupené tak, aby bylo možno použít jeden ze zkompletovaných kabelů, které dodává GE Fanuc. Toto není úplný seznam kabelů pro napájení servomotorů α Series, obsahuje ale nejčastěji používané typy. Úplný seznam najdete v *Průvodci specifikacemi serva*, GFH-001.

Tabulka 2-2. Příklady čísla zkompletovaných kabelů pro napájení servomotoru (K4) α Series

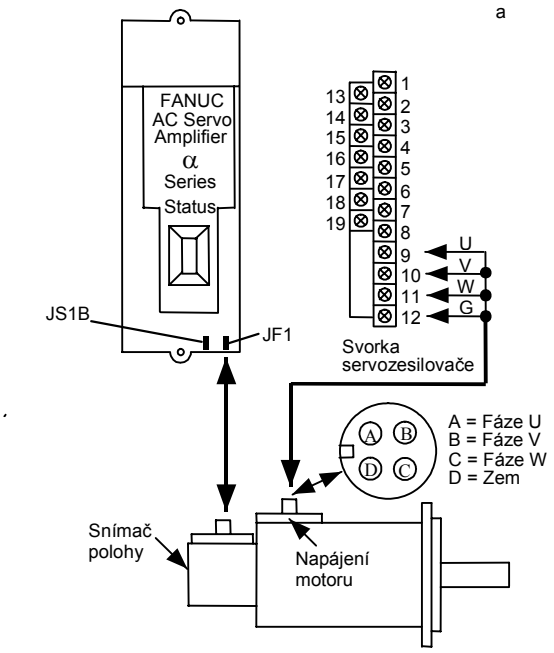
Typ motoru	Katalogové číslo výkonového kabelu	Popis kabelu	Délka kabelu
α3/3000 α6/3000	IC800CBL061	Konektor MS s kolenem	14 metrů
α12/3000 α22/2000 α30/1200	IC800CBL062	Konektor MS s kolenem	14 metrů
α30/3000 α40/2000	IC800CBL063	Konektor MS s kolenem	14 metrů

B. Jeden konec tohoto kabelu má čtyři dráty s označením U, V, W a GND, které se připojují ke šroubovým svorkám 9-12 na servozesilovači. Tyto čtyři dráty připojte ke svorkovnicovému pásku podle obrázku 2-5.

C. Po odstranění plastových víček pro ochranu konektoru motoru připojte druhý konec kabelu k motoru. Všimněte si, že tento kabel je klíčovaný a je možno ho připojit pouze do jediného připojovacího místa motoru.

(Tento postup zopakujte podle potřeby pro další osy v systému.)

Nejnovější informace o kabelech pro napájení motorů nebo zapojení vlastních kabelů pro napájení motorů najdete v nejnovější verzi *Manuálu popisu servomotoru α Series*, GFZ-65142E.



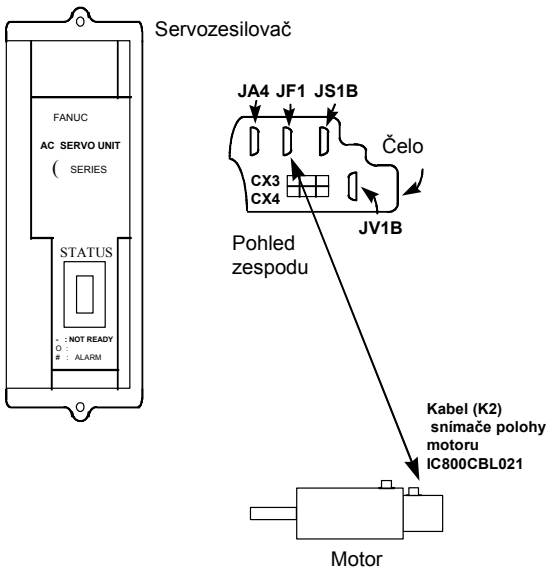
Obrázek 2-5. Připojení motoru ke svorkovnicovému pásku servozesilovače α Series

3. Připojte snímač polohy motoru k digitálnímu servozesilovači α Series.

A. Odstraňte plastová ochranná víčka z konektoru snímače polohy na motoru a zastrčte kabel zpětné vazby K2 **IC800CBL021**. Kabel je nakonfigurovaný tak, že ho lze připojit pouze k jedinému místu na motoru.

B. Opačný konec zastrčte do přípojky s označením **JF1** ve spodní části servozesilovače α Series (viz obrázek 2-6).

Tento postup zopakujte pro všechny osy v systému.



Obrázek 2-6. Připojení snímače polohy motoru α Series

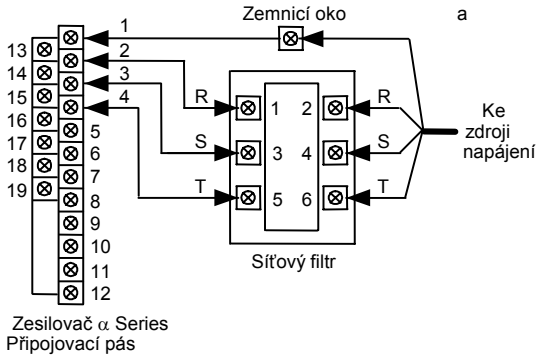
Tabulka 2-3. Zkompletovaný kabel snímače polohy servomotoru (K2) α Series pro modely α3 až α40

Modely motoru	Výkonový kabel	Délka kabelu
α3 až α40	IC800CBL021	14 metrů

Poznámka: Podrobnosti o kabelech α Series můžete najít v Manuálu popisu střídavého servomotoru α Series, GFZ-65142E, a v Průvodci specifikacemi výrobků α a β Series**,** GFH-001.

4. Připojte napájení 220 V, 3 fáze, k digitálnímu zesilovači α Series

Střídavý síťový filtr sníží vliv harmonického šumu na napájení; jeho použití se doporučuje. Na jeden síťový střídavý filtr je možno připojit dva nebo více zesilovačů, pokud se nepřekročí jeho kapacita. Na obrázku 2-6 je ukázáno připojení zesilovače k síťovému filtru.



Poznámka: Vyžaduje se napájení 220 V, tři fáze.

Obrázek 2-7. Připojení servozesilovače k síťovému filtru a napájecímu zdroji

Poznámka: Kabel pro obě propojení mezi síťovým filtrem a servozesilovačem a mezi síťovým filtrem a napájecím zdrojem si musíte opatřit sami. Mezi síťový filtr a servozesilovač použijte 4-žilový kabel 600 V, 60°C (140°F) se schválením UL nebo CSA.

Průřez drátu používaného pro připojení filtru k napájecímu zdroji musí být dimenzovaný podle jističe mezi napájecím zdrojem a síťovým filtrem a podle počtu serv připojených k síťovému filtru.

Pokud se pro přívod napájení zesilovače použije samostatný oddělovací transformátor, síťový filtr není nutný.

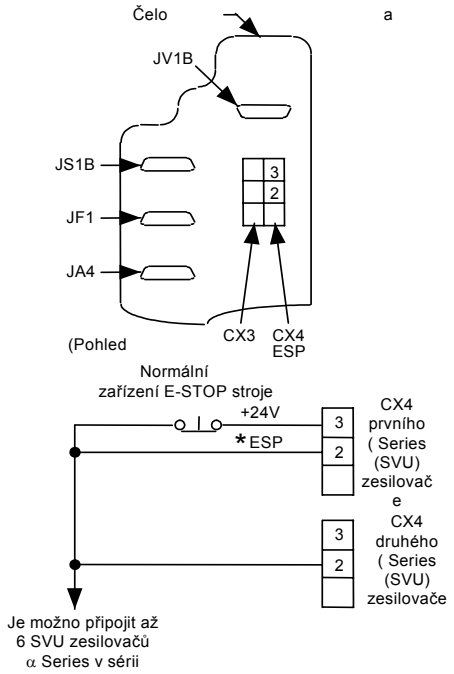
5. Připojte nouzové zastavení stroje k digitálnímu servozesilovači α Series

Pin 3 konektoru **CX4 umístěný ve spodní části zesilovače α Series (SVU)** přivádí napájení +24 V ss pro obvod E-STOP. Toto napětí veďte skrz obvod E-STOP stroje tak, aby napětí +24 V ss bylo na pinu 2, když **nebude** aktivován E-STOP. Pokud se používá spínač E-STOP, toto spojení se **musí** provést pomocí zkratovací propojky.

Poznámka: Kabel pro toto spojení si musíte opatřit sami. Klíčované konektorové zásuvky označené jako konektor X a piny konektoru svorkovnice jsou součástí balení zesilovače. Aby tento zesilovač pracoval, musíte toto spojení nainstalovat jako spínač nebo zkratovací propojku.

Upozornění

Nepřivádějte na tento konektor žádné externí napětí.



Obrázek 2-8. Připojení nouzového zastavení k servozesilovači α Series

Více informací najdete v *Manuálu popisu servozesilovače α Series (SVU)*, GFZ-65192EN.

Připojení digitálního servozesilovače SVU β Series

Digitální servozesilovač β Series neobsahuje žádná uživatelská nastavení. Chcete-li připojit servozesilovač β Series, postupujte podle následujících kroků. ***Zesilovače α Series viz předchozí část.***

1. Připojte digitální servozesilovač β Series k DSM314

A. Než budete připojovat povelový kabel serva, přesvědčte se, že je připojený zemnicí vodič stínění čelní desky DSM314. Tento vodič se dodává s modulem DSM314 a musí být připojený ze 1/4 palcové nožové svorky ve spodní části modulu ke vhodnému zemnícímu bodu panelu.

B. Kabel povelů serva obsahuje výstupní signál pulsně-šířkové modulace (PWM) z kontroléru pohybu, sériová data ze snímače polohy motoru a diagnostické signály ze zesilovače. Signály přenášené tímto kabelem mají napěťovou úroveň datové komunikace a je nutno je vést mimo vnější vodiče s vysokým napětím.

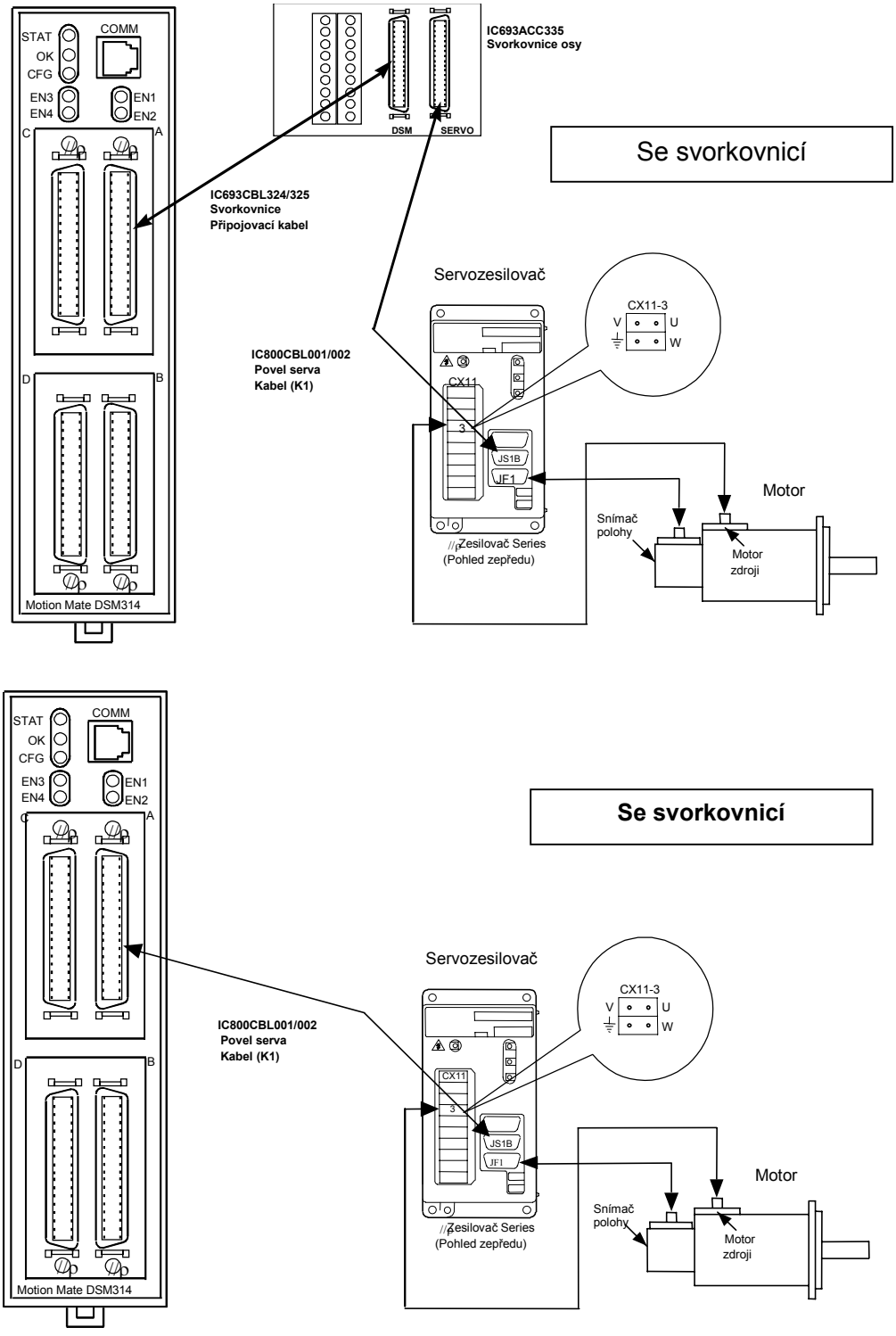
C. Opatřete si kabel povelů serva **IC800CBL001** (1 metr) nebo **IC800CBL002** (3 metry). Zastrčte protilehlý konec tohoto kabelu do konektoru **JS1B**, který je v přední části servozesilovače (viz obrázek 2-9).

D. Tento krok závisí na tom, jestli používáte nebo nepoužíváte svorkovnici:

- **Pokud nebudete používat svorkovnici osy IC693ACC335** k přerušení uživatelských I/O, například vstupy mezí přejetí nebo nájezdu do výchozí polohy, druhý konec kabelu zastrčte do konektoru označeného **A** pro servoosu 1 nebo **B** pro servoosu 2 v přední části DSM314.

- **Pokud budete používat svorkovnici osy IC693ACC335**, zastrčte druhý konec kabelu do konektoru svorkovnice označeného **SERVO**. Dále si opatřete kabel povelů serva **IC693CBL324** (1 metr) nebo **IC693CBL325** (3 metry). Jeden konec tohoto kabelu zastrčte do konektoru svorkovnice s označením **DSM**. Druhý konec kabelu zastrčte do konektoru s označením **A** v případě serva pro osu 1 nebo **B** v případě serva pro osu 2 v přední části DSM314.

(Chcete-li připojit další zesilovač, pro každý další zesilovač zopakujte kroky B - D výše.)



Obrázek 2-9. Připojení servozesilovače β Series

Více informací najdete v části o připojení v *Průvodci specifikacemi výrobků serva*, GFH-001.

Připojte kabel napájení motoru (K4) k digitálnímu servozesilovači β Series.

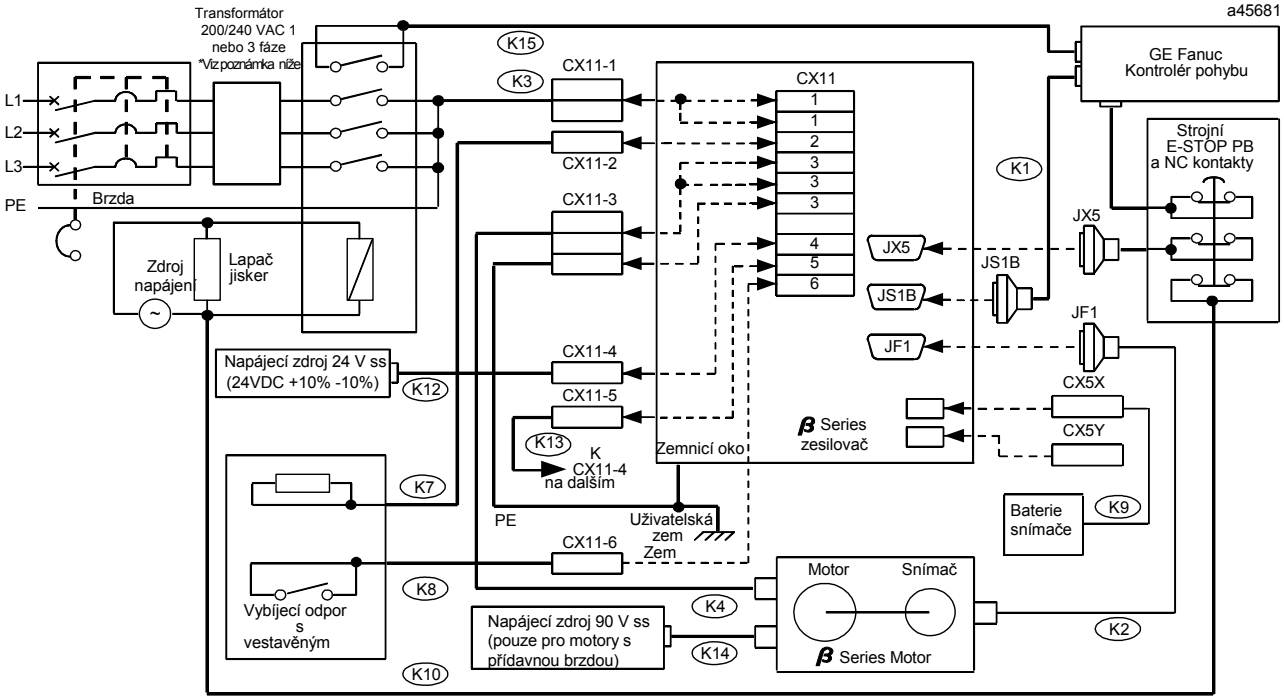
Upozornění

Připojení ke konektoru CX-11 proveďte opatrně. Tento konektor není klíčovaný. Před přivedením napájení připojení radši dvakrát zkontrolujte. Nesprávné zapojení může vést k chybné funkci nebo poškození zařízení. Pozdější verze zesilovače než revize G mají konektory klíčované.

A. Velikost motoru objednaného pro váš systém určuje kabel pro napájení motoru (K4), který musíte použít. Můžete se rozhodnout koupit již hotový kabel nebo si vyrobit vlastní kabel. Informace o vlastních kabelech nebo instalaci pro vyhovění značce CE najdete v *Manuálu popisu řízení motoru β Series*, GFZ-65232EN. Strana připraveného kabelu pro napájení motoru na straně zesilovače je přizpůsobená k připojení do svorkovnice **CX11-3** na zesilovači.

Tabulka 2-4. Kabel K4 – Příklady kabelu motoru β Series

Typ servomotoru	Číslo kabelu motoru K-4	Popis kabelu
β 0.5/3000	IC800CBL067	14 metrů
β 1/3000, β 2/3000, β3/3000 a β 6/2000	IC800CBL068	14 metrů
α C12/2000	IC800CBL069	14 metrů
βM 0.5/5000	CP8B-OWPB-0140-AA	14 metrů
βM 1/5000	CP8B-OWPB-0140-AA	14 metrů



***Poznámky:**

1. Pokud ve skříni je napětí 200-240 V stř., místo transformátoru je možno použít síťový filtr a bleskojistku.
2. Pro jednofázový provoz se fáze L3 nepřipojuje. Snížení výstupního proudu viz specifikace servosystému v *Průvodci výrobků serva*, GFH-001.

Obrázek 2-10. Připojení digitálního servozesilovače β Series ke svorkovnicovému pásku

B. Po odstranění plastového víčka pro ochranu konektoru motoru připojte druhý konec kabelu napájení motoru. Všimněte si, že tento kabel je klíčovaný a je možno ho připojit pouze do jediného připojovacího místa motoru.

C. Kabel napájení motoru zakoupený od GE Fanuc je 1 metr dlouhý jednožilový drát s konektorem CX11-3 na jedné straně a kruhovou svorkou na druhé straně. Tento kabel umožňuje připojit uzemnění kostry motoru a je nutno ho vždy připojit. Vlastní kabely musí tento drát vždy obsahovat. Správné připojení k zesilovači viz předchozí schéma zapojení.

(Tento postup zopakujte podle potřeby pro další osu v systému.)

Bližší podrobnosti jsou uvedeny v publikaci *Průvodce specifikacemi výrobků*, (GFH-001.

3. Připojte kabel snímače polohy motoru (K2) k digitálnímu servozesilovači β Series

Velikost motoru objednaného pro váš systém určuje kabel pro napájení motoru K4, který použijete, když pro svůj systém objednáte předem zkompletované kabely. Správné katalogové číslo kabelu snímače polohy najdete v následující tabulce.

A. Odstraňte plastové ochranné víčko z konektoru na motoru a zjistěte kabel snímače polohy K2 (viz tabulka 2-5). Tento kabel má dva různé konektory.

B. Zastrčte stranu kabelu s konektorem typu D do konektoru s označením **JF1** na servozesilovači (viz obrázek 2-9).

C. Druhý konec kabelu je uspořádaný tak, aby bylo možno ho zastrčit pouze do konektoru na snímači polohy motoru (červené víčko).

(Tento postup zopakujte pro všechny osy v systému.)

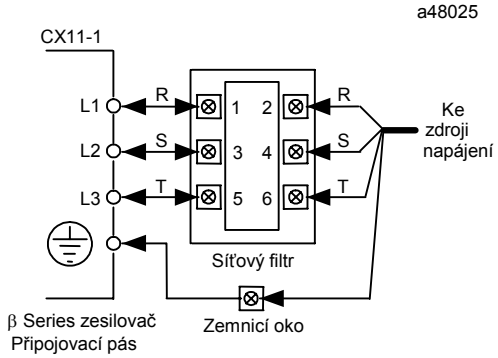
Tabulka 2-5. Kabel K2 – Příklady kabelu snímače polohy β Series

Typ motoru	Číslo kabelu snímače polohy K-2	Popis kabelu
β 0.5/3000	IC800CBL022	14 metrů
β1/3000, β 2/3000, β3/3000 a β6/2000	IC800CBL023	14 metrů
α C12/2000	IC800CBL021	14 metrů
βM 0.5/5000	CFBA-OWPB-0140-AA	14 metrů
βM 1/5000	CFBA-OWPB-0140-AA	14 metrů

4. Připojte kabel napájení 220 V stř. (K3) k digitálnímu zesilovači β Series

Kabel střídavého napájení je kabel dodávaný uživatelem, který se připojuje k **CX11–1** na čele zesilovače β Series. Konektor tohoto kabelu na straně zesilovače je součástí sady A06B-6093-K305 dodávané s každým zesilovačem. Bližší podrobnosti jsou uvedeny v publikaci *Průvodce specifikacemi výrobků serva*, GFH-001.

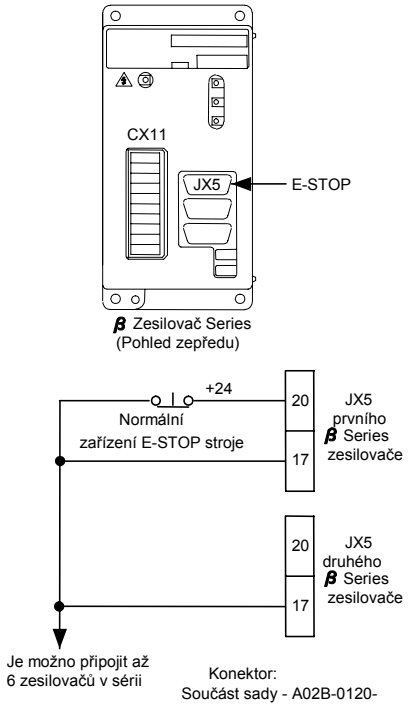
Střídavý síťový filtr sníží vliv harmonického šumu na napájení; jeho použití se doporučuje. Síťový filtr není nutný, pokud se používá oddělovací transformátor nebo samostatný napájecí transformátor. Na jeden síťový střídavý filtr nebo transformátor je možno připojit dva nebo více zesilovačů, pokud se nepřekročí jeho kapacita. Na obrázku 2-11 je ukázáno připojení zesilovače k síťovému filtru.



Obrázek 2-11. Připojení servozesilovače β Series k síťovému filtru a napájecímu zdroji

Poznámka: Propojovací kabel mezi síťovým filtrem a napájecím zdrojem si musíte opatřit sami. Mezi síťový filtr a servozesilovač použijte 4-žilový kabel 600 V, 60°C (140°F) se schválením UL nebo CSA. Průřez drátu používaného pro připojení filtru k napájecímu zdroji musí být dimenzovaný podle velikosti jističe mezi napájecím zdrojem a síťovým filtrem a podle počtu serv připojených k síťovému filtru. Napájecí konektory a svorky se dodávají jako součást balení zesilovače.

5. Připojte nouzové zastavení stroje k digitálnímu servozesilovači β Series



Obrázek 2-12. Připojení E-STOP k servozesilovači β Series

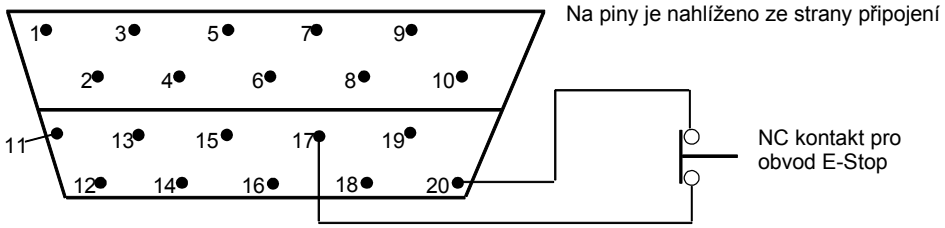
Poznámka: Kabel pro toto spojení si musíte opatřit sami. Konektor JX5 a kryt konektoru jsou součástí zesilovače jako číslo A02B-0120-K301. Pokud se vyžaduje obvod E-STOP, toto spojení se musí provést pomocí zkratovací propojky, jinak zesilovač nebude funkční.

Konektor **JX5 Pin 20** přivádí +24 V ss pro obvod E-STOP. Pin 20 připojte přes normálně sepnutý kontakt nebo spínač tak, aby na **JX5 pin 17** bylo napětí +24V ss, když **není** aktivován E-STOP. **GE Fanuc používá dva typy konektorů pro konektor JX5. Správné připojení ke každému typu viz obrázek 2-13.**

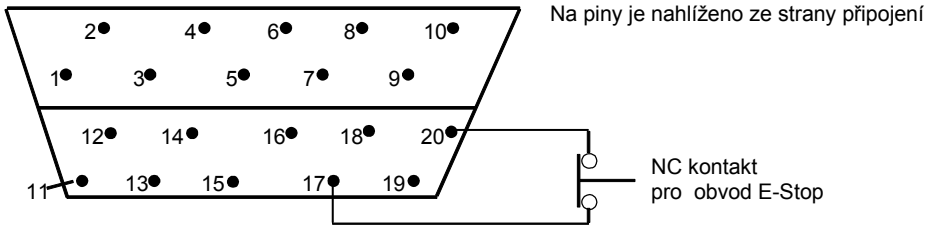
Upozornění

Nepřivádějte na do tohoto připojení žádné externí napětí.

Uspořádání pinů konektoru HIROSE s 20 piny typu PCR



Uspořádání pinů konektoru HONDA s 20 piny typu PCR



Obrázek 2-13. Uspořádání 20-pinového konektoru PCR

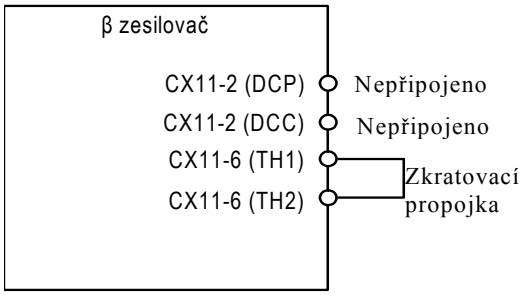
6. Připojte 24 V ss kabel (K12) k digitálnímu servozesilovači β Series

Konektor pro externí napájení 24 V ss je součástí dodávky zesilovače jako součást sady A06B-6093-K305 a je nutno ho připojit k **CX11-4.** Druhý konec kabelu je nutno připojit ke zdroji 24 V ss, který má kapacitu minimálně 450 miliampér na každý zesilovač β Series. Doporučuje se napájecí zdroj GE Fanuc IC690PWR024. **Napájení zatím nepřipojujte**.

7. Připojte kabel K8 – Zkratovací propojka nebo externí regenerační odpor k digitálnímu servozesilovači β Series

Bez externího regeneračního odporu (použití zkratovací propojky)

Pokud nemáte externí regenerační odpor, musíte ponechat spoj na CX11-2 (DCP a DCC) rozpojený. Musíte však zkratovat svorky CX11-6 (TH1 a TH2), jak je znázorněno na obrázku níže. (Tato zkratovací propojka uzavírá obvod, který by jinak byl uzavřený normálně sepnutým spínačem nadměrné teploty v jednotce externího regeneračního odporu.) Pokud tato zkratovací propojka nebude nainstalovaná, zesilovač nebude fungovat. Zkratovací propojka a její konektor jsou součástí konektorové sady A06B-6093-K305, která se dodává s každým zesilovačem.



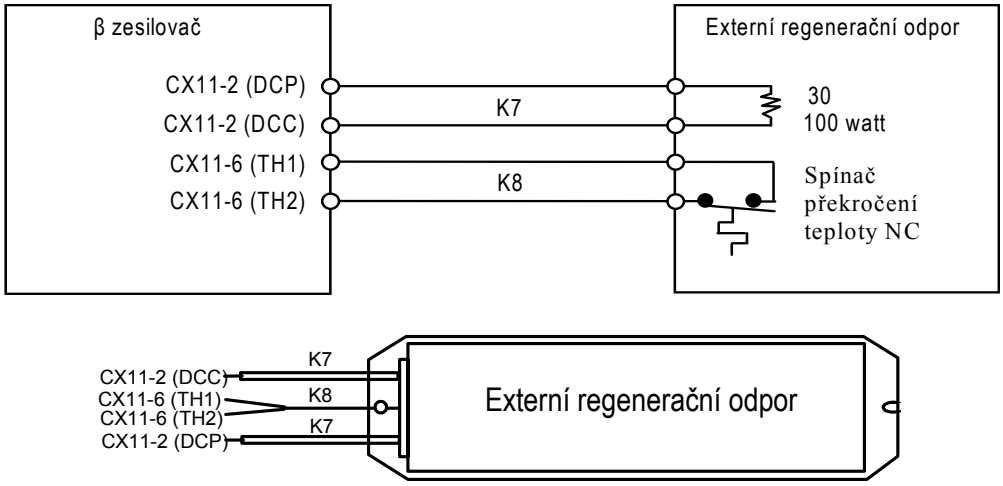
Obrázek 2-14. Instalace zkratovací propojky, když se nepoužívá externí regenerační odpor

S externím regeneračním odporem

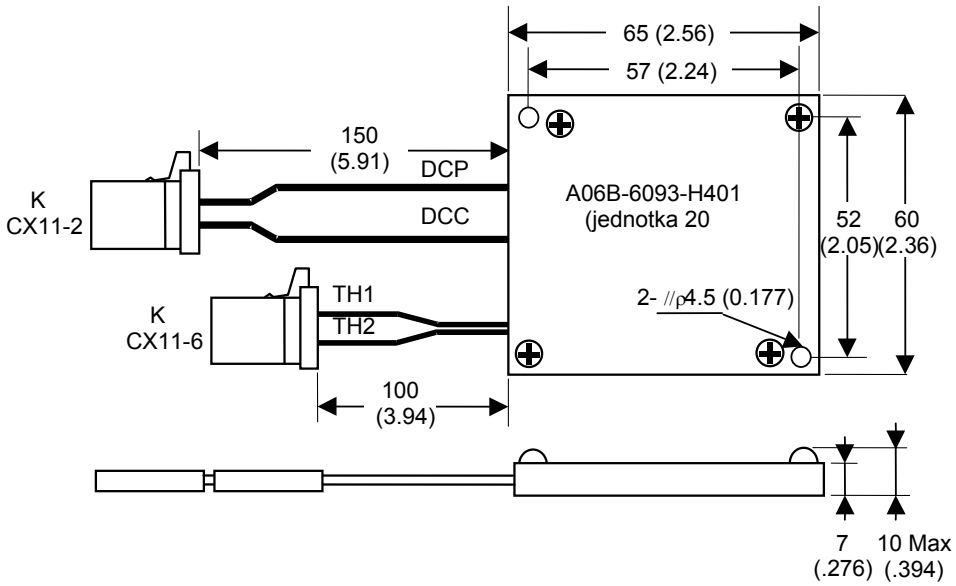
Pokud máte externí regenerační odpor, všimněte si, že má 4 dráty. Dva menší dráty (K8) se připojují k internímu normálně sepnutému teplotnímu spínači odporu. Tento spínač se rozpojí a zkratuje zesilovač, když se odpor příliš ohřeje. Dva větší dráty (K7) se připojují k odporu. Všechny konektory potřebné k připojení této odporové jednotky k zesilovači jsou součástí dodávky zesilovače.

Připojte tyto dva dráty spínače překročení teploty (K8) k CX11-6, svorky TH1 a TH2. (U těchto spojů nezáleží na polaritě.)

Připojte tyto dva dráty odporu (K7) k CX11-2, svorky DCP a DCC. (U těchto spojů nezáleží na polaritě.)



Obrázek 2-15. Připojení externího regeneračního odporu



Obrázek 2-1. Regenerační 20 W odpor A06B-6093-H401

Instalace a zapojení DSM314 pro analogový režim

Důležité poznámky k analogovému servu:

1. Povel rychlosti analogového serva nebo výstup režimu analogového krouticího momentu je jednobodový signál na pinu 6 pomocné svorkovnice. Tento signál je vztažený k 0 V modulu DSM a PLC. Tento signál je nutno připojit ke vstupu povelu rychlosti nebo krouticího momentu servozesilovače.

   Poznámka: Je důležité správně nakonfigurovat DSM buď na režim analogového krouticího momentu nebo na režim analogové rychlosti. Zvolený režim závisí na typu použitého servozesilovače.

2. DSM314 má nízkoproudový výstup (30 mA) polovodičového relé na pinu 15 pomocné svorkovnice pro připojení ke vstupu povolení servozesilovače.

3. DSM314 vyžaduje v analogovém režimu vstup Drive Ready (signál IN_4) na pinu 5 pomocné svorkovnice. **Když je zesilovač připravený řídit servo, tento signál se musí přepnout na 0 V. DSM začne kontrolovat vstup Drive Ready jednu sekundu po sepnutí relé Drive Enable jako odezva na bit %Q Enable Drive. Pokud servozesilovač nemá vhodný výstup, vstup IN_4 do DSM314 je možno připojit na 0 V nebo funkci je možno v konfiguraci modulu zakázat. Podrobnosti najdete v kapitole 4.**

4. V analogovém režimu se používá zpětná vazba snímače polohy s fázovým posunutím. Zapojení snímače polohy je na obrázcích 3-19 až 3-23.

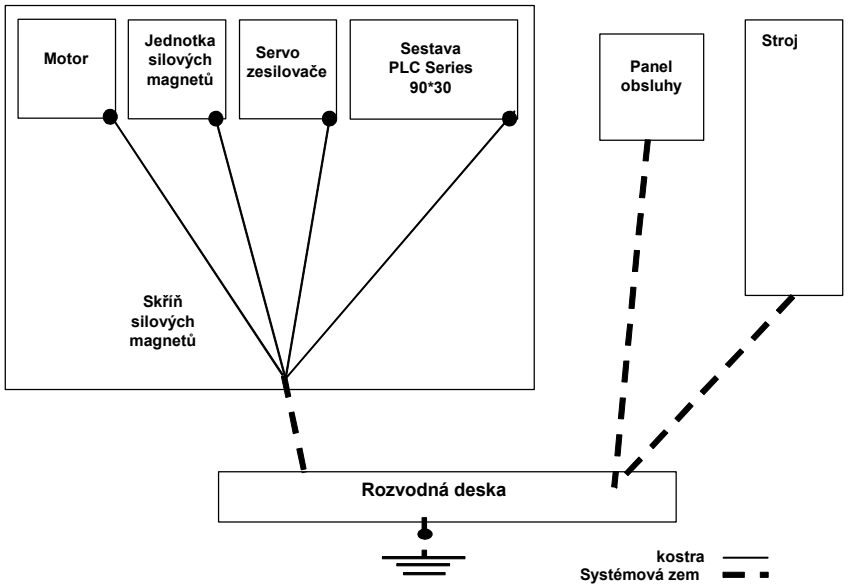
5. Obrázky 3-19 až 3-23 jsou obecná schémata zapojení pro DSM.

6. Podrobnosti o rozhraní mezi DSM314 a GE Fanuc produkty SL Servo najdete v manuálu *Uživatelský manuál SL Series*, GFK-1581.

Uzemnění pohybového systému Motion Mate DSM314

Systém DSM314 musí být řádně uzemněný. Mnoho problémů vzniká právě z důvodu nedodržení těchto pravidel. Při řádném uzemňování systému Motion Mate DSM314 dodržujte následující zásady:

- Zemní odpor systémové země musí být 100 ohmů nebo méně (uzemnění třídy 3).
- Vodič uzemnění stínění čelní desky DSM314 (dodává se s modulem) musí být připojený ze 1/4-palcové nožové svorky ve spodní části modulu na kostru panelu.
- Pokud se používá svorkovnice osy, jsou zde dva stínicí spoje (“S”) a jeden z nich musí být připojený ke kostře panelu.
- Kabel pro uzemnění systému musí mít dostatečně velký průřez, aby mohl spolehlivě přenést případný proud do systémové země, když dojde k poruše, například ke zkratu. Normálně musí mít minimální průřez jako napájecí kabel. Obrázek 2-16 znázorňuje systémy uzemnění.
- Připojení země zesilovače, připojení silové země (PE) a připojení kostry musí být zapojeno vždy, aby byly splněny místní předpisy pro elektrická zapojení. Když budete provádět zapojení pro splnění značky CE, pro svorkovnici ke kabelu zesilovače se vyžaduje zemnící tyč a spony (objednávají se samostatně). Více podrobností najdete v kapitole 3, Instalace a zapojení DSM314, část kabel pro uzemnění I/O.



Obrázek 2-16. Spoje pro uzemnění systému Motion Mate DSM314

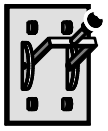
Tabulka2-6. Systémy uzemnění

Uzemnění systému	Popis
Ukostření systému	Ukostření systému se používá z důvodu bezpečnosti a k potlačení externího a interního šumu. Při ukostření systému jsou propojené kostry, skříně jednotek, panely a stínění kabelů rozhraní mezi jednotkami.
Uzemnění systému	Uzemnění systému se používá k připojení kostry systémů spojených mezi zařízeními nebo jednotkami se zemí.

Tím jsou kompletní kroky požadované pro sestavení systému Motion Mate DSM314.

Když budete vytvářet rozhraní se strojem, budete muset provést další práce, ale nyní můžete přejít na další krok. . .

☞ . . . . . ***Zapnutí systému Motion Mate DSM314***



Část 3: Zapnutí systému Motion Mate DSM314

Než zapnete napájení, musíte:

- Zkontrolovat, že napájecí kabely jsou dobře zastrčené do příslušných konektorů.
- Zkontrolovat, že veškeré zapojení napájecích zdrojů je správné.
- Přesvědčit se, že motory jsou dobře zajištěné.
- Zkontrolovat, že všechny komponenty jsou správně uzemněné včetně stínění čelní desky DSM314.
- Pokud budete používat více než jeden motor, přesvědčte se, že zapojení servozesilovače a kabely zpětné vazby nejsou mezi motory křížem.

Pro zapínání napájení řídicího systému DSM314 je specifický postup. **V uvedeném pořadí** proveďte následující kroky:

1. Zapněte napájení 220V stř. digitálních serv. Přesvědčte se, že na zesilovači svítí LED kontrolka *Nabíjení*.
2. U digitálních zesilovačů β Series zapněte zdroj 24 V ss. Přesvědčte se, že svítí kontrolka *Power*.
3. Zapněte napájení PLC Series 90-30 PLC a zkontrolujte, že svítí LED kontrolka PWR na napájecím zdroji Series 90-30.
4. Pomocí správného komunikačního kabelu připojte ruční programovací zařízení (HHP) nebo osobní počítač s konfiguračním/programovacím softwarem k PLC. Více informací najdete v GFK-0356*, Manuál instalace a hardwaru PLC Series 90-30.*
5. Přepněte PLC do režimu **STOP/Zakázáno**. To lze provést pomocí konfiguračního/ programovacího softwaru spuštěného na osobním počítači nebo pomocí tlačítek ručního programovacího zařízení Series 90-30 (režim Run –).
6. Pokud budete používat přídavnou přídržnou brzdu namontovanou na motoru, na vodiče brzdy přiveďte příslušné napájení (90 V ss pro motory α a β Series a 24 V ss pro motory SL Series) k uvolnění přídržné brzdy.

Další krok. . .

☞. . . . . ***Nakonfigurování Motion Mate DSM314***

Část 4: Postup při konfigurování Motion Mate DSM314

Konfigurace kontroléru DSM314 se provádí s použitím konfiguračního softwaru VersaPro (verze 1.1 nebo pozdější) nebo CIMPLICITY Machine Edition Logic Developer-PLC (verze 2.1 nebo pozdější).

DSM314 má rozsáhlý soubor funkcí, které mu umožňují přizpůsobit se mnoha různým aplikacím. Můžete snadno provést nastavení pro svůj systém. Registry parametrů v paměti DSM314 umožňují používat proměnné v pohybových programech DSM314.

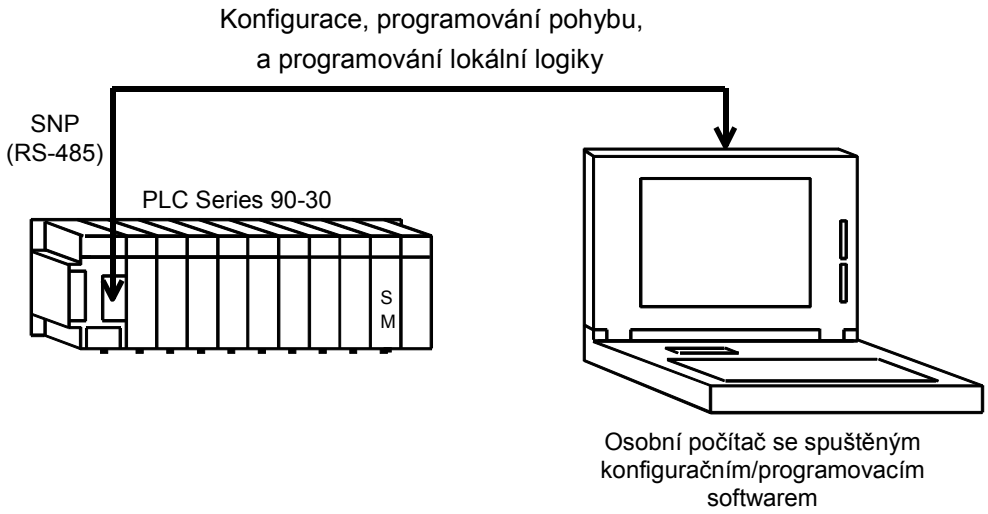
DSM314 obsahuje několik konfiguračních parametrů; pro většinu aplikací je však nutno nastavit pouze některé z nich. Zbývající konfigurační parametry jsou obvykle nastavené na výchozí hodnoty.

Tato část krátce popisuje konfigurační pole DSM314 a jak si prohlížet a nastavit konfigurační parametry pro jogování osy.

Připojení programovacího zařízení k PLC

Veškeré programování DSM314 se provádí přes konfigurační/programovací software poskytující jediný bod programování modulu. Více informací najdete v GFK-0356*, Manuál instalace a hardwaru PLC Series 90-30.* Programovací prostředí má několik možností komunikace. Jedna možnost komunikace je připojit programovací zařízení přímo k SNP portu PLC, jak je znázorněno v následujícím obrázku. Další komunikační metody najdete v dokumentaci k softwaru.

Poznámka: DSM314 má také sériový port na čelní desce modulu. Tento sériový port se používá pouze pro aktualizaci firmwaru DSM314.



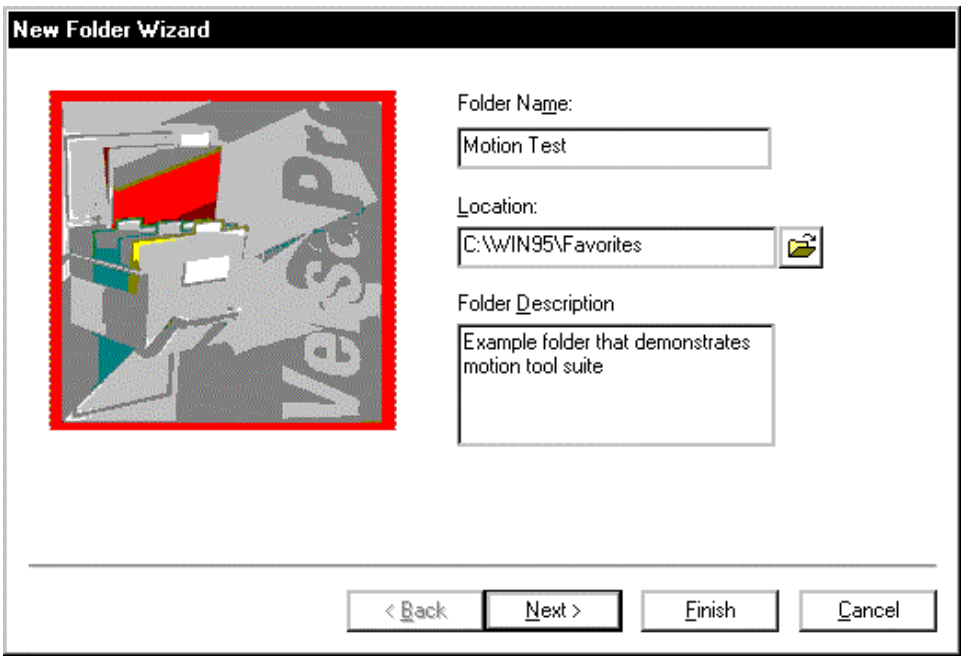
Obrázek 2-17. Schéma zapojení programovacího zařízení DSM

Konfigurace VersaPro

Konfigurace PLC a program logiky a konfigurace DSM314 jsou uložené v adresáři programu VersaPro na vašem pevném disku nebo disketě. Uživatelské prostředí VersaPro je nezávislé prostředí, které umožňuje provádět všechny činnosti potřebné pro nakonfigurování, vytvoření, editování a načtení programů do PLC a DSM.

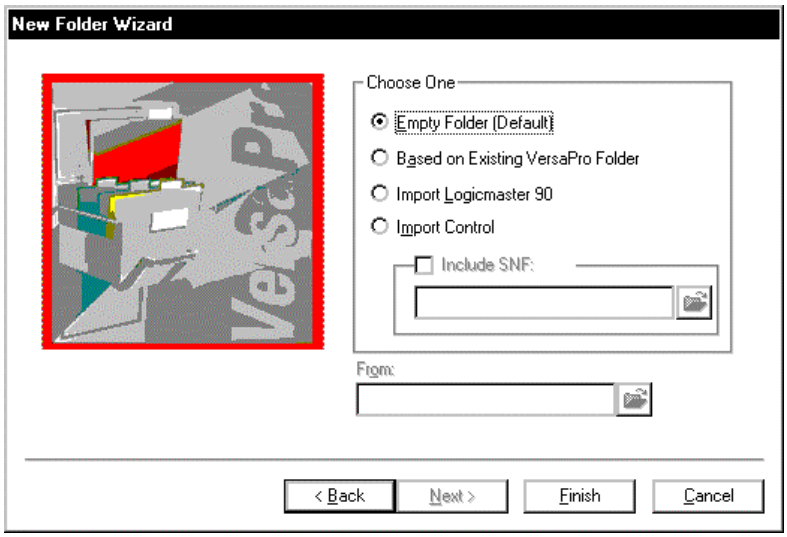
1. Spusťte software VersaPro.
2. Pod VersaPro vytvořte nový adresář VersaPro.

   A. Chcete-li vytvořit adresář VersaPro, zvolte menu File a pak New Folder. Tím se otevře obrazovka New Folder Wizard. Na ní do pole Folder Name zapište název adresáře, umístění (Location) a popis adresáře (Folder Description). Pro účely tohoto příkladu zapište text ukázaný v dialogovém okně. (Obrázek 2-18)



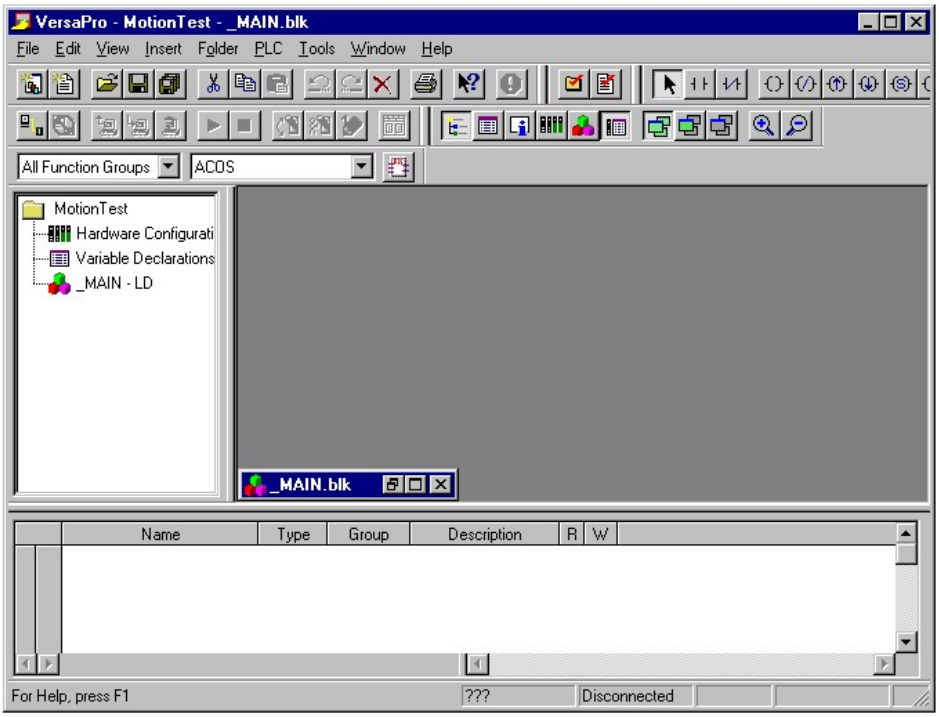
Obrázek 2-18. Stránka 1 obrazovky New Folder Wizard

   B. Klikněte na tlačítko Next. Následující dialogové okno vám umožní buď začít s prázdným adresářem nebo ho naimportovat z jiného zdroje. Tento příklad začíná prázdným adresářem. (Obrázek 2-19)



Obrázek 2-19. Stránka 2 obrazovky New Folder Wizard

C. Kliknutím na tlačítko Finish vytvořte nový adresář. Výsledné zobrazení VersaPro je ukázáno na obrázku 2-20.

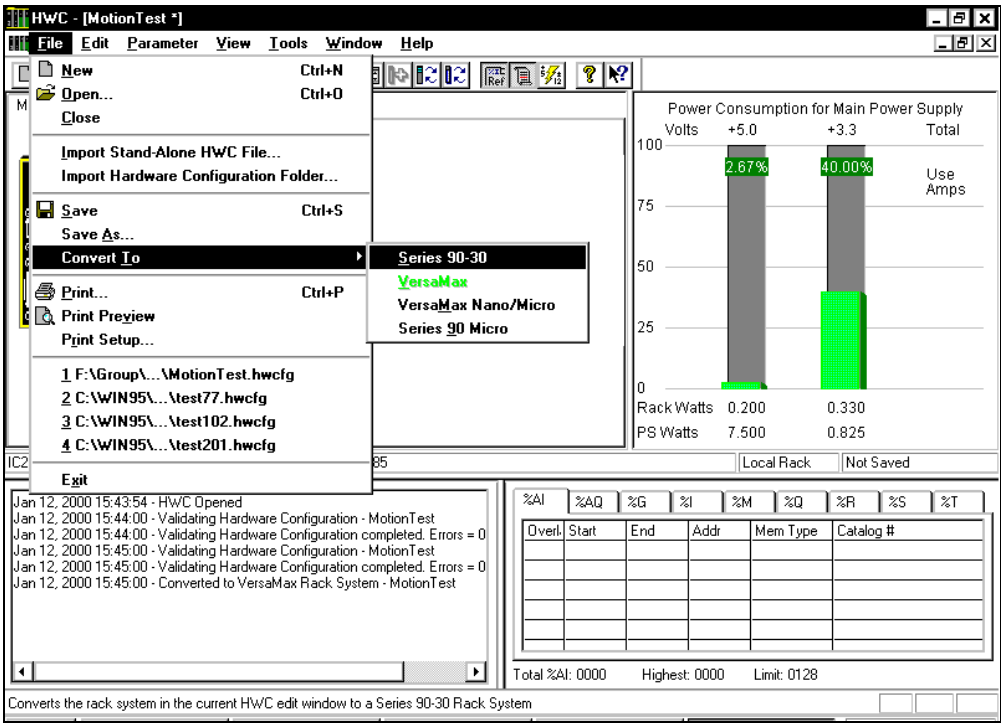


Obrázek 2-20. Hlavní obrazovka New Folder VersaPro

3. Spusťte konfigurační nástroj hardwaru. Je několik možností, jak to provést. (Další podrobnosti najdete v dokumentaci k VersaPro.) Dva způsoby jsou následující:

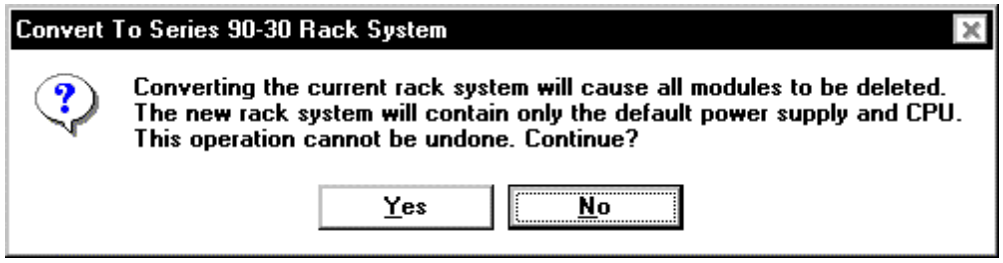
   - V menu View zvolte Hardware Configuration
   - Klikněte na ikonu Hardware Configuration v prohlížeči adresářů.

4. Možná budete potřebovat změnit výchozí konfiguraci hardwaru na Series 90-30. K tomu účelu na liště menu zvolte File, z menu File zvolte Convert To a pak Series 90-30 z menu Convert To. Tím se změní výchozí konfigurace PLC na Series 90-30. Volba menu je ukázána na Obrázek 2-21.



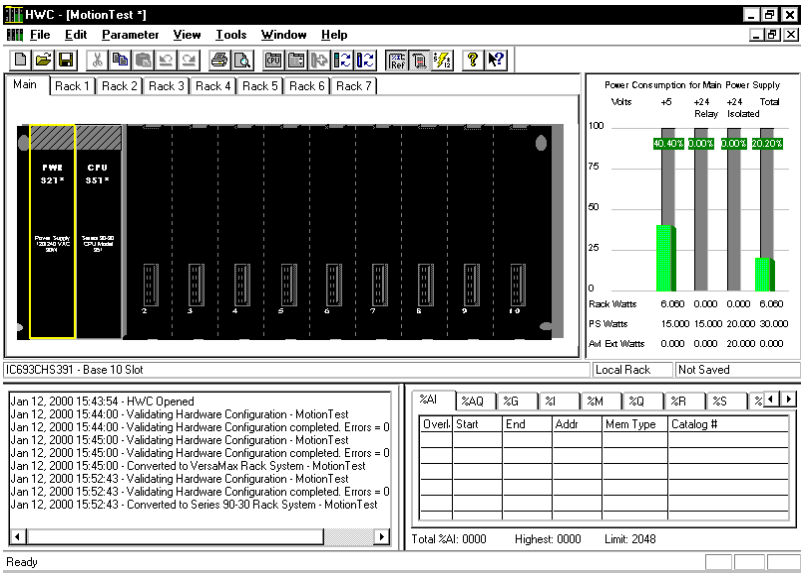
Obrázek 2-21. Volba konfigurace sestavy hardwaru

Dialogové okno vás upozorní, že informace budou smazané. Protože adresář vytvořený v tomto příkladu je nový, žádné informace se neztratí. **Pokud nevytváříte nový adresář, dejte pozor na to, že vykonáním této operace dojde ke ztrátě konfiguračních informací.** Doporučuje se, abyste pro tento příklad použili nový prázdný adresář. V dialogovém okně odpovězte Yes. (Obrázek 2-22)



Obrázek 2-22. Dialogové okno převodu konfigurace sestavy hardwaru

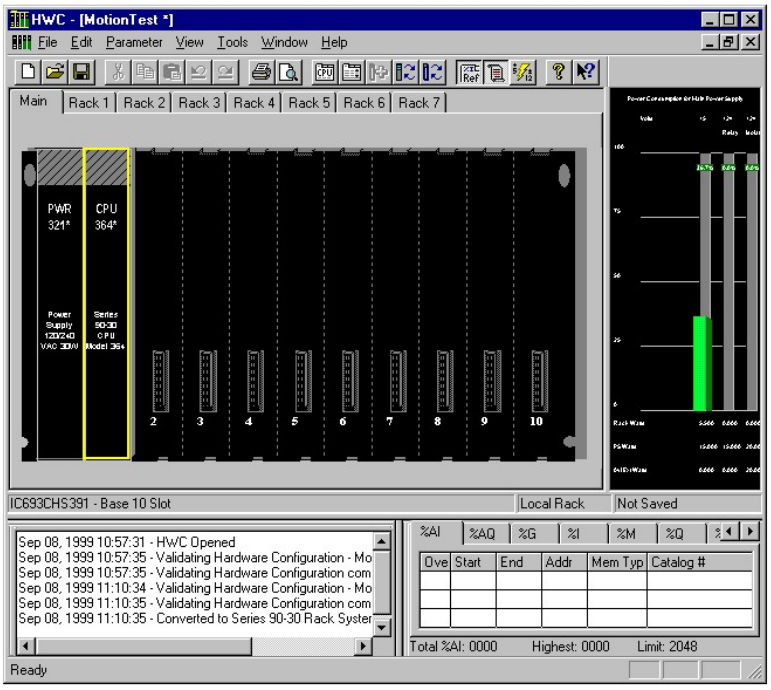
Jakmile bude tato operace hotová, budete mít prázdnou sestavu 90-30, kterou je nutno nakonfigurovat. Výslednou obrazovku pro konfiguraci hardwaru ukazuje Obrázek 2-23.



Obrázek 2-23. Hardwarová konfigurace sestavy 90-30 s CPU

5. Zvolte napájecí zdroj a CPU, které jsou vhodné pro vaší instalaci. Všimněte si, že DSM314 vyžaduje CPU s firmwarem verze 10 nebo vyšší. Výchozí CPU, "CPU351," nepodporuje firmware verze 10.0. Dokumentace CPU obsahuje seznam hardwaru CPU, které podporují funkce verze 10.0. Tento příklad používá CPU364.

   Chcete-li změnit model CPU, klikněte pravým tlačítkem myši na CPU a zvolte Replace Module. Objeví se dialogové okno Module Catalogue. Zvolte model CPU, které chcete použít ve svém systému, a klikněte na OK. Klikněte na Yes v zobrazeném výstražném dialogovém okně. Výsledné zobrazení bude vypadat jako na Obrázek 2-24.



Obrázek 2-24. Hardwarová konfigurace sestavy 90-30 s CPU364

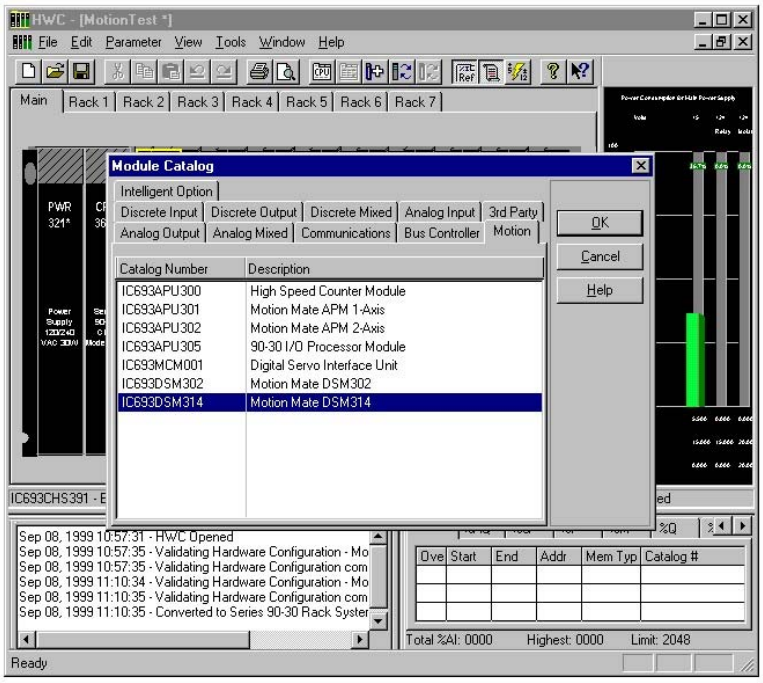
6. Přidání DSM314 do sestavy.

   Zvolte pozici sestavy, kde má být DSM314 nainstalované (v tomto příkladu pozice 2). Otevřete dialogové okno Module Catalog pomocí některé z následujících metod:

   - Dvakrát klikněte na požadovanou pozici (v tomto případě pozice číslo 2).
   - Na liště menu zvolte Edit, z podmenu Edit zvolte Module Operations a pak z menu Module Operations zvolte Add Module.

7. V dialogovém okně Module Catalog zvolte záložku Motion, pak ze seznamu zvolte modul DSM314. Výsledné zobrazení bude vypadat jako na Obrázek 2-25.

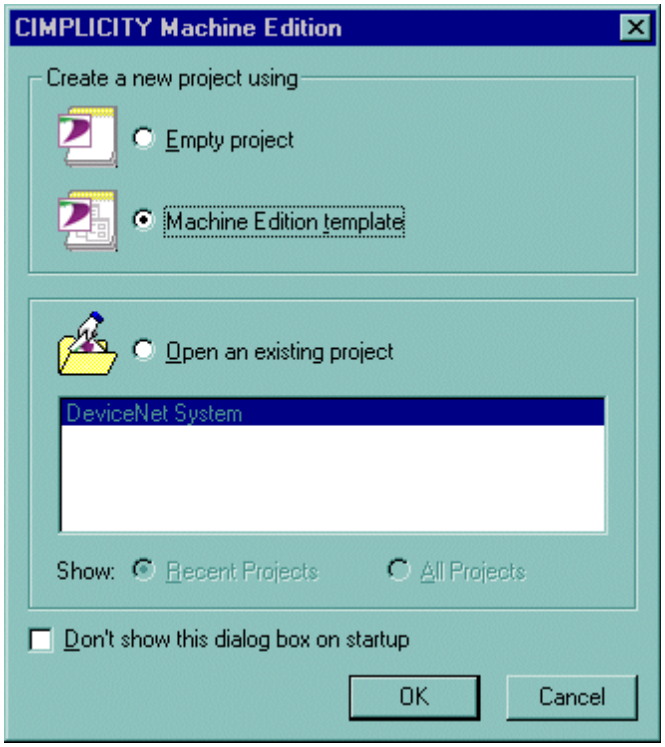
   Tato operace přidá DSM314 do sestavy a zobrazí konfigurační obrazovky DSM314, které vám umožní přizpůsobit DSM314 pro vaší konkrétní aplikaci. Část "Konfigurační data modulu (pouze stručný úvod do výuky)" na straně 2-35 uvádí příklad konfigurace. Podrobnosti ohledně konfiguračních nastavení DSM314 najdete v kapitole 4.



Obrázek 2-25. Volba hardwarové konfigurace 90-30 se sestavou DSM314

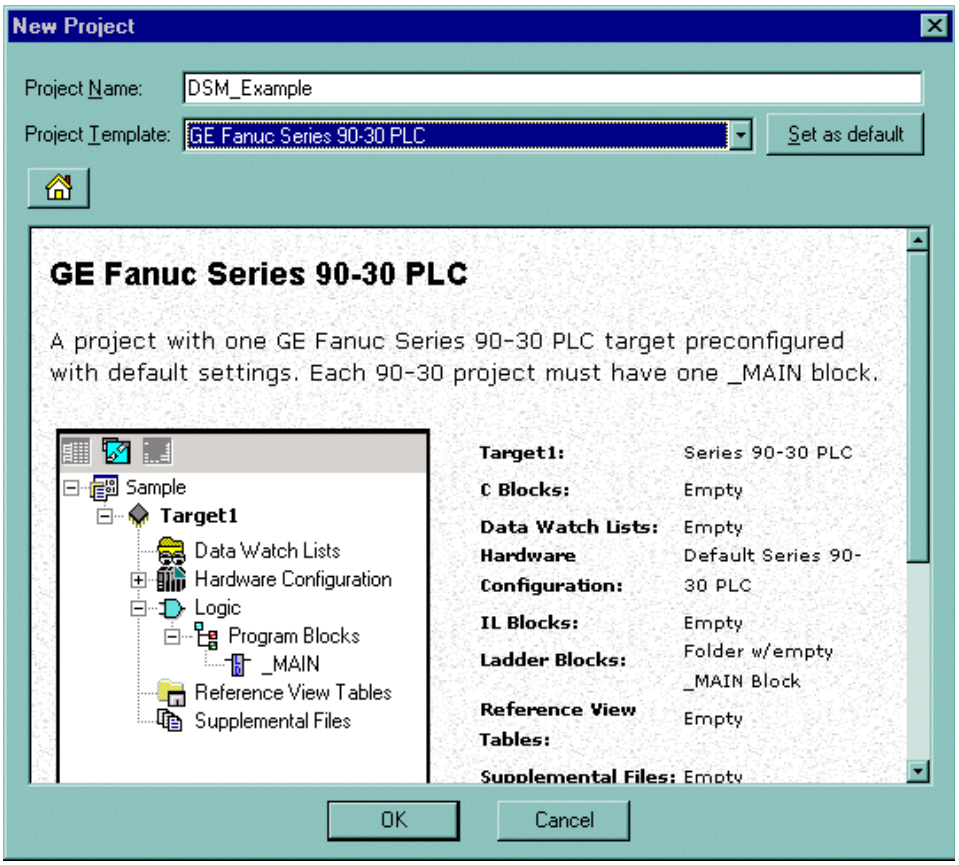
Konfigurace CIMPLICITY Machine Edition

1. Spusťte software CIMPLICITY Machine Edition Logic Developer – PLC. Zobrazí se dialogové okno CIMPLICITY Machine Edition.

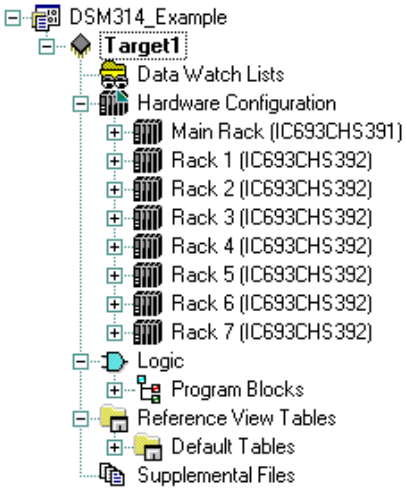


2. V části Create a New Project zvolte Machine Edition Template a klikněte na OK. Zobrazí se dialogové okno New Project .

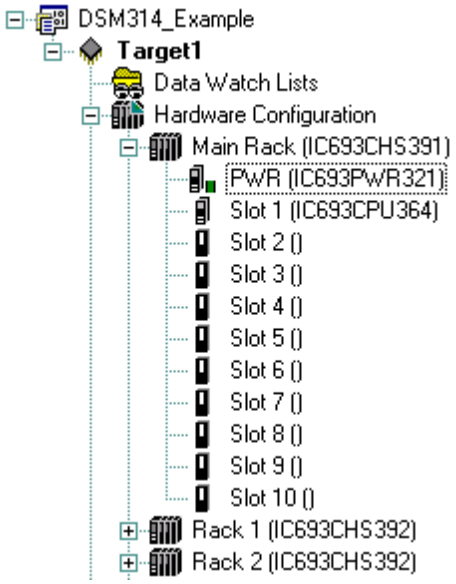
3. Zapište název projektu do Project Name. V rozbalovacím menu Project Template zvolte GE Fanuc PLC Series 90-30. Klikněte na OK.



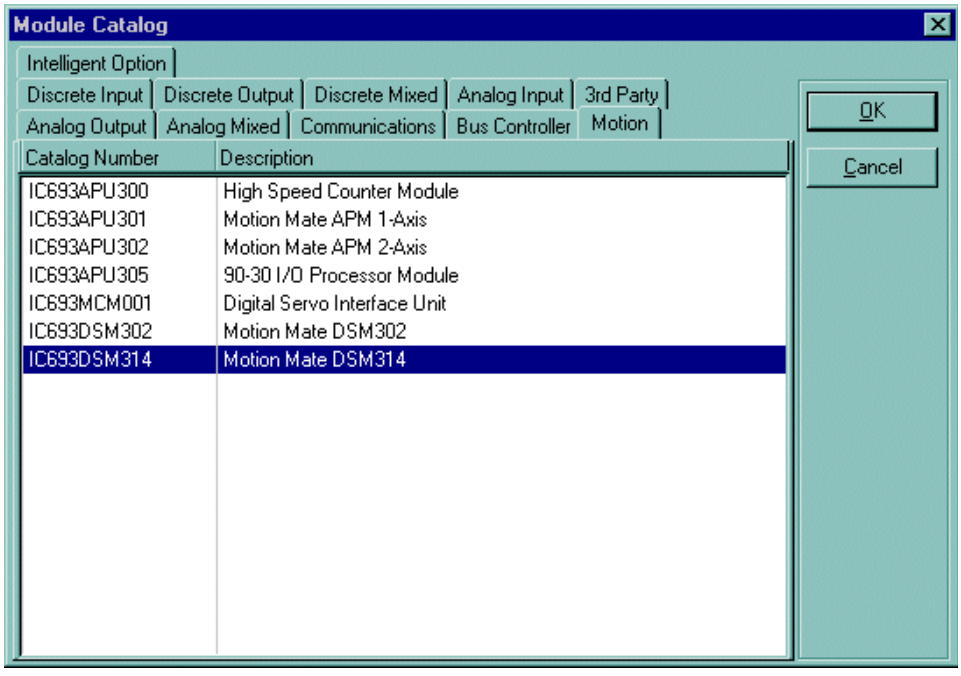
Váš projekt se objeví v okně Navigátor, jak je znázorněno v následujícím obrázku.



4. Rozbalte ⊞ uzel Main Rack, který obsahuje výchozí napájecí zdroj a CPU.



5. Pokud bude nutné, nahraďte napájecí zdroj a/nebo CPU modelem, který budete používat ve své aplikaci. Chcete-li vyměnit modul, klikněte pravým tlačítkem myši a zvolte Replace Module.
6. Přidejte DSM314 do konfigurace sestavy.
   A. Klikněte pravým tlačítkem myši na prázdnou pozici a zvolte Add Module. Objeví se dialogové okno Module Catalog.
   B. Zvolte záložku Motion, vyberte DSM314 a klikněte na OK.



Tato operace přidá do sestavy DSM314 a zobrazí konfigurační obrazovky DSM314, které vám umožní přizpůsobit DSM314 pro vaší konkrétní aplikaci. Část “Konfigurační data modulu (pouze stručný úvod do výuky)” na straně 2-35 uvádí příklad konfigurace. Podrobnosti ohledně konfiguračních nastavení DSM314 najdete v kapitole 4.

Konfigurační data modulu (pouze stručný úvod do výuky)

Pokud jste postupovali výše uvedeným způsobem, otevřou se konfigurační obrazovky DSM314. Pokud budete začínat v tomto bodě, klikněte dvakrát na modul DSM314, který chcete editovat. Tato operace otevře konfigurační obrazovky. Nyní z následujících tabulek zvolte nebo zapište hodnoty typu servosystému, digitálního nebo analogového, který budete používat. Kapitola 4 tohoto manuálu uvádí další podrobnosti o jednotlivých konfiguračních parametrech, které vám umožní přizpůsobit nastavení podle požadavků vaší konkrétní aplikace.

Tabulka 2-7. Záložka nastavení (pouze stručný úvod do výuky)

Konfigurační parametr	Popis	Pro digitální režim	Pro analogový režim	Výchozí nastavení
Počet os	Počet řízených os	**2**	4	4
Adresa %I	Počáteční adresa pro typ adresy %I (80 bitů)	%I00001	%I00001	Další nejvyšší použitelná adresa %I
Délka %I	Délka adresy %I	**48**	80	80
Adresa %Q	Počáteční adresa pro typ adresy %Q (80 bitů)	%Q00001	%Q00001	Další nejvyšší použitelná adresa %Q
Délka %Q	Délka adresy %Q	**48**	80	80
Adresa %AI	Počáteční adresa pro typ adresy %AI (84 bitů)	%AI00001	%AI00001	Další nejvyšší použitelná adresa %AI
Délka %AI	Délka adresy %AI	**44**	84	84
Adresa %AQ	Počáteční adresa pro typ adresy %AQ (12 bitů)	%AQ00001	%AQ00001	Další nejvyšší použitelná adresa %AQ
Délka %AQ	Délka adresy %AQ	**6**	12	12
Režim osy 1	Režim řízení osy 1	**Digitální servo**	Analogové servo	Analogové servo
Režim osy 2	Režim řízení osy 2	**Digitální servo**	Analogové servo	Analogové servo
Režim osy 3	Režim řízení osy 3	**Pomocná osa**	Analogové servo	Pomocná osa
Režim osy 4	Režim řízení osy 4	**Zakázáno**	Analogové servo	Zakázáno
Režim lokální logiky	Režim nástroje lokální logiky	Zakázáno	Zakázáno	Zakázáno
Celkový výkon snímače polohy	Požadavky na výkon snímače polohy	0	**	0
Název bloku pohybového programu	Název pohybového programu vykonávaný modulem	<prázdné>	<prázdné>	<prázdné>
Název bloku lokální logiky	Název programu lokální logiky vykonávaného modulem	<prázdné>	<prázdné>	<prázdné>
Název bloku vačky	Název bloku vačky vykonávaného modulem	<prázdné>	<prázdné>	<prázdné>

Tabulka 2-8. Záložka osy (pouze stručný úvod do výuky)

Konfigurační parametr	Popis	Pro digitální režim	Pro analogový režim	Výchozí nastavení
Uživatelské jednotky	Hodnota uživatelských jednotek	1	1	1
Počet	Počet jednotek zpětné vazby	1	1	1
Spínač omezení přejezdu	Povolení/zakázání spínače omezení přejezdu	**Zakázáno**	**Zakázáno**	Povoleno
Vstup připravenosti pohonu	Řízení vstupu připravenosti pohonu	Povoleno	***Závisí na servozesilovači***	Povoleno
Horní mez polohy	Horní mez polohy	+8 388 607	+8 38 8607	+8 388 607
Dolní mez polohy	Dolní mez polohy	-8 388 608	-8 388 608	-8 388 608
Horní softwarová mez konce posuvu	Horní softwarová mez konce posuvu	+8 388 607	+8 388 607	+8 388 607
Dolní softwarová mez konce posuvu	Dolní softwarová mez konce posuvu	-8 388 608	-8 388 608	-8 388 608
Softwarový konec posuvu	Řízení softwarového konce posuvu	Zakázáno	Zakázáno	Zakázáno
Mez rychlosti	Mez rychlosti osy	1 000 000	1 000 000	1 000 000
Směr povelu	Přípustný zadaný směr	Obousměrný	Obousměrný	Obousměrný
Směr osy	Směr osy	Normální	Normální	Normální
Zdroj zpětné vazby	Typ zpětné vazby	Výchozí	Výchozí	Výchozí
Režim zpětné vazby (pouze digitální režim)	Režim zpětné vazby	Inkrementální	**nepoužívá se**	Inkrementální
Kompenzace obrácení chodu	Kompenzace obrácení chodu	0	0	0
Prodleva zákazu pohonu	Prodleva zákazu pohonu	100	100	100
Rychlost jogu	Rychlost jogu	+1000	+1000	+1000
Zrychlení jogu	Zrychlení jogu	+10,000	+10,000	+10,000
Režim zrychlení jogu	Režim zrychlení jogu	Lineární	Lineární	Lineární
Výchozí poloha	Výchozí poloha	0	0	0
Offset výchozí polohy	Hodnota offsetu výchozí polohy	0	0	0
Konečná rychlost výchozí polohy	Konečná rychlost výchozí polohy	+500	+500	+500
Rychlost pro nalezení výchozí polohy	Rychlost pro nalezení výchozí polohy	+2000	+2000	+2000
Režim výchozí polohy	Režim nalezení výchozí polohy	**Posuv +**	**Posuv +**	Spínač výchozí polohy
Režim dat návratu 1	Režim dat návratu 1	0	0	0
Offset dat návratu 1	Offset dat návratu 1	0	0	0
Režim dat návratu 2	Režim dat návratu 2	0	0	0
Offset dat návratu 2	Offset dat návratu 2	0	0	0
Master zdroj vačky	Master zdroj vačky	Aktuální poloha 3	Aktuální poloha 3	Aktuální poloha 3
Řídicí smyčka vlečené osy	Povolení řídicí smyčky vlečené osy	Zakázáno	Zakázáno	Zakázáno
Hodnota poměru A	Poměr A vlečené osy A/B	1	1	1
Hodnota poměru B	Poměr B vlečené osy A/B	1	1	1
Master zdroj vlečené osy 1	Master zdroj vlečené osy 1	Žádný	Žádný	Žádný
Master zdroj vlečené osy 2	Master zdroj vlečené osy 2	Žádný	Žádný	Žádný

Tabulka 2-8, pokračování

Start povolení vlečené osy	Vstup startu povolení vlečené osy	Žádný	Žádný	Žádný
Start zákazu vlečené osy	Vstup startu povolení vlečené osy	Žádný	Žádný	Žádný
Akce při zákazu vlečené osy	Akce při zákazu vlečené osy	Stop	Stop	Stop
Zrychlení pozvolného náběhu	Zrychlení pozvolného náběhu vlečené osy	10,000	10,000	10,000
Režim pozvolného náběhu	Režim pozvolného náběhu vlečené osy	Doba pozvolného náběhu	Doba pozvolného náběhu	Doba pozvolného náběhu
Doba pozvolného náběhu	Doba zrychlení pozvolného náběhu vlečené osy	0	0	0
Rychlost pozvolného náběhu	Rychlost pozvolného náběhu vlečené osy	+1000	+1000	+1000

Tabulka 2-9. Záložka ladění (pouze stručný úvod do výuky)

Konfigurační parametr	Popis	Pro digitální režim	Pro analogový rychlostní režim	Výchozí nastavení
Typ motoru	Typ motoru	**Z Tabulka 2-10**	0	0
Povel analogového serva	Typ povelu analogového serva	Rychlost	Rychlost	Rychlost
Mez polohové odchylky	Mez polohové odchylky	60,000	60,000	60,000
Zóna dosažení polohy	Zóna dosažení polohy	10	10	10
Časová konstanta polohové smyčky (0,1 ms)	Časová konstanta polohové smyčky	**600***	1000*	1000
Rychlost při MaxCmd	Rychlost při maximálním povelu	**nepoužívá se**	100,000*	100,000
Procento rychlosti posuvu	Procento rychlosti posuvu	0	0	0
Procento zrychlení posuvu	Procento zrychlení posuvu	0	0	0
Režim integrátoru	Režim polohové smyčky integrátoru	Vypnuto	Vypnuto	Vypnuto
Časová konstanta integrátoru	Časová konstanta polohové smyčky integrátoru	0	0	0
Zisk rychlostní smyčky	Zisk rychlostní smyčky	16	16	16

* Správné nastavení je určeno během spuštění/ladění systému. Viz strana “Informace o spuštění a ladění pro analogové servosystémy” v Dodatku D.

** Správná nastavení jsou určená požadavky na napájení 5 V.

Poznámka: Výše uvedená nastavení v případě potřeby zopakujte pro konfiguraci os 2, 3 a 4.

Typ motoru (digitální režim)

Zvolte typ střídavého servomotoru FANUC používaného s DSM314. DSM314 má interně uloženou tabulku výchozích parametrů nastavení motoru pro jednotlivá podporovaná serva GE Fanuc. Konkrétní motor pro uvedenou osu se zvolí přes konfigurační pole *Motor1 Type* (pro osu 1) nebo *Motor2 Type* (pro osu 2). Podporované typy motorů jsou uvedené v následující tabulce.

Model motoru FANUC: Informace o modelu motoru je ve tvaru *řada, souvislý krouticí moment v Nm / maximální otáčky*. Příklad: β2/3000 udává motor Beta (β) Series se souvislým kroutícím momentem 2 Nm a maximální souvislou rychlostí 3000 ot./min.

Specifikace motoru FANUC: Konfigurační data pro pole typu motoru jsou určena datovým štítkem motoru. Čísla motoru jsou ve tvaru A06B-xxxx-yyyy, kde xxxx označuje specifikační pole motoru. Například: významné číslice 0032 ze štítku motoru A06B-0032-B078 udávají model motoru β2/3000. Následující tabulka ukazuje, že pro specifikační číslo motoru 0032 je správný typ motoru 36.

Tabulka 2-10. Volba kódu typu motoru FANUC

Kód typu motoru	Model motoru	Specifikace motoru
61	α 1/3000	0371
46	α 2/2000	0372
62	α 2/3000	0373
15	α 3/3000	0123
16	α 6/2000	0127
17	α 6/3000	0128
18	α 12/2000	0142
19	α 12/3000	0143
27	α 22/1500	0146
20	α 22/2000	0147
21	α 22/3000	0148
28	α 30/1200	0151
22	α 30/2000	0152
23	α 30/3000	0153
30	α 40/2000	0157
29	α 40/FAN	0158
56	αL 3/3000	0561
57	αL 6/3000	0562
58	αL 9/3000	0564
59	αL 25/3000	0571
60	αL 50/2000	0572

Pokračování na následující stránce

Tabulka 2-10, pokračování

Kód typu motoru	Model motoru	Specifikace motoru
7	αC 3/2000	0121
8	αC 6/2000	0126
9	αC 12/2000	0141
10	αC 22/1500	0145
3	α 12HV/3000	0176
4	α 22HV/3000	0177
5	α 30HV/3000	0178
24	αM 3/3000	0161
25	αM 6/3000	0162
26	αM 9/3000	0163
13	β 0.5/3000	0113
115	βM 0.5/5000	0115
35	β 1/3000	0031
116	βM1/5000	0116
36	β 2/3000	0032
33	β 3/3000	0033
34	β 6/2000	0034

Uložení konfigurace do PLC

Konfiguraci svého systému Series 90-30 je nutno dokončit tak, aby obsahovala napájecí zdroj, sestavu, CPU a další moduly splňující účel systému. Podle potřeby vyhledejte informace v uživatelském manuálu softwaru a v on-line nápovědě.

DŮLEŽITÉ

Kompletní konfiguraci je nutno uložit do PLC. Instrukce, jak to provést, najdete v odstavci "Připojení a uložení konfigurace do PLC" v kapitole 15. Další podrobnosti najdete v uživatelském manuálu softwaru a v on-line nápovědě.

Pokud se všechno bude zdát v pořádku (to znamená, že po uložení nejsou žádné chyby stavu PLC nebo IO atd.),

dalším krokem bude: ***Testování vašeho systému***.

Poznámka: Chyba stavu PLC "Nesoulad konfigurace systému" se stejným umístěním sestavy/pozice jako DSM314 indikuje, že byl nakonfigurovaný parametr a poslaný do DSM314, který byl v DSM314 odmítnutý. Pečlivě zkontrolujte každý parametr konfigurace vašeho DSM314 s konfiguračním nastavením v tomto manuálu, jestli zde není nějaká neshoda. Opravte neshodu, vynulujte chybu PLC a uložte konfiguraci znovu. Přesvědčte se, že se chyba opravila. Instrukce k prohlížení a vynulování chyb PLC najdete v další části, Povolení režimu Run na PLC.

DSM314 může zjišťovat mnoho typických chyb konfigurace. Ty se vrátí jako chybové kódy ve tvaru Dxxx (hex) ve slově %AI *Stavový kód modulu* nebo ve slově %AI *Chybový kód osy*. Tyto chyby nevyvolají stav PLC "Nesoulad konfigurace systému". Popis těchto chyb viz Dodatek A. Opravte všechny konfigurační chyby a uložte konfiguraci PLC znovu v režimu Stop.

Část 5: Testování vašeho systému

Generování pohybu

Upozornění

Aby stroj správně pracoval, je nutno provést správný postup spouštění. Sem patří potvrzení činnosti spínačů mezí přejezdu a výchozí polohy, kontrola správného směru točení motoru a vyladění rychlostní a polohové smyčky. To musí provést zkušený pracovník. Podrobné instrukce pro spouštění najdete v Dodatku D, "Spouštění a ladění digitálního nebo analogového servosystému GE Fanuc".

Povolení režimu Run na PLC

Dalším krokem pro činnost systému DSM314 je přepnutí PLC do režimu RUN.

VersaPro

1. Z hlavního menu VersaPro zvolte menu PLC. Z podmenu PLC zvolte Connect. Zobrazí se menu Connect. Z tohoto menu zvolte metodu komunikace/port, který chcete používat při komunikaci s PLC. Pokud jste ještě neprovedli konfiguraci komunikačního portu, další informace najdete v dokumentaci k VersaPro GFK-1670 nebo v on-line nápovědě VersaPro.
2. Z hlavního menu VersaPro zvolte menu Tools. Z podmenu Tools zvolte položku Fault Table. Tím se vyvolá zobrazení tabulky chyb PLC. Prohlédněte si problémy v tabulce chyb a pak použijte funkční tlačítko F9 – Clear k vymazání všech aktuálních chyb PLC. Zvolte záložku tabulky chyb I/O a proveďte stejnou operaci také s touto tabulkou. Uzavřete tabulku chyb a vraťte se na hlavní obrazovku VersaPro.
3. Z hlavního menu VersaPro zvolte menu PLC. Z podmenu PLC zvolte položku Run. Tím se PLC převede do režimu běhu.

Výstraha

Přesvědčte se, že váš motor je dobře připevněný. Pokud by nebyl, může dojít k jeho poškození a/nebo ke zranění obsluhy.

4. Vizuálně zkontrolujte, že LED kontrolky PLC s označením PWR, OK a RUN svítí. Vizuálně zkontrolujte, že LED kontrolky DSM314 s označením STAT, OK a CFG svítí. V případě digitálních zesilovačů α Series vizuálně zkontrolujte, že stavový údaj ukazuje záporné znaménko (–), což indikuje režim Standby. V případě digitálních zesilovačů β Series zkontrolujte, že LED kontrolka Power svítí a LED kontrolky ALM jsou zhasnuté. V případě zesilovačů SL Series zkontrolujte, jestli displej na předním panelu na každém zesilovači nezobrazuje nějaké chyby (jsou indikované číslicemi na displeji, které se střídavě rozsvítí a zhasínají).

5. Pokud zesilovač bude indikovat chybu, může být nutné zesilovač a PLC zcela vypnout a zopakovat sekvenci zapnutí napájení popsanou v části 3.
6. Pokud se žádné chyby nevyskytují, měli byste být schopní provést jog připojených os. Aby se u výše popsané spouštěné konfigurace vynulovaly všechny chyby indikované blikající LED kontrolkou STAT na DSM314, překlopte (do jedničky a pak do nuly) bit *Clear Error %Q* (u této konfigurace bit %Q1) v datové tabulce PLC. Stavy, které způsobují chybu, je nutno opravit, aby se vynuloval indikátor stavu. Informace o chybovém stavu DSM najdete v Dodatku A, "Chybové kódy".

CIMPLICITY Machine Edition

1. Chcete-li nakonfigurovat komunikaci v navigačním okně CIMPLICITY Machine Edition, klikněte pravým tlačítkem myši na uzel Target a zvolte Properties. Otevře se okno Inspektor zobrazující vlastnosti cíle, které zahrnují konfigurační parametry pro komunikační linku. Podrobnosti najdete v on-line nápovědě nebo v manuálu *CIMPLICITY® Machine Edition Logic Developer - Stručný úvod do PLC*, GFK-1918.
2. Klikněte pravým tlačítkem myši na uzel Target a zvolte Go Online. Objeví se dialogové okno Connecting. Když se programovací zařízení úspěšně připojí k PLC, ikona uzlu Target se změní a indikuje Online and Equal nebo Online and Not Equal. (Pokud spojení bude Online and Not Equal, logika a/nebo konfigurace v PLC nesouhlasí s konfigurací v programovacím zařízení. Pro účely této výuky může být nutné načíst konfiguraci.)
3. Chcete-li otevřít Fault Table Viewer, klikněte pravým tlačítkem myši na uzel Target a zvolte Diagnostics. Prohlédněte tabulku chyb PLC, jestli v ní nejsou problémy, a pak všechny aktuální chyby PLC vynulujte. Proveďte stejnou operaci s chybami I/O.
4. V okně Navigátor klikněte pravým tlačítkem myši na uzel Target a zvolte Online Commands, pak Start PLC. Tím se PLC převede do režimu běhu.

Výstraha

Přesvědčte se, že váš motor je dobře připevněný. Pokud by nebyl, může dojít k jeho poškození a/nebo ke zranění obsluhy.

5. Vizuálně zkontrolujte, že LED kontrolky PLC s označením PWR, OK a RUN svítí. Vizuálně zkontrolujte, že LED kontrolky DSM314 s označením STAT, OK a CFG svítí. V případě digitálních zesilovačů α Series vizuálně zkontrolujte, že stavový údaj ukazuje záporné znaménko (–), což indikuje režim Standby. V případě digitálních zesilovačů β Series zkontrolujte, že LED kontrolka Power svítí a LED kontrolky ALM jsou zhasnuté. V případě zesilovačů SL Series zkontrolujte, jestli displej na předním panelu na každém zesilovači nezobrazuje nějaké chyby (jsou indikované číslicemi na displeji, které se střídavě rozsvítí a zhasínají).
6. Pokud zesilovač bude indikovat chybu, může být nutné zesilovač a PLC zcela vypnout a zopakovat sekvenci zapnutí napájení popsanou v části 3.
7. Pokud se žádné chyby nevyskytují, měli byste být schopní provést jog připojených os. Aby se u výše popsané spouštěné konfigurace vynulovaly všechny chyby indikované blikající LED kontrolkou STAT na DSM314, překlopte (do jedničky a pak do nuly) bit *Clear Error %Q* (u této konfigurace bit %Q1) v datové tabulce PLC. Stavy, které způsobují chybu, je nutno opravit, aby se vynuloval indikátor stavu. Informace o chybovém stavu DSM najdete v Dodatku A, "Chybové kódy".

Jogování s Motion Mate DSM314

Rychlost jogu, Zrychlení jogu a ***Režim zrychlení jogu*** je možno v modulu DSM314 nakonfigurovat. Tyto hodnoty se používají vždy, když bit %Q *Jog Plus* nebo *Jog Minus* přejde do stavu ON. Všimněte si, že pokud oba bity Jog budou ve stavu ON, nebude se generovat žádný pohyb. Výchozí hodnoty pro rychlost jogu byly nastavené během konfigurace.

Jog je možno provést, když není zadaný žádný jiný pohyb. K provedení jogu bit %Q *Enable Drive* nemusí být ve stavu ON. Nastavení bitu %Q *Jog* do jedničky se automaticky zapne nebo povolí servo. Povolení zesilovače je slyšitelné a motor začne na hřídeli vyvíjet krouticí moment. Obráceně, když se bit jogu nastaví do nuly, pohon se automaticky zakáže.

Chcete-li provést jog osy v kladném směru, přepněte do jedničky %Q0021 pro osu 1 nebo %Q0037 pro osu 2. Všimněte si, že zde používané adresy jsou pro adresy %Q začínající na %Q0001. Počáteční adresu %Q je možno nakonfigurovat.

Pokud motor neprovede jog, jděte do části 6, “Lokalizace chyb pohybového systému”.

Část 6: Lokalizace chyb pohybového systému

Alarmy

Prvním krokem při opravě chyb je zjištění, jestli se vyskytly nějaké alarmy. Alarmy PLC nebo chyby je možno prohlížet v tabulce chyb PLC. Alarmy serva a pohybového podsystému je možno prohlížet v DSM314 slově %AI *Module Status Code* nebo jednom ze slov %AI *Axis Error Code*. Další informace o chybách hlášených pomocí dat %AI najdete v kapitole 5.

Více informací o alarmech Motion Mate DSM314 najdete v Dodatku A, "Chybové kódy", který obsahuje seznam chybových kódů a jejich popis.

Více informací o lokalizaci chyb najdete v Dodatku D, "Spouštění a ladění digitálního nebo analogového servosystému GE Fanuc".

Nastavení konfigurace

Pokud systém po zapnutí napájení bude hlásit alarm, může to být v důsledku nesprávného nastavení konfigurace.

Konfigurace PLC Series 90-30 musí být uložená v CPU PLC a PLC musí být v režimu Run/Output Enabled.

Pokud nemůžete vykonat pohyb osy nebo vykonat jog, zkontrolujte, že jsou splněné podmínky k provedení těchto operací. Viz příslušné části v tomto manuálu.

Kde hledat pomoc

Webová stránka GE Fanuc

http://www.gefanuc.com/support

Telefonní čísla GE Fanuc

Pokud stále nebudete schopni vyřešit problém, můžete zavolat na některé telefonní číslo GE Fanuc pro vaší oblast, uvedené v následující tabulce:

Oblast	Číslo
Severní Amerika, Kanada, Mexiko (horká linka technické pomoci)	Bez poplatku: 1-800 GE Fanuc Přímé volání: (780) 420-2000
Latinská Amerika (Mexiko viz výše)	Přímé volání: (780) 420-2000
Francie, Německo, Lucembursko, Švýcarsko a Spojené Království	Bez poplatku: 00800 433 268 23
Itálie	Bez poplatku: 16 77 80 596
Ostatní evropské země	352 (72) 79 79 309
Asie / Pacifik - Singapur	65-566 4918
Indie	91-80-552 0107
Česká Republika a Slovensko	420-233 372 503

Část 7: Další kroky

Co dělat dále po úspěšném vykonání pohybu osy?

Tato kapitola o spouštění nezahrnuje všechny aspekty pohybového systému DSM314. Proto byste měli projít také informace uvedené ve všech manuálech (viz "Související publikace" v Úvodu tohoto manuálu). GE Fanuc dále nabízí příslušné kurzy. Pokud vaše aplikace bude vyžadovat speciální rozhraní stroje, měli byste také absolvovat kurs programování GE Fanuc. Informace o školení si zjistěte na čísle 1-800-GE FANUC nebo se spojte se svým prodejcem GE Fanuc.

Kapitola instalace a zapojování v tomto manuálu uvádí návod pro dokončení instalace hardwaru DSM.

Kapitola konfigurace uvádí podrobnosti potřebné k nakonfigurování modulu DSM pro vaší aplikaci. Měli byste začít čtením části *Konfigurační parametry* v kapitole 4.

⇨ ***….. Dvě doporučení na závěr****:*

- Uložte papírové doklady, které jste dostali se systémem.

  List *Důležité informace o systému* obsahuje nejnovější informace k tomuto výrobku, z nichž některé nemusí být v tomto manuálu.

- Udělejte si zálohu adresáře žebříkové logiky.

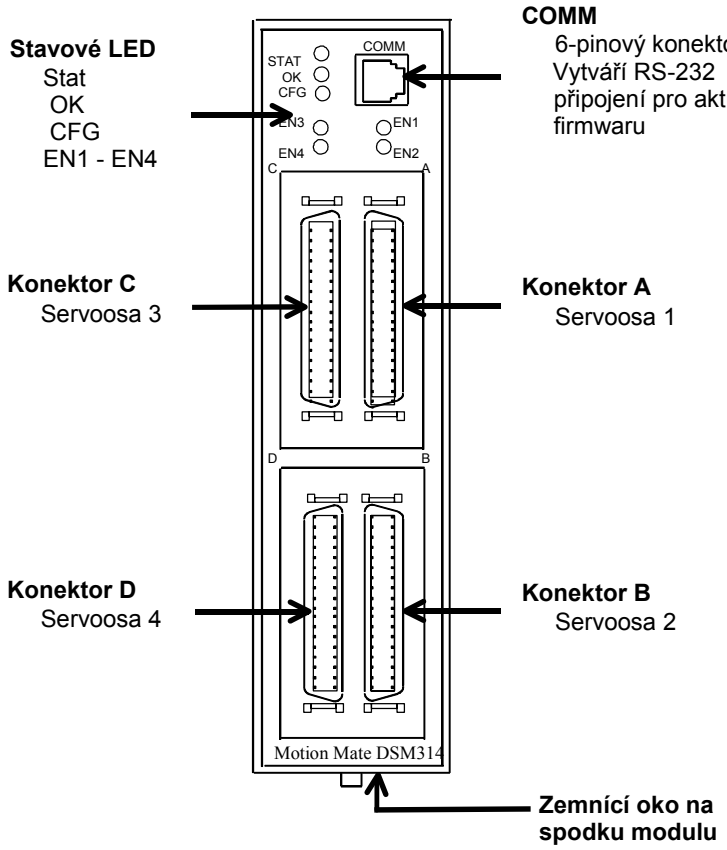
Důležité! . . . . Během vývoje své aplikace to provádějte často.

Kapitola 3

Instalace a zapojení DSM314

Část 1: Popis hardwaru

Tato část popisuje hlavní vlastnosti hardwaru modulu. Čelní deska modulu má sedm stavových LED, jeden konektor RJ-11 komunikačního portu a čtyři konektory uživatelských I/O (36 pinů). Zemnící oko ve spodní části modulu představuje snadný způsob, jak připojit stínění čelní desky modulu k zemi panelu.



Obrázek 3-1. Modul DSM314

LED kontrolky

Na modulu DSM314 je sedm stavových kontrolek LED popsaných níže:

STAT Normálně SVÍTÍ. BLIKÁNÍ indikuje chybu provozu. Bliká *pomalu* (čtyřikrát/sekundu) v případě pouze stavových chyb. Bliká *rychle* (osmkrát/sekundu) v případě chyb, které způsobí zastavení serva.

SVÍTÍ: Když LED bude trvale SVÍTIT, DSM314 funguje správně. Normálně by tato kontrolka LED měla vždy SVÍTIT.

NESVÍTÍ: Když tato LED bude ZHASNUTÁ, DSM314 nefunguje. Je to výsledkem hardwarové nebo softwarové závady, která brání, aby bylo možno modul zapnout.

Bliká: Když kontrolka LED bude BLIKAT, signalizuje se chybový stav.

Konstantní rychlost, LED kontrolka CFG SVÍTÍ:

LED bliká pomalu (čtyřikrát/sekundu) v případě pouze stavových chyb a rychle (osmkrát/sekundu) v případě chyb, které způsobí zastavení serva. Stavový bit %I *Chyba modulu* bude v jedničce. Chybový kód (hexadecimální formát) se uloží do slova %AI *Stavový kód modulu* nebo některého slova %AI *Chybový kód osy*.

Konstantní rychlost, LED kontrolka CFG bliká:

Pokud LED kontrolky STAT a CFG blikají **společně** konstantní rychlostí, modul DSM314 je v bootovacím režimu a čeká na načtení nového firmwaru. Pokud obě LED kontrolky STAT a CFG blikají **střídavě** konstantní rychlostí, firmware DSM314 zjistil překročení doby softwarového hlídacího obvodu v důsledku chyby hardwaru nebo softwaru.

Nepravidelná rychlost, LED kontrolka CFG NESVÍTÍ:

Pokud k tomuto dojde hned po zapnutí napájení, pak se zjistila chyba hardwaru nebo softwaru. LED kontrolka modulu STAT bude blikáním zobrazovat dvě chybová čísla oddělené krátkou prodlevou. Čísla jsou určená počtem blikání v obou sekvencích. Poznamenejte si tato čísla a vyžádejte si u GE Fanuc informace, jakým způsobem problém opravit.

OK LED kontrolka OK indikuje aktuální stav modulu DSM314.

SVÍTÍ: Když LED bude trvale SVÍTIT, DSM314 funguje správně. Normálně by tato kontrolka LED měla vždy SVÍTIT.

NESVÍTÍ: Když tato LED bude ZHASNUTÁ, DSM314 nefunguje. Je to výsledkem hardwarové nebo softwarové závady, která brání, aby bylo možno modul zapnout.

CFG Tato kontrolka LED bude SVÍTIT, když konfigurace modulu přišla z PLC.

EN1 Když tato LED bude SVÍTIT, výstup relé Axis 1 Drive Enable je aktivní.

EN2 Když tato LED bude SVÍTIT, výstup relé Axis 2 Drive Enable je aktivní.

EN3 Když tato LED bude SVÍTIT, výstup relé Axis 3 Drive Enable je aktivní.

EN4 Když tato LED bude SVÍTIT, výstup relé Axis 4 Drive Enable je aktivní.

Konektor DSM COMM (sériová komunikace)

Čelní panel modulu obsahuje jeden konektor RJ-11 pro sériovou komunikaci s označením "COMM". Používá se k načtení aktualizací firmwaru do modulu DSM z osobního počítače, na kterém běží GE Fanuc softwarová utilita PC Loader nebo Win Loader. (Podrobnosti viz Dodatek D.)

Tento sériový port COMM se připojuje k sériovému portu osobního počítače a používá GE Fanuc protokol SNP a normu sériové komunikace RS-232. Přenosovou rychlost je možno nastavit od 300 do 19,200 baud. Konfigurace portu COMM se provádí pomocí konfiguračního softwaru.

K připojení portu COMM k osobnímu počítači je možno od GE Fanuc zakoupit kabel IC693CBL316 v délce 1 metr. Tento kabel má 9-pinovou zásuvku D na straně počítače a konektor RJ-11 na straně DSM314. **Pokud budete používat delší kabel, maximální doporučená délka je 15 metrů (50 stop).**

Tabulka 3-1. Přiřazení pinů portu COMM na modulu DSM314

Číslo pinu RJ-11	Číslo pinu na 9-pinovém konektoru (zásuvce)	Název signálu	Popis
1	7	CTS	Smazat před odesláním
2	2	TXD	Přenos dat
3	5	0V	Signální zem
4	5	0V	Signální zem
5	3	RXD	Příjem dat
6	8	RTS	Požadavek na odeslání

Poznámka: Pin 1 je na konektoru dole při pohledu na modul zepředu.

I/O konektory

DSM314 je kontrolér dvouosého digitálního serva/jednoosého analogového rychlostního rozhraní nebo čtyřosého analogového serva (režim krouticího momentu a/nebo rychlostní režim) se čtyřmi 36-pinovými I/O konektory s označením A, B, C a D. Konektory jsou přiřazené následovně:

Tabulka 3-2. Přiřazení I/O konektorů osy

Konektor	Číslo osy	Typ osy	Použití I/O
A	1	Servoosa	Řízení digitálního nebo analogového serva v uzavřené smyčce
B	2	Servoosa Pomocná osa	Řízení digitálního nebo analogového serva v uzavřené smyčce Nebo Polohová zpětná vazba a pomocné analogové/digitální I/O
C	3	Servoosa Pomocná osa	Řízení analogového serva v uzavřené smyčce Nebo Polohová zpětná vazba a pomocné analogové/digitální I/O
D	4	Servoosa Pomocná osa	Řízení analogového serva v uzavřené smyčce Nebo Polohová zpětná vazba a pomocné analogové/digitální I/O

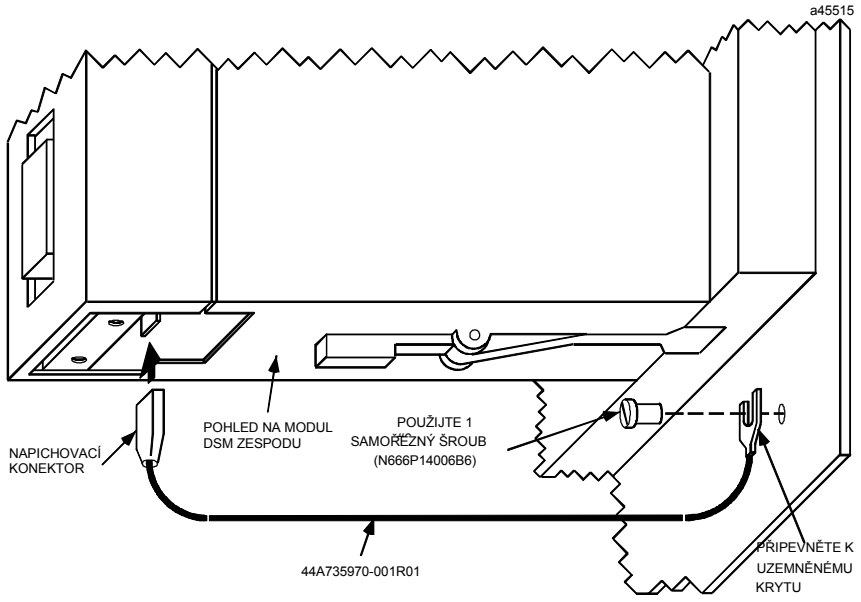
Všechny čtyři konektory mají podobné analogové a digitální I/O obvody. Pro řízení digitálních serv je možno nakonfigurovat **pouze** osu 1 a osu 2**. Pokud se používají digitální serva, osa 1 a osa 2 musí být obě nakonfigurované pro režim digitálního serva**. Když osa 1 a osa 2 budou nakonfigurované pro digitální serva, osu 3 je možno použít pro rozhraní analogového rychlostního serva nebo pro řízení pomocné osy. **Pokud osa 1 a 2 budou nakonfigurované pro digitální serva, osu 4 nelze použít pro rozhraní analogového rychlostního serva, rozhraní serva kroutícího momentu nebo řízení pomocné osy.**

Kdy osa 1 bude nakonfigurovaná pro řízení analogového serva (rozhraní krouticího momentu nebo rychlostní rozhraní), osu 2 až osu 4 je možno použít také pro analogové servo (rozhraní krouticího momentu nebo rychlostní rozhraní) nebo pro řízení pomocné osy. Funkce pomocné osy zahrnují vstup polohy pro master vlečené osy a generování interního (virtuálního master) povelu.

Každý z těchto čtyř konektorů použitý v systému je obvykle zapojený k příslušné svorkovnici kabelem IC693CBL324 (1 metr) nebo IC693CBL325 (3 metry). Pro připojení externích zařízení slouží tři různé svorkovnice se šroubovými svorkami. Svorkovnice jsou popsané dále v části "Svorkovnice" této kapitoly.

Připojení uzemnění stínění

Stínění čelní desky DSM314 musí být připojené ke kostře. Toto spojení z DSM314 ke kostře je možno provést pomocí zeleného zemnícího drátu (číslo dílu 44A735970-001R01) dodávaného s modulem. Tento drát má napichovací konektor na jednom konci pro připojení k 1/4-palcové svorce na spodní části DSM314 a koncové očko na druhém konci pro spoj k zemněnému krytu.



Obrázek 3-2. Připojení zemnění stínění

Část 2: Instalace modulu DSM314

Motion Mate DSM314 může pracovat v *každé* expanzní nebo vzdálené základní desce Series 90-30 (Series 90-30 verze 6.50 nebo pozdější). Konfigurační soubory vytvořené konfiguračním softwarem musí souhlasit s fyzickou konfigurací modulů.

> **Poznámka:** Všeobecné informace o instalaci a prostředí pro Series 90-30 najdete v *Manuálu pro instalaci a hardware PLC Series 90-30*, GFK-0356.

Při instalaci DSM314 na základní desku postupujte následovně:

1. K zastavení PLC použijte konfigurační software nebo ruční programovací zařízení. Tím se zabrání tomu, aby lokální aplikační program, pokud existuje, vyvolal povel, který by mohl ovlivnit činnost modulu při následném zapnutí napájení.
2. Vypněte PLC systém Series 90-30.
3. Dejte modul před požadovanou pozici a konektor. Nakloňte modul tak, aby horní zadní háček modulu zapadnul do pozice na horním okraji základní desky.
4. Sklopte modul dolů, až se konektory zasunou do sebe a zajišťovací páčka ve spodní části modulu zapadne do drážky v základní desce.
5. Připojte stínící drát čelní desky ze 1/4-palcové nožové svorky ve spodní části modulu na vhodný zemnící bod panelu.
6. Požadavky na zapojení I/O najdete na obrázcích 3-10 až 3-23 a v tabulkách 3-7 až 3-14.
7. Zapněte sestavu PLC. Stavová LED modulu Motion Mate DSM314 se rozsvítí, když kontrolér vykoná diagnostiku zapínání napájení.
8. Tento postup zopakujte pro každý modul DSM314 v systému PLC.
9. Nakonfigurujte moduly DSM314 podle popisu v kapitole 4.

Následující tabulka uvádí odběr proudu DSM314 a definuje počet modulů, které je možno nainstalovat v konkrétním PLC systému.

Počet modulů v systému může omezit:

- Kapacita napájecího zdroje roštu PLC
- Prostor I/O tabulky PLC. Modul DSM vyžaduje použití paměti %I, %Q, %AI a %AQ v I/O tabulce PLC přičemž typ %I a %Q je obvykle z těchto čtyř nejvíce omezující. %AI je také omezující u CPU, která nepodporují konfigurovatelnou paměť %AQ (například 350 CPU). Velikost použitelné paměti se mění podle modelu CPU PLC, které se má použít.
- Kapacita uložení konfiguračních dat PLC
- Použitelná paměť PLC

Absolutní meze jednotlivých typů PLC se nemusí překročit, protože v některých případech vycházejí z I/O tabulky a kapacity konfiguračních dat PLC.

Praktický počet os musí vzít v úvahu použití I/O a dobu cyklu celého systému.

Tabulka 3-3. Maximální počet DSM modulů na systém podle typu základní desky a napájecího zdroje

Napětí napájecího zdroje: **Proud, který odebírá DSM:** **Použitelný proud z +5 V/modul pro napájení externího snímače polohy, pokud se používá:**	5 V ss z propojovací roviny PLC 800 mA plus proud pro napájení snímače polohy (viz další položka). 500 mA (pokud se používá, musí se připočítat k proudu odebíranému z +5 V)
PLC model 350, 352, 360, 363, 364: **(základní desky CPU s 5 a 10 pozicemi, expanzní nebo vzdálené základní desky s 5 a 10 pozicemi - celkem 8 základních desek na systém)**	2 moduly DSM314 v základní desce CPU s PWR321/322/328 5 modulů DSM314 v základní desce CPU s PW330/331 3 moduly DSM314 v expanzní/vzdálené základní desce s PWR321/322/328 6 modulů DSM314 ve vzdálené základní desce s PWR330/331 7 modulů DSM314 v expanzní základní desce s PWR330/331 **Celkem 20 modulů DSM314 na PLC systém s PWR321/322/328*** **Celkem 20 modulů DSM314 na PLC systém s PWR330/331***

***Poznámka:** Series 90-30 normálně podporuje 20 modulů DSM314 na systém. Tento počet mohou snížit další moduly v systému, například moduly APM a GBC. Dále ho může snížit nastavení datagramů, které čtou referenční nebo chybové tabulky. Pokud se konfigurace a uživatelský program budou ukládat současně, přítomnost buď bloků C v LD programu nebo logický program C může mít také vliv na počet modulů DSM314, které je možno začlenit do systému. Pokud se uložení neprovede, lze konfiguraci do systému uložit při prvním ukládání logického programu a pak konfiguraci uložit při samostatném požadavku na uložení.

Počty uvedené v tabulce výše jsou teoretické maximální počty. Důležitým faktorem při určování počtu smíšených modul v základní desce však je, že celková spotřeba všech modulů nesmí přesáhnout celkové zatížení napájecího zdroje. Je možné, že konkrétní skladba modulů neumožní nainstalovat maximální počet modulů DSM314 do základní desky z důvodu omezení napájecího zdroje. Konfigurační software má automatické zobrazování využití napájecího zdroje, které je možno použít k tomuto účelu. Tento výpočet je také možno provést ručně, jak je ukázáno v *Manuálu pro instalaci PLC Series 90-30* (GFK-0356, verze P nebo pozdější), který také uvádí specifikace požadavků na zatížení pro moduly Series 90-30. Použitelné napájecí zdroje jsou:

Standardní napájecí zdroje Series 90-30

- *IC693PWR321* – Standardní AC/DC napájecí zdroj – výkon 15 W (3000 mA) z +5 V ss
- *IC693PWR322* – Napájecí zdroj s napájením 24/48 V ss – výkon 15 W (3000 mA) z +5 V ss
- *IC693PWR328* – Napájecí zdroj s napájením 48 V ss – výkon 15 W (3000 mA) z +5 V ss

Vysokokapacitní napájecí zdroje Series 90-30

- *IC693PWR330* – Vysokokapacitní AC/DC napájecí zdroj - výkon 30 W (6000 mA) z +5 V ss
- *IC693PWR331* – Vysokokapacitní s napájením 24 V ss, celkový výkon 30 W (6000 mA) z +5 V ss

Poznámky: Normy produktů a všeobecné specifikace najdete v GFK-0867B, (Agenturní schválení produktů GE Fanuc, Normy, Všeobecné specifikace), nebo pozdější verzi.

Pokyny pro instalaci v tomto manuálu jsou určené pro instalace, které nevyžadují zvláštní postupy pro hlučná a nebezpečná prostředí. Instalace, které musí splňovat přísnější požadavky (například značka CE), viz GFK-1179, *Požadavky na instalaci v souladu s normami.*

Část 3: Zapojení a připojení I/O

Typy I/O obvodů

Každý ze čtyř konektorů modulu (konektor A, B, C a D) umožňuje připojení následujících typů I/O obvodů:

- Tři diferenciální / jednodrátové 5 V vstupy (IN1-IN3)
- Napájení snímače polohy 5 V ss (P5V)
- Jeden jednodrátový vstup 5 V (IN4)
- Čtyři jednodrátové vstupní/výstupní obvody 5 V (IO5-IO8)
- Tři vstupy 24 V (IN9-IN11)
- Jeden výstup 24 V, 125 mA z polovodičového relé (OUT1)
- Dva diferenciální 5 V výstupy linkového budiče (OUT2-OUT3)
- Jeden výstup 24 V, 30 mA z polovodičového relé (OUT4)
- Dva diferenciální analogové vstupy +/- 10 V (AIN1-AIN2)
- Jeden jednodrátový analogový výstup +/- 10 V (AOUT1)

Ne všechny z těchto I/O obvodů je možno použít pro uživatelská připojení. Některé obvody se používají pro řízení digitálního servozesilovače GE Fanuc. Další informace najdete v tabulkách 3-11 až 3-14.

Svorkovnice

- **Svorkovnice osy, Katalogové číslo IC693ACC335** – Používá se pouze v digitálním režimu. Připojuje DSM konektor A nebo B k digitálnímu servozesilovači α nebo β GE Fanuc. Umožňuje také připojení I/O zařízení přes šroubové svorky. Tato svorkovnice má dva 36-pinové konektory. Jeden z nich se připojuje k DSM pomocí kabelu IC693CBL324/325 druhý se připojuje k digitálnímu servozesilovači GE Fanuc pomocí kabelu povelů serva IC800CBL001 / 002. Viz obrázky 3-10, 3-16, 3-17 a 3-18.

  Poznámka: U aplikací digitálního serva, které nevyžadují použití I/O signálů na DSM konektorech A nebo B je možno konektor DSM připojit přímo k digitálnímu servozesilovači GE Fanuc. Další informace najdete v části 3, “Zapojení a přípoje I/O” dále v této kapitole.

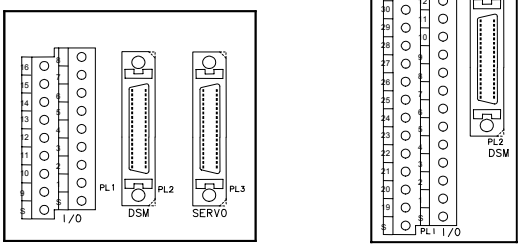
- **Pomocná svorkovnice, Katalogové číslo IC693ACC336 –** Tato svorkovnice obsahuje jeden 36-pinový konektor, který se připojuje k modulu DSM314. Tato svorkovnice má dvě základní aplikace (viz obrázky 3-10 a 3-11):

    1. U analogových serv se připojuje k DSM konektoru A, B, C nebo D, aby se umožnilo šroubové připojení analogového servozesilovače a I/O zařízení jiných výrobců. Viz obrázky 3-19 až 3-23.

    2. U pomocných os se připojuje k DSM konektoru B, C nebo D, aby se umožnilo šroubové připojení k externím zařízením, například čidla vzorkování, spínače nájezdu do výchozí polohy a spínače přejezdu. **Poznámka:** Viz obrázek 3-23

    - **Servo SL-Series ke svorkovnici APM/DSM, Katalogové číslo IC800SLT001** – Používá se k připojení DSM konektoru A, B, C nebo D k servozesilovači analogového rychlostního rozhraní GE Fanuc SL-Series a k vytvoření šroubových svorek pro připojení I/O zařízení. Obsahuje dva konektory. Jeden se připojuje k DSM modulu a druhý k servozesilovači SL-Series. Další informace najdete v *Uživatelském manuálu serva SL-Series*, GFK-1581.

Tabulka pro rychlý výběr svorkovnice DSM			
Použití DSM	**Konektor DSM**	**Režim osy DSM**	**Požadovaná svorkovnice**
Připojení k digitálnímu servu α Series nebo β-Series a I/O	A nebo B	Digitální	IC693ACC335
Připojení přímo k digitálnímu servu α Series nebo β-Series. Není nutné připojit žádné I/O.	A nebo B	Digitální	Žádná
Připojení k analogovému servu jiného výrobce a I/O	A, B, C nebo D	Analogový	IC693ACC336
Připojení k analogovému servu SL-Series a I/O	A, B, C nebo D	Analogové rychlostní rozhraní	IC800SLT001
Připojení k I/O pomocné osy na DSM konektoru B, C nebo D	B, C nebo D	Analogový nebo pomocný	IC693ACC336



IC693ACC335
Svorkovnice servoosy

IC693ACC336
Pomocná svorkovnice

Obrázek 3-3. Sestavy osy a pomocné svorkovnice

Poznámka: Každá svorkovnice se dodává s montážní lištou DIN. Instrukce pro převedení svorkovnice na montáž na panel jsou obsažené v této kapitole.

Svorkovnice digitální servoosy – IC693ACC335

Popis

Svorkovnice digitální servoosy IC693ACC335 se používá k připojení DSM314 k digitálním serzvozesilovačům GE Fanuc. Svorkovnice obsahuje dva 36-pinové konektory s označením **DSM** a **SERVO**. Kabel IC693CBL324 (1 metr) nebo IC693CBL325 (3 metry) se připojuje z konektoru **DSM** (PL2) na konektor A nebo B čelní desky DSM314. Kabel povelů serva IC800CBL001 (1 metr) nebo IC800CBL002 (3 metry) se připojuje z konektoru **SERVO** (PL3) na konektor JS1B na servozesilovači α Series nebo β Series GE Fanuc.

Na svorkovnici servoosy je osmnáct šroubových svorek pro připojení uživatelských zařízení. Tyto svorky mají následující přiřazení:

Tabulka 3-4. Přiřazení pinů svorkovnice servoosy IC693ACC335

Šroubová svorka I/O na svorkovnici servoosy	Pin čelní desky DSM314	Identifikátor obvodu	Typ obvodu	Funkce obvodu servoosy 1, 2	Název signálu (uvedena osy 1)*	Maximální napětí
1 9	1 19	IN1	Jednodrátové / diferenciální 5 V vstupy	Vzorkovací vstup 1 (+) Vzorkovací vstup 1 (-)	IN1P_A IN1M_A	5 V ss
2 10	2 20	IN2		Vzorkovací vstup 2 (+) Vzorkovací vstup 2 (-)	IN2P_A IN2M_A	5 V ss
3	4	P5V	Napájení 5 V	Napájení 5 V	P5V_A	5 V ss
11	22	0V	0 V	0 V	0V_A	5 V ss
6	16	IN9	24 V opticky oddělené vstupy	Přejetí (+)	IN9_A	30 V ss
14	34	IN10		Přejetí (-)	IN10_A	30 V ss
7	17	IN11		Spínač výchozí polohy	IN11_A	30 V ss
15	35	INCOM	Nulový vodič 24 V vstupu	Nulový vodič 24 V vstupu	INCOM_A	30 V ss
8 16	18 36	OUT1	24 V, 125 mA Výstup DC SSR	Výstup PLC 24 V (+) Výstup PLC 24 V (-)	OUT1P_A OUT1M_A	30 V ss
5 13	14 32	OUT3	Diferenciální výstup 5 V	výstup PLC 5 V (+) Výstup PLC 5 V (-)	OUT3P_A OUT3M_A	5 V ss
4	6	AOUT	Analogový výstup +/- 10 V	Analogový výstup PLC	AOUT_A	5 V ss
12	24	ACOM	Nulový vodič analogového výstupu	Nulový vodič analogového výstupu	ACOM_A	5 V ss
S (2 piny)		SHIELD	Stínění kabelu	Stínění kabelu	SHIELD_A	5 V ss

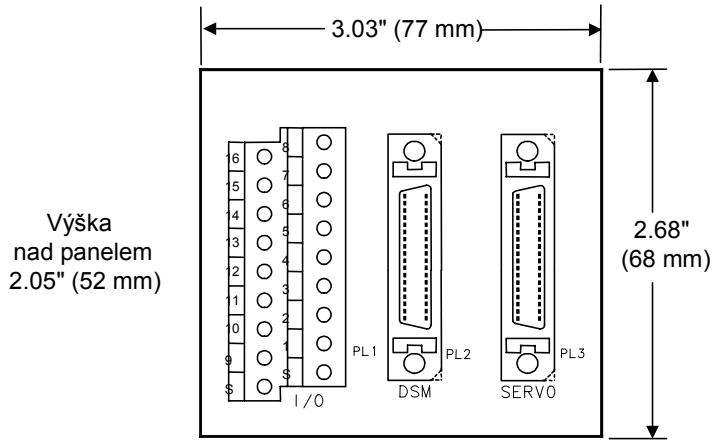
* **U názvů signálů týkajících se servoosy 2 změňte všechna "_A" na "_B".**

Mezi zvolené I/O body a stínění (kostra) je připojeno šest 130 V MOV pro potlačení šumu. Takto připojené I/O svorky jsou 6, 7, 8, 14, 15 a 16.

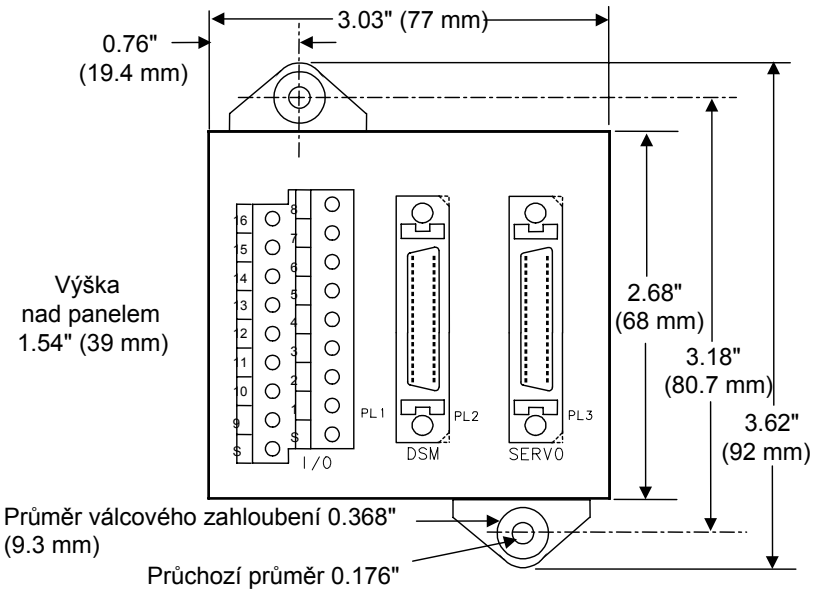
Do I/O svorek je možno zasunout dráty o velikosti 14-28 AWG. Maximální moment pro utahování šroubu je 5 palců-liber.

Poznámka: Dvě šroubové svorky mají označení S jako Stínění. Z jedné z těchto svorek S je nutno připojit krátký zemnící drát přímo na zem panelu. Stínění kabelu od uživatelských zařízení je nutno připojit k některé svorce **S**.

Montážní rozměry



Montáž na lištu



Montáž na panel

Obrázek 3-4. Montážní rozměry svorkovnice digitální osy IC693ACC335

Převod z montáže na lištu DIN na montáž na panel

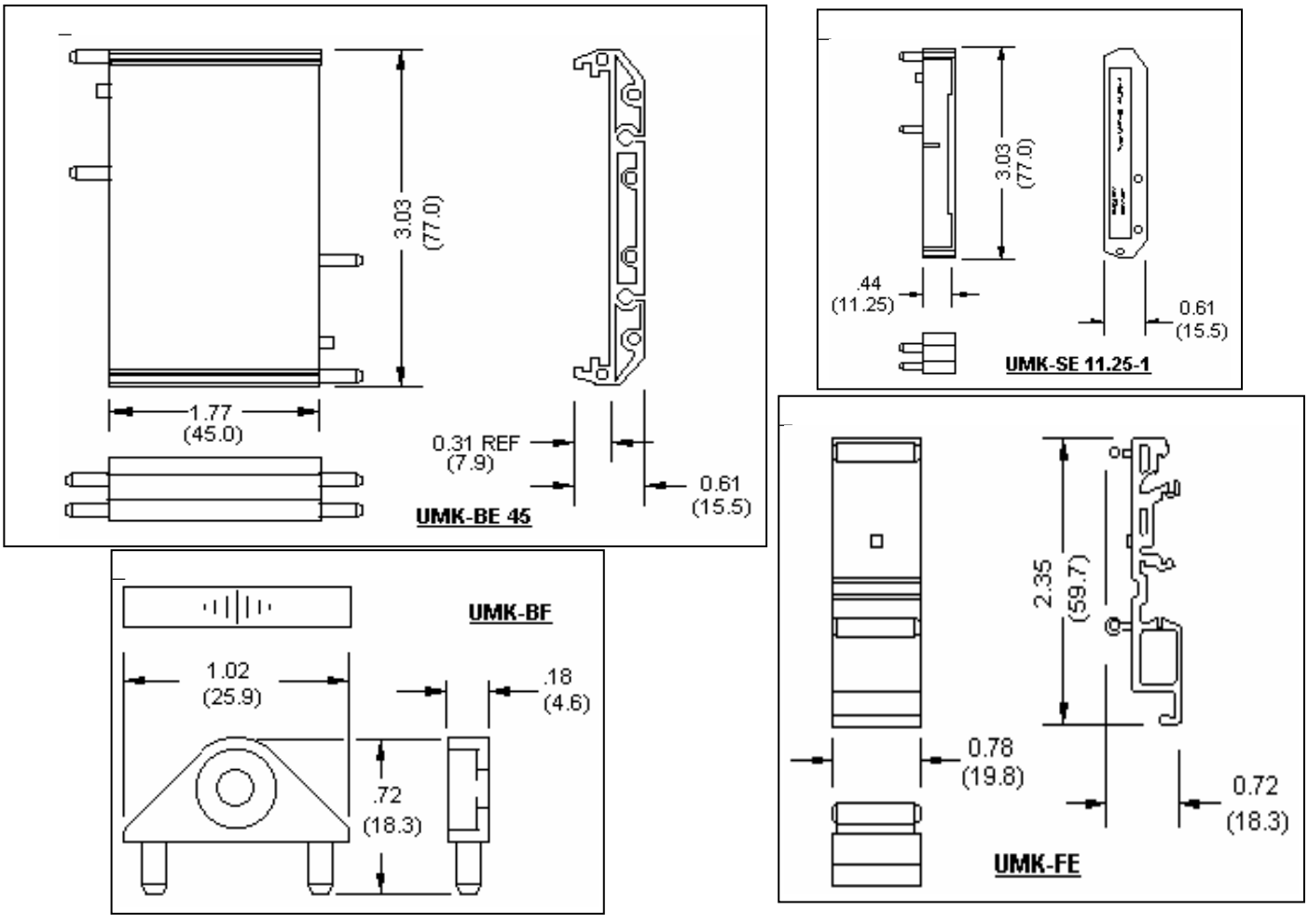
Při montáži na lištu DIN nebo na panel se používají následující díly. Svorkovnice osy se dodává již připravená pro montáž na lištu DIN. Instrukce v této části vám pomohou při převodu svorkovnice pro uspořádání pro montáž na panel.

Následující tabulka a výkres popisují různé plastové díly které tvoří sestavu svorkovnice, a ukazují boční pohled svorkovnice pro montáž na lištu DIN.

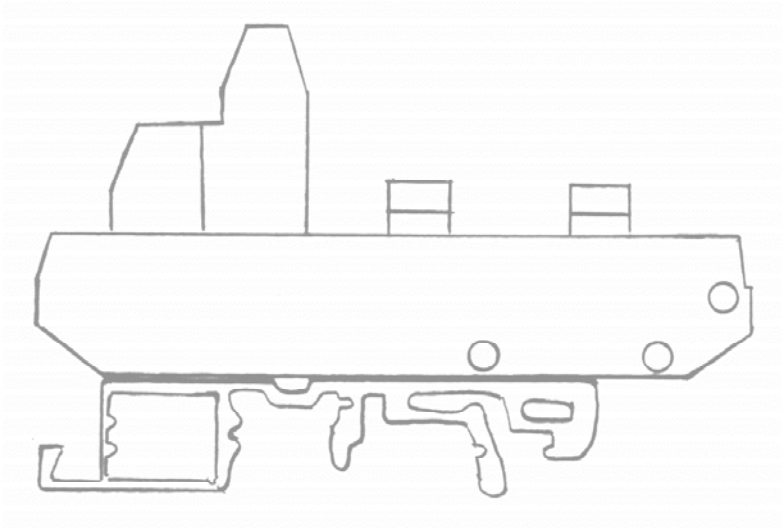
Tabulka 3-5. Komponenty sestavy svorkovnice osy

Číslo dílu plastového komponentu	Popis	Počet	Použitý způsob montáže
UMK-BE 45	Základní díl	1	DIN, Panel
UMK-SE 11.25-1	Boční díl	2	DIN, Panel
UMK-FE	Patkový díl	2	DIN
UMK-BF*	Montážní oko	2	Panel

* Díly dodávané se svorkovnicí osy pro volitelnou montáž na panel.



Obrázek 3-5. Výkresy sestavy svorkovnice digitální servoosy



Obrázek 3-6. Boční pohled sestavy svorkovnice digitální servoosy

Pro převod svorkovnice digitální servoosy pro montáž na panel je nutno použít následující postup. Nezapomeňte si uschovat všechny díly pro případný zpětný převod na montáž na lištu DIN.

1. Opatrně sundejte jeden boční prvek UMK-SE 11.25-1 ze základního prvku UMK-BE 45. Pokud použijete šroubovák nebo jiný nástroj, dávejte velký pozor, aby nedošlo k poškození některého plastového krytu nebo desky s obvody.
2. Vysuňte patkový díl UMK-FE ven ze základního dílu. Uschovejte si tento díl pro případný zpětný převod svorkovnice na montáž na lištu DIN.
3. Zacvakněte boční prvek, který jste sundali v kroku 1, zpátky na základní prvek.
4. Zasuňte jedno montážní oko UMK-BF do příslušných dvou děr v bočním prvku. Všimněte si, že montážní oko má zahloubenou díru pro následné vložení montážního šroubu (dodá si uživatel). Zahloubená díra musí být směrem nahoru, aby do ní bylo možno zasunout montážní šroub.
5. Kroky 1-4 výše zopakujte pro druhou stranu svorkovnice.

Pomocná svorkovnice – IC693ACC336

Popis a montážní rozměry

Pomocná svorkovnice IC693ACC336 se používá k připojení DSM314 k analogovým servoosám a pomocným zařízením, jako například inkrementální snímače polohy s fázovým posunutím, detektory vzorkování a externí přepínače. Svorkovnice obsahuje jeden 36-pinový konektor s označením **DSM**. Kabel IC693CBL324 (1 metr) nebo IC693CBL325 (3 metry) se připojuje z konektoru **DSM** (PL2) k čelní desce DSM314.

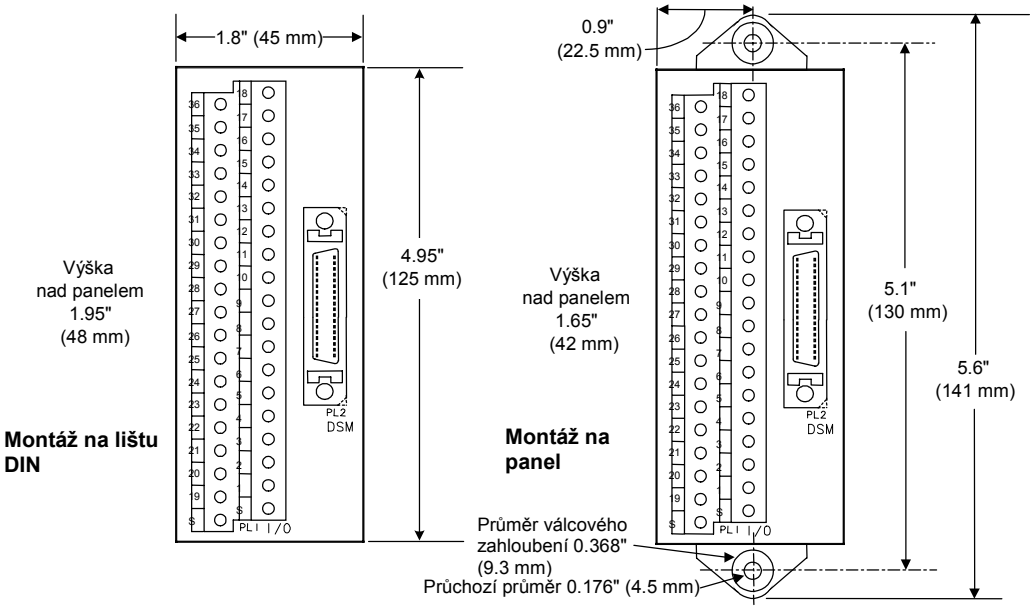
Na pomocné svorkovnici je třicet dva šroubových svorek pro připojení uživatelských zařízení. Tyto šroubové svorky mají stejné označení pinů jako 36-pinový konektor na čelní desce DSM314. Podrobné informace o připojení najdete v kapitole 3.

Maximální napětí, které je možno připojit na I/O svorky 16-18 a 34-36, je 30 V ss. Maximální napětí pro každou jinou vstupní svorku je 5 V ss.

Mezi zvolené I/O body a stínění (kostra) je připojeno šest 130 V MOV pro potlačení šumu. Takto připojené I/O svorky jsou 16, 17, 18, 34, 35 a 36.

Do I/O svorek je možno zasunout dráty o velikosti 14-28 AWG. Maximální moment, který **je možno použít, je 5 palců-liber.**

Poznámka: Dvě šroubové svorky mají označení **S** jako **Stínění**. Z jedné z těchto svorek **S** je nutno připojit krátký zemnící drát přímo na zem panelu. Stínění kabelu od uživatelských zařízení je nutno připojit k některé svorce **S**.



Obrázek 3-7. Montážní rozměry svorkovnice IC693ACC336

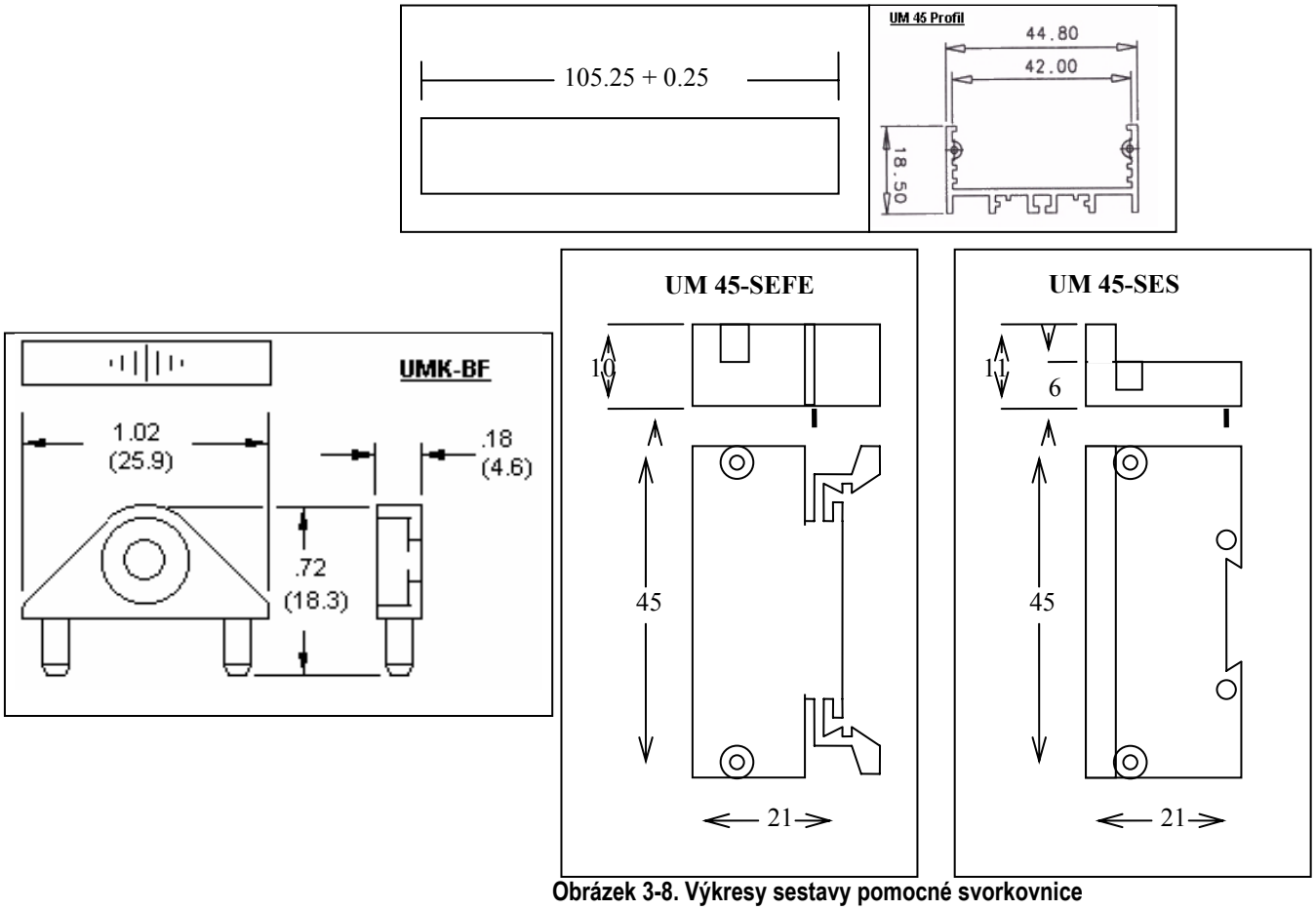
Převod z montáže na lištu DIN na montáž na panel

Při montáži na lištu DIN nebo na panel se používají následující díly. Pomocná svorkovnice se dodává již připravená pro montáž na lištu DIN. Instrukce v této části vám pomohou při převodu svorkovnice pro uspořádání pro montáž na panel.

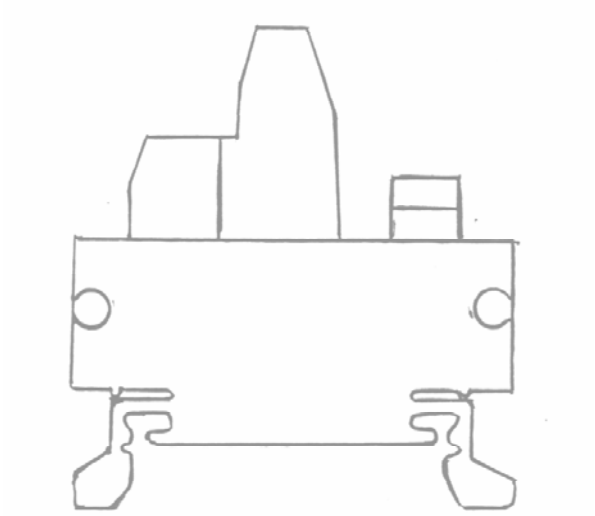
Následující tabulka a výkres popisují různé plastové díly které tvoří sestavu pomocné svorkovnice, a ukazují boční pohled svorkovnice pro montáž na lištu DIN.

Tabulka 3-6. Komponenty pomocné svorkovnice

Číslo dílu kontaktu Phoenix	Popis	Počet
UM45 Profil 105.25	Nosič desky plošného spoje	1
UM 45-SEFE se 2 šrouby	Boční prvek s paticí	2
UMK 45-SES se 2 šrouby*	Boční díl	2
UMK-BF*	Montážní oko	2
* Díly dodávané s pomocnou svorkovnicí pro volitelnou montáž na panel.		



Obrázek 3-8. Výkresy sestavy pomocné svorkovnice



Obrázek 3-9. Boční pohled sestavy pomocné svorkovnice

Pokud budete chtít montovat pomocnou svorkovnici přímo na panel místo na lištu DIN, je nutno dodržet následující postup. Nezapomeňte si uschovat všechny díly pro případný zpětný převod na montáž na lištu DIN.

1. Pomocí malého křížového šroubováku opatrně vyšroubujte dva šrouby, které drží jeden boční prvek UM-45 SEFE s patkou na nosiči desky plošného spoje UM 45 Profil. Uschovejte si tento díl pro případný zpětný převod svorkovnice na montáž na lištu DIN.
2. Připevněte jeden boční díl UMK 45-SES k nosiči desky plošného spoje místo bočnice odstraněné v kroku 1 výše opět pomocí těchto dvou šroubů. Dejte pozor, abyste šrouby neutáhli příliš velkou silou.
3. Zasuňte jedno montážní oko UMK-BF do příslušných dvou děr v bočním prvku. Všimněte si, že montážní oko má zahloubenou díru pro následné vložení montážního šroubu (dodá si uživatel). Zahloubená díra musí být směrem nahoru, aby do ní bylo možno zasunout montážní šroub.
4. Kroky 1-3 výše zopakujte pro druhou stranu svorkovnice.

Kabely

Pro DSM314 je možno použít pět kabelů:

Tabulka 3-7. Kabely pro DSM314

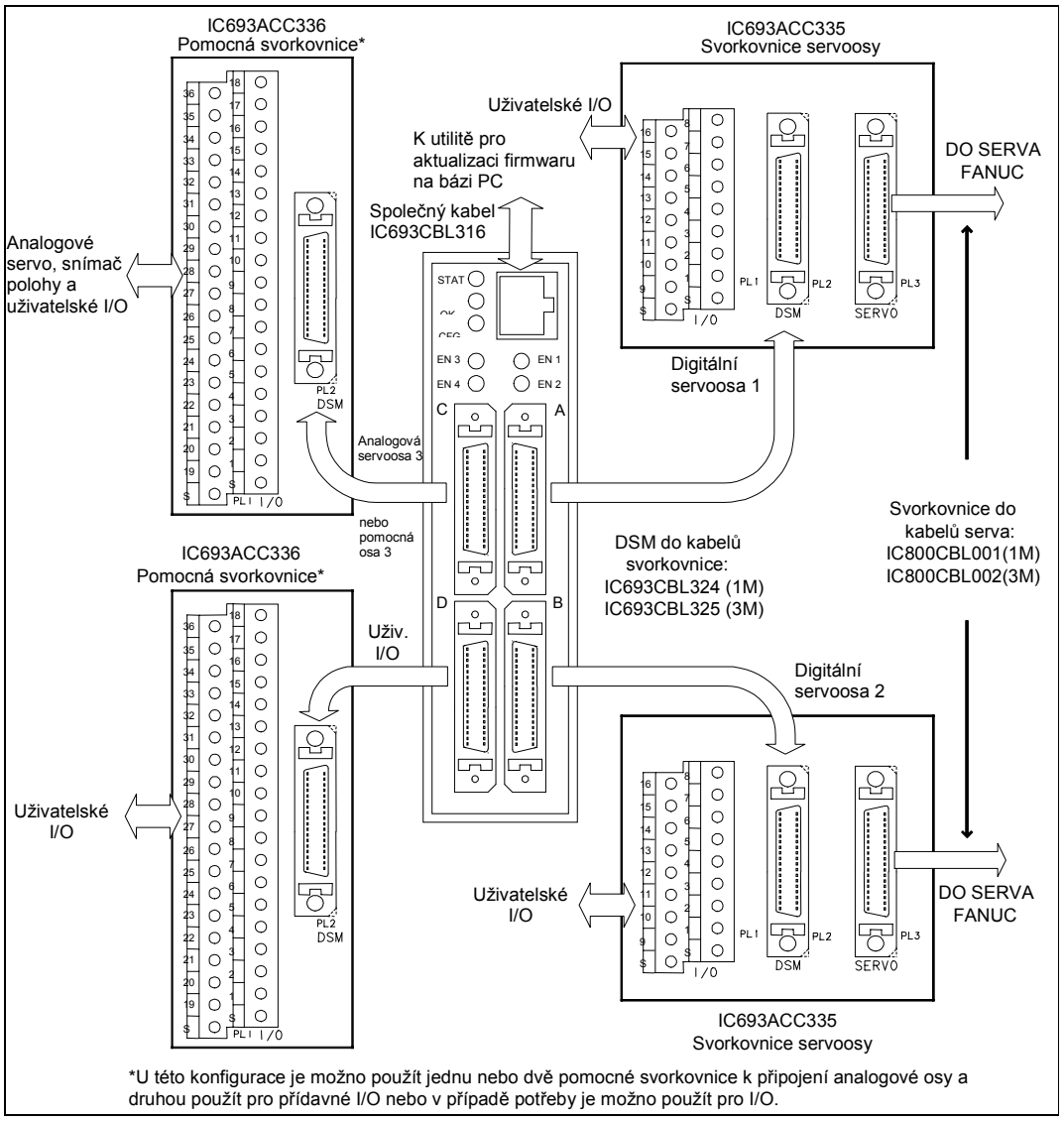
Kabel	Popis	Délka	Aplikace
IC693CBL316	Kabel Station Manager	1 metr	DSM314 Comm pro upgrade firmwaru
IC693CBL324	Kabel pro připojení svorkovnice	1 metr	DSM314 na svorkovnici servoosy nebo pomocnou svorkovnici
IC693CBL325	Kabel pro připojení svorkovnice	3 metry	DSM314 na svorkovnici servoosy nebo pomocnou svorkovnici
IC800CBL001	Kabel povelů digitálního serva	1 metr	Svorkovnice digitální servoosy nebo DSM na digitální servozesilovač
IC800CBL002	Kabel povelů digitálního serva	3 metry	Svorkovnice digitální servoosy nebo DSM na digitální servozesilovač

Pokud budete potřebovat kabely zákaznické svorkovnice a serva ve větších délkách, spojte se se svým distributorem GE Fanuc. Maximální doporučené délka kabelu pro konektor DSM na servozesilovač α a β Series je 50 metrů.

Kabely používají speciální stínění a konstrukci pro zajištění spolehlivé činnosti serva. GE Fanuc doporučuje uživatelům, aby se nesnažili provádět nějaké změny na kabelech nebo konektorech.

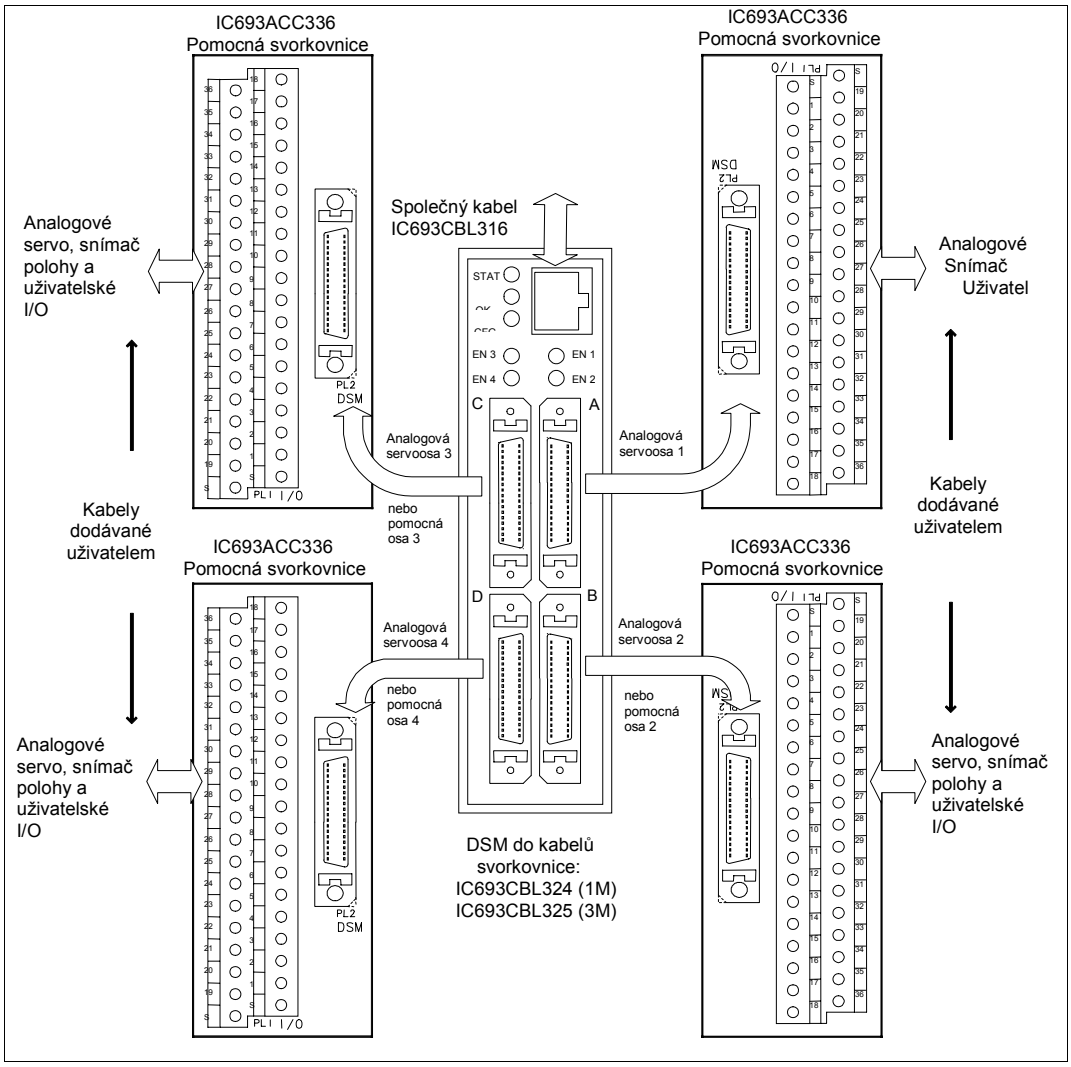
Poznámka: Pokud digitální servoosa nepoužívá žádné zařízení, které se normálně připojuje ke šroubovým svorkám svorkovnice digitálního serva IC693ACC335, svorkovnice a kabel svorkovnice IC693CBL324/325 nejsou zapotřebí. Místo toho je možno kabel povelů digitálního serva IC800CBL001/002 připojit přímo z digitálního zesilovače na konektor A nebo B čelní desky DSM314. Když toto provedete, konfigurační parametr ***OT Limit Sw*** je nutno v konfiguračním softwaru nastavit na **Zakázáno**, jinak DSM nebude pracovat.

Následující obrázek znázorňuje svorkovnici digitální servoosy a kabely související s DSM314.



Obrázek 3-10. Svorkovnice a konektory digitálního serva DSM314

Následující obrázek znázorňuje svorkovnici analogového serva a kabely související s DSM314.



Obrázek 3-11. Svorkovnice a konektory DSM314 pro analogová serva jiných výrobců (Serva SL viz GFK-1581)

Uzemnění I/O kabelu

Řádné vedení signálových kabelů, kabelů napájení serva a kabelů napájení motoru a instalace s řádným uzemněním třídy 3 zajistí spolehlivou činnost. Uzemnění třídy 3 obvykle předepisuje zemnící vodič s minimálním průměrem větším než je průměr vodiče přívodního napájení připojený k zemi přes maximální odpor 100 ohmů. Postupujte podle místních elektrických norem a instalaci proveďte v souladu s místními předpisy.

Specifikace pro kompletaci instalace a zapojení digitálního servozesilovače α a β Series včetně uzemnění zesilovače jsou kompletně popsané v manuálu *GFH-001, Průvodce specifikacemi produktů serva.*

Při vedení signálových vodičů, vodičů pro napájení zesilovače a napájení motoru je nutno signálové vodiče oddělit od napájecích vodičů. Následující tabulka uvádí, jak tyto kabely oddělit.

Tabulka 3-8. Oddělení signálových vodičů

Skupina	Signál	Akce
A	Vstupní napájení zesilovače Napájení motoru Cívka ovládání hlavního řídicího stykače (MCR). MCC připojí vstupní napájení zesilovače.	Ponechejte minimálně 10 cm od signálů skupiny “B” tak, že je svážete dohromady nebo použijte elektromagnetické stínění (uzemněné ocelové desky). Použijte ochranu proti šumu na MCC.
B	Kabel z DSM na svorkovnici osy Kabel svorkovnice osy do zesilovače Kabel z DSM na pomocnou svorkovnici Kabel zpětné vazby snímače polohy	Ponechejte minimálně 10 cm od signálů skupiny “A” tak, že je svážete dohromady nebo použijte elektromagnetické stínění (uzemněné ocelové desky). Použijte všechna požadovaná uzemnění stínění kabelů a připojení zemnících tyčí.

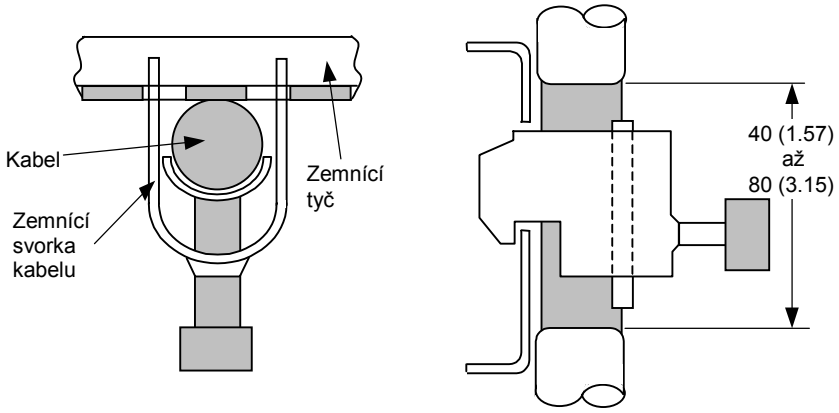
Z DSM na digitální servozesilovač α nebo β Series – uzemnění signálového kabelu

Signálové kabely používané s DSM314 obsahují stínění, která je nutno řádně uzemnit, aby byla zaručena spolehlivá činnost. Následující obrázek ukazuje doporučení pro kabel uzemnění pro typické instalace. Je nutno vzít v úvahu následující body:

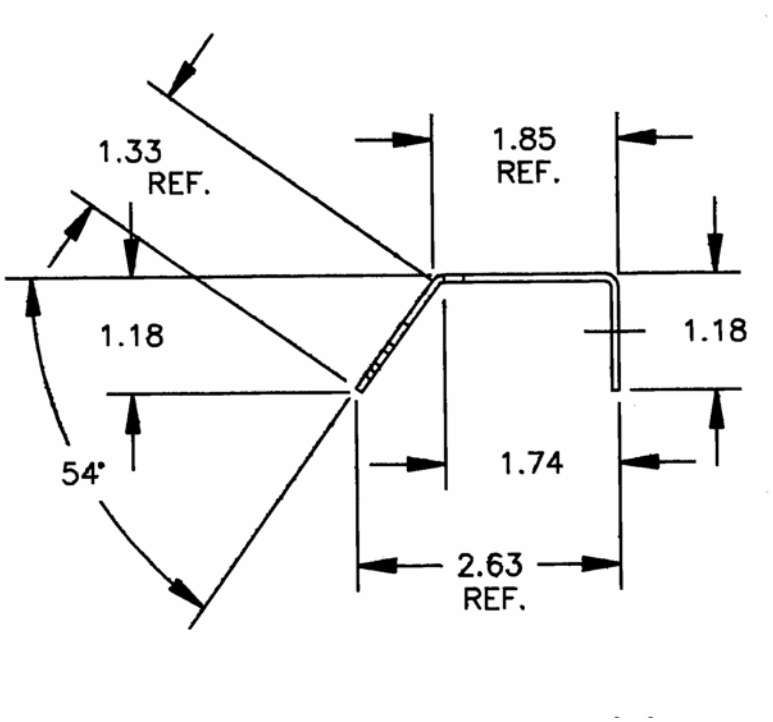
1. Zemnící vodič čelní desky DSM314 musí být připojený ke spolehlivé zemi panelu.
2. Svorkovnice digitální servoosy a pomocná svorkovnice mají dvě šroubové svorky s označením **S**. Z jedné z těchto svorek **S** je nutno připojit krátký zemnící drát, aby bylo zaručeno spolehlivé uzemnění panelu.

Kabel zpětné vazby snímače polohy digitálního servozesilovače α a β Series vždy vyžaduje svorku pro uzemnění stínění kabelu A99L-0035-0001 a jednu z 11 použitelných pozic na zemnící tyči 44B295864-001 na kabelu na straně zesilovače. Toto uspořádání se sponou slouží jako odlehčení

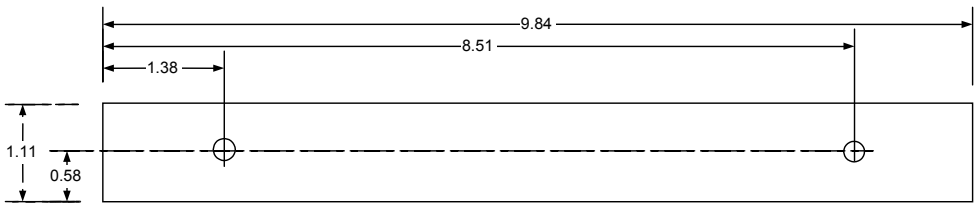
mechanického pnutí a jako uzemnění stínění kabelu. Vnější izolaci kabelu digitálního servozesilovače je nutno odstranit a obnažit tak stínění kabelu v místě styku se sponou.



Obrázek 3-12. Detaily spony pro uzemnění kabelu A99L-0035-0001



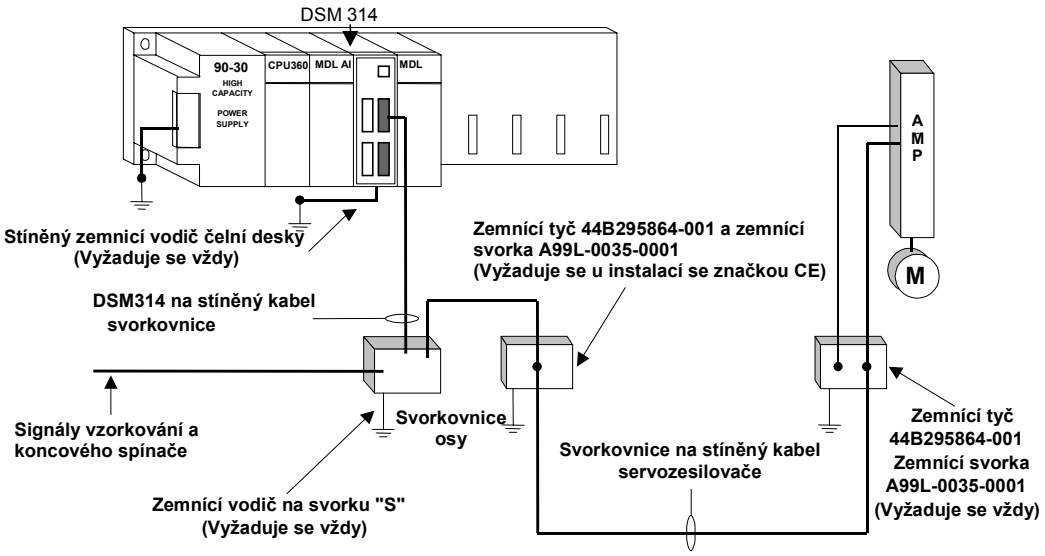
Obrázek 3-13. . Zemnící tyč 44B295864-001, rozměry při bočním pohledu



Obrázek 3-14. Rozměry zemnící tyče 44B295864-001, pohled zezadu se znázorněním montážních děr

3. U instalací, které musí splňovat normy IEC na odolnost proti elektrickému šumu, je nutno u kabelu servozesilovače IC800CBL001/002 na straně svorkovnice digitální servoosy použít zemnící sponu stínění kabelu A99L-0035-0001 a jednu z 11 použitelných pozic na zemnící tyči 44B295864-001. Pokud kabel digitálního servozesilovače bude připojený přímo k čelní desce DSM314 (nepoužívá se žádná svorkovnice digitální servoosy), zemnící spona a tyč se na kabelu na straně čelní desky nevyžadují.

Další informace najdete v *Požadavky na instalaci pro vyhovění normám*, GFK-1179.



Obrázek 3-15. Uzemnění I/O kabelu DSM314

Identifikátory I/O obvodu a názvy signálů

Identifikátory I/O obvodů představují důslednou metodu pojmenovávání I/O obvodů. Například IN1 se vztahuje k prvnímu ze tří diferenciálních / jednodrátových 5 V vstupů každé osy.

Názvy signálů jsou přiřazené identifikátorům obvodu pro každou osu. *Název signálu se skládá z identifikátoru obvodu, za kterým následuje přípona A-D jako identifikace konektoru osy*. Diferenciální obvody také mají příponu P (kladný) a M (záporný) jako identifikaci signálu (+) a (-) pro každý diferenciální pár.

Příklad: OUT2 je identifikátor obvodu pro první diferenciální 5 V výstup na každém konektoru. Názvy signálů spojené s obvodem OUT2 jsou:

Tabulka 3-9. Názvy signálů spojené s OUT2

Osa:	**Osa 1**	**Osa 2**	**Osa 3**	**Osa 4**
Konektor:	A	B	C	D
(+) Výstupní signál:	OUT2P_A	OUT2P_B	OUT2P_C	OUT2P_D
(-) Výstupní signál:	OUT2M_A	OUT2M_B	OUT2M_C	OUT2M_D

Funkce I/O obvodu a přiřazení pinů

Následující tři tabulky uvádějí funkční přiřazení I/O obvodu a přiřazení konektoru a pinů svorkovnice pro jednotlivé konektory osy. I když každý konektor má stejné I/O obvody, funkční přiřazení I/O obvodů závisí na ose:

Tabulka 3-10. Přiřazení a funkce konektoru osy

Konektor	**Číslo osy**	**Typ osy**	**Použití I/O**
A	1	Servoosa	Řízení digitálního/analogového serva v uzavřené smyčce a uživatelské I/O
B	2	Servoosa	Řízení digitálního/analogového serva v uzavřené smyčce nebo pomocného analogového a digitálního I/O
C	3	Servoosa	Řízení analogového serva v uzavřené smyčce nebo pomocného analogového a digitálního I/O
D	4	Servoosa	Řízení analogového serva v uzavřené smyčce nebo pomocného analogového a digitálního I/O

Obvod digitální servoosy 1, 2 a přiřazení pinů

Tato tabulka identifikuje všechny obvody a přiřazení pinů pro digitální servoosu 1 a digitální servoosu 2. *Stínovaná pole označují signály, které jsou kabelem připojené k servozesilovači a nelze je použít pro uživatelská spojení.*

Tabulka 3-11. Obvod a přiřazení pinů pro digitální servoosu 1 a digitální servoosu 2

Identifikátor obvodu	Typ obvodu	Funkce obvodu digitální servoosy 1, 2	Název signálu osy 1	Název signálu osy 2	Pin čelní desky	Pin na svorkovnici osy
IN1	Jednodrátové / diferenciální 5 V vstupy	Vzorkovací vstup 1 (+) Vzorkovací vstup 1 (-)	IN1P_A IN1M_A	IN1P_B IN1M_B	1 19	1 9
IN2		Vzorkovací vstup 2 (+) Vzorkovací vstup 2 (-)	IN2P_A IN2M_A	IN2P_B IN2M_B	2 20	2 10
IN3		Data sériového snímače polohy (+) Data sériového snímače polohy (-)	IN3P_A IN3M_A	IN3P_B IN3M_B	3 21	
P5V	Napájení 5 V	Napájení 5 V	P5V_A	P5V_B	4	3
0 V	0 V	0 V	0V_A	0V_B	22,23	11
IN4	Jednodrátový vstup 5 V	Vstup servo připraveno	IN4_A	IN4_B	5	
IO5	Jednodrátové 5 V vstupy/výstupy	Servo PWM / Alarm	IO5_A	IO5_B	9	
IO6		Servo PWM / Alarm	IO6_A	IO6_B	10	
IO7		Servo PWM / Alarm	IO7_A	IO7_B	11	
IO8		Servo ENBL / Alarm	IO8_A	IO8_B	12	
0 V	0 V	0 V	0V_A	0V_B	27-30	
IN9	24 V opticky oddělené vstupy	Přejetí (+)	IN9_A	IN9_B	16	6
IN10		Přejetí (-)	IN10_A	IN10_B	34	14
IN11		Spínač výchozí polohy	IN11_A	IN11_B	17	7
INCOM	Nulový vodič 24 V vstupu	Nulový vodič 24 V vstupu	INCOM_A	INCOM_B	35	15
OUT1	24 V, 125 mA Výstup DC SSR	výstup PLC 24 V (+) Výstup PLC 24 V (-)	OUT1P_A OUT1M_A	OUT1P_B OUT1M_B	18 36	8 16
OUT2	Diferenciální 5 V výstupy	Požadavek sériového snímače polohy (+) Požadavek sériového snímače polohy (-)	OUT2P_A OUT2M_A	OUT2P_B OUT2M_B	13 31	
OUT3		Výstup PLC 5 V (+) Výstup PLC 5 V (-)	OUT3PA OUT3M_A	OUT3P_B OUT3M_B	14 32	5 13
ENBL	24 V, 30 mA výstup SSR	Servo MCON (+) Servo MCON 0 V	ENBL1_A ENBL2_A	ENBL1_B ENBL2_B	15 33	
AIN1	Diferenciální +/- 10 V Analogové vstupy	Proud IR fází (+) Proud IR fází (-)	AIN1P_A AIN1M_A	AIN1P_B AIN1M_B	7 25	
AIN2		Proud IS fází (+) Proud IS fází (-)	AIN2P_A AIN2M_A	AIN2P_B AIN2M_B	8 26	
AOUT1	Analogový výstup +/- 10 V	Analogový výstup PLC	AOUT_A	AOUT_B	6	4
ACOM	Nulový vodič analogového výstupu	Nulový vodič analogového výstupu	ACOM_A	ACOM_B	24	12
SHIELD	Stínění kabelu	Stínění kabelu	SHIELD_A	SHIELD_B		S

Obvod analogové servoosy 1 - 4 a přiřazení pinů

Tato tabulka identifikuje všechny obvody a přiřazení pro analogovou servoosu 1 – analogovou servoosu 4. *Stínovaná pole označují signály, které se nepoužívají a nelze je použít pro uživatelská spojení.*

Tabulka 3-12. Obvod a přiřazení pinů pro analogovou servoosu 1 až analogovou servoosu 4

Identifikátor obvodu	Typ obvodu	Funkce obvodu analogové servoosy 1 - 4	Název signálu osy 1	Název signálu osy 2	Název signálu osy 3	Název signálu osy 4	Pin čelní desky	Pin na pomocné svorkovnici
IN1	Jednodrátové / diferenciální 5 V vstupy	Kanál A snímače polohy (+)	IN1P_A	IN1P_B	IN1P_C	IN1P_D	1	1
		Kanál A snímače polohy (-)	IN1M_A	IN1M_B	IN1M_C	IN1M_D	19	19
IN2		Kanál B snímače polohy (+)	IN2P_A	IN2P_B	IN2P_C	IN2P_D	2	2
		Kanál B snímače polohy (-)	IN2M_A	IN2M_B	IN2M_C	IN2M_D	20	20
IN3		Nulový puls snímače polohy (+)	IN3P_A	IN3P_B	IN3P_C	IN3P_D	3	3
		Nulový puls snímače polohy (-)	IN3M_A	IN3M_B	IN3M_C	IN3M_D	21	21
P5V	Napájení 5 V	Napájení snímače polohy 5 V	P5V_A	P5V_B	P5V_C	P5V_D	4	4
0 V	0 V	0 V	0V_A	0V_B	0V_C	0V_D	22,23	22,23
IN4	Jednodrátový vstup 5 V	Vstup servo připraveno	IN4_A	IN4_B	IN4_C	IN4_D	5	5
IO5	Jednodrátové 5 V vstupy/výstupy	Vstup vzorkování 1	IO5_A	IO5_B	IO5_C	IO5_D	9	9
IO6		Vstup vzorkování 2	IO6_A	IO6_B	IO6_C	IO6_D	10	10
IO7		Nepoužívá se	IO7_A	IO7_B	IO7_C	IO7_D	11	11
IO8		Nepoužívá se	IO8_A	IO8_B	IO8_C	IO8_D	12	12
0 V	0 V	0 V	0V_A	0V_B	0V_C	0V_D	27-30	27-30
IN9	24 V opticky oddělené vstupy	Přejetí (+)	IN9_A	IN9_B	IN9_C	IN9_D	16	16
IN10		Přejetí (-)	IN10_A	IN10_B	IN10_C	IN10_D	34	34
IN11		Spínač výchozí polohy	IN11_A	IN11_B	IN11_C	IN11_D	17	17
INCOM	Nulový vodič 24 V vstupu	Nulový vodič 24 V vstupu	INCOM_A	INCOM_B	INCOM_C	INCOM_D	35	35
OUT1	24 V, 125 mA Výstup DC SSR	Výstup PLC 24 V (+)	OUT1P_A	OUT1P_B	OUT1P_C	OUT1P_D	18	18
		Výstup PLC 24 V (-)	OUT1M_A	OUT1M_B	OUT1M_C	OUT1M_D	36	36
OUT2	Diferenciální 5 V výstupy	Nepoužívá se	OUT2P_A	OUT2P_B	OUT2P_C	OUT2P_D	13	13
		Nepoužívá se	OUT2M_A	OUT2M_B	OUT2M_C	OUT2M_D	31	31
OUT3		Výstup PLC 5 V (+)	OUT3P_A	OUT3P_B	OUT3P_C	OUT3P_D	14	14
		Výstup PLC 5 V (-)	OUT3M_A	OUT3M_B	OUT3M_C	OUT3M_D	32	32
ENBL	24 V, 30 mA výstup SSR	Povolení serva (+)	ENBL1_A	ENBL1_B	ENBL1_C	ENBL1_D	15	15
		Povolení serva (-)	ENBL2_A	ENBL2_B	ENBL2_C	ENBL2_D	33	33
AIN1	Diferenciální +/- 10 V Analogové vstupy	Analogový vstup PLC (+)	AIN1P_A	AIN1P_B	AIN1P_C	AIN1P_D	7	7
		Analogový vstup PLC (-)	AIN1M_A	AIN1M_B	AIN1M_C	AIN1M_D	25	25
AIN2		Analogový vstup PLC (+)	AIN2P_A	AIN2P_B	AIN2P_C	AIN2P_D	8	8
		Analogový vstup PLC (-)	AIN2M_A	AIN2M_B	AIN2M_C	AIN2M_D	26	26
AOUT1	Analogový výstup +/- 10 V	Povel rychlosti serva (+) nebo povel kroutícího momentu serva (+)	AOUT_A	AOUT_B	AOUT_C	AOUT_D	6	6
ACOM	Nulový vodič analogového výstupu	Povel rychlosti serva (-) nebo povel kroutícího momentu serva (-)	ACOM_A	ACOM_B	ACOM_C	ACOM_D	24	24
SHIELD	Stínění kabelu	Stínění kabelu	SHIELD_A	SHIELD_B	SHIELD_C	SHIELD_D		S

Obvod pomocné osy 2 - 4 a přiřazení pinů

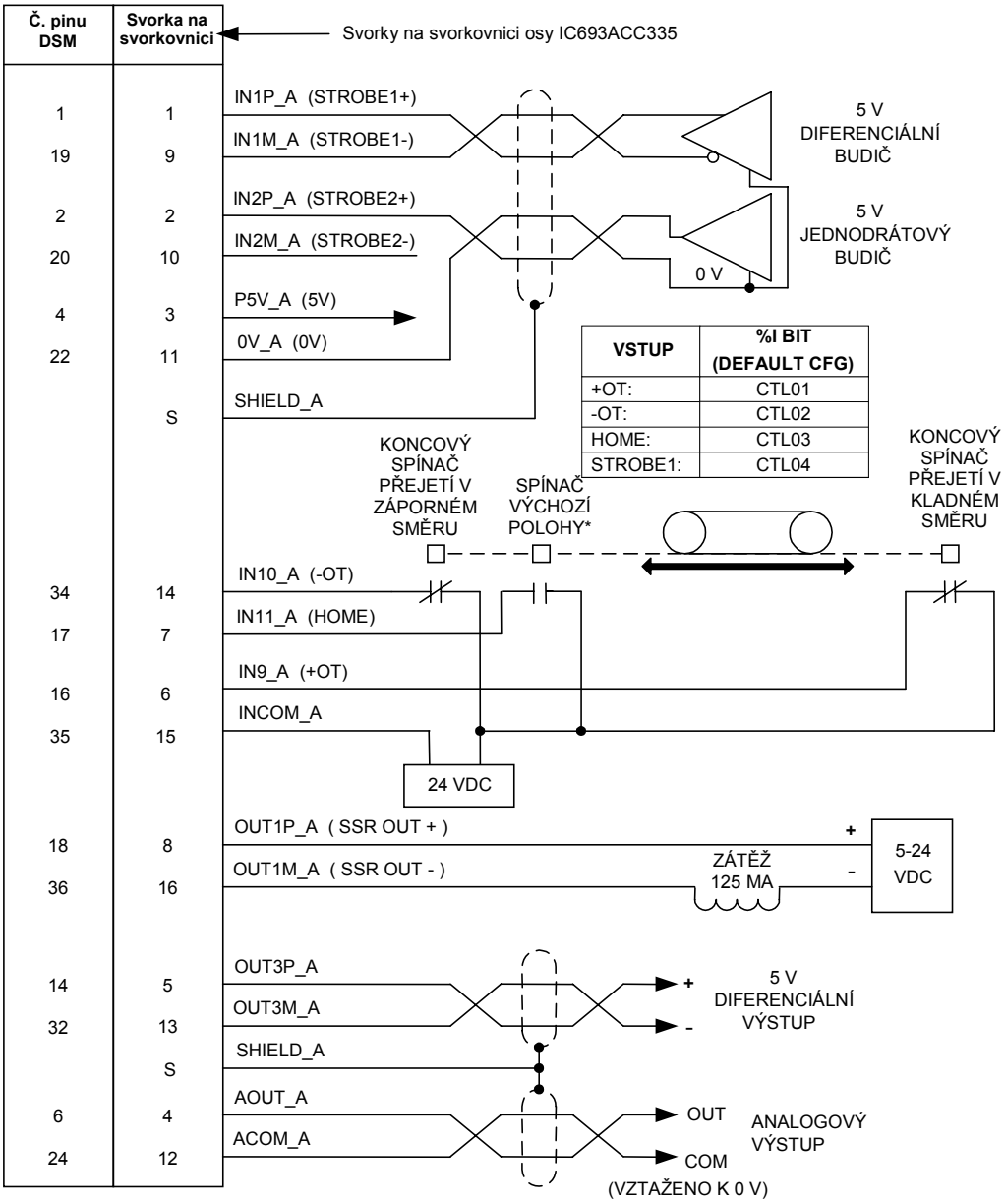
Tato tabulka identifikuje všechny obvody a přiřazení pro pomocnou osu 2 - pomocnou osu 4. *Stínovaná pole označují signály, které se nepoužívají a nelze je použít pro uživatelská spojení.*

Tabulka 3-13. Obvod a přiřazení pinů pro pomocnou osu 3 (konektor C)

Identifikátor obvodu	Typ obvodu	Funkce obvodu pomocné osy 2-4	Název signálu osy 2	Název signálu osy 3	Název signálu osy 4	Pin čelní desky	Špička na pomocné svorkovnici
IN1	Jednodrátový / diferenciální vstup 5 V	Kanál A snímače polohy (+) Kanál A snímače polohy (-)	IN1P_B IN1M_B	IN1P_C IN1M_C	IN1P_D IN1M_D	1 19	1 19
IN2		Kanál B snímače polohy (+) Kanál B snímače polohy (-)	IN2P_B IN2M_B	IN2P_C IN2M_C	IN2P_D IN2M_D	2 20	2 20
IN3		Nulový puls snímače polohy (+) Nulový puls snímače polohy (-)	IN3P_B IN3M_B	IN3P_C IN3M_C	IN3P_D IN3M_D	3 21	3 21
P5V	5 V z PLC	Napájení snímače polohy 5 V	P5V_B	P5V_C	P5V_D	4	4
0 V	0 V	0 V	0V_B	0V_C	0V_D	22,23	22,23
IN4	Jednodrátový vstup 5 V	Vstup PLC 5 V	IN4_B	IN4_C	IN4_D	5	5
IO5	Jednodrátové 5 V vstupy/výstupy	Vstup vzorkování 1	IO5_B	IO5_C	IO5_D	9	9
IO6		Vstup vzorkování 2	IO6_B	IO6_C	IO6_D	10	10
IO7		Nepoužívá se	IO7_B	IO7_C	IO7_D	11	11
IO8		Nepoužívá se	IO8_B	IO8_C	IO8_D	12	12
0 V	0 V	0 V	0V_B	0V_C	0V_D	27-30	27-30
IN9	24 V opticky oddělené vstupy	Vstup PLC 24 V	IN9_B	IN9_C	IN9_D	16	16
IN10		Vstup PLC 24 V	IN10_B	IN10_C	IN10_D	34	34
IN11		Spínač výchozí polohy	IN11_B	IN11_C	IN11_D	17	17
INCOM	Nulový vodič 24 V vstupu	Nulový vodič 24 V vstupu	INCOM_B	INCOM_C	INCOM_D	35	35
OUT1	24 V, 125 mA Výstup DC SSR	výstup PLC 24 V (+) Výstup PLC 24 V (-)	OUT1P_B OUT1M_B	OUT1P_C OUT1M_C	OUT1P_D OUT1M_D	18 36	18 36
OUT2	Diferenciální 5 V výstupy	Nepoužívá se Nepoužívá se	OUT2P_B OUT2M_B	OUT2P_C OUT2M_C	OUT2P_D OUT2M_D	13 31	13 31
OUT3		výstup PLC 5 V (+) Výstup PLC 5 V (-)	OUT3P_B OUT3M_B	OUT3P_C OUT3M_C	OUT3P_D OUT3M_D	14 32	14 32
ENBL	24 V, 30 mA výstup SSR	ON, když vynucený analogový výstup povelu %AQ je aktivní	ENBL1_B ENBL2_B	ENBL1_C ENBL2_C	ENBL1_D ENBL2_D	15 33	15 33
AIN1	Diferenciální +/- 10 V Analogové vstupy	Analogový vstup PLC (+) Analogový vstup PLC (-)	AIN1P_B AIN1M_B	AIN1P_C AIN1M_C	AIN1P_D AIN1M_D	7 25	7 25
AIN2		Analogový vstup PLC (+) Analogový vstup PLC (-)	AIN2P_B AIN2M_B	AIN2P_C AIN2M_C	AIN2P_D AIN2M_D	8 26	8 26
AOUT1	Analogový výstup +/- 10 V	Analogový výstup PLC	AOUT_B	AOUT_C	AOUT_D	6	6
ACOM	Nulový vodič analogového výstupu	Nulový vodič analogového výstupu	ACOM_B	ACOM_C	ACOM_D	24	24
SHIELD	Stínění kabelu	Stínění kabelu	SHIELD_B	SHIELD_C	SHIELD_D		S

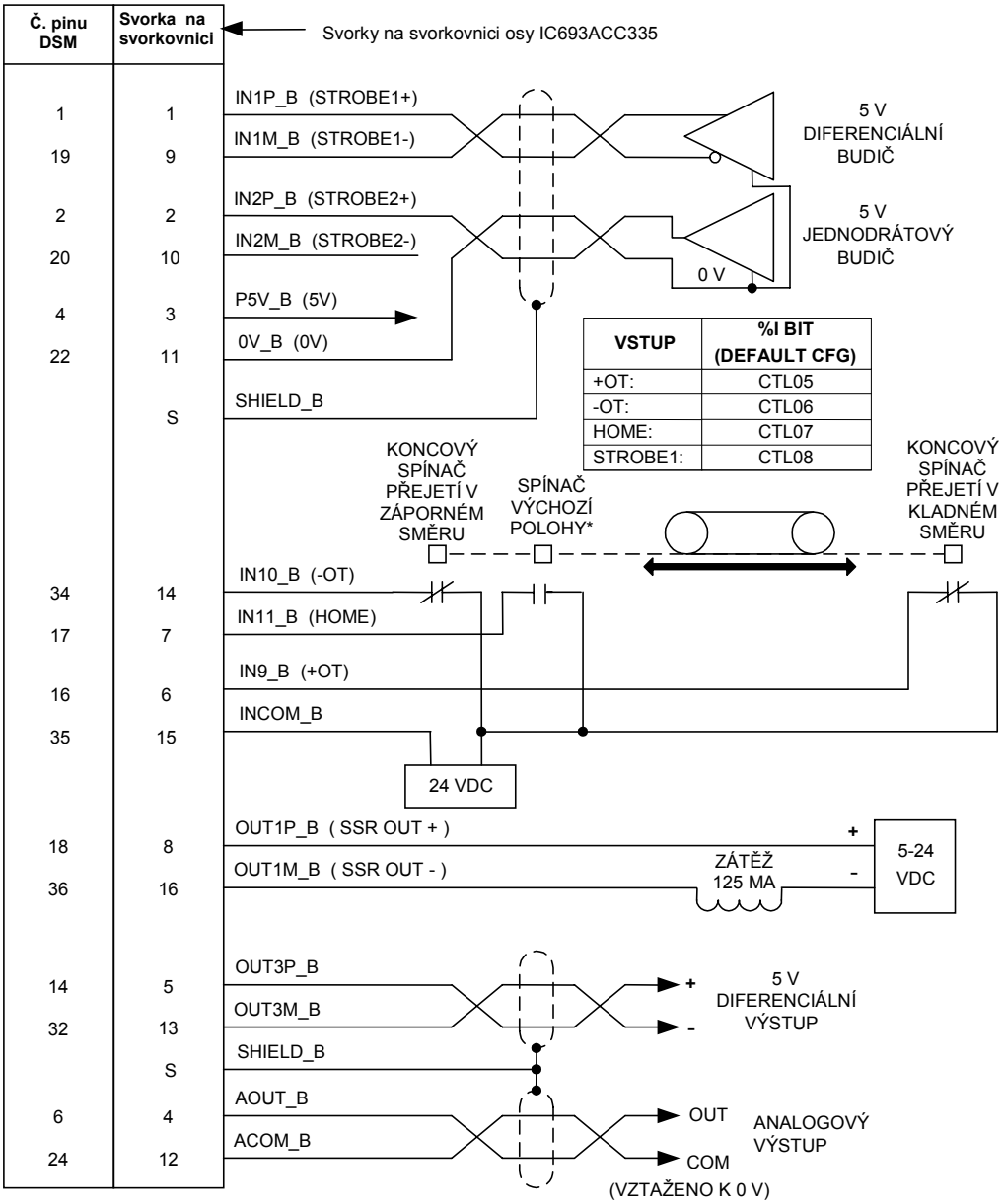
Schéma připojení I/O

Následující obrázky znázorňují typická uživatelská připojení DSM314.



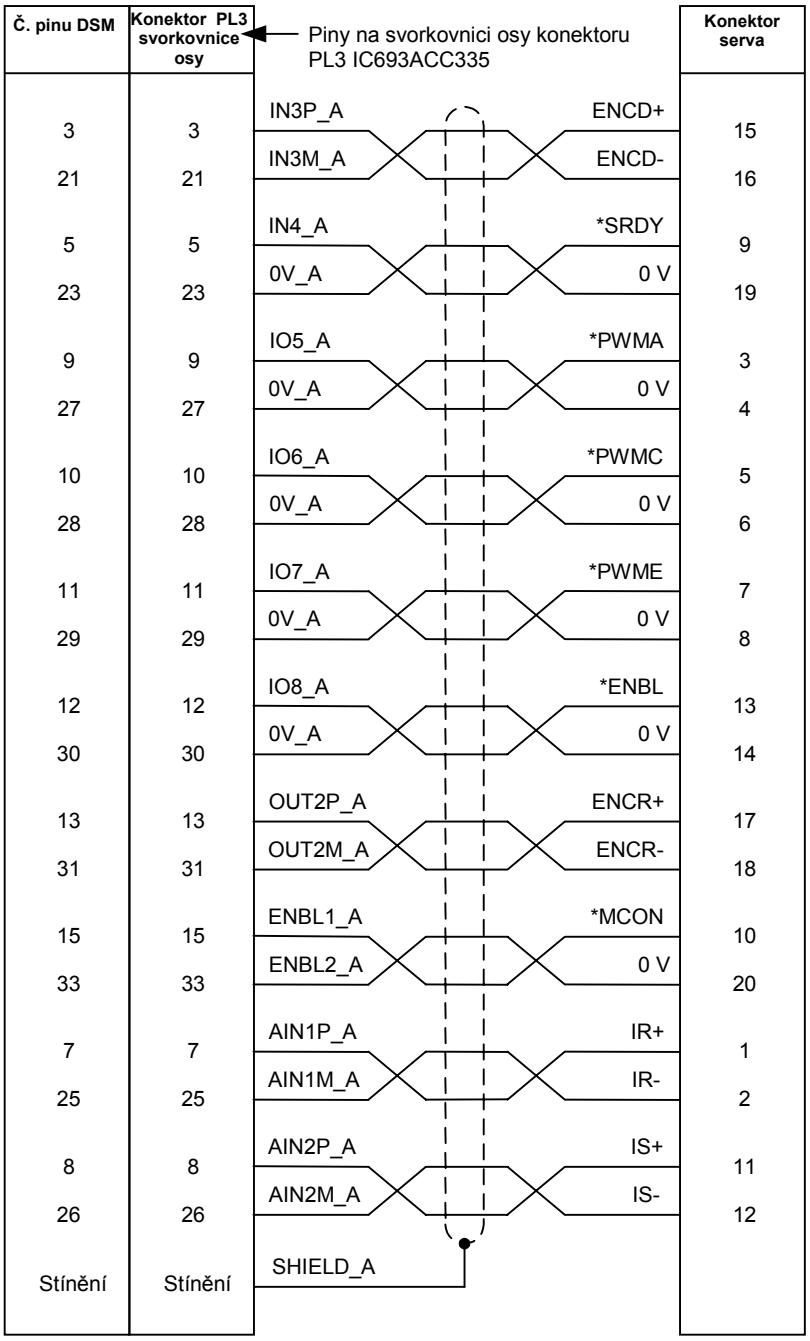
*Poznámka: Informace o spínači výchozí polohy najdete v kapitole 6.

Obrázek 3-16. Připojení digitální servoosy 1



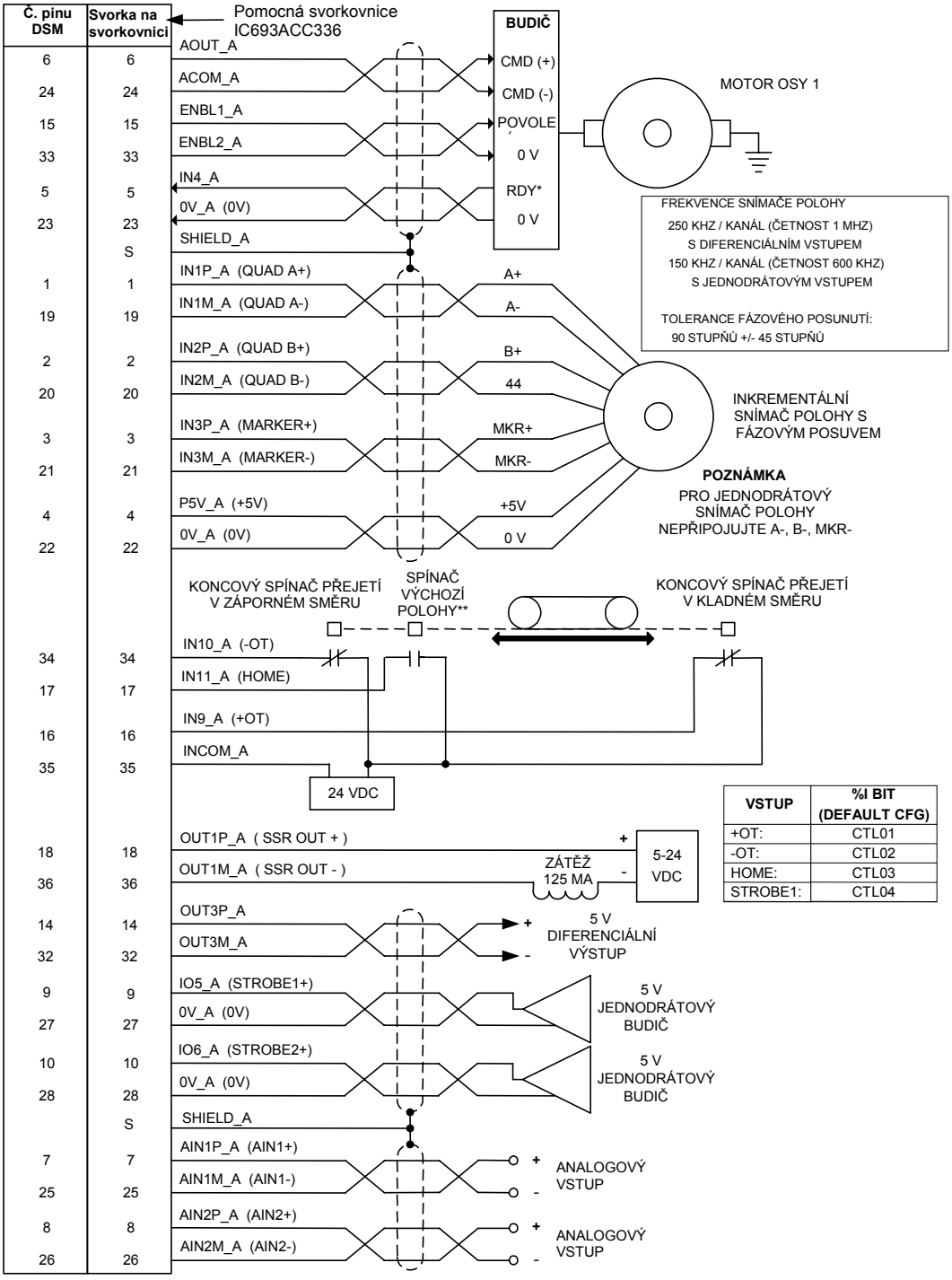
Poznámka: Informace o spínači výchozí polohy najdete v kapitole 6.

Obrázek 3-17. Připojení digitální servoosy 2

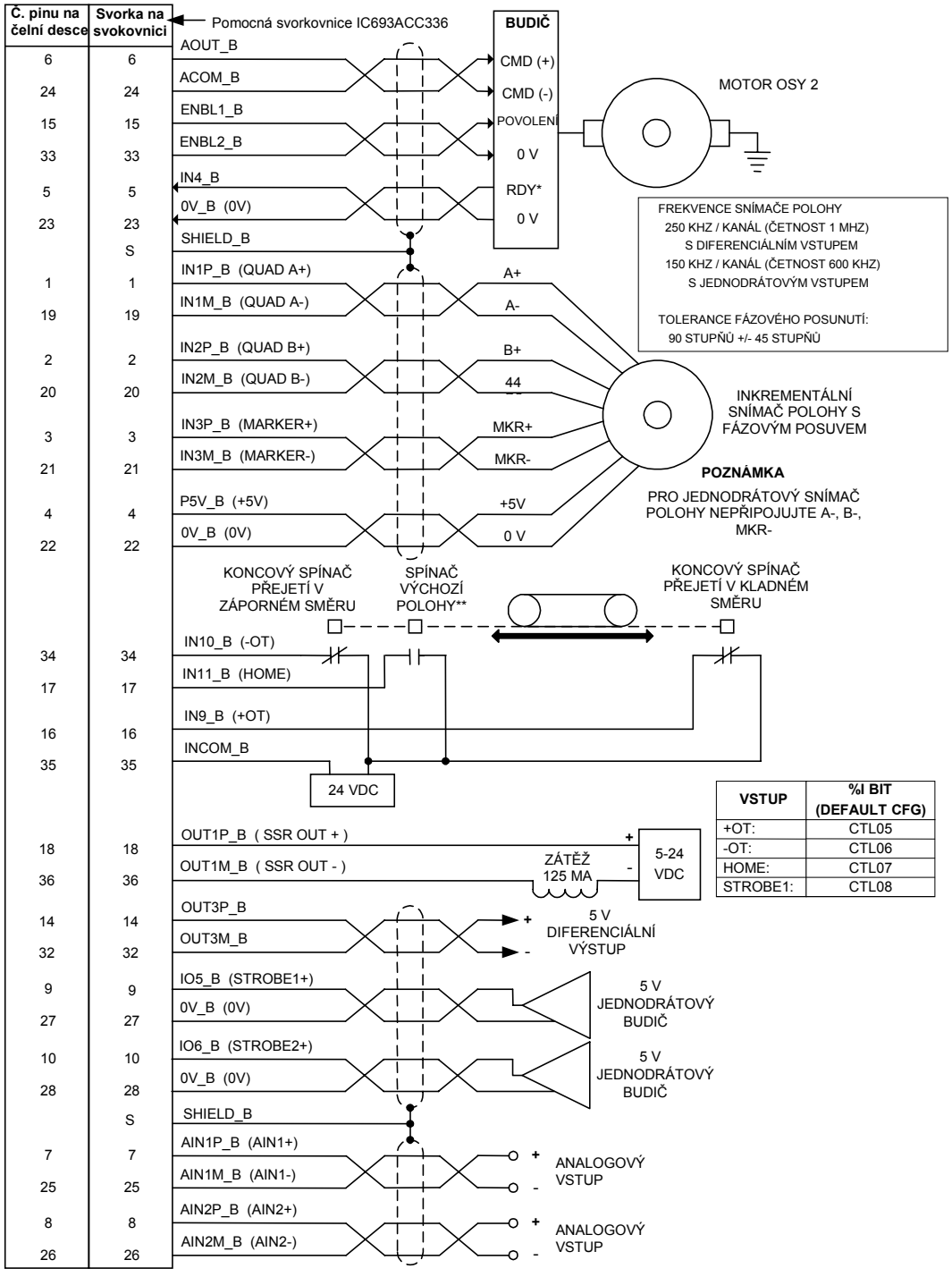


* Označuje negovaný signál

Obrázek 3-18. Připojení povelového kabelu digitálního serva α a β Series (IC800CBL001/002)

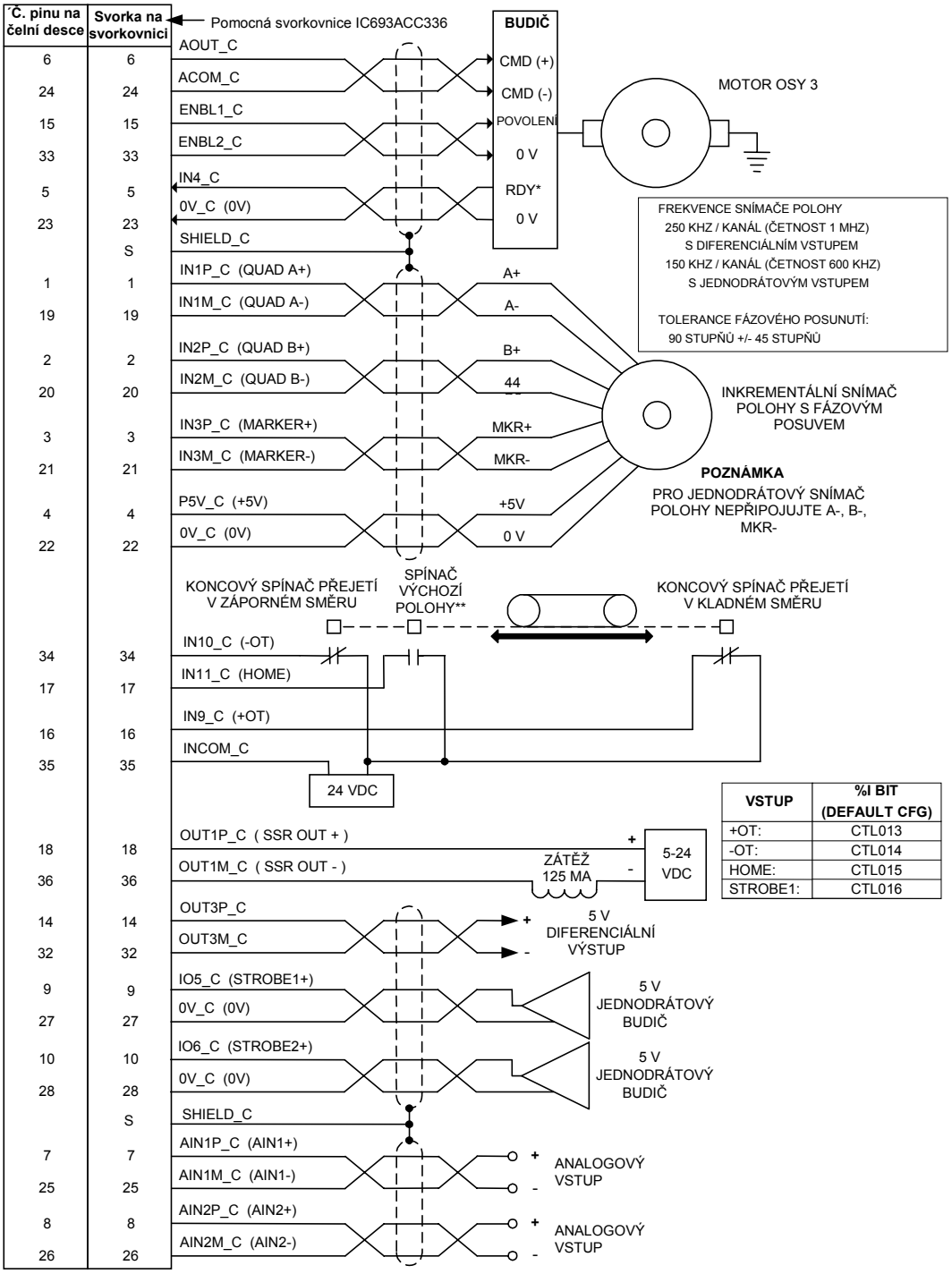


Obrázek 3-19. Připojení analogové servoosy 1



POZNÁMKY: * Označuje negovaný signál
** Informace o spínači výchozí polohy najdete v kapitole 6.

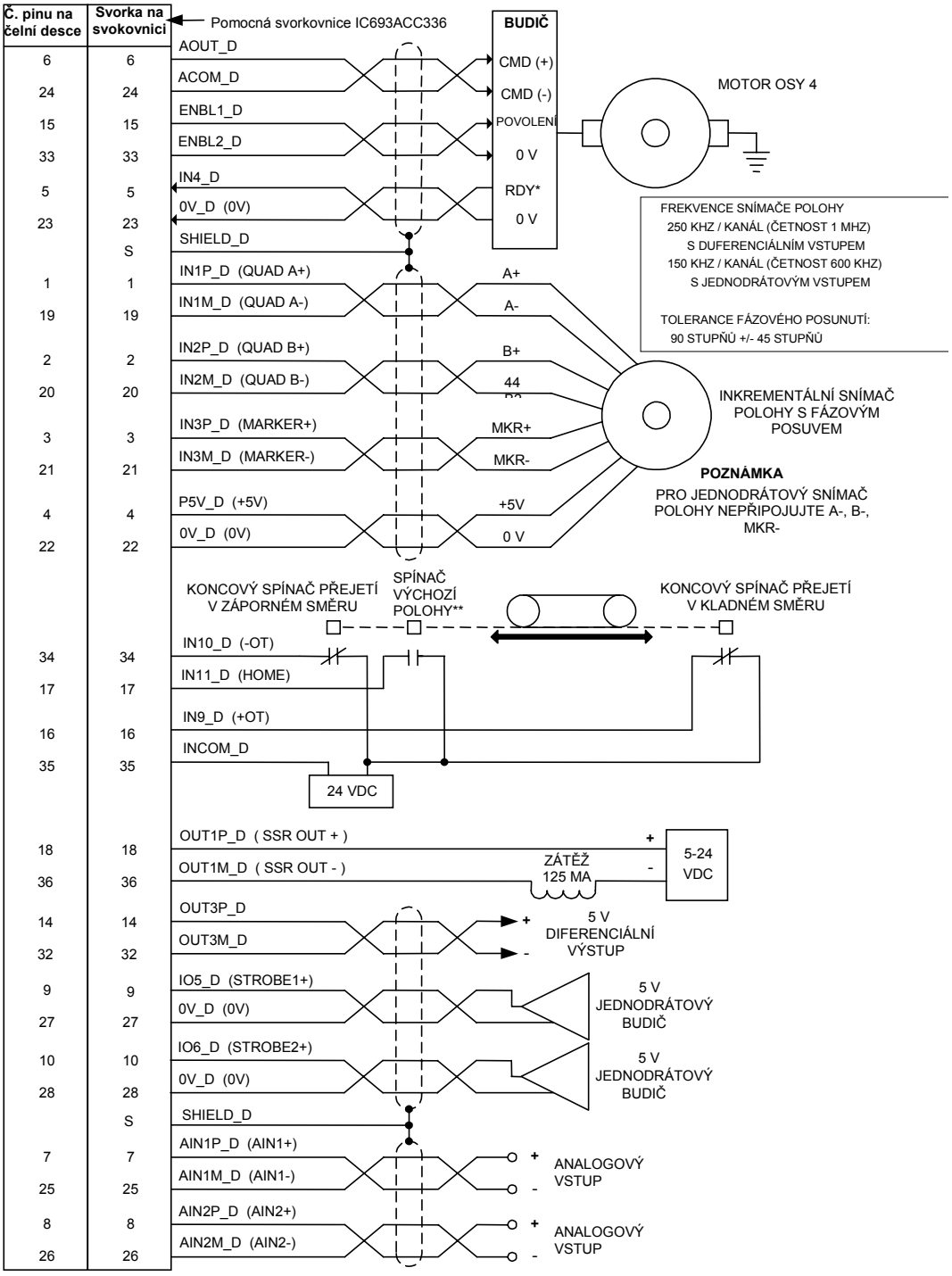
Obrázek 3-20. Připojení analogové servoosy 2



Č. pinu na čelní desce	Svorka na svorkovnici		
6	6	AOUT_C	
24	24	ACOM_C	
15	15	ENBL1_C	
33	33	ENBL2_C	
5	5	IN4_C	
23	23	0V_C (0V)	
	S	SHIELD_C	
1	1	IN1P_C (QUAD A+)	A+
19	19	IN1M_C (QUAD A-)	A-
2	2	IN2P_C (QUAD B+)	B+
20	20	IN2M_C (QUAD B-)	44
3	3	IN3P_C (MARKER+)	MKR+
21	21	IN3M_C (MARKER-)	MKR-
4	4	P5V_C (+5V)	+5V
22	22	0V_C (0V)	0 V
34	34	IN10_C (-OT)	
17	17	IN11_C (HOME)	
16	16	IN9_C (+OT)	
35	35	INCOM_C	
18	18	OUT1P_C (SSR OUT +)	+
36	36	OUT1M_C (SSR OUT -)	-
14	14	OUT3P_C	+
32	32	OUT3M_C	-
9	9	IO5_C (STROBE1+)	
27	27	0V_C (0V)	
10	10	IO6_C (STROBE2+)	
28	28	0V_C (0V)	
	S	SHIELD_C	
7	7	AIN1P_C (AIN1+)	+
25	25	AIN1M_C (AIN1-)	-
8	8	AIN2P_C (AIN2+)	+
26	26	AIN2M_C (AIN2-)	-

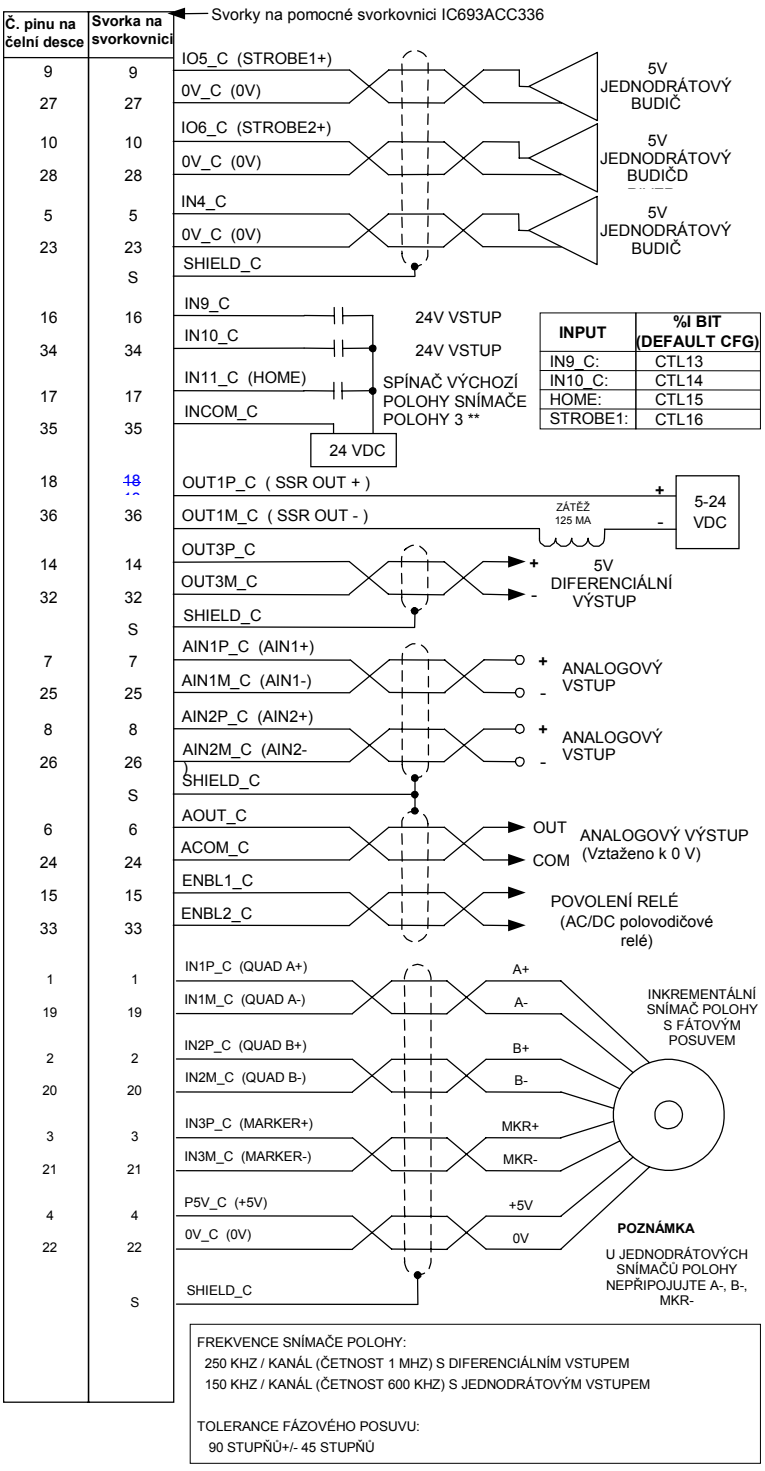
POZNÁMKY: * Označuje negovaný signál
** Informace o spínači výchozí polohy najdete v kapitole 6.

Obrázek 3-21. Připojení analogové servoosy 3



POZNÁMKY: * Označuje negovaný signál
** Informace o spínači výchozí polohy najdete v kapitole 6.

Obrázek 3-22. Připojení analogové servoosy 4



** Poznámka: Informace o spínači výchozí polohy najdete v kapitole 6.

Obrázek 3-23. Připojení pomocné osy (zobrazena osa 3)

Specifikace I/O

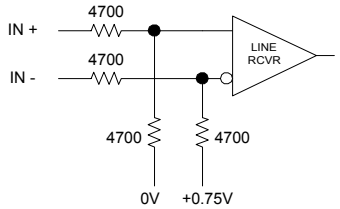
Specifikace a zjednodušené schéma I/O obvodů modulu jsou uvedené na následujících stránkách. Jsou popsané následující obvody:

- Diferenciální/jednodrátové 5 V vstupy (IN1, IN2, IN3)
- Jednodrátový 5 V vstup typu země (IN4)
- Opticky oddělené 24 V vstupy zdroj/země (IN9, IN10, IN11, INCOM)
- Jednodrátové 5 V vstupy/výstupy (IO5, IO6, IO7, IO8)
- 5 V diferenciální výstupy (OUT2, OUT3)
- 24 V ss opticky oddělený výstup (OUT1)
- Opticky oddělené povolení reléového výstupu (OUT4)
- Diferenciální +/- 10 V analogové vstupy (AIN1, AIN2)
- Jednodrátové +/- 10 V analogové výstupy (AOUT1, ACOM)
- Napájení +5 V (P5V, 0V)

Diferenciální / jednodrátové 5 V vstupy

Identifikátor obvodu	*Funkce obvodu digitální servoosy 1, 2*	*Funkce obvodu analogové servoosy 1-4 a pomocné osy 2-4*	*Název signálu (X = konektor A, B, C nebo D)*	*Pin čelní desky*	*Pomocná svorkovnice*	*Svorkovnice serva*
IN1	Vzorkovací vstup 1 (+)	Kanál A snímače polohy (+)	IN1P_X	1	1	1
	Vzorkovací vstup 1 (-)	Kanál A snímače polohy (-)	IN1M_X	19	19	9
IN2	Vzorkovací vstup 2 (+)	Kanál B snímače polohy (+)	IN2P_X	2	2	2
	Vzorkovací vstup 2 (-)	Kanál B snímače polohy (-)	IN2M_X	20	20	10
IN3	Data sériového snímače polohy (+)	Nulový puls snímače polohy (+)	IN3P_X	3	3	nepřipojeno
	Data sériového snímače polohy (-)	Nulový puls snímače polohy (-)	IN3M_X	21	21	nepřipojeno

Typ I/O:	Diferenciální / jednodrátové 5 V vstupy
Typ obvodu:	Zdrojový vstup (9.4 KΩ staženo na 0 V)
Vstupní impedance: Vstup (+) nebo (-)	9.4 KΩ společný režim na 0 V 18.8 KΩ diferenciální
Maximální vstupní napětí:	Společný režim +/- 15 V diferenciální +/- 20 V
Prahové napětí logické 0:	jednodrátový +0.8 V maximálně diferenciální +0.4 V maximálně
Prahové napětí logické 1:	jednodrátový +2.0 V minimálně diferenciální +1.5 V minimálně
Filtrace vstupu:	Typicky 0.5 mikrosekundy
Frekvence snímače polohy s fázovým posunutím:	250 KHz/kanál (četnost pulsů 1 MHz) maximálně s diferenciálními vstupy 150 KHz/kanál (četnost pulsů 600 MHz) maximálně s jednodrátovými vstupy Tolerance fázového posunutí: 90 stupňů+/- 45 stupňů
Charakteristika vzorkování:	Minimální šířka pulsu: 3 mikrosekundy Přesnost zachycení polohy: +/- 2 pulsy s přídavným rozptylem 10 mikrosekund
Poznámky:	Vstup (+) použijte pro jednodrátový režim a vstup (-) ponechejte plovoucí. Piny 0 V čelní desky použijte jako referenci společného režimu nebo jako vratný jednodrátový signál. Na vstupy je možno přivádět logiku 5 V TTL nebo CMOS.



Jednodrátový 5 V vstup typu země

Identifikátor obvodu	*Funkce obvodu servoosy 1, 4*	*Funkce obvodu pomocné osy 2-4*	*Název signálu (X = konektor A, B, C nebo D)*	*Pin čelní desky*	*Pomocná svorkovnice*	*Svorkovnice serva*
IN4	Vstup servo připraveno	5 V vstup na čelní desce	IN4_X	5	5	nepřipojeno

Typ I/O: Jednodrátový 5 V vstup typu země

Typ obvodu: Vstup typu země (4.7 KΩ vztaženo na interních +5 V)

Vstupní impedance: 4.7 KΩ na +5 V

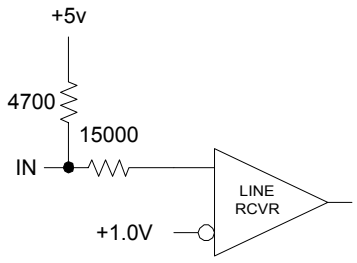
Maximální vstupní napětí: +/- 10.0 V

Prahové napětí logické 0: +0.8 V max

Prahové napětí logické 1: +2.0 V min

Filtrace vstupu: Hardwarový filtr 1.0 mikrosekundy (typicky) + vzorkovací frekvence polohové smyčky (0.5, 1.0 nebo 2.0 milisekundy).

Poznámky: Aby se provedlo sepnutí, tento vstup musí být stažený na 0 V.



Opticky oddělené 24 V vstupy zdroj/země

Identifikátor obvodu	*Funkce obvodu servoosy 1, 4*	*Funkce obvodu pomocné osy 2-4*	*Název signálu (X = konektor A, B, C nebo D)*	*Pin čelní desky*	*Pomocná svorkovnice*	*Svorkovnice serva*
IN9	Přejetí (+)	24 V vstup na čelní desce	IN9_X	16	16	6
IN10	Přejetí (-)	24 V vstup na čelní desce	IN10_X	34	34	14
IN11	Spínač výchozí polohy	Spínač výchozí polohy	IN11_X	17	17	7
INCOM	Nulový vodič 24 V vstupu	Nulový vodič 24 V vstupu	INCOM_X	35	35	15

Typ I/O: Opticky oddělené 24 V vstupy typu zdroj/země

Typ obvodu: Zdroj / země (5 K odpor na INCOM)

Vstupní impedance: 5.4 KΩ na INCOM (při 24 V ss)

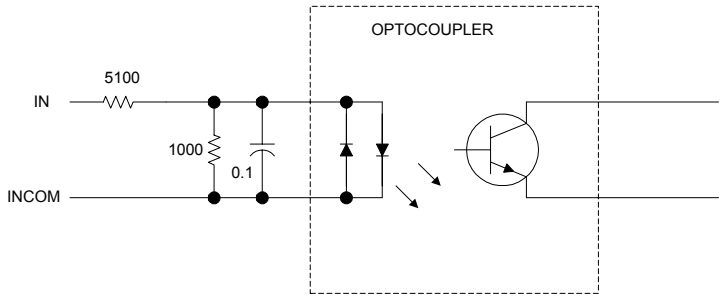
Maximální vstupní napětí: +/- 30.0 V (vztaženo k INCOM)

Prahové napětí logické 0: +/- 6.0 V max (vztaženo k INCOM)

Prahové napětí logické 1: +/- 18.0 V min (vztaženo k INCOM)

Filtrace vstupu: Typicky 5 milisekund

Poznámky: Tyto vstupy používají obousměrné optické vazební členy a je možno je sepnout kladným nebo záporným vstupem vzhledem k INCOM.



Jednodrátové 5 V vstupy/výstupy

Identifikátor obvodu	*Funkce obvodu digitální servoosy 1, 2*	*Funkce obvodu analogová servoosy 1-4 a pomocná osa 2-4*	*Název signálu (X = konektor A, B, C nebo D)*	*Pin čelní desky*	*Pomocná svorkovnice*	*Svorkovnice serva*
IO5	Servo PWM / Alarm	Vstup vzorkování 1	IO5_X / IN5_X	9	9	nepřipojeno
0 V	0 V	0 V	0V_X	27	27	nepřipojeno
IO6	Servo PWM / Alarm	Vstup vzorkování 2	IO6_X / IN6_X	10	10	nepřipojeno
0 V	0 V	0 V	0V_X	28	28	nepřipojeno
IO7	Servo PWM / Alarm	Nepoužívá se	IO7_X / IN7_X	11	11	nepřipojeno
0 V	0 V	0 V	0V_X	29	29	nepřipojeno
IO8	Servo ENBL / Alarm	Nepoužívá se	IO8_X / IN8_X	12	12	nepřipojeno
0 V	0 V	0 V	0V_X	30	30	nepřipojeno

Typ I/O: Jednodrátové 5 V vstupy/výstupy

Typ obvodu: Spotřebič (4.7 KΩ vztaženo na interních +5 V)

Vstupní impedance: 4.7 KΩ na interních +5 V

Maximální vstupní napětí: -1.0 V , +7.0 V

Vstupní prahové napětí logické 0: +0.8 V max

Vstupní prahové napětí logické 1: +2.4 V min

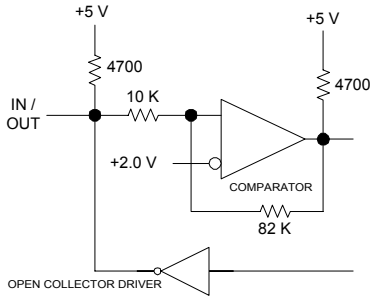
Filtrace vstupu: Typicky 10 mikrosekund

Výstupní proud do země 10 mA max

Výstupní napětí při sepnutém stavu +0.5 V při 10 mA

Charakteristika vzorkování: Minimální šířka pulsu: 10 mikrosekund
Přesnost zachycení polohy: +/- 2 pulsy s přídavným rozptylem 10 mikrosekund

Poznámky: U digitálních serv tyto body fungují jako výstupy PWM / ENBL a alarmové vstupy. U analogových serv a pomocných os tyto body jsou pouze vstupy. Uvedené piny 0 V se musí normálně použít jako návrat signálu.



5 V diferenciální výstupy

Identifikátor obvodu	*Funkce obvodu digitální servoosy 1, 2*	*Funkce obvodu analogové servoosy 1-4 a pomocné osy 2-4*	*Název signálu (X = konektor A, B, C nebo D)*	*Pin čelní desky*	*Pomocná svorkovnice*	*Svorkovnice serva*
OUT2	Požadavek sériového snímače polohy (+)	Nepoužívá se	OUT2P_X	13	13	nepřipojeno
	Požadavek sériového snímače polohy (-)	Nepoužívá se	OUT2M_X	31	31	nepřipojeno
OUT3	5 V výstup na čelní desce (+)	5 V výstup na čelní desce (-)	OUT3P_X	14	14	5
	5 V výstup na čelní desce (-)	5 V výstup na čelní desce (-)	OUT3M_X	32	32	13

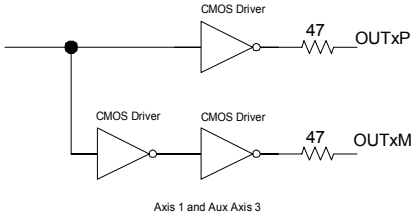
Typ I/O: 5 V diferenciální výstupy

Typ obvodu: Diferenciální dvojčinný výstupní obvod (zdroj / země)

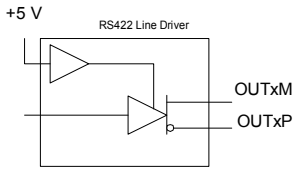
Proud výstupu/Proud spotřebiče: 20 mA max

Výstupní napětí: +/- 1.5 V min na diferenciální zátěži 120 ohm

Poznámky: Osa 1 a osa 3 používají CMOS budiče se sériovými odpory 47 ohm, osa 2 a osa 4 používají linkové budiče RS-422.



Axis 1 and Aux Axis 3



Axis 2 and Aux Axis 4

24 V ss optický oddělený výstup

Identifikátor obvodu	*Funkce obvodu servoosy 1, 4*	*Funkce obvodu pomocné osy 2-4*	*Název signálu (X = konektor A, B, C nebo D)*	*Pin čelní desky*	*Pomocná svorkovnice*	*Svorkovnice serva*
OUT1	24 V výstup na čelní desce (-)	24 V výstup na čelní desce (-)	OUT1P_X	18	18	8
	24 V výstup na čelní desce (-)	24 V výstup na čelní desce (-)	OUT1M_X	36	36	16

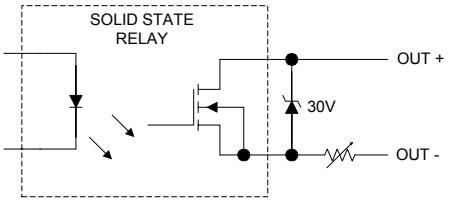
Typ I/O: 24 V ss optický oddělený výstup

Typ obvodu: Oddělené polovodičové relé (SSR)

Výstupní proud: 125 mA trvale, 500 mA po dobu 10 ms (na odporové nebo induktivní zátěži)

Pokles výstupního napětí: 1.0 V max při 0.125 A

Poznámky: Výstup je chráněný diakem 30 V a přepínačem 0.2 A. Pokud dojde ke zkratu, výstup se automaticky přepne do stavu vyšší impedance, dokud se zátěž neodstraní. Zátěž se nesmí znovu připojit po dobu 60 sekund. Toto je stejnosměrný výstup a pokud se přípoje k němu obrátí, bude vždy ve stavu ON.



Opticky oddělené povolení reléového výstupu

Identifikátor obvodu	*Funkce obvodu digitální servoosy 1, 2*	*Funkce obvodu analogové servoosy 1-4*	*Funkce obvodu pomocné osy 2-4*	*Název signálu (X = konektor A, B, C nebo D)*	*Pin čelní desky*	*Pomocná svorkovnice*	*Svorkovnice serva*
ENBL	Servo MCON (+)	Povolení pohonu (+)	Povolení pohonu (+)	ENBL1_X	15	15	nepřipojeno
	Servo MCON 0 V	Povolení pohonu (-)	Povolení pohonu (-)	ENBL2_X	33	33	nepřipojeno

Typ I/O: Opticky oddělené povolení reléového výstupu

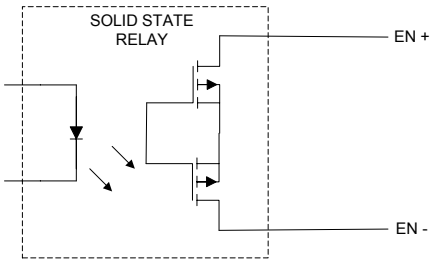
Typ obvodu: Oddělené střídavé polovodičové relé (SSR)

Výstupní proud: 30 mA trvale, 50 mA po dobu 10 ms

Pokles výstupního napětí: 1.0 V max při 10 mA

Poznámky: Toto je nízkoproudový výstup SSR. Výstup bude ve stavu ON, když související LED dioda Osa povolena na čelní desce bude svítit. K tomu dojde. když:

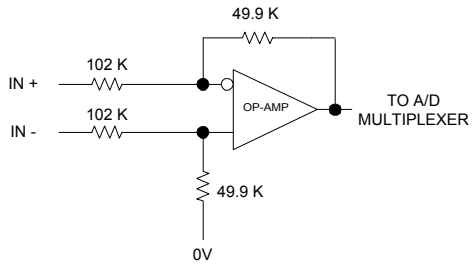
- Servo je povoleno
- Používá se povel %AQ *Vynucení rychlosti digitálního serva* (osa 1, 2)
- Používá se povel %AQ *Vynucení analogového výstupu*



Diferenciální +/- 10 V analogové vstupy

Identifikátor obvodu	*Funkce obvodu digitální servoosy 1, 2*	*Funkce obvodu analogové servoosy 1-4 a pomocné osy 2-4*	*Název signálu (X = konektor A, B, C nebo D)*	*Pin čelní desky*	*Pomocná svorkovnice*	*Svorkovnice serva*
AIN1	Proud IR fází (+)	Analogový vstup (+) na čelní desce	AIN1P_X	7	7	nepřipojeno
	Proud IR fází (-)	Analogový vstup (-) na čelní desce	AIN1M_X	25	25	nepřipojeno
AIN2	Proud IS fází (+)	Analogový vstup (+) na čelní desce	AIN2P_X	8	8	nepřipojeno
	Proud IS fází (-)	Analogový vstup (-) na čelní desce	AIN2M_X	26	26	nepřipojeno

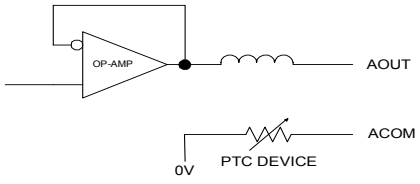
Typ I/O:	Diferenciální analogové vstupy +/- 10 V
Typ obvodu:	Diferenciální vstup
Vstupní impedance:	102 KΩspolečný režim vztaženo ke konektoru 0 V čelní desky 204 KΩdiferenciální
Maximální vstupní napětí:	+/- 15 V společný režim vztaženo ke konektoru 0 V čelní desky diferenciální +/- 20 V
Rozlišitelnost:	15 bitů
Linearita:	13 bitů
Vstupní offset:	+/- 1.0 mV
Vynucený koeficient zisku D/A převodníku:	+/- 10.0 V = +/- 32 000 jednotek
Přesnost zisku:	+/- 0.5 %
Vynucená rychlost aktualizace analogového výstupu:	2 ms + doba cyklu PLC, když se předávají data do PLC tabulky %AI.
Poznámky:	Piny 0 V čelní desky použijte jako referenci společného režimu.



Jednodrátový +/- 10 V analogový výstup

Identifikátor obvodu	*Funkce obvodu analogové servoosy 1-4*	*Funkce obvodu digitální servoosy 1,2 a pomocné osy 2-4*	*Název signálu (X = konektor A, B, C nebo D)*	*Pin čelní desky*	*Pomocná svorkovnice*	*Svorkovnice serva*
AOUT1	Povel rychlosti analogového serva nebo povel kroutícího momentu analogového serva	Analogový výstup na čelní desce	AOUT_X	6	6	4
ACOM	Nulový vodič analogového výstupu	Nulový vodič analogového výstupu	ACOM_X	24	24	12

Typ I/O:	Jednodrátový analogový výstup
Typ obvodu:	Napěťový sledovací výstup operačního zesilovače
Nízká impedance:	2 KΩ minimum
Výstupní proud:	5 mA max
Rozlišitelnost:	13 bitů
Linearita:	13 bitů
Výstupní offsetové napětí:	+/- 500 mikrovolt max
Činitel vynuceného D/A zisku:	+/- 10.0 V = +/- 32000 000 jednotek
Přesnost zisku:	+/- 1.0 %
Rychlost aktualizace vynuceného analogového výstupu:	1. Rychlost cyklu PLC, když se používá povel %AQ *Vynucený analogový výstup*. 2. 250 ms, když se používá jako výstup ladění digitálního serva.
Poznámky:	Protože toto je jednodrátový výstup, musí normálně řídit uživatelské zařízení s diferenciálním vstupem, aby se zabránilo problémům se šumem v uživatelském režimu. Kladný diferenciální vstup musí být připojený k AOUT a záporný diferenciální vstup k ACOM. Povel %AQ *Volba analogového výstupního režimu* je možno použít ke zvolení zdroje pro analogový výstup. Více informací najdete v kapitole 5.



Napájení +5 V

Identifikátor obvodu	*Funkce obvodu servoosy*	*Funkce obvodu pomocné osy*	*Název signálu (X = konektor A, B, C nebo D)*	*Pin čelní desky*	*Pomocná svorkovnice*	*Svorkovnice serva*
P5V	Napájení 5 V	Napájení 5 V	P5V_X	4	4	3
0 V	0 V	0 V	0V_X	22	22	11

Typ I/O: Napájení snímače polohy +5 V

Typ obvodu: Napájení +5 V s elektronickou ochranou proti zkratu

Výstupní napětí: 4.70 V až 5.20 V při 0.5 A

Výstupní proud: 0.5 A max (celkem na všechny konektory)

Poznámky: Tento výstup je určený k napájení externích zařízení, například inkrementálního snímače polohy s fázovým posunutím vyžadujícího méně než 0,5 A celkem ze všech 4 konektorů osy. Výstupní proud zajišťuje napájecí zdroj +5 V propojovací roviny PLC a je chráněný elektronickou ochranou proti zkratu v modulu DSM314.

Celkový proud odebíraný externím zařízením z tohoto obvodu +5 V se musí přičíst k hodnotě odběru z napájecího zdroje na obrazovce konfigurace DSM314 v konfiguračním softwaru a musí se zahrnout, pokud se provádí ruční výpočet odběru z napájecího zdroje.

Uvedený pin 0 V se normálně používá jako vratný signál napájení.

Kapitola 4

Konfigurace

Tato kapitola popisuje všechny konfigurační podrobnosti potřebné k nastavení DSM314 pro konkrétní aplikaci. V kapitole 2 najdete instrukce, jak nakonfigurovat systém, aby se poslal povel *Jog* do DSM pro účely ověření, že systémové komponenty jsou funkční. Informace o konfiguraci elektronické vačky najdete v kapitole 16.

Modul DSM314 vyžaduje ***CPU s firmwarem verze 10.00 nebo pozdější***. V dokumentaci k CPU PLC najdete, kterou platformu hardwaru CPU tato revize firmwaru podporuje. Konfigurace DSM314 se provádí pomocí některého z následujících konfiguračních/programovacích nástrojů:

- CIMPLICITY Machine Edition Logic Developer – PLC verze 2.1 nebo pozdější
- VersaPro verze 1.1 nebo pozdější.

Konfigurace je dvoustupňový proces, který se skládá z:

1. Konfigurace sestavy/pozice
2. Konfigurace modulu

Konfigurace sestavy/pozice

Konfigurace hardwaru definuje typ a umístění všech modulů přítomných v sestavě PLC. To se provádí nejdříve vyplněním nastavovacích obrazovek, které představují moduly na základní desce, pak uložením informací do konfiguračního souboru, který se pak načte do CPU PLC.

Viz kapitola 2, část 4, “Výuka konfigurace Motion Mate DSM314”, kde najdete podrobnosti pro volbu základní desky DSM314 a umístění pozice v sestavě PLC. Po definování konfigurace modulu sestavy/pozice pokračujte druhou částí konfiguračního postupu, Konfigurace modulu, kde nakonfigurujete DSM314 na požadavky vaší konkrétní aplikace.

Konfigurace modulu

Nastavení konfiguračních parametrů

Stejně jako u konfigurace sestavy I/O, konfigurace modulu se provádí vyplněním obrazovek pomocí konfiguračního softwaru pro hardware. Data hardwarové konfigurace se uvádějí v tabulkovém tvaru. Záložky odpovídají následujícím skupinám. Záložka a/nebo záložky odpovídající skupinám jsou uvedené v závorkách za názvem skupiny. Všimněte si, že záložky se objevují a mizí v závislosti na zvolené konfiguraci provedené na záložce Nastavení. Pokud například osa 4 bude zakázána, záložky osy 4 se nezobrazí.

- Data konfigurace modulu (Nastavení, bity CTL, výstupní bity)
- Sériová komunikace (port SNP)
- Data konfigurace osy
- Ladění osy
- Rozšířená nastavení
- Spotřeba

Základní obsah jednotlivých záložek je následující:

Název záložky	Funkce nebo popis
Nastavení	Obsahuje přiřazení a délku adres PLC, nastavení osy DSM a další globální data.
SNP	Nastavení SNP portu předního panelu DSM
Bity CTL	Konfigurace 24 řídicích bitů DSM
Výstupní bity	Konfigurace 8 digitálních výstupů čelní desky DSM
Osa #1 - Osa #4	Konfigurace parametrů osy, například meze polohy, nalezení rychlosti nájezdu do výchozí polohy a zrychlení jogu
Ladění #1 – Ladění #4	Konfigurace údajů ladění servosmyčky, například typ motoru, časová konstanta polohové smyčky a rychlostní zpětná vazba.
Rozšířené	Umožňuje zápis uživatelských parametrů ladění pro každou osu
Spotřeba	Uvádí pří DSM požadovaný z napájení propojovací roviny (4.0 W + výkon snímače polohy)

Další podrobnosti týkající se činnosti konfiguračního softwaru najdete v online nápovědě k vašemu softwaru nebo v odpovídajícím uživatelském manuálu softwaru:

Uživatelský manuál softwaru VersaPro™, GFK-1670

Krátký úvod pro CIMPLICITY® Machine Edition Logic Developer-PLC, GFK-1918

Nastavení

Záložka Nastavení obsahuje konfigurační informace, které umožňují definovat základní činnost modulu. Tato nastavení se týkají počtu řízených os, provozních režimů os, atd. Volby na těchto záložkách mohou mít za následek, že se jiné záložky v konfigurace objeví nebo zmizí. Pokud například zakážete osu 4, záložka Osa a Ladění týkající se osy 4 se nezobrazí.

Všimněte si, že záložka Nastavení je tam, kde definujete pohybový program, lokální logiku a názvy bloků vačky. Tyto názvy určují, které programy uložené v CPU se při zapnutí napájení přenesou do každého DSM. Další informace týkající se těchto otázek najdete v příslušných kapitolách tohoto manuálu.

V průběhu každého cyklu CPU se mezi DSM314 a CPU automaticky přenášejí data. Záložka Nastavení obsahuje adresy dat rozhraní CPU a počáteční umístění pro automatické přesuny. Konfigurační parametry na kartě Nastavení jsou popsané v Tabulka 4-1. Všechna označení v *Referenčním odstavci* uvedená v tabulce se týkají této kapitoly.

Tabulka 4-1. Záložka Nastavení

Konfigurační parametr	Popis	Hodnoty	Výchozí	Jednotky	Referenční odstavec
Počet os	Počet řízených os	1 2 3 4	4	---	1.01
Adresa %I	Počáteční adresa pro typ adresy %I (80 bitů)	Závisí na CPU	%I00001 nebo další vyšší adresa	---	1.02
Délka %I	Délka adresy %I	32 = 1 osa 48 = 2 osy 64 = 3 osy 80 = 4 osy	--- Délka je automaticky určena nastavením počtu os	---	1.02
Adresa %Q	Počáteční adresa pro typ adresy %Q (80 bitů)	Závisí na CPU	%Q00001 nebo další vyšší adresa	---	1.02
Délka %Q	Délka adresy %Q	32 = 1 osy 48 = 2 osy 64 = 3 osy 80 = 4 osy	--- Délka je automaticky určena nastavením počtu os	---	1.02
Adresa %AI	Počáteční adresa pro typ adresy %AI (84 bitů)	Závisí na CPU	%AI00001 nebo další vyšší adresa	---	1.02
Délka %AI	Délka adresy %AI	24 = 1 osy 44 = 2 osy 64 = 3 osy 84 = 4 osy	--- Délka je automaticky určena nastavením počtu os	---	1.02
Adresa %AQ	Počáteční adresa pro typ adresy %AQ (12 bitů)	Závisí na CPU	%AQ00001 nebo další vyšší adresa	---	1.02
Délka %AQ	Délka adresy %AQ	3 = 1 osy 6 = 2 osy 9 = 3 osy 12 = 4 osy	--- Délka je automaticky určena nastavením počtu os	---	1.02
Režim osy 1	Režim řízení osy 1	Analogové servo Digitální servo	Analogové servo	---	1.03
Režim osy 2	Režim řízení osy 2	Analogové servo Digitální servo Pomocná osa	Analogové servo	---	1.03

Tabulka 4-1. Záložka Nastavení , pokračování

Konfigurační parametr	Popis	Hodnoty	Výchozí	Jednotky	Referenční odstavec
Režim osy 3	Režim řízení osy 3	Analogové servo Pomocná osa	Pomocná osa	---	1.03
Režim osy 4	Režim řízení osy 4	Zakázáno Analogové servo Pomocná osa	Zakázáno	---	1.03
Režim lokální logiky	Režim nástroje lokální logiky	Zakázáno Povoleno	Zakázáno	---	1.04
Celkový příkon snímačů polohy	Požadavky na příkon snímače polohy	Číselné pole	0	W	1.05
Název bloku pohybového programu	Specifikuje název pohybového programu vykonávaný modulem	ASCII-20	<prázdné>	---	1.06
Název bloku lokální logiky	Specifikuje název programu lokální logiky vykonávaného modulem	ASCII-20	<prázdné>	---	1.07
Název bloku vačky	Specifikuje název bloku vačky vykonávaného modulem	ASCII-20	<prázdné>	---	1.08

1.01 Počet os. Tento parametr zvolí počet os, které DSM314 má řídit, a velikost automatického přenosu dat mezi PLC a DSM. (výchozí = 4.) Následující dvě tabulky uvádějí možné kombinace os pro analogový a digitální režim. Osy identifikované jako **Omezená pomocná osa** zajišťují polohovou zpětnou vazbu, ale žádné generování interního pohybového povelu.

Tabulka 4-2. Počet os

Údaj	Počet os	Osa 1	Osa 2	Osa 3	Osa 4	Lokální logika	Rychlost vzorkování (ms)
1.	4	Analogové servo	Analogové servo	Analogové servo	Analogové servo	Zakázáno	2.0
2.	4	Analogové servo	Analogové servo	Analogové servo	Pomocná osa	Zakázáno	2.0
3.	4	Analogové servo	Analogové servo	Analogové servo	Zakázáno	Zakázáno	2.0
4.	4	Analogové servo	Analogové servo	Analogové servo	Zakázáno	Povoleno	2.0
5.	4	Analogové servo	Analogové servo	Analogové servo	Omezená pomocná osa	Povoleno	2.0
6.	4	Analogové servo	Analogové servo	Pomocná osa	Analogové servo	Zakázáno	2.0
7.	4	Analogové servo	Analogové servo	Pomocná osa	Pomocná osa	Zakázáno	2.0
8.	4	Analogové servo	Analogové servo	Pomocná osa	Zakázáno	Zakázáno	2.0
9.	4	Analogové servo	Analogové servo	Pomocná osa	Zakázáno	Povoleno	2.0
10.	4	Analogové servo	Analogové servo	Pomocná osa	Omezená pomocná osa	Povoleno	2.0
11.	4	Analogové servo	Pomocná osa	Analogové servo	Analogové servo	Zakázáno	2.0
12.	4	Analogové servo	Pomocná osa	Analogové servo	Pomocná osa	Zakázáno	2.0
13.	4	Analogové servo	Pomocná osa	Analogové servo	Zakázáno	Zakázáno	2.0
14.	4	Analogové servo	Pomocná osa	Analogové servo	Zakázáno	Povoleno	2.0
15.	4	Analogové servo	Pomocná osa	Analogové servo	Omezená pomocná osa	Povoleno	2.0
16.	4	Analogové servo	Pomocná osa	Pomocná osa	Analogové servo	Zakázáno	2.0
17.	4	Analogové servo	Pomocná osa	Pomocná osa	Pomocná osa	Zakázáno	2.0
18.	4	Analogové servo	Pomocná osa	Pomocná osa	Zakázáno	Zakázáno	2.0
19.	4	Analogové servo	Pomocná osa	Pomocná osa	Zakázáno	Povoleno	2.0
20.	4	Analogové servo	Pomocná osa	Pomocná osa	Omezená pomocná osa	Povoleno	2.0

Údaj	Počet os	Osa 1	Osa 2	Osa 3	Osa 4	Lokální logika	Rychlost vzorkování (ms)
21.	3	Analogové servo	Analogové servo	Analogové servo	nepoužívá se	Zakázáno	2.0
22.	3	Analogové servo	Analogové servo	Analogové servo	nepoužívá se	Povoleno	2.0
23.	3	Analogové servo	Analogové servo	Pomocná osa	nepoužívá se	Zakázáno	2.0
24.	3	Analogové servo	Analogové servo	Pomocná osa	nepoužívá se	Povoleno	2.0
25.	3	Analogové servo	Pomocná osa	Analogové servo	nepoužívá se	Zakázáno	2.0
26.	3	Analogové servo	Pomocná osa	Analogové servo	nepoužívá se	Povoleno	2.0
27.	3	Analogové servo	Pomocná osa	Pomocná osa	nepoužívá se	Zakázáno	2.0
28.	3	Analogové servo	Pomocná osa	Pomocná osa	nepoužívá se	Povoleno	2.0
30.	2	Analogové servo	Analogové servo	nepoužívá se	nepoužívá se	Zakázáno	1.0
31.	2	Analogové servo	Analogové servo	nepoužívá se	nepoužívá se	Povoleno	2.0
32.	2	Analogové servo	Pomocná osa	nepoužívá se	nepoužívá se	Zakázáno	1.0
33.	2	Analogové servo	Omezená pomocná osa	nepoužívá se	nepoužívá se	Povoleno	1.0
34.	1	Analogové servo	nepoužívá se	nepoužívá se	nepoužívá se	Zakázáno	0.5
35.	1	Analogové servo	nepoužívá se	nepoužívá se	nepoužívá se	Povoleno	1.0

Tabulka 4-3. Konfigurace digitální osy

Údaj	Počet os	Osa 1	Osa 2	Osa 3	Osa 4	Lokální logika	Rychlost vzorkování (ms)
1.	4	Digitální servo	Digitální servo	Analogové servo	Zakázáno	Zakázáno	2.0
2.	4	Digitální servo	Digitální servo	Analogové servo	Zakázáno	Povoleno	2.0
3.	4	Digitální servo	Digitální servo	Pomocná osa	Zakázáno	Zakázáno	2.0
4.	4	Digitální servo	Digitální servo	Pomocná osa	Zakázáno	Povoleno	2.0
5.	3	Digitální servo	Digitální servo	Analogové servo	nepoužívá se	Zakázáno	2.0
6.	3	Digitální servo	Digitální servo	Analogové servo	nepoužívá se	Povoleno	2.0
7.	3	Digitální servo	Digitální servo	Pomocná osa	nepoužívá se	Zakázáno	2.0
8.	3	Digitální servo	Digitální servo	Pomocná osa	nepoužívá se	Povoleno	2.0
9.	2	Digitální servo	Digitální servo	nepoužívá se	nepoužívá se	Zakázáno	2.0
10.	2	Digitální servo	Digitální servo	nepoužívá se	nepoužívá se	Povoleno	2.0
11.	1	Digitální servo	nepoužívá se	nepoužívá se	nepoužívá se	Zakázáno	2.0
12.	1	Digitální servo	nepoužívá se	nepoužívá se	nepoužívá se	Povoleno	2.0

1.02 Délka I/Q/AI/AQ. Zobrazí počáteční adresy a počet adres %I, %Q, %AI a %AQ přiřazených modulu DSM314. Velikost adres se nastaví, když uživatel nakonfiguruje počet os.

1.03 Režim osy n. Tyto parametry definují typy povelových výstupů pro podsystémy serva. *Digitální servo* zvolí speciální digitální výstup pro digitální servopohony GE Fanuc. Pokud bude zvoleno **Digitální servo**, osa 1 a 2 musí být digitální. **Analogové servo** zvolí pro standardní analogové servopohony rychlostní povel +/-10 V nebo povel krouticího momentu +/- 10 V. Rozhraní krouticího momentu nebo rychlosti se nakonfiguruje nastavením povelu analogového serva v konfiguraci modulu. **Pomocná osa** zakáže polohovou smyčku, takže interní generátor povelu a vstup snímače polohy je možno použít pro funkce vlečené osy nebo vačky. Pokud se některý konektor osy použije jako vstup master zdroje pro režim vlečené osy, je nutno ho nakonfigurovat jako **Pomocnou osu**. **Pomocná osa** bude generovat analogové napětí úměrné *Zadané rychlosti*, pokud ***Rychlostní zpětná vazba*** bude nastavena na nenulovou hodnotu. Když osa bude

nakonfigurovaná jako **Pomocná osa** a identifikovaná jako **Omezená pomocná osa** v tabulce 4-2, polohová zpětná vazba bude k dispozici, ale interní generátor pohybového povelu nelze použít. Osa nakonfigurovaná jako **Zakázaná** (platí pouze pro osu 4) vytváří analogové a digitální I/O, ale bez polohové zpětné vazby nebo interního generátoru pohybového povelu. (Výchozí = *Analogové servo* (Osa 1-2), *Pomocná osa* (Osa 3), *Zakázaná* (Osa 4)).

1.04 Režim lokální logiky. Tento parametr definuje stav nástroje Lokální logiky. Má-li se povolit Lokální logika, tento parametr musí být nastavený do stavu Povoleno. Pokud Lokální logika bude povolená, maximální počet povolených servoos bude 3. ***Název bloku lokální logiky*** musí být zadaný také, když ***Režim lokální logiky*** = Povoleno (Výchozí = *Zakázáno).*

1.05 Celkový příkon snímačů polohy. Tento parametr definuje celkovou spotřebu všech snímačů polohy připojených k modulu DSM. (výchozí = *0*). Tento parametr musí vzít v úvahu všechny analogové osy a master snímače polohy a používá se k aktualizaci zobrazení spotřeby v konfiguračním softwaru.

1.06 Název bloku pohybového programu. Tento parametr definuje volitelný název bloku pohybového programu vykonávaného v modulu DSM. Pokud nebude zadaný žádný název, DSM bude předpokládat, že bloky pohybových programů se nepoužívají. Pokud bude zadaný název, v aktivním adresáři musí existovat blok pohybového programu stejného názvu. Zadání neplatného názvu bude mít za následek, že při ukládání hardwarové konfigurace do PLC se bude hlásit chyba. Název se může skládat až z 31 znaků, ale nesmí mít prázdné mezery, je však možno používat znak podtržítko. Jsou přípustná malá i velká písmena. (Výchozí = *<prázdný>*).

1.07 Název bloku lokální logiky. Tento parametr definuje volitelný název bloku lokální logiky vykonávaného v modulu DSM. Pokud nebude zadaný žádný název, DSM bude předpokládat, že bloky lokální logiky se nepoužívají. Pokud bude zadaný název, v aktivním adresáři musí existovat blok lokální logiky stejného názvu. Zadání neplatného názvu bude mít za následek, že při ukládání hardwarové konfigurace do PLC se bude hlásit chyba. Název se může skládat až z 31 znaků, ale nesmí mít prázdné mezery, je však možno používat znak podtržítko. Jsou přípustná malá i velká písmena. Aby Lokální logika fungovala, ***Režim lokální logiky*** musí být také nastavený na stav Povoleno. (Výchozí = *<prázdný>*).

1.08 Název bloku vačky. Definuje volitelný název bloku vačky vykonávaného DSM modulem. Pokud nebude zadaný žádný název, DSM bude předpokládat, že bloky CAM se nepoužívají. Pokud bude zadaný název, v aktivním adresáři musí existovat blok CAM stejného názvu. Zápis neplatného názvu bude mít za následek, že při ukládání hardwarové konfigurace do PLC se bude hlásit chyba. Pravidla pro názvy bloků vačky jsou:

- Jsou přípustné pouze znaky A-Z, a-z, 0-9 a _ (znak podtržítka). Podtržítka a prázdné mezery za sebou nejsou přípustné.
- Název bloku vačky musí začínat písmenem nebo znakem podtržítka.
- Blok nesmí mít stejný název jako jiný blok existující v otevřeném adresáři.
- Název bloku vačky smí obsahovat maximálně sedm znaků.

Tuto vlastnost podporoval zpočátku firmware DSM314 verze 2.00 a vyžaduje použití některého z následujících softwarových nástrojů:

- CIMPLICITY Machine Edition Logic Developer – PLC verze 2.1 nebo pozdější
- VersaPro 1.5 nebo pozdější a CAM Editor 1.0 nebo pozdější

Podrobnosti o funkci vačky viz kapitola 16. (Výchozí = *<prázdný>*).

Konfigurační data portu sériové komunikace

Sériový komunikační port DSM314 používá konektor RJ-11 s označením COMM na čelní desce modulu a podporuje protokol RS-232. Používá se pro upgrade firmwaru do paměti flash a musí být správně nakonfigurovaný tak, aby komunikoval se softwarem pro upgrade spuštěným na vašem programovacím zařízení. Přesvědčte se, že konfigurační parametry programovacího zařízení a konfigurační parametry portu sériové komunikace DSM314 navzájem souhlasí. Tyto konfigurační parametry jsou popsané v Tabulka 4-4.

Tabulka 4-4. Záložka portu SNP

Konfigurační parametr	Popis	Hodnoty	Výchozí	Jednotky	Odstavec
Přenosová rychlost	Přenosová rychlost portu SNP	300, 600, 1200, 2400, 4800, 9600, 19200	19200	---	2.01
Stop bity	Počet stop bitů	1 nebo 2	1	---	2.02
Parita	Parita	ODD, EVEN, NONE	ODD	---	2.03
Doba prodlevy	Maximální doba prodlevy linky	1...255	10	s	2.04
Doba obrátky modemu	Doba obrátky modemu	0…255	0	0,01 s/jednotku	2.05
SNP ID	SNP ID	Sedm znaků skládajících se z A-F a 0-9. První znak musí být A-F.	A000001	---	2.06

2.01 Přenosová rychlost. Parametr přenosové rychlosti udává rychlost přenosu dat přes sériový port v bitech za sekundu.

2.02 Stop bity. Zařízení sériové komunikace používají minimálně jeden (**1**) stop bit. U pomalejších zařízení nastavte tento parametr na dva (**2**) stop bity.

2.03 Parita. Udává, jestli se používá nebo nepoužívá bit parity (**NONE** pokud ne) a pokud ano, jestli má být **ODD - lichá** nebo **EVEN- sudá**.

2.04 Doba prodlevy. Udává dobu v sekundách, po kterou DSM314 bude čekat na příchod nové zprávy z master zařízení, než bude předpokládat, že došlo ke ztrátě nebo ukončení komunikace. V takovém případě se DSM314 znovu inicializuje pro spuštění nové sekvence připojení SNP.

2.05 Doba obrátky modemu. Když se používá modem, je nutno zadat dobu obrátky modemu. Je to doba vyžadovaná modemem pro zahájení datového přenosu po obdržení požadavku na vysílání. Pokud se modem nepoužívá, je nutno zadat 0. Pokud se modem používá, je nutno zadat hodnotu větší než 0.

2.06 SNP ID. Identifikátor skládající se z 0 až 7 znaků A-F a 0-9. První zadaný znak musí být písmeno A-F. Identifikátor se musí použít pro vícebodové sítě. DSM314 bude podporovat vícebodové připojení, pouze pokud připojení RS232 bude převedeno na RS422/485.

Poznámka: Protože tento sériový komunikační port se používá pouze pro aktualizaci firmwaru DSM314, doporučuje se ponechat výchozí hodnoty nastavení komunikačního portu. Pro připojení tohoto portu k sériovému portu osobního počítače, na kterém běží software s aktualizovaným firmwarem, použijte kabel IC693CBL316.

Řídicí (CTL) bity

Konfigurační záložka Bity CTL umožňuje uživateli nakonfigurovat zdroj vstupu pro Řídicí bity (CTL01-CTL24). Konfigurační obrazovka umožňuje uživateli zvolit konfigurační bit CTL, který odpovídá požadavkům pohybového programu a programu lokální logiky. Konfigurační parametry Bity CTL jsou popsané v Tabulka 4-5. . Další informace týkající se konfiguračního bitu CTL najdete v kapitole 14.

Tabulka 4-5. Záložka Bity CTL

Konfigurační parametr	Popis	Výchozí	Odstavec
CTL01 Config	Konfigurační bit CTL01	IN9_A (osa 1 +OT)	Kapitola 14
CTL02 Config	Konfigurační bit CTL02	IN10_A (osa 1 -OT)	Kapitola 14
CTL03 Config	Konfigurační bit CTL03	IN11_A (spínač výchozí polohy osy 1)	Kapitola 14
CTL04 Config	Konfigurační bit CTL04	Úroveň vzorkování 1 (osa 1)	Kapitola 14
CTL05 Config	Konfigurační bit CTL05	IN9_B (osa 2 +OT)	Kapitola 14
CTL06 Config	Konfigurační bit CTL06	IN10_B (osa 2 -OT)	Kapitola 14
CTL07 Config	Konfigurační bit CTL07	IN 11_B (spínač výchozí polohy osy 2)	Kapitola 14
CTL08 Config	Konfigurační bit CTL08	Úroveň vzorkování 1 (osa 2)	Kapitola 14
CTL09 Config	Konfigurační bit CTL09	%Q bit Offset 12	Kapitola 14
CTL10 Config	Konfigurační bit CTL10	%Q bit Offset 13	Kapitola 14
CTL11 Config	Konfigurační bit CTL11	%Q bit Offset 14	Kapitola 14
CTL12 Config	Konfigurační bit CTL12	%Q bit Offset 15	Kapitola 14
CTL13 Config	Konfigurační bit CTL13	IN9_C (osa 3 +OT)	Kapitola 14
CTL14 Config	Konfigurační bit CTL14	IN10_C (osa 3 -OT)	Kapitola 14
CTL15 Config	Konfigurační bit CTL15	IN 11_C (spínač výchozí polohy osy 3)	Kapitola 14
CTL16 Config	Konfigurační bit CTL16	Úroveň vzorkování 1 (osa 3)	Kapitola 14
CTL17 Config	Konfigurační bit CTL17	%Q bit Offset 24	Kapitola 14
CTL18 Config	Konfigurační bit CTL18	%Q bit Offset 25	Kapitola 14
CTL19 Config	Konfigurační bit CTL19	%Q bit Offset 40	Kapitola 14
CTL20 Config	Konfigurační bit CTL20	%Q bit Offset 41	Kapitola 14
CTL21 Config	Konfigurační bit CTL21	%Q bit Offset 56	Kapitola 14
CTL22 Config	Konfigurační bit CTL22	%Q bit Offset 57	Kapitola 14
CTL23 Config	Konfigurační bit CTL23	%Q bit Offset 72	Kapitola 14
CTL24 Config	Konfigurační bit CTL24	%Q bit Offset 73	Kapitola 14

Každý bit CTL uvedený v předchozí tabulce je možno nakonfigurovat na některou hodnotu v následující tabulce.

Tabulka 4-6. Přípustné hodnoty pro kartu bitů CTL

Řízení lokální logikou	IN9_D (osa 4 +OT)	Úroveň vzorkování 2 (osa 4)	%Q bit Offset 57
IN9_A (osa 1 +OT)	IN10_D (osa 4 +OT)	%Q bit Offset 12	%Q bit Offset 72
IN10_A (osa 1 -OT)	IN11_D (spínač výchozí polohy osy 4)	%Q bit Offset 13	%Q bit Offset 73
IN11_A (spínač výchozí polohy osy 1)	Úroveň vzorkování 1 (osa 1)	%Q bit Offset 14	FBSA* Zápisový bit 1
IN9_B (osa 2 +OT)	Úroveň vzorkování 2 (osa 1)	%Q bit Offset 15	FBSA* Zápisový bit 2
IN10_B (osa 2 -OT)	Úroveň vzorkování 1 (osa 2)	%Q bit Offset 24	FBSA* Zápisový bit 3
IN 11_B (spínač výchozí polohy osy 2)	Úroveň vzorkování 2 (osa 2)	%Q bit Offset 25	FBSA* Zápisový bit 4
IN9_C (osa 3 +OT)	Úroveň vzorkování 1 (osa 3)	%Q bit Offset 40	Příznak aktivní lokální logiky
IN10_C (osa 3 -OT)	Úroveň vzorkování 2 (osa 3)	%Q bit Offset 41	
IN 11_C (spínač výchozí polohy osy 3)	Úroveň vzorkování 1 (osa 4)	%Q bit Offset 56	

* FBSA je zkratka pro "Fast Backplane Status Access" (Přístup ke stavu rychlé vnitřní sběrnice) (Service Request #46). Podrobnosti viz GFK-0467L nebo pozdější.

Výstupní bity

Záložka konfigurace výstupních bitů umožňuje uživateli nakonfigurovat digitální výstupy čelní desky DSM314 buď na řízení programem lokální logiky nebo na program PLC. Parametry výstupních bitů jsou popsané v Tabulka 4-7. Další informace týkající se konfigurace výstupních bitů najdete v kapitole 14.

Tabulka 4-7. Záložka výstupních bitů

Konfigurační parametr	Popis	Hodnoty	Výchozí	Odstavec
Out1_A Config	Zdroj řízení pro Out1_A	Řízení PLC (%Q bit Offset 24) Řízení DSM (digitální výstup 1_1)	Řízení PLC	Kapitola 14
Out3_A Config	Zdroj řízení pro Out3_A	Řízení PLC (%Q bit Offset 25) Řízení DSM (digitální výstup 3_1)	Řízení PLC	Kapitola 14
Out1_B Config	Zdroj řízení pro Out 1_B	Řízení PLC (%Q bit Offset 40) Řízení DSM (digitální výstup 1_2)	Řízení PLC	Kapitola 14
Out3_B Config	Zdroj řízení pro Out 3_B	Řízení PLC (%Q bit Offset 41) Řízení DSM (digitální výstup 3_2)	Řízení PLC	Kapitola 14
Out 1_C Config	Zdroj řízení pro Out 1_C	Řízení PLC (%Q bit Offset 56) Řízení DSM (digitální výstup 1_3)	Řízení PLC	Kapitola 14
Out 3_C Config	Zdroj řízení pro Out 3_C	Řízení PLC (%Q bit Offset 57) Řízení DSM (digitální výstup 3_3)	Řízení PLC	Kapitola 14
Out1_D Config	Zdroj řízení pro Out 1_D	Řízení PLC (%Q bit Offset 72) Řízení DSM (digitální výstup 1_4)	Řízení PLC	Kapitola 14
Out3_D Config	Zdroj řízení pro Out 3_D	Řízení PLC (%Q bit Offset 73) Řízení DSM (digitální výstup 3_4)	Řízení PLC	Kapitola 14

Konfigurační data osy

Parametry pro konfiguraci osy DSM314 definují takové údaje, jako například poměr uživatelských jednotek na puls, rychlost jogu, zrychlení jogu, konec posuvu a meze rychlosti. Jsou zde definované a krátce popsané konfigurační parametry pro jednotlivé režimy řízení smyčky. Čísla ve sloupci "Odstavec" udávají číslo referenčního odstavce v této kapitole. Hodnoty MaxPosnUu, MaxVelUu a MaxAccUu v následující tabulce je možno vypočítat pomocí vztahů v tabulce 4-12 (“Výpočet proměnných pro meze dat”).

Tabulka 4-8. Konfigurační data osy

Konfigurační parametr	Popis	Hodnoty	Výchozí	Jednotky	Odstavec
Uživatelské jednotky	Hodnota uživatelských jednotek	1...65,535	1	---	5.01
Počet	Počet jednotek zpětné vazby	1...65,535	1	---	5.01
Spínač omezení přejezdu	Povolení/zakázání spínače omezení přejezdu	Povoleno Zakázáno	Povoleno	---	5.02
Vstup Pohon povolen	Řízení vstupu Pohon povolen	Povoleno Zakázáno	Povoleno	---	5.03
Horní mez polohy	Horní mez polohy	-MaxPosnUu …+MaxPosnUu-1*	+8388607	Uživatelské jednotky	5.04
Dolní mez polohy	Dolní mez polohy	-MaxPosnUu …+MaxPosnUu-1*	-8388608	Uživatelské jednotky	5.05
Horní softwarová mez konce posuvu	Horní softwarová mez konce posuvu	-MaxPosnUu …+MaxPosnUu-1*	+8388607	Uživatelské jednotky	5.06
Dolní softwarová mez konce posuvu	Dolní softwarová mez konce posuvu	-MaxPosnUu …+MaxPosnUu-1*	-8388608	Uživatelské jednotky	5.07
Softwarový konec posuvu	Řízení softwarového konce posuvu	Zakázáno Povoleno	Zakázáno	---	5.08
Mez rychlosti	Mez rychlosti osy	1…MaxvelUu	1,000,000	Uživatelské jednotky/sekundu	5.09
Směr povelu	Přípustný zadaný směr	Obousměrný Pouze kladný Pouze záporný	Obousměrný	---	5.10
Směr osy	Směr osy	Normální Reverzní	Normální	---	5.11
Zdroj zpětné vazby	Typ zpětné vazby	Výchozí Externí snímač polohy s fázovým posuvem Externí sériový snímač polohy	Výchozí	---	5.12
Režim zpětné vazby (pouze digitální režim)	Režim zpětné vazby	Inkrementální Absolutní	Inkrementální	---	5.13
Kompenzace obrácení chodu	Kompenzace obrácení chodu	0...255	0	Uživatelské jednotky	5.14
Prodleva zákazu pohonu	Prodleva zákazu pohonu	0...60,000	100	ms	5.15
Rychlost jogu	Rychlost jogu	1...MaxVelUu	+1000	Uživ. jedn./s^2	5.16
Zrychlení jogu	Zrychlení jogu	1...MaxAccUu*	+10,000	Uživ. jedn./s^2	5.17

Konfigurační parametr	Popis	Hodnoty	Výchozí	Jednotky	Odstavec
Režim zrychlení jogu	Režim zrychlení jogu	Lineární S křivka	Lineární	---	5.18
Výchozí poloha	Výchozí poloha	Dolní mez polohy ... Horní mez polohy	0	Uživatelské jednotky	5.19
Offset výchozí polohy	Hodnota offsetu výchozí polohy	-32,768...+32,767	0	Uživatelské jednotky	5.20
Konečná rychlost výchozí polohy	Konečná rychlost výchozí polohy	1...MaxVelUu*	+500	Uživ. jedn./s^2	5.21
Rychlost pro nalezení výchozí polohy	Rychlost pro nalezení výchozí polohy	1...MaxVelUu*	+2000	Uživ. jedn./s^2	5.22
Režim výchozí polohy	Režim nalezení výchozí polohy	Spínač výchozí polohy Posuv + Posuv -	Spínač výchozí polohy	---	5.23
Režim dat návratu 1	Režim dat návratu 1	0…FF	0		5.24
Offset dat návratu 1	Offset dat návratu 1	-2,147483,648 až 2,147,483,647	0		5.24
Režim dat návratu 2	Režim dat návratu 2	0…FF	0		5.24
Offset dat návratu 2	Offset dat návratu 2	-2,147483,648 až 2,147,483,647	0		5.24
Master zdroj vačky	Master zdroj vačky	Zadaná poloha 1 Aktuální poloha 1 Zadaná poloha 2 Aktuální poloha 2 Zadaná poloha 3 Aktuální poloha 3 Zadaná poloha 4 Aktuální poloha 4	Aktuální poloha 3	---	5.25
Řídicí smyčka vlečené osy	Povolení řídicí smyčky vlečené osy	Zakázáno Povoleno	Zakázáno	---	5.26
Hodnota poměru A	Poměr A vlečené osy A/B	-32768...+32767	1	---	5.27
Hodnota poměru B	Poměr B vlečené osy A/B	1...32767	1	---	5.27
Master zdroj vlečené osy 1	Master zdroj vlečené osy 1	Žádná Zadaná poloha 1 Aktuální poloha 1 Zadaná poloha 2 Aktuální poloha 2 Zadaná poloha 3 Aktuální poloha 3 Zadaná poloha 4 Aktuální poloha 4	Žádná	---	5.28
Master zdroj vlečené osy 2	Master zdroj vlečené osy 2	Žádná Zadaná poloha 1 Aktuální poloha 1 Zadaná poloha 2 Aktuální poloha 2	Žádná	---	5.29

Konfigurační parametr	Popis	Hodnoty	Výchozí	Jednotky	Odstavec
		Zadaná poloha 3 Aktuální poloha 3 Zadaná poloha 4 Aktuální poloha 4			
Start povolení vlečené osy	Vstup startu povolení vlečené osy	Žádná CTL01-CTL32	Žádná	---	5.30
Start zákazu vlečené osy	Vstup startu povolení vlečené osy	Žádná CTL01-CTL32	Žádná	---	5.31
Zákaz činnosti vlečené osy	Zákaz činnosti vlečené osy	Stop Inkrementální poloha Absolutní poloha	Stop	---	5.32
Zrychlení pozvolného náběhu	Zrychlení pozvolného náběhu vlečené osy	1...MaxAccUu*	10,000	Uživ. jedn./s^2	5.33
Režim pozvolného náběhu	Režim pozvolného náběhu vlečené osy	Čas úpravy náběhu Rychlost úpravy náběhu	Čas úpravy náběhu	---	5.34
Doba pozvolného náběhu	Doba zrychlení pozvolného náběhu vlečené osy	0…32000	0	ms	5.35
Rychlost pozvolného náběhu	Rychlost pozvolného náběhu vlečené osy	1...MaxVelUu*	+100,000	Uživ. jedn./s^2	5.36

* Výpočet MaxAccUu, MaxPosnUu, and MaxVelUu viz tabulka 4-8.

5.01 Uživatelské jednotky, Jednotky. Poměr uživatelských jednotek na jednotku nastaví počet programovacích jednotek na jednotku polohové zpětné vazby. To umožní uživateli naprogramovat DSM314 v jednotkách specifických pro aplikaci. Hodnoty uživatelských jednotek a Jednotek musí být v rozsahu 1 až 65,535. Poměr uživatelských jednotek na jednotku musí být rozsahu 8:1 až 1:32. Pokud například na pulsů 8192 zpětné vazby se vykoná posuv 1 000 palců, poměr Uživatelská jednotka:Jednotka o hodnotě 1000:8192 bude představovat 1 Uživatelskou jednotku na 0.001 palec. Výchozí hodnota je 1:1.

Poměr uživatelských jednotek na jednotku nastaví počet polohových programovacích jednotek na jednotku zpětné vazby. Je nutné, aby tato hodnota byla správně nastavená pro mechanické systémy spojené s osou, jinak pohyb nebude bezpečný a může dojít k nepřesné poloze.

Poznámka: ***Je důležité tento vztah nastavit na začátku konfigurace;*** většina ostatních konfiguračních polí je udávaná v uživatelských jednotkách.

Například rychlost se zadává v uživatelských jednotkách za sekundu a zrychlení se zadává v uživatelských jednotkách za sekundu za sekundu.

Tento poměr je velmi výkonná funkce změny měřítka. Poměr Uživatelské jednotky na Jednotku je možno nakonfigurovat tak, aby se umožnilo programování v jiných než výchozích jednotkách. Předpokládejme ve zjednodušeném příkladu použití snímače polohy ve zpětné vazbě, který dává 1000 fázově posunutých jednotek na otáčku (250 pulsů) a je spřažený se strojem, který vykoná posuv 1 palec na otáčku. Výchozí jednotka bude jedna tisícina palce na jednotku. Můžete ale chtít psát programy a používat modul DSM300 Series s metrickými jednotkami. K tomu je možno nakonfigurovat poměr 2540 Uživatelských jednotek na 1000 jednotek. S tímto poměrem bude

jedna uživatelská jednotka představovat 0,01 milimetru. 2540 uživatelských jednotek bude představovat 25.40 milimetrů (jeden palec) posuvu.

Následující příklad znázorňuje, jak splnit požadavky, aby hodnoty Uživatelské jednotky a Jednotky byly v rozmezí 1 až 65 535 a poměr Uživatelské jednotky k Jednotkám v rozmezí 8:1 až 1:32.

Základní rovnice pro tento požadavek je:

$$\frac{\text{Uživatelské jednotky}}{\text{Jednotky}} = \frac{\text{(Posuv zátěže na otáčku motoru)} \div \text{(Požadované rozlišení uživatelských jednotek)}}{\text{Počet jednotek snímače polohy na otáčku motoru}}$$

Čitatel a jmenovatel musí být v mezích ROZSAHU. Zkrácený zlomek musí být v mezích POMĚRU. Desetinná tečka je vždy zahrnutá, ale nepoužívá se. Poměr Uživatelské jednotky k Jednotkám je vždy vyjádřený jako celočíselný poměr.

Příklad použití

K nakonfigurování DSM314 použijte poměr Uživatelské jednotky k Jednotkám tak, abyste mohli programovat v technických jednotkách a ne v jednotkách snímače polohy. Jako příklad předpokládejme, že stroj má motor s namontovaným snímačem polohy s fázovým posunutím připojeným přes převod do pomala na čelní ozubené kolo. Čelní ozubené kolo je namontované na konec hřídele unášecího válečku. Unášecí váleček posouvá plech pro stříhání na danou délku. Pohybový program určí délku stříhaného plechu. Programátor chce programovat s rozlišením 0,01 palce.

Jsou dána následující data:

- Snímač polohy s 2000 pulsy (×4 = 8000 pulsů na otáčku snímače polohy)
- Převod do pomala 20:1
- Čelní ozubené kolo s roztečí 14,336 palce
- Požadováno programování v palcových jednotkách (0,01)

I když existuje několik možností, nejpřímější je vycházet při výpočtu z jedné otáčky čelního ozubeného kola.

1. Nejdříve určete počet uživatelských jednotek na otáčku čelního ozubeného kola:

   Průměr roztečné kružnice14,336 palce * π (pi) = obvod 45,0378 palce

   Požadované programovací jednotky 45,0378 palce / 0,01 palce = 4503,78 uživatelských jednotek na otáčku čelního ozubeného kola

2. Pak určete počet pulsů snímače polohy na otáčku čelního ozubeného kola:

$$2000 \text{ pulsů} * \frac{4 \text{ pulsy}}{\text{puls}} * \frac{20 \text{ otáček motoru}}{1 \text{ otáčka převodu}} = 160\,000 \text{ pulsů snímače polohy na otáčku čelního ozubeného kola}$$

3. Pak zkontrolujte hodnotu poměru uživatelských jednotek k jednotkám. Poměr musí být v rozmezí 8:1 až 1:32 (8 až 0,03125) a tato dvě čísla musí být v rozsahu 1 až 65535.

   4503,78 uživatelských jednotek / 160 000 pulsů snímače polohy = 0,02815 nebo 1:35,5

Tento poměr je příliš malý, takže je nutno něco změnit. K vyřešení problému je možno změnit kteroukoliv z následujících složek systému:

- Změnit průměr čelního ozubeného kola na 15,92 palce nebo větší.
- Změnit počet pulsů na otáčku snímače polohy na 1800 nebo méně
- Změnit převodový poměr na 18:1 nebo menší
- Zvolit požadovanou programovací jednotku na 0,001 palce

Zdaleka nejsnazší je provést změnu u programovací jednotky na 0,001 palce.

4. Nyní novým výpočtem určete upravené uživatelské jednotky na otáčku pomocí programovací jednotky 0,001 palce.

   Průměr 14,336 palce * pi = obvod 45,0378 palce

   45,0378 palce / programovací jednotka 0,001 palce = 45 037,8 uživatelských jednotek na otáčku čelního ozubeného kola

   Poměr Uživatelských jednotek k Jednotkám tedy bude 45 038 / 160 000 = 0,2815 nebo asi 1:3,6, což je v platném rozsahu poměru.

   Takže poměr 45 038 / 160 000 by se mohl použít, jenže 160 000 je větší než maximální hodnota rozsahu 65 535. Problém je možno vyřešit vydělením obou čísel 10 a vznikne poměr 4 504 / 16 000. Všimněte si, že v příkladu výše jsme jednoduše zkrátili zlomek a zanedbali jsme malou chybu zaokrouhlením.

   Jedním ze způsobů, jak se vyhnout chybě "zaokrouhlením", je vyjádřit číselný poměr jako zlomek. Z předchozího příkladu je možno použít každé číslo, které dá poměr 0,2815. Příkladem může být 2815 / 10000.

   Jiný způsob je racionalizace zlomku (zkrátit ho na nejmenší činitele). To se provede rovnoměrným dělením čitatele a jmenovatele postupně menšími prvočísly, přičemž se začne největším prvočíslem, které bude rovnoměrně dělit čitatele a jmenovatele, dokud nebude možné provést další dělení beze zbytku.

Když budete provádět nastavení poměru Uživatelských jednotek k Jednotkám na nejlepší přesnost, vždy dodržujte přesný celočíselný zlomek, desetinné číslo vyjádřené jako zlomek nebo racionalizovaný zlomek. Uživatel musí stanovit, jestli chyba zaokrouhlováním, pokud vznikne, je významná nebo ne. Použití rotačního režimu, který vždy pracuje jedním směrem, bude mít za následek akumulaci chyby zaokrouhlením v průběhu doby a "drift". Lineární aplikace bude pouze akumulovat chybu podél dráhy posuvu a pak se při obrácení osy "vrátí".

5.02 Koncový spínač přejetí. Zvolí, jestli modul DSM300 Series používá hardwarové vstupy koncového spínače přejetí.

ZAKÁZÁNO, vstupy přejetí na čelní desce (IN09 a IN10) je možno použít jako univerzální vstupy pro řízení toku pohybového programu a větvení programu (označené CTL01-CTL24).

POVOLENO, označuje, že DSM300 bude vstupy přejetí osy kontrolovat souvisle každých 10 milisekund vždy, když vstup %I *Pohon povolen* bude v jedničce. Pokud se některý z koncových spínačů rozpojí (vstup přejde do logické nuly, vypnuto), všechny pohyby se okamžitě zastaví. Neprovádí se žádné zpomalování; povel rychlosti serva je nastavený na nulu. Polovodičové relé povolení osy se nerozpojí, dokud povel *%Q Povolení pohonu* nebude nastavený na nulu. Chybový

kód udávající, který koncový spínač se rozepnul, se předá do %AI *Chybový kódu osy*. V tomto okamžiku je přípustná pouze jedna akce DSM314: odpovídající bity %Q *Jog* a %Q *Vynulovat chybu* je možno použít současně pro odjetí z koncového spínače. Aby bylo možno provést jogování mimo koncový spínač, bit %Q *Vynulovat chybu* musí zůstat v logické jedničce. Uživatel také může provést ruční posuv zakázané osy mimo koncový spínač. Po vynulování alarmu je možno obnovit normální činnost.

Upozornění

Vynucené povely pro D/A převodníky ignorují koncové spínače a musí se používat opatrně.

5.03 Vstup Pohon povolen. Povolí nebo zakáže vstup Pohon povolen pro analogová serva. Tento konfigurační údaj se pro digitální servo nebo pro pomocnou osu ignoruje. Pokud vstup Pohon připravený bude povolený, vstupní signál čelní desky Pohon připravený (IN4) se musí sepnout (nastavit na 0 V) **během 1 sekundy** po přechodu bitu *Povolit pohon* %Q do jedničky. Pokud vstup na čelní desce Pohon připraven přejde do nuly, když bit *Pohon povolen* %I bude v jedničce, bude se hlásit chybový kód C0h a osa se zastaví. Pro analogová serva, která nemají kompatibilní výstupní signál Pohon připraven, vstup Pohon povolen musí být nastavený na Zakázáno.

5.04 Horní mez polohy. (Uživatelské jednotky). Když se při pohybu v kladném směru dosáhne dolní meze, okamžitá poloha se přetočí na tuto hodnotu. **Meze polohy je možno použít pro aplikace souvislého otáčení, když softwarový konec pohybu bude nastavený na Zakázáno.** Horní mez polohy je nutno vždy nastavit o jednu uživatelskou jednotku menší než je požadovaný cyklus. Například, stroj 360° bude mít Horní mez polohy nastavenou na 359. Při další jednotce za 359 se počet přetočí na hodnotu nastavenou v parametru Dolní mez polohy (v tomto příkladu 0). **Pro správnou funkci modul přetočení (Horní mez polohy - Dolní mez polohy +1) musí být vždy větší než vzdálenost, kterou vykoná osa během jedné vzorkovací doby polohové smyčky (normálně 2 ms).** V Dodatku C najdete poznámky týkající se používání absolutního snímače polohy. Výchozí: 8 388 607.

5.05 Dolní mez polohy. Dolní mez polohy (Uživatelské jednotky). Když se při pohybu v záporném směru dosáhne horní meze, okamžitá poloha se přetočí na tuto hodnotu. **Meze polohy je možno použít pro aplikace souvislého otáčení, když softwarový konec pohybu bude nastavený na Zakázáno. Pro správnou funkci modul přetočení (Horní mez polohy - Dolní mez polohy +1) musí být vždy větší než vzdálenost, kterou vykoná osa během jedné vzorkovací doby polohové smyčky (normálně 2 ms).** V dodatku C najdete poznámky týkající se používání absolutního snímače polohy. Výchozí: -8,388,608.

5.06 Horní softwarová mez konce posuvu. Horní softwarová mez konce posuvu (Uživatelské jednotky). Pokud mez bude povolená a DSM314 bude naprogramovaný na posuv do polohy větší než hodnota horní softwarové meze konce posuvu, bude se generovat chyba a DSM314 tento pohyb nepovolí**.** Pokud smyčka řízení vlečené osy bude povolena, horní softwarová mez konce posuvu se pro pohyb podřízené osy, který je výsledkem povelu nadřízené osy, bude ignorovat. Mez bude platit pouze pro pohyb podřízené osy, který je výsledkem interně generovaného jogu a povelů pohybových programů. **Mez se bude ignorovat pro %AQ povely *Pohyb rychlostí***. Výchozí: +8 388 607.

V režimech analogového nebo digitálního serva se horní softwarová mez konce posuvu používá, pouze když ***softwarový konec posuvu*** bude nastavený na Povoleno. Pokud horní softwarová mez konce posuvu bude povolená a její hodnota bude větší než horní mez polohy, horní softwarová mez konce posuvu se interně nastaví na horní mez polohy. Současně se bude hlásit chyba osy 17h, která

indikuje, že se upravila mez. **Horní softwarová mez konce posuvu se bude ignorovat pro povely *Jog*, pokud %I bit *Poloha platná* bude v nule**.

V režimu pomocné osy horní softwarová mez konce posuvu má jiné účely, které závisí na nastavení ***softwarového konce posuvu***:

Softwarový konec posuvu nastavený na Povoleno – povely pohybových programů a *Jog* jsou omezené na hodnotu horní softwarové meze konce posuvu. Povel %AQ *Pohyb rychlostí* může mít za následek, že *Zadaná poloha* přesáhne mez konce posuvu. *Zadaná poloha* se na hodnotách maximální kladné a maximální záporné polohy (-2 147 483 648 … +2 147 483 647 při měřítku 1:1) nepřetočí.

Softwarový konec posuvu nastavený na Zakázáno – Horní softwarová mez konce posuvu se používá jako hodnota přetočení pro *Zadanou polohu*. Povely pohybového programu, *Jog* a *Pohyb rychlostí* budou mít za následek, že *Zadaná poloha* se přetočí na horní softwarovou mez konce posuvu.

5.07 Dolní softwarová mez konce posuvu. Dolní softwarová mez konce posuvu (Uživatelské jednotky). Pokud mez bude povolená a DSM314 bude naprogramovaný na posuv do polohy menší než hodnota dolní softwarové meze konce posuvu, bude se generovat chyba a DSM314 tento pohyb nepovolí**.** Pokud smyčka řízení vlečené osy bude povolena, horní softwarová mez konce posuvu se pro pohyb podřízené osy, který je výsledkem povelu nadřízené osy, bude ignorovat. Mez bude platit pouze pro pohyb podřízené osy, který je výsledkem interně generovaného jogu a povelů pohybových programů. **Mez se bude ignorovat pro %AQ povely *Pohyb rychlostí***. Výchozí: -8 388 608

V režimech analogového nebo digitálního serva se ***dolní softwarová mez konce posuvu používá, pouze když*** softwarový konec posuvu bude nastavený na Povoleno. Pokud dolní softwarová mez konce posuvu bude povolená a její hodnota bude menší než dolní mez polohy, dolní softwarová mez konce posuvu se interně nastaví na dolní mez polohy. Současně se bude hlásit chyba osy 17h, která indikuje, že se upravila mez. **Dolní softwarová mez konce posuvu se bude ignorovat pro povely *Jog*, pokud %I bit *Poloha platná* bude v nule**.

V režimu pomocné osy má dolní softwarová mez konce posuvu jiné účely, které závisí na nastavení ***softwarového konce posuvu***:

Softwarový konec posuvu nastavený na Povoleno – povely pohybových programů a *Jog* jsou omezené na hodnotu dolní softwarové meze konce posuvu. %AQ povel *Pohyb rychlostí* může mít za následek, že *Zadaná poloha* přesáhne mez konce posuvu. *Zadaná poloha* se na hodnotách maximální kladné a maximální záporné polohy (-2 147 483 648 … +2 147 483 647 při měřítku 1:1) nepřetočí.

Softwarový konec posuvu nastavený na Zakázáno – Dolní softwarová mez konce posuvu se používá jako hodnota přetočení pro *Zadanou polohu*. Povely pohybového programu, *Jog* a *Pohyb rychlostí* budou mít za následek, že *Zadaná poloha* se přetočí na dolní softwarovou mez konce posuvu.

5.08 Softwarový konec posuvu. Povolí nebo zakáže Horní softwarovou mez konce posuvu a Dolní softwarovou mez konce posuvu. Výchozí: Zakázáno

5.09 Mez rychlosti. Mez rychlosti osy (Uživatelské jednotky/sekundu). Mez rychlosti platí pro součet všech zdrojů rychlostních povelů pro osu včetně povelů interního generátoru dráhy a povelů nadřízené osy pro vlečenou osu. Pokud povel rychlosti serva bude přesahovat meze, **bude se hlásit chybový kód F2h** a povel serva se interně nastaví na mezní hodnotu. Výchozí: 1 000 000

5.10 Směr povelu. Umožňuje nakonfigurovat osu na jednosměrný nebo obousměrný provoz. Pokud bude zvolený jednosměrný provoz (pouze kladný nebo pouze záporný), povely serva v opačném směru se do serva polohové smyčky neodešlou. Výchozí: Obousměrný

5.11 Směr osy. Pro všechny digitální serva GE Fanuc směr osy nakonfigurovaný jako Normální definuje kladný směr osy při pohledu na hřídel motoru jako otáčení hřídele motoru proti směru hodinových ručiček (CCW). Směr osy nakonfigurovaný jako Reverzní definuje kladný směr osy jako otáčení hřídele ve směru hodinových ručiček (CW).

U analogových serv směr osy nakonfigurovaný jako Normální definuje kladný směr osy jako kanál A snímače polohy před kanálem B. Směr osy nakonfigurovaný jako reverzní definuje kladný směr osy jako kanál B snímače polohy před kanálem A. V praxi nakonfigurování směru osy umožňuje uživateli snadno změnit její směr všemi povely, aniž by bylo nutno měnit pohybový program. Výchozí: Normální

5.12 Zdroj zpětné vazby. Tento konfigurační údaj se v současném firmwaru DSM314 nepoužívá. Musí být nastavený na Výchozí.

5.13 Režim zpětné vazby. Používá se pouze když režim osy je nastavený na Digitální servo. Tento údaj nakonfiguruje inkrementální nebo absolutní typ zpětné vazby pro sériový snímač polohy GE Fanuc. Inkrementální znamená, že sériový snímač polohy se používá jako Inkrementální snímač polohy a alarm baterie snímače polohy se nehlásí. Absolutní znamená, že sériový snímač polohy se používá jako absolutní snímač polohy (baterie pro zálohování snímače polohy je nainstalovaná), který udržuje polohu při vypnutí a zapnutí napájení systému. V absolutním režimu se hlásí alarm baterie snímače polohy. Více informací najdete v Dodatku C *Zařízení polohové zpětné vazby*. Výchozí: Inkrementální

5.14 Kompenzace obrácení chodu. Činitel kompenzace, který umožňuje, aby servo obrátilo směr a stále zachovávalo přesnost polohování u systémů, které mají mrtvý chod. Mrtvý chod má servomotor, který musí vykonat pohyb o malou vzdálenost (mrtvý chod) před tím, než se po obrácení směru břemeno začne pohybovat. Předpokládejme například šroub zámku dveří. Představme si, že servo ovládá klíč v zámku a zpětná vazba hlásí pohyb šroubu. Když servo otočí klíčem proti směru hodinových ručiček, šroub vykoná pohyb doleva. Když však servo otočí klíčem ve směru hodinových ručiček, šroub se nepohne, dokud se klíč nenatočí do určitého bodu. Funkce kompenzace obrácení chodu přidá nutný mrtvý chod tak, aby se servo rychle posunulo tam, kde u zařízení zpětné vazby má začít pohyb. DSM314 odstraní kompenzační vzdálenost, když se zadá pohyb v záporném směru, a kompenzační vzdálenost přičte před vykonáním pohybu v kladném směru. Výchozí: 0.

> **Poznámka:** Kompenzaci obrácení chodu nelze použít, když ***Řídicí smyčka vlečené osy*** bude nastavena na Povoleno.

5.15 Prodleva zákazu pohonu. Prodleva zákazu pohonu serva (milisekundy). Časová prodleva od doby příchodu povelu nulové rychlosti do přechodu signálu povolení serva (digitální servo MCON) do nuly. Zakázání prodlevy bude mít účinek, když %Q bit *Povolení pohonu* bude v nule nebo se vyskytnou určité chybové stavy (režim Stop). Zákaz prodlevy nesmí být delší než nejhorší případ doby zpomalení serva z maximální rychlosti. Protože nastavení %Q bitu *Povolení pohonu* do nuly zabrání, aby DSM314 generovalo povely pro servo, existují okamžiky, kdy signál povolení pohonu musí zůstat v jedničce. Pokud se například servo bude pohybovat na mez konce pohybu a signál povolení pohonu v důsledku chyby okamžitě přejde do nuly, servo může pokračovat v pohybu, dokud se nezastaví. Aby proto DSM314 mělo možnost zadat povel a řídit rychlé zastavení, prodleva zákazu pohonu musí být delší než doba zpomalení serva z maximální rychlosti.

Prodleva zákazu se může použít k řízení, když se z hřídele motoru odejme krouticí moment. Aplikace používající elektromechanickou brzdu obecně potřebují čas na to, aby se před uvolněním krouticího momentu serva brzda dostala do záběru. Prodlevu je nutno nastavit na hodnotu delší, než kterou potřebuje brzda na to, aby se dostala do záběru. Výchozí: 100.

5.16 Rychlost jogu. Rychlost jogu (Uživatelské jednotky/sekundu). Rychlost, kterou se servo pohybuje během operace *Jog*. *Rychlost jogu* používají pohybové programy, když program neobsahuje žádný povel rychlosti. Povel pohybu %AQ (27h) používá vždy *Rychlost jogu*. Výchozí: 1000.

5.17 Zrychlení jogu. Velikost zrychlení jogu (Uživatelské jednotky/sekundu/sekundu). Velikost zrychlení a zpomalení během operací *Jog*, *Nájezd do výchozí polohy*, *Pohyb rychlostí, Zrušení všech pohybů* a **Normální zastavení**. K **Normálnímu zastavení** dojde, když PLC přejde ze stavu Run do stavu Stop nebo po určitých chybách programování (viz Dodatek A). *Zrychlení jogu* používají pohybové programy, když v programu nebude povel zrychlení. Povel pohybu %AQ (27h) používá vždy *Zrychlení jogu*. Hodnota *Zrychlení jogu* musí být nastavená dostatečně vysoká, aby se během operací *Zrušit všechny pohyby* a **Normální zastavení** chovala uspokojivě. Výchozí: 10000.

> **Poznámka:** V DSM314 se používá minimální hodnota po změně měřítka. Tato hodnota je určena pravidlem: Zrychlení jogu * (uživatelské jednotky/jednotku) >= 32 jednotek/sec/sec.

5.18 Režim zrychlení jogu. Režim zrychlení jogu (LINEÁRNÍ nebo S-KŘIVKA). Režim zrychlení pro operace *Jog*, *Nájezd do výchozí polohy*, *Pohyb rychlostí, Zrušení všech pohybů* a **Normální zastavení**. K **Normálnímu zastavení** dojde, když PLC přejde ze stavu Run do stavu Stop nebo po určitých chybách programování (viz Dodatek A). LINEÁRNÍ (konstanta zrychlení) bude mít za následek, že zadaná rychlost se bude měnit lineárně s časem. S-KŘIVKA (zrychlení s omezeným škubáním) způsobí, že zadaná rychlost se na začátku a na konci intervalu zrychlování bude měnit pomaleji než u lineárního režimu. **Pohyby používající S-křivku zrychlení vyžadují k provedení změny rychlosti dvojnásobek času a vzdálenosti ve srovnání s pohyby používajícími stejnou hodnotu zrychlení s lineárním zrychlením.** Aby se zachovaly stejné doby strojních cyklů, profil pohybu s S-křivkou vyžaduje hodnotu zrychlení (a špičkový kroutící moment motoru) dvakrát tak velký než ekvivalentní profil pohybu s lineárním zrychlením. Proto může být nutný kompromis mezi cenou motoru a dobou strojního cyklu. Výchozí: LINEÁRNÍ.

5.19 Výchozí poloha. Výchozí poloha (Uživatelské jednotky). Hodnota přiřazená *Zadané poloze*, když se dokončí cyklus *Nalezení výchozí polohy*.

5.20 Offset výchozí polohy. Offset výchozí polohy (Uživatelské jednotky). Hodnota přičtená nebo odečtená od konečného bodu zastavení serva, když se dokončí cyklus *Nalezení výchozí polohy*. **Offset výchozí polohy** upraví konečný bod zastavení serva vzhledem ke značce snímače polohy. Podrobnosti k cyklu výchozí polohy najdete v kapitole 6. Výchozí: 0.

5.21 Rychlost pro nalezení výchozí polohy. Rychlost nalezení výchozí polohy (Uživatelské jednotky/sekundu). Rychlost, kterou servo hledá počáteční přechody spínače výchozí polohy během cyklu *Nalezení výchozí polohy*, když spínač výchozí polohy je nastavený na HOMESW. Pokud bude zapotřebí, Rychlost pro nalezení výchozí polohy je možno nastavit na vysokou hodnotu a umožnit tak, aby servo rychle nalezlo Spínač výchozí polohy. Výchozí: 2000

5.22 Konečná rychlost výchozí polohy. Konečná rychlost výchozí polohy (Uživatelské jednotky/sekundu). Rychlost, kterou servo hledá konečný přechod spínače výchozí polohy a značkovací puls snímače polohy na konci cyklu Nalezení výchozí polohy. Tato rychlost se také používá pro režimy nájezdu do výchozí polohy MOVE+ a MOVE-. Podrobnosti k cyklu výchozí

polohy najdete v kapitole 6. Konečná rychlost výchozí polohy musí být dostatečně nízká, aby se umožnila prodleva 10 milisekund (doba filtrace) mezi konečným přechodem spínače výchozí polohy a značkovacím pulsem snímače polohy. Výchozí: 500

5.23 Režim výchozí polohy. Režim nalezení výchozí polohy. Metoda používaná k nalezení výchozí polohy během cyklu *Nalezení výchozí polohy*. HOME SWITCH indikuje, že pro nalezení výchozí polohy se má monitorovat Spínač výchozí polohy. MOVE+ a MOVE – udávají přímý pohyb v kladném a záporném směru k další značce snímače polohy při konečné rychlosti výchozí polohy. Podrobnosti o cyklu výchozí polohy, spínači výchozí polohy, režimech Move+ a Move- najdete v kapitole 6, "Neprogramované pohyby". Výchozí: HOMESW.

5.24 Režim a offset dat návratu 1, Režim a offset dat návratu 2. Tyto konfigurační parametry umožňují předávat pro jednotlivé osy alternativní data na adrese %AI *Uživatelská data volby 1* a *Uživatelská data volby 2*. Alternativní data obsahují informace, jako například obsah paměti parametrů a revizi firmwaru DSM314.

Existují dva konfigurační parametry *Data návratu*, volba režimu a volba offsetu. Parametr režimu zvolí typ dat návratu. Parametr offsetu se používá, pouze když je zvolený režim Data parametru (18h). Výchozí režim = 0 (povel kroutícího režimu). Výchozí offset = 0. Jsou přípustné následující volby dat návratu:

Tabulka 4-9. Uživatelem volitelná data návratu

Digitální	Analogový kroutící moment	Analogová rychlost	Volitelná data návratu	Režim dat	Offset dat
Y	Y	N	Povel kroutícího momentu	00h	Nepoužívá se
Y	Y	Y	Revize firmwaru DSM	10h	Nepoužívá se
Y	Y	Y	Identifikační číslo firmwaru DSM (hex)	11h	Nepoužívá se
Y	N	N	Absolutní zpětná vazba (pulsy)	17h	Nepoužívá se
Y	Y	Y	Data parametrů	18h	Číslo parametru (0-255)
Y	Y	Y	Bity CTL 1-32	19h	Nepoužívá se
Y	Y	Y	Analogové vstupy - osa 1	1Ch	Nepoužívá se
Y	Y	Y	Analogové vstupy - osa 2	1Dh	Nepoužívá se
Y	Y	Y	Analogové vstupy - pomocná osa 3	1Eh	Nepoužívá se
Y	Y	Y	Analogové vstupy - pomocná osa 4	1Fh	Nepoužívá se
Y	Y	Y	Zadaná poloha (uživatelské jednotky)	20h	Nepoužívá se
Y	Y	Y	poloha vlečené osy zadaná povelem (pulsy)	21h	Nepoužívá se
Y	Y	Y	Nepřizpůsobená okamžitá poloha (pulsy)	28h	Nepoužívá se
Y	Y	Y	Nepřizpůsobená poloha vzorkování 1 (pulsy)	29h	Nepoužívá se
Y	Y	Y	Nepřizpůsobená poloha vzorkování 2 (pulsy)	2Ah	Nepoužívá se

Povel kroutícího momentu má měřítko nastavené tak, že +/- 10000 = +/- 100% krouticího momentu.

Revize firmwaru DSM je uvedená jako dvě samostatná slova s kódy hlavní-vedlejší revize.

Identifikační číslo firmwaru DSM je uváděno jako jedno slovo v hexadecimálním tvaru.

Absolutní offset zpětné vazby je polohový offset (v jednotkách), který se používá k inicializaci *okamžité polohy*, když se používá digitální absolutní snímač polohy GE Fanuc. *Okamžitá poloha* = data absolutního snímače polohy + offset absolutní zpětné vazby.

Analogové vstupy se skládají ze dvou datových slov pro každou osu: dolní slovo = AIN1 a horní slovo = AIN2. Data mají měřítko nastavené tak, že +/- 32000 = +/- 10.0 V.

Zadaná poloha (uživatelské jednotky) je kopie %AI dat *Zadané polohy* předávaných pro jednotlivé osy. Viz odstavec 2.04 v kapitole 5.

Poloha vlečené osy zadaná povelem (pulsy) je aktualizovaná aktivní zadaná poloha (v jednotkách zpětné vazby) a používaná interním generátorem povelového pohybu. Viz kapitola 9 – Kombinovaný pohyb vlečené osy a zadaný pohyb.

Nepřizpůsobená okamžitá poloha je kumulovaná okamžitá poloha (v jednotkách, ne v uživatelských jednotkách) s 32-bitovou binární hodnotou překlopení -2 147 483 648 … +2 147 483 647.

Nepřizpůsobená poloha vzorkování 1 je hodnota *nepřizpůsobené okamžité polohy* zachycené, když se vyskytne vstup Vzorkování 1.

Nepřizpůsobená poloha vzorkování 2 je hodnota *nepřizpůsobené okamžité polohy* zachycené, když se vyskytne vstup Vzorkování 2.

Než v PLC budou k dispozici nová volitelná data návratu, musí uplynout minimálně tři cykly PLC nebo 10 milisekund (podle toho co je delší).

5.25 Master zdroj vačky. Tento konfigurační údaj se v současném firmwaru DSM314 nepoužívá.

5.26 Řídicí smyčka vlečené osy. Když tento konfigurační údaj bude nastavený na Povoleno, servoosa bude kromě standardních funkcí interně generovaného pohyb sledovat vstup master osy. Výchozí: Zakázáno

5.27 Hodnota poměru A a Hodnota poměru B. (Řídicí smyčka vlečené osy musí být povolena) Poměr A ku B nastavuje převodový poměr slave/master pro vlečenou osu.

$$\text{Pohyb vlečené osy (jednotky)} = \frac{\text{A * master referenční jednotky}}{\text{B}}$$

Rozsah A je –32 768 až +32 767 a rozsah B je 1 až +32 767. Když A bude záporné, podřízená osa se bude pohybovat v opačném směru než nadřízená. Firmware DSM podporuje poměry slave/master vlečené osy A/B v rozsahu 32:1 až 1:10 000. Výchozí: 1:1.

5.28 Master zdroj vlečené osy 1. (Řídicí smyčka vlečené osy musí být povolená) Konfiguruje master zdroj osy 1. Přípustné volby pro každou ze čtyř os (pokud to jsou nakonfigurované osy) jsou zadaná nebo okamžitá poloha. Master zdroj vlečené osy 1 je aktivní, když %Q bit *Volba master zdroje vlečené osy* bude v nule.

Zadaná poloha nebo okamžitá poloha podřízené osy se nesmí pro tuto osu zvolit jako master zdroj. Pokud se pro Master zdroj vlečené osy 1 zvolí nenakonfigurovaná osa, bude se ignorovat. Výchozí: Žádné. Více informací o režimu vlečené osy najdete v kapitole 8.

5.29 Master zdroj vlečené osy 2. (Řídicí smyčka vlečené osy musí být povolená) Konfiguruje master zdroj osy 2. Přípustné volby pro každou ze čtyř os (pokud to jsou nakonfigurované osy) jsou zadaná nebo okamžitá poloha. Master zdroj vlečené osy 2 je aktivní, když %Q bit *Volba master zdroje vlečené osy* bude v jedničce.

Zadaná poloha nebo okamžitá poloha podřízené osy se nesmí pro tuto osu zvolit jako master zdroj. Pokud se pro Master zdroj vlečené osy 2 zvolí nenakonfigurovaná osa, bude se ignorovat. Výchozí: Žádné. Více informací o režimu vlečené osy najdete v kapitole 8.

5.30 Aktivace povolení vlečené osy. Aktivační vstup povolení vlečené osy. Zvolí řídicí bit, CTL01-CTL32, který se používá jako aktivační vstup povolení vlečené osy. Vlečená osa je povolená, když zvolený aktivační vstup přejde do jedničky a %Q bit *Povolení vlečené osy* bude také v jedničce. Po povolení vlečené osy %Q bit *Povolení vlečené osy* PLC a přídavný bit zákazu aktivace vlečené osy řídí aktivní stav následující funkce. Žádná znamená, že vlečená osa je povolená pouze %Q bitem *Povolení vlečené osy*. Výchozí: Žádná.

5.31 Aktivace zákazu vlečené osy. Aktivace zákazu vlečené osy. Zvolí řídicí bit, CTL01-CTL32, který se používá jako aktivační vstup zákazu vlečené osy. Aktivační vstup se testuje, pouze když %Q bit *Povolení vlečené osy* bude v jedničce. Když %Q bit *Povolení vlečené osy* bude v jedničce, přechod aktivačního bitu z nuly do jedničky zakáže vlečenou osu. Nastavení %Q bitu *Povolení vlečené osy* do nuly okamžitě zakáže vlečenou osu bez ohledu na nastavení aktivace zákazu. Výchozí: Žádná.

5.32 Zákaz činnosti vlečené osy. Stop znamená, že vlečená osa se okamžitě začne zpomalovat na nulovou rychlost nastavenou velikostí rampy zrychlování vlečené osy. Inkrementální poloha znamená, že vlečená osa bude pokračovat aktuální rychlostí, pak se zpomalí a zastaví po vykonání zadané vzdálenosti. Inkrementální vzdálenost je zadaná v registru parametrů pro každou osu:

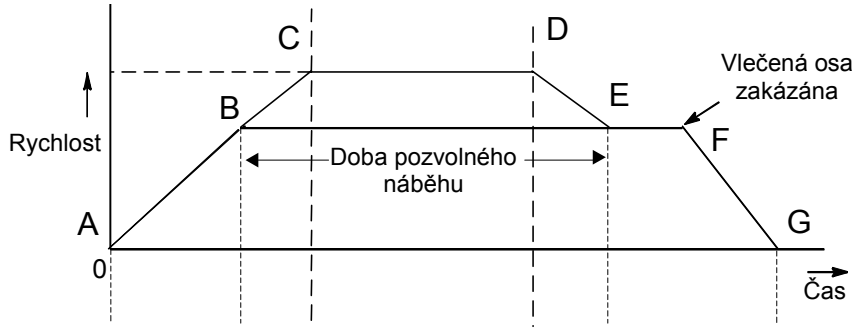
P227 = Inkrementální vzdálenost osy 1
P235 = Inkrementální vzdálenost osy 2
P242 = Inkrementální vzdálenost osy 3
P250 = Inkrementální vzdálenost osy 4

Inkrementální vzdálenost představuje celkovou změnu okamžité polohy, ke které dojde od bodu, kde vlečená osa byla zakázána, až do jejího zastavení.

Současný firmware DSM314 nepodporuje nastavení Absolutní polohy. Výchozí: Stop

5.33 Zrychlení pozvolného náběhu. Zrychlení pozvolného náběhu vlečené osy (uu/sec^2). Určuje zrychlení používané pro:

- Zrychlení vlečené osy tak, aby souhlasilo s master rychlostí po povolení vlečené osy (úsek AB na Obrázek 4-1),
- Úpravu master povelových pulsů ztracených během zrychlování vlečené osy (úsek BC a DE na Obrázek 4-1),
- Zpomalení a zastavení po zakázání vlečené osy (úsek FG na Obrázek 4-1).



Obrázek 4-1. Profil rychlosti během cyklu rampy vlečené osy

5.34 Režim pozvolného náběhu. Možnosti jsou následující Doba pozvolného náběhu nebo Rychlost pozvolného náběhu.

- **Režim doby pozvolného náběhu** – v tomto režimu proces pozvolného náběhu vezme dobu zadanou v parametru Doba pozvolného náběhu (viz Obrázek 4-1). To je výchozí režim.
- **Režim pozvolného náběhu rychlosti** – tento režim je vyhrazený pro budoucí použití.

5.35 Doba pozvolného náběhu. Doba pozvolného náběhu zrychlení vlečené osy (milisekundy). Udává čas v milisekundách používaný k úpravě master povelových pulsů ztracených během zrychlování vlečené osy. Pokud korekci vzdálenosti nebude možné provést během nakonfigurované doby pozvolného náběhu (protože hodnota bude příliš malá), pak doba korekce je delší a bude se hlásit chyba. Toto nastavení má význam, pouze když režim pozvolného náběhu bude nastavený na Dobu pozvolného náběhu.

Pokud bude požadována rampa zrychlení bez korekce ztracených pulsů, Doba pozvolného náběhu musí být nastavená na 0. V takovém případě motor bude synchronizovat rychlost podle masteru, ale nebude se snažit opravit žádnou odchylku, která se vyskytne během zrychlování vlečené osy.

Doba pozvolného náběhu má minimální hodnotu 10, takže pro hodnoty zadané v rozsahu 1…10 místo toho použijte 10.

Výchozí: 0.

Mnohem podrobnější diskusi o této funkci najdete v kapitole 8, Pohyb vlečené osy v části Řízení rampy zrychlení vlečené osy.

5.36 Rychlost pozvolného náběhu. Toto pole je vyhrazené.

Data ladění

Záložky ladění DSM314 se používají k nakonfigurování dat ladění servoosy. Na těchto záložkách se provádí například konfigurace parametrů Typ motoru, Rychlost při maximálním povelu, Procento rychlosti přísuvu a Časová konstanta polohové smyčky. V konfiguračním okně DSM314 se může objevit jedna až čtyři konfigurační záložky ladění, jedna záložka na každou ***Servoosu*** nakonfigurovanou na kartě Nastavení.

Čísla ve sloupci "Odstavec" v následující tabulce udávají číslo referenčního odstavce v této kapitole.

Tabulka 4-10. Údaje karty ladění

Konfigurační parametr	Popis	Hodnoty	Výchozí	Jednotky	Odstavec
Typ motoru	Typ motoru	0…65535	0	---	6.01
Povel analogového serva	Typ povelu analogového serva	Rychlost Krouticí moment (poznámka 1)	Rychlost	---	6.02
Mez polohové odchylky	Mez polohové odchylky	100...60,000	60 000	Uživatelské jednotky	6.03
Zóna dosažení polohy	Zóna dosažení polohy	1…60,000	10	Uživatelské jednotky	6.04
Časová konstanta polohové smyčky	Časová konstanta polohové smyčky	0…65535	1000	0.1 ms	6.05
Rychlost při MaxCmd	Rychlost při maximálním povelu	256.. MaxVelUu (poznámka 2)	100 000	Uživatelské jednotky	6.06
Procento rychlosti posuvu	Procento rychlosti posuvu	0…12000	0	.01%	6.07
Procento zrychlení posuvu	Procento zrychlení posuvu	0…12000	0	.01%	6.08
Režim integrátoru	Režim polohové smyčky integrátoru	Vypnuto Souvisle Servo Null	Vypnuto	---	6.09
Časová konstanta integrátoru	Časová konstanta polohové smyčky integrátoru	0…10000	0	ms	6.10
Zisk rychlostní smyčky	Zisk rychlostní smyčky	0…65535	16	---	6.11

Poznámka 1: Firmware DSM verze 3.0 nebo pozdější podporuje režim krouticího momentu.

Poznámka 2: Výpočet MaxVelUu viz tabulka 4-8.

6.01 Typ motoru. Zvolí typ střídavého servomotoru FANUC používaného s DSM314 POUZE v digitálním režimu. DSM314 má interně uloženou tabulku výchozích parametrů nastavení motoru pro každý podporovaný motor GE FANUC. Motor typu *0* zakáže řízení digitálního serva modulem DSM314 pro digitální servoosu. **Pokud není připojené žádné digitální servo a když %Q bit povely nebo %AQ data povely se pošlou na osu, typ motoru musí být nastavený 0.** Podporované typy motorů FANUC jsou uvedené v následujících tabulkách.

Pro ANALOGOVÝ režim nebo pokud k ose nebude připojený žádný motor, typ motoru musí být 0. Výchozí: 0.

Čísla dílů motorů FANUC se používají ke zjištění správného kódu typu motoru FANUC a jsou ve tvaru A06B-xxxx-yyyy, kde xxxx představuje pole specifikace motoru. Například: Když údaj čísla motoru na jeho štítku bude A06B-0032-B078, specifikační číslice motoru 0032 udávají model motoru β2/3000. Tabulka β Series odkazuje na *kód typu motoru* (36) potřebný pro konfigurační pole. Podporované typy motorů FANUC jsou uvedené v následujících tabulkách. Seznam podporovaných motorů se v pozdějších verzích může rozšířit.

Servomotor FANUC α Series

Kód typu motoru	Model motoru	Specifikace motoru
61	α 1/3000	0371
46	α 2/2000	0372
62	α 2/3000	0373
15	α 3/3000	0123
16	α 6/2000	0127
17	α 6/3000	0128
18	α 12/2000	0142
19	α 12/3000	0143
27	α 22/1500	0146
20	α 22/2000	0147
21	α 22/3000	0148
28	α 30/1200	0151
22	α 30/2000	0152
23	α 30/3000	0153
30	α 40/2000	0157
29	α 40/FAN	0158

Servomotor FANUC αL Series

Kód typu motoru	Model motoru	Specifikace motoru
56	α L3/3000	0561
57	α L6/3000	0562
58	α L9/3000	0564
59	α L25/3000	0571
60	α L50/2000	0572

Servomotor FANUC αC Series

Kód typu motoru	Model motoru	Specifikace motoru
7	α C3/2000	0121
8	α C6/2000	0126
9	α C12/2000	0141
10	α C22/1500	0145

Servomotor FANUC αHV Series

Kód typu motoru	Model motoru	Specifikace motoru
3	α 12HV/3000	0176
4	α 22HV/3000	0177
5	α 30HV/3000	0178

Servomotor FANUC αM Series

Kód typu motoru	Model motoru	Specifikace motoru
24	α M3/3000	0161
25	α M6/3000	0162
26	α M9/3000	0163

Servomotor FANUC β Series

Kód typu motoru	Model motoru	Specifikace motoru
13	β 0.5/3000	0013
35	β 1/3000	0031
36	β 2/3000	0032
33	β 3/3000	0033
34	β 6/2000	0034

Servomotor FANUC βM Series

Kód typu motoru	Model motoru	Specifikace motoru
115	βM 0.5/5000	0115
116	βM 1/5000	0116

6.02 Povel analogového serva. Povel analogového serva určuje, jestli analogový povel vydaný modulem řady DSM300 je rychlostní povel nebo povel kroutícího momentu. Volbu povelu kroutícího momentu podporuje firmware DSM314 verze 3.0 DSM. Výchozí: Rychlost

6.03 Mez polohové odchylky. Mez polohové odchylky (Uživatelské jednotky). Mez polohové odchylky je maximální přípustná *Polohová odchylka (Zadaná poloha - Okamžitá poloha*), když DSM314 řídí servo. Mez polohové odchylky musí být normálně nastavená na hodnotu o 10% až 20% vyšší než je největší *Polohová odchylka* zjištěná během normální činnosti serva. Výchozí: 60000.

Vztah pro rozsah meze polohové odchylky je:

256 × (uživatelské jednotky/jednotky) ≤ Mez polohové odchylky ≤ 60,000 x (uživatelské jednotky/jednotky)

Pokud se rychlost posuvu nepoužívá, Mez polohové odchylky je možno nastavit na hodnotu přibližně o 20% vyšší než *Polohová odchylka* požadovaná k vytvoření povelu 4000 ot./min. *Polohová odchylka* (Uživatelské jednotky) požadovaná k vytvoření povelu 4000 ot./min s 0% rychlostí posuvu je:

$$\text{Polohová odchylka (uživatelské jednotky)} = \frac{\text{Časová konstanta polohové smyčky (ms) x rychlost serva při 4000 ot./min (uživatelské jednotky/sekundu)}}{1000}$$

Příklad

Poměr uživatelských jednotek : jednotkám je 2:1 a časová konstanta polohové smyčky je 50 ms.

Krok 1:

$$\text{Vypočtěte rychlost serva při 4000 ot./min} = \frac{\text{(2 uživatelské jednotky/puls) x (8192 pulsů/ot.) x (4000 otáček/minutu)}}{\text{(60 sekund/minutu)}}$$

$$= 1\,092\,266 \text{ uživatelských jednotek/sekundu}$$

Krok 2:

$$\text{Vypočtěte } \textit{Polohovou odchylku} \text{ při 4000 ot./min} = \frac{\text{(50 milisekund) x (1,092,266 uživatelských jednotek/sekundu)}}{\text{1000 milisekund/sekundu}}$$

$$= 54613 \text{ uživatelských jednotek}$$

Pokud se rychlost posuvu použije ke snížení odchylky vlečení, je možno použít menší hodnotu meze odchylky, ale obecně hodnota meze odchylky musí být o 10% - 20% vyšší než největší očekávaná odchylka vlečení.

Poznámka: Pokud hodnota *meze polohové odchylky* bude větší o více než 1000 pulsů, objeví se chyba *Porušená synchronizace* a způsobí rychlé zastavení. DSM314 se snaží zabránit chybě *Porušení synchronizace* přechodným zastavením interního generátoru povelů vždy, když polohová odchylka přesáhne *Mez polohové odchylky*. Zastavení generátoru povelů umožňuje, aby se polohová zpětná vazba chytila a polohová odchylka se snížila pod hodnotu meze odchylky.

Pokud se zpětná vazba nechytí a polohová odchylka bude narůstat, objeví se stav *porušení synchronizace*. Možné příčiny jsou:

1. Chybné zapojení zpětné vazby
2. Prokluzování spojky zpětnovazebního zařízení
3. Porucha servopohonů.
4. Mechanické přinucení hřídele motoru/snímače polohy za kapacitu kroutícího momentu serva.
5. Zadané zrychlení motoru nebo zpomalení motoru je větší než je kapacita systému.

6.04 Zóna dosažení polohy. Zóna dosažení polohy (Uživatelské jednotky). Když *Polohová odchylka* bude menší nebo se bude rovnat aktivní hodnotě ***Zóna dosažení polohy***, %I bit *V poloze* bude v jedničce. Výchozí: 10.

6.05 Časová konstanta polohové smyčky (0,1 ms). Časová konstanta polohové smyčky (jednotky = 0,1 milisekundy). Požadovaná časová konstanta polohy polohové smyčky serva. Tato hodnota nastaví délku času potřebnou k tomu, aby výstup rychlosti serva dosáhnul 63% své konečné hodnoty, když se v povelu *Rychlost* vyskytne kroková změna. Čím nižší je hodnota, tím rychlejší je odezva systému. Hodnoty, které jsou příliš nízké, způsobí nestabilitu systému a oscilace. Výchozí: 1000 = 100 ms.

Poznámka: Pro přesné sledování zadaného profilu rychlosti *Časová konstanta polohové smyčky* musí být 1/4 až 1/2 MINIMÁLNÍ doby zrychlení nebo zpomalení systému. Pokud například nejrychlejší zrychlení, které musí nastat, bude trvat 100 ms doby, *Časová konstanta polohové smyčky* musí být mezi 25 až 50 ms. Aby se udržela stabilita systému, použijte co největší hodnotu.

Pro uživatele používající šířku pásma serva vyjádřenou v rad/s:

Šířka pásma (rad/s) = 1000 / Časová konstanta polohové smyčky (ms)

Pro uživatele používající zisk serva vyjádřený v ipm/mil:

Zisk (ipm/mil) = 60 / Časová konstanta polohové smyčky (ms)

Tabulka 4-11. Zisk / Šířka pásma Časová konstanta polohové smyčky

Zisk (ipm/mil)	Šířka pásma (rad/s)	Časová konstanta polohové smyčky (ms)
0.5	8.5	120
0.75	12.5	80
1.0	16.6	60
1.5	25.1	40
2.0	33.4	30
2.5	41.8	24
3.0	50	20

U aplikací, které nevyžadují zpětnovazební řízení nebo používají velmi hrubé polohovací systémy, existuje ***režim otevřené smyčky***. Nastavením časové konstanty polohové smyčky na nulu, což indikuje zákaz polohové smyčky, se zvolí tento režim. Všimněte si, že v režimu otevřené smyčky jediný způsob, jak vygenerovat pohyb, je naprogramovat nenulovou rychlost posuvu. *Polohová odchylka* se již nepoužívá pro vygenerování pohybu, protože *Polohová odchylka* vychází z polohové zpětné vazby a režim otevřené zpětnou vazbu smyčky ignoruje.

Upozornění

U analogových os *Časová konstanta polohové smyčky* nebude přesná, dokud hodnota *rychlosti při maximálním povelu* nebude nastavena správně.

6.06 Rychlost při MaxCmd. (Uživatelské jednotky/sekundu). Na této hodnotě závisí správná funkce všech analogových funkcí serva DSM314.

V režimu digitálního serva se konfigurační pole *Rychlost při maximálním povelu* nepoužívá.

V režimu analogového serva v rychlostním režimu konfigurační pole ***Režim při maximálním povelu*** se Okamžitá rychlost serva (Uživatelské jednotky/sekundu) požaduje pro výstup 10 V

analogového rychlostního povelu DSM314. Pokud bude nutné, %AQ okamžitý povel *Vynucený D/A výstup* a %AI stavové slovo *Okamžitá rychlost* se mohou použít pro napětí povelu k empirickému určení správné konfigurační hodnoty.

V režimu analogového serva v režimu kroutícího momentu konfigurační pole ***Rychlost při maximálním povelu*** je maximální rychlost, kterou uživatel chce, aby servo běželo. Hodnota je určena schopností řízeného servosystému a schopnostmi ovládaného břemena.

Pouze v případě digitálního režimu, pokud uživatel pošle rychlostní povel DSM314, který přesáhne schopnosti servosystému, DSM314 *omezí* tuto hodnotu povelu na příslušnou maximální mezní rychlost motoru. **Všimněte si, že se do DSM314 nebude hlásit žádná chyba**.

Více informací o stanovení správné hodnoty najdete v Dodatku D, "Spouštění a ladění digitálního a analogového servosystému GE Fanuc".

Výchozí: 100000.

Upozornění

Rychlost při 10 V **musí být nakonfigurována správně, aby koeficienty analogového serva *Časová konstanta polohové smyčky* a *Rychlost posuvu* byly přesné.**

6.07 Rychlost posuvu (0.01%). Zisk rychlosti posuvu (jednotky = 0,01 procenta). Procento *Zadané rychlosti*, které se připočítá k výstupu povelu rychlosti polohové smyčky DSM314. Zvyšující se rychlost posuvu způsobí, že servo bude pracovat s rychlejší odezvou a menší polohovou odchylkou. Optimální hodnotu pro každou osu je nutno určit individuálně. U digitálních serv je hodnota 95% ***rychlosti posuvu*** dobrým výchozím bodem. U analogových serv je dobrým výchozím bodem hodnota 70%. Schopnosti servosystému určí optimální hodnotu. Pokud se ***Rychlost posuvu*** změní, ***Pos Err Limit*** může být nutné upravit. Výchozí: 0.

Upozornění

U analogových os *Procento rychlosti posuvu* nebude přesné, dokud nebude nejdříve správně nastavená hodnota Rychlost při MaxCmd.

6.08 Procento zrychlení posuvu. Tento konfigurační údaj se v současném firmwaru DSM314 nepoužívá.

6.09 Režim integrátoru. Režim integrátoru. Režim činnosti integrátoru polohové odchylky polohové smyčky. Vypnuto znamená, že se integrátor nepoužívá. Souvisle znamená, že integrátor běží souvisle i během pohybu serva. Servo Null znamená, že integrátor poběží, pouze když %I stavový bit *Pohyb* bude v nule. ***Režim integrátoru*** musí být normálně nastavený na nulu. Souvislý režim se může použít pro činnost vlečené osy, pouze když se očekává konstantní nebo pomalu se měnící rychlost masteru. Tento parametr se nesmí použít k tlumení vzruchů ve zpětné vazbě polohové smyčky. Nikdy nevolte Souvisle pro aplikace polohování z bodu do bodu. Výchozí: VYPNUTO.

6.10 Časová konstanta integrátoru. Časová konstanta integrátoru (milisekundy). Je to časová konstanta integrátoru polohové odchylky v polohové smyčce. Tato hodnota udává čas potřebný ke snížení polohové odchylky o 63%. Pokud například Časová konstanta integrátoru bude 1000 (1 sekunda), *Polohová odchylka* by se po 1 sekundě snížila na 37% své výchozí hodnoty. Nulová hodnota integrátor vypne. **Pokud se používá, Časová konstanta integrátoru musí být 5 až 10 krát větší než *Časová konstanta polohové smyčky*, aby nedocházelo k nestabilitě a oscilacím.** Výchozí: 0.

6.11 Zisk rychlostní smyčky. Používá se k nastavení zisku rychlostní smyčky. To platí pouze pro digitální a analogová serva GE Fanuc v režimu kroutícího momentu. **Tento parametr se nepoužívá pro analogová serva v rychlostním režimu.** Vztah

$$\text{Zisk rychlostní smyčky} = \frac{\text{Setrvačnost břemena } (J_L)}{\text{Setrvačnost motoru } (J_M)} \times 16$$

je možno použít ke zvolení výchozí hodnoty zisku rychlostní smyčky. Přípustný rozsah hodnot je 0 až 255. Hodnotu 0 je nutno použít, pokud hřídel motoru nebude spojena s břemenem.

Výchozí: 16 (setrvačnost břemena se rovná setrvačnosti motoru).

Výpočet proměnných pro meze dat

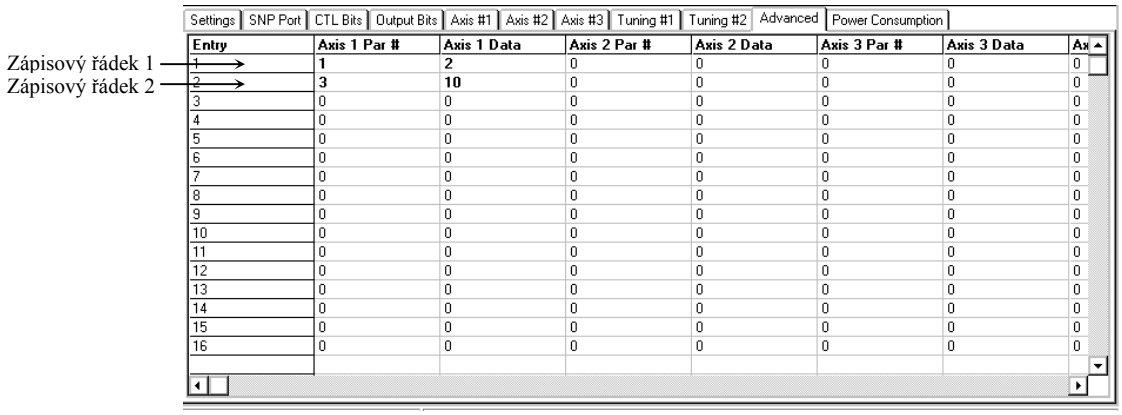
Hodnoty pro meze dat pro parametry MaxPosnUu, MaxVelUu a MaxAccUu, které se uvádějí v některých tabulkách v této kapitole, je možno vypočítat pomocí následujících vztahů:

Tabulka 4-12. Výpočet proměnných pro meze dat

Vztahy pro výpočet proměnných pro meze dat		
Mez polohy MaxPosnUu	**Mez rychlosti MaxVelUu**	**Mez zrychlení MaxAccUu**
IF uu:cts >= 1:1 MaxPosnUu = 536,870,912 ELSE (uu:cts < 1:1) MaxPosnUu = 536,870,912 * uu/cts	MaxVelUu = 1,000,000* uu/cts	IF uu:cts >= 1:1 MaxAccUu = 1,073,741,823 ELSE (uu:cts < 1:1) MaxAccUu = 1,073,741,823* uu/cts

Záložka rozšířených dat

I když rozšířená záložka má 16 řádků pro zápis dat parametrů ladění osy, firmware DSM314 verze 1.0 umožňuje používat pouze zápis do řádků 1 a 2. Následující obrázek ukazuje data v buňkách pro osu 1 v zápisových řádcích 1 a 2. Firmware DSM verze 3.0 nebo pozdější již toto omezení nemá.



Entry	Axis 1 Par #	Axis 1 Data	Axis 2 Par #	Axis 2 Data	Axis 3 Par #	Axis 3 Data	Ax
1	1	2	0	0	0	0	0
2	3	10	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

Obrázek 4-2. Rozšířená záložka

Parametry ladění podporované verzí 1.0

DSM314 verze 1.0 podporuje pouze parametry ladění 1 a 3:

Parametr ladění 1: Nastaví rozlišení snímače polohy (pouze pro digitální serva GE Fanuc). Jiné nastavení než 0 bude mít za následek snížení maximální podporované rychlosti motoru. Všimněte si, že u nastavení 0 a 1 jsou některé hodnoty rychlostí pod maximální podporovanou rychlostí uvedenou v tabulce. Rozsah přípustných nastavení: 0 – 3. Na obrázku 4.2 výše je parametr ladění 1 nastavený na hodnotu 2 pro osu 1.

Tabulka 4-13. Hodnoty parametru ladění 1

Hodnoty parametru ladění 1	Pulsy/otáčku	Maximální podporovaná rychlost motoru
0	8192	4400[1,2]
1	16384	3662[2]
2	32768	1831
3	65536	915

Poznámka 1: Výchozí nastavení

Poznámka 2: Maximální rychlost některých motorů je nižší než hodnota uvedená v tabulce.

Parametr ladění 3: Nastaví minimální výstup rychlosti (v mV) pro analogová serva. Přípustný rozsah je 0 -1000 mV. Doporučené nastavení je 5 - 10 mV nebo stačí, když servo použije +/- 1 puls na polohovou odchylku. Na obrázku 4.2 výše je parametr ladění 3 nastavený na hodnotu 10 pro osu 1.

Parametry ladění podporované verzí 3.0

DSM314 verze 3.0 podporuje další parametry ladění pro podporu režimu krouticího momentu.

Parametr ladění 6: Nastavuje rozlišení snímače polohy. Parametr se používá pouze v režimu krouticího momentu. Pro správnou činnost v režimu kroutícího momentu tato hodnota musí být nastavena na počet pulsů snímače polohy s fázovým posuvem (4× počet pulsů snímače polohy) generovaných zpětnovazebním zařízením motoru na otáčku. Uživatel může určit hodnotu ze specifikace zpětnovazebního zařízení. Jako dvojnásobnou kontrolu je možno připojit zpětnovazební zařízení k DSM a ručně otočit hřídel motoru o jednu otáčku. Údaj %AI dat DSM pro okamžitou polohu musí přesně souhlasit (odchylky jsou způsobené nepřesným otočením hřídele o jednu otáčku) s hodnotou zapsanou v tomto parametru. Přípustný rozsah je 100 - 32 767 pulsů/otáčku. Výchozí hodnota 4096 pulsů na otáčku.

Parametr ladění 7: Nastaví proporcionální zisk regulátoru rychlosti. Parametr se používá pouze v režimu krouticího momentu. Vynásobením proporcionálního zisku rychlostní odchylkou (povel rychlosti - rychlostní zpětná vazba) se vygeneruje část povelu kroutícího momentu odpovídajícího proporcionální části. Správné nastavení této hodnoty určí, jak dobře bude regulátor rychlosti v řídicím systému fungovat. Dodatek D popisuje metodu správného naladění tohoto parametru. Přípustný rozsah proporcionálního zisku rychlostní smyčky je 0 - 32 767. Výchozí hodnota je 1500.

Parametr ladění 8: Nastaví integrální zisk regulátoru rychlosti. Parametr se používá pouze v režimu krouticího momentu. Integrální zisk je hodnota, která po vynásobení plochou rychlostní odchylky (povel rychlosti - rychlostní zpětná vazba) vygeneruje část povelu kroutícího momentu odpovídajícího integrální hodnotě. Správné nastavení této hodnoty určí, jak dobře bude regulátor rychlosti v řídicím systému fungovat. Dodatek D popisuje metodu správného naladění tohoto parametru. Přípustný rozsah proporcionálního zisku rychlostní smyčky je 0 - 32 767. Výchozí hodnota je 0.

Parametr ladění 10: Nastaví filtr povelu kroutícího momentu. Filtr povelu kroutícího momentu umožňuje uživateli aktivovat filtr dolní propusti pro výstup regulátoru rychlosti. Filtr se typicky používá k tomu, aby kontrolér nevybudil rezonanci stroje. Přípustný rozsah nastavení filtru krouticího momentu je 0 - 3. Výchozí hodnota je 0.

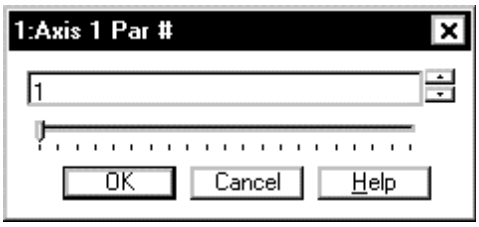
Tabulka 4-14. Hodnoty parametru ladění 10

Hodnoty parametru ladění 10	Nastavení filtru dolní propusti povelu kroutícího momentu
0	VYPNUTO[1]
1	Nízkopásmový filtr (bod 150 Hz 3dB)
2	Středopásmový filtr (bod 250 Hz 3dB)
3	Vysokopásmový filtr (bod 350 Hz 3 dB)

Poznámka 1: Výchozí nastavení

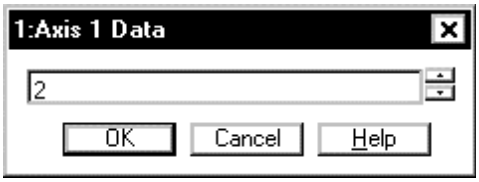
Zápis čísel a dat parametrů ladění do polí rozšířené karty

Chcete-li začít, klikněte na požadované pole. Pokud například dvakrát kliknete na pole na zápisovém řádku 1, ve sloupci Axis 1 Par # se objeví dále uvedené dialogové okno 1:Axis 1 Par #. Zapište požadované číslo parametru ladění. Nezapomeňte, že firmware DSM314 verze 1.0 podporuje pouze parametry ladění 1 a 3. V následujícím příkladu byla zapsaná hodnota 1 představující parametr ladění 1:



Kliknutím na tlačítko OK se na zápisovém řádku 1 do pole Axis 1 Par # umístí 1. (Ukázáno na obrázku 4-2.)

Potom dvakrát klikněte na pole napravo, zápisový řádek 1, data osy 1. Zobrazí se následující dialogové okno. V tomto příkladu bylo zapsáno číslo 2. Nezapomeňte, že pro parametr ladění 1 jsou platné pouze hodnoty 0 až 3 (viz tabulka 4-13).



Kliknutím na tlačítko OK se na zápisovém řádku 1 do pole data osy 1 umístí hodnota 2. (Ukázáno na obrázku 4-2.)

Údaje o spotřebě

Toto je pouze zobrazovaná záložka, která udává výkon požadovaný modulem DSM314.

Kapitola 5

Rozhraní mezi Motion Mate DSM314 a PLC

Tato kapitola definuje data, která se automaticky přenášejí mezi CPU a modulem Motion Mate DSM314 při každém cyklu PLC bez naprogramování uživatelem. Tato data mají následující členění:

- Stavová data vstupu (přenášená z modulu Motion Mate DSM314 do CPU)
  - Stavové bity: 32 (1 osa), 48 (2 osy), 64 (3 osy), 80 (4 osy) bitů dat %I
  - Stavová slova: 24 (1 osa),44 (2 osy), 64 (3 osy), 84 (4 osy) slov dat %AI
- Data výstupního povelu (přenášená z CPU do modulu Motion Mate DSM314)
  - Diskrétní povely: 32 (1 osa), 48 (2 osy), 64 (3 osy), 80 (4 osy) bitů dat %Q
  - Okamžité povely: 3 (1 osa) ,6 (2 osy), 9 (3 osy), 12 (4 osy) slov dat %AQ

Poznámka: Slova uvedená v této kapitole *kurzívou* se týkají skutečných adres dat v PLC (%I, %A, %AI, %AQ).

Část 1: Stavové bity %I

Z DSM314 do CPU se při každém cyklu přenášení následující stavové bity %I. Aktuální adresy stavových bitů závisí na počáteční adrese pro adresy %I (viz tabulka 4-1, "Záložka nastavení"). Offsety bitů uvedené v následující tabulce jsou offsety od počáteční adresy. Veškerá označení adres se týkají této kapitoly.

Tabulka 5-1. Stavové bity %I

Offset bitu	Popis	Osa	Odstavec	Offset bitu	Popis	Osa	Odstavec
00	Výskyt chyby modulu	---	1.01	40	Mez polohové odchylky	Servo 2	1.12
01	Aktivní lokální logika	---	1.02	41	Mez krouticího momentu	Servo 2	1.13
02	Přijatá nová konfigurace	---	1.03	42	Servo připraveno / IN4_B (5v)	Servo 2	1.14
03	*Vyhrazeno*			43	*Vyhrazeno*		
04	CTL01 (funkce zvolená konfigurací)	---	1.04	44	Vlečená osa povolena	Servo 2	1.15
05	CTL02 (funkce zvolená konfigurací)	---	1.04	45	Mez rychlosti	Servo 2	1.16
06	CTL03 (funkce zvolená konfigurací)	---	1.04	46	Rampa vlečené osy aktivní	Servo 2	1.17
07	CTL04 (funkce zvolená konfigurací)	---	1.04	47	*Vyhrazeno*		
08	CTL05 (funkce zvolená konfigurací)	---	1.04	48	Osa OK	Servo 3	1.05
09	CTL06 (funkce zvolená konfigurací)	---	1.04	49	Poloha platná	Servo 3	1.06
10	CTL07 (funkce zvolená konfigurací)	---	1.04	50	Pohon povolen	Servo 3	1.07
11	CTL08 (funkce zvolená konfigurací)	---	1.04	51	Program aktivní	Servo 3	1.08
12	CTL13 (funkce zvolená konfigurací)	---	1.04	52	Pohyb	Servo 3	1.09
13	CTL14 (funkce zvolená konfigurací)	---	1.04	53	V poloze	Servo 3	1.10
14	CTL15 (funkce zvolená konfigurací)	---	1.04	54	Vzorkování příznaku 1 (5v)	Servo 3	1.11
15	CTL16 (funkce zvolená konfigurací)	---	1.04	55	Vzorkování příznaku 2 (5v)	Servo 3	1.11
16	Osa OK	Servo 1	1.05	56	Mez polohové odchylky	Servo 3	1.12
17	Poloha platná	Servo 1	1.06	57	*Vyhrazeno*		
18	Pohon povolen	Servo 1	1.07	58	Vstup servo připraveno /IN4_C	Servo 3	1.14
19	Program aktivní	Servo 1	1.08	59	*Vyhrazeno*		
20	Pohyb	Servo 1	1.09	60	Vlečená osa povolena	Servo 3	1.15
21	V poloze	Servo 1	1.10	61	Mez rychlosti	Servo 3	1.16
22	Vzorkování příznaku 1 (5v)	Servo 1	1.11	62	Rampa vlečené osy aktivní	Servo 3	1.17
23	Vzorkování příznaku 2 (5v)	Servo 1	1.11	63	*Vyhrazeno*	Servo 3	
24	Mez polohové odchylky	Servo 1	1.12	64	Osa OK	Servo 4	1.05
25	Mez krouticího momentu	Servo 1	1.13	65	Poloha platná	Servo 4	1.06
26	Servo připraveno / IN4_A (5v)	Servo 1	1.14	66	Pohon povolen	Servo 4	1.07
27	*Vyhrazeno*			67	Program aktivní	Servo 4	1.08
28	Vlečená osa povolena	Servo 1	1.15	68	Pohyb	Servo 4	1.09
29	Mez rychlosti	Servo 1	1.16	69	V poloze	Servo 4	1.10
30	Rampa vlečené osy aktivní	Servo 1	1.17	70	Vzorkování příznaku 1 (5v)	Servo 4	1.11
31	*Vyhrazeno*			71	Vzorkování příznaku 2 (5v)	Servo 4	1.11

Offset bitu	Popis	Osa	Odstavec	Offset bitu	Popis	Osa	Odstavec
32	Osa OK	Servo 2	1.05	72	Mez polohové odchylky	Servo 4	1.12
33	Poloha platná	Servo 2	1.06	73	*Vyhrazeno*	Servo 4	
34	Pohon povolen	Servo 2	1.07	74	Servo připraveno / IN4_D (5v)	Servo 4	1.14
35	Program aktivní	Servo 2	1.08	75	*Vyhrazeno*	Servo 4	
36	Pohyb	Servo 2	1.09	76	Vlečená osa povolena	Servo 4	1.15
37	V poloze	Servo 2	1.10	77	Mez rychlosti	Servo 4	1.16
38	Vzorkování příznaku 1 (5v)	Servo 2	1.11	78	Rampa vlečené osy aktivní	Servo 4	1.17
39	Vzorkování příznaku 2 (5v)	Servo 2	1.11	79	*Vyhrazeno*	Servo 4	

1.01 Výskyt chyby modulu. Tento stavový bit se nastaví, když DSM314 zjistí nějakou chybu. Chyby vztahující se ke konkrétnímu servu nebo pomocné ose se označí v souvisejícím slově %AI *Chybový kód osy n*. Chyby modulu nevztahující se ke konkrétní ose se označí v %AI slově *Stavový kód modulu*. Více informací najdete v části 2 "Stavová slova %AI". Bit %Q *Vynulovat chybu* je jediný povel, který vynuluje stavový bit %I *Výskyt chyby modulu* a související slovo (slova) %AI *Stavový kód modulu* a *Chybový kód osy n*. Pokud stav, který vyvolal chybu, bude trvat i nadále, stavový bit %I *Výskyt chyby modulu* se nevynuluje.

1.02 Aktivní lokální logika. Když tento stavový bit bude v jedničce, indikuje, že se vykonává program lokální logiky.

1.03 Přijatá nová konfigurace. Stavový bit %I *Přijatá nová konfigurace* se nastaví do jedničky vždy, když PLC pošle do DSM314 povel reset nebo novou konfiguraci. Program PLC musí bit *Přijatá nová konfigurace* vynulovat před tím, než se do DSM314 pošlou nějaké okamžité povely %AQ, například *Zóna dosažení polohy* nebo *Časová konstanta polohové smyčky*. Stavový bit je možno monitorovat v PLC. Pokud bit bude nastavený do jedničky, pak se provedl reset nebo nová konfigurace DSM314. PLC musí bit vynulovat a pak znovu poslat všechny potřebné povely %AQ. Bit se vynuluje okamžitým povelem %AQ 49h. Více podrobností o rozhraních okamžitého povelu %AQ najdete v části 4, “Okamžité povely %AQ” dále v této kapitole.

1.04 Konfigurovatelné stavové bity %I. Tyto vstupy indikují stav konfigurovatelných bitů CTL CTL01-CTL08 a CTL13-CTL16. Výchozí nastavení bitů CTL udávají úroveň externích vstupních zařízení připojených k signálům čelní desky. Všechny bity CTL je možno testovat během vykonávání povelů pohybového programu *Čekat* a *Podmíněný skok*. Bity CTL se také mohou použít k aktivaci funkcí povolení/zakázání rampy vlečené osy. Nastavení bitů CTL se provádí nakonfigurováním. Další informace najdete v kapitole 4 a 14. Výchozí nastavení CTL01-CTL08 a CTL13-CTL16 je uvedeno v tabulce 5-2.

Tabulka 5-2. Výchozí hodnoty konfigurovatelných stavových bitů %I

Název bitu	Název signálu	Použití signálu	Typ vstupu	Pin na konektoru čelní desky	Pin svorkovnice digitálního serva	Pin svorkovnice analogového serva/pomocné osy
CTL01	*IN9_A*	Přejezd servoosy 1 (+)	24 V	A-16	6	16
CTL02	*IN10_A*	Přejezd servoosy 1 (-)	24 V	A-34	14	34
CTL03	*IN11_A*	Spínač výchozí polohy servoosy 1	24 V	A-17	7	17
CTL04	*IN1_A*	Úroveň vzorkování 1 servoosy 1	5 V	A-1,19	1,9	9
CTL05	*IN9_B*	Přejezd servoosy 2 (+)	24 V	B-16	6	16
CTL06	*IN10_B*	Přejezd servoosy 2 (-)	24 V	B-34	14	34
CTL07	*IN11_B*	Spínač výchozí polohy servoosy 2	24 V	B-17	7	17
CTL08	*IN1_B*	Úroveň vzorkování 2 servoosy 1	5 V	B-1,19	1,9	9
CTL13	*IN9_C*	Přejezd servoosy 3 (+)	24 V	C-16	NA	16
CTL14	*IN10_C*	Přejezd servoosy 3 (-)	24 V	C-34	NA	34
CTL15	*IN11_C*	Spínač výchozí polohy servoosy 3	24 V	C-17	NA	17
CTL16	*IN5_C*	Úroveň vzorkování 1 servoosy 3	5 V	C-9	NA	9

1.05 Osa OK. Stavový bit *Osa OK* bude v jedničce, když DSM314 bude připravený pro příjem povelů a k řízení serva. Chybový stav, který zastaví servo, nastaví bit *Osa OK* do nuly. **Když bit *Osa OK* bude v nule, osa nebude přijímat žádné jiné povely než bit %Q *Vynulovat chybu*.**

1.06 Poloha platná. U servoosy stavový bit *Poloha platná* indikuje, že povel *Nastavit polohu* nebo úspěšně dokončený cyklus *Nalezení výchozí polohy* nastavil do stavového slova %AI hodnotu polohy *Okamžitá poloha*. U servoosy *Poloha platná* musí být v jedničce, aby se vykonal pohybový program.

U pomocné osy stavový bit *Poloha platná* indikuje, že povel *Nastavení polohy pomocného snímače polohy* nebo úspěšně dokončený cyklus Nalezení výchozí polohy nastavil do stavového slova %AI hodnotu polohy *Okamžitá poloha*. U pomocné osy *Poloha platná* **nemusí** být v jedničce, aby se vykonal pohybový program.

Pokud DSM314 bude nakonfigurováno pro použití snímače polohy s absolutní zpětnou vazbou (servo GE Fanuc α nebo β Series s přídavnou baterií snímače polohy), *Poloha platná* se automaticky nastaví vždy, když digitální snímač polohy bude hlásit platnou absolutní polohu. Podrobnosti funkce, když se používá absolutní režim digitálních snímačů polohy, viz Dodatek C.

1.07 Pohon povolen. Stavový bit *Pohon povolen* indikuje stav bitu %Q *Povolit pohon* a výstupu polovodičového relé, který přichází z DSM314. Jednička v bitu %I *Pohon povolen* odpovídá stavu výstupu relé SEPNUTO a stavu SVÍTÍ diody EN LED na čelní desce. V digitálním režimu polovodičové relé dává signál MCON do digitálního serva GE Fanuc přes povelový kabel serva. *Pohon povolen* se vynuluje po zapnutí napájení nebo chybovým stavem, který zastaví servo.

1.08 Program aktivní. Stavový bit *Program aktivní* pro každou osu indikuje, že se v této ose vykonává Pohybový program (1-10) nebo povel %AQ *Pohyb* (27h). Vykonání programu pro několik os nastaví do jedničky bit %I *Program aktivní* pro osu 1 a osu 2.

1.09 Pohyb. Stavový bit *Pohyb* bude v jedničce, když *Zadaná rychlost* bude nenulová, jinak bude v nule. Všechny povely *Pohyb*, *Jog* a *Pohyb rychlostí* způsobí, že bit *Pohyb* se nastaví do jedničky. Povel %AQ *Vynucení rychlosti serva* a rampa zrychlení vlečené osy bit *Pohyb* nenastaví.

V režimu vlečené osy je bit *Pohyb* v jedničce pro výše uvedené stavy a stav povolení nebo zákazu master vstupu vlečené osy na něj nemá vliv. Když rampa zrychlení/zpomalení vlečené osy bude aktivní, samostatný bit %I *Rampa vlečené osy aktivní* bude v jedničce. Další informace o rampě zrychlení vlečené osy najdete v kapitole 8, pohyb vlečené osy.

1.10 V poloze. Činnost bitu *V poloze* závisí pouze na hodnotě *Polohová odchylka* a nevztahuje se ke stavu bitu *Pohyb*. Bit *V poloze* bude v jedničce vždy, když *Polohová odchylka* bude menší nebo se bude rovnat nakonfigurované hodnotě ***Zóna dosažení polohy***. Bit *V poloze* (v jedničce) je možno použít v kombinaci s bitem *Pohyb* (v nule) ke zjištění, jestli se osa dostala do své cílové polohy.

Tabulka 5-3. *V poloze* Činnost bitu

Generátor povelů aktivní (bit %I *Pohyb* v jedničce)	***Polohová odchylka* ≤ Zóna dosažení polohy**	Bit ***V poloze***	**Osa v cíli**
Ne	Ne	v nule	Ne
Ne	Ano	v jedničce	Ano
Ano	Ne	v nule	Ne
Ano	Ano	v jedničce	Ne

1.11 Příznak vzorkování 1, Příznak vzorkování 2. Stavové bity *Příznak vzorkování 1* a *Příznak vzorkování 2* indikují, že na odpovídajícím vstupu vzorkování čelní desky došlo k přechodu z nuly do jedničky. Když k tomuto dojde, zachytí se poloha osy a předá se do stavového slova %AI *Vzorkování polohy n*, kde "n" je osa 1 – osa 4. Bit %I *Vzorkování příznaku n* se vynuluje odpovídajícím bitem %Q *Reset vzorkování n*. **Pro vynulování bitu %I *Vzorkování příznaku n* v PLC po nastavení bitu %Q *Reset vzorkování n* do jedničky se potřebují maximálně 2 cykly PLC.** Jakmile se bit *Vzorkování příznaku n* vynuluje, nová data je možno zachytit jiným vstupem vzorkování. Rozlišení pro načtení polohy je +/- 2 pulsy s dalšími 10 mikrosekundami rozptylu na prodlevu vstupního filtru vzorkovacích impulsů.

Poznámka: Bity *Vzorkování příznaku n* neindikují logickou úroveň vstupu na čelní desce, pouze indikují, že na vstupu došlo k přechodu z nuly do jedničky.

1.12 Mez polohové odchylky. Stavový bit *Mez polohové odchylky* se nastaví do jedničky, když absolutní hodnota polohové odchylky přesáhne nakonfigurovanou hodnotu *Meze polohové odchylky*. Když stavový bit *Mez polohové odchylky* bude v jedničce, *Zadaná rychlost* a *Zadaná poloha* se zmrazí, aby se osa mohla "chytit" *Zadané polohy*.

1.13 Mez krouticího momentu. Stavový bit *Mez krouticího momentu* se nastaví do jedničky, když zadaný krouticí moment přesáhne mez krouticího momentu pro nakonfigurovaný typ motoru.

1.14 Servo připraveno. Tento stavový bit bude v jedničce, když signál čelní desky IN4 souvisejícího konektoru (A, B, C nebo D) bude ve stavu ON (aktivní nula: ON = 0 V, OFF = +5 V). Tento vstup do servozesilovače hlásí po každou servoosu stav *Servo připraveno*.

1.15 Vlečená osa povolena. Tento stavový bit indikuje, kdy bude vlečená osa pro osu povolená. Bit %Q *Povolení vlečené osy* a přídavný vstup aktivace čelní desky CTL01-CTL32 povolí funkci vlečené osy. Pokud bude aktivní řízení rampy zrychlení vlečené osy, když bit *Vlečená osa povolena* přejde do jedničky, osa se zrychlí na rychlost masteru, a když přejde do nuly, osa se zpomalí na nulový povel rychlosti masteru. Zrychlení i zpomalení během procesu rampy používají nastavené zrychlení rampy vlečené osy.

1.16 Mez rychlosti. Stavový bit *Mez rychlosti* bude v jedničce, když rychlost požadovaná některým povelem osy (interní generátor dráhy nebo interní/externí zdroj vlečené osy) přesáhne nastavené meze rychlosti. Proto *Mez rychlosti* je indikací toho, že osa už není pevně svázaná s povelem její polohy. Pokud vlečená osa bude povolená, bude se hlásit chybový kód v související proměnné chybového kódu osy, když *Mez rychlosti* bude v jedničce.

Výjimka je, když jednosměrný pohyb bude nakonfigurovaný nastavením Směru povelu na Pouze kladný nebo Pouze záporný. Pouze kladný znamená, že mez rychlosti je nula pro záporný pohyb. Pouze záporný znamená, že mez rychlosti bude nula pro kladný pohyb. Pro mez, která je nastavená na nulu, se negeneruje žádná chyba. Pokud například Směr povelu bude nastaveno na Pouze záporný a budou zadané + pulsy, stavový bit *Mez rychlosti* bude v jedničce, ale nebude se hlásit žádný kód stavové chyby.

1.17 Rampa vlečené osy aktivní. Když bude vlečená osa povolená, *Rampa vlečené osy aktivní* bude během výchozího pozvolného náběhu zrychlení a vzdálenosti v jedničce. Když bude vlečená osa zakázána, *Rampa vlečené osy aktivní* bude v jedničce, dokud se neodjede inkrementální vzdálenost *Zákaz činnosti vlečené osy* (pokud je zvolena) a vlečená osa zpomalí na nulovou rychlost.

Část 2: Stavová slova %AI

Následující stavová slova %AI se přenášejí automaticky z DSM314 do při každém cyklu CPU. Celkový počet stavových slov %AI je nastavený pomocí konfiguračního softwaru na délku 24, 44, 64 nebo 84. Aktuální adresy stavových slov závisí na počáteční adrese nakonfigurované pro adresy %AI. Viz tabulka 4-1, "Záložka Nastavení". Čísla slov uvedená v následující tabulce jsou offsety od počáteční adresy. Veškerá označení adres se týkají této kapitoly. Všechna data %AI kromě *Okamžité rychlosti* se aktualizují v DSM314 vzorkovací rychlostí polohové smyčky (2 ms digitální serva, 0,5 ms nebo 1,0 ms pro některé konfigurace analogového serva). *Okamžitá rychlost* se aktualizuje jednou za 128 milisekund.

Tabulka 5-4. Stavová slova %AI

Offset slova	Popis	Osa	Odstavec	Offset slova	Popis	Osa	Odstavec
00	Kód stavu modulu	---	2.01				
01-03	*Vyhrazeno*						
04	Chybový kód osy 1	Servo 1	2.02	44	Chybový kód osy 3	Servo 3	2.02
05	Číslo povelového bloku	Servo 1	2.03	45	Číslo povelového bloku	Servo 3	2.03
06-07	Zadaná poloha	Servo 1	2.04	46-47	Zadaná poloha	Servo 3	2.04
08-09	Aktuální poloha	Servo 1	2.05	48-49	Aktuální poloha	Servo 3	2.05
10-11	Vzorkování polohy 1	Servo 1	2.06	50-51	Vzorkování polohy 1	Servo 3	2.06
12-13	Vzorkování polohy 2	Servo 1	2.06	52-53	Vzorkování polohy 2	Servo 3	2.06
14-15	Polohová odchylka	Servo 1	2.07	54-55	Polohová odchylka	Servo 3	2.07
16-17	Zadaná rychlost	Servo 1	2.08	56-57	Zadaná rychlost	Servo 3	2.08
18-19	Aktuální rychlost	Servo 1	2.09	58-59	Aktuální rychlost	Servo 3	2.09
20-21	Uživatelem volitelná data 1	Servo 1	2.10	60-61	Uživatelem volitelná data 1	Servo 3	2.10
22-23	Uživatelem volitelná data 2	Servo 1	2.11	62-63	Uživatelem volitelná data 2	Servo 3	2.11
24	Chybový kód osy 2	Servo 2	2.02	64	Chybový kód osy 4	Servo 4	2.02
25	Číslo zadaného bloku	Servo 2	2.03	65	Číslo povelového bloku	Servo 4	2.03
26-27	Zadaná poloha	Servo 2	2.04	66-67	Zadaná poloha	Servo 4	2.04
28-29	Aktuální poloha	Servo 2	2.05	68-69	Aktuální poloha	Servo 4	2.05
30-31	Vzorkování polohy 1	Servo 2	2.06	70-71	Vzorkování polohy 1	Servo 4	2.06
32-33	Vzorkování polohy 2	Servo 2	2.06	72-73	Vzorkování polohy 2	Servo 4	2.06
34-35	Polohová odchylka	Servo 2	2.07	74-75	Polohová odchylka	Servo 4	2.07
36-37	Zadaná rychlost	Servo 2	2.08	76-77	Zadaná rychlost	Servo 4	2.08
38-39	Aktuální rychlost	Servo 2	2.09	78-79	Aktuální rychlost	Servo 4	2.09
40-41	Uživatelem volitelná data 1	Servo 2	2.10	80-81	Uživatelem volitelná data 1	Servo 4	2.10
42-43	Uživatelem volitelná data 2	Servo 2	2.11	82-83	Uživatelem volitelná data 2	Servo 4	2.11

2.01 Kód stavu modulu. *Kód stavu modulu* indikuje aktuální provozní stav DSM314. Když příznak %I *Výskyt chyby modulu* bude v jedničce a chyba se nebude týkat konkrétní osy, ve stavovém slově *Kód stavu modulu* se bude hlásit číslo chybového kódu popisující stav, který chybu způsobil. Nový *Kód stavu modulu* **nenahradí** předchozí *Kód stavu modulu*, dokud nový *Kód stavu modulu* nebude mít prioritu Rychlé zastavení nebo Systémová chyba.

Slovo *Kód stavu modulu* se také používá k hlášení *Stavových chyb systému*. Mají formát Dxxx, Exxx a Fxxx. Podrobnosti o kódech *Stavových chyb systému* najdete v Dodatku A.

Seznam chybových kódů Motion Mate DSM314 najdete v Dodatku A.

2.02 Chybový kód osy 1 - Chybový kód osy 4. *Chybový kód servoosy n*, kde n = osa 1 - osa 4, indikuje aktuální provozní stav každé osy. Když příznak %I *Výskyt chyby modulu* bude v jedničce a chyba se bude týkat konkrétní osy, bude se hlásit číslo chybového kódu popisující stav, který chybu způsobil. Nový *Chybový kód osy* nahradí předchozí *Chybový kód osy*, pokud má v porovnání s předchozím *Chybovým kódem osy* stejnou nebo vyšší prioritu (Výstraha, Normální zastavení, Rychlé zastavení).

Seznam chybových kódů Motion Mate DSM314 najdete v Dodatku A.

2.03 Číslo povelového bloku. *Číslo povelového bloku* indikuje číslo bloku povelu, který se momentálně vykonává v aktivním programu nebo podprogramu. Mění se na začátku každého nového bloku s tím, jak se vykonává program, a tak identifikuje aktuální pracovní místo v programu. Čísla bloků se zobrazí, pouze když je pohybový program používá. Kromě toho se číslo naposledy použitého bloku bude zobrazovat, dokud nebude nahrazeno novou hodnotou. *Číslo povelového bloku* se při zapnutí napájení nebo při resetu nastaví na nulu.

2.04 Zadaná poloha. *Zadaná poloha* (uživatelské jednotky) je poloha, která zadává ose, kde se má v každém okamžiku vyskytovat. U servoosy rozdíl mezi *Zadanou polohou* a *Okamžitou polohou* je hodnota *Polohová odchylka*, která vygeneruje rychlostní povel pro pohon osy. Rychlost, kterou se *Zadaná poloha* mění, určuje rychlost pohybu osy.

Pokud *Zadaná poloha* bude přesahovat některou mezní hodnotu počtu pulsů, přetočí se na druhou mez a bude pokračovat ve směru pohybu osy.

2.05 Okamžitá poloha. *Okamžitá poloha* (uživatelské jednotky) je hodnota, kterou udržuje DSM314 pro znázornění fyzické polohy osy. Nastavuje se na výchozí polohu okamžitým povelem %AQ *Nastavení polohy* nebo na *Výchozí poloha* cyklem *Nalezení výchozí polohy*. Když se používají digitální absolutní snímače polohy, *Okamžitá poloha* se automaticky nastaví vždy, když snímač polohy bude hlásit platnou polohu. Pohyb zpětnovazebního zařízení osy plynule aktualizuje *Okamžitou polohu* osy.

Pokud *Okamžitá poloha* bude přesahovat některou mezní hodnotu počtu pulsů, přetočí se na druhou mez a bude pokračovat ve směru pohybu osy.

2.06 Vzorkování polohy 1, 2. *Vzorkování polohy 1* a *Vzorkování polohy 2* (uživatelské jednotky) obsahují okamžitou polohu osy, když se vyskytne Vzorkování vstupu 1 nebo Vzorkování vstupu 2. Když se vyskytne vzorkování vstupu, bit %I *Vstup vzorkování příznaku 1* nebo *Vzorkování příznaku 2* se nastaví do jedničky jako indikace pro PLC, že jsou k dispozici nová vzorkovací data v souvisejícím stavovém slově *Vzorkování polohy 1* nebo *Vzorkování polohy 2*. PLC musí nastavit správný příznakový bit %Q *Reset vzorkování 1* nebo *Reset vzorkování 2*, aby se vynuloval související bit %I *Vzorkování příznaku 1, 2*.

Vzorkování polohy 1, 2 se bude udržovat a nepřepíše se dalšími vzorkovacími vstupy, dokud bit %I *Vzorkování příznaku 1, 2* nebude vynulovaný. Pokud bit %Q *Reset příznaku vzorkování* zůstane v jedničce (a tak přidrží bit %I *Vzorkování příznaku 1, 2* ve vynulovaném stavu), pak každý vzorkovací vstup, který se vyskytne, vyvolá zachycení polohy osy do *Vzorkování polohy 1, 2*.

Hodnoty okamžité polohy *Vzorkování polohy 1, 2* se také uloží do datových registrů parametrů pro použití s povely pohybových programů. Nastavení datového registru parametrů je následující:

	Servoosa 1	**Servoosa 2**	**Servoosa 3**	**Servoosa 4**
Vzorkování polohy 1	P224	P232	P240	P248
Vzorkování polohy 2	P225	P233	P241	P249

Tato funkce umožňuje, aby vzorkovací vstup aktivoval Podmíněný skok v bloku programu s použitím *Vzorkování polohy 1* nebo *Vzorkování polohy 2* jako cíle povelu CMOVE nebo PMOVE.

Informace o čekací době a dobách zpracování najdete v kapitole 1, "Přehled výrobků, Vzorkování polohy DSM314".

2.07 Polohová odchylka. *Polohová odchylka* (uživatelské jednotky) je rozdíl mezi *Zadanou polohou* a *Okamžitou polohou*. U řídicí servosmyčky se *Polohová odchylka* násobí ziskem a tím se získá povel rychlosti serva.

2.08 Zadaná rychlost. *Zadaná rychlost* (uživatelské jednotky/sekundu) je hodnota, kterou generuje generátor povelů osy DSM314. *Zadaná rychlost* indikuje povel okamžité rychlosti, který vytváří pohyb osy. Na začátku pohybu se zvýší velikost zrychlení a jakmile se dosáhne naprogramované rychlosti, stabilizuje se na hodnotě naprogramované rychlosti.

V režimu vlečené osy *Zadaná rychlost* pouze představuje výstup generátoru povelů osy. Master vstup vlečené osy nebo kontrolér rampy zrychlování vlečené osy nemají na *Zadanou rychlost* vliv.

2.09 Okamžitá rychlost. *Okamžitá rychlost* (uživatelské jednotky/sekundu) představuje rychlost osy odvozenou od zpětnovazebního zařízení a aktualizuje se v DSM314 jednou za 128 milisekund.

2.10 Uživatelem volitelná data 1. Pro každou ze čtyř os existuje jedno z těchto slov. Informace předávané v *Uživatelem volitelných datech 1* jsou určené konfigurací modulu (viz kapitola 4) nebo povelem %AQ *Volba dat návratu 1* (viz kapitola 4, "Okamžité povely %AQ" v této kapitole).

2.11 Uživatelem volitelná data 2. Pro každou ze čtyř os existuje jedno z těchto slov. Informace předávané v *Uživatelem volitelných datech 2* jsou určené konfigurací modulu (viz kapitola 4) nebo povelem %AQ *Volba dat návratu 2* (viz kapitola 4, "Okamžité povely %AQ" v této kapitole). Další informace najdete v kapitole 4 "Okamžité povely %AQ".

Část 3: Diskrétní povely %Q

Následující povely %Q představují diskrétní povely, které se posílají automaticky do DSM314 z CPU při každém cyklu PLC. Povel se vykoná nastavením odpovídajícího bitu do jedničky. Aktuální adresy bitů diskrétních povelů závisí na počáteční adrese nakonfigurované pro adresy %Q. Viz tabulka 4-1, "Záložka Nastavení". Offsety bitů uvedené v následující tabulce jsou offsety od této počáteční adresy. Čísla ve sloupci "Odstavec" se týkají částí v této kapitole.

Tabulka 5-5. Diskrétní povely %Q

Offset bitu	Popis	Osa	Odstavec	Offset bitu	Popis	Osa	Odstavec
00	Vynulování chyby	---	3.01	40	OUT1_B / zdroje konfig. bitů CTL	Servo 2	3.12
01	Povolení lokální logiky	---	3.02	41	OUT3_B / zdroje konfig. bitů CTL	Servo 2	3.13
02	Vykonání pohybového programu 1	---	3.03	42	*Vyhrazeno*		
03	Vykonání pohybového programu 2	---	3.03	43	*Vyhrazeno*		
04	Vykonání pohybového programu 3	---	3.03	44	Povolení vlečené osy	Servo 2	3.14
05	Vykonání pohybového programu 4	---	3.03	45	Volba master zdroje vlečené osy	Servo 2	3.15
06	Vykonání pohybového programu 5	---	3.03	46	*Vyhrazeno*		
07	Vykonání pohybového programu 6	---	3.03	47	*Vyhrazeno*		
08	Vykonání pohybového programu 7	---	3.03	48	Zrušit všechny pohyby	Servo 3	3.05
09	Vykonání pohybového programu 8	---	3.03	49	Zastavení posuvu (zastavení programu)	Servo 3	3.06
10	Vykonání pohybového programu 9	---	3.03	50	Povolení pohonu / MCON	Servo 3	3.07
11	Vykonání pohybového programu 10	---	3.03	51	Nalezení výchozí polohy	Servo 3	3.08
12	Zdroj konfigurovatelného bitu CTL	---	3.04	52	Jog Plus	Servo 3	3.09
13	Zdroj konfigurovatelného bitu CTL	---	3.04	53	Jog Minus	Servo 3	3.10
14	Zdroj konfigurovatelného bitu CTL	---	3.04	54	Reset vzorkování 1	Servo 3	3.11
15	Zdroj konfigurovatelného bitu CTL	---	3.04	55	Reset vzorkování 2	Servo 3	3.11
16	Zrušit všechny pohyby	Servo 1	3.05	56	OUT1_C / zdroje konfig. bitů CTL	Servo 3	3.12
17	Zastavení posuvu (zastavení programu)	Servo 1	3.06	57	OUT3_C / zdroje konfig. bitů CTL	Servo 3	3.13
18	Povolení pohonu / MCON	Servo 1	3.07	58	*Vyhrazeno*		
19	Nalezení výchozí polohy	Servo 1	3.08	59	*Vyhrazeno*		
20	Jog Plus	Servo 1	3.09	60	Povolení vlečené osy	Servo 3	3.14
21	Jog Minus	Servo 1	3.10	61	Volba master zdroje vlečené osy	Servo 3	3.15
22	Reset vzorkování 1	Servo 1	3.11	62	*Vyhrazeno*		
23	Reset vzorkování 2	Servo 1	3.11	63	*Vyhrazeno*		
24	OUT1_A / zdroje konfig. bitů CTL	Servo 1	3.12	64	Zrušit všechny pohyby	Servo 4	3.05
25	OUT3_A / zdroje konfig. bitů CTL.	Servo 1	3.13	65	Zastavení posuvu (zastavení programu)	Servo 4	3.06
26	*Vyhrazeno*			66	Povolení pohonu / MCON	Servo 4	3.07
27	*Vyhrazeno*			67	Nalezení výchozí polohy	Servo 4	3.08
28	Povolení vlečené osy	Servo 1	3.14	68	Jog Plus	Servo 4	3.09
29	Volba master zdroje vlečené osy	Servo 1	3.15	69	Jog Minus	Servo 4	3.10
30	*Vyhrazeno*			70	Reset vzorkování 1	Servo 4	3.11
31	*Vyhrazeno*			71	Reset vzorkování 2	Servo 4	3.11
32	Zrušit všechny pohyby	Servo 2	3.05	72	OUT1_B / Config. CTL bit src.	Servo 4	3.12
33	Zastavení posuvu (zastavení programu)	Servo 2	3.06	73	OUT3_B / Config. CTL bit src.	Servo 4	3.13
34	Povolení pohonu / MCON	Servo 2	3.07	74	*Vyhrazeno*	Servo 4	
35	Nalezení výchozí polohy	Servo 2	3.08	75	*Vyhrazeno*	Servo 4	
36	Jog Plus	Servo 2	3.09	76	Povolení vlečené osy	Servo 4	3.14
37	Jog Minus	Servo 2	3.10	77	Volba master zdroje vlečené osy	Servo 4	3.15
38	Reset vzorkování 1	Servo 2	3.11	78	*Vyhrazeno*	Servo 4	
39	Reset vzorkování 2	Servo 2	3.11	79	*Vyhrazeno*	Servo 4	

3.01 Vynulování chyby. Když se bude hlásit chyba, tento povel se použije k vynulování stavového bitu %I *Výskyt chyby modulu* i souvisejících stavových slov %AI *Kód stavu modulu* a *Chybový kód osy 1 - osy 4*. Chybové stavy, které stále zůstávají (například chyba koncového spínače *Konec posuvu*), se nevynulují a je nutno je vynulovat nějakou jinou nápravnou akcí. Pokud bit *Vynulování chyby* zůstane v jedničce, povel Jog je možno použít k odjezdu ze sepnutého koncového spínače hardwarového přejetí.

3.02 Povolení lokální logiky. Tento povel povoluje vykonání aktuálního programu lokální logiky v DSM. Informace o nakonfigurování názvu programu lokální logiky najdete v kapitole 4.

3.03 Vykonání pohybového programu 1 - 10. Tyto povely se používají ke zvolení uložených pohybových programů k okamžitému vykonání. Každý povel používá jednorázovou akci; proto, když se má vykonat program, povelový bit musí vždy přejít z nuly do jedničky. Programy je možno přechodně zastavit povelem *Zastavení posuvu*.

Když se program začne vykonávat, přepis rychlosti je vždy nastavený na 100%**.** Povel %AQ *Přepis rychlosti* je možno poslat při stejném cyklu jako bit %Q *Vykonání pohybového programu n* a bude v platnosti od okamžiku startu programu.

V jedné ose je možno najednou vykonávat pouze jeden pohybový program. Stavový bit %I *Program aktivní* musí být v nule, jinak se pohybový program nebude moct spustit. Pohybový program pro více os používá osu 1 a osu 2, takže aby se pohybový program pro více os mohl spustit, oba bity *Program aktivní* musí být v nule.

3.04 Zdroje konfigurovatelného bitu CTL. Offsety bitu %Q 12-15 jsou konfigurovatelné jako zdroje pro CTL bity CTL01-CTL24. Další informace najdete v kapitole 4. Výchozí konfigurace je:

Offset bitu %Q 12: CTL09

Offset bitu %Q 13: CTL10

Offset bitu %Q 14: CTL11

Offset bitu %Q 15: CTL12

3.05 Zrušit všechny pohyby. Tento povel způsobí, že <u>všechny</u> probíhající pohyby se zastaví aktuální velikostí *Zrychlení jogu* a nakonfigurovaným *režimem zrychlení jogu***. Proto je důležité použít *Zrychlení jogu*, které provede zpomalení během vyhovující vzdálenosti**. Všechny čekající programy nebo okamžité povely se zruší a proto nemají povolenou aktivaci. Stav zrušení zůstane v platnosti, dokud tento povel bude aktivní. Pokud se vykonával pohyb, když se přijal tento povel, stavový bit *Pohyb* zůstane v jedničce, dokud zadaná rychlost nedosáhne nuly.

3.06 Zastavení posuvu (přechod do jedničky). Tento povel způsobí, že všechny probíhající pohybové programy zastaví aktivní velikostí zrychlení programu. Povel *Zastavení posuvu* nezastaví pohyb zadaný master zdrojem v režimu Vlečená osa povolena. Jakmile se pohyb zastaví, stavový bit *Pohyb* se vynuluje a stavový bit *V poloze* se nastaví do jedničky, když se dosáhne stavu *V poloze*. Povely *Jog* jsou povolené ve stavu *Zastavení posuvu*. Po přechodu do jedničky se naprogramovaný pohyb zastaví, i když povelový bit přejde zpět do nuly před zastavením pohybu.

Zastavení posuvu (přechod do nuly). Tento povel způsobí, že se všechny pohybové programy přeruší *Zastavením posuvu* a obnoví se naprogramovaným zrychlením a rychlostí. Zpracují se pak další naprogramované pohyby a bude pokračovat normální vykonávání programu. *Zastavení posuvu* přechod do nuly se chová podobně jako povel *Vykonání programu* ale s tou výjimkou, že software pro generování dráhy používá v programu pouze zbývající vzdálenost.

Pokud se vyskytlo jogování, když bylo aktivní *Zastavení posuvu*, přerušený povel *Pohyb* se obnoví z místa, kde se osa nacházela po *Jogu*. *Pohyb* se dokončí správnou naprogramovanou rychlostí a bude pokračovat do původní naprogramované polohy, jako by se žádné posunutí v důsledku *jogu* nevyskytlo.

3.07 Povolení pohonu / MCON. Pokud stavový bit %I *Výskyt chyby modulu* a *Pohon povolen* budou vynulované, tento povel způsobí, že kontakt relé pro povolení pohonu se sepne a bit %I *Pohon povolen* se nastaví do jedničky. Když bit %I *Pohon povolen* bude v jedničce, funkce generování dráhy a řízení polohy budou povolené a je možno zadat pohyb serva. Do digitálního serva se pošle signál (*MCON*), který povolí pohon. Aby se povolil normální pohyb serva, *Povolení pohonu* musí zůstat v jedničce (kromě případu, kdy se používají povely *Jog*). Pokud se používá okamžitý povel *Vynucený analogový výstup* (viz část 4.06, "Vynucený analogový výstup"), příslušný signál *Povolení pohonu* musí být v jedničce, aby se tímto povelem vygeneroval analogový výstup.

3.08 **Nalezení výchozí polohy.** Tento povel způsobí, že DSM314 nastaví ***Výchozí polohu***. Vstup snímače výchozí polohy z I/O konektoru přibližně udává referenční výchozí polohu a další zjištěná značka snímače polohy indikuje přesnou výchozí polohu. Když konfigurace osy ***Režimu výchozí polohy*** bude nastavený na MOVE+ nebo MOVE-, vstup spínače výchozí polohy se bude ignorovat. U servoosy nakonfigurovaný ***Offset výchozí polohy*** definuje polohu ***Výchozí polohy*** jako offsetovou vzdálenost od značky výchozí polohy. Indikace bitu %I *Poloha platná* se nastaví do jedničky při dokončení cyklu nájezdu do výchozí polohy. ***Informace o přídavném cyklu nájezdu do výchozí polohy najdete v kapitole 6. Informace o absolutním snímači polohy najdete v Dodatku C.***

3.09 **Jog Plus.** Když tento povelový bit bude v jedničce, osa se bude pohybovat v kladném směru nakonfigurovaným *zrychlením jogu* a *rychlostí jogu*. Nastavení *Jogu Plus* do nuly bude mít za následek zpomalení a zastavení. Pokud *Jog Plus* bude momentálně v nule, i třeba na jeden cyklus PLC, osa se zpomalí až do zastavení, pak zrychlí a bude pokračovat jogem. Osa se bude pohybovat, dokud povel *Jog Plus* bude platný a nezjistí se nakonfigurovaná softwarová mez ***kladný konec posuvu*** nebo kladný koncový spínač přejetí. Vstupy koncových spínačů přejetí je možno zakázat pomocí konfiguračního parametru ***OT Limit***. ***Jog Plus*** **je možno použít k odjezdu ze záporného koncového spínače přejetí jogem, pokud také bit %Q** ***Vynulování chyby*** **bude v jedničce.** ***Více informací o jogování s DSM314 najdete v kapitole 6, Neprogramovaný pohyb.***

3.10 **Jog Minus.** Když tento povelový bit bude v jedničce, osa se bude pohybovat v záporném směru nakonfigurovaným *zrychlením jogu* a *rychlostí jogu*. Nastavení *Jogu Minus* do nuly bude mít za následek zpomalení a zastavení. Pokud *Jog Minus* bude momentálně v nule, i třeba na jeden cyklus PLC, osa se zpomalí až do zastavení, pak zrychlí a bude pokračovat jogem. Osa se bude pohybovat, dokud povel *Jog Minus* bude platný a nezjistí se nakonfigurovaná softwarová mez ***záporný konec posuvu*** nebo záporný koncový spínač přejetí. Vstupy koncových spínačů přejetí je možno zakázat pomocí konfiguračního parametru ***OT Limit***. ***Jog Minus*** **je možno použít k odjezdu ze kladného koncového spínače přejetí jogem, pokud také bit %Q** ***Vynulování chyby*** **bude v jedničce.** ***Více informací o jogování s DSM314 najdete v kapitole 6, Neprogramovaný pohyb.***

3.11 Reset vzorkování příznaku 1, 2. Stavový bit %I *Vzorkování příznaku n* informuje PLC, že vzorkovací vstup zachytil polohu osy, která je nyní uložená v souvisejícím stavovém slově %AI *Vzorkování polohy n*. Když PLC rozpozná tato data, může použít povelový bit %Q příznaku *Reset vzorkování n* k vynulování stavového bitu příznaku *Vzorkování příznaku n*. Jakmile se bit %I *Vzorkování příznaku n* nastaví do jedničky, další vzorkovací vstupy zachycení nových dat nevyvolají. Příznak se musí vynulovat před tím, než se zachytí jiná vzorkovaná poloha. Dokud povelový bit %Q příznaku *Reset vzorkování n* bude v jedničce, bit *Vzorkování příznaku n* zůstane ve vynulovaném stavu. Za tohoto stavu se poslední poloha vstupu vzorkování zobrazí ve stavovém slově *Vzorkování polohy n*, i když PLC nemůže použít příznak k indikaci, když přijdou nová data.

3.12 Řízení výstupu OUT1_A, B, C, D / Zdroj konfigurovatelného bitu CTL. Každý konektor osy má výstup vedený přes 24 V polovodičové relé (SSR) se zatížením 125 mA. Bity *%Q řízení výstupu OUT1_A, OUT1_B, OUT1_C* a *OUT1_D* mohou řídit stav souvisejícího výstupu, ale pouze pokud související výstupní bity budou nastavené pro řízení PLC. Informace o konfiguraci najdete v kapitole 4.

Pro jednotlivé osy jsou přiřazené následující svorky:

	Pin na konektoru čelní desky	**Svorka na pomocné svorkovnici IC693ACC336**	**Svorka na svorkovnici serva IC693ACC335**
Svorka OUT1 SSR (+)	18	18	8
Svorka OUT1 SSR (-)	36	36	16

Tyto bity %Q jsou možno použít také jako zdroje pro konfigurovatelné bity CTL nezávisle na konfiguraci výstupních bitů. Informace o konfiguraci zdrojů bitů CTL01-CTL24 najdete v kapitole 4.

Poznámka: Bity OUT_1A, B, C, D nebudou řídit výstupy čelní desky, dokud související výstupní bity nebudou nakonfigurované na řízení PLC. Informace o konfiguraci najdete v kapitole 4.

3.13 Řízení výstupu OUT3_A, B, C, D / Zdroj konfigurovatelného bitu CTL. Konektor každé osy má diferenciální výstup 5V, který je vhodný pro řízení 5 V zátěží TTL nebo CMOS. Bity *%Q řízení výstupu OUT3_A, OUT3_B, OUT3_C* a *OUT3_D* mohou řídit stav souvisejícího výstupu, ale pouze pokud související výstupní bity budou nastavené pro řízení PLC. Informace o konfiguraci najdete v kapitole 4.

Pro jednotlivé osy jsou přiřazené následující svorky:

	Pin na konektoru čelní desky	**Svorka na pomocné svorkovnici IC693ACC336**	**Svorka na svorkovnici serva IC693ACC335**
Svorka OUT3 (+)	14	14	5
Svorka OUT3 (-)	32	32	13

Poznámka: Bity OUT_3A, B, C, D nebudou řídit výstupy čelní desky, dokud související výstupní bity nebudou nakonfigurované na řízení PLC. Informace o konfiguraci najdete v kapitole 4.

Tyto bity %Q jsou možno použít také jako zdroje pro konfigurovatelné bity CTL nezávisle na konfiguraci výstupních bitů. Informace o konfiguraci zdrojů bitů CTL01-CTL24 najdete v kapitole 4.

3.14 Povolení vlečené osy. Když tento bit bude nastavený do jedničky a stavový bit %I *Vlečená osa povolena* indikuje, že vlečená osa je povolena, pohyb zadaný externím nebo interním masterem bude působit jako vstup do smyčky vlečené osy. Přídavný bit pro aktivaci vlečené osy je možno nakonfigurovat tak, že vyvolá pohyb vlečené osy. Když se používá aktivace vlečené osy, *Povolení vlečené osy* musí být pro testovaný stav aktivace v jedničce. Vynulování bitu *Povolení vlečené osy* odpojí smyčku vlečené osy od master zdroje. Povely *Jog, Pohyb rychlostí* a *Vykonat program n* budou povolené bez ohledu na stav bitu *Povolení vlečené osy*. Když vlečená osa bude povolená, povely *Jog*, *Pohyb rychlostí* nebo *Vykonat program n* se přepočítají do povelu master rychlosti nebo polohy. *Nalezení výchozí polohy* nebude povoleno, dokud se *Povolení vlečené osy* nevynuluje. Další informace najdete v kapitole 8. Tento bit se používá pouze v režimu vlečené osy.

3.15 Volba master zdroje vlečené osy. Tento bit přepíná zdroj master osy pro vlečenou osu z Master zdroje vlečené osy 1 (bit v nule) na Master zdroje vlečené osy 2 (bit v jedničce). Zdroje masteru vlečené osy jsou konfigurovatelné jako Zadaná poloha nebo Okamžitá poloha z kterékoliv ze 4 os.

Část 4: Okamžité povely %AQ

Následující slova okamžitých povelů %AQ se přenášejí při každém cyklu PLC z CPU dat %AQ do DSM314. Nakonfigurovaný počet slov %AQ (6, 9 nebo 12) závisí na nakonfigurovaném počtu řízených os. Aktuální adresy slov okamžitého povelu závisí na počáteční adrese nakonfigurované pro slova %AQ. Viz tabulka 4-1, "Záložka Nastavení". Offsetová čísla slov uvedená v následující tabulce jsou offsety od počáteční adresy. Slova jsou přiřazená následovně:

Tabulka 5-6. Přiřazení slov %AQ

Offset slova	Popis	Osa
00	Slovo okamžitého povelu	Servo 1
01-02	Data povelu	Servo 1
03	Slovo okamžitého povelu	Servo 2
04-05	Data povelu	Servo 2
06	Slovo okamžitého povelu	Servo 3
07-08	Data povelu	Servo 3
09	Slovo okamžitého povelu	Servo 4
10-11	Data povelu	Servo 4

Při každém cyklu PLC je možno na každou osu poslat pouze jeden okamžitý povel %AQ, jedinou výjimkou je povel *Okamžité načtení parametru*, který je nezávislý na ose. Počet povelů *Okamžité načtení parametru*, které je možno poslat během jednoho cyklu, závisí na počtu nakonfigurovaných slov %AQ (podrobnosti viz tabulka 5-8).

I když se povely posílají každý cyklus, DSM314 bude reagovat na povel, POUZE pokud se od posledního cyklu změnil. Když se změní některé ze 3 slov, DSM314 přijme data jako nový povel a odpoví odpovídajícím způsobem.

Bit %I *Osa OK* musí být v jedničce, aby se přijal nový okamžitý povel %AQ. Za stejných podmínek, například odpojený digitální snímač polohy, servozesilovač bez napájení nebo nevynulovaná chyba, *Osa OK* bude v nule a zpracování povelu %AQ pro tuto osu bude zakázáno. Pokud se digitální servoosa 1 nebo 2 nebude používat pro řízení motoru, nakonfigurovaný ***Typ motoru*** musí být nastavený na 0, jinak by se hlásila chyba a *Osa OK* zůstane v nule.

6-bajtový formát okamžitých povelů je definovaný v tabulce 5-7. Aktuální adresy slov okamžitého povelu závisí na počáteční adrese nakonfigurované pro adresy %AQ. **Čísla slov uvedená v následující tabulce jsou offsety od počáteční adresy.**

Offsety slov jsou uvedené v opačném pořadí a v hexadecimálním tvaru, aby se zjednodušil zápis dat. Následující příklad posílá do osy 1 povel Nastavit polohu. První slovo, slovo 0, obsahuje skutečné číslo povelu. Pro povel Nastavení polohy číslo povelu je 0023h. Druhé a třetí slovo obsahuje data pro povel Nastavení polohy, což je poloha. Druhé slovo, slovo 1, je nejméně významné slovo polohy a třetí slovo, slovo 2, je nejvýznamnější slovo.

Příklad:

Má-li se nastavit poloha 3 400 250, nejdříve je nutno hodnotu převést na hexadecimální tvar. Hodnota 3 400 250 dekadicky se rovná 0033E23A hexadecimálně. Pro tuto hodnotu je 0033 nejvýznamnější slovo a E23A je nejméně významné slovo. Data, která se mají poslat do DSM314, jsou:

Slovo 2	*Slovo 1*	*Slovo 0*	*Povel*
0033	E23A	0023	Nastavení polohy 3 400 250

Když se v referenční tabulce nastaví slovo 0 jako hexadecimální slovo a slova 1 a 2 jako celé číslo s dvojitou délkou, zjednoduší se tím zápis okamžitého povelu.

Hodnoty mezních dat MaxPosnUu, MaxVelUu a MaxAccUu se vypočítají následovně:

Vztahy pro výpočet proměnných mezních dat		
Mez polohy MaxPosnUu	**Mez rychlosti MaxVelUu**	**Mez zrychlení MaxAccUu**
IF uu:cts >= 1:1 MaxPosnUu = 536,870,912 ELSE (uu:cts < 1:1) MaxPosnUu = 536,870,912 * uu/cts	MaxVelUu = 1,000,000* uu/cts	IF uu:cts >= 1:1 MaxAccUu = 1,073,741,823 ELSE (uu:cts < 1:1) MaxAccUu = 1,073,741,823* uu/cts

V následující tabulce povelů %AQ jsou uvedené pouze offsety slov pro servoosu 1. Offsety slov pro ostatní osy se vypočítají tak, že k uvedeným offsetům slov se přičte 3 (servoosa 2), 6 (servoosa 3) nebo 9 (servoosa 4). Čísla ve sloupci "Odstavec" udávají číslo referenčního odstavce v této kapitole.

Tabulka 5-7. Okamžité povely %AQ používající 6- bajtový formát

Slovo 2		Slovo 1		Slovo 0			
Bajt 5	**Bajt 4**	**Bajt 3**	**Bajt 2**	**Bajt 1**	**Bajt 0**	**Definice okamžitého povelu**	**Odstavec**
xx	xx	xx	xx	00	00h	**Prázdný**	**4.01**
xx	xx	xx	RO%	00	20h	**Přepis rychlosti** RO% = 0 ...120%	**4.02**
xx	xx	*	Incr	00	21h	**Polohový inkrement bez aktualizace polohy** Inkrementace = -128 ... +127 uživatelských jednotek	**4.03**
Rychlost				00	22h	**Pohyb rychlostí** Rychlost = -MaxVelUu … +MaxVelUu	**4.04**
Poloha				00	23h	**Nastavení polohy** Poloha = -MaxPosnUu ... + MaxPosnUu-1	**4.05**
xx	xx	Analogový výstup		00	24h	**Vynucený analogový výstup** Analogový výstup = -32,000 ... + 32,000	**4.06**
xx	xx	*	Inkrementace	00	25h	**Polohový inkrement s aktualizací polohy** Inkrementace= -128 ... +127 uživatelských jednotek	**4.07**
xx	xx	xx	Zóna dosažení polohy	00	26h	**Zóna dosažení polohy** Rozsah = 0 ... 255	**4.08**
Poloha nebo číslo parametru				Typ pohybu	27h	**Povel pohybu** Poloha = -MaxPosnUu ... + MaxPosnUu-1 Par # = 0 ... 255	**4.09**
Rychlost				00	28h	**Rychlost jogu** Rychlost = +1 … +MaxVelUu	**4.10**
Zrychlení				00	29h	**Zrychlení jogu** Zrychlení = +1 ... + MaxAccUu	**4.11**
xx	xx	Časová konstanta (jednotky 0,1 ms)		00	2Ah	**Časová konstanta polohové smyčky** Časová konstanta = 0 - 65535 (jednotky 0,1 ms)	**4.12**
xx	xx	VFF (jednotky 0,01%)		00	2Bh	**Posuv rychlostí** VFF = 0 ... 12000 (jednotky 0,01%)	**4.13**
xx	xx	Časová konstanta integrátoru		00	2Ch	**Časová konstanta integrátoru** Časová konstanta = 0, 10 ... 10 000 ms	**4.14**
Poměr B		Poměr A		00	2Dh	**Poměr vlečené osy A/B** Poměr A = –32,768 … +32,767 Poměr B = +1 ... +32,767	**4.15**
xx	xx	xx	VLGN	00	2Eh	**Zisk rychlostní smyčky (pouze digitální režim)** VLGN = 0 ... 255	**4.16**
xx	xx	Mez krouticího momentu (jednotky 0,01%)		00	2Fh	**Mez krouticího momentu (pouze digitální režim a režim analogového kroutícího momentu)** Rozsah = 0-10000 (jednotky 0,01%)	**4.17**

Tabulka 5-7. - pokračování - %Okamžité povely %AQ používající 6- bajtový formát

Slovo 2		Slovo 1		Slovo 0			
Bajt 5	**Bajt 4**	**Bajt 3**	**Bajt 2**	**Bajt 1**	**Bajt 0**	**Definice okamžitého povelu**	**Odstavec**
Poloha				00	31h	**Nastavení polohy pomocného snímače polohy** Poloha = -MaxPosnUu ... + MaxPosnUu-1	**4.18**
xx	xx	Povel rychlosti serva (**Poznámka:** Nepoužívá se v analogovém rychlostním režimu - viz povel vynuceného D/A výstupu)		00	34h	**Vynucená rychlost serva** Povel rychlosti serva = -4,095 ... +4,095 OT./MIN	**4.19**
xx	xx	Offset		Režim	40h	**Volba dat návratu 1**	**4.20**
xx	xx	Offset		Režim	41h	**Volba dat návratu 2**	**4.21**
xx	xx	Čas úpravy náběhu		00	42h	**Doba úpravy vzdálenosti náběhu vlečené osy** Aktivní rozsah = 0, 10 ... 32000 ms	**4.22**
xx	xx	KpVel		07	46h	**Proporcionální zisk regulátoru rychlosti** (pouze režim analogového kroutícího momentu) KpVel = 0 - 32767	**4.23**
xx	xx	KiVel		08	46h	**Integrační zisk regulátoru rychlosti** (pouze režim analogového kroutícího momentu) KiVel = 0 - 32767	**4.24**
xx	xx	Režim TqFilt		0A	46h	**Filtr povelu kroutícího momentu** (pouze režim analogového kroutícího momentu) TqFilt = 0 - 3	**4.25**
xx	xx	Režim		Osa	47h	**Volba analogového výstupního režimu (pouze digitální režim)**	**4.26**
xx	xx	xx	xx	00	49h	**Vynulovat novou přijatou konfiguraci**	**4.27**
Data parametrů				Par #h	50h	**Okamžité načtení parametru** Par # = 0 ... 255 Data parametrů = Rozsah závisí na použití parametru.	**4.28**

* = Jsou přípustné pouze 00 nebo FFh.
xx = libovolné

4.01 Prázdný. Toto je výchozí okamžitý povel %AQ. Protože slova %AQ se automaticky přenášejí při každém cyklu PLC, povel *Prázdný* se musí použít vždy, aby nedošlo k náhodnému vykonání jiného okamžitého povelu %AQ.

4.02 Přepis rychlosti. Tento povel okamžitě změní % hodnoty rychlosti posuvu, který změní zadanou rychlost pro všechny následující naprogramované pohyby. Tato nová hodnota bude platná okamžitě, jakmile jí DSM314 přijme. Uloží se a zůstane v platnosti, dokud nebude přepsaná jinou hodnotou. Přepis rychlosti nemá žádný vliv na neprogramované pohyby nebo zrychlení. Při každém spuštění programu je *Přepis rychlosti* nastavený na 100%. Povel *Přepis rychlosti* je možno poslat při stejném cyklu PLC jako bit %Q *Vykonat program* a hodnota přepisu bude platná okamžitě. Přepis rychlosti je možno použít k upravení naprogramované rychlosti (ne zrychlení) konkrétního pohybu nebo souboru pohybů kterékoliv osy.

4.03 Polohový inkrement bez aktualizace polohy. (Uživatelské jednotky) Tento povel provede offset polohy osy o -128 až +127 uživatelských jednotek bez aktualizace *Okamžité polohy* nebo *Zadané polohy*. Pokud servo bude povolené, DSM314 okamžitě provede posuv osy o zadaný inkrement. Polohové inkrementy je možno použít k vytvoření menších oprav strojní polohy tak, aby se kompenzovala změna aktuálního stavu. ***Více informací o povelech polohového inkrementu pomocí DSM314 najdete v kapitole 6, "Neprogramované pohyby".***

4.04 Pohyb rychlostí. (Uživatelské jednotky/sekundu) Tento povel vykonává PLC a provádí pohyb osy konstantní rychlostí. Aktivní velikost ***Zrychlení jogu*** a nakonfigurovaný ***Režim zrychlení*** se používají pro povely *Pohyb rychlostí*. Když se během těchto pohybů dosáhne nakonfigurované horní nebo dolní meze, data okamžité polohy osy se na těchto mezích překlopí. ***Více informací o povelu Pohyb rychlostí najdete v kapitole 6, "Neprogramovaný pohyb".***

4.05 Nastavení polohy. (Uživatelské jednotky) Tento povel změní hodnoty registru polohy osy, aniž by osa vykonala pohyb. Činnost povelu závisí na konfiguraci osy:

Servoosa – Hodnoty *Zadaná poloha* a *Okamžitá poloha* se změní tak, že se nebude generovat žádný pohybový povel. *Okamžitá poloha* se nastaví na určenou hodnotu a *Zadaná poloha* se nastaví na nastavenou hodnotu + *Polohová odchylka*. *Nastavení polohy* nelze provést, když bit %I *Pohyb* nebo bit %I *Program aktivní* bude v jedničce. *Nastavení polohy* bude povoleno, když bit %I *V poloze* bude v nule, přičemž *Okamžitá rychlost* je ≤100 pulsů/sec. Hodnota polohy musí být v mezích pohybu a v mezích čítání, jinak se bude hlásit chyba. Bit %I *Poloha platná* se nastaví do jedničky po úspěšném povelu *Nastavení polohy*. V dodatku C najdete poznámky týkající se používání absolutního snímače polohy. Povel *Nastavení polohy* se běžně používá k nastavení počáteční polohy referenčního bodu do nuly (nebo jiné hodnoty) bez provedení nájezdu osy do výchozí polohy.

Pomocná osa – *Zadaná poloha* se zapíše do dat povelu. Pro pomocnou osu je *Okamžitá poloha* nezávislá na *Zadané poloze* a *Nastavení polohy* na ní nemá vliv. **V odstavci 4.18 *Nastavení polohy pomocného snímače* najdete jak nastavit *Okamžitou polohu* pomocného snímače polohy.** *Nastavení polohy* nelze provést, když bit %I *Pohyb* nebo bit %I *Program aktivní* bude v jedničce. Hodnota polohy musí být v mezích pohybu, jinak se bude hlásit chyba.

> **Poznámka:** Když digitální servosystém GE Fanuc s absolutním snímačem polohy (*Režim zpětné vazby* = absolutní) se poprvé zapne po vyjmutí nebo výměně baterie snímače polohy, snímač polohy se musí otočit za jeho interní referenční bod. Pokud by se toto neprovedlo, povel *Nastavení polohy* se bude ignorovat a bude se hlásit chybový kód 53h (Nastavení polohy před tím, než snímač polohy přejde referenčním bodem).

4.06 Vynucený analogový výstup. Každý konektor osy podporuje jeden analogový výstupní signál. Okamžitý povel *Vynucený analogový výstup* je možno použít v aplikačním programu PLC k nastavení hodnoty tohoto stejnosměrného výstupního napětí. Povel *Vynucený analogový výstup* ovládá jeden z analogových výstupů na konektoru C nebo D čelní desky DSM v digitálním režimu nebo v analogovém rychlostním režimu na konektoru A, B, C nebo D. K ovládání výstupů na různých konektorech s použitím příslušných offsetů slova %AQ (viz odstavec před tabulkou 5-7) je možno použít několik povelů *Vynucený analogový výstup*. Povel *Vynucený analogový výstup* má rozsah +32000 (+10.00 V ss) až -32000 (-10.00 V ss). Když osa bude nakonfigurovaná v režimu analogového kroutícího momentu, povel Vynucený analogový výstup NELZE použít.

Aby se aktivovaly analogové výstupní hodnoty nastavené tímto povelem, je nutné povolit příslušný bit %Q "Povolení pohonu" (pro každou osu je jen jeden).

> **Poznámka:** To je odlišné od funkce IC693DSM302.

K udržení napětí vynuceného analogového výstupu jsou dva požadavky: (1) povel *Vynucený analogový výstup* a hodnota musí zůstat trvale v datech %AQ a, (2) související bit %Q "Povolení pohonu" musí být v jedničce. Bit %Q "Povolení pohonu" je možno použít k zapnutí a vypnutí analogového výstupního napětí.

Když povel Vynucený analogový výstup bude aktivní pro danou osu, povel *Vynucený analogový výstup* je možno zrušit a vypnout související analogové napětí některým jiným okamžitým povelem %AQ pro tuto osu.

Při používání tohoto povelu existují mezi digitálním a analogovým *režimem osy* rozdíly, které jsou popsané níže:

Digitální režim

- Povel *Vynucený analogový výstup* je možno použít pouze na konektorech C a D v digitálním režimu (v digitálním režimu musí být osa 1 a osa 2 na konektorech A a B digitální). Ve skutečnosti *Vynucený analogový výstup* je výchozí signál na konektorech C a D v digitálním režimu.
- Pokud osa 1 a 2 (konektory A a B) budou nakonfigurované jako digitální servo, jejich analogové výstupy se budou používat pouze pro ladění serva a tuto funkci nelze změnit povelem *Vynucený analogový výstup*. Zadání povelu *Vynucený analogový výstup* na digitální servoosu (konektor A nebo B) nebude mít žádný vliv a nebude se hlásit žádná chyba.
- V digitálním režimu je možno signál *Vynucený analogový výstup* nahradit, pokud povel *Volba režimu analogového výstupu* přivede jiný signál na konektor C nebo D. Pokud výchozí povel *Vynucený analogový výstup* byl změněný na konektorech C nebo D, je možno ho vracet do původního stavu buď (1) zadáním okamžitého povelu *Zvolit analogový výstup* (kód signálu 00) do každé dotčené osy nebo (2) vypnutím a zapnutím napájení DSM314. Viz kapitola 4.25, "Volba analogového výstupního režimu".

Příklad Vynuceného analogového výstupu (digitální režim)

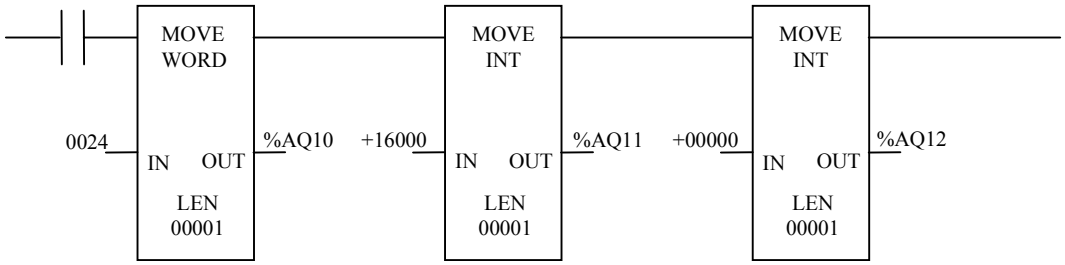
V tomto příkladu jsou osa 1 a osa 2 nakonfigurované jako digitální, počáteční adresa %Q DSM314 je nakonfigurovaná jako %Q1 a počáteční adresa %AQ je nakonfigurovaná jako %AQ1. Konektory C a D jsou nastavené do výchozího stavu analogového výstupu (*Vynucený analogový výstup*).

Má-li se na konektoru D vynutit +5 V ss, v programu žebříkové logiky se vydá okamžitý povel *Vynucený analogový výstup*. Protože první slovo %AQ bylo nakonfigurováno jako %AQ1, tři slova vztahující se ke konektoru D ("osa 4") jsou %AQ10, %AQ11 a %AQ12 (podrobnosti viz odstavec nad tabulkou 5-7). Protože %Q1 bylo nakonfigurováno jako první bit %Q, bit *Povolení pohonu (Servo 4)* pro osu 4 je %Q67 (viz tabulka 5-5, "Diskrétní povely %Q").

Takže následující hodnoty je nutno přesunout do příslušných slov pomocí instrukce Přesunutí v žebříkové logice (použití instrukce Přesunutí typu WORD usnadní přesun hexadecimálního čísla):

- %AQ10 Nastavte na 24h (což zadává povel *Vynucený analogový výstup*)
- %AQ11 Nastavte na +16000 (což se rovná +5 V ss)
- %AQ12 nastavte na 0 (toto slovo se nepoužívá k přenosu významných dat)

Kromě toho bit %Q67 (*Povolení pohonu*) musí být nastavený na logickou jedničku.



Analogový rychlostní režim

- V analogovém rychlostním režimu je možno povel *Vynucený analogový výstup* použít na všech čtyřech konektorech k vynucení napěťového výstupu.
- Povel *Zvolit analogový výstup* popisovaný v odstavci "Digitální režim" výše nefunguje v analogovém režimu.

Režim analogového kroutícího momentu

- V režimu analogového kroutícího momentu povel *Vynucený analogový výstup* NELZE použít.

4.07 Polohový inkrement s aktualizací polohy. (Uživatelské jednotky) Tento povel je podobný povelu *Polohový inkrement bez aktualizace polohy* (#21h) s tou výjimkou, že *Okamžitá poloha* a *Zadaná poloha* (vrácené v datech %AI) se aktualizují o hodnotu inkrementu. Pokud servo bude povoleno, DSM314 okamžitě provede posuv v ose o hodnotu inkrementu. Polohové inkrementy je možno použít k vytvoření menších oprav strojní polohy tak, aby se kompenzovala změna aktuálního stavu. ***Více informací o povelech polohového inkrementu pomocí DSM314 najdete v kapitole 6, "Neprogramované pohyby".***

4.08 Zóna dosažení polohy. (Uživatelské jednotky) Tento povel je možno použít k nastavení aktivní ***Zóny dosažení polohy*** na jinou hodnotu, než je nakonfigurovaná hodnota.

DSM314 porovná ***Zónu dosažení polohy*** s *Polohovou odchylkou*, aby mohlo řídit bit %I *V poloze*. Když *Polohová odchylka* bude ≤ ***Zóna dosažení polohy***, bit %I *V poloze* bude v jedničce.

Pokud se s DSM314 provede vypnutí a zapnutí napájení nebo se z nějakého důvodu provede reset CPU PLC, hodnota nastavená tímto povelem se ztratí a obnoví se hodnota zóny *V poloze* nastavená konfiguračním softwarem.

4.09 Povel pohybu. Tento povel vykoná jeden pohybový profil, který při každém zadání provede pohyb v ose do zadané polohy. Pro pohyb se použije aktuální ***Zrychlení jogu*** a ***Rychlost jogu*** (které je možno změnit také pomocí povelů %AQ). Povel PMOVE se nedokončí (bit %I *Program aktivní* přejde do nuly), dokud se nedosáhne *Zadané polohy* a bit %I *V poloze* nebude v jedničce. Povel CMOVE se dokončí (bit %I *Program aktivní* přejde do nuly), kdykoliv se dosáhne *Zadané polohy*, i když bit *V poloze* bude v nule. Proto se CMOVE dokončí, i když se ještě nedosáhne *Okamžité polohy* CMOVE. Bit %I *Program aktivní* je možno monitorovat a tak zjistit, kdy povel AQ Pohyb je aktivní.

Datové pole pro tento povel může obsahovat polohu nebo vzdálenost pro pohyb v bytech 2-5 s typem povelu (v hexadecimálním formátu) podle následující definice:

Typ pohybu (bajt 1):

00h = Abs, Pmove, Lineární
01h = Abs, Cmove, Lineární
10h = Abs, Pmove, S křivka
11h = Abs, Cmove, S křivka
40h = Inc, Pmove, Lineární
41h = Inc, Cmove, Lineární
50h = Inc, Pmove, S křivka
51h = Inc, Cmove, S křivka

Datové pole pro tento povel může obsahovat číslo parametru v bajtu 2 (bajty 3 - 5 se nepoužívají) s typem povelu podle následující definice:

Typ pohybu (bajt 1):

80h = Abs, Pmove, Lineární
81h = Abs, Cmove, Lineární
90h = Abs, Pmove, S křivka
91h = Abs, Cmove, S křivka
C0h = Inc, Pmove, Lineární
C1h = Inc, Cmove, Lineární
D0h = Inc, Pmove, S křivka
D1h = Inc, Cmove, S křivka

Povel *Přesunutí* se vykoná jako pohybový program s jediným pohybem. Proto všechna omezení, která platí pro vykonání pohybového programu, také platí pro *Povel pohybu*. Pokud například program už bude aktivní pro osu 1, pak snaha poslat tento povel pro osu 1 bude mít za následek hlášení chyby.

4.10 Rychlost jogu. (Uživatelské jednotky/sekundu) Tento povel nastaví rychlost používanou, když se bit %Q *Jog* používá pro vykonání jogu v kladném nebo záporném směru. *Rychlost jogu* používají pohybové programy, když program neobsahuje žádný povel rychlosti. Povel pohybu %AQ (27h) používá vždy *Rychlost jogu*. Reset nebo vypnutí a zapnutí napájení PLC vrátí tuto hodnotu do nakonfigurovaných dat.

4.11 Zrychlení jogu. (Uživatelské jednotky/sekundu/sekundu) Tento povel nastaví hodnotu zrychlení používanou při operacích *Jog*, *Nalezení výchozí polohy*, *Pohyb rychlostí, Zrušit všechny pohyby* a **Normální zastavení**. K **Normálnímu zastavení** dojde, když PLC přejde ze stavu Run do stavu Stop nebo po určitých chybách programování (viz Dodatek A). *Zrychlení jogu* používají pohybové programy, když v programu nebude povel zrychlení. Povel pohybu %AQ (27h) používá vždy *Zrychlení jogu*. Reset nebo vypnutí a zapnutí napájení PLC vrátí tuto hodnotu na nakonfigurovaná data.

Poznámka: V DSM314 se používá minimální hodnota po změně měřítka. Tato hodnota je určena pravidlem:

Zrychlení jogu * (uživatelské jednotky/jednotku) ≥ 32 jednotek/sec/sec.

4.12 Časová konstanta polohové smyčky. (0,1 milisekundy) Tento povel umožňuje provést změnu časové konstanty polohové smyčky z původní nakonfigurované hodnoty. Čím menší bude Časová konstanta polohové smyčky, tím rychlejší bude odezva systému. Hodnoty, které budou příliš nízké, způsobí nestabilitu a oscilace systému. Pro přesné sledování profilu zadané rychlosti ***Časová konstanta polohové smyčky*** musí být 1/4 až 1/2 MAXIMÁLNÍ doby zrychlení nebo zpomalení systému. Aby ***Časová konstanta polohové smyčky*** v analogovém režimu fungovala dobře, nakonfigurovaná hodnota "Vel at Max Cmd" musí být nastavená správně. Reset nebo vypnutí a zapnutí napájení PLC vrátí tuto hodnotu na nakonfigurovaná data.

4.13 Posuv rychlostí. Tento povel nastaví zisk ***Posuv rychlostí*** (0,01 procent). K výstupu rychlostního povelu DSM314 se přičítá procento *Zadané rychlosti*. Zvýšení ***Rychlosti posuvu*** způsobí, že servo bude pracovat s rychlejší odezvou a menší polohovou odchylkou. Optimální hodnoty ***Rychlosti posuvu*** jsou 90-100 %. Aby koeficient zisku ***Posuv rychlostí*** fungoval dobře, analogová serva musí mít správně nakonfigurovanou hodnotu “Vel at Max Cmd”. Reset nebo vypnutí a zapnutí napájení PLC vrátí tuto hodnotu na nakonfigurovaná data.

4.14 Časová konstanta integrátoru. (Milisekundy) Tento povel nastaví ***Časovou konstantu integrátoru*** pro polohovou odchylku integrátoru. Hodnota určuje čas, během kterého se odstraní 63% *Polohové odchylky*. Aby nedocházelo k nestabilitě a oscilacím, ***Časová konstanta integrátoru*** musí být 5 až 10 krát větší než ***Časová konstanta polohové smyčky***. **Doporučuje se, aby se polohová odchylka integrátoru používala pouze při souvislých aplikacích vlečené osy. Použití integrátoru v aplikacích polohování mezi dvěma body může mít při zastavení za následek překmitnutí polohy.**

4.15 Poměr vlečené osy A/B. Tento povel umožňuje, by PLC aktualizovalo slave**:** poměr master A/B používaný v každé smyčce vlečené osy. “A” je 16-bitové celé číslo se znaménkem s minimální hodnotou –32 768 a maximální hodnotou +32 767. “B” je 16-bitové celé číslo se znaménkem s minimální hodnotou -1 a maximální hodnotou +32 767. Velikost poměru A/B musí být v rozsahu 32:1 až 1:10 000, jinak se bude hlásit stavová chyba. Více informací o poměru A/B najdete v kapitole 8.

4.16 Zisk rychlostní smyčky. (VLGN) Pouze digitální režim a režim analogového kroutícího momentu. Zisk rychlostní smyčky pro digitální servoosy GE Fanuc a serva v režimu analogového kroutícího momentu je možno nastavit pomocí povelu *Zisk rychlostní smyčky*. Hodnota VLGN se používá pro přizpůsobení setrvačnosti břemena (J_L) se setrvačností motoru (J_M). VLGN je definováno s výchozí hodnotou 16, která představuje poměr setrvačnosti 1 k 1. Hodnota VLGN se vypočítává za předpokladu, že břemeno je pevně spojeno s motorem. Proto se u konkrétního nastavení stroje požadovaná hodnota může podstatně lišit od vypočítané hodnoty v důsledku tuhosti, tření mrtvého chodu a dalších faktorů. Reset nebo vypnutí a zapnutí napájení PLC vrátí VLGN na hodnotu nastavenou v konfiguračním softwaru. Doporučený počáteční bod pro *Zisk rychlostní smyčky* je:

$$\text{Zisk rychlostní smyčky} = \frac{\text{Setrvačnost břemena } (J_L)}{\text{Setrvačnost motoru } (J_M)} \times 16$$

Přípustný rozsah *Zisku rychlostní smyčky* je 0 až 255.

Například: Setrvačnost motoru (J_M) konkrétního serva je 0,10 lb-in-s^2. Setrvačnost břemena (J_L) u této aplikace je 0.05 lb-in-s^2. VLGN = (0.05 / 0.10) ∗ 16 = 8

Výchozí *Zisk rychlostní smyčky* je nastavený pomocí nastavení ***Zisku rychlostní smyčky*** v konfiguračním softwaru.

Upozornění

Nesprávná hodnota VLGN může mít za následek nestabilitu osy. Při změně hodnoty VLGN je nutno dávat pozor.

4.17 Mez krouticího momentu. (0,01 procent) Pouze digitální režim a režim analogového kroutícího momentu. Povel *Mez krouticího momentu* představuje metodu omezení krouticího momentu vytvářeného servomotorem GE Fanuc. V režimu analogového kroutícího momentu hodnota omezení krouticího momentu omezuje povel krouticího momentu na procento plné hodnoty. Zejména omezuje plný rozsah analogového výstupu, kde se plný rozsah rovná 10 V. DSM314 při každém zapnutí napájení nebo resetu nastaví *Mez krouticího momentu* na výchozí hodnotu 10 000 (100 %). Jakoukoliv jinou požadovanou hodnotu *Meze krouticího momentu* musí nastavit aplikační logika PLC. Platný rozsah *Meze krouticího momentu* je 0 až 10 000 v jednotkách 0.01%. To představuje 0 - 100 % špičkového kroutícího momentu při zadané rychlosti. Pokud bude poslána hodnota mimo rozsah 10 001 – 65 535, mez krouticího momentu se nastaví na 10000. *Mez krouticího momentu* je možno změnit během pohybu osy a bude platná okamžitě. Potřebujete-li určit aktuální hodnotu výstupního kroutícího momentu použitelnou při dané rychlosti, křivku kroutícího momentu motoru najdete v příslušném manuálu servomotoru. Jednoduchým příkladem může být použití *Meze krouticího momentu* k zabránění přetažení stroje.

4.18 Nastavení polohy pomocného snímače polohy. (Uživatelské jednotky) Tento povel nastaví hodnotu *Okamžité polohy* pro pomocný snímač polohy bez použití operace Nalezení výchozí polohy. Bit %I *Poloha platná* pro pomocnou osu se nastaví do jedničky, když přijde povel.

4.19 Vynucená rychlost serva. (ot./min) Pouze digitální režim a režim analogového kroutícího momentu. Tento povel vyřadí polohovou smyčku a digitálnímu servu vnutí rychlostní povel pro účely ladění. V režimu analogového kroutícího momentu vyřadí polohovou smyčku a regulátoru rychlosti vnutí rychlostní povel. Řízení zrychlení se nepoužívá a změny rychlosti se provedou okamžitě. Hodnota povelu *Vynucená rychlost serva* +4095 vygeneruje rychlost motoru + 4 095 ot./min a -4095 vygeneruje rychlost motoru –4 095 ot./min (v závislosti na jednotlivých maximálních rychlostech motoru). Řídicí smyčka digitálního serva může omezit skutečnou hodnotu rychlosti motoru na nižší hodnotu. Je nutno dát pozor na to, aby se servomotor neprovozoval mimo rozsah jmenovitého zatížení.

Bit %Q *Povolení pohonu* musí být aktivní a nesmí být zadaný žádný jiný pohyb, aby povel *Vynucená rychlost serva* byl funkční. Aby povel správně pracoval, musí zůstat trvale v datech %AQ. Když povel Vynucená rychlost serva bude aktivní pro danou osu, jakýkoliv jiný okamžitý povel %AQ pro tuto osu odstraní data *Vynucené rychlosti serva* a servo se zastaví. ***Kapitola 6, Neprogramované pohyby, také obsahuje informace o Vynucené rychlosti serva.***

4.20 Volba dat návratu 1. Tento povel umožňuje pro každou osu předat alternativní data na adrese %AI *Uživatelem volitelná data 1*. Alternativní data obsahují informace, jako například obsah paměti parametrů a revizi firmwaru DSM314.

Povel *Volba dat návratu 1* používá volbu režimu a volbu offsetu. Volba režimu (bajt offsetu +1 šesti-bajtového povelu) určuje typ dat návratu. Volba offsetu (bajty offsetu +2, +3 šesti-bajtového povelu) určují jednotlivé datové položky některých modelů. Nastavení režimu na 00h bude mít za následek, že se předá Povel kroutícího momentu. ***Výchozí režim a offset pro Uživatelem volitelná data 1 je možno nastavit v konfiguračním softwaru modulu.***

4.21 Volba dat návratu 2. Tento povel umožňuje pro každou osu předat alternativní data na adrese %AI *Uživatelem volitelná data 2*. Alternativní data obsahují informace, jako například obsah paměti parametrů a revizi firmwaru DSM314.

Povel *Volba dat návratu 2* používá volbu režimu a volbu offsetu. Volba režimu (bajt offsetu +1 šesti-bajtového povelu) určuje typ dat návratu. Volba offsetu (bajty offsetu +2, +3 šesti-bajtového povelu) určují jednotlivé datové položky některých modelů. Nastavení režimu na 00h bude mít za následek, že se předá Povel kroutícího momentu.**Výchozí režim a offset pro Uživatelem volitelná data 2 je možno nastavit v konfiguračním softwaru modulu.**

Pro Volbu dat návratu 1 a Volbu dat návratu 2 jsou přípustné následující volby:

Digitální	Analogový kroutící moment	Analogová rychlost	Volitelná data návratu	Režim dat:	Offset dat
A	A	N	Povel kroutícího momentu	00h	Nepoužívá se
A	A	A	Revize firmwaru DSM	10h	Nepoužívá se
A	A	A	Identifikační číslo firmwaru DSM (hex)	11h	Nepoužívá se
A	N	N	Absolutní zpětná vazba (cts)	17h	Nepoužívá se
A	A	A	Data parametrů	18h	Číslo parametru (0-255)
A	A	A	Bity CTL 1-32	19h	Nepoužívá se
A	A	A	Analogové vstupy - osa 1	1Ch	Nepoužívá se
A	A	A	Analogové vstupy - osa 2	1Dh	Nepoužívá se
A	A	A	Analogové vstupy - pomocná osa 3	1Eh	Nepoužívá se
A	A	A	Analogové vstupy - pomocná osa 4	1Fh	Nepoužívá se
A	A	A	Zadaná poloha (uživatelské jednotky)	20h	Nepoužívá se
A	A	A	Poloha vlečené osy zadaná povelem (cts)	21h	Nepoužívá se

Digitální	Analogový krouticí moment	Analogová rychlost	Volitelná data návratu	Režim dat:	Offset dat
A	A	A	Nepřizpůsobená okamžitá poloha (cts)	28h	Nepoužívá se
A	A	A	Nepřizpůsobená poloha vzorkování 1 (cts)	29h	Nepoužívá se
A	A	A	Nepřizpůsobená poloha vzorkování 2 (cts)	2Ah	Nepoužívá se

Povel kroutícího momentu má měřítko nastavené tak, že +/- 10000 = +/- 100% krouticího momentu.

Revize firmwaru DSM je uvedená jako dvě samostatná slova s kódy hlavní-vedlejší revize.

Identifikační číslo firmwaru DSM je uváděno jako jedno slovo v hexadecimálním tvaru.

Absolutní offset zpětné vazby je polohový offset (v jednotkách), který se používá k inicializaci *Okamžité polohy*, když se používá digitální absolutní snímač polohy GE Fanuc. *Okamžitá poloha* = data absolutního snímače polohy + offset absolutní zpětné vazby.

Analogové vstupy se skládají ze dvou datových slov pro každou osu: dolní slovo = AIN1 a horní slovo = AIN2. Data mají měřítko nastavené tak, že +/- 32000 = +/- 10,0 V.

Zadaná poloha (uživatelské jednotky) je kopie %AI dat *Zadané polohy* předávaných pro jednotlivé osy. Viz odstavec 2.04 v kapitole 5.

Poloha vlečené osy zadaná povelem (cts) je aktualizovaná aktivní zadaná poloha (v jednotkách zpětné vazby) a používaná interním generátorem pohybového povelu. Viz kapitola 9 – Kombinovaný pohyb vlečené osy a zadaný pohyb.

Nepřizpůsobená okamžitá poloha je kumulovaná okamžitá poloha (v jednotkách, ne v uživatelských jednotkách) s 32-bitovou binární hodnotou překlopení –2 147 483 648 … +2 147 483 647.

Nepřizpůsobená poloha vzorkování 1 je hodnota *nepřizpůsobené okamžité polohy* zachycené, když se vyskytne vstup Vzorkování 1.

Nepřizpůsobená poloha vzorkování 2 je hodnota *nepřizpůsobené okamžité polohy* zachycené, když se vyskytne vstup Vzorkování 2.

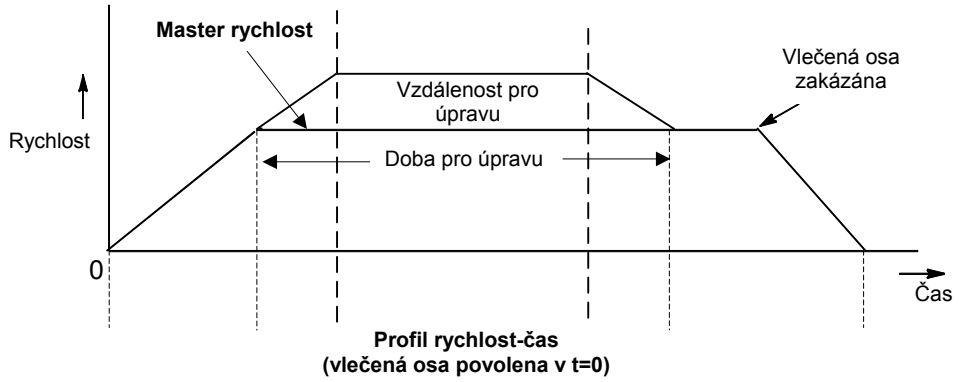
Poznámka: Než v PLC budou k dispozici nová volitelná data návratu, musí uplynout minimálně tři cykly PLC nebo 10 milisekund (podle toho co je delší).

4.22 Doba úpravy vzdálenosti náběhu vlečené osy. Když bude zvolena funkce náběhu vlečené osy a vlečená osa bude povolená, vlečená osa vykoná náběh na master rychlost nakonfigurovaným ***Zrychlením náběhu vlečené osy***, když master rychlost bude v době povolení vlečené osy nenulová. Master pulsy, které se kumulují během zrychlování vlečená osy, se uloží. V tomto režimu se osa zrychlí na rychlost, která přesahuje master rychlost, aby se srovnala polohová odchylka, která narostla, když vlečená osa zrychlovala na master rychlost. Tato korekce vzdálenosti náběhu má lichoběžníkový profil rychlosti určený ***Dobou úpravy vzdálenosti náběhu vlečené osy*** a ***Zrychlením pozvolného náběhu*** na začátku korekce. Tento režim se používá, když vlečená osa musí synchronizovat polohu a rychlost s master polohou v okamžiku, kdy byl povolený režim vlečené osy.

Pokud ***Doba úpravy vzdálenosti náběhu vlečené osy*** bude příliš krátká, pak rychlost bude mít trojúhelníkový profil. Pokud rychlost během korekce rychlosti přesáhne 80% meze rychlosti, pak automaticky vypočítaná rychlost se omezí na 80% nakonfigurované meze rychlosti. V obou případech se bude hlásit výstražné hlášení a skutečná doba adjustace vzdálenosti bude delší než naprogramovaná doba, ale vzdálenost se zkoriguje správně.

Nastavení ***Doby úpravy vzdálenosti náběhu vlečené osy*** na 0 umožní, aby funkce náběhu provedla zrychlování osy bez adjustování nakumulovaných pulsů. V tomto případě rychlost vlečené osy nepřesáhne master rychlost. U aplikací, kde je nutno vlečenou osu pouze synchronizovat s master rychlostí a ztracené pulsy nejsou na závadu, nastavte dobu úpravy vzdálenosti = 0.

Typický profil rychlosti během cyklu náběhu vlečené osy je znázorněný níže.



Profil rychlost-čas (vlečená osa povolena v t=0)

Mnohem více podrobností o této funkci najdete v kapitole 8, v části "Pohyb vlečené osy, řízení náběhu zrychlování vlečené osy".

4.23 Proporcionální zisk rychlostní smyčky. Pouze režim analogového kroutícího momentu. Povel AQ proporcionálního zisku rychlostní smyčky umožňuje uživateli nastavit proporcionální zisk regulátoru rychlosti v režimu analogového kroutícího momentu. Proporcionální zisk se násobí odchylkou rychlosti (povel rychlosti - rychlostní zpětná vazba) a vygeneruje se část povelu kroutícího momentu odpovídající proporcionální hodnotě. Správné nastavení této hodnoty určuje, jak dobře regulátor rychlosti bude provádět řízení systému. Dodatek D popisuje způsob, jak tento parametr správně naladit. Přípustný rozsah proporcionálního zisku rychlostní smyčky je 0-32767. Výchozí hodnota je 1500.

4.24 Integrální zisk rychlostní smyčky. Pouze režim analogového kroutícího momentu. Povel AQ integrálního zisku rychlostní smyčky umožňuje uživateli nastavit integrální zisk rychlostní smyčky v režimu analogového kroutícího momentu. Integrální zisk je hodnota násobená plochou rychlostní odchylky (povel rychlosti - rychlostní zpětná vazba), která vygeneruje část povelu kroutícího momentu odpovídající integrální hodnotě. Správné nastavení této hodnoty určuje, jak dobře regulátor rychlosti bude provádět řízení systému. Dodatek D popisuje způsob, jak tento parametr správně naladit. Přípustný rozsah integrálního zisku rychlostní smyčky je 0-32767. Výchozí hodnota je 0.

4.25 Filtr povelu kroutícího momentu. Pouze režim analogového kroutícího momentu. Povel AQ filtru povelu kroutícího momentu umožňuje uživateli aktivovat filtr s dolní propustí pro výstup regulátoru rychlosti (Povel kroutícího momentu). Filtr se typicky používá k tomu, aby kontrolér nevybudil resonance stroje. Přípustné nastavení filtru povelu kroutícího momentu je uvedeno v tabulce 5-9.

Tabulka 5-8. Povely filtru kroutícího momentu

Režim TqFilt	Nastavení filtru dolní propusti pro povel kroutícího momentu
0	VYPNUTO
1	Filtr dolní propusti (bod 150 Hz 3dB)
2	Filtr středního pásma (bod 250 Hz 3dB)
3	Filtr horní propusti (bod 350 Hz 3dB)

Poznámka 1: Výchozí nastavení

4.26 Volba analogového výstupního režimu. Pouze digitální režim. U digitálních serv GE Fanuc tento povel umožňuje zvolit, jaké analogové signály se mají posílat na analogové výstupní piny (piny 6 a 24) na čtyřech konektorech čelní desky DSM. Povel *Volba analogového výstupního režimu* používá kód signálu ke specifikaci signálu, který se má poslat, a kód konektoru ke specifikaci konektoru DSM, na kterém se signál má přijímat. Tento povel je částečně užitečný při ladění serva. Tento povel je možno poslat z registrů povelu na libovolnou osu (1-4).

K nastavení 6-bajtového okamžitého povelu %AQ (popsaný v tabulce 5-7) použijte následující strukturu:

- Bajt 0 obsahuje kód povelu Volba analogového výstupního režimu (47h).
- Bajt 1 obsahuje kód konektoru, hexadecimální číslo.
- Bajty 2-3 obsahují kód signálu, dekadické číslo.
- Bajty 4-5 se nepoužívají a musí obsahovat 0.

Kódy konektoru

Kód konektoru	Zvolený konektor	Piny konektoru
01h	Konektor A	Pin 6 = OUT Pin 24 = COM (Vztaženo k 0 V) Připojení svorkovnice najdete v odstavci Schéma připojení I/O v kapitole 3.
02h	Konektor B	
03h	Konektor C	
04h	Konektor D	

Kódy signálu

V následující tabulce kódů signálů si všimněte, že pouze některé signály mají výchozí výstup.

Kód signálů	Popis signálu	Výchozí výstup na:
00 dekadicky*	Data %AQ *Vynucený analogový výstup**	Konektor C nebo D
10 dekadicky	Povel kroutícího momentu pro servoosu 1	Žádný
15 dekadicky	Okamžitá rychlost servoosy 1	Konektor A
20 dekadicky	Povel kroutícího momentu pro servoosu 2	Žádný
25 dekadicky	Okamžitá rychlost servoosy 2	Konektor B

* Nelze vést jinam. Tento kód signálu je možno použít pouze pro návrat tohoto signálu zpět na výchozí výstup.

Poznámka: Analogový výstup nelze použít pro uživatelské řízení u digitálně řízených os. Vydání povelu Vynucený analogový výstup nebo Volba analogového výstupu pro digitální osy nebude mít na tyto analogové výstupy žádný vliv.

Volba režimu analogového výstupu má tři základní použití:

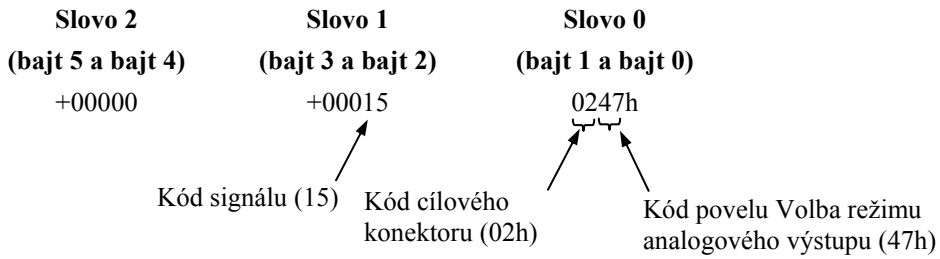
(1) Převedení buď *Okamžité rychlosti servoosy 1* nebo *Okamžité rychlosti servoosy 2* z výchozího výstupu na jiný výstup. Data signálu %AQ *Vynucený analogový výstup* nelze vést na jiný konektor; je však možno ho nahradit na výchozím výstupním konektoru (C nebo D) jiným signálem, který je sem vedený povelem *Volba režimu analogového výstupu*.

(2) Vedení jednoho ze dvou signálů, u kterých není výchozí výstup, *Povel kroutícího momentu pro servoosu 1* a *Povel kroutícího momentu pro servoosu 2,* na jeden z výstupů jako náhrada předchozího signálu na tomto výstupu. To je ukázáno v příkladu 2 níže.

(3) Obnovení signálů s výchozími výstupy, které byly nahrazené převedenými signály. V příkladu 2 je signál *%AQ Vynucený analogový výstup*, který se normálně nachází jako výchozí na konektoru D, nahrazený signálem *Povel kroutícího momentu pro servoosu 1*, který byl vedený na konektor D povelem *Volba režimu analogového výstupu*. V příkladu 3 je signál *%AQ Vynucený analogový výstup* obnovený na konektor D použitím povelu *Volba režimu analogového výstupu*.

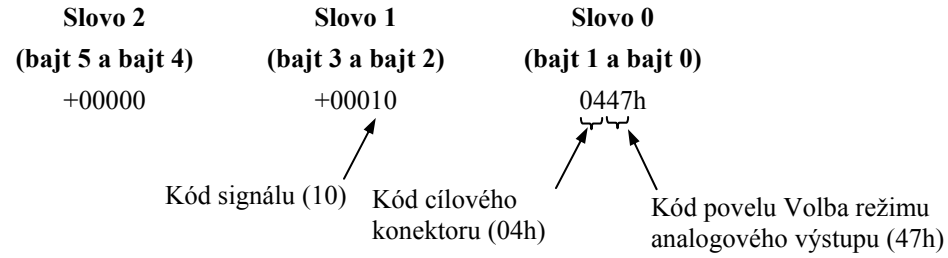
Příklad 1:

V tomto příkladu signál *Okamžitá rychlost servoosy 1* (kód signálu = 15) se převede z výchozího výstupu na konektoru A na konektor B (kód konektoru = 02h) a nahradí předchozí signál na konektoru B. To se provede umístěním následujících dat do slov okamžitého povelu %AQ:



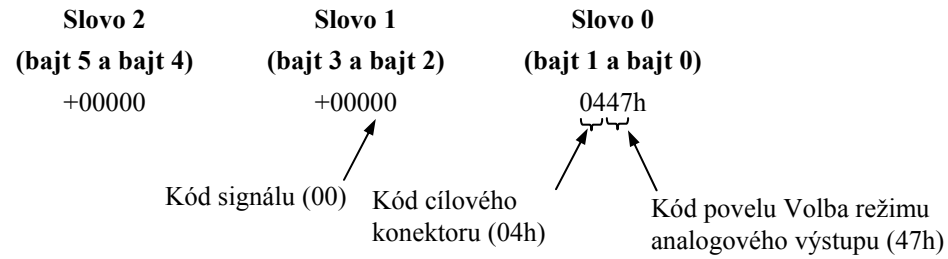
Příklad 2:

V tomto příkladu se signál *Povel kroutícího momentu pro servoosu 1* (kód signálu = 10) zvolí jako analogový výstup na konektoru D (kód konektoru = 04h) a nahradí předchozí signál na konektoru D. K tomu je nutno do slov okamžitého povelu %AQ umístit následující data:



Příklad 3:

V příkladu 2 se výchozí signál %AQ *Vynucený analogový výstup* nahradil jako analogový výstup na konektoru D signálem *Povel kroutícího momentu pro servoosu 1*. K obnovení signálu %AQ *Vynucený analogový výstup* (kód signálu = 00) na konektoru D (kód konektoru = 04h) do slov okamžitého povelu %AQ umístěte následující data:



4.27 Vynulovat novou přijatou konfiguraci. Tento povel vynuluje bit %I *Přijatá nová konfigurace*. Po vynulování se bit *Konfigurace dokončena* nastaví do jedničky, pouze když se provede reset PLC nebo se provede nová konfigurace modulu. PLC může monitorovat bit a určit, jestli musí poslat znovu další povely %AQ, například *Zóna dosažení polohy* nebo *Zrychlení jogu*. To by bylo nutné, pouze pokud by se povely %AQ používaly k přepisu konfiguračních dat DSM314 naprogramovaných pomocí konfiguračního softwaru PLC. Tento povel je možno poslat z registrů povelu na libovolnou osu (1-4).

4.28 Okamžité načtení parametru. Tento povel se vykoná z PLC a okamžitě změní hodnoty parametrů DSM314. Je možno ho poslat z registrů povelu na libovolnou osu (1-4). Parametry dat používají pouze pohybové programy. Pro každou změnu parametru je zapotřebí povel. Bajt 1 slova 0 obsahuje číslo parametru (v hexadecimálním tvaru), který se má změnit. DSM314 obsahuje 256 parametrů s dvojitou délkou číslované 0-255 (dekadicky). Podrobnosti viz kapitola 7, část "Parametry (P0-P255) v DSM314".

Tabulka 5-9. Počet povelů *Okamžité načtení parametru* přípustných na cyklus

Počet nakonfigurovaných os	Počet slov %AQ	Počet povelů *Okamžité načtení parametru* přípustných na cyklus
2	6	2
3	9	3
4	12	4

Kapitola 6

Neprogramovaný pohyb

DSM314 může generovat pohyb v ose jedním nebo několika způsoby, aniž by se použil pohybový program.

- *Nalezení výchozí polohy* a *Jog Plus/Minus* k zadání pohybu.
- *Pohyb rychlostí*, *Přesunutí*, *Vynucená rychlost serva*, *Vynucený analogový výstup* a *Inkrement polohy*.

Během povelů *Jog*, *Nalezení výchozí polohy, Pohyb rychlostí, Přesunutí* a *Vynucená rychlost serva* bude mít jiný zadaný pohyb, programovaný nebo neprogramovaný, za následek chybové hlášení. Jedinou výjimkou je povel %AQ *Inkrement polohy*, který je možno zadat kdykoliv. Více podrobností najdete v popisu k pohybu *Inkrement polohy* uvedený níže.

Neprogramované pohyby (*Zrušit všechny pohyby*, *Jog Plus/Minus*, *Pohyb rychlostí, AQ povel Přesunutí* a Normální zastavení) používají ***Zrychlení jogu*** a ***Režim zrychlení jogu***. Povel %Q *Zastavení posuvu* používá režim zrychlení a zpomalení.

DSM314 cyklus nájezdu do výchozí polohy

Cyklus nájezdu do výchozí polohy je možno použít k nastavení správné Okamžité polohy vzhledem ke strojnímu referenčnímu bodu. Nakonfigurovaný **Offset výchozí polohy** definuje polohu *Výchozí polohy* jako offsetovou vzdálenost od značky výchozí polohy.

Bit %Q *Povolení pohonu* musí být v jedničce po celou dobu cyklu nájezdu do výchozí polohy. Bit %Q *Nalezení výchozí polohy* však nemusí být během cyklu v jedničce; může se nastavit jednorázově na okamžik do jedničky. Všimněte si, že přechod bitu %Q *Nalezení výchozí polohy* do jedničky okamžitě nastaví do nuly bit %I *Poloha platná* až do konce cyklu nájezdu do výchozí polohy. Bit %Q *Zrušit všechny pohyby* zastaví cyklus nájezdu do výchozí polohy, ale bit *Poloha platná* se do jedničky nevrátí. Pokud bit *Poloha platná* nebude v jedničce, nelze vykonat žádný pohybový program.

Režim spínače výchozí polohy

Pokud ***Režim nalezení výchozí polohy*** bude nakonfigurovaný jako HOMESW (spínač HOME), vstup Spínač výchozí polohy z I/O konektoru osy se použije nejdříve k hrubé indikaci referenční výchozí polohy. Pak další značka snímače polohy zjištěná při pohybu v záporném směru indikuje přesnou polohu. Rozepnutý vstup Spínače výchozí polohy indikuje, že servo je na kladné straně spínače výchozí polohy, a sepnutý spínač indikuje, že osa je na záporné straně spínače výchozí

polohy. Přechod povelu %Q *Nalezení výchozí polohy* z nuly do jedničky vykoná následující cyklus nájezdu do výchozí polohy. Pokud nebude uvedeno jinak, zrychlení bude na aktuálním ***Zrychlení jogu*** a nakonfigurovaném ***Režimu zrychlení jogu***.

Program Nalezení výchozí polohy pro Spínač výchozí polohy

Pokud program bude spuštěný z polohy na kladné straně spínače výchozí polohy, kdy spínač výchozí polohy musí být ROZPOJENÝ (logická 0), program Nalezení výchozí polohy začne krokem 1 níže. (První kroky následujícího programu jsou nutné k tomu, aby se počítalo s různým možným provedením spínače výchozí polohy a počátečními polohami.) Pokud program nalezení výchozí polohy bude spuštěný z polohy na záporné straně spínače výchozí polohy, kdy spínač výchozí polohy musí být SEPNUTÝ (logická 1), program začne krokem 3 níže.

1. Osa se bude pohybovat v záporném směru nakonfigurovanou ***Rychlostí pro nalezení výchozí polohy***, dokud nedojde k sepnutí Spínače výchozí polohy.
2. Osa provede zpomalení a zastaví se.
3. Osa provede zrychlování v kladném směru a bude se pohybovat nakonfigurovanou ***Rychlostí pro nalezení výchozí polohy***, dokud nedojde k rozpojení vstupu Spínače výchozí polohy.
4. Osa provede zpomalení a zastaví se.
5. Osa provede zrychlování v záporném směru a bude se pohybovat nakonfigurovanou ***Konečnou rychlostí výchozí polohy***, dokud nedojde k sepnutí vstupu Spínače výchozí polohy.
6. Osa se bude pohybovat v záporném směru nakonfigurovanou ***Konečná rychlostí výchozí polohy***, dokud se nezjistí puls značky. Značka nastaví výchozí referenční polohu.
7. Osa provede zpomalení a zastaví se (v poloze za pulsem značky).
8. Osa se posune aktuální ***Rychlostí jogu*** o počet uživatelských jednotek určených hodnotou ***Offset výchozí polohy*** od výchozí referenční polohy. Pokud ***Offset výchozí polohy*** = 0, osa se posune zpět do polohy pulsu značky.
9. Osa provede zpomalení a zastaví se.
10. DSM314 nastaví stavová slova %AI *Zadaná poloha* a *Aktuální poloha* na nakonfigurovanou hodnotu ***Výchozí poloha***. Nakonec DSM314 nastaví bit %I *Poloha platná* jako indikaci toho, že se dokončil cyklus nájezdu do výchozí polohy.

Příklad spínače výchozí polohy

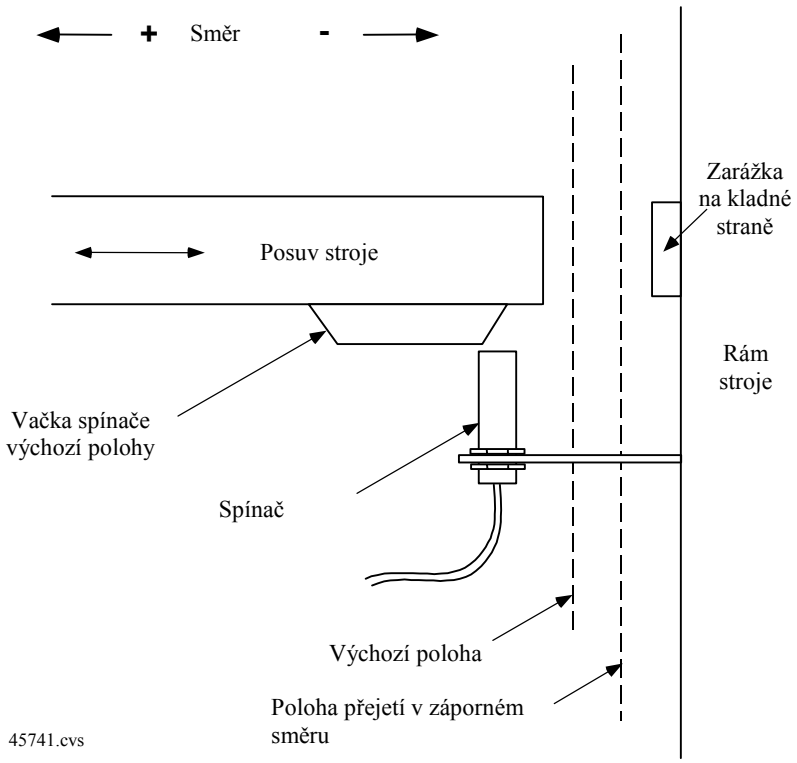
Existuje mnoho různých provedení spínačů výchozí polohy. Spínač může být spínací nebo rozpínací kontakt a může být namontovaný na několika možných místech. Příklad uvedený v tomto odstavci uvádí celkem běžné uspořádání používané pro lineární osy. V následujícím obrázku je spínač výchozí polohy spínací přibližovací kontakt namontovaný blízko konce rozsahu posuvu stroje (v záporném směru). Imaginární přímka, která rozděluje kladnou a zápornou stranu spínače výchozí polohy, je pracovní bod spínače umístěný přibližně v ose spínače. Pokud se posuv stroje posune v záporném směru dostatečně daleko tak, aby pravá hrana vačky spínače výchozí polohy způsobila sepnutí spínače, budeme předpokládat, že posuv stroje překročil "zápornou stranu" spínače výchozí polohy. ***Vačka spínače výchozí polohy je dostatečně dlouhá, aby když posuv stroje bude na záporné straně spínače, spínací kontakt výchozí polohy zůstal sepnutý.***

Všimněte si vztahů mezi výchozí polohou, polohou přejetí v záporném směru a polohou zastavení v kladném směru. Mezi výchozí polohou a polohou přejetí v záporném směru je malá vzdálenost v záporném směru. Je to z důvodu, aby zůstal nějaký "pracovní prostor" pro seřízení a nastavení těchto poloh a pro program "nalezení výchozího bodu", který vyžaduje, aby se poslední pohyb vykonal v záporném směru.

Vzdálenost také existuje mezi polohou meze přejetí a zarážkou v kladném směru. Zde musí zůstat dostatečná vzdálenost, aby posuv stroje nenarazil do narážky v kladném směru. Správná vzdálenost musí být větší než nejhorší vzdálenost zastavení, kterou posuv stroje potřebuje k dosažení polohy meze přejetí.

V tomto příkladu pracovní rozsah posuvu stroje je na kladné straně spínače výchozí polohy. Pokud parametr DSM *Výchozí poloha* byl nastavený na 0, zjednoduší se tím povely programování absolutní polohy, protože se používají pouze kladná čísla.

Často je nutné výchozí polohu nastavit na přesnou vzdálenost od referenčního bodu stroje. Aby se toto seřízení usnadnilo, vačku spínače výchozí polohy je možno vyrobit s drážkovými montážními dírami, které umožní hrubé nastavení vačky tak, aby se kalibrace dostala do rozmezí jedné otáčky snímače polohy. Zbývající malá vzdálenost se pak přesně změří a získaná hodnota se zapíše do parametru DSM *Offset výchozí polohy*.



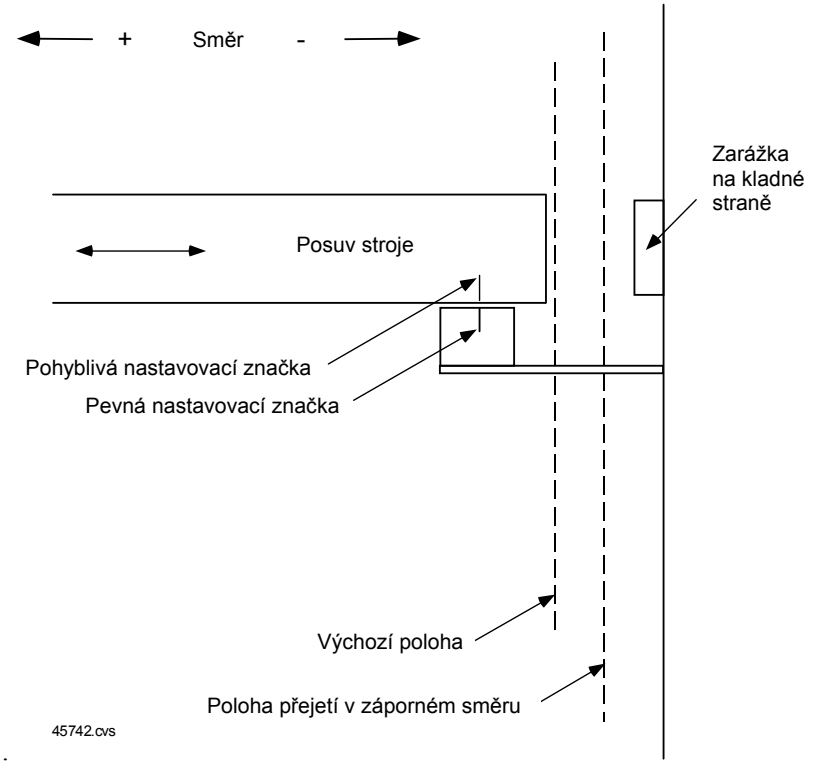
Obrázek 6-1. Příklad spínače výchozí polohy

Režimy Move+ a Move–

Pokud ***Režim nalezení výchozí polohy*** bude nakonfigurovaný na MOVE+ nebo MOVE–, první zjištěný puls značky snímače polohy při pohybu v příslušném směru (kladném pro MOVE+, záporném pro MOVE–) po povelu nalezení výchozí polohy se použije k nastavení přesné polohy. V tomto režimu obsluha obvykle provádí jogování osy do polohy blízko (v rozmezí jedné otáčky snímače polohy) první výchozí polohy, pak vyvolá povel nalezení výchozí polohy. Aby obsluha měla jogování do správné polohy jednodušší, na stroji a na strojní ose jsou někdy umístěné seřizovací značky, které indikují blízkost výchozí polohy.

Příklad cyklu nájezdu do výchozí polohy Move – (mínus)

Na následujícím obrázku je ukázaný příklad parametru Výchozí poloha nastaveného na Move – (mínus). V tomto příkladu obsluha provádí jogování osy, dokud pohyblivá značka na posuvu stroje nebude v zákrytu s pevnou značkou na seřizovací desce namontované na rámu stroje. (Všimněte si, že značky jsou v zákrytu na kladné straně výchozí polohy, protože parametr Výchozí poloha je nastavený na Move -). Pak obsluha vyvolá program nalezení výchozího bodu, který způsobí, že se osa bude posunovat v záporném směru, dokud se neobjeví puls značky.



Obrázek 6-2. Příklad nájezdu do výchozí polohy Move - (mínus)

Program nalezení výchozí polohy pro Move + nebo Move -

Když bude zadaný povel nalezení výchozí polohy (přechod bitu %I *Nalezení výchozí polohy* z nuly do jedničky), vykoná se následující sekvence dějů:

1. Osa provede zrychlení ***Zrychlením jogu*** a bude se pohybovat naprogramovanou ***Konečnou rychlosti výchozí polohy*** (kladný směr pro MOVE+, záporný směr pro MOVE–), dokud se nezjistí puls značky. Tento puls značky nastaví výchozí referenční polohu.
2. Osa se zastaví (v poloze za pulsem značky) s použitím nakonfigurovaného ***Zrychlení jogu*** a s nakonfigurovaným ***Režimem zrychlení jogu***.
3. Osa se posune nakonfigurovanou ***Rychlostí jogu*** a s nakonfigurovaným ***Zrychlením jogu*** a ***Režimem zrychlení jogu*** o počet uživatelských jednotek určených hodnotou ***Offset výchozí polohy*** od výchozí referenční polohy. Pokud ***Offset výchozí polohy*** = 0, osa se posune zpět do polohy pulsu značky.
4. Osa se zastaví nakonfigurovaným ***Zrychlením jogu*** a s nakonfigurovaným ***Režimem zrychlení jogu***.
5. DSM314 nastaví stavové slovo %AI *Zadaná poloha* a *Aktuální poloha* na nakonfigurovanou hodnotu ***Výchozí poloha***; DSM314 nastaví bit %I *Poloha platná* jako indikaci toho, že cyklus nájezdu do výchozí polohy se dokončil.

Jogování s DSM314

Rychlost jogu, ***Zrychlení jogu*** a ***Režim zrychlení jogu*** jsou konfigurační parametry DSM314. Tyto hodnoty se používají vždy, když bit %Q *Jog Plus* nebo *Jog Minus* přejde do stavu ON. Všimněte si, že pokud oba bity Jog budou současně v jedničce, nebude se generovat žádný pohyb. ***Zrychlení jogu*** a ***Režim zrychlení jogu*** se také používají během *Nalezení výchozí polohy*, *Pohybu rychlostí, Zrušení všech pohybů* a Normálního zastavení. Naprogramované pohyby používají ***Rychlost jogu*** a ***Zrychlení jogu*** jako výchozí.

Povel %Q *Jog Plus/Minus* je možno vykonat, když není zadaný žádný jiný povel nebo když bude naprogramovaný pohyb přechodně zastavený v důsledku povelu %Q *Zastavení posuvu*. K provedení jogu bit %Q *Povolení pohonu* nemusí být v jedničce, ale může být v jedničce. Přechodem bitu %Q *Jog Plus/Minus* do jedničky automaticky sepne Povolení relé a bit %I *Pohon povelen* se nastaví do jedničky. Když spínač přejetí meze bude rozepnutý, bity %Q *Jog Plus/Minus* a *Vynulování chyby* mohou současně přejít do jedničky, aby se provedl odjezd z rozepnutého koncového spínače. Proto povel %Q *Jog Plus* nebude fungovat, když bude rozepnutý spínač pojezdu na kladné straně a *Jog Minus* nebude fungovat, když bude rozepnutý spínač pojezdu na záporné straně. Nastavení bitu %Q Jog do nuly bude mít za následek zpomalení a zastavení. Pokud bit %Q *Jog* bude momentálně v nule, i třeba na jeden cyklus PLC, osa se zpomalí až do zastavení, pak zrychlí a bude pokračovat jogem.

Povel Pohyb rychlostí

Povel %AQ *Pohyb rychlostí* se vygeneruje vložením hodnoty 22h do prvního slova dat %AQ přiřazeného ose. Druhé a třetí slovo představují 32-bitovou rychlost se znaménkem. Všimněte si, že třetí slovo je nejvýznamnější slovo rychlosti. Jakmile povel bude zadaný, data %AQ je možno vynulovat posláním povelu *NULL* nebo je podle potřeby změnit. *Pohyb rychlostí* nebude fungovat, dokud servopohon nebude povolený (bit %Q *Povolit pohon* a stavový bit *Pohon povolen* jsou v jedničce).

Výpis okamžitých povelů %AQ ukazuje slova pro snazší pochopení v obráceném pořadí. Má-li se například zadat rychlost 512 uživatelských jednotek za sekundu v DSM314 nakonfigurovaném s daty %AQ začínajícími na %AQ1, použijí se následující hodnoty: 0022h (34 dekadicky) v %AQ1, 0200h (512 dekadicky) v %AQ2 a 0 v %AQ3. Když na DSM314 přijdou tyto hodnoty a %I *Pohon povolen* bude v jedničce, %Q *Zrušit všechny pohyby* bude v nule a nebude zadaný žádný jiný povel, spustí se pohyb osy rychlostí 512 uživatelských jednotek za sekundu v kladném směru s použitím aktuálního ***Zrychlení jogu*** a ***Režimu zrychlení***.

Bit %I *Pohon povolen* musí být v jedničce před tím, než na DSM314 přijde okamžitý povel, jinak se bude hlásit chyba. Také pokud povel *Pohyb rychlostí* již bude v datech %AQ, hodnota rychlosti se musí změnit, když bit *Pohon povolen* bude v jedničce, aby ho DSM314 mohlo přijmout. DSM314 zjistí povel *Pohyb rychlostí*, když se změní hodnoty %AQ.

Když DSM314 bude vykonávat povel *Pohyb rychlostí*, bude ignorovat softwarové meze přejetí (***Pos EOT*** a ***Neg EOT***). Pokud jsou hardwarové meze přejetí povolené, musí být v jedničce.

Povel *Pohyb rychlostí* je možno zastavit bez vyvolání chyby dvěma způsoby: povelem *Pohyb rychlostí* s nulovou rychlostí nebo nastavením bitu %Q *Zrušit všechny pohyby* do jedničky na dobu alespoň jednoho cyklu PLC.

Povel Vynucená rychlost serva (DIGITÁLNÍ serva; Režim analogového kroutícího momentu)

Tento povel vyřadí polohovou smyčku a vnutí rychlostní povel ot./min *digitálnímu* servu nebo analogovému rozhraní kroutícího momentu pro účely ladění. Řízení zrychlení se nepoužívá a změny rychlosti se provedou okamžitě. Hodnota povelu *Vynucená rychlost serva* +4095 vygeneruje rychlost motoru + 4,095 ot./min a -4095 vygeneruje rychlost motoru -4,095 ot./min (v závislosti na jednotlivých maximálních rychlostech motoru). Řídicí smyčka digitálního serva může omezit skutečnou hodnotu rychlosti motoru na nižší hodnotu.

UPOZORNĚNÍ

Je nutno dát pozor na to, aby se servomotor neprovozoval mimo rozsah jmenovitého zatížení.

Bit %Q *Povolení pohonu* musí být aktivní a nesmí být zadaný žádný jiný pohyb, aby povel *Vynucená rychlost serva* byl funkční. Aby povel správně pracoval, musí zůstat trvale v datech %AQ. Když povel Vynucená rychlost serva bude aktivní pro danou osu, jakýkoliv jiný okamžitý povel %AQ pro tuto osu odstraní data *Vynucené rychlosti serva* a servo se zastaví. Jednorázový povel *Vynucená rychlost serva* proto bude fungovat pouze během cyklu, ve kterém se objeví.

Více informací o tomto povelu najdete v kapitole 5, Rozhraní mezi Motion Mate DSM314 a PLC.

Poznámka: Povel Vynucený analogový výstup popsaný níže se používá pro analogová serva s rozhraním rychlostního povelu.

Povel Vynucený analogový výstup (ANALOGOVÁ serva s rychlostním rozhraním)

V režimu rozhraní analogové rychlosti okamžitý povel %AQ *Vynucený analogový výstup* ovládá analogový výstup na konektorech A, B, C nebo D čelní desky DSM. Hodnota *Vynucený analogový výstup* +32000 vytvoří +10,00 V ss a hodnota *Vynucený analogový výstup* -32000 vytvoří -10,00 V ss.

Vynucený analogový výstup funguje, pouze když data %AQ budou aktivní. Když povel Vynucený analogový výstup bude aktivní pro danou osu, povel Vynucený analogový výstup je možno zrušit a vypnout související analogové napětí některým jiným okamžitým povelem %AQ pro tuto osu.

Více informací o tomto povelu najdete v kapitole 5, Rozhraní mezi Motion Mate DSM314 a PLC.

Povely polohového inkrementu

Mají-li se vygenerovat malé opravy mezi polohou osy a sledováním DSM314, povely %AQ *Polohový inkrement* je možno použít k posunutí *Okamžité polohy* o daný počet uživatelských jednotek. Pokud bit %I *Pohon povolen* bude v jedničce, osa okamžitě vykoná pohyb o velikost inkrementu. Pokud se použije polohový inkrement bez aktualizace polohy (povel %AQ 21h), stavové slovo %AI *Okamžitá poloha* předávané z DSM314 zůstane nezměněné. Pokud se použije

Polohový inkrement s aktualizací polohy (povel %AQ 25h), stavová slova %AI *Okamžitá poloha* a *Zadaná poloha* předávané z DSM314 se změní o hodnotu inkrementu. *Polohový inkrement* je možno použít kdykoliv, i když současné použití s povelem *Vynucená rychlost serva* je zakázané, protože povel *Vynucená rychlost serva* musí zůstat v datové oblasti povelu %AQ, jinak dojde k zastavení serva.

Další poznámky

Další poznámky při používání neprogramovaných pohybů jsou následující:

- Když bit %Q *Zrušit všechny pohyby* bude v jedničce, nespustí se žádný neprogramovaný pohyb.
- Přechod bitu %Q *Zrušit všechny pohyby* do jedničky okamžitě zastaví všechny aktuální neprogramované pohyby aktuálním ***Zrychlením jogu***.
- Povel %AQ *Nastavení polohy* během neprogramovaného pohybu způsobí chybový stav.
- Přechod bitu %Q *Povolit pohon* do nuly během vykonávání cyklu nájezdu do výchozí polohy nebo vykonávání povelu %AQ *Pohyb rychlostí* způsobí chybu zastavení.
- Bit %Q *Zastavení posuvu* nemá žádný vliv na neprogramovaný pohyb.
- Povel %AQ *Přepis rychlosti* nemá žádný vliv na neprogramovaný pohyb.
- Změna ***Rychlosti jogu*** nebo ***Zrychlení jogu*** nemá žádný vliv na probíhající pohyby.

Kapitola 7 Programovaný pohyb

Programovaný pohyb se skládá ze skupiny uživatelem naprogramovaných pohybových povelů, které jsou uložené a vykonávají se v DSM314. Jakmile je program vyvolaný k vykonání, DSM314 vykonává povely pohybového programu sekvenčně blok po bloku. Pohybový program se vykonává nezávisle na PLC, i když PLC spustí pohybový program DSM314 a během vykonávání může působit jako rozhraní s ním (s parametry a určitými povely). Kromě toho externí vstupy (bity CTL) připojené přímo k čelní desce DSM314 nebo řízené lokální logikou je možno použít v pohybových programech k pozdržení nebo ke změnění průběhu vykonávání programu. PLC dostává stavové informace (například polohu, rychlost a číslo povelového bloku) z DSM314 během vykonávání programu. Pohybové programy 1-10 a podprogramy 1-40 se vytvářejí pomocí programovacího softwaru PLC a ukládají se společně s konfiguračním nastavením modulu do DSM314 přes propojovací rovinu PLC.

Další informace najdete v online nápovědě k vašemu softwaru nebo v odpovídajícím uživatelském manuálu softwaru:

Uživatelský manuál softwaru VersaPro™, GFK-1670

Krátký úvod pro *CIMPLICITY® Machine Edition Logic Developer-PLC*, GFK-1918

Pohybové programy a podprogramy pro jednu osu

Program pro jednu osu obsahuje programové příkazy pouze pro jednu osu. Naprogramovaná osa je uvedená na prvním řádku programu, například: PROGRAM 1 AXIS1. DSM314 může obsluhovat až čtyři programy pro jednu osu. Tyto programy mohou běžet každý zvlášť nebo společně. Například pro osu 1 je možno napsat pohybový program 1 a pro osu 2 je možno napsat pohybový program 2. Pro každou osu je možno provést nájezd do referenčního bodu a pohybový program vykonat nezávisle bez ohledu na stav druhé osy. Nebo Program 1 a Program 2 se mohou spustit současně (přes bity spuštění programu %Q) během stejného cyklu PLC.

Pohybové programy DSM314 podporují funkci podprogramu, které mohou obsahovat všechny použitelné povely pohybových programů včetně povelu *CALL*. Povel *SYNC Block* je vyhrazený pro povely a podprogramy pro více os (osa 1 a 2). “Vnořování” podprogramů pomocí příkazů *CALL* je podporováno až do 8 úrovní. Podprogramy pro jednu osu, podobně jako pohybové programy, obsahují povely pouze pro jednu osu. Rozdíl je v tom, že v podprogramu pro jednu osu není zadáno číslo osy. Pohybový program pro jednu osu může *vyvolat příkazem CALL* libovolný podprogram pro jednu osu uložený v paměti modulu. Například pohybový program 1 pro jednu osu obsluhující osu 1, může obsahovat příkaz CALL pro podprogram 1 pro jednu osu. Dále pohybový program 2 pro jednu osu obsluhující osu 2, může obsahovat příkaz CALL pro podprogram 1 pro jednu osu. Pohybové programy pro jednu osu nemohou *vyvolat příkazem CALL* podprogramy pro více os.

Struktura pohybového programu a podprogramu umožňuje flexibilitu při vykonávání a řízení osy modulem DSM314. Praktické omezení je, že v každé ose se může vykonávat současně jen jeden program. Pokud například bude povolený Program 1 pro vykonávání v Ose 1, před povolením Programu 2 pro vykonávání v Ose 2 se tento program musí buď dokončit nebo přerušit.

Pohybové programy a podprogramy pro více os

Termín více os je zadaný v definičním příkazu (na prvním řádku) programu nebo podprogramu, například: PROGRAM 2 MULTI-AXIS nebo SUBROUTINE 7 MULTI-AXIS. **Osa 1 a Osa 3 jsou jediná dvě čísla os povolená v programu nebo podprogramu pro více os.** Před vykonáním programu v obou osách musí být provedený nájezd do referenčního bodu a obě osy musí splňovat zbývající předpoklady (viz odstavec “Předpoklady pro programovaný pohyb” v této kapitole). Pohybový program pro více os může *vyvolat příkazem CALL* pouze podprogramy pro více os. V pohybovém programu nebo podprogramu pro více os je možno použít pouze jednu instrukci pohybového programu *SYNC Block*. Omezení pro "vnořování" podprogramů jsou stejná jako pro pohybový program pro jednu osu. V programu pro více os jsou dvě kategorie pohybů: Pohyby v 1 ose a pohyby ve 2 osách.

Pohyby v 1 ose: Když jsou naprogramované dva následné pohyby v 1 ose, druhý pohyb se začne vykonávat během 2 milisekund od skončení prvního pohybu.

Pohyby v 2 osách: Pohyb ve 2 osách je naprogramovaný ve třech po sobě jdoucích blocích. První ze tří bloků musí obsahovat povel *SYNC Block*. Další dva bloky obsahují pohybové povely, jeden pro osu 1 a jeden pro osu 2. Když se vykoná povel *SYNC Block*, tyto dva pohyby se spustí "společně" (během 2 milisekund). Všimněte si, že je synchronizované pouze spuštění pohybů.

Více informací o programování pro více os, struktuře programového bloku, řízení toku programu (JUMP) a povelu *SYNC Block* je uvedeno dále v této kapitole.

Typy povelů pro pohybové programy

Povely pro pohybové programy jsou řazené do čtyř kategorií:

Povely typu 1

- CALL (Podprogram)
- JUMP

Povely typu 2

- Číslo bloku
- SYNC (Synchronizace bloku)
- LOAD (Parametr)
- ACCEL (Zrychlení)
- VELOC (Rychlost)

Povely typu 3

- PMOVE (Polohovací pohyb)
- CMOVE (Souvislý pohyb)
- DWELL
- WAIT

Definiční povel programu/podprogramu

- PROGRAM
- ENDPROG
- SUBROUTINE
- ENDSUB

Povely typu 1 mohou přesměrovat cestu vykonávání programu, ale nemohou přímo ovlivnit polohování.

- Volání (podprogramu) vykoná podprogram před vrácením vykonávání programu na další povel.
- Skoky mohou být podmíněné nebo nepodmíněné. Nepodmíněný skok vždy přesměruje vykonávání na zadané místo v programu. Podmíněný skok má přiřazený bit CTL, který kontroluje. Pokud bit CTL bude v jedničce, skok přesměruje vykonávání na zadané místo v programu. Pokud bit CTL bude v nule, skok se bude ignorovat.

Povely typu 2 také nemají vliv na polohování.

- Čísla bloků představují identifikaci nebo návěští pro povel typu 3, který následuje. Čísla bloků se vyžadují pro povely JUMP; jinak jsou volitelná. Pokud programový blok neobsahuje číslo bloku, předchozí číslo bloku, pokud existuje, zůstává v platnosti.
- Povel SYNC (synchronizace bloku) je povel synchronizace pro dvě osy (ta může nebo nemusí znamenat prodlevu pohybu v jedné ose).
- Povel Načtení parametru umožňuje uživateli načíst hodnotu do registru parametrů.
- Povely Rychlost (VELOC) a Zrychlení (ACCEL) zadávají rychlost a zrychlení pro povel typu 3 MOVE nebo povely, které následují. Povely rychlosti a zrychlení zůstávají v platnosti, dokud se nezmění.

Povely typu 3 spustí nebo zastaví pohyb a tak ovlivňují řízení polohy.

- Povel pro polohování (PMOVE) a souvislý pohyb (CMOVE) zadává pohyb.
- Povely Prodleva, Čekání a Konec programu zastaví pohyb.

Programové bloky a zpracování pohybového povelu

"Programový blok" se skládá a je definovaný jako jeden (a pouze jeden) povel typu 3 s libovolným počtem a kombinací předcházejících povelů typu 1 a typu 2.

Číslo bloku má dvě základní použití: (1) představuje identifikaci Skoku (návěští), a (2) přes stavové slovo %AI *Číslo bloku* pro každou osu identifikuje část programu, která se momentálně vykonává. Povely typu 2 jsou volitelné; programový blok může obsahovat jeden povel typu 3. Povely typu 2 a podmíněné skoky se neprovedou, dokud DSM nevykoná následující povel typu 3.

Když DSM314 bude vykonávat programový blok, následující programový blok se zpracuje do zásobníkové paměti povelů. Toto načtení dopředu minimalizuje dobu na přesun bloku. Proto parametry používané při pohybu musí být načtené před tím, než se dokončí vykonávání pohybového povelu, který byl naprogramovaný o dva bloky dříve. Jinými slovy, aby se minimalizovala doba přesunu mezi dvěma bloky, nový blok se zpracuje předem během vykonávání předcházejícího bloku. Parametry programového bloku se musí načíst před tím, než se začne vykonávat předcházející blok.

Když DSM314 vykonává program pro více os, programové povely se načítají nezávisle pro každou osu a vykonají se pouze data určená pro tuto osu. Všimněte si, že programové povely pro více os neurčují osu (číslo bloku, skok, volání a konec) a proto platí pro obě osy.

Program pro více os může obsahovat povely *SYNC* pro synchronizování os v zadaných bodech. Když první osa dojde k bloku *SYNC* (blok obsahující povel *SYNC*), nevykoná následující blok, dokud druhá osa také nedojde k bloku *SYNC*. Vysvětlení k tomuto najdete v příkladu 18, "Programování pro více os" dále v této kapitole.

Předpoklady pro Programovaný pohyb

Před vyvoláním pohybového programu musí být splněné následující podmínky (pro program pro více os musí být podmínky splněné pro obě osy):

- Bit %Q *Povolit pohon* musí být v jedničce
- Bit %I *Pohon povolen* musí být v jedničce
- Bit %I *Poloha platná* musí být v jedničce
- Bit %I *Pohyb* musí být v nule
- Bit %I *Program aktivní* musí být v nule
- Bit %Q *Zrušit všechny pohyby* musí být v nule
- Poloha osy musí ležet uvnitř nakonfigurovaných mezí posuvu osy (***Horní softwarový EOT*** a ***Dolní softwarový EOT***), pokud však režim ***Softwarový konec posuvu*** nebude nakonfigurovaný jako Zakázaný
- Vstupy koncového spínače přejetí musí být v jedničce (vstup 24 V je v jedničce), pokud jsou povolené
- Povel %AQ *Vynucená rychlost digitálního serva* nesmí být aktivní
- Program, který se má vykonat, musí být platný program uložený v DSM314

Podmínky, které zastaví pohybový program

Pohybový program se okamžitě zruší, když se vyskytne některá z následujících podmínek:

- Bit %Q *Zrušit všechny pohyby* přejde do jedničky
- Bit %Q *Povolit pohon* přejde do nuly
- Koncový spínač přejetí přejde do nuly, když v konfiguraci bude ***Koncový spínač OT*** nastavený na POVOLENO.
- Další naprogramovaný pohyb, buď PMOVE nebo CMOVE, přejdou přes ***Softwarovou mez EOT*** (pokud však režim ***Softwarový konec posuvu*** nebude nakonfigurovaný jako Zakázaný).
- Vyskytne se metoda odezvy Normální zastavení nebo Rychlé zastavení. Viz Dodatek A, "Hlášení chyby".

Základy pohybového programu

Počet programů, podprogramů a příkazů

DSM314 podporuje 10 pohybových programů, 40 podprogramů a maximálně celkem 1000 příkazů pohybových programů.

Formát

- Pohybové programy a podprogramy jsou napsané s použitím ASCII textu.
- Na jednom řádku smí být pouze jeden příkaz jazyka pro programování pohybu a jeden příkaz jazyka pro programování pohybu nesmí sahat přes více než jeden řádek. Normální poznámky mohou sahat přes více řádků.
- Pro zlepšení čitelnosti a k oddělení určitých údajů je možno použít prázdné místo a prázdné řádky.
- Motion Editor není citlivý na velká nebo malá písmena.
- Všechny pohybové programy a podprogramy musí být obsažené v jednom souboru.

Programy a podprogramy pro jednu osu a pro více os

Daný program pro jednu osu musí být schopný běžet pro libovolnou osu zadanou v definičním příkazu programu. Proto povely jazyka pro programování pohybu v programech a podprogramech pro jednu osu tuto osu nezadávají. Místo toho osa zadaná příkazem PROGRAM se používá pro všechny pohybové povely v programu. Programy a podprogramy pro více os mohou vyvolat podprogramy pro více os. Obdobně programy a podprogramy pro jednu osu mohou vyvolat podprogramy pro jednu osu.

Definiční příkazy programu a podprogramu

Motion Editor vyžaduje definiční příkazy “Programu” a ”Podprogramu”, které určují číslo program/podprogramu a konfiguraci osy (PROGRAM 1 AXIS2 nebo SUBROUTINE 2 MULTI-AXIS). Tyto příkazy jsou umístěny na prvním řádku programu nebo podprogramu. Programy jsou zakončené příkazem ENDPROG, podprogramy jsou zakončené příkazem ENDSUB. Tyto příkazy slouží jako oddělovače mezi programy a podprogramy, identifikují čísla programu a podprogramu a indikují typ programu (jedna osa nebo více os).

Čísla bloků a synchronizační bloky

Čísla bloků mají příponu s dvojtečkou (například 1:). Synchronizační bloky jsou identifikované řádkem s číslem bloku následovaným povelem SYNC (například 2: SYNC). Čísla bloku se mohou objevit na řádku samostatně nebo mohou být na stejném řádku před pohybovým povelem.

Syntaxe a povely pohybového jazyka

Prázdné místo

Prázdné místo nemá žádný význam a ignoruje se kromě případu, kdy je nutné ho použít jako oddělovač. Například v "CMOVE AXIS1 50000,ABS,S-CURVE" se mezera vyžaduje jako oddělovač mezi CMOVE a AXIS1, ale nevyžaduje se ve výrazu 50000,ABS protože čárka odděluje parametry. Mezery, prázdné řádky a tabulátory se pokládají za prázdné místo.

Číselné konstanty

Číselné konstanty jsou omezené na 32-bitové celočíselné hodnoty, které mohou být se znaménkem nebo bez znaménka v závislosti na kontextu, ve kterém se používají. Všechny pohybové povely dále tento rozsah omezují. Číselné konstanty mohou být zapisované jako dekadické, hexadecimální nebo binární hodnoty. Hexadecimální a binární konstanty jsou identifikované předponami 16# respektive 2# (nepoužívejte mezeru mezi předponou a číslem). Hexadecimální a binární konstanty nemohou mít předponu se záporným znaménkem. Proto záporné hodnoty musí být zapsané ve tvaru dvojkového doplňku. Číselné konstanty mohou mezi číslicemi pro zlepšení čitelnosti velkých čísel nebo pro reprezentaci vložených desetinných teček u čísel s pevnou desetinnou tečkou obsahovat jeden znak podtržení (např. 5_000_000).

Komentáře

Znaková dvojice **(*** uvádí normální komentář, který je ukončený znakovou dvojicí ***)**. Tyto komentáře se mohou objevit kdekoliv, kde může být prázdné místo, například uvnitř nebo za příkazem pohybového programu, samostatně na řádku nebo mohou sahat přes několik řádků. Tyto komentáře se nevnořují. Znaková dvojice // uvádí jednořádkový komentář. Veškerý text, který následuje za // až do konce řádku, Motion Editor bude ignorovat. Pokud však budete používat //, nepřerušujte řádek (použitím znaku Return), jinak se bude generovat chyba. Pokud budete chtít vytvořit dlouhý komentář, který bude možno číst na obrazovce Motion Editoru bez potřeby rolování doprava, můžete použít symboly (* a *) (jsou zapotřebí na víceřádkové komentáře) společně se znakem Return (vytvoří se stisknutím klávesy Enter), který tvrdě přeruší text na další řádek.

Klíčová slova pohybového programu

Následující slova mají v jazyku pro programování pohybu zvláštní význam.

ABS	AXIS3	ENDSUB	MULTI-AXIS	SUB
ABSOLUTE	AXIS4	ENDS	PMOVE	SYNC
ACCEL	CALL	INCR	PROGRAM	VELOC
ACC	CMOVE	INCREMENTAL	PROG	VEL
ACCELERATION	DWELL	JUMP	S-CURVE	VELOCITY
AXIS1	ENDP	LINEAR	SINGLE-AXIS	WAIT
AXIS2	ENDPROG	LOAD	SUBROUTINE	

Proměnné

Pohybové programy podporují omezený soubor předdefinovaných proměnných: registry dat parametrů a bity CTL. V následující tabulce znak x představuje dekadickou hodnotu v zadaném rozsahu. Hodnota x je interpretována podle své číselné hodnoty. Proto daná proměnná může být adresována několika způsoby. Například P1 a P001 adresují Registr dat parametrů 1 a přijímá je Motion Editor.

Proměnná	Omezení
Px	$0 \leq x \leq 255$
CTLx	$01 \leq x \leq 32$

Oddělovače

Oddělovače se používají k oddělování prvků nebo se přidávají k prvkům pro účely indikace, že slouží nějaké výhradní funkci.

Oddělovač	Funkce
,	Odděluje parametry povelu
:	Identifikuje konstantu jako číslo bloku

Povely pohybového programu

Tato kapitola popisuje pohybové povely. Většina pohybových povelů má dvě formy, pro více os a pro jednu osu. Forma pro více os se používá v programech a podprogramech pro více os a vyžaduje, aby v určitých povelech osa byla zadána jako parametr (například: VELOC AXIS1 5000). V programech pro jednu osu je číslo osy zadáno v záhlaví programu (například: PROGRAM 2 AXIS1) a nesmí být uvedeno uvnitř programu.

Některá povelová klíčová slova mají přezdívky. Přezdívky klíčových slov jsou funkčně ekvivalentní skutečným klíčovým slovům. Použití přezdívky je volitelné a do značné míry je záležitostí toho, čemu kdo dává přednost.

Položky, které se objeví v ostrých závorkách ("<", ">") představují třídy položek a jsou popsané podrobněji. Položky, které se objeví v hranatých závorkách ("[", "]") jsou volitelné. Položky, které se objeví ve složených závorkách ("{", "}") se vyžadují v programech a podprogramech pro více os, ale jsou nepřípustné, když se používají v programech a podprogramech pro jednu osu.

Obecný formát povelů jazyka pro programování pohybu je KEYWORD {osa} <parametr [, parametr]>. Pokud osa bude zadána, následuje hned za klíčovým slovem. Pokud jsou zadané parametry, následují za osou. Pokud je parametrů více, jsou oddělené čárkou.

Poznámka: DSM314 nepodporuje povel NULL nebo Program Nula.

ACCEL

Příkaz ACCEL nastaví zrychlení osy pro následující pohyby a zůstane v platnosti v daném programu, dokud se nezmění. Pokud se příkaz ACCEL v programu nepoužije, pohyby se budou zrychlovat aktuální hodnotou *Zrychlení jogu*. Pohyby naprogramované před prvním příkazem ACCEL se budou zrychlovat aktuální hodnotou *Zrychlení jogu*. Pohyby naprogramované po příkazu ACCEL budou používat hodnotu v příkazu ACCEL.

Poznámka: Příkazy ACCEL pro danou osu v programu nebo podprogramu musí být oddělené příkazem PMOVE, příkazem CMOVE nebo nepodmíněným skokem.

Syntaxe:

ACCEL {<osa>} <zrychlení>

Parametr	Popis
<osa>	Číslo osy je možno zadat pouze v programu nebo podprogramu pro více os. Osu je možno zadat pomocí klíčových slov nebo konstant.
<zrychlení>	Zrychlení se zadává buď pomocí konstanty bez znaménka v rozsahu 1 - 1 073 741 823 nebo pomocí registru dat parametru.

Přezdívky:
ACC, ACCELERATION

Chyby:

1. Povely ACCEL musí být oddělené alespoň jedním pohybovým povelem.
2. Zadaná konstanta zrychlení není v rozsahu 1 - 1 073 741 823
3. Registr dat parametrů není v rozsahu 0 - 255.
4. Zadaná osa v programu pro jednu osu.
5. V programu pro více os není zadaná žádná osa.
6. Zadaná osa nepodporuje programovaný pohyb.

Číslo bloku

Čísla bloku je možno použít jako cíl pro povely JUMP. Mohou se objevit na řádku samostatně nebo před povelem.

Syntaxe:
<číslo bloku>: [<povel>]

Parametr	Popis
<číslo bloku>	Číslo bloku musí být v rozsahu 1 – 65 535
<povel>	Za číslem bloku na stejném řádku smí následovat libovolný povel kromě PROGRAM, SUBROUTINE, ENDPROG, ENDSUB nebo jiné číslo bloku.

Přezdívky:
Žádné

Chyby:

1. Všechna čísla bloku a čísla synchronizačních bloků musí být v rámci programu nebo podprogramu jednoznačná.
2. Číslo bloku musí být v rozsahu 1 - 65 535

CALL

Povel CALL vyvolá podprogram z programu nebo podprogramu.

Syntaxe:
CALL <cíl podprogramu>

Parametr	Popis
<cíl podprogramu>	Cíl podprogramu zadaný jako konstanta, 1 - 40, nebo registr dat parametrů.

Přezdívky:
Žádné

Chyby:

1. Číslo podprogramu musí být v rozsahu 1 - 40, nebo registr dat parametru v rozsahu 0 - 255.
2. Pokud volajícím je podprogram, nemůže volat sebe sama (rekurzivní volání je nepřípustné) nebo volat jiný podprogram, který ho přímo nebo nepřímo adresuje.
3. Cíl volání podprogramu musí být definovaný ve stejném souboru.
4. Programy a podprogramy pro jednu osu mohou vyvolat pouze podprogramy pro jednu osu. Programy a podprogramy pro více os mohou vyvolat podprogramy pro více os.

CMOVE

Povel CMOVE naprogramuje souvislý pohyb s použitím režimu zadané polohy a zrychlení.

Syntaxe:
CMOVE {<osa>} <poloha>, <režim polohování>, <režim zrychlování>

Parametr	Popis
<osa>	Osu je možno zadat pouze v programu nebo podprogramu pro více os. Osu je možno zadat pomocí klíčového slova AXISx nebo konstant.
<poloha>	Poloha cíle. Může být konstanta nebo registr dat parametru.
<režim polohování>	Zadává inkrementální (INCR) nebo absolutní (ABS) polohování.
<režim zrychlování>	Zadává pro pohyb lineární zrychlení (LINEAR) nebo zrychlení po S křivce (S-CURVE).

Přezdívky:
Žádné

Chyby:
1. Zadaná osa v programu pro jednu osu.
2. V programu pro více os není zadaná žádná osa.
3. Poloha musí být v rozsahu –536 870 912 až 536 870 911 nebo registr dat parametru v rozsahu 0 - 255.
4. Režim polohování musí být buď INCR nebo ABS.
5. Režim zrychlování musí být buď LINEAR nebo S-CURVE.
6. Zadaná osa nepodporuje programovaný pohyb.

DWELL

Povel DWELL způsobí, že se pohyb zastaví na zadanou dobu před zpracováním dalšího povelu. Zadání nulové prodlevy (buď jako konstanta nebo hodnota v registru dat parametru) bude mít za následek, že se neprovede žádná prodleva *(to je změna oproti funkci APM a DSM302)*.

Jeden povel DWELL platí pouze pro jednu osu. Proto v programu pro více os je nutno pro každý povel DWELL zadat číslo osy. Například: DWELL AXIS1 2000. Pokud budete v programu pro více os chtít zastavit obě osy, je nutno použít povel DWELL pro každou osu.

Syntaxe:
DWELL {<osa>} <prodleva>

Parametr	Popis
<osa>	Osu je možno zadat pouze v programu nebo podprogramu pro více os. Osu je možno zadat pomocí klíčového slova AXISx nebo konstant.
<prodleva>	Prodleva v milisekundách zadaná jako konstanta nebo registr dat parametrů. Rozsah je 0 - 60 000 ms. Hodnota 0 se interpretuje jako nulový povel.

Přezdívky:
Žádné

Chyby:
1. Zadaná osa v programu pro jednu osu.
2. V programu pro více os není zadaná žádná osa.
3. Prodleva musí být v rozsahu 0 - 60 000 nebo registr dat parametru musí být v rozsahu 0 - 255.
4. Zadaná osa nepodporuje programovaný pohyb.

ENDPROG

Příkaz ENDPROG ukončí definici programu pohybu.

Syntaxe:
ENDPROG

Přezdívky:
ENDP

ENDSUB

Příkaz ENDSUB ukončí definici podprogramu pohybu.

Syntaxe:
ENDSUB

Přezdívky:
ENDS

JUMP

Skok na číslo bloku nebo synchronizovaný blok uvnitř aktuálního programu nebo podprogramu. Skok může být podmíněný na základě stavu bitu CTL nebo nepodmíněný.

Syntaxe:
JUMP <podmínka>, <cíl>

Parametr	Popis
<podmínka>	Podmínka skoku, musí zadávat CTL01 - CTL32 nebo UNCOND
<cíl>	Cílový blok nebo číslo synchronizačního bloku

Přezdívky:
Žádné

Chyby:

1. Podmínka skoku musí být CTL v rozsahu 1 - 32, nebo klíčové slovo UNCOND.
2. Cílový blok musí být v rozsahu 1 – 65 535 a musí být definovaný ve stejném programu nebo podprogramu jako příkaz JUMP.

LOAD

Inicializuje nebo mění registr dat parametru 32-bitovým dvojkovým doplňkem celočíselné hodnoty.

Syntaxe:
LOAD <registr dat parametru>, <načítaná hodnota>

Parametr	Popis
<registr dat parametru>	Registr dat parametru, který se má inicializovat. Omezeno na registry P000 - P255.
<načítaná hodnota>	32-bitová číselná konstanta.

Přezdívky:
Žádné

Chyby:

1. Registr dat parametru musí být v rozsahu P000 - P255.
2. Načítaná hodnota musí být v rozsahu 32-bitového dvojkového doplňku hodnoty.

PMOVE

Povel PMOVE naprogramuje polohovací pohyb používající režim polohování a režim zrychlování.

Syntaxe:
PMOVE {<osa>} <poloha>, <režim polohování>, <režim zrychlování>

Parametr	Popis
<osa>	Osu je možno zadat pouze v programu nebo podprogramu pro více os. Osu je možno zadat pomocí klíčového slova AXISx nebo konstant.
<poloha>	Poloha cíle. Může být konstanta nebo registr dat parametru.
<režim polohování>	Zadává inkrementální (INCR) nebo absolutní (ABS) polohování.
<režim zrychlování>	Zadává pro pohyb lineární zrychlení (LINEAR) nebo zrychlení po S křivce (S-CURVE).

Přezdívky:
Žádné

Chyby:

1. Zadaná osa v programu pro jednu osu.
2. V programu pro více os není zadaná žádná osa.
3. Poloha musí být v rozsahu -536 870 912 až 536 870 911 nebo registr dat parametru v rozsahu 0 - 255.
4. Režim polohování musí být buď INCR nebo ABS.
5. Režim zrychlování musí být buď LINEAR nebo S-CURVE.
6. Zadaná osa nepodporuje programovaný pohyb.

PROGRAM

Příkaz PROGRAM je prvním příkazem v pohybovém programu. Programový příkaz identifikuje číslo programu (1-10) a konfiguraci osy. Definice programů nelze vnořovat.

Existují dva typy pohybových programů, pro jednu osu, kde jsou všechny povely směřované na stejnou osu, a pro více os, které mohou obsahovat povely pro osu 1 a osu 2. Typ programu je určený příkazem PROGRAM. Program pro jednu osu je identifikovaný zadáním cílové osy, za kterým následuje číslo programu (například PROGRAM 3 AXIS1). Program pro více os je identifikovaný slovem MULTI-AXIS, za kterým následuje číslo programu (například PROGRAM 4 MULTI-AXIS).

Konfigurace programové osy se používá k zajištění, jestli se v pohybových povelech programu musí nebo nemusí dodat parametr osy. Také stanoví omezení, že programy pro více os smí volat podprogramy pro více os a programy pro jednu osu smí volat podprogramy pro jednu osu. Osa zadaná v programu pro jednu osu se používá v každém podprogramu, který tento program volá; proto uvnitř podprogramu pro jednu osu se nesmí nikde zadat číslo osy.

Syntaxe:

PROGRAM <číslo programu> <konfigurace osy>

Parametr	Popis
<číslo programu>	Číslo programu musí být dekadická hodnota v rozsahu 1 - 10. Uvnitř zdrojového souboru každý definovaný PROGRAM musí mít jednoznačné číslo.
<konfigurace osy>	Konfigurace osy musí mít hodnotu MULTI-AXIS pro programy pro více osy nebo cíl osy (například AXIS1) pro programy pro jednu osu. Osy je možno zadávat pomocí klíčových slov AXISx nebo konstant, kde x = 1 - 4.

Přezdívky:

PROG

SUBROUTINE

Příkaz SUBROUTINE je první příkaz v pohybovém podprogramu. Příkaz podprogramu identifikuje číslo podprogramu (1 - 40) a konfiguraci osy. Definice podprogramů nelze vnořovat.

Existují dva typy pohybových podprogramů, pro jednu osu, kde jsou všechny povely směřované na stejnou osu, a pro více os, které mohou obsahovat povely pro osu 1 a osu 2. Typ podprogramu je zadaný příkazem SUBROUTINE. Podprogram pro jednu osu je identifikovaný slovem SINGLE-AXIS, za kterým následuje číslo podprogramu. Podprogram pro více os je identifikovaný slovem MULTI-AXIS, za kterým následuje číslo podprogramu.

Konfigurace podprogramové osy se používá k určení, jestli se pro pohybové povely podprogramu musí nebo nemusí dodat parametr osy. Také stanoví omezení, že podprogramy pro více os smí volat podprogramy pro více os a podprogramy pro jednu osu smí volat podprogramy pro jednu osu. Podprogram pro jednu osu používá číslo osy určené volajícím programem.

Syntaxe:
SUBROUTINE <číslo podprogramu> <konfigurace osy>

Parametr	Popis
<číslo podprogramu>	Číslo podprogramu musí být dekadická hodnota v rozsahu 1 - 40. Uvnitř zdrojového souboru musí mít každý definovaný podprogram jednoznačné číslo.
<konfigurace osy>	Konfigurace osy musí mít hodnotu MULTI-AXIS nebo SINGLE-AXIS.

Přezdívky:
SUB

Synchronizační blok

Synchronizační blok je zvláštní případ čísla bloku. Synchronizační blok je možno použít pouze v programu pro více os.

Synchronizační blok je identifikovaný číslem bloku, za kterým následuje povel SYNC. Povel SYNC se musí objevit na stejném řádku jako číslo bloku.

Syntaxe:
<číslo bloku>: SYNC

Parametr	Popis
<číslo bloku>	Číslo bloku musí být v rozsahu 1 - 65 535

Přezdívky:
žádné

Chyby:

1. Synchronizační bloky se mohou objevit pouze v programech pro více os.
2. Všechna čísla bloku a čísla synchronizačních bloků musí být v rámci programu nebo podprogramu jednoznačná.
3. Synchronizační bloky a čísla bloků se nesmí objevit v po sobě jdoucích příkazech bez povelu mezi nimi.
4. Čísla synchronizačních bloků musí být v rozsahu 1 - 65 535.

VELOC

Nastaví rychlost osy používanou následujícími programovými pohybovými povely a zůstává v platnosti, dokud nebude změněná jiným příkazem VELOC. Pokud se příkaz VELOC v programu nepoužije, pohyby budou používat aktuální hodnotu *Rychlost jogu*. Rovněž pohyby naprogramované před prvním příkazem VELOC budou používat aktuální hodnotu *Rychlost jogu.*.

Poznámka: Příkazy VELOC pro danou osu v programu nebo podprogramu musí být oddělené příkazem PMOVE, příkazem CMOVE nebo nepodmíněným skokem.

Syntaxe:
VELOC {<osa>} <rychlost>

Parametr	Popis
<osa>	Osu je možno zadat pouze v programu nebo podprogramu pro více os. Osu je možno zadat pomocí klíčového slova AXISx nebo konstant.
<rychlost>	Požadovaná rychlost. Může být konstanta nebo registr dat parametru.

Přezdívky:
VEL, VELOCITY

Chyby:

1. Zadaná osa v programu pro jednu osu.
2. V programu pro více os není zadaná žádná osa.
3. Rychlost musí být konstanta v rozsahu 1 - 838 8607.
4. Povely VELOC musí být oddělené alespoň jedním pohybovým povelem.
5. Zadaná osa nepodporuje programovaný pohyb.

WAIT

Umožňuje synchronizaci s některým externím dějem pomocí bitů CTL. Vykonání dalšího povelu se pozdrží do doby, než určený bit CTL bude v jedničce.

Jeden povel WAIT platí pouze pro jednu osu. Proto v programu pro více os je nutno určit číslo osy, pro kterou WAIT platí. Například: WAIT AXIS1 CTL01. Pokud budete chtít v programu pro více os vyvolat čekání obou os, je nutno použít povel WAIT pro každou osu.

Syntaxe:

WAIT {<osa>} <ctl>

Parametr	Popis
<osa>	Osu je možno zadat pouze v programu nebo podprogramu pro více os. Osu je možno zadat pomocí klíčového slova AXISx nebo konstant.
<ctl>	Udává CTL01 – CTL32.

Přezdívky:

žádné

Chyby:

1. Zadaná osa v programu pro jednu osu.
2. V programu pro více os není zadaná žádná osa.
3. CTL musí být v rozsahu 1 - 32.
4. Zadaná osa nepodporuje programovaný pohyb.

Struktura programů a podprogramů

Struktura programu pro jednu osu

- **Definiční příkaz PROGRAM**. Musí být na prvním řádku programu. Musí identifikovat číslo programu a číslo osy. Mezi klíčovým slovem PROGRAM a číslem programu je mezera. Naopak, v číslu programu se mezera nesmí vyskytovat. Například:

  PROGRAM 1 AXIS3

- **Tělo**. Tělo programu obsahuje konkrétní programové povely. Všimněte si, že v programu pro jednu osu se v žádném povelu nesmí zadávat číslo osy. V opačném případě se bude generovat chyba. Příkladem správné syntaxe programu pro jednu osu je:

  ACCEL 50000

- **Konec programu**. Používá příkaz ENDPROG. Tento příkaz jasně identifikuje konec programu a pomáhá oddělit jeden program nebo podprogram od druhého. ENDPROG musí být jediná věc na posledním řádku každého programu:

  ENDPROG

Příklad programu pro jednu osu

Všimněte si, že číslo osy se zadává na prvním řádku a nezadává se v těle programu. Všimněte si také, že v názvu AXIS1 není žádná mezera.

```
PROGRAM 2 AXIS1
        ACCEL   50000
        VELOC   5000
        PMOVE  10000, ABS, LINEAR
        DWELL  6000
        PMOVE  5000, ABS, LINEAR
ENDPROG
```

Struktura programu pro více os

- **Definiční příkaz PROGRAM**. Musí být na prvním řádku programu. Pomocí názvu MULTI-AXIS musí identifikovat číslo programu a skutečnost, že toto je program pro více os. Například:

  PROGRAM 3 MULTI-AXIS

- **Tělo**. Tělo programu obsahuje konkrétní programové povely. Všimněte si, že v programu pro více os musíte číslo osy zadávat v mnoha povelech. V opačném případě se bude generovat chyba. Všimněte si, že v termínu čísla osy, například AXIS1, se nesmí vyskytovat žádná mezera. Příkladem správné syntaxe programového povelu pro více os je:

  ACCEL AXIS1 50000

- **Konec programu**. Používá příkaz ENDPROG. Tento příkaz jasně identifikuje konec programu a pomáhá oddělit jeden program nebo podprogram od druhého. ENDPROG musí být jediná věc na posledním řádku každého programu:

  ENDPROG

Příklad programu pro více os

Všimněte si, že termín MULTI-AXIS se musí použít v příkazu PROGRAM na prvním řádku a že čísla osy se zadávají v odpovídajících povelech v těle programu.

```
PROGRAM 1 MULTI-AXIS
        ACCEL AXIS1 500000
        VELOC AXIS1 5000
1:      CMOVE AXIS2 -100000, ABS, LINEAR
        DWELL AXIS2 6000
        JUMP CTL31, 1
        CALL P255
        LOAD P215, 2000
        PMOVE AXIS1 8388607, INCR, S-CURVE
ENDPROG
```

Struktura podprogramu pro jednu osu

- **Definiční příkaz SUBROUTINE**. Musí být na prvním řádku podprogramu. Musí identifikovat číslo podprogramu a obsahovat příkaz SINGLE-AXIS. Například:

  SUBROUTINE 3 SINGLE-AXIS

- **Tělo**. Tělo podprogramu obsahuje konkrétní programové povely. Všimněte si, že v podprogramu pro jednu osu se v žádném povelu nesmí zadávat číslo osy. V opačném případě se bude generovat chyba. Příkladem správné syntaxe povelu podprogramu pro jednu osu je:

  ACCEL 50000

- **Konec podprogramu**. Používá příkaz ENDSUB. Tento příkaz jasně identifikuje konec podprogramu a pomáhá oddělit jeden program nebo podprogram od druhého. ENDSUB musí být jediná věc na posledním řádku každého podprogramu:

  ENDSUB

Příklad podprogramu pro jednu osu

V podprogramu pro jednu osu nesmí být zadáno číslo osy. Je to z toho důvodu, že podprogram pro jednu osu bude platit pro osu zadanou v programu pro jednu osu, který ho volá. To umožňuje, aby podprogram použily jiné programy pro jednu osu bez ohledu na konkrétní číslo osy, které specifikují.

```
SUBROUTINE 15 SINGLE-AXIS
        ACCEL   50000
        VELOC   10000
        PMOVE  200000, ABS, LINEAR
        DWELL  3000
        PMOVE  50000, ABS, LINEAR
ENDSUB
```

Struktura podprogramu pro více os

- **Definiční příkaz SUBROUTINE**. Musí být na prvním řádku podprogramu. Pomocí názvu MULTI-AXIS musí identifikovat číslo podprogramu a skutečnost, že toto je program pro více os. Například:

  SUBROUTINE 7 MULTI-AXIS

- **Tělo**. Tělo podprogramu obsahuje konkrétní programové povely. Všimněte si, že v podprogramu pro více os musíte číslo osy zadávat v mnoha povelech. V opačném případě se bude generovat chyba. Příkladem správné syntaxe povelu podprogramu pro více os je:

  ACCEL AXIS2 50000

- **Konec podprogramu**. Používá příkaz ENDSUB. Tento příkaz jasně identifikuje konec podprogramu a pomáhá oddělit jeden program nebo podprogram od druhého. ENDSUB musí být jediná věc na posledním řádku každého podprogramu:

```
ENDSUB
```

Příklad podprogramu pro více os

```
SUBROUTINE 2 MULTI-AXIS
        ACCEL  AXIS2 P100
        VELOC  AXIS2 P105
2:      SYNC
        CMOVE AXIS2 P001, INCR, S-CURVE
        DWELL  AXIS2 P001
        JUMP     CTL01, 2
        PMOVE  AXIS2 P214, ABS, LINEAR
ENDSUB
```

Příklady použití povelů

Následující příklady nejsou kompletní programy. Například v mnoha případech nejsou ukázané příkazy PROGRAM a ENDPROG. *Aby se program úspěšně zkompiloval, bylo by nutné doplnit tyto příkazy (ve správném kontextu).*

Naprogramované pohyby mají tři parametry:

1. *Vzdálenost* (data) pro ujetí nebo *poloha*, na kterou se má provést posuv,
2. *Typ adresy polohování* (modifikátor povelu) používaný pro pohyb, a
3. *Typ zrychlení* (modifikátor povelu) používaný při vykonávání pohybu.

Poznámka: Pohybové programy mohou obsahovat příkazy, které jako data související s povely nebo proměnnými používají konstanty, které se také nazývají parametry (P0 - P255).

Absolutní nebo inkrementální polohování

Absolutní polohování

V pohybu s absolutním polohováním je první parametr poloha, do které se má pohyb vykonat. Níže je uvedený příklad pohybu s absolutním polohováním.

```
PMOVE 5000, ABS, LINEAR
```

V tomto příkladu osa vykoná pohyb ze své okamžité polohy, ať je jakákoliv, na polohu 5000. Proto skutečná vzdálenost pohybu závisí na aktuální poloze osy, když přijde tento povel pohybu. Pokud výchozí poloha je 0, osa vykoná pohyb o 5000 uživatelských jednotek v kladném směru. Pokud výchozí poloha je 8000, osa vykoná pohyb o 3000 uživatelských jednotek v záporném směru. Pokud výchozí poloha bude 5000, osa pohyb nevykoná.

Inkrementální polohování

V inkrementálním pohybu první parametr udává vzdálenost pohybu z okamžité polohy. DSM314 přeloží vzdálenost inkrementálního pohybu do absolutní polohy pohybu. Tím se eliminuje kumulace chyb. Níže je uvedený příklad pohybu s inkrementálním polohováním.

```
PMOVE 5000, INCR, LINEAR
```

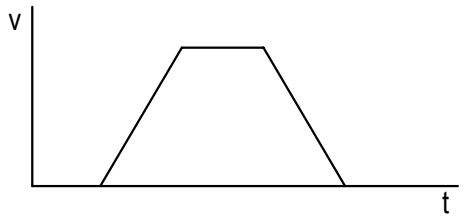
V tomto příkladu osa vykoná pohyb ze své okamžité polohy do polohy větší o 5000 uživatelských jednotek. U inkrementálního pohybu první parametr udává skutečný počet uživatelských jednotek, o které se má osa přesunout.

Typy zrychlení

Lineární zrychlení

Příklad profilu lineárního pohybu s grafem rychlosti jako funkce času je uvedený na obrázku 7-1. Jak je ukázáno, lineární pohyb používá konstantní (lineární) zrychlení. Plocha pod grafem představuje vzdálenost posuvu.

```
ACCEL  1000
VELOC  2000
PMOVE 6000, INCR, LINEAR
```

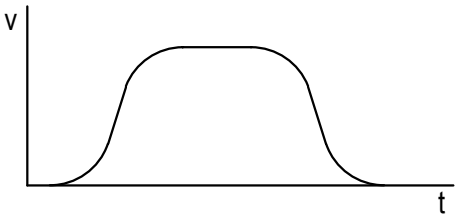


Obrázek 7-1. Příklad lineárního pohybu

Zrychlení po S křivce

Příklad pohybu po S křivce se zobrazením rychlosti jako funkce času je ukázaný níže. Jak je znázorněno, zrychlení po S křivce je nelineární. Když pohyb začne, zrychlení začne pomalu a narůstá až do dosažení naprogramovaného zrychlení. To by měl být střední bod zrychlení. Pak se zrychlení začne snižovat až do nuly, kdy se dosáhne naprogramované rychlosti. Pohyb po S křivce vyžaduje dvojnásobek času a vzdálenosti ke zrychlování a zpomalování než potřebuje srovnatelný lineární pohyb. Plocha pod grafem představuje vzdálenost posuvu.

```
ACCEL      2000
VELOC      2000
PMOVE      8000, INCR, S-CURVE
```



Obrázek 7-2. Příklad pohybu po S křivce

Typy povelů programovaných pohybů

Následující příklady nejsou kompletní programy. Například v mnoha případech nejsou ukázané příkazy PROGRAM a ENDPROG. Aby se program úspěšně zkompiloval, bylo by nutné doplnit tyto příkazy (ve správném kontextu).

Polohovací pohyb (PMOVE)

PMOVE se musí vždy dostat až do úplného zastavení. Zastavení musí být dostatečně dlouhé, aby se bit %I *V poloze* mohl nastavit do jedničky před tím, než může začít další pohyb.

PMOVE používá poslední naprogramovanou rychlost a zrychlení. Pokud v pohybovém programu nebyl zjištěný povel VELOC, použije se implicitně ***Rychlost jogu***. Pokud v pohybovém programu nebyl zjištěný povel ACCEL, použije se implicitně ***Zrychlení jogu***.

Plynulý pohyb (CMOVE)

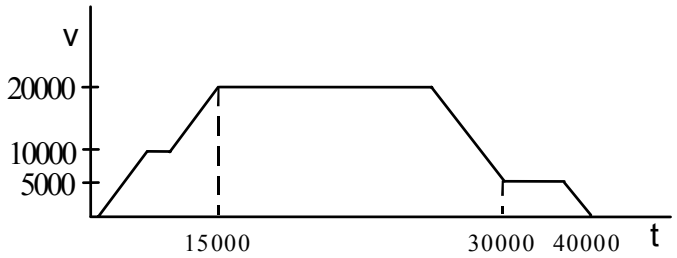
CMOVE se při dokončení nezastaví, pokud za ním nebude následovat povel DWELL nebo WAIT, další naprogramovaná rychlost bude nulová nebo to bude poslední programový povel. Nebude čekat na to, až před přechodem na další pohyb bit %I *V poloze* bude v jedničce. Normální CMOVE je povel, který dosáhne své naprogramované polohy ve stejném čase, kdy dosáhne rychlosti *následujícího* povelu Pohyb.

CMOVE používá poslední naprogramovanou rychlost a zrychlení. Pokud v pohybovém programu nebyl zjištěný povel VELOC, použije se implicitně ***Rychlost jogu***. Pokud v pohybovém programu nebyl zjištěný povel ACCEL, použije se implicitně ***Zrychlení jogu***.

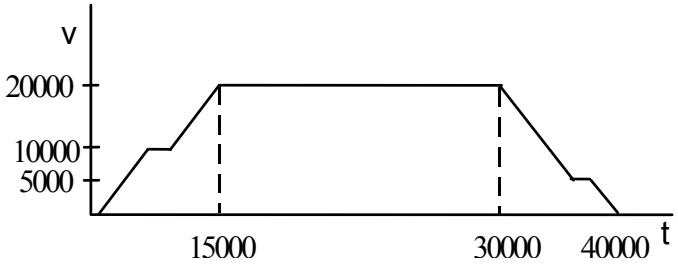
Pro přinucení DSM314, aby dosáhlo naprogramované polohy CMOVE *před* zahájením změny rychlosti související s *dalším* pohybovým povelem (to je vykonání celého povelu CMOVE konstantní rychlostí), je možno použít zvláštní tvar povelu CMOVE. **Naprogramování inkrementálního povelu CMOVE s operandem 0 (například: CMOVE 0, INCR, LINEAR) vyvolá prodlevu změny rychlosti serva až do *dalšího* pohybového povelu v řadě.** Tento vliv znázorňuje následující sekvence povelů (předpokládejme, že byly zvolené povely ACCEL, aby se pohyby mohly dokončit normálně):

Povel	Data	Komentáře
VELOC	10000	//Nastavení rychlosti prvního pohybu = 10000
CMOVE	15000, ABS, LINEAR	//Dosažení rychlosti druhého pohybu (20000) na poloze = 15000
VELOC	20000	//Nastavení rychlosti druhého pohybu = 20000
CMOVE	30000, ABS, LINEAR	(*Zůstat na rychlosti = 20000 až do polohy = 30000, pak změnit na rychlost = 5000*)
CMOVE	0, INCR, LINEAR	(*Příznak indikuje, aby DSM314 počkalo na další pohyb před změnou na další rychlost*)
VELOC	5000	//Nastavení rychlosti třetího pohybu = 5000
PMOVE	40000, ABS, LINEAR	//Poloha konečného zastavení = 40000

Poznámka: Pro zvýšení čitelnosti se v programu použily znaky prázdného místa (mezera, tabulátor, atd.).



Obrázek 7-3. Příklad 1, Před vložením CMOVE (0)



Obrázek 7-4. Příklad 2 , Po vložení CMOVE (0)

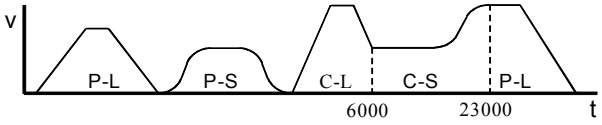
Naprogramované pohyby

Kombinací povelů CMOVE a PMOVE, absolutních a inkrementálních pohybů a lineárního pohybu a pohybu po S křivce je možno virtuálně naprogramovat jakýkoliv profil pohybu. Následující příklady ukazují některé příklady profilů pohybu i některé běžné chyby při programování pohybu.

Příklad 1: Kombinace povelů PMOVE a CMOVE

Tento příklad ukazuje, jak je jednoduché kombinovat povely PMOVE a CMOVE k vytvoření profilů pohybu.

```
ACCEL   1000
VELOC   2000
PMOVE   5000, ABS, LINEAR
VELOC   1200
PMOVE   10000, ABS, S-CURVE
ACCEL   1500
VELOC   2800
CMOVE   6000, INCR, LINEAR
VELOC   1200
CMOVE   23000, ABS, S-CURVE
ACCEL   1000
VELOC   2800
PMOVE   5000, INCR, LINEAR
```



Obrázek 7-5. Kombinace povelů PMOVE a CMOVE

První PMOVE provede zrychlení na naprogramovanou rychlost, ujede vzdálenost a zpomalí až do zastavení. Je to proto, že pohyb se zastaví po každém PMOVE. Když se zastaví první pohyb, bude v naprogramované vzdálenosti.

Druhý pohyb je pohyb PMOVE po S křivce. Podobně jako první pohyb, zrychlí se na naprogramovanou rychlost, bude se pohybovat určitou dobu a zpomalí na nulovou rychlost, protože to je PMOVE.

Další pohyb je lineární CMOVE. Zrychlí na naprogramovanou rychlost, bude se pohybovat určitou dobu a pak zpomalí na nižší rychlost s použitím lineárního zrychlení. Když CMOVE bude končit, bude v naprogramované poloze právě skončeného pohybu a rychlosti dalšího pohybu. Takže když bude začínat čtvrtý pohyb, bude již na naprogramované rychlosti.

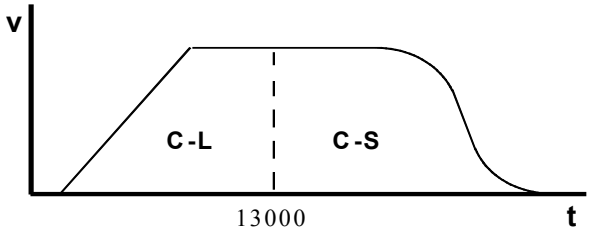
Čtvrtý pohyb je CMOVE, takže když se bude blížit konečné poloze, zrychlí tak, aby se při dokončení dostal na rychlost pátého pohybu. Graf ukazuje, že zrychlení čtvrtého pohybu je S křivka.

A konečně pátý pohyb začne a pohybuje se naprogramovanou rychlostí po dobu, dokud se nezpomalí na nulu. Všechny následující pohyby po pátém pohybu budou začínat na nulové rychlosti, protože pátý pohyb je PMOVE.

Příklad 2: Změna režimu zrychlení během profilu

Následující příklad ukazuje, jak je možno použít rozdílné zrychlení a režim rovnoměrného zrychlení s použitím povelů CMOVE. První CMOVE zrychlí lineárně na naprogramovanou rychlost. Protože druhá rychlost CMOVE je shodná s první rychlostí, první CMOVE dokončí svůj pohyb bez změny rychlosti. Zrychlení druhého pohybu je po S křivce, kterým se zpomaluje na nulovou rychlost.

```
ACCEL    2000
VELOC    6000
CMOVE    13000, ABS, LINEAR
ACCEL    4000
CMOVE    15000, INCR, S-CURVE
```

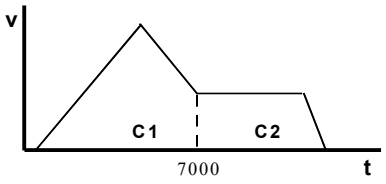


Obrázek 7-6. Změna režimu zrychlení během profilu

Příklad 3: Nedostatek vzdálenosti k dosažení naprogramované rychlosti

Povely CMOVE a PMOVE je možno naprogramovat tak, že nebudou mít dostatek vzdálenosti k dosažení naprogramované rychlosti. Následující graf ukazuje CMOVE, který nemůže dosáhnout naprogramované rychlosti. DSM314 zrychluje do bodu, kde musí začít zpomalování, aby dosáhlo naprogramované polohy C1 rychlostí druhého CMOVE.

```
ACCEL   2000
VELOC   8000
CMOVE   7000, INCR, LINEAR
ACCEL   10000
VELOC   2000
CMOVE   4400, INCR, LINEAR
```

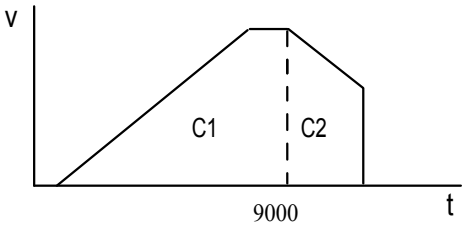


Obrázek 7-7. Nedostatek vzdálenosti k dosažení naprogramované rychlosti

Příklad 4: Uváznutí pohybu, když je nedostatečná vzdálenost

Vážná chyba programování vyvolá "uváznutí" (tj. generátor povelů nebude mít žádnou přijatelnou volbu) pohybu při vysoké rychlosti, když je nedostatečná vzdálenost. V následujícím příkladu CMOVE zrychluje na vysokou rychlost. Druhý CMOVE má shodnou rychlost. Avšak vzdálenost zadaná v druhém CMOVE, je příliš krátká. Proto se osa bude pohybovat velmi vysokou rychlostí a musí zastavit na krátké vzdálenosti. Pokud naprogramované zrychlení nebude dostatečně velké, dojde k následujícímu profilu. DSM314 se bude snažit zabránit přejezdu konečné polohy zadáním nulové rychlosti. Tato prudká změna rychlosti je nežádoucí a může způsobit poškození stroje.

```
ACCEL   500
VELOC   3000
CMOVE  9000, ABS, LINEAR
ACCEL   600
CMOVE  4800, INCR, LINEAR
```



Obrázek 7-8. Uváznutí DSM314, když je nedostatečná vzdálenost

Povel DWELL

Povel DWELL se používá ke generování zastavení na zadaný počet milisekund. Povel DWELL může použít hodnotu uloženou v určeném parametru.

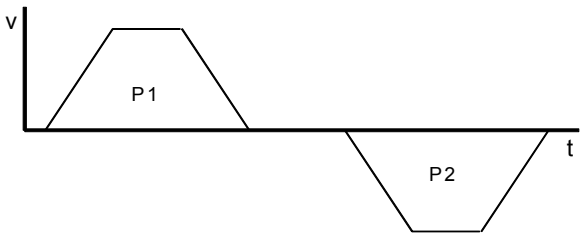
Povel DWELL po povelu CMOVE bude mít za následek, že CMOVE zastaví před dalším pohybem, pokud není zadaná doba prodlevy nula milisekund. DWELL se bude pokládat za "prázdný" povel a přeskočí se (CMOVE bude pokračovat na další pohyb za povelem DWELL), pokud povel DWELL bude mít nulovou hodnotu nebo bude adresovat registr parametrů, který má nulovou hodnotu.

Jeden povel DWELL platí pouze pro jednu osu. Proto v programu pro více os je nutno pro každý povel DWELL zadat číslo osy. Například: DWELL AXIS1 2000. Pokud budete chtít v programu pro více os zastavit obě osy, je nutno použít povel DWELL pro každou osu.

Příklad 5: DWELL

Na následujícím obrázku je ukázaný jednoduchý profil pohybu, který vykoná pohyb do určitého bodu, počká a pak se vrátí do původního bodu.

```
ACCEL   30000
VELOC   15000
PMOVE  120000, ABS, LINEAR
DWELL  4000
PMOVE  0, ABS, LINEAR
```



Obrázek 7-9. Příklad povelu prodlevy

Povel čekání

Povel WAIT je podobný povelu DWELL. Místo, aby se generovalo zastavení na zadanou dobu, WAIT zastaví programový pohyb, dokud určený bit CTL nepřejde do jedničky. Pohyb se tak zastaví při každém zjištění povelu WAIT, i když bit CTL bude v jedničce před tím, než se v programu dojde k povelu WAIT. Aktivací pro pokračování programu může být libovolný z dvanácti bitů CTL.

Pokud v předchozím příkladu bude povel DWELL nahrazený povelem WAIT, pohybový profil bude stejný ale s tím rozdílem, že druhý PMOVE se nespustí, dokud bit CTL nepřejde do jedničky. Pokud bit CTL byl v jedničce, když program dospěl k povelu WAIT, druhý PMOVE by se spustil hned po dokončení prvního PMOVE.

Také pokud se v předchozím příkladu povel WAIT použije místo povelu DWELL, povely CMOVE a PMOVE by generovaly stejné rychlostní profily. Povel WAIT zastaví pohyb, ať už předchozí pohyb bude CMOVE nebo PMOVE.

Jeden povel WAIT platí pouze pro jednu osu. Proto v programu pro více os je nutno pro každý povel WAIT zadat číslo osy. Například: WAIT AXIS1 CTL001. Pokud budete chtít v programu pro více os vyvolat čekání obou os, je nutno použít povel WAIT pro každou osu.

Podprogramy

DSM314 může mít uložených až deset samostatných programů a čtyřicet podprogramů. Podprogramy je možno definovat jako dva typy: *pro jednu osu* a *pro více os*. Podprogramy je možno použít pro všechny pohybové programy vytvořené pomocí Motion Editoru. Povely uvnitř podprogramů pro jednu osu neobsahují číslo osy; to umožňuje vyvolat podprogramy pro jednu osu z libovolného programu pro jednu osu (povely v podprogramu používají číslo osy určené volajícím programem). Povely v podprogramech pro více os obsahují čísla os stejně jako povely v programech pro více os. Podprogramy pro více os je možno vyvolat pouze z programů nebo podprogramů pro více os. Podprogramy pro jednu osu je možno vyvolat pouze z programů nebo podprogramů pro jednu osu. Například program pro osu 1 a program pro osu 1 může vyvolat stejný podprogram pro jednu osu současně. Každý podprogram musí mít přiřazené jednoznačné číslo v rozmezí 1 až 40.

Podprogramy se programují pomocí povelu CALL, který určuje číslo volaného podprogramu. Když se během vykonávání programu zjistí CALL, vykonávání programu se přesměruje na podprogram. Po dokončení podprogramu vykonávání programu pokračuje povelem za povelem CALL. Podprogramy je možno vyvolat z jiného podprogramu, ale jakmile se podprogram vyvolá, musí se dokončit před tím, než ho je možno vyvolat pro stejnou osu. Proto je rekurzivní volání nepřípustné.

Čísla bloků a skoky

Čísla bloků se používají jako referenční body uvnitř pohybového programu a pro kontrolu testování skoku. Datové slovo %AI zobrazuje aktuální číslo bloku, které je možno monitorovat, aby se zajistilo správné vykonávání programu nebo se určilo, kdy by se měly objevit nějaké děje. Číslo bloku také může sloužit jako cíl pro povel JUMP. Skoky mohou být podmíněné nebo

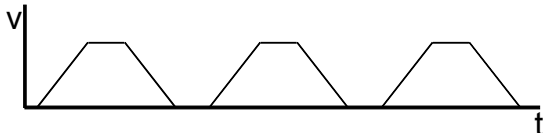
nepodmíněné. Povel nepodmíněného skoku jednoduše řekne DSM314, že má pokračovat ve vykonávání programu s cílovým číslem bloku. Podmíněný skok se vykoná, pouze když se vyskytne zadaná podmínka. Příklady obou typů jsou uvedené níže.

Nepodmíněné skoky

Příklad 6: Nepodmíněný skok

V následujícím příkladu program vykoná PMOVE, provede prodlevu 2 sekundy, pak nepodmíněně skočí zpět na začátek programu na blok 1. Povel PMOVE se bude opakovat, dokud se nedosáhne meze posuvu (***Horní softwarový EOT*** nebo ***Dolní softwarový EOT***) nebo Koncového spínače přejetí. K zastavení programu je také možno použít bitový povel %Q *Zrušit všechny pohyby*.

```
      ACCEL   10000
      VELOC   30000
1:
      PMOVE   200000, INCR, LINEAR
      DWELL   2000
      JUMP    UNCOND, 1
```



Obrázek 7-10. Nepodmíněný skok

Podmíněné skoky

Podmíněný skok je povel JUMP s bitem CTL zadaným v povelu. Podmíněné skoky jsou povely typu 1 v tom, že mohou ovlivňovat cestu vykonávání programu, ale jsou také podobné povelům typu 2, protože se nevykonají, dokud se nevykoná povel typu 3 následující po povelu JUMP. Když se vykoná podmíněný povel JUMP, DSM314 zjistí stav zadaného bitu CTL. Pokud bit bude v jedničce, vykonávání programu skočí na číslo cílového bloku; pokud bit bude v nule, vykonávání programu bude pokračovat povelem za povelem JUMP. Všimněte si, že povel typu 3 za podmíněným skokem a cíl skoku ovlivní chování skoku.

Povely podmíněného skoku se nesmí používat v programech pro více os obsahujících bloky SYNC, až na stav, kdy JUMP bude aktivovaný na základě testování stejného povelu JUMP oběma osami. Pokud byste nedodrželi toto doporučení, může nastat nepředvídatelná činnost.

Testování podmíněného skoku **začne**, když další povel PMOVE, CMOVE, DWELL nebo WAIT následující po podmíněném JUMP bude aktivní.

Když testování podmíněného skoku bude aktivní, určený bit CTL se testuje s rychlostí aktualizace polohové smyčky (0.5, 1.0 nebo 2.0 milisekund v závislosti na konfiguraci).

Testování podmíněného skoku **skončí**, když zadaný bit CTL přejde do jedničky (dojde k aktivaci skoku) nebo když bude aktivní nové číslo bloku.

Pokud bude naprogramovaný více než jeden Podmíněný skok bez vloženého povelu PMOVE, CMOVE, DWELL nebo WAIT, provede se pouze poslední Podmíněný skok.

Podmíněný skok nelze použít jako poslední řádek podprogramu (nebo na řádku před nepodmíněným skokem na konci podprogramu), protože testování skoku se skončí, když se zpracuje povel Konec podprogramu.

V podstatě Podmíněný skok přenese řízení na nový programový blok na základě přechodu externích vstupních bitů CTL do jedničky. Testování stavu bitů CTL je možno provádět jednou nebo souvisle během následujícího povelu typu 3, pokud je ve stejném programovém bloku. Vícenásobné podmíněné skoky nelze ve stejném programovém bloku použít (následující příklad ukazuje toto nesprávné použití povelu Podmíněný skok).

Příklad podmíněného skoku 1:

```
PROGRAM 1 MULTI-AXIS
      VELOC    AXIS1 10000
      ACCEL    AXIS1 10000
1:    JUMP     CTL01, 2                  //Tento povel JUMP se bude ignorovat
      JUMP     CTL02, 3                  //Tento povel JUMP se provede
      CMOVE    AXIS1 +40000, INCR, LINEAR
2:    CMOVE    AXIS2 +20000, INCR, LINEAR
3:    PMOVE    AXIS2 +100000, ABS, LINEAR
4:    DWELL    AXIS2 100
ENDPROG
```

První JUMP není naprogramovaný správně, protože (1) za ním nenásleduje vložený povel typu 3, a (2) je ve stejném bloku jako jiný povel JUMP. Když nové číslo bloku bude aktivní PO povelu podmíněný JUMP, testování skoku se provede jednou na konci.

Příklad podmíněného skoku 2:

```
PROGRAM 2 AXIS1
      VELOC    10000
      ACCEL    10000
1:    CMOVE    20000, ABS, LINEAR
      JUMP     CTL01, 3
2:    PMOVE    40000, ABS, LINEAR        //CTL01 testováno pouze jednou
3:    DWELL    100
ENDPROG
```

Ve výše uvedeném příkladu se test bitu *CTL01* provede jen jednou, protože PMOVE následující za JUMP obsahuje nové číslo bloku (2). Avšak změna umístění bloku číslo 2 způsobí testování bitu CTL během PMOVE následujícím po povelu JUMP, jak je vidět v následujícím příkladu:

Příklad podmíněného skoku 3:

```
PROGRAM 3 AXIS1
      VELOC    10000
      ACCEL    10000
1:    CMOVE    20000, ABS, LINEAR
2:    JUMP     CTL01, 3
      PMOVE    40000, ABS, LINEAR        //CTL01 testováno během PMOVE
3:    DWELL    100
ENDPROG
```

V tomto příkladu se bit CTL01 testuje během PMOVE, protože povely PMOVE a JUMP jsou ve stejném bloku.

DSM314 může vykonat Podmíněný skok z aktivního CMOVE na programový blok obsahující CMOVE nebo PMOVE bez zastavení. **Aby osa mohla provést skok bez zastavení, vzdálenost představovaná povelem CMOVE nebo PMOVE v bloku skoku musí být větší než vzdálenost pro zastavení serva.** Vzdálenost pro zastavení serva se vypočítá s použitím současně zadaných parametrů rychlosti a zrychlení, které by byly v platnosti, když blok skoku byl aktivní.

Osa se ZASTAVÍ před skokem, když se aktivace podmíněného skoku objeví za některé z následujících podmínek:

- Když je aktivní PMOVE
- Když je aktivní CMOVE a cílový blok skoku obsahuje CMOVE nebo PMOVE představující pohyb v opačném směru.
- Když je aktivní CMOVE a cílový blok skoku obsahuje CMOVE nebo PMOVE představující pohyb ve stejném směru s dostatečnou vzdáleností pro zastavení osy.
- Když je aktivní CMOVE a cílový blok skoku obsahuje povel DWELL, WAIT nebo END (program).

Pokud osa ZASTAVÍ před podmíněným skokem, použije se aktuální naprogramované zrychlení a režim zrychlení.

Nepodmíněné skoky nepřinutí osu zastavit před tím, než se provede skok na nový programový blok. Například povel CMOVE následovaný nepodmíněným skokem na jiný CMOVE se bude chovat, jako by se tyto dva povely CMOVE vyskytly bez vloženého nepodmíněného skoku.

Pokud testování podmíněného skoku bude aktivní, když procesor povelů DSM314 zjistí povel CALL SUBROUTINE, osa se **zastaví** a ukončí testování skoku před vykonáním povelu CALL.

Pokud testování podmíněného skoku bude aktivní, když procesor povelů DSM314 zjistí povel END SUBROUTINE, osa se **zastaví** a ukončí testování skoku před vykonáním povelu END SUBROUTINE.

Testování skoku

Podmíněné skoky provádějí testování skoku. Pokud bit CTL bude v jedničce, skok se provede okamžitě. Pokud bit CTL bude v nule, DSM314 bude sledovat bit CTL a udržuje cestu k cíli povelu JUMP. Toto sledování bitu CTL se nazývá testování. Pokud se během testování stav bitu CTL změní do jedničky před povelem BLOCK, zjistí se jiný povel JUMP nebo jiný povel CALL, skok se provede. Tyto povely ukončí testování skoku.

Příklad 7: Testování skoku

Uvažujme následující dva příklady programů pro jednu osu. V příkladu 1 se posuv na polohu 2000 dokončí před začátkem testování skoku. Číslo bloku, které se objeví hned po povelu JUMP, ukončí testování skoku. Proto doba, po kterou se bude monitorovat bit CTL, bude velmi krátká. Avšak v příkladu 2 se povel JUMP zjistil před povelem CMOVE. Tím se testování skoku spustí před začátkem pohybu a testování bude pokračovat, dokud bude trvat pohyb. Pokud bit CTL bude v jedničce, když se vykonává pohyb, skok se provede. Po dokončení pohybu se najde další číslo bloku, který skončí testování skoku a vykonávání programu bude pokračovat normálně. Pokud před číslem dalšího bloku byly naprogramované ještě jiné pohyby, testování skoku by pokračovalo

během těchto pohybů, dokud se nezjistí další číslo bloku.

Příklad 1

```
     ACCEL   5000
     VELOC   1000
1:
     CMOVE  2000, ABS, LINEAR
     JUMP     CTL01, 3
2:
```

Příklad 2

```
     ACCEL   5000
     VELOC   1000
1:
     JUMP      CTL01, 3
     CMOVE   2000, ABS, LINEAR
2:
```

Normální zastavení před JUMP

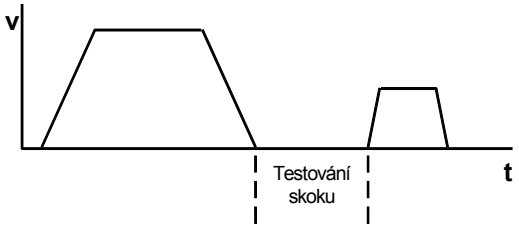
Povel podmíněného skoku je podobný povelům typu 2 v tom, že testování skoku nezačne, dokud se nevykoná povel typu 3 následující hned po povelu JUMP. Pokud by tento povel typu 3 normálně zastavil pohyb, pak se pohyb zastaví před tím, než začne testování skoku. Povely typu 3, které zastaví pohyb, jsou: DWELL, WAIT, ENDPROG a pohyby v opačném směru.

Proto i když bit CTL bude v jedničce před tím, než se vykoná povel s podmíněným skokem a povelem typu 3, pohyb osy se zastaví před tím, než vykonávání programu bude pokračovat na cíl skoku. Toto zastavení NENÍ zastavení skoku, které je popisováno v příkladu 10.

Příklad 8: Normální zastavení před JUMP

Následující příklad obsahuje skok následovaný povelem DWELL. Protože DSM314 provádí zpracování dopředu, ví, že musí zastavit po povelu CMOVE. Proto se zastaví před vykonáním povelu DWELL. Protože testování skoku nezačne, dokud se nevykoná povel DWELL, testování začne po zastavení pohybu. Testování skoku skončí, když se spustí následující CMOVE v důsledku souvisejícího povelu BLOCK. Přerušovaná čára v profilu rychlosti indikuje, kdy se provádí testování skoku. V tomto příkladu bit CTL03 nepřejde do jedničky během vykonávání programu.

```
1:   ACCEL   5000
     VELOC   10000
     CMOVE   60000, INCR, LINEAR
2:   JUMP     CTL03, 4
     DWELL   4000
3:   ACCEL   10000
     VELOC   5000
     CMOVE   15000, INCR, LINEAR
4:
```



Obrázek 7-11. Normální zastavení před JUMP

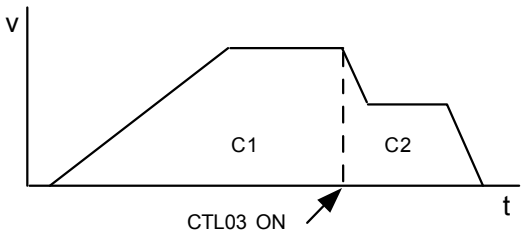
Skok bez zastavení

Pokud povel typu 3, který následuje za podmíněným skokem, bude CMOVE a povel typu 3 v místě cíle bude povel pohybu s dostatečnou vzdáleností k tomu aby se při dokončení mohlo úplně zpomalit až na nulu, skok se provede bez zastavení. To je jediný způsob, jak udržet pohyb, když se vykonává skok.

Příklad 9: JUMP bez zastavení

Toto je jednoduchý příklad podmíněného skoku z jednoho CMOVE na druhý. Když skok testuje bit *CTL03*, první CMOVE zrychlí na naprogramovanou rychlost. Před přerušovanou čárou bit *CTL03* je v nule, ale na přerušované čáře bit *CTL03* přejde do jedničky. Vykonávání programu se okamžitě přenese na blok 3 a CMOVE se spustí. Protože rychlost v místě skoku je odlišná, rychlost se změní naprogramovaným zrychlením cílového bloku skoku. Nakonec, když se dokončí druhý CMOVE, rychlost se sníží na nulu a program se skončí.

```
1:  ACCEL   2000
    VELOC   10000
    JUMP    CTL03, 3
    CMOVE   120000, INCR, LINEAR
3:  ACCEL   20000
    VELOC   5000
    CMOVE   15000, INCR, LINEAR
```



Obrázek 7-12. JUMP bez zastavení

Zastavení vyvolané skokem

Zastavení vyvolané skokem je zastavení, které je způsobeno skokem. **Když se vyskytne zastavení vyvolané skokem, použije se aktuální naprogramované zrychlení a režim zrychlení.** Všimněte si, že pohyb po S křivce dosáhne konstantní rychlosti před tím, než se začne zpomalovat. Více podrobností najdete v příkladech skoku s S křivkou. Existují dva způsoby generování zastavení vyvolané skokem, které jsou popsané níže.

Podmíněný JUMP aktivovaný během PMOVE bude vždy generovat zastavení vyvolané skokem. Protože PMOVE vždy zastaví před tím, než bude pokračovat následující pohyb, zastavení vyvolané skokem se objeví vždy, když se během PMOVE vykoná skok.

Když se však vyskytne aktivace podmíněného skoku během CMOVE, zastavení vyvolané skokem se neobjeví, pokud pohyb naprogramovaný v cíli skoku bude PMOVE nebo CMOVE, které budou mít dostatek vzdálenosti ve stejném směru. Zastavení vyvolané skokem se objeví, když PMOVE nebo CMOVE v cíli skoku nebudou mít dostatečnou vzdálenost nebo budou představovat pohyb v opačném směru.

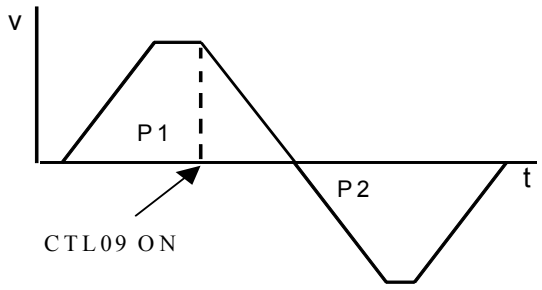
U pohybu po S křivce zastavení vyvolané skokem provede jednu z následujících dvou věcí: Pokud se skok provede po prostředním bodu zrychlování nebo zpomalování, toto zrychlování nebo zpomalování se dokončí před vyvoláním skoku. Pokud se skok objeví před prostředním bodem zrychlování nebo zpomalování, profil se okamžitě začne stabilizovat. Jakmile zrychlování nebo zpomalování bude na nule, začne se vykonávat zastavení vyvolané skokem. Viz příklady skoku s S křivkou.

Příklad 10: Zastavení vyvolané skokem

Toto je příklad podmíněného skoku se zastavením vyvolaným skokem. Vylepšením příkladu 5, DWELL, by bylo sledovat externí bit CTL, který by indikoval problém s pohybem v kladném směru. Pokud se bit CTL nikdy nenastaví do jedničky, profil pro následující program bude shodný s profilem ukázaným v příkladu pro DWELL. Pokud bit CTL přejde do jedničky během prvního PMOVE nebo DWELL, okamžitě by začal pohyb v opačném směru.

Následující profil by se objevil, pokud bit CTL přejde do jedničky během prvního PMOVE v místě přerušované čáry. Protože první pohyb se dokončil předčasně v důsledku toho, že bit CTL přešel do jedničky, druhý pohyb by se nemusel pohybovat tak daleko, aby se dostal zpět na polohu 0, tak jak tomu bylo v příkladu DWELL. Všimněte si, že vzhledem k tomu, že naprogramovaný pohyb v cíli skoku je v opačném směru než výchozí pohyb, profil by byl stejný, pokud by pohyby byly povely CMOVE a ne PMOVE.

```
     ACCEL    30000
     VELOC    15000
1:   JUMP     CTL09, 2
     PMOVE    120000, ABS, LINEAR
     DWELL    4000
2:   PMOVE    0, ABS, LINEAR
```

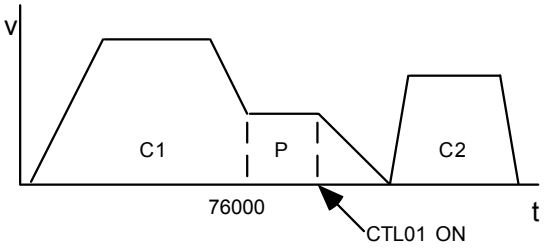


Obrázek 7-13. Zastavení vyvolané skokem

Příklad 11: Skok následovaný povelem PMOVE

V tomto příkladu povelu JUMP je za povelem JUMP povel PMOVE ve stejném směru. Níže uvedený profil rychlosti ukazuje zrychlení a pohyb pro první CMOVE a zpomalení na rychlost PMOVE. Bit *CTL01*, který je v nule při začátku PMOVE, přejde do jedničky v místě druhé přerušované čáry. Pohyb se zastaví po PMOVE, i když podmíněný skok přejde na jiný blok. Proto bit *CTL01* aktivuje zpomalení na nulu před začátkem posledního CMOVE.

```
1:  ACCEL    2000
    VELOC    8000
    CMOVE    76000, INCR, LINEAR
2:  ACCEL    1000
    VELOC    4000
    JUMP     CTL01, 3
    PMOVE    50000, INCR, LINEAR
3:  ACCEL    6000
    VELOC    6000
    CMOVE    6000, INCR, LINEAR
```



Obrázek 7-14. Skok následovaný povelem PMOVE

Skoky podle S křivky

Skoky během lineárního pohybu a skoky během pohybu s konstantní rychlostí podle S křivky začnou okamžitě zrychlovat nebo zpomalovat na novou rychlost. Skoky během zrychlování nebo zpomalování podle S křivky však vyžadují jiná pravidla, aby se udržel profil S křivky. Co se stane, když se vyskytne skok během pohybu podle S křivky a mění se rychlost, závisí na tom, jestli se skok objeví před nebo po prostředním bodu (bod, kde velikost zrychlování je největší) a jestli rychlost v cíli skoku je větší nebo menší než aktuální rychlost.

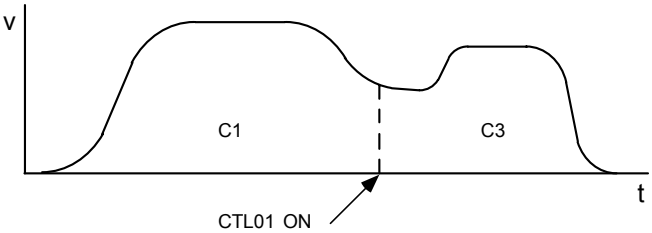
Skoky podle S křivky po prostředním bodu zrychlování nebo zpomalování

Pokud ke skoku dojde po prostředním bodu změny rychlosti, změna bude pokračovat normálně až do dosažení konstantní rychlosti; pak se rychlost změní na novou rychlost s použitím režimu zrychlení pohybu v cíli skoku.

Příklad 12: Skoky podle S křivky po prostředním bodu zrychlování nebo zpomalování

V následujícím příkladu ke skoku dojde během poslední fáze zpomalování v okamžiku přerušované čáry. Zpomalování bude pokračovat až do dosažení konstantní rychlosti a pak se začne zrychlovat na vyšší rychlost.

```
    ACCEL   50000
    VELOC   100000
1:  JUMP    CTL01, 3
    CMOVE   500000, ABS, S-CURVE
2:  VELOC   60000
    CMOVE   500000, INCR, S-CURVE
3:  VELOC   85000
    ACCEL   100000
    CMOVE   250000, INCR, S-CURVE
```



Obrázek 7-15. Skoky podle S křivky po prostředním bodu zrychlování nebo zpomalování

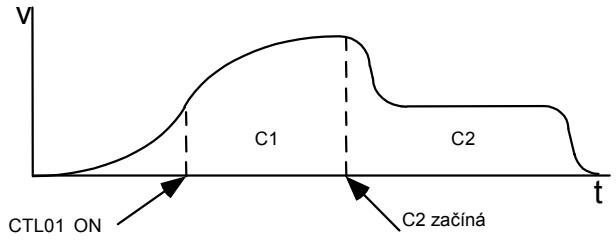
Skoky podle S křivky před prostředním bodem zrychlování nebo zpomalování

Pokud se skok provede před prostředním bodem zrychlování nebo zpomalování, výsledek bude záviset na tom, jestli rychlost v cíli skoku bude vyšší nebo nižší než rychlost před provedením skoku. V prvním případě, když se zrychluje, ale nová rychlost je nižší, nebo se zpomaluje a nová rychlost je větší, DSM314 okamžitě začne zmenšovat zrychlování nebo zpomalování na nulu. Jakmile se dosáhne nulové rychlosti, DSM314 použije zrychlení a rychlost cíle skoku a provede změnu na novou rychlost.

Příklad 13: Skok podle S křivky před prostředním bodem zrychlování nebo zpomalování

V následujícím příkladu se provede skok během zrychlování prvního CMOVE v okamžiku první přerušované čáry. Protože rychlost v cíli skoku je nižší než rychlost prvního CMOVE, DSM314 zmenší zrychlování na nulu. Konstantní rychlost, nulové zrychlení se objeví v okamžiku druhé přerušované čáry. Proto DSM314 začne zpomalovat na novou rychlost s použitím zrychlení v cíli skoku. Nakonec se dokončí druhý CMOVE.

```
ACCEL   1000
    VELOC   50000
1:  JUMP    CTL01, 3
    CMOVE   50000, INCR, S-CURVE
3:  VELOC   5000
    ACCEL    10000
    CMOVE   15000, INCR, S-CURVE
```



Obrázek 7-16. Skok před prostředním bodem zrychlování nebo zpomalování

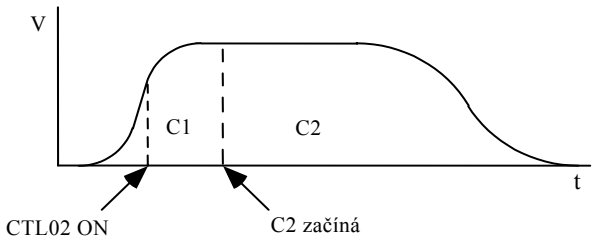
Skoky podle S křivky na vyšší zrychlení, když probíhá zrychlování, nebo na nižší zpomalení, když probíhá zpomalování

Druhý případ vyvolá skok na vyšší rychlost, když probíhá zrychlování, nebo na nižší rychlost, když probíhá zpomalování. Když k tomuto dojde, DSM314 bude pokračovat zrychlením nebo zpomalením prvního pohybu. Toto zrychlení nebo zpomalení se bude udržovat, obdobně jako u lineárního zrychlení, dokud osa nedosáhne nové rychlosti. Pak se použije normální S křivka ke snížení zrychlení nebo zpomalení na nulu.

Příklad 14: Skok podle S křivky na vyšší rychlost, když probíhá zrychlování, nebo na nižší rychlost, když probíhá zpomalování

V tomto příkladu se povel JUMP aktivuje během počáteční fáze zrychlování (v okamžiku první přerušované čáry) a rychlost v cíli skoku je větší než rychlost aktuálního pohybu. První přerušovaná čára indikuje maximální zrychlení prvního CMOVE. Tato hodnota zůstává během zrychlování osy, dokud se podle S křivky nevrátí zpět na konstantní rychlost. Konstantní rychlost, druhá přerušovaná čára, indikuje začátek druhého CMOVE. Tento pohyb bude pokračovat, dokud se na konci programu nezpomalí na nulu.

```
     ACCEL   50000
     VELOC   30000
1:   JUMP    CTL02, 2
     CMOVE   150000, INCR, S-CURVE
2:   VELOC   90000
     ACCEL   25000
     CMOVE   500000, INCR, S-
             CURVE
```



Obrázek 7-17. Skok na vyšší zrychlení, když probíhá zrychlování, nebo na nižší zpomalení, když probíhá zpomalování

Další poznámky k programovanému pohybu

Následující příklady nejsou kompletní programy. Například v mnoha případech nejsou ukázané příkazy PROGRAM *a ENDPROG. Aby se program úspěšně zkompiloval, bylo by nutné doplnit tyto příkazy (ve správném kontextu).*

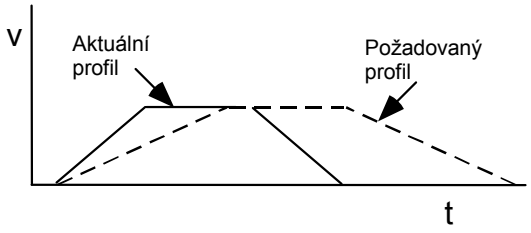
Maximální doba zrychlení

Maximální doba pro naprogramované zrychlení nebo zpomalení je 131 sekund. Pokud vypočítaná doba pro zrychlení nebo zpomalení bude delší než tato doba, DSM314 spočítá zrychlení, které se použije během 131 sekund. Aby se získaly delší časy zrychlení, je nutno použít několik povelů CMOVE se zvyšujícími se nebo snižujícími se rychlostmi.

Příklad 15: Maximální doba zrychlení

Následující dva příklady programu platí pouze pro DSM314 používající dobu aktualizace polohové smyčky 2 ms. Ukazují hypotetický problém s velmi dlouhou dobou zrychlování v příkladu 1 a možné řešení v příkladu 2. V následujícím příkladu 1 se vyžaduje 240 sekund k dosažení naprogramované rychlosti 24 000 zrychlením 100 (24 000 ÷ 100 = 240). Protože toto je větší než limit DSM 131 sekund na zrychlování nebo zpomalování, DSM spočítá hodnotu, která bude v tomto rozmezí. V tomto případě DSM spočítá, že k dosažení rychlosti 24 000 během 131 sekund je nutné použít zrychlení 183. V příkladu 1 profil rychlosti znázorněný plnou čárou ukazuje vyšší (183) zrychlení, které DSM použije. Profil znázorněný přerušovanou čárou v tomto grafu ukazuje požadované (naprogramované) zrychlení a profil rychlosti, který nelze dosáhnout.

```
ACCEL   100
VELOC   24000
PMOVE  8000000, INCR, LINEAR
```



Obrázek 7-18. Příklad 1 s maximální dobou zrychlování

Jedním řešením (které vyžaduje určité výpočty navíc) k získání nižšího zrychlení během delší doby, rozloží pohyb na samostatné souvislé pohyby (pomocí povelů CMOVE), kdy doba zrychlování pro jednotlivé pohyby bude menší než 131 sekund. U problému uvedeného v příkladu 1 by každý naprogramovaný pohyb pro zrychlování a zpomalování vyžadoval 240 sekund. Rozdělením této doby na polovinu s použitím dvou pohybů s dobou zrychlování nebo zpomalování 120 sekund se pohyby dostanou do 131-sekundového rozmezí DSM. Toto schéma se používá v následujícím příkladu.

Příklad 2 ukazuje, jak je možno výsledku požadovaného v příkladu 1 dosáhnout náhradou jednoho pohybu v příkladu 1 za čtyři pohyby. Požadují se čtyři pohyby, protože úseky profilu se zrychlením i zpomalením se musí rozdělit na dva pohyby. Má-li se rozdělit celková doba

zrychlování (nebo zpomalování) na polovinu, vypočtěte vzdálenost v prostředním bodu obou náběhů, když rychlost je 12000, tak aby to odpovídalo 720 000 uživatelským jednotkám.

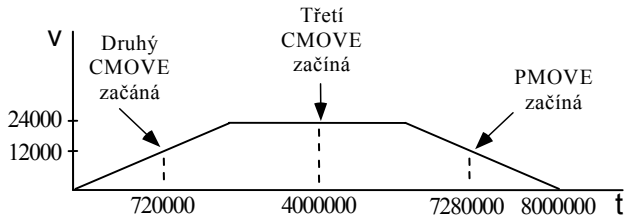
Vzdálenost ujetá během zrychlování nebo zpomalování se vypočítá podle následujícího vztahu:

$$\text{Ujetá vzdálenost} = \frac{\text{Změna rychlosti} \times \text{požadovaná doba}}{2}$$

$$720\,000 = \frac{12\,000 \times 120}{2}$$

(Protože k dosažení rychlosti 24 000 je nutných 240 sekund, rychlost 12 000 je možno dosáhnout během 120 sekund.) Tuto vzdálenost používají počáteční CMOVE a konečný PMOVE. Druhý CMOVE "převezme řízení" v prostředním bodu náběhu zrychlování od prvního CMOVE a bude zrychlovat na cílovou rychlost 24 000. Třetí CMOVE se vyžaduje pro rozdělení zpomalovací části profilu. Poslední pohyb PMOVE "převezme řízení" od třetího CMOVE ve vzdálenosti prostředního bodu zpomalování (720 000 uživatelských jednotek od konečné polohy). Jak se třetí CMOVE blíží konečné poloze, automaticky začne zpomalovat na rychlost 12 000 povelu PMOVE. Přerušovaná čára v grafu příkladu 2 odděluje tyto čtyři pohyby. Mají-li se vypočítat vzdálenosti druhého a třetího CMOVE, odečtěte vzdálenosti vypočítané pro první CMOVE a poslední PMOVE (každý 720 000 pro celkových 1 440 000) od konečné vzdálenosti 8 000 000. Tím se získá zbývající vzdálenost 6 560 000, která se rozdělí rovnoměrně mezi druhý a třetí CMOVE (3 280 000 každý).

```
ACCEL   100
VELOC   12000
CMOVE   720000, INCR, LINEAR
VELOC   12000
CMOVE   3280000, INCR, LINEAR
VELOC   24000
CMOVE   3280000, INCR, LINEAR
VELOC   12000
PMOVE   720000, INCR, LINEAR
```



Obrázek 7-19. Příklad 2 s maximální dobou zrychlování

Zastavení posuvu s DSM314

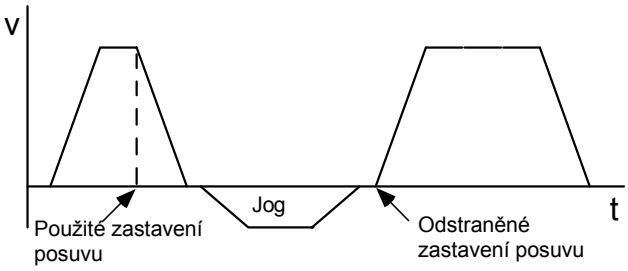
Zastavení posuvu se používá k přechodnému zastavení vykonávání programu bez jeho ukončení, často k přezkoumání některých aspektů systému. Způsobí zastavení pohybu všech os naprogramovaným zpomalením. Když se zastavení posuvu ukončí, vykonávání programu bude pokračovat. Přerušený pohyb bude pokračovat naprogramovaným zrychlením a rychlostí.

Zastavení posuvu se provede nastavením bitu %Q *Zastavení posuvu* do jedničky a bude trvat, dokud bit %Q nepřejde do nuly. Přechod bitu %Q *Zrušit všechny pohyby* do jedničky nebo chyba, která by normálně způsobila chybu vyvolávající zastavení, ukončí zastavení posuvu a ukončí program. Během zastavení posuvu, je přípustné jogování v kladném nebo záporném směru, ale žádný jiný pohyb. Když zastavení posuvu bude ukončeno a vykonávání programu se obnoví, DSM314 si pamatuje a provede posuv do předchozího místa.

Příklad 16: Zastavení posuvu

Následující příklad znázorňuje profil pohybu, když se použije zastavení posuvu. Lineární pohyb se zrychlí na naprogramovanou rychlost naprogramovaným zrychlením. Zastavení posuvu se přivede v okamžiku přerušované čáry, takže rychlost se sníží naprogramovaným zrychlením na nulu. Pak se provede Jog s použitím bitu %Q *Jog Minus*. To je zřejmé, protože rychlost jogu je záporná. Všimněte si, že zrychlení použité během jogu je aktuální ***Zrychlení jogu***, které je odlišné od naprogramovaného zrychlení. Všimněte si také, že povel bitu %Q *Zastavení posuvu* musí být přiváděný během celého trvání jogu. Po skončení pohybu jogu se zastavení posuvu ukončí a program bude pokračovat až do konce.

```
ACCEL 1000
VELOC 2000
PMOVE 12000, INCR, LINEAR
```



Obrázek 7-20. Příklad zastavení posuvu

Přepis rychlosti posuvu

Některé aplikace vyžadují menší úpravy naprogramované rychlosti, aby bylo možno provést vnější změny. Okamžitý povel %AQ *Přepis rychlosti*, který se posílá do DSM přes žebříkovou logiku, umožňuje provést změny naprogramované rychlosti posuvu (rychlosti) během vykonávání programu. (Podrobnosti o povelu Přepis rychlosti najdete v kapitole 5.) Když se program začne vykonávat, přepis rychlosti je nastavený na 100%. Proto změny rychlosti posuvu před nastavením bitu pro vykonání programu do jedničky se budou ignorovat. Avšak přepis rychlosti zadaný ve stejném cyklu jako bit pro vykonání programu *bude* platný.

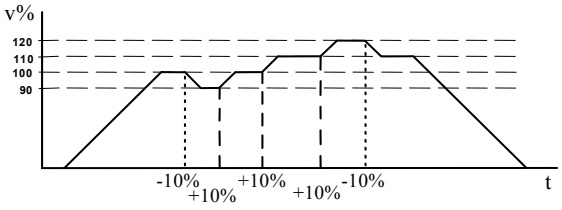
Procenta přepisu rychlosti posuvu je možno nastavit v rozmezí 0% až 120%. Když bude zadaný *Přepis rychlosti*, DSM314 interně vynásobí procento rychlosti posuvu naprogramovanou rychlostí a tak získá novou rychlost. Pokud se osa bude pohybovat, ke změně rychlosti na novou rychlost se použije ***Režim zrychlení jogu*** aktuálního pohybu. Změna rychlosti posuvu bude mít vliv na všechny budoucí rychlosti pohybu. Všimněte si, že když se použije rychlost posuvu 0%, nebude se generovat žádný pohyb, dokud se nezadá nová rychlost posuvu. Také se všimněte, že když rychlost posuvu bude 0%, bit %I *Pohyb* zůstane v jedničce.

Přepis rychlost nemá žádný vliv na neprogramový pohyb, například *Jog, Nalezení výchozí polohy,* nebo *Pohyb rychlostí.*

Příklad 17: Přepis rychlosti posuvu

Během vykonávání tohoto programu jsou zadávané změny rychlosti posuvu + nebo -10%. Tečkovaná čára označuje -10%, přerušovaná čára označuje +10%.

```
ACCEL     1000
VELOC     6000
PMOVE     110000, INCR,
LINEAR
```



Obrázek 7-21. Příklad přepisu rychlosti posuvu

Programování pro více os

Synchronizační bloky je možno použít v programu pro více os k synchronizaci povelů pro pohyb osy v místech, kde je kritické časování.

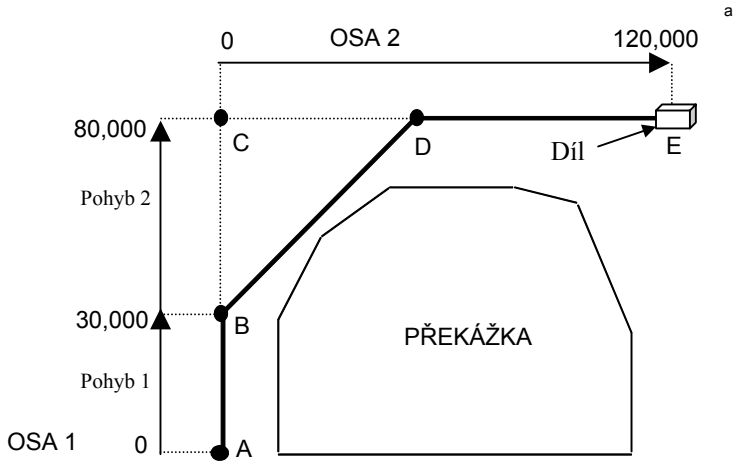
Příklad 18: Programování pro více os

Tento příklad předpokládá, že osa 1 řídí vertikální pohyb a osa 2 řídí horizontální pohyb. Cílem je přemístit kus materiálu z bodu A do bodu E co nejrychleji a při tom se vyhnout překážce, která brání přímému pohybu mezi těmito body.

Jednoduchý způsob by bylo se přemístit přímo z bodu A do bodu C a pak z bodu C do bodu E. Tento postup je však časově ztrátový. Lepší bude začít horizontálním pohybem před dosažením bodu C. Bylo zjištěno, že když se osa 1 přemístí do polohy 30 000 uživatelských jednotek (do bodu B), osa 2 se může začít pohybovat a při tom se vyhnout překážce. Programový segment je možno naprogramovat následovně:

```
10:   CMOVE   AXIS1 30000, INCR, LINEAR
20:   SYNC
      PMOVE   AXIS1 50000, INCR, LINEAR
      PMOVE   AXIS2 120000, INCR, LINEAR
```

Když se vykoná blok 10, osa 1 se začne pohybovat v poloze 30 000 jednotek, osa 2 bude stát. Když osa 1 dokončí pohyb, stanou se dvě věci: osa 1 se začne pohybovat na poloze 50 000 jednotek zadaným povelem PMOVE v bloku 20 (SYNC) bez zastavení (protože první pohyb byl CMOVE) a osa 2 se začne pohybovat v poloze 120 000 jednotek. Na následujícím obrázku osa 1 nejdříve přenese díl z bodu A do bodu B. V bodu B osa 1 bude pokračovat v pohybu (bude vykonávat druhý pohyb) a osa 2 se začne pohybovat a přenese díl do bodu 2. Osa 1 dokončí svůj druhý pohyb v bodě D a zastaví se; avšak osa 2 bude pokračovat a přemístí díl do bodu E.



Obrázek 7-22. Příklad programování pro více os

Pokud tento programový segment nebude na začátku programu a z nějakého důvodu osa 2 ještě nedosáhne bloku 20, když osa 1 vykonala pohyb o 30 000 jednotek, objeví se chyba. Osa 1 by pokračovala na 80 000 jednotek a DSM314 by ve stavovém kódu hlásilo“Chyba synchronizačního bloku během CMOVE”.

Je nezbytné, aby osy byly v bloku 20 synchronní, změna bloku 10 na PMOVE by synchronizaci zaručila, ale osa 1 by pak zastavila na 30 000 jednotkách.

Parametry (P0-P255) v DSM314

DSM314 udržuje v paměti 256 parametrů s dvojitou délkou slova (0 až 255). Tyto parametry je možno použít jako proměnné v pohybových povelech ACCEL, VELOC, DWELL, PMOVE a CMOVE. Dejte pozor na to, že stále platí meze rozsahu a pokud parametr bude obsahovat hodnotu mimo rozsah, může se objevit chyba. Parametry 216 - 255 jsou parametry se speciálním účelem. Některé parametry se speciálním účelem se automaticky zapisují do DSM314. Například P224 se automaticky aktualizuje, když se vyskytne Vzorkování polohy 1 na ose 1. Následující tabulka popisuje funkci parametrů se speciálním účelem.

Tabulka 7-1. Parametry se speciálním účelem

Číslo parametru	Funkce speciálního účelu	Osa	Jednotky
216-223	*Vyhrazeno*		
224	*Vzorkování polohy 1*	Osa 1	Uživatelské jednotky
225	*Vzorkování polohy 2*	Osa 1	Uživatelské jednotky
226	*Zadaná poloha* při aktivaci povolení vlečené osy	Osa 1	Uživatelské jednotky
227	Inkrementální vzdálenost zastavení vlečené osy	Osa 1	Uživatelské jednotky
228-231	*Vyhrazeno*		
232	*Vzorkování polohy 1*	Osa 2	Uživatelské jednotky
233	*Vzorkování polohy 2*	Osa 2	Uživatelské jednotky
234	*Zadaná poloha* při aktivaci povolení vlečené osy	Osa 2	Uživatelské jednotky
235	Inkrementální vzdálenost zastavení vlečené osy	Osa 2	Uživatelské jednotky
236-239	*Vyhrazeno*		
240	*Vzorkování polohy 1*	Osa 3	Uživatelské jednotky
241	*Vzorkování polohy 2*	Osa 3	Uživatelské jednotky
242	*Zadaná poloha* při aktivaci povolení vlečené osy	Osa 3	Uživatelské jednotky
243	Inkrementální vzdálenost zastavení vlečené osy	Osa 3	Uživatelské jednotky
244-247	*Vyhrazeno*		
248	*Vzorkování polohy 1*	Osa 4	Uživatelské jednotky
249	*Vzorkování polohy 2*	Osa 4	Uživatelské jednotky
250	*Zadaná poloha* při aktivaci povolení vlečené osy	Osa 4	Uživatelské jednotky
251	Inkrementální vzdálenost zastavení vlečené osy	Osa 4	Uživatelské jednotky
252-255	*Vyhrazeno*		

Po zapnutí napájení nebo po uložení konfigurace DSM314 do PLC se všechny parametry resetují na nulu. Parametry je možno přiřadit třemi způsoby:

- Pohybovým povelem programu LOAD.

- Povelem %AQ *Okamžité načtení parametru*.
- Funkčním blokem COMM_REQ. To je nejlepší způsob, pokud potřebujete poslat několik parametrů během jednoho čtení. Funkční blok COMM_REQ je popsaný v Dodatku B.

Přiřazením hodnoty parametru se přepíšou všechny předchozí hodnoty. Hodnoty parametru je možno měnit během vykonávání programu, ale změna se musí provést před tím, než DSM314 začne vykonávat povel typu 3 (Pohyb, Čekání nebo Prodleva), který předchází povelu typu 3 používající tento parametr. Je to v důsledku zpracování povelů typu 3 dopředu, které DSM314 provádí. Všimněte si, že povel JUMP vynuluje zpracování dopředu a vynutí zpracování povelů programu v cíli skoku.

Níže je uvedený příklad pohybového programu používající parametry. Hodnoty parametrů 1 se načtou předem pomocí povelu COMREQ z PLC alespoň dva programové bloky před použitím. (Pamatujte, že "programové bloky" nejsou stejné jako sekce pohybového programu, které jsou označené povelem BLOCK #.)

Blok/Povel/Data		**Komentáře**
1:	VELOC P001	// Nastavení rychlosti prvního pohybu = hodnota v Param. 1
	ACCEL P002	// Nastavení zrychlení prvního pohybu = hodnota v Param. 2
	CMOVE P003, ABS, LINEAR	// Dosažení rychlosti 2. pohybu (20000) v poloze = Par. 3
2:	VELOC 20000	// Nastavení rychlosti druhého pohybu = 20000
	PMOVE 20000, INCR, LINEAR	// Normální PMOVE
	DWELL P004	// Prodleva po dobu parametru 4
	PMOVE P005, INCR, LINEAR	// PMOVE na hodnotu v Parametru 5
(* Vzorkování #1 se provede na ose 1 během pohybu do polohy Param. 5*)		
	DWELL 1000	// Prodleva na jednu sekundu
	LOAD P006,2000	// Načtení Parametru 6
3:	PMOVE P224, INCR, LINEAR	// Pohyb do vzorkované polohy pro vzorkování #1 na ose 1
	DWELL 2000	// Prodleva na dvě sekundy
	PMOVE P006, ABS, S-CURVE	// Zastavení v konečné poloze = hodnota v Parametru 6

Výpočet hodnot zrychlení, rychlosti a polohy

Jedním ze způsobů určení hodnoty pro proměnné pohybového programu APM nebo DSM například *Zrychlení*, *Rychlost* nebo *Poloha* je vynést do grafu požadovaný pohyb nebo segment pohybu jako profil rychlosti. Profil rychlosti má na vodorovné ose grafu čas a na svislé ose rychlost. Klíčem k porozumění tvorby profilu je rozložit kompletní pohyb na menší segmenty, které je možno analyzovat geometricky. Většina aplikací bude používat ekonomický lichoběžníkový pohyb, jehož profil rychlosti je znázorněný níže. Má-li se pohyb provést co nejrychleji a pokud servo má dostatečný rozsah rychlosti, použijte trojúhelníkový profil rychlosti. Trojúhelníkový pohyb se bude zrychlovat polovinu vzdálenosti a pak druhou polovinu zpomalovat. Jinou možností je použít lichoběžníkový profil s kratším ustáleným segmentem.

Kinematické rovnice

Kinematika je odvětví mechaniky, které studuje pohyb tělesa nebo systémů těles bez zvažování jejich hmotnosti nebo sil, které na ně působí. Následující tabulka obsahuje převody základních lineárních rovnic použitých pro úsek zrychlování profilu pohybu. Tyto vztahy použijte pro výpočet rychlosti a zrychlení pro úseky zrychlování pohybu.

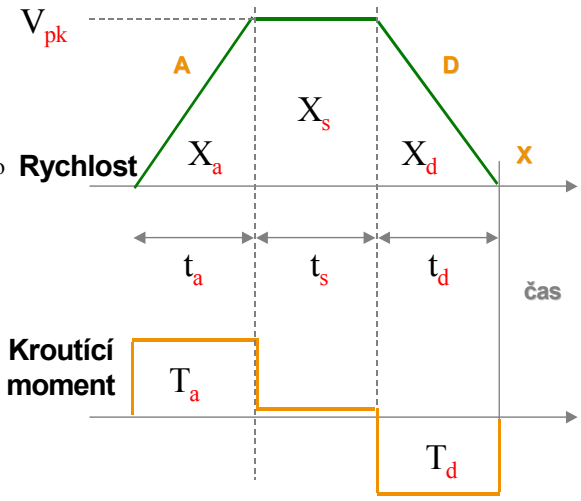
Tabulka 7-2. Lineární transformace rovnic

Dáno / *Řešení pro*	**A, X**	**A, V**	**A, t**	**V, t**	**V, X**	**X, t**
Zrychlení (A)				V/t	$V^2/2X$	$2X/t^2$
Rychlost (V)	$\sqrt{2AX}$		At			2X/t
Vzdálenost (X)		$V^2/2A$	$At^2/2$	Vt/2		
čas(t)	$\sqrt{2X/A}$	V/A			2X/V	

Obrázek 7-23 uvádí příklad lichoběžníkového pohybu. Z nulové rychlosti osa začne zrychlovat v kladném směru (t_a), bude se pohybovat (ustálenou) rychlostí po určitou dobu (t_s), pak zpomalí zpět na nulovou rychlost (t_d). To je celý pohyb nebo pohybový segment. Když se podíváte na obrázek, snadno lze oddělit různé části pohybu. ***Jednoduchým pravidlem je rozdělit lichoběžníkový pohyb na tři časové úseky, jednu třetinu na zrychlení, jednu třetinu na ustálenu rychlost a zbývající třetinu na zpomalení.*** Ustálená (X_s) část rovnoměrně rozděleného profilu rychlosti představuje ½ ujeté vzdálenosti a části zrychlení a zpomalení představují ¼ celkové vzdálenosti. Pravidlo třetin minimalizuje efektivní hodnotu proudu kroutícího momentu motoru a je nejekonomičtějším využitím energie.

Lichoběžníkový pohyb

- Maximálně omezuje rychlost motoru
- Vyšší kroutící moment pro zrychlení než při trojúhelníkovém pohybu
- Symetrický profil (čas 1/3, 1/3, 1/3) maximalizuje přenos energie na břemeno
- Nejběžnější pro dlouhé pohyby



A = zrychlení
D = zpomalení
X = vzdálenost
V_{pk} = špičková rychlost
t_a = doba zrychlování
t_s = doba s ustálenou rychlostí
t_d = doba zpomalování
T_a = kroutící moment zrychlování
T_d = kroutící moment zpomalování
X_a = vzdálenost zrychlování
X_s = vzdálenost s ustálenou rychlostí
X_d = vzdálenost zpomalování

Rovnice

$$x = V_{pk}\left(0.5t_a + t_s + 0.5t_d\right)$$

$$V_{pk} = \frac{x}{\left(0.5t_a + t_s + 0.5t_d\right)}$$

$$a = \frac{V_{pk}}{t_a} \Rightarrow D = \frac{V_{pk}}{t_d}$$

Obrázek 7-23. Lichoběžníkový pohyb

Jakmile bude rozvržený segment pohybu, je nutno zkontrolovat specifikace nebo fyzická omezení platná pro pohyb. Například pohyb se musí dokončit během určitého intervalu ($t_a + t_s + t_d$) nebo se musí ujet pevná vzdálenost (X). Maximální rychlost (V_{pk}) servomotoru je jedním příkladem fyzického omezení. Při dvou známých hodnotách části zrychlení pohybového segmentu je možno zbývající proměnnou najít pomocí kinematických rovnic, jak je znázorněno v následujícím příkladu.

Příklad použití profilu lichoběžníkové rychlosti

U tohoto příkladu předpokládejme, že se úplný pohyb 16 palců musí provést během tří sekund a maximální rychlost přenášená přes převody je 15 palců za sekundu. Použijeme obecné pravidlo pro lichoběžníkové pohyby a rozdělíme dobu pohybu na třetiny: t_a = 1 s, t_s = 1 s a t_d = 1 s. Pohyb v délce 16 palců je také možno rozdělit na tři vzdálenosti. Ustálená část (X_s) rovnoměrně rozděleného lichoběžníkového profilu rychlosti představuje ½ ujeté vzdálenosti a části zrychlení (X_a) a zpomalení (X_d) představují ¼ celkové vzdálenosti. X_a =4 palce, X_s =8 palce a X_d =4 palce.

Má-li se vypočítat špičková rychlost (V_{pk}), během první části zrychlení pohybu se musí ujet daná vzdálenost (X_a) během daného času (t_a). Z výše uvedeného kinematického vztahu pro rychlost (2X/t) s použitím daného X_a =4 palců a t_a = 1 sekunda, (2*4 palce) / 1 sekunda = 8 palců/sekundu.

Má-li se vypočítat zrychlení, nejjednodušší vztah je (V/T)=(8 palců/sekundu / 1 sekunda) = 8 palců/sekundu/sekundu.

Poloha (vzdálenost = X) je celá ujetá vzdálenost ($X_a + X_s + X_d$) nebo 16 palců.

Trojúhelníkové profily rychlosti

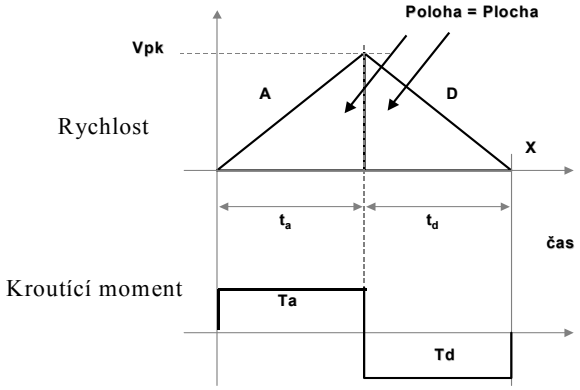
V porovnání s lichoběžníkovým profilem pro stejnou vzdálenost a čas, trojúhelníkový profil rychlosti minimalizuje velikost zrychlení serva a vyžaduje vyšší rychlost servomotoru. Trojúhelníkový profil použijte pro rychlé kratší pohyby.

Rovnice:

$$x = \frac{1}{2} V_{pk} (t_a + t_d)$$

$$V_{pk} = \frac{2(x)}{(t_a + t_d)}$$

$$a = \frac{V_{pk}}{t_a}$$



Obrázek 7-24. Trojúhelníkový profil rychlosti

Nelineární zrychlení nebo zrychlení po S křivce

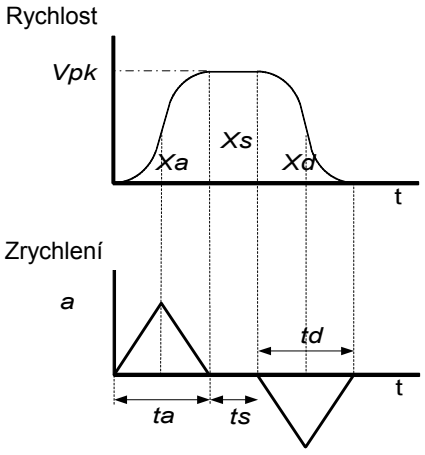
Výpočet zrychlení po S křivce nebo s omezeným škubáním se provádí jednoduše, pokud se nejdříve provede lineární výpočet. Kontroléry pohybu APM a DSM používají 100% omezení škubání. Má-li se převést lineární zrychlení na zrychlení se 100% omezením škubání, buď se zdvojnásobí hodnota zrychlení (2*A) nebo se zdvojnásobí doba použitá pro zrychlení ($2t_a$). Použití zrychlení po S křivce se stejnou velikostí zrychlení (A) jako lineární zrychlení bude vyžadovat dvojnásobek času (t_a) pro dosažení rychlosti. Pokud doba trvání pohybu musí zůstat stejná a servo bude mít dostatečný špičkový kroutící moment, k dosažení rychlosti za stejnou dobu použijte dvojnásobné zrychlení (2*A).

Rovnice:

$$X_a = X_d = \frac{V_{pk}^{\ 2}}{a}$$

$$t_a = t_d = \frac{2V_{pk}}{a}$$

$$t_s = \frac{X_s}{V_{pk}}$$



Obrázek 7-25. Zrychlení po S křivce

Chybová a výstražná hlášení Motion Editoru

Editor generuje tři typy chybových hlášení; syntaktické chyby, sémantické chyby a výstrahy. Ty jsou vysvětlené níže.

Editor vygeneruje programový kód, pouze pokud váš zdrojový program pohybu nebude obsahovat žádné syntaktické nebo sémantické chyby. Pokud editor zjistí neznámou syntaktickou nebo sémantickou chybu, vygeneruje chybové hlášení, které je možno použít ke zjištění závady v programu. Tento problém je popsaný na poslední stránce této kapitoly ("Používání chybových hlášení k lokalizaci chyb pohybových programů").

Chybová hlášení zobrazená ve stavovém okně obsahují číselný chybový kód. Následující seznam uvádí společně chybový kód, popis chyby a obecné informace o příčině.

Motion Editor uplatňuje maximální omezení pro polohu, rychlost a zrychlení na základě měřítka 8:1 uu/cts.

Syntaktické chyby

Programovací software editoru pohybu převádí programy na kód, který používá DSM314. Pokud zdrojový kód porušuje syntaktická pravidla, editor nemůže rozpoznat kód a vygeneruje syntaktickou chybu. Syntaktické chyby se snaží popsat zdroj chyby.

Sémantické chyby

Tato část popisuje chyby kompilační analýzy, které hlásí kompilační analyzátor pohybu, a jejich typické příčiny.

(M200) Nedefinovaný identifikátor
Textový řetězec není přípustná proměnná nebo klíčové slovo pohybového programu.

(M201) Registr parametrů musí být v rozsahu P000 - P255
Pohybový program adresuje registr parametrů mimo rozsah P000 - P255.

(M203) Proměnná CTL musí být v rozsahu CTL01 - CTL32 (DSM314)
Pohybový program adresuje bit CTL mimo platný rozsah.

(M204) Neplatný vstup pohybového programu
Soubor pohybového programu obsahuje neplatný znak. Soubory pohybového programu smí obsahovat pouze ASCII text nebo prázdné místo.

(M210) Hexadecimální konstanty musí být v rozsahu 16#0 - 16#FFFFFFFF
Pohybový program obsahuje hexadecimální číslo mimo platný rozsah.

(M211) Binární konstanty musí být v rozsahu 0 až (2^32)-1
Pohybový program obsahuje binární číslo mimo platný rozsah. Binární číslo nesmí obsahovat více než 32 binárních číslic.

(M212) Celočíselné konstanty musí být v rozsahu 0 až 4294967294
Pohybový program obsahuje celočíselnou hodnotu bez znaménka, kterou nelze zapsat pomocí 32 bitů.

(M213) Celočíselné konstanty se znaménkem musí být v rozsahu 2147483648 až 2147483647
Pohybový program obsahuje celočíselnou hodnotu se znaménkem, kterou nelze zapsat pomocí 32 bitů.

(M214) Příkaz SYNC je platný pouze v programech a podprogramech pro více os.
Pohybový program nebo podprogram pro jednu osu se snaží definovat synchronizační blok.

(M215) Programy pro více os nepodporují osu 3 nebo 4
Povely v programech pro více os mohou adresovat pouze osu 1 nebo 2.

(M220) Zadaná osa je mimo rozsah
Pohybový program pro jednu osu smí adresovat pouze osu 1, 2, 3 nebo 4

(M221) Zrychlení smí být v rozsahu 1 - 1073741823
Povel ACCEL zadal zrychlení mimo platný rozsah.

(M222) Rychlost musí být v rozsahu 1 - 8388607
Povel VELOC zadal rychlost mimo platný rozsah.

(M223) Poloha musí být v rozsahu -536870912 - 536870911
Povel CMOVE nebo PMOVE zadal polohu mimo platný rozsah.

(M224) Prodleva musí být v rozsahu 0 - 60000
Povel DWELL zadal prodlevu mimo platný rozsah.

(M225) Číslo bloku musí být v rozsahu 1 - 65 535
Pohybový program nebo podprogram se snaží definovat číslo bloku mimo platný rozsah.

(M230) V programu pro jednu osu se musí zadat osa
Povely ACCEL, VELOC, CMOVE, PMOVE, DWELL a WAIT v programu nebo podprogramu pro jednu osu musí specifikovat osu.

(M231) V programu pro jednu osu nelze zadat osu
ACCEL, VELOC, CMOVE, PMOVE, DWELL a WAIT v programu nebo podprogramu pro jednu osu *nesmí* specifikovat osu.

(M233) Opakovaně zadané zrychlení bez vloženého pohybového povelu
Je zakázáno měnit zrychlení pro danou osu, pokud nebude vložený povel PMOVE nebo CMOVE.

(M234) Opakovaně zadané rychlost bez vloženého pohybového povelu
Je zakázáno měnit rychlost pro danou osu, pokud nebude vložený povel PMOVE nebo CMOVE.

(M235) Číslo bloku je v tomto programovém bloku již definováno
Pohybový program nebo podprogram se snaží definovat číslo bloku, které již bylo definováno.

(M236) Není definovaný cílový blok skoku
Pohybový program nebo podprogram obsahuje příkaz JUMP na číslo bloku, který není definovaný.

(M237) Cílový podprogram volání není definovaný
Pohybový program nebo podprogram obsahuje volání podprogramu, který není definovaný.

(M238) Program musí být v rozsahu 1 - 10
Příkaz PROGRAM se snaží definovat číslo programu, které je mimo platný rozsah.

(M239) Snaha znovu definovat program. Program je již definovaný.
Příkaz PROGRAM se snaží definovat program pomocí čísla programu, které již je definované.

(M240) Definice konce programu příkazem ENDPROG
PROGRAM byl ukončený příkazem ENDSUB nebo v programu byl zjištěný příkaz ENDSUB.

(M242) Chybějící příkaz ENDPROG
Když se zjistil konec souboru, PROGRAM nebyl ukončený příkazem ENDPROG.

(M243) Podprogram musí být v rozsahu 1 - 40
Příkaz SUBROUTINE se snaží definovat číslo podprogramu, které je mimo platný rozsah.

(M244) Snaha znovu definovat podprogram. Podprogram již existuje
Příkaz SUBROUTINE se snaží definovat program pomocí čísla podprogramu, které již je definované.

(M245) Definice konce podprogramu příkazem ENDSUB
PODPROGRAM byl ukončený příkazem ENDPROG nebo v podprogramu byl zjištěný příkaz ENDPROG.

(M246) Není definovaný žádný podprogram
Programový blok obsahuje povel ENDSUB, ale není otevřený žádný PODPROGRAM.

(M247) Podprogram nemůže volat sebe sama
DSM nepodporuje žádný typ rekurzívnosti. Po vyvolání podprogram nemůže vyvolat sebe nebo nemůže být vyvolaný již vyvolaným podprogramem.

(M248) Definice osy podprogramu musí souhlasit s volajícím programem
Snaha vyvolat program pro jednu osu z programu nebo podprogramu pro více os nebo vyvolat podprogram pro více os z programu nebo podprogramu pro jednu osu.

(M249) Program nebo podprogram se již definuje
V neukončeném příkazu PROGRAM nebo SUBROUTINE byl zjištěný příkaz PROGRAM nebo SUBROUTINE.

(M280) Překročení meze instrukce, max. 1000
Programový blok pohybu smí obsahovat maximálně 1000 programových příkazů. Tato chyba se bude generovat, když počet příkazů překročí tuto mez.

(M281) Soubor musí obsahovat alespoň jeden program
Programový blok pohybu musí obsahovat minimálně jeden PROGRAM; jinak ho nelze vyvolat. Tato chyba se bude generovat, když programový blok pohybu neobsahuje žádné PROGRAMY.

(M282) Příkaz musí být v programu nebo podprogramu
Tato chyba se bude generovat, když se povely pohybového programu vyskytnou mimo PROGRAM nebo PODPROGRAM.

(M283) Tato instrukce je pro daný typ modulu neplatná
Blok pohybového programu obsahuje instrukci, která je pro cílový modul neplatná.

(M293) Překročen maximální počet chyb.
Kompilační analyzátor pohybového programu při analýze bloku pohybového programu může hlásit maximálně 30 chyb. Když se tento limit překročí, generuje se tato chyba a další chyby se již nehlásí.

(M300) Instrukce kompilační analýzy musí předcházet před vykonatelnými příkazy
Pseudoinstrukce #pragma se musí zadat na začátku bloku pohybového programu, tj. před příkazy pohybového programu.

(M301) Neplatný výběr instrukce
Byla zadaná neplatná pseudoinstrukce #pragma.

(M302) Neplatný parametr instrukce
Byl zadaný neplatný výběr jako parametr pseudoinstrukce #pragma.

Výstrahy

Výstrahy se generují pro kód, který se zdá být sporný, ale nezpůsobí specificky chybu. Tato část popisuje výstrahy syntaktické analýzy, které hlásí syntaktický analyzátor pohybu, a jejich typické příčiny.

(M482) Neočekávaný konec programu: Neuzavřený komentář
Když se zjistil konec souboru, nebyl uzavřený komentář.

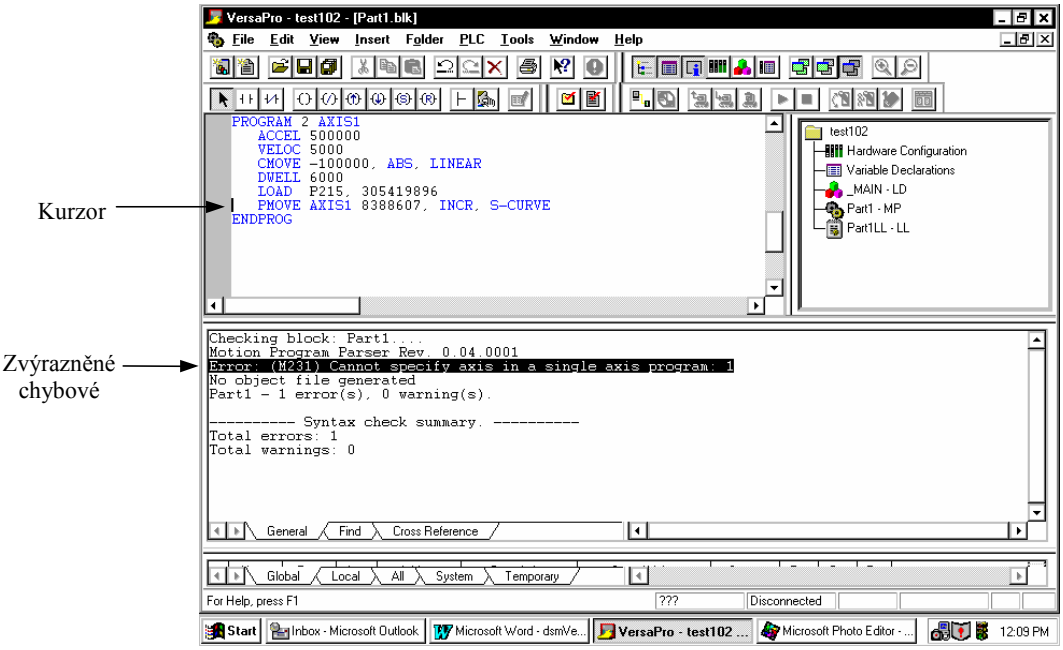
(M483) Vnořované komentáře.
Syntaktický analyzátor pohybu nepodporuje vnořované komentáře. Výstraha se generuje, když komentář bude definovaný uvnitř komentáře.

(M490) Program neobsahuje žádné vykonatelné příkazy.
Výstraha se generuje, když programový blok nebude obsahovat žádné vykonatelné příkazy.

Používání chybových hlášení k lokalizaci chyb pohybových programů

Po vytvoření pohybových programů nebo podprogramů v okně Motion Editoru můžete hledat základní chyby kliknutím na ikonu Kontrola bloku na panelu nástrojů. 

Editor zkontroluje blok pohybového programu a nahlásí všechny zjištěné chyby v informačním okně. Následující obrázek ukazuje příklad chyby zjištěné během kontroly.



Obrázek 7-26. Používání chybových hlášení k lokalizaci chyb pohybových programů

Chyba ukázaná na obrázku výše, “Chyba: (M231) V programu pro jednu osu nelze zadat osu: 1”, se týká posledního řádku programu hned před příkazem ENDPROG. Všimněte si, že na tomto řádku bylo zjištěno AXIS1. Protože PROGRAM 2 je program pro jednu osu, použití čísla osy uvnitř programu není povoleno a tak se generovala chyba.

Ve výše uvedeném příkladu se poklepalo na chybové hlášení, jak je indikováno skutečností, že je zvýrazněno inverzním zobrazením. Když se toto provede, kurzor skočí na místo v programu, které chybu způsobilo. Vidíte, že kurzor je na začátku řádku obsahujícího příkaz AXIS1.

Jako další pomoc při lokalizaci chyb odstavec “Chybová a výstražná hlášení Motion Editoru” této kapitoly uvádí běžné příčiny různých chybových kódů. Například popis chyby (M231) v příkladu výše uvádí:

> **(M231) V programu pro jednu osu nelze zadat osu**
>
> ACCEL, VELOC, CMOVE, PMOVE, DWELL a WAIT v programu nebo podprogramu pro jednu osu *nesmí* specifikovat osu.

Kapitola

8

Pohyb vlečené osy

Nakonfigurování DSM314 pro *Řídicí smyčky vlečené osy* = Povoleno (na kartě konfiguračního softwaru Konfigurace osy) umožňuje, aby servoosa (podřízená) reagovala na vstup Master osy použitím programovatelného poměru slave **:** master. DSM314 definuje poměr slave **:** master jako poměr dvou celých čísel A a B. Základní vztah pro výpočet pohybu vlečené osy je:

Pohyb vlečené servoosy (slave osy) = pohyb Master osy × (A/B)

nebo

poměr slave : master = poměr A : B

Pokud bude vyvolaný také povel *Jog*, *Pohyb rychlostí* nebo *Vykonání pohybového programu*, pohyb osy bude reprezentovat kombinaci pohybu master osy a interně zadaného pohybu. Tato kapitola uvádí podrobnosti pohybu serva ve vztahu ke vstupu master osy. Další podrobnosti o kombinovaném pohybu vlečené osy a zadaného pohybu najdete v kapitole 9.

Když bit %Q *Start povolení vlečené osy* bude v jedničce, osa okamžitě začne sledovat zvolený *Master zdroj*, pokud však nebude zvolený vstup *Aktivace povolení vlečené osy*. Pokud bude zvolený vstup *Start povolení vlečené osy*, pak bit %Q *Povolení vlečené osy* musí být v jedničce a na vstupu musí dojít k přechodu z nuly do jedničky. Externí vstup aktivace CTL01 - CTL032 se volí v konfiguračním softwaru.

DSM314 vždy obsluhuje vlečenou osu v režimu “úpravy náběhu”. Pokud master osa bude mít nenulovou rychlost, když vlečená osa bude povolená, slave osa bude zrychlovat nakonfigurovaným ***Zrychlením úpravy náběhu*** na rychlost, která umožní dohonit master osu.

Master zdroje

Servoosu DSM314 je možno nakonfigurovat tak, aby sledovala libovolné dvě z osmi možných master vstupních zdrojů. Tyto dva zdroje se označují jako Zdroj 1 a Zdroj 2. Bit %Q *Volba master zdroje vlečené osy* určuje, jestli je aktivní Zdroj 1 nebo Zdroj 2. Použitelné volby jako Zdroj 1 a Zdroj 2 jsou:

- Zadaná poloha osy 1
- Okamžitá poloha osy 1
- Zadaná poloha osy 2
- Okamžitá poloha osy 2
- Zadaná poloha osy 3
- Okamžitá poloha osy 3
- Zadaná poloha osy 4
- Okamžitá poloha osy 4

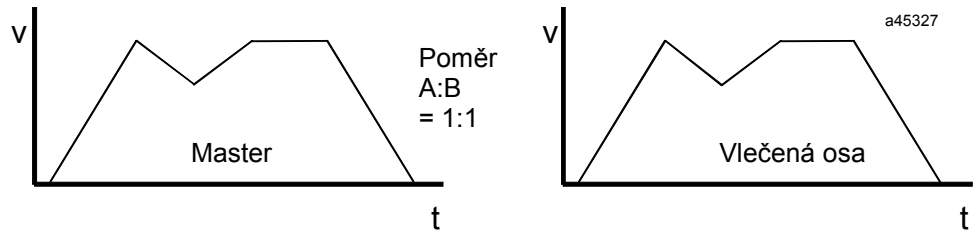
Všimněte si, že pohyb vlečené osy se sčítá s povelem *Jog*, *Pohyb rychlostí* nebo pohybovými programy. Pokud slave osa bude sledovat vstup rychlostí V1 a bude zadaný *Jog* s rychlostí V2, osa se bude pohybovat rychlostí V1 + V2.

Externí master vstupy

Okamžitá poloha pro osu 1 - osu 4 představuje externí zdroje master osy. Jako zdroj okamžité polohy se může použít snímač polohy připojený k ose nebo zpětná vazba servosystému. Smyčka vlečené osy DSM314 umožňuje sledovat zvolený externí zdroj, jak je znázorněno v tomto příkladu:

Příklad 1: Sledování master vstupu okamžité polohy osy 3

V tomto příkladu je uvedený graf rychlosti (v) jako funkce času (t), který znázorňuje rychlosti master vstupu (Okamžitá poloha 3) a slave osy, která sleduje master. DSM314 je nakonfigurované tak, že ***Master zdroj vlečené osy* 1 = Okamžitá poloha 3** a bit %Q *Zvolený master zdroj* je v nule. Poměr A:B je 1:1. Profil rychlosti vlečené (slave) osy je shodný s master vstupem.



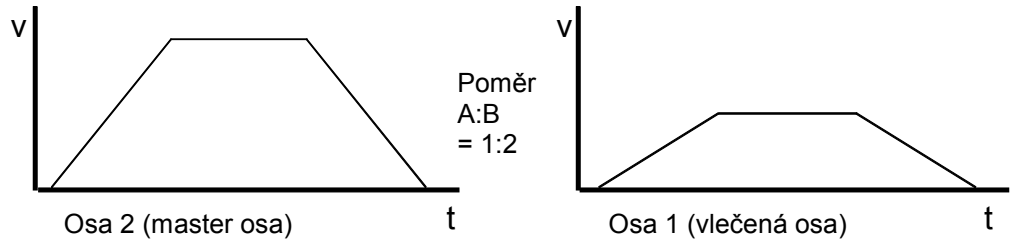
Obrázek 8-1. Sledování master vstupu snímače polohy 3

Interní generátory povelů master osy

Zadaná poloha pro osu 1 - osu 4 představuje interní zdroje master osy. Smyčka vlečené osy DSM314 umožňuje sledovat zvolený zdroj interního povelu, jak je znázorněno v tomto příkladu:

Příklad 2: Sledování interního master povelu

V tomto příkladu je osa 1 DSM314 nakonfigurovaná tak, že ***Master zdroj vlečené osy 2*** = **Zadaná poloha 2** a bit %Q *Zvolený interní master* je v jedničce. Poměr A:B je 1:2. Pro osu 2 je zadaný *Pohyb rychlostí* 12000 a pak 0. Osa 1 sleduje osu 2 poloviční rychlostí zrychlením osy 2 a pohyb vykoná pouze na polovině vzdálenosti, než kterou vykoná osa 2.



Obrázek 8-2. Sledování snímače polohy servoosy 2

Poměr A:B

Osa DSM314, která sleduje master vstup, může sledování provádět v širokém rozsahu poměrů slave **:** master (A:B). Hodnota "A" může být libovolné číslo od -32768 do 32767. Hodnota "B" může být kdekoliv mezi 1 a 32767. Velikost poměru A:B může být v rozsahu 1:10 000 až 32:1. Proto je možno používat velmi přesné poměry, například 12 356:12 354 nebo 32 000:1024.

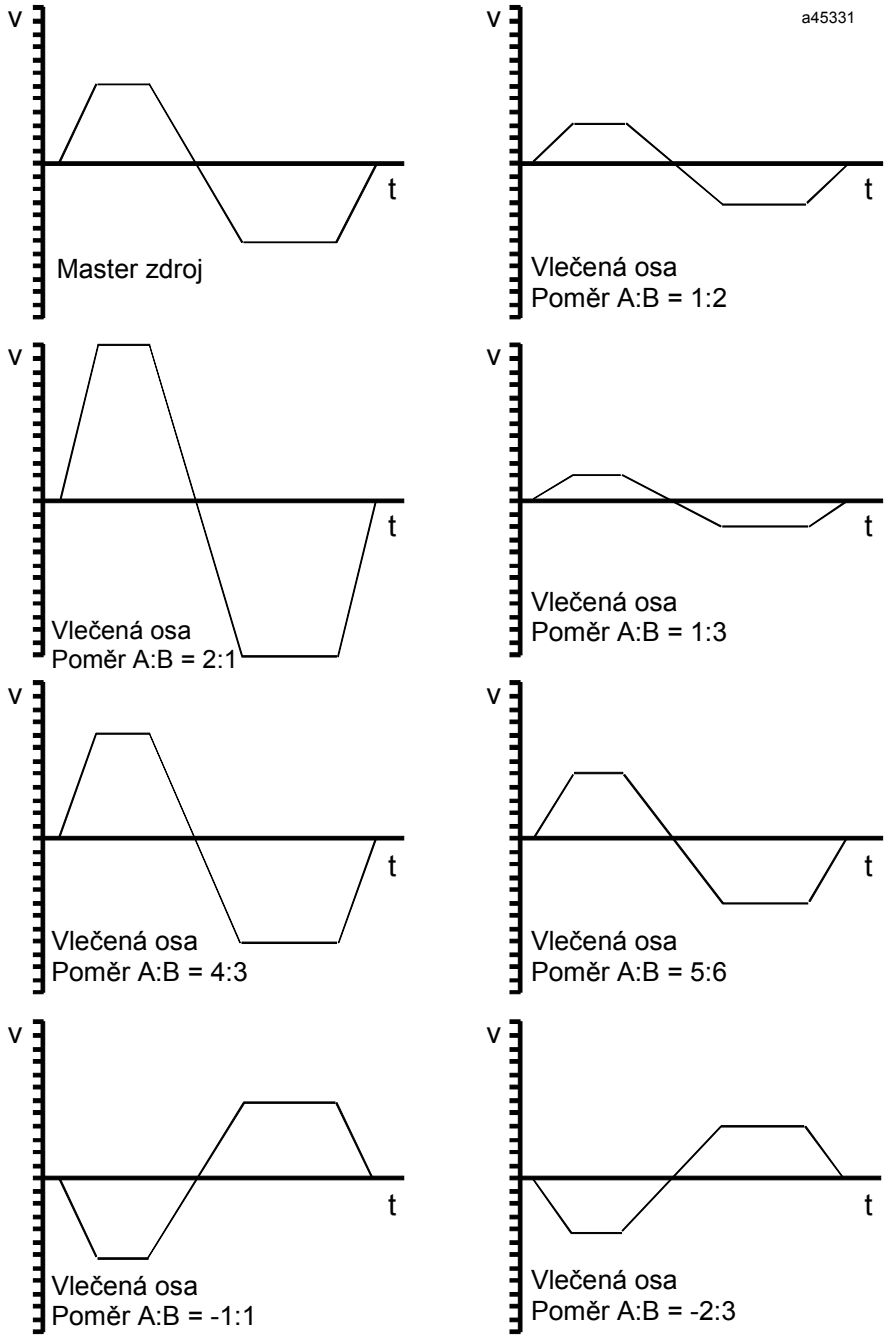
Povel %AQ *Poměr vlečené osy A/B* je možno použít ke změnění poměru A:B kdykoliv, dokonce i během vlečení. Avšak neplatný poměr bude mít za následek generování stavové chyby a bude se ignorovat. Neplatný poměr je poměr, kde se B rovná nebo je menší než nula, nebo kdy velikost A:B je větší než 32:1 nebo menší než 1:10 000.

Když se provádí vlečení s jiným poměrem než 1:1, profil rychlosti masteru a vlečené osy bude vypadat poněkud odlišně. Vodorovné čáry označující konstantní rychlost a šikmé čáry označující zrychlení a zpomalení budou odlišné. **Pokud poměr A:B bude menší než 1:1, rychlost vlečené osy a zrychlení bude menší než u masteru. Obdobně pokud poměr A:B bude větší než 1:1, rychlost vlečené osy a zrychlení bude větší než u masteru.** Doba trvání pohybu a doba, kdy osa zrychluje, zpomaluje nebo zůstává na konstantní rychlosti, je pro master a vlečenou osu stejná.

Ujetá vzdálenost, která je v profilu rychlosti dána plochou mezi grafem a časovou osou, bude vzdálenost masteru násobená poměrem A:B. Pokud A bude nula, nebude se generovat žádný vlečný pohyb. Pokud A bude záporné, vlečená osa se bude pohybovat v opačném směru.

Příklad 3: Příklady poměrů A:B

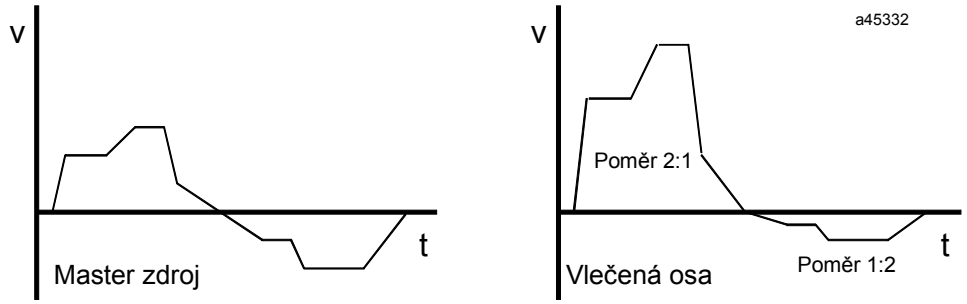
Všechny následující příklady sledují vstup master zdroje s různým poměrem A:B.



Obrázek 8-3. Příklady poměrů A:B

Příklad 4: Změna poměru A:B

Jednou z možností proměnných poměrů A:B je použít jeden poměr během pohybu v kladném směru a jiný poměr při pohyb v záporném směru. Všimněte si, že určení kladné a záporné rychlosti a aktualizace poměru A:B se musí provést v PLC nebo v programu lokální logiky DSM314. U níže uvedeného profilu vlečená osa používá poměr 2:1 při pohybu v kladném směru a poměr 1:2 při pohybu v záporném směru.



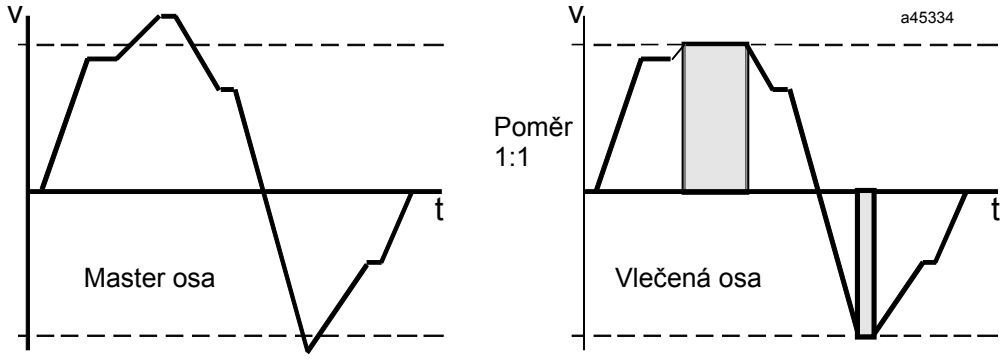
Obrázek 8-4. Změna poměru A:B

Omezení rychlosti

Omezení rychlosti je možno provést pomocí *Mezní rychlosti* nastavené v konfiguračním softwaru. Když rychlost masteru přesáhne nastavenou mez, vlečená osa bude pokračovat v pohybu omezenou rychlostí násobenou poměrem A:B. Bit %I *Mezní rychlost* je v jedničce a generuje se stavová chyba, která indikuje, že osa slave již není pevně svázaná s polohováním podle master vstupu. Osa slave v podstatě *zůstane viset* za master vstupem.

Příklad 5: Omezení rychlosti

V tomto příkladu je ***Mezní rychlost*** nastavená na 100 000. Proto rychlost osy slave bude v obou směrech pevně omezená na 100 000 uživatelských jednotek/sekundu. Když master osa bude mít špičku větší než je mez, vlečená osa zůstane na této mezi. Jakmile master osa zpomalí pod mez, vlečená osa bude pokračovat ve sledování rychlosti master osy. Pulsy generované nad ***Mezní rychlostí*** jsou pro vlečenou osu ztracené. Vodorovné přerušované čáry označují meze rychlosti. Vyšrafovaná plocha označuje časy, kdy bit *V mezní rychlosti* je v jedničce a vlečená osa zůstane zaostávat za masterem.



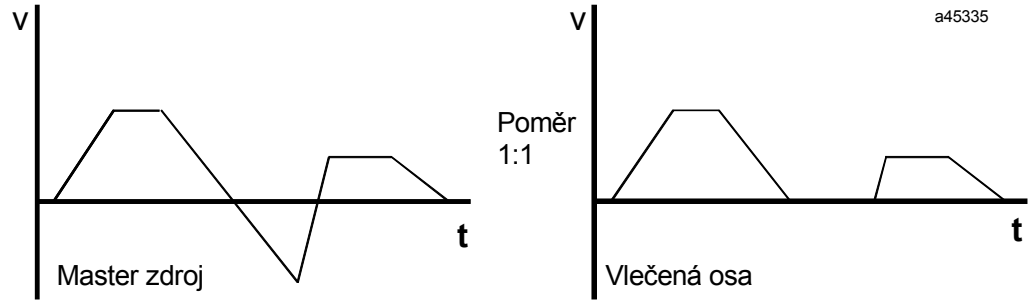
Obrázek 8-5. Omezení rychlosti

Jednosměrná činnost

Nastavení ***Směr povelu*** při konfiguraci osy na Pouze kladný nebo Pouze záporný bude mít za následek jednosměrný pohyb vlečené osy. Všechny pulsy master osy ve směru omezeném nulou se budou ignorovat. Pulsy ve směru omezeném nulou nebudou generovat žádnou chybu. Bit %I *V mezní rychlosti* však vyjadřuje přítomnost master povelu ve směru omezeném nulou.

Příklad 9: Jednosměrný provoz

V tomto příkladu je ***Směr povelu*** nastavený na Pouze kladný. Jak je znázorněno v profilu rychlosti níže, slave osa sleduje kladné pulsy, ale ignoruje záporné pulsy. Všimněte si, že když se master bude pohybovat v záporném směru, bit %I *V mezní rychlosti* bude v jedničce, ale nebude se generovat žádná stavová chyba.



Obrázek 8-6. Jednosměrný provoz

Povolení vlečené osy pomocí externího vstupu

Každý bit CTL01- CTL32 je možno nakonfigurovat jako start povolení pro vlečenou osu. Pokud CTL bit bude nakonfigurovaný jako externí vstup na čelní desce, tento vstup je možno použít ke spuštění vlečené osy. Když nebude zvolený žádný vstup, vlečená osa bude povolená a zakázaná přímo bitem %Q *Povolení vlečené osy*. Když bude zvolený vstup pro Start povolení a bit %Q *Povolení vlečené osy* bude v jedničce, další kladný přechod definovaného vstupu okamžitě povolí vlečenou osu. Vlečená osa zůstane povolená, dokud se bit %Q *Povolení vlečené osy* nevynuluje. Vstupy 24 V na čelní desce mají 5 ms filtry, které vytváří časovou odezvu Startu povolení vlečené osy 5-7 milisekund. Vstupy 5 V na čelní desce tyto filtry nemají a doba odezvy Startu povolení je 2 milisekundy nebo méně.

Když se vyskytne start *Povolení vlečené osy*, *Zadaná poloha* v tomto bodě se zachytí do registru parametrů tak, aby bylo možno jí použít v povelu *Programovaný pohyb*. Poloha se zachytí do parametru 226 (pro servoosu 1), parametru 234 (pro servoosu 2), parametru 242 (pro servoosu 3) nebo parametru 250 (pro servoosu 4).

Stav *Vlečená osa povolena* se vrátí v bitu %I pro každou osu.

Zakázání vlečené osy pomocí externího vstupu

Každý bit CTL01- CTL32 je možno nakonfigurovat jako Start zákazu pro vlečenou osu. Aktivační vstup se testuje, pouze když %Q bit *Povolení vlečené osy* bude v jedničce. Když %Q bit *Povolení vlečené osy* bude v jedničce, přechod aktivačního bitu z nuly do jedničky zakáže vlečenou osu. Nastavení %Q bitu *Povolení vlečené osy* do nuly okamžitě zakáže vlečenou osu bez ohledu na nastavení aktivace zákazu.

Akce zákazu vlečené osy pro inkrementální polohu

Nastavení ***Zákaz činnosti vlečené osy*** pro inkrementální polohování umožňuje, aby vlečená osa prováděla **Inkrementální registrační pohyb**. Zakázání vlečené osy pomocí bitu %Q *Povolení vlečené osy* nebo volitelného ***Zákazu startu*** způsobí, že osa bude pokračovat svou aktuální rychlostí, pak zpomalí a zastaví po ujetí zadané vzdálenosti. Inkrementální vzdálenost je zadaná v registru parametrů pro každou osu:

P227 = Inkrementální vzdálenost osy 1

P235 = Inkrementální vzdálenost osy 2

P242 = Inkrementální vzdálenost osy 3

P250 = Inkrementální vzdálenost osy 4

Inkrementální vzdálenost představuje celkovou změnu okamžité polohy, ke které dojde od bodu, kde vlečená osa byla zakázána, až do jejího zastavení. Během registračního pohybu vlečené osy nesmí být aktivní superimponované pohybové povely (*Jog, Pohyb rychlostí* nebo pohybové programy).

Řízení náběhu zrychlení vlečené osy

U aplikací, kde vlečená osa je povolená po tom, co master osa již je na zadané rychlosti, je možno funkci náběhu vlečené osy použít k řízení velikosti zrychlení, aby se vlečená osa dostala na tuto rychlost. To je možno provést bez ztráty pulsů master povelu od bodu, ve kterém byla vlečená osa povolena. Během automaticky generovaného upraveného pohybu s řízením náběhu vlečené osy zrychlení/zpomalení nepřesáhne nakonfigurovanou hodnotu ***Náběhu zrychlení vlečené osy*** a zajistí hladký pohyb. Když vlečená osa bude povolená, osa slave nabíhá na master rychlost aktivní nakonfigurovanou velikostí ***Náběhu zrychlení vlečené osy***. Tato funkce je velmi užitečná, když master zdroj bude v pohybu před povolením režimu vlečené osy. Kromě PLC bitu %Q Povolení vlečené osy je možno nakonfigurovat bit CTL (CTL01-CTL32) jako signál povolení vlečené osy pro funkce registrace polohy. Když bit %Q *Povolení vlečené osy* bude v jedničce, zvolený bit CTL bude působit jako spouštění náběžnou hranou pro povolení režimu vlečené osy. Po povolení vlečené osy řídí aktivní stav následující funkce pouze PLC %Q bit *Povolení vlečené osy*. Když vlečená osa bude povolená k pohybujícímu se master zdroji, některé pulsy master zdroje nelze použít okamžitě. Master pulsy, které se kumulují během zrychlování vlečené osy, se uloží. Když vlečená osa dosáhne master rychlosti, vloží se během úpravy pohybu korekční vzdálenosti. Tento pohyb má automaticky vypočítaný lichoběžníkový profil rychlosti určený ***Dobou úpravy vzdálenosti náběhu vlečené osy***, počtem nakumulovaných pulsů a nakonfigurovaným ***Náběhem zrychlení vlečené osy. Dobu úpravy vzdálenosti náběhu vlečené osy*** nastavte na požadovanou dobu v konfiguračním softwaru nebo je možno jí změnit pomocí PLC povelu %AQ 42h.

Pokud ***Doba úpravy vzdálenosti náběhu vlečené osy*** bude příliš krátká, pak automaticky generovaná rychlost bude mít trojúhelníkový profil. Pokud během korekce vzdálenosti rychlost

přesáhne 80% nakonfigurované ***Mezní rychlosti***, pak automaticky vypočítaná rychlost pohybu bude omezená na 80% mezní hodnoty. Omezení rychlosti úpravy pohybu na 80% mezní rychlosti dává systému určitou rezervní kapacitu rychlosti k tomu, aby mohl pokračovat ve sledování rychlosti master zdroje. V obou případech se bude hlásit výstražné hlášení a skutečná doba úpravy vzdálenosti bude delší než naprogramovaná doba, ale vzdálenost se zkoriguje správně.

Nastavení ***Doby úpravy vzdálenosti náběhu vlečené osy*** na 0 umožňuje, aby funkce náběhu zrychlila osu, aniž by se prováděla kompenzace nakumulovaných pulsů. V tomto případě rychlost nepřesáhne master rychlost. U aplikací, kde ztracené pulsy nejsou na závadu, nastavte dobu úpravy vzdálenosti = 0.

Superimponovaný profil pohybu, který se automaticky generuje funkcí náběhu vlečené osy (s nenulovou dobou úpravy), je implicitně lichoběžníkový, používá ***Náběhové zrychlení vlečené osy*** a vzdálenost odvozenou od aktivní ***Doby úpravy náběhu***.

V závislosti na vztazích s rychlostí master zdroje hodnota ***Mezní rychlosti*** může ovlivnit funkci jinak. Následující příklady znázorňují tyto body.

Případ 1: Rychlost master zdroje je menší než 80% nakonfigurované ***Mezní rychlosti*** a doba úpravy (doba Mkup) představuje dostatečně dlouhý interval, aby výsledná rychlost zůstala menší než 80% hodnoty Vlim. To je provoz, kterému se dává přednost, nehlásí se žádné chyby a pohyb s nadměrnou rychlostí funkce náběhu se objeví v rámci zadané doby úpravy. Pokud se nezvýší rychlost master zdroje, rychlost vlečené osy nepřesáhne 80% hodnoty Vlim.

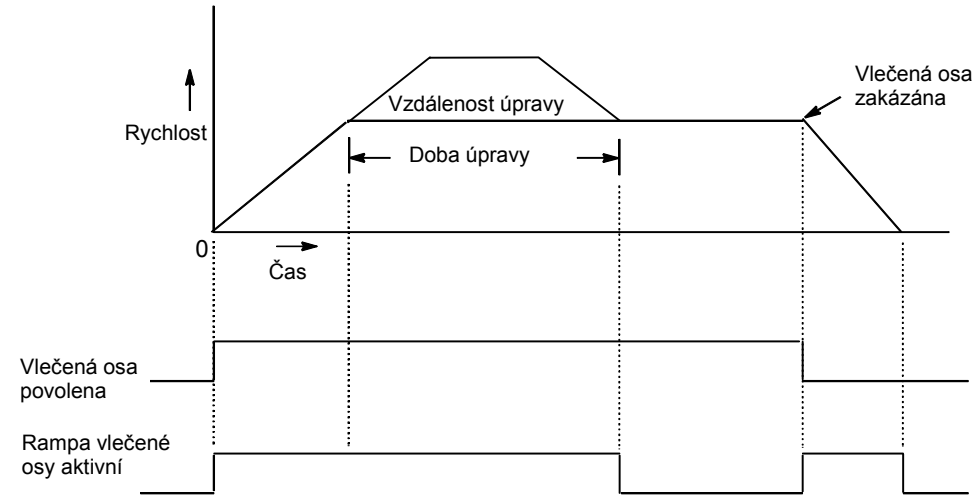
Případ 2: Rychlost master zdroje je menší než 80% nakonfigurované ***Mezní rychlosti***, ale interval doby úpravy je příliš krátký, aby se umožnil provoz jako v případě 1. Když rychlost vlečené osy bude souhlasit se zadanou rychlostí masteru, bude se hlásit pouze stavová chyba (ECh). Pohyb úpravy náběhu se bude zrychlovat s použitím aktivního ***Náběhu zrychlování vlečené osy*** až na 80% mezní rychlosti (Vlim). Objeví se pohyb úpravy náběhu a použijí se všechny nakumulované pulsy uložené během počátečního zrychlování.

Případ 3: Rychlost master zdroje je větší než 80% nakonfigurované ***Mezní rychlosti***, když rychlost vlečené osy bude souhlasit se zadanou rychlostí masteru. Vrátí se pouze stavová chyba (EAh) a neprovede se žádný korekční pohyb.

Případ 4: V okamžiku, když rychlost vlečené osy bude souhlasit se zadanou rychlostí masteru a vykonává se pohyb úpravy náběhu a podmínky jsou stejné jako v případě 1 a případě 2 a vyvolal se pohyb úpravy náběhu, master zdroj se zvýší na >80% ***Mezní rychlosti***. Počet nakumulovaných pulsů a aktivní hodnota doby úpravy určí, jestli se pohyb úpravy náběhu dokončí během zadané doby úpravy náběhu. Chyba pouze stavu (F2h) se objeví, pokud zadaná rychlost masteru a rychlost úpravy pohybu dosáhnou dohromady 100% mezní rychlosti. Zadaná rychlost masteru nepřesáhne 100% hodnoty ***Mezní rychlosti***. Nakumulované pulsy se mohou ztratit a pohyb úpravy náběhu se nedokončí.

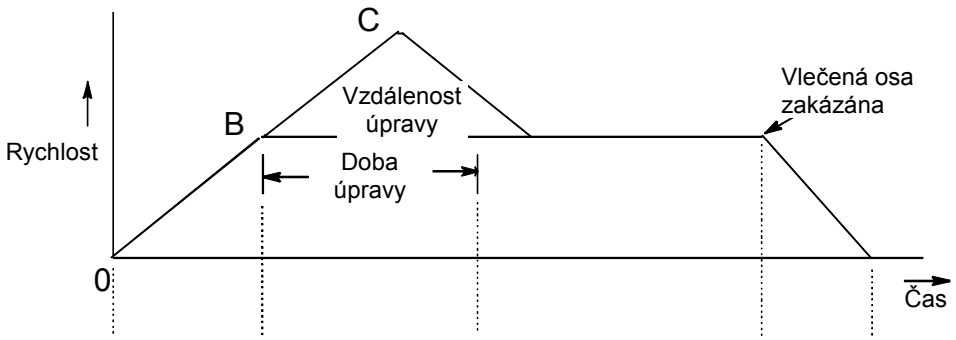
Bit %I *Náběh vlečené osy aktivní* přejde do jedničky, když řízení náběhu platí pro úpravu náběhu nahoru i dolů.

Bity %I *Vlečená osa povolena* a *Náběh vlečené osy aktivní* je možno monitorovat v PLC a určit tak, jestli je aktivní cyklus části náběhu vlečené osy směrem nahoru nebo dolů. Následující obrázek ukazuje stav *Vlečená osa povolena* a *Náběh vlečené osy aktivní* během cyklu vlečené osy.



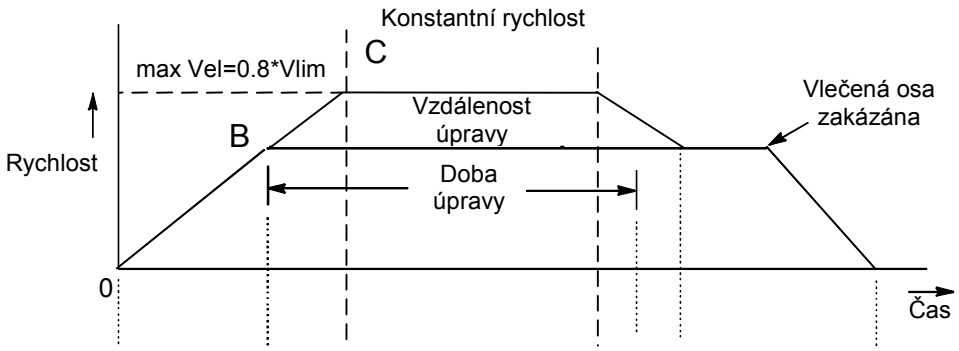
Obrázek 8-7. Cyklus náběhu vlečené osy nahoru/dolů (případ 2)

Naprogramovaná doba úpravy může být příliš krátká pro provedení požadované korekce vzdálenosti. V takovém případě se bude hlásit výstraha (v bodě B trajektorie), ale systém bude pokračovat ve zrychlování a zajistí minimální možnou dobu korekce vzdálenosti. Profil rychlosti pro takový případ je znázorněný na obrázku 8-12.



Obrázek 8-8. Cyklus náběhu vlečené osy nahoru/dolů – Případ 2 s příliš krátkou dobou úpravy náběhu

Během fáze náběhu korekce vzdálenosti se řídí mezní rychlost. Pokud vypočítaná rychlost bude příliš vysoká, pak se rychlost pevně omezí a nastaví se výstražný chybový kód (v bodě C trajektorie). Obrázek 8-13 ukazuje profil rychlosti během cyklu náběhu vlečené osy pro tento případ.



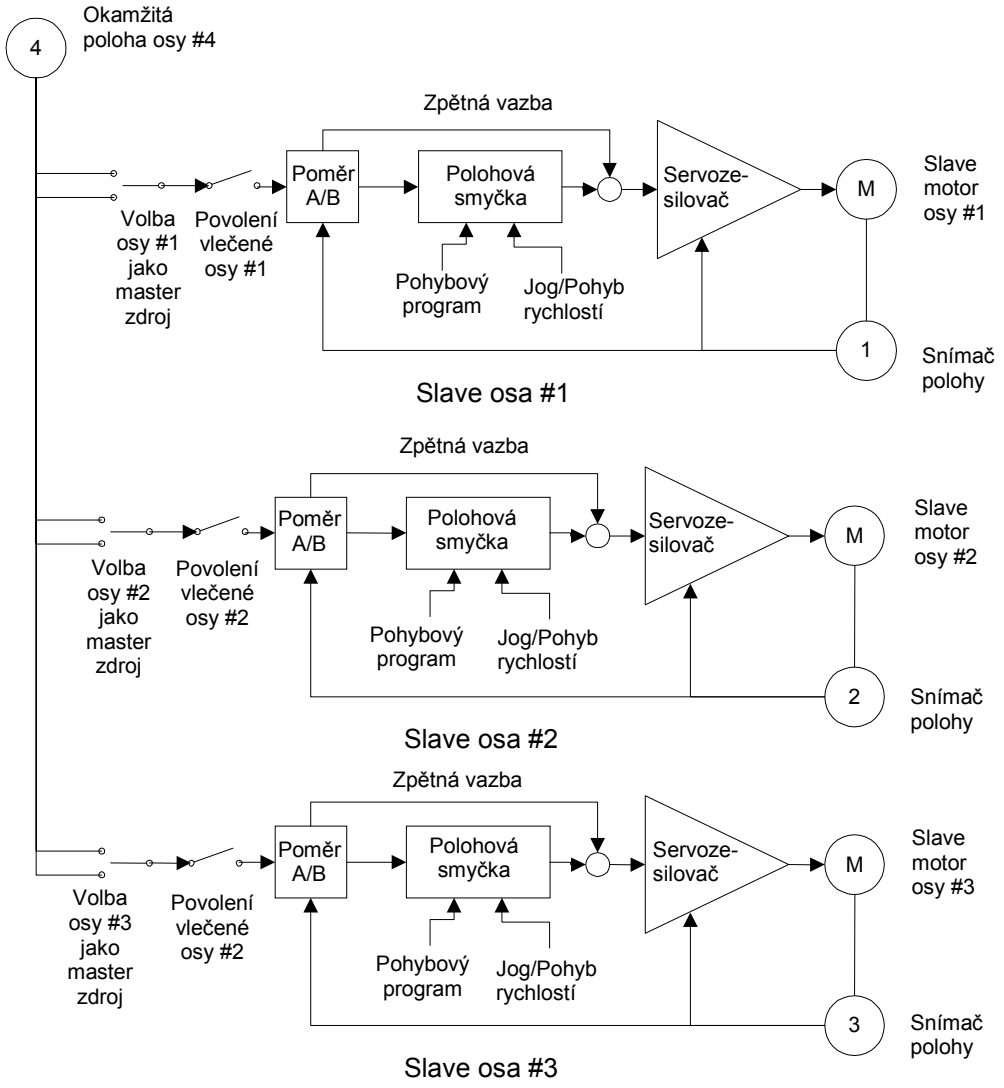
Obrázek 8-9. Cyklus náběhu vlečené osy nahoru/dolů – případ s aktivní mezní rychlostí

Pokud doba zrychlování (úsek BC trajektorie na obrázku 8-13) přesahuje 128 sekund, pak se bude hlásit jiný výstražný kód. V takovém případě se korekce vzdálenosti provede také správně.

Povelový zdroj režimu vlečené osy a možnosti připojení

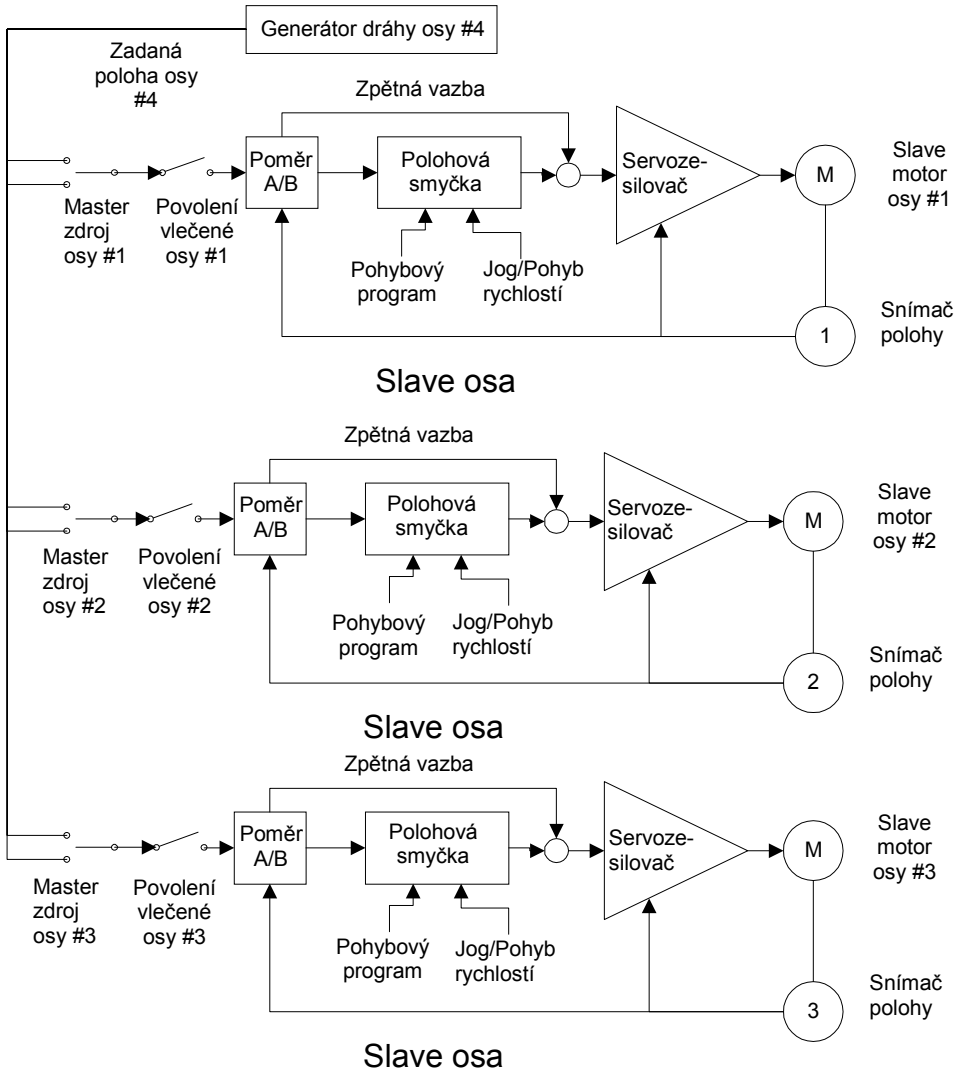
Schémata na následujících stránkách znázorňují různé možnosti připojení smyčky master osy a vlečené slave osy.

Obrázek níže ukazuje tři analogové osy DSM314 připojené paralelně s Okamžitou polohou pro osu #4. Všimněte si, že u této konfigurace *může* běžet funkce lokální logiky. Je to proto, že generátor povelů pro osu #4 se pro tuto konfiguraci nevyžaduje. Všechny konfigurační údaje master zdroje jsou nastavené na Okamžitou polohu osy #4. To není požadavek. Eliminuje to však zdroj chyby v důsledku toho, že bit master zdroje bude zvolený nesprávně.



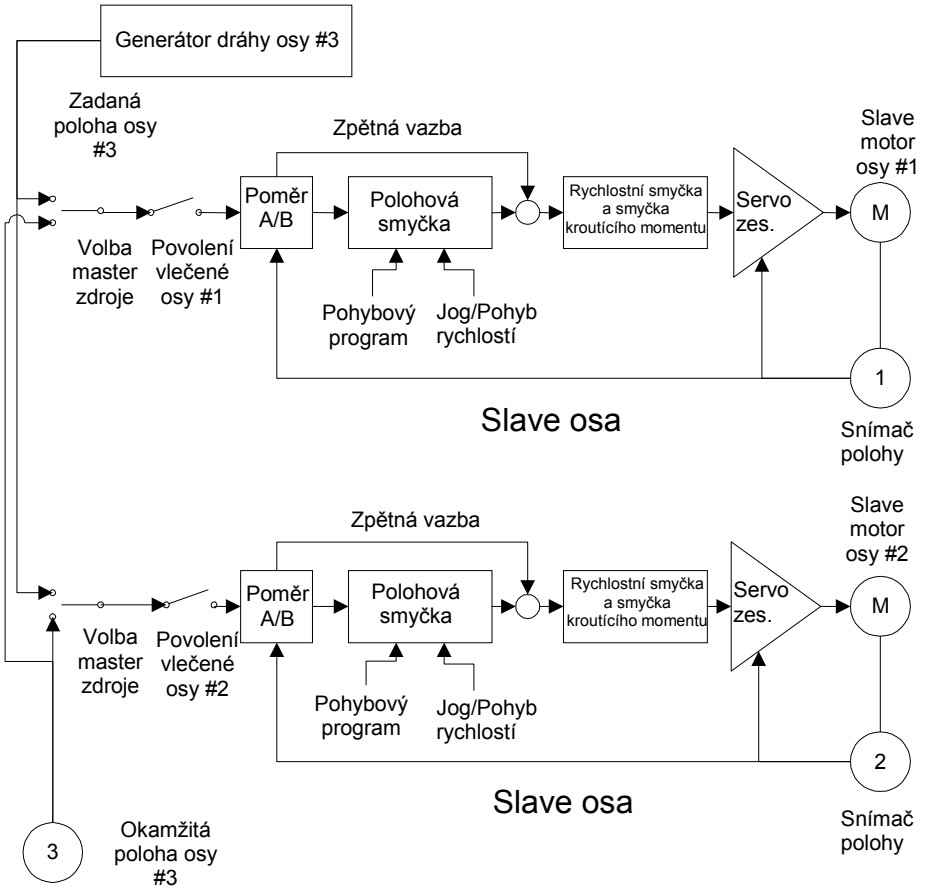
Obrázek 8-10. Analogová vlečená osa pro 3 osy / Paralelní struktura / Zdroj vlečené osy = Okamžitá poloha 4

Obrázek níže ukazuje tři analogové osy DSM314 připojené paralelně se Zadanou polohou pro osu #4. Všimněte si, že u této konfigurace *nemůže* běžet funkce lokální logiky. Je to proto, že generátor povelů pro osu #4 se pro tuto konfiguraci vyžaduje. Všechny konfigurační údaje master zdroje jsou nastavené na Zadanou polohu osy #4. To není požadavek. Eliminuje to však zdroj chyby v důsledku toho, že bit master zdroje bude zvolený nesprávně.



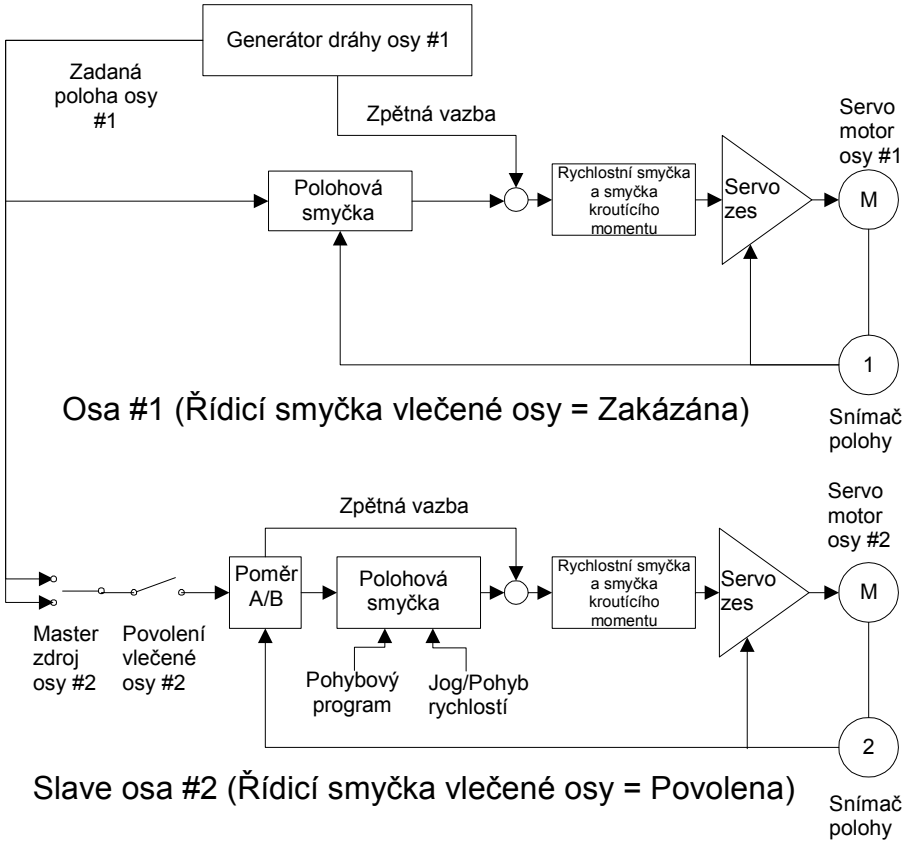
Obrázek 8-11. Analogová osa pro 3 osy / Paralelní struktura / Zdroj = Zadaná poloha 4

Následující obrázek znázorňuje dvě digitální osy DSM314 připojené paralelně se Zadanou polohou nebo Okamžitou polohou pro osu #3. Všimněte si, že u této konfigurace *může* běžet funkce lokální logiky. Je to proto, že generátor povelů pro osu #4 se pro tuto konfiguraci nevyžaduje.



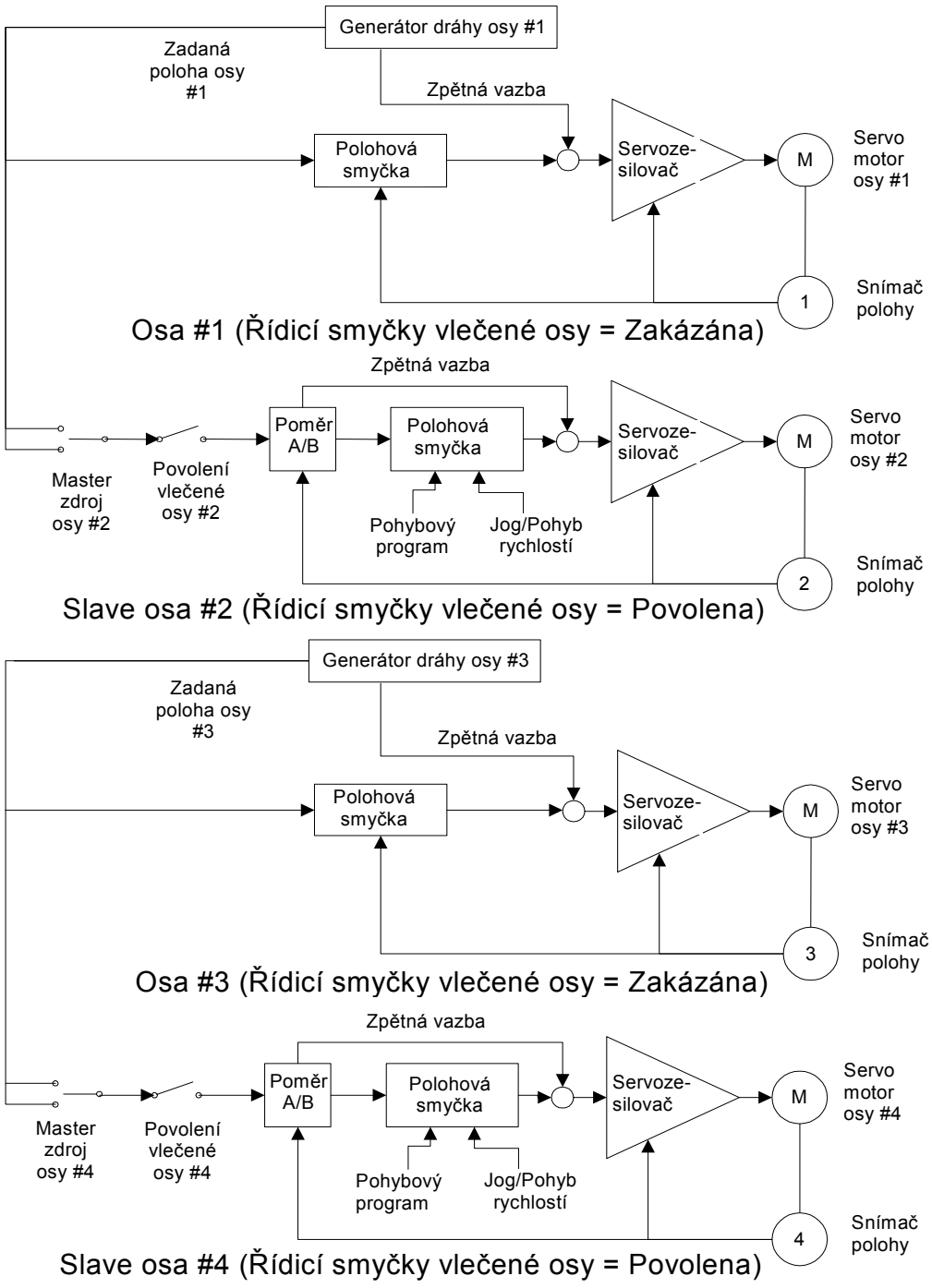
Obrázek 8-12. Digitální vlečená osa pro 2 osy / Paralelní struktura / Zdroj = Zadaná nebo Okamžitá poloha 3

Následující schéma znázorňuje dvě digitální osy DSM314 připojené paralelně se Zadanou polohou ze servosmyčky řízení osy 1 pro osu 1 a osu 2. To umožní, aby obě osy běžely po stejné zadané dráze. Všimněte si, že osa 1 je nakonfigurovaná s ***Řídicí smyčkou vlečené osy*** = Zakázáno. Tato konfigurace neumožňuje sdílení břemena mezi osami, které jsou pevně svázané. Všimněte si, že u této konfigurace *může* běžet funkce lokální logiky. Je to proto, že generátor povelů pro osu #4 se pro tuto konfiguraci nevyžaduje.



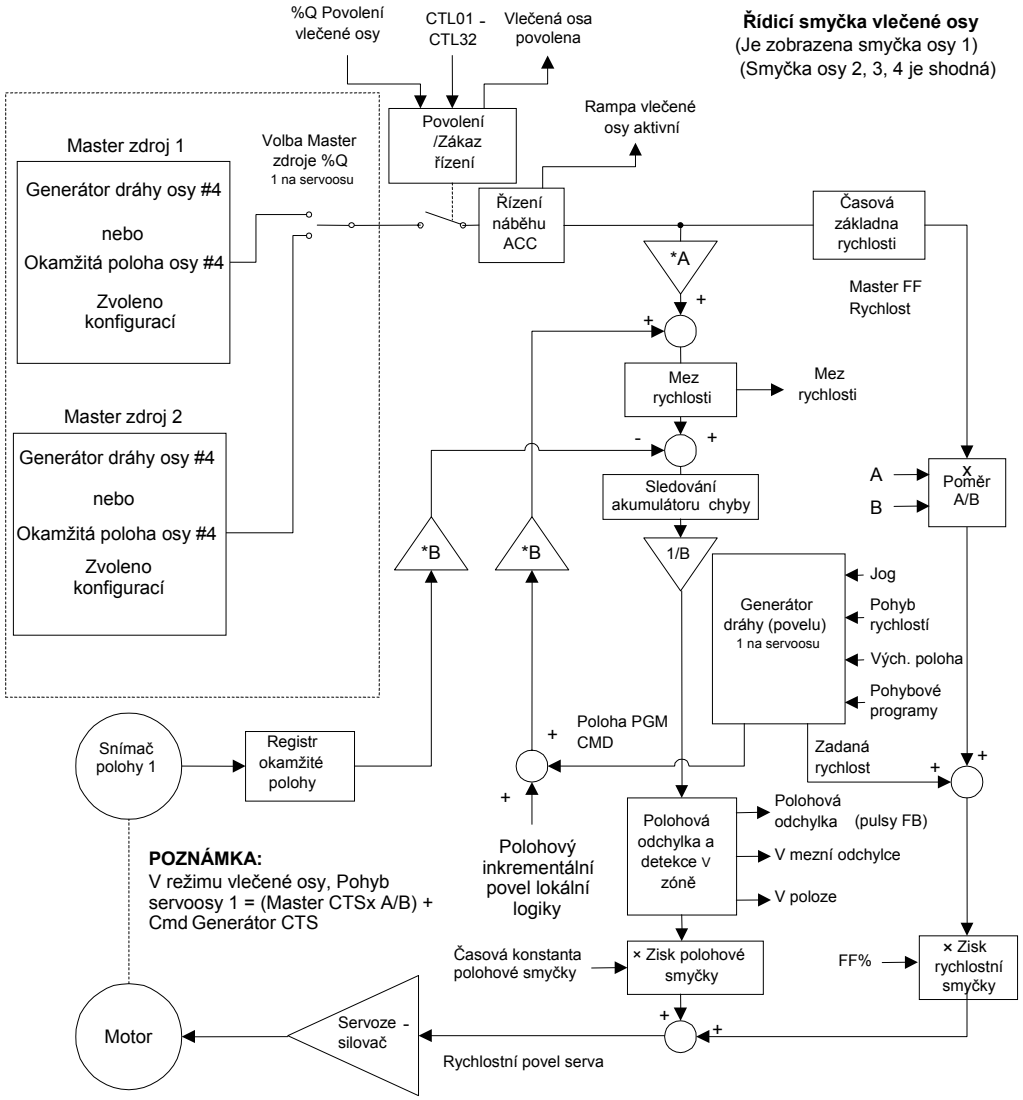
Obrázek 8-13. Digitální vlečená osa pro 2 osy / Paralelní struktura / Zdroj = Zadaná poloha 1

Následující schéma znázorňuje čtyři analogové osy DSM314 připojené ve dvou paralelních párech. Všimněte si, že u této konfigurace *nemůže* běžet funkce lokální logiky. Je to proto, že polohová servosmyčka pro osu #4 se pro tuto konfiguraci vyžaduje.



Obrázek 8-14. Analogová vlečená osa pro čtyři osy / Paralelní struktura / Zdroj = Zadaná poloha 1 a Zadaná poloha 3

Blokové schéma řídicí smyčky vlečené osy



Obrázek 8-15. Blokové schéma řídicí smyčky vlečené osy

Kapitola 9

Kombinace pohybu vlečené osy a zadaného pohybu

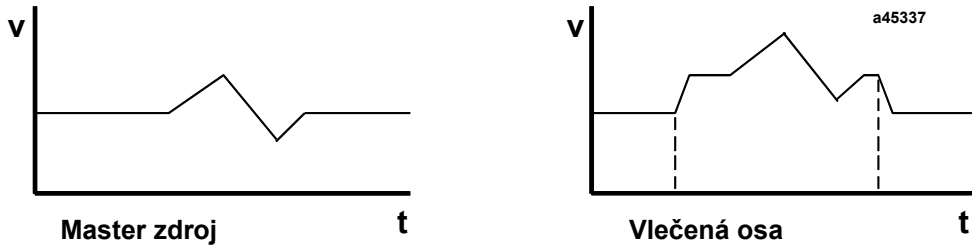
Kombinované pohyby se skládají z pohybu vlečené osy zadaného master osou a jednoho z těchto interně zadaných pohybů:

- Povel %Q *Jog Plus/Minus*
- Povel %AQ *Pohyb rychlostí*
- Povel %AQ *Pohyb*
- Uložený pohybový program

Kombinované pohyby se přičítají. Pohyb slave osy se rovná součtu pohybů zadaných master osou a interně zadaným pohybem.

Příklad 1: Pohyb vlečené osy kombinovaný s Jogem

V tomto příkladu se nastaví bit %Q *Povolení vlečené osy* do jedničky, což způsobí, že slave osa bude sledovat master vstup. Během doby, kdy slave osa bude provádět sledování, bude bit %Q *Jog Plus* v jedničce. Sledující osa bude zrychlovat z rychlosti masteru na rychlost masteru zvýšenou o aktuální ***Rychlost jogu***. Toto zrychlení bude stejné, jako by osa v tomto okamžiku nesledovala master zdroj. Když se bit %Q *Jog Plus* vynuluje, sledující osa se zpomalí na rychlost svého masteru. V níže uvedených profilech rychlosti tečkované čáry udávají, kdy bit %Q *Jog Plus* je v jedničce a pak v nule.



Obrázek 9-1. Kombinovaný pohyb (Vlečená osa + Jog)

Pohyb vlečené osy kombinovaný s pohybovými programy

Pohybové programy z uložených programů nebo povel %AQ *Pohyb* je možno také kombinovat s master povelem pro řízení vlečené osy. Tyto povely pro pohyb z bodu do bodu mohou být z některého z uložených pohybových programů 1 až 10 a některých podprogramů, které mohou volat. Povel %AQ *Pohyb* se bere jako jednořádkový pohybový program, který používá aktuální ***Rychlost jogu*** a ***Zrychlení jogu***. Vykonávání programu se spustí tak, že PLC nastaví bit %Q *Vykonávání programu n* nebo pošle povel %AQ *Pohyb*.

Pokud nebude existovat žádný master povel, osu je možno ovládat výhradně z programových dat uloženého programu. Proto když nebude žádný master vstup do servoosy 2 a zadaná poloha 2 zvolená jako master zdroj pro servoosu 1, uložený program je možno použít pro řízení servoosy 2 tak, že servoosa 1 bude sledovat zadaný poměr.

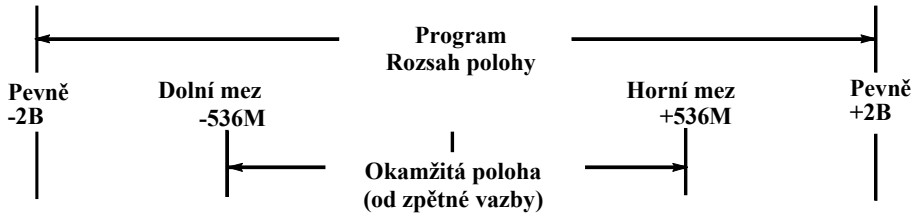
Když se povely PMOVE vykonají a vlečená osa nebude povolená, bit %I *V poloze* se musí nastavit do jedničky na konci pohybu před tím, než naprogramovaný pohyb bude pokračovat. Když vlečená osa bude povolená a protože bit *V poloze* se nemůže nastavit do jedničky při současném sledování master povelu, indikace *V poloze* se pro pokračování nevyžaduje. Další pohyb se vykoná, když se dokončí zadaná vzdálenost předchozího pohybu. Bit %I *V poloze* bude vždy indikovat skutečný stav dosažení polohy.

Aktivní zadaná poloha aktualizovaná a používaná uloženým pohybovým programem se nazývá Poloha zadaná povelem. Tento registr polohy se inicializuje jedním z následujících dvou způsobů vždy, když je program vybraný pro vykonání.

1. Pokud vlečená osa **nebude povolená**, Poloha zadaná povelem se nastaví na aktuální hodnotu *Zadaná poloha = Okamžitá poloha + Polohová odchylka*.

2. Pokud vlečená osa **bude povolená**, Poloha zadaná povelem se nastaví na Programovou referenční polohu **(0).** Protože Poloha zadaná povelem se aktualizuje pouze interně generovanými povely (a ne master povelem), bude pak indikovat polohu zadanou uloženým programem. Absolutní pohybové povely z uloženého programu budou vztaženy k programové referenční poloze.

Rozsahy polohy (v pulsech) pro registr Okamžité polohy a Polohy zadané povelem jsou uvedené na následujícím obrázku.



Když přijde pohybový povel ve stejném směru, Poloha zadaná povelem se přetočí na +2 147 483 647 nebo –2 147 483 648 pulsech.

Okamžitá poloha však bude omezená nakonfigurovanou ***Horní mezí polohy*** a ***Dolní mezí polohy***.

Tabulka 9-1 níže uvádí, které zdrojové povely mají vliv na tyto registry polohy a okamžité a zadané rychlosti. Poloha zadaná povelem se aktualizuje pouze interně generovanými pohybovými povely (programové povely, *Jog Plus Minus*, *Nalezení výchozí polohy* a *Pohyb rychlostí*). *Zadaná rychlost* (vrácená v datech %AI) také indikuje pouze rychlost zadanou těmito interně generovanými pohybovými povely. Vrácená data %AI *Okamžitá poloha* a *Okamžitá rychlost* zohledňují kombinaci master vstupu a pohybových povelů. Jinými slovy pulsy přicházející z master zdroje mají vliv **pouze** na *Okamžitou polohu* a *Okamžitou rychlost*. Pokud nebudou existovat žádné interně generované pohybové povely, *Zadaná rychlost* bude 0 a Poloha zadaná povelem se nezmění.

Tabulka 9-1. Vliv povelového vstupu na registry polohy

POVELOVÝ vstup	**Vlečená osa povolená**	**Registry vlečené osy ovlivněné vstupem**
Master povely (ze zvoleného Master zdroje)	Ne	Bez vlivu
	Ano	Aktualizuje se stavové slovo %AI *Okamžitá poloha* Aktualizuje se stavové slovo %AI *Zadaná poloha* (*Okamžitá poloha* + *Polohová odchylka*) Poloha zadaná povelem **není** ovlivněná Aktualizuje se stavové slovo %AI *Okamžitá Rychlost* Stavové slovo %AI *Zadaná rychlost* **není** ovlivněno
Programové povely	Ne	Aktualizuje se stavové slovo %AI *Okamžitá poloha* Aktualizuje se stavové slovo %AI *Zadaná poloha* *Okamžitá poloha* + *Polohová odchylka*) Poloha zadaná povelem se aktualizuje Aktualizuje se stavové slovo %AI *Okamžitá Rychlost* Stavové slovo %AI *Zadaná rychlost* se aktualizuje (pouze programem zadanou rychlostí)
	Ano	Aktualizuje se stavové slovo %AI *Okamžitá poloha* (Programovým povelem + Master povelem) Aktualizuje se stavové slovo %AI *Zadaná poloha* (*Okamžitá poloha* + *Polohová odchylka*) Poloha zadaná povelem se aktualizuje (pouze programovým povelem) Aktualizuje se stavové slovo %AI *Okamžitá Rychlost* (programem zadanou rychlostí + masterem zadanou rychlostí) Stavové slovo %AI *Zadaná rychlost* se aktualizuje (pouze programem zadanou rychlostí)

Tabulka 9-1. - Pokračování - Vliv povelového vstupu na registry polohy

POVELOVÝ vstup	Vlečená osa povolená	Registry vlečené osy ovlivněné vstupem
Ostatní interně generované pohyby Povely (Výchozí poloha, Jog a Pohyb rychlostí)	Ne	Aktualizuje se stavové slovo %AI *Okamžitá poloha* Aktualizuje se stavové slovo %AI *Zadaná poloha* (*Okamžitá poloha* + *Polohová odchylka*) Poloha zadaná povelem se aktualizuje, ale nepoužívá Aktualizuje se stavové slovo %AI *Okamžitá Rychlost* Stavové slovo %AI *Zadaná rychlost* se aktualizuje (pouze interně zadanou rychlostí)
	Ano (Nalezení výchozí polohy není přípustné)	*Okamžitá poloha* (interním povelem + Master povelem) Aktualizuje se stavové slovo %AI *Zadaná poloha* (*Okamžitá poloha* + *Polohová odchylka*) Poloha zadaná povelem se aktualizuje, ale nepoužívá Aktualizuje se stavové slovo %AI *Okamžitá Rychlost* (interní zadanou rychlostí + masterem zadanou rychlostí) Stavové slovo %AI *Zadaná rychlost* se aktualizuje (pouze interně zadanou rychlostí)

Polohu zadanou povelem je možno synchronizovat s hodnotou %AI *Okamžité polohy* třemi způsoby:

- Vykonáním povelu %Q *Nalezení výchozí polohy*
- Povelem %AQ *Nastavení polohy*
- Povelem %Q *Vykonání pohybového programu n* (pokud vlečená osa nebude povolená)

Vliv těchto povelů je uvedený v následující tabulce 9-2.

Tabulka 9-2. Akce s vlivem na Polohu zadanou povelem

AKCE	Vlečená osa povolená	Výsledná aktualizace registru polohy vlečené osy
Nalezení výchozí polohy	Ne	Stavové slovo %AI *Okamžitá poloha* se nastaví na hodnotu výchozí polohy Poloha zadaná povelem se nastaví na *Okamžitou polohu* + *Polohovou odchylku*
	Ano	Povel %Q *Nalezení výchozí polohy* není přípustný Vrátí se stavová chyba
Povel %AQ *Nastavení polohy*	Nepoužívá se	Stavové slovo %AI *Okamžitá poloha* se nastaví na hodnotu %AQ Poloha zadaná povelem se nastaví na *Okamžitou polohu* + *Polohovou odchylku* **Poznámka:** Nastavení polohy je nepřípustné, pokud bit %I *Pohyb* bude v jedničce.
Vykonání programu	Ne	Stavové slovo %AI *Okamžitá poloha* není ovlivněno Poloha zadaná povelem se nastaví na *Okamžitou polohu* + *Polohovou odchylku*
	Ano	Stavové slovo %AI *Okamžitá poloha* není ovlivněno Poloha zadaná povelem se nastaví na referenční polohu (0)

Programové pohyby se vykonávají souvislým způsobem tak, že inkrementální povely PMOVE nebo CMOVE překračující meze se přetočí přes mez a budou pokračovat. Absolutní povely PMOVE nebo CMOVE je možno také použít pro aplikace, které nevyžadují pohyb za horní/dolní meze pulsů.

Všechny interně generované pohybové povely je možno okamžitě ukončit povelem %Q *Zrušit všechny pohyby*.

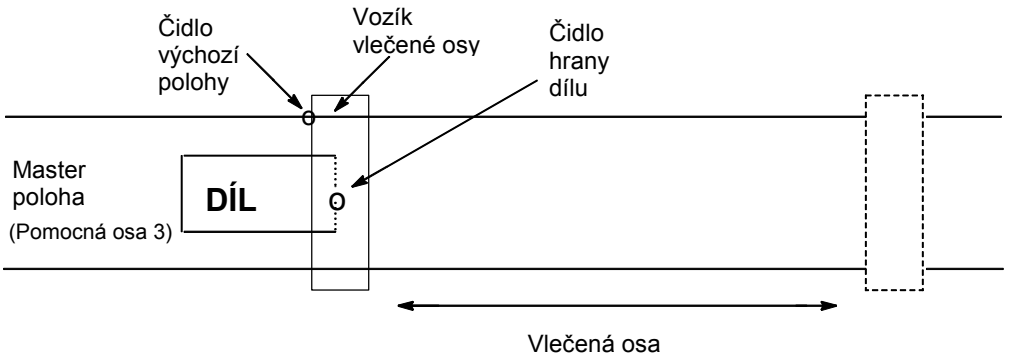
Stavové slovo %AI *Uživatelem volitelná data* je možno změnit a hlásit tak Polohu zadanou povelem pomocí povelu %AQ *Volba dat návratu*. Podrobnosti viz kapitola 5.

Následující příklad aplikace uvádí, jak je možno použít uložený program k řízení polohovacích operací vzhledem k detekované hraně předmětu pohybujícího se rychlostí zjištěnou vstupem snímače polohy master osy (pomocná osa 3).

Příklad 2: Pohyb vlečené osy kombinovaný s pohybovým programem

U aplikací, které vyžadují úpravu dílů za běhu (například čepování, značení, nýtování, bodové svařování, bodové lepení atd.), se využijí pohyby z bodu do bodu přepočítané na pohyb vlečené osy a aktivace vstupních vlastností vlečené osy.

Typická konfigurace a sekvence řízení vyžadovaná pro tyto aplikace je uvedena níže.



Sekvence řízení:

1. Když je bit %Q *Povolení vlečené osy* nastavený na nulu, PLC zadá pohyb Vlečené osy do výchozí polohy, kde *Okamžitá poloha* a Poloha zadaná povelem jsou synchronizované a nastavené na hodnotu Výchozí poloha. Bit %I *Poloha platná* indikuje, kdy tento krok je dokončený.

2. PLC nastaví bitový povel %Q *Povolení vlečené osy* do jedničky.

   Poznámka: Bit CTL01- CTL24, ke kterému je čidlo hrany připojené, je již nakonfigurovaný v konfiguračním parametru *Start povolení vlečené osy*.

3. Když čidlo zjistí hranu dílu, DSM314 povolí, aby vlečená osa začala sledovat vstupy snímače polohy masteru (pomocná osa 3). Bit %I *Vlečená osa povolena* indikuje, kdy osa bude

sledovat master povel. Všimněte si, že funkci náběhu zrychlení a dobu úpravy náběhu je možno použít k tomu, aby se vlečená osa v případě potřeby mohla chytit master osy.

4. Jakmile vlečená osa bude povolená, PLC pošle bit %Q *Vykonání pohybového programu n*, kterým se spustí vykonávání zvoleného programu pro vlečenou osu. V okamžiku, kdy je program zvolený, Poloha zadaná povelem se pošle do programu jako referenční poloha (0), protože vlečená osa je povolená. Vykonávání programu pak bude vztažené k pohybující se hraně dílu v závislosti na tom, jak vlečená osa sleduje díl. Poloha zadaná povelem nyní bude obsahovat polohu vlečené osy vzhledem k hraně dílu a *Okamžitá poloha* bude udávat celkovou vzdálenost, kterou vlečená osa ujela od výchozího bodu (programové povely master +/–).

5. Na konci programu PLC vrátí bit %Q *Povolení vlečené osy* do nuly a vrátí smyčku na krok 1 s opakováním pro další díl.

Poznámka: Protože DSM314 ušetřilo Vstup startu povolení vlečené osy *Zadaná poloha* v registru parametrů (#226 pro osu 1, #234 pro osu 2), krok 1 se tentokrát může použít k vykonání jiného programu, který vykoná absolutní pohybový povel zpátky na polohu podle hodnoty parametru a bude pokračovat krokem 2. V tomto případě je možno bity %I *Pohyb* a *V poloze* použít k indikaci, kdy krok 1 bude hotový.

Tato metoda je možná, protože když vykonání pohybového programu bude zadáno se zakázanou vlečenou osou, poloha zadaná povelem bude nastavená na *Okamžitou polohu + Polohovou odchylku*.

Kapitola 10

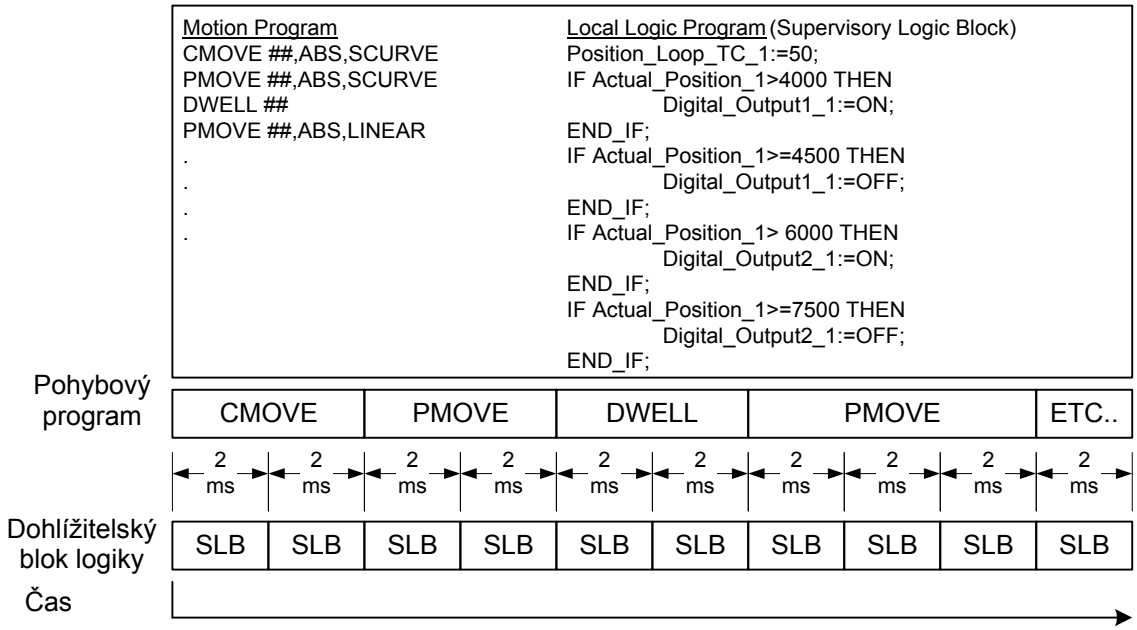
Úvod do programování lokální logiky

Tato kapitola obsahuje úvod do základů koncepce programování lokální logiky. Modul DSM a DSM jazyk pro programování pohybu se v této kapitole do podrobností neprobírají. Tyto koncepce jsou probírány v jiných kapitolách tohoto manuálu.

Úvod do programování lokální logiky

Program lokální logiky pracuje ve spojení s programem logiky PLC a pohybovým programem a vytváří tak flexibilní programovací prostředí. Programy lokální logiky konkrétně dávají uživateli možnost vykonávat matematické a logické operace, které jsou deterministické a synchronizované s rychlostí vykonávání polohové smyčky DSM. Tato možnost je kritická pro mnoho aplikací, kde přesnost a/nebo rychlost vyžadují tuto přísnou synchronizaci.

Funkce lokální logiky DSM dává uživateli možnost vykonávat základní logiku a matematické funkce pomocí modulu DSM. Kromě toho lokální logika umožňuje rychlý přístup čtení/zápisu k lokálním digitálním a analogovým I/O DSM. Úplný seznam použitelných I/O najdete v kapitole 14 a 15. Metoda vykonávání programu lokální logiky zaručuje, že program lokální logiky poběží vzorkovací rychlostí polohové smyčky a dokončí se každou vzorkovací periodu. Poznámka: Pokud modul nebude schopný dokončit vykonávání programu lokální logiky v rámci přiřazeného času, modul bude generovat chybové hlášení. Kapitola 13 a Dodatek E obsahují více informací o časech pro vykonávání programu. Kromě toho program lokální logiky běží paralelně s normálními pohybovými programy DSM. Paralelní vykonávání programů umožňuje, aby program lokální logiky dohlížel na pohybový program. Proto se programy lokální logiky také nazývají dohlížitelské bloky logiky (SLB). Vykonávání lokální logiky v porovnání s vykonáváním pohybového programu je znázorněno na Obrázek 10-1.



Obrázek 10-1. Vykonávání lokální logiky v porovnání s vykonáváním pohybového programu

Než budete psát programy lokální logiky, je důležité, abyste pochopili koncepci znázorněnou na Obrázek 10-1. Program lokální logiky se vykoná až do konce během každé vzorkovací periody polohové smyčky. Program pak vykoná znovu kompletní program lokální logiky během další vzorkovací periody polohové smyčky. Tento způsob vykonávání se liší od způsobu vykonávání pohybového programu. Pohybové programy vykonávají každý povel až do konce sekvenčním způsobem bez jakýchkoliv časových záruk. Tato koncepce je znázorněna v Tabulka 10-1. , která uvádí první čtyři periody vykonávání lokální logiky pro programy lokální logiky a pohybové programy ukázané na obrázku 10-1. V příkladu si všimněte, že program lokální logiky se vykoná až do konce během každé vzorkovací periody polohové smyčky. Příkazy pohybového programu se budou vykonávat, dokud řízený pohyb nedosáhne požadovaného výsledku. Další podrobnosti týkající se vykonávání příkazů pohybového programu najdete v kapitole 7.

Tabulka 10-1. Lokální logika – příklad vykonávání pohybového programu

Číslo vzorkování polohové smyčky	Aktivní příkaz pohybového programu	Příkazy programu lokální logiky
n	CMOVE ##,ABS,S-CURVE	Position_Loop_TC_1:=50;
		IF Actual_Position_1>4000 THEN
		Digital_Output1_1:=ON;
		END_IF;
		IF Actual_Position_1>=4500 THEN
		Digital_Output1_1:=OFF;
		END_IF;
		IF Actual_Position_1> 6000 THEN
		Digital_Output3_1:=ON;
		END_IF;
		IF Actual_Position_1>=7500 THEN
		Digital_Output3_1:=OFF;
		END_IF;
n+1	CMOVE ##,ABS,SCURVE	Position_Loop_TC_1:=50;
		IF Actual_Position_1>4000 THEN
		Digital_Output1_1:=ON;

Číslo vzorkování polohové smyčky	Aktivní příkaz pohybového programu	Příkazy programu lokální logiky
		END_IF;
		IF Actual_Position_1>=4500 THEN
		Digital_Output1_1:=OFF;
		END_IF;
		IF Actual_Position_1> 6000 THEN
		Digital_Output3_1:=ON;
		END_IF;
		IF Actual_Position_1>=7500 THEN
		Digital_Output3_1:=OFF;
		END_IF;
n+2	CMOVE ##,ABS,SCURVE	Position_Loop_TC_1:=50;
		IF Actual_Position_1>4000 THEN
		Digital_Output1_1:=ON;
		END_IF;
		IF Actual_Position_1>=4500 THEN
		Digital_Output1_1:=OFF;
		END_IF;
		IF Actual_Position_1> 6000 THEN
		Digital_Output3_1:=ON;
		END_IF;
		IF Actual_Position_1>=7500 THEN
		Digital_Output3_1:=OFF;
		END_IF;
n+3	CMOVE ##,ABS,SCURVE	Position_Loop_TC_1:=50;
		IF Actual_Position_1>4000 THEN
		Digital_Output1_1:=ON;
		END_IF;
		IF Actual_Position_1>=4500 THEN
		Digital_Output1_1:=OFF;
		END_IF;
		IF Actual_Position_1> 6000 THEN
		Digital_Output3_1:=ON;
		END_IF;
		IF Actual_Position_1>=7500 THEN
		Digital_Output3_1:=OFF;
		END_IF;

Kdy použít lokální logiku nebo logiku PLC

Programovací jazyk lokální logiky obsahuje základní matematické a logické konstrukce. Tyto vlastnosti nejsou určené jako náhrada logických vlastností PLC. Lokální logika je určená jako doplněk logiky PLC a matematických funkcí. Konkrétně lokální logika je určena k řešení malé logické a matematické sady, která vyžaduje přísnou synchronizaci s řízeným pohybem. Program lokální logiky se musí vykonat až do konce během každé vzorkovací periody polohové smyčky. Proto programy lokální logiky mají omezenou velikost. Výchozí velikost programu lokální logiky je omezená na 150 řádků. Pokud se překročí limit 150 řádků, proces výstavby lokální logiky bude generovat chybové hlášení. Když se překročí 100 řádků, bude se generovat upozornění. Pokud program bude příliš velký a výpočetně náročný, může se překročit přípustná doba vykonávání a dojít k upozornění/chybě časovače hlídacího obvodu (viz Dodatek E). Naproti tomu velikost programu PLC je omezená pouze velikostí použitelné paměti. Avšak jak se zvětšuje velikost programu PLC, zvyšují se i doby cyklu PLC. (Další informace o dobách cyklu PLC najdete v *Referenční příručce CPU PLC Series 90-30/20/Micro*, GFK-0467.) To však neplatí o programech lokální logiky. Programy lokální logiky se vždy vykonají až do konce během každé vzorkovací periody polohovací smyčky. Když budete používat PLC logiku, přidaná doba čekání související s dobami cyklu PLC pro časově kritické operace, které jsou těsně svázané s pohybem, může být nepřijatelná nebo omezit výkon procesu. Tyto těsně svázané a časově kritické procesy jsou potenciální aplikace pro lokální logiku. Každý proces je nutno samostatně vyhodnotit a zjistit, které části by se měly napsat PLC logikou a které části by se měly napsat lokální logikou.

Krátký úvod do programování lokální logiky a pohybu

Následující odstavce poskytují informace, jak začít s editorem lokální logiky a editorem pohybového programu. Odstavce se zaměřují na používání programu s důrazem na tvorbu programu, kontrolu syntaxe a načítání programu.

Požadavky

Editory lokální logiky a pohybu jsou zakomponované do prostředí programovacího softwaru. Budete potřebovat některý z následujících softwarových balíků. Návod k instalaci najdete v dokumentaci k softwaru.

- CIMPLICITY Machine Edition Logic Developer – PLC verze 2.1 nebo pozdější
- VersaPro verze 1.1 nebo pozdější

Nastavení funkce DSM314 vyžaduje také CPU 90-30 s firmwarem verze 10.0 nebo pozdější.

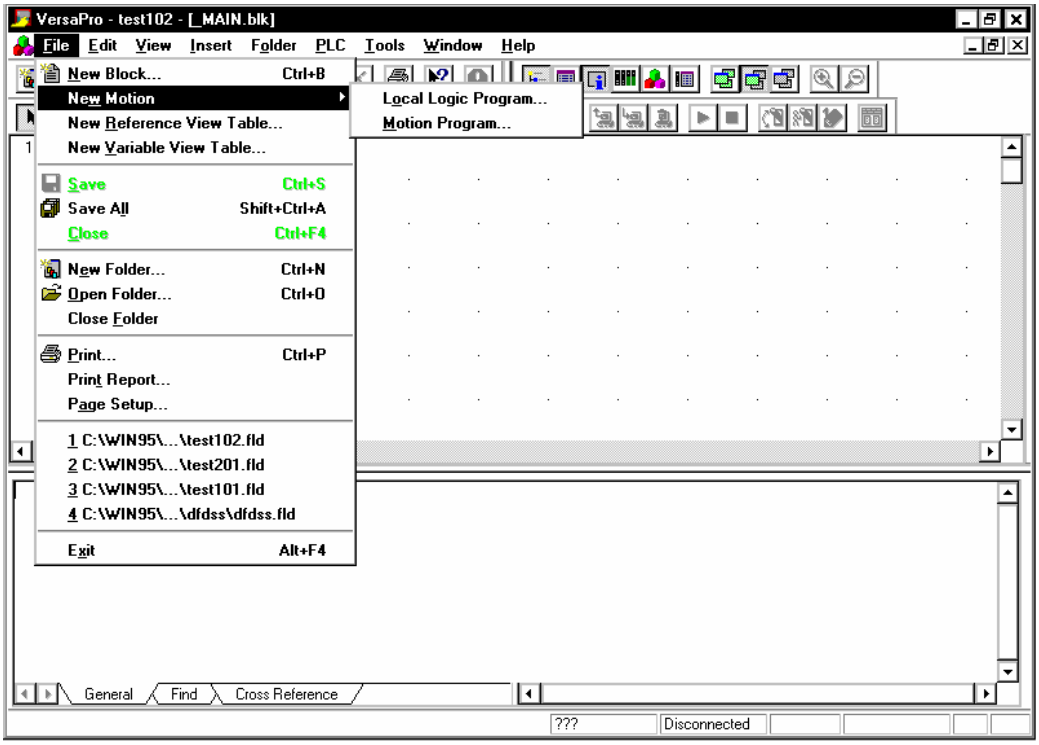
Vytvoření programu lokální logiky

Editor lokální logiky je zakomponovaný do prostředí programovacího softwaru. Editor umožňuje snadné vytváření, editování, ukládání a načítání programu lokální logiky. Program lokální logiky se vytváří v adresáři VersaPro nebo v projektu CIMPLICITY Machine Edition. Podrobnosti, jak vytvořit nebo otevřít projekt, najdete v dokumentaci k softwaru.

Software VersaPro

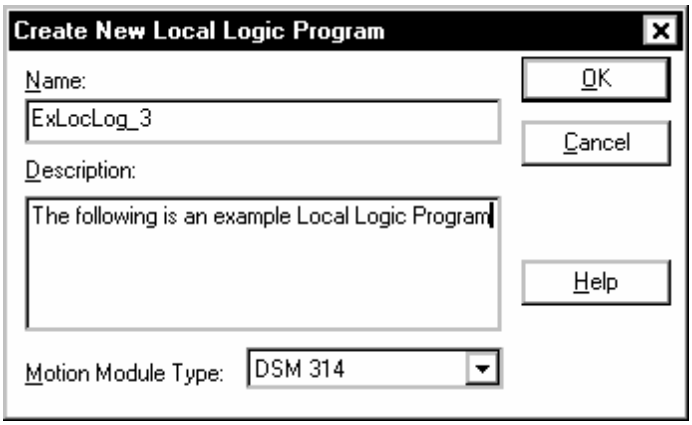
Podrobnosti, jak začít se softwarem VersaPro, najdete v "Konfigurace VersaPro" v kapitole 2.

1. Chcete-li vytvořit program lokální logiky, otevřete adresář VersaPro, zvolte menu File, pak zvolte menu New Motion a pak menu Local Logic. (Obrázek 10-2.).



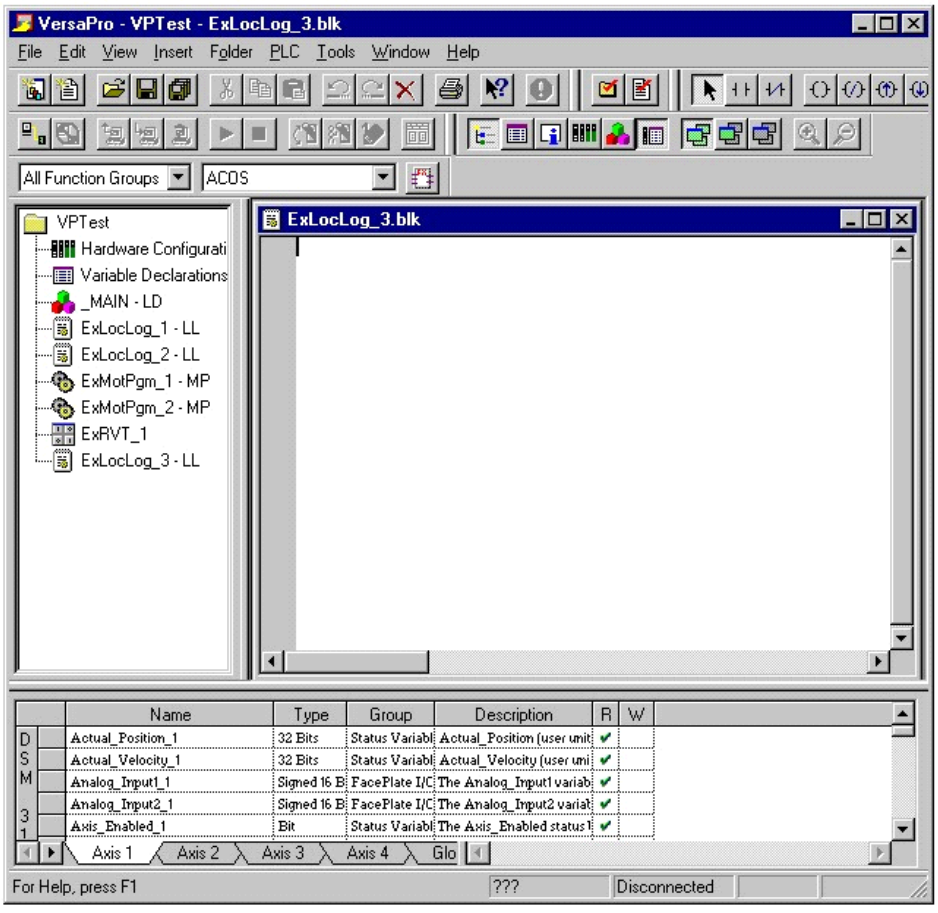
Obrázek 10-2. Vytvoření programu lokální logiky ve VersaPro

2. V dialogovém okně Create New Local Logic Program zapište název a v případě potřeby i popisné poznámky. V současné době lokální logiku podporuje pouze DSM314. Výchozí volba v Motion Module Type by se neměla měnit.



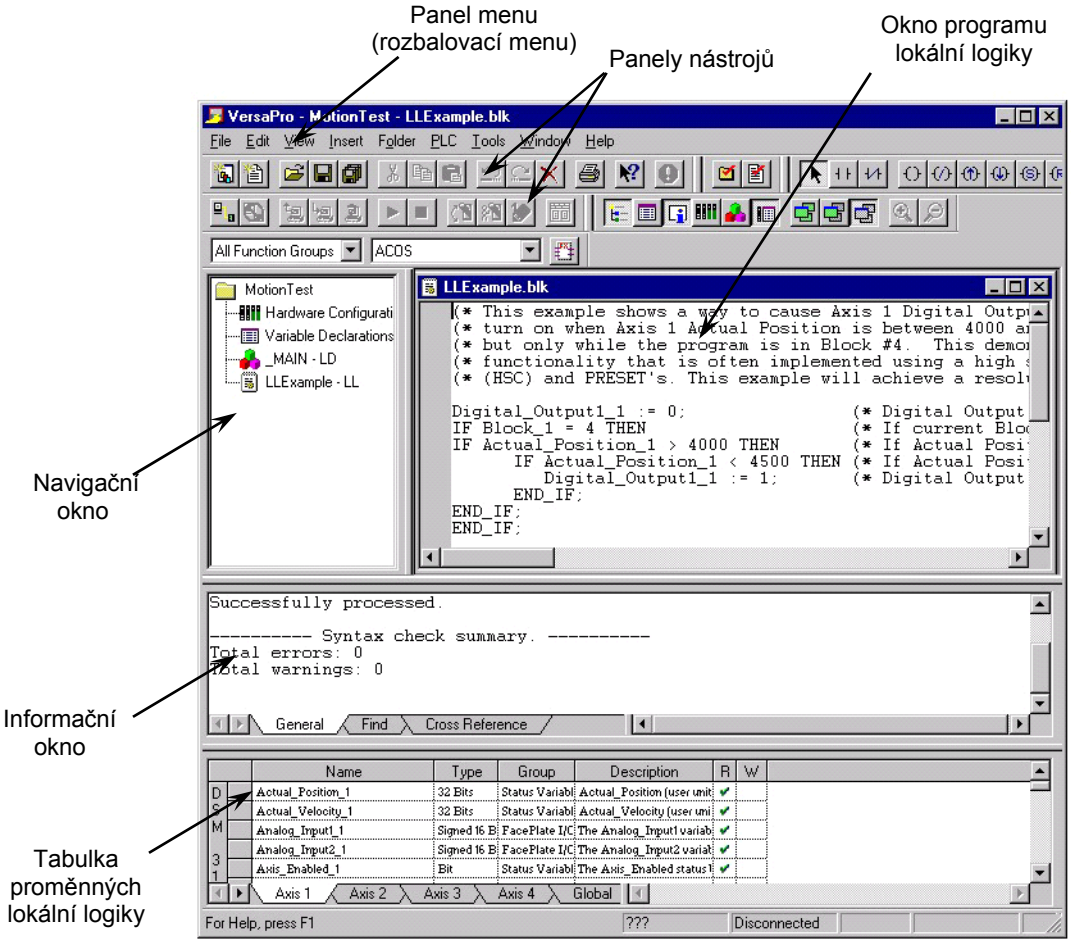
Obrázek 10-3. Vytvoření nového programu lokální logiky

3. Kliknutím na tlačítko OK se vytvoří program lokální logiky. Objeví se obrazovka editoru lokální logiky.



Obrázek 10-4. Prázdný editor lokální logiky

Uspořádání editoru VersaPro/lokální logika je znázorněno na Obrázek 10-5. .

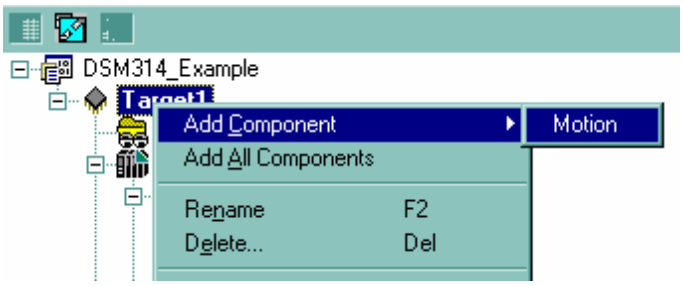


Obrázek 10-5. Uspořádání hlavní obrazovky editoru lokální logiky VersaPro

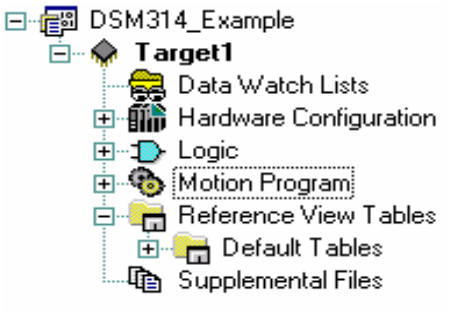
Software CIMPLICITY Machine Edition

Podrobnosti k úvodu do CIMPLICITY Machine Edition najdete v odstavci “Konfigurace CIMPLICITY Machine Edition” v kapitole 2.

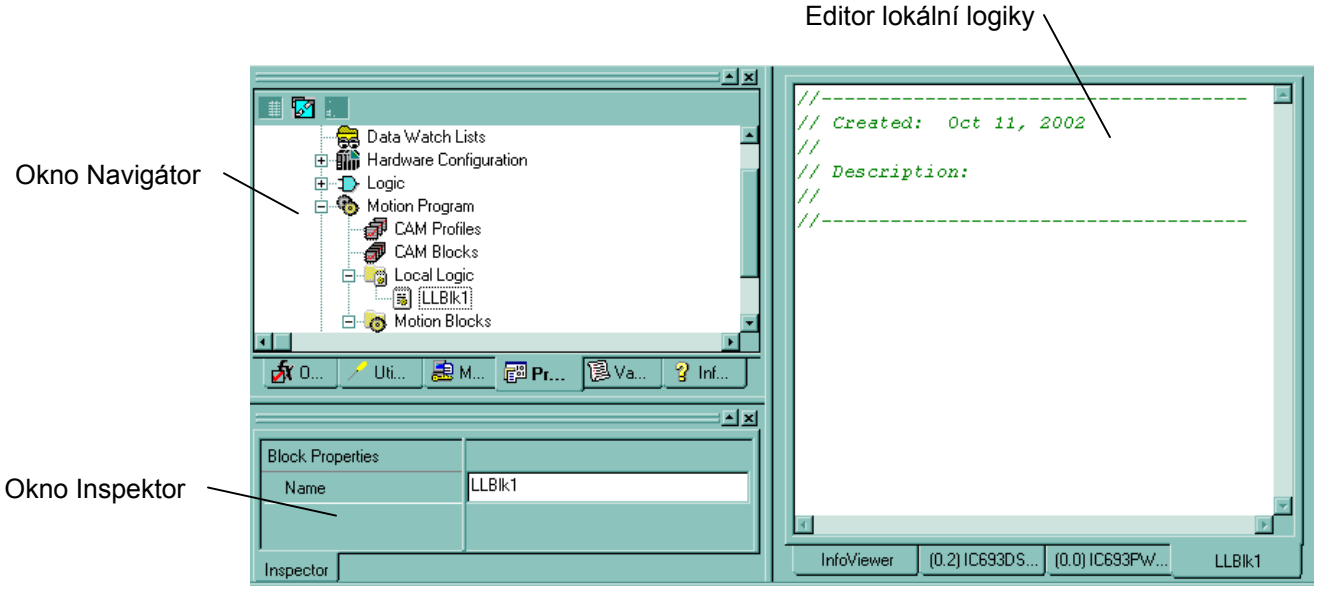
1. Chcete-li vytvořit program lokální logiky, otevřete program v CIMPLICITY Machine Edition.
2. Na záložce Project v okně Navigátor klikněte pravou myší na Target (Cíl) obsahující DSM314, zvolte Add Component (Přidat komponentu) a pak zvolte Motion (Pohyb).



V okně Navigátor se objeví adresář Motion Program.



3. Rozbalte adresář Motion Program. Vyberte Local Logic a zvolte New. V adresáři Local Logic se vytvoří blok lokální logiky a otevře se editor lokální logiky.
4. Chcete-li změnit název svého bloku lokální logiky, proveďte editaci jeho názvu v Block Properties, který se zobrazí v okně Inspektor.



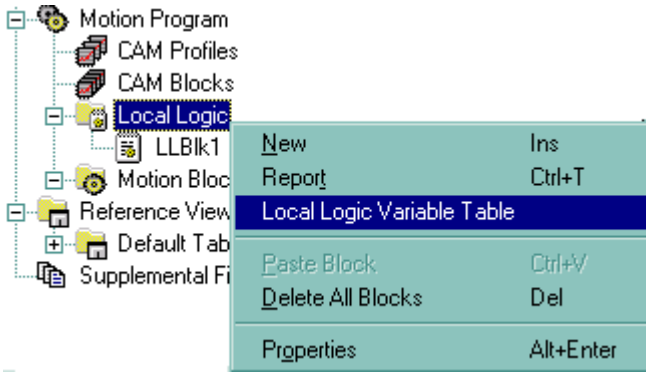
Obrázek 10-6. Uspořádání hlavní obrazovky editoru lokální logiky, CIMPLICITY Machine Edition

Tabulka proměnných lokální logiky

Programovací prostředí zahrnuje okno, které obsahuje proměnné lokální logiky. Tabulka proměnných lokální logiky (LLVT) umožňuje přesunout text z tabulky do programu pomocí operací "drag and drop" nebo "cut and paste". (Viz obrázek 10-7.).

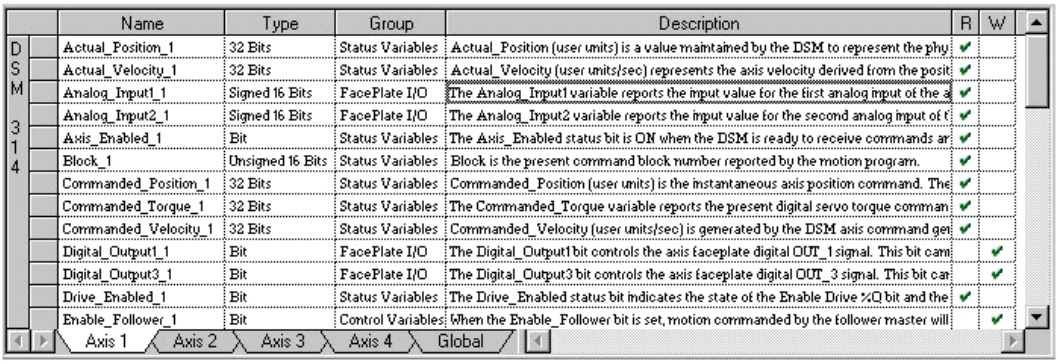
- Chcete-li otevřít LLVT ve **VersaPro**, zvolte tabulku proměnných lokální logiky z menu View, stiskněte Alt + 6 nebo na panelu nástrojů klikněte na tlačítko Přepnout tabulku proměnných lokální logiky.

- Chcete-li otevřít LLVT v **CIMPLICITY Machine Edition,**, klikněte pravou myší na adresář Local Logic v okně Navigátor a zvolte tabulku proměnných lokální logiky.



Tabulka má několik záložek, které dělí proměnné podle kategorií. Tyto kategorie jsou:

- Axis 1 – Proměnné týkající se osy číslo jedna
- Axis 2 – Proměnné týkající se osy číslo dvě
- Axis 3 – Proměnné týkající se osy číslo tři
- Axis 4 – Proměnné týkající se osy číslo čtyři
- Global – Globální data například Stavový kód modulu
- Bity CTL – Všeobecné řídicí/stavové bity DSM
- Registry parametrů – Registry parametrů DSM



DSM 314

Name	Type	Group	Description	R	W
Actual_Position_1	32 Bits	Status Variables	Actual_Position (user units) is a value maintained by the DSM to represent the phy	✔	
Actual_Velocity_1	32 Bits	Status Variables	Actual_Velocity (user units/sec) represents the axis velocity derived from the posit	✔	
Analog_Input1_1	Signed 16 Bits	FacePlate I/O	The Analog_Input1 variable reports the input value for the first analog input of the a	✔	
Analog_Input2_1	Signed 16 Bits	FacePlate I/O	The Analog_Input2 variable reports the input value for the second analog input of t	✔	
Axis_Enabled_1	Bit	Status Variables	The Axis_Enabled status bit is ON when the DSM is ready to receive commands ar	✔	
Block_1	Unsigned 16 Bits	Status Variables	Block is the present command block number reported by the motion program.	✔	
Commanded_Position_1	32 Bits	Status Variables	Commanded_Position (user units) is the instantaneous axis position command. The	✔	
Commanded_Torque_1	32 Bits	Status Variables	The Commanded_Torque variable reports the present digital servo torque comman	✔	
Commanded_Velocity_1	32 Bits	Status Variables	Commanded_Velocity (user units/sec) is generated by the DSM axis command ge	✔	
Digital_Output1_1	Bit	FacePlate I/O	The Digital_Output1 bit controls the axis faceplate digital OUT_1 signal. This bit cam		✔
Digital_Output3_1	Bit	FacePlate I/O	The Digital_Output3 bit controls the axis faceplate digital OUT_3 signal. This bit can		✔
Drive_Enabled_1	Bit	Status Variables	The Drive_Enabled status bit indicates the state of the Enable Drive %Q bit and the	✔	
Enable_Follower_1	Bit	Control Variables	When the Enable_Follower bit is set, motion commanded by the follower master will		✔

Axis 1 | Axis 2 | Axis 3 | Axis 4 | Global

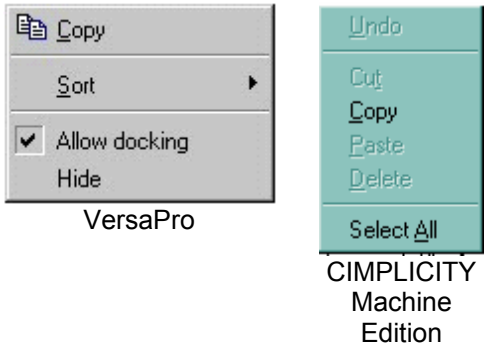
Obrázek 10-7. Tabulka prohlížení proměnných lokální logiky

Tabulka má šest sloupců. Sloupce jsou následující:

- **Name (název)** – Tento sloupec obsahuje název proměnné, který je platný pro použití v programu lokální logiky.
- **Type (typ)** – Udává typ dat pro tuto proměnnou. Například 32 Bits znamená, že proměnná má 32 bitů.
- **Group (skupina)** – Udává skupinu, ve které je tato proměnná umístěná. Například FacePlate I/O znamená, že tato proměnná se vztahuje k bodu na čelní desce modulu.
- **Description (popis)** – Tento sloupec obsahuje textový popis proměnné. Pokud se budete ukazatelem myši pohybovat nad popisem, bude se generovat špička nástroje, která umožní snadné přečtení popisu.
- **R** – Tento sloupec udává, jestli program lokální logiky může proměnnou přečíst.
- **W** – Tento sloupec udává, jestli program lokální logiky může proměnnou zapsat.

Pokud ve VersaPro budete chtít změnit velikost sloupce v tabulce proměnných lokální logiky, nastavte ukazatel myši na horní pravý okraj sloupce. Kurzor se změní na vertikální čáru se šipkami. Stiskněte primární tlačítko myši a táhnutím nastavte potřebnou šířku sloupce.

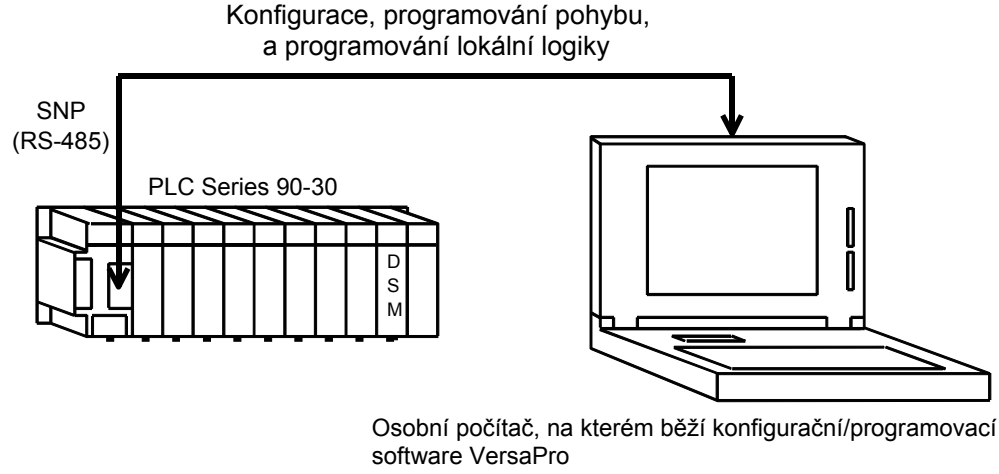
Tabulka má plovoucí menu, do kterého je přístup po stisknutí pravého tlačítka myši, když ukazatel bude nastavený nad tabulkou. Rozbalovací menu je znázorněno na obrázku 10-8. Chcete-li v CIMPLICITY Machine Edition tabulku uspořádat, klikněte na záhlaví sloupce, podle kterého jí chcete uspořádat.



Obrázek 10-8. Rozbalovací menu LLVT

Připojení editoru lokální logiky k DSM

Konfigurační/programovací software má několik možností komunikace. Nejčastěji používaná volba komunikace je přímé připojení k SNP portu PLC znázorněné na obrázku 10-14 níže. Je také možno použít Ethernet. Veškeré programování DSM314 se provádí přes softwarové rozhraní, které vede na jediné místo programování modulu. (DSM314 má také sériový port na čelní desce modulu, který se používá pouze pro aktualizaci firmwaru DSM314.) Programy lokální logiky a pohybové programy ukládá VersaPro do vyhrazené paměti uvnitř CPU PLC. DSM314 pak volá tyto programy během konfigurace z CPU podle názvů. Spoj k programům, které DSM314 vyžaduje od CPU, je obsažený v hardwarové konfiguraci sestavy PLC. Výhodou je, že programy nejsou určené konkrétním modulem, ale jsou určené sestavou/pozicí. Proto pokud bude nutné uvnitř PLC prohodit DSM314 nebo vyměnit DSM314, je nutno pouze provést následující kroky: (1) vypnout napájení PLC, (2) vyměnit moduly DSM314 a (3) znovu zapnout napájení PLC. Po zapnutí napájení PLC pošle do DSM314 správné programy a nastavení konfigurace.



Obrázek 10-9. Schéma zapojení programovacího zařízení

Vytvoření programu lokální logiky

Programovací software vytváří nezávislé prostředí, které umožňuje vykonávat všechny kroky potřebné k vytvoření, editování a načtení programu lokální logiky do modulu DSM314.

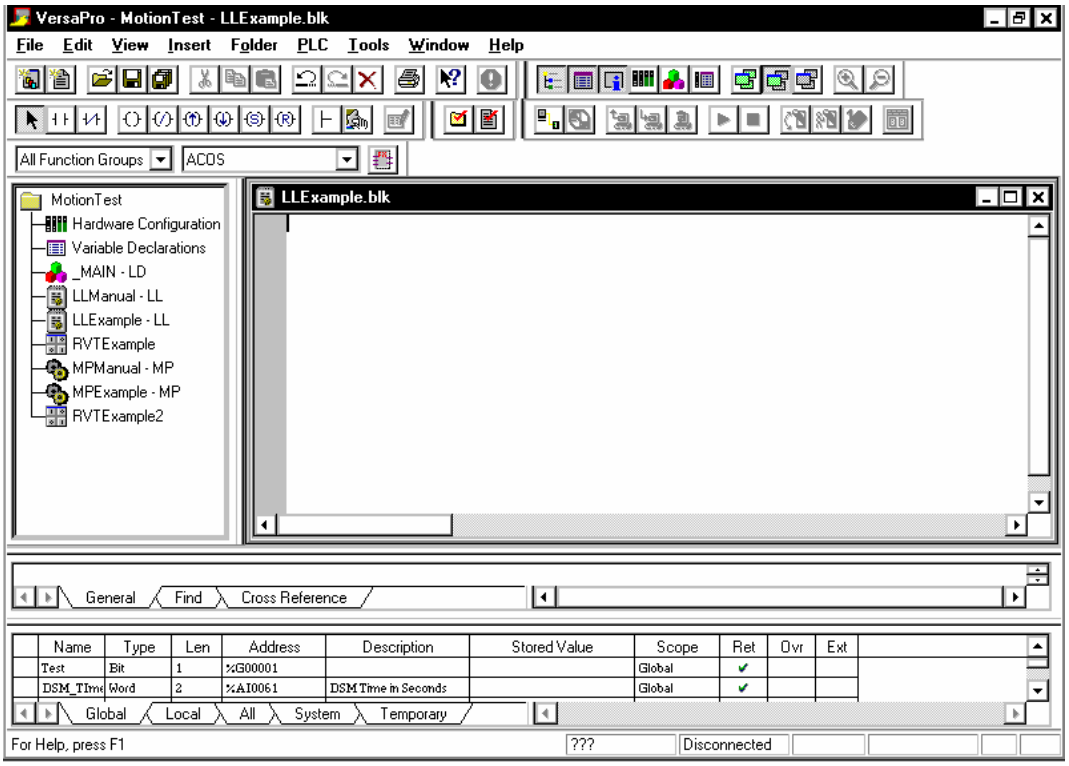
Vytvoření programu lokální logiky

Vytvořte program lokální logiky s názvem LLExample. Podrobnosti, jak to provést, najdete v:

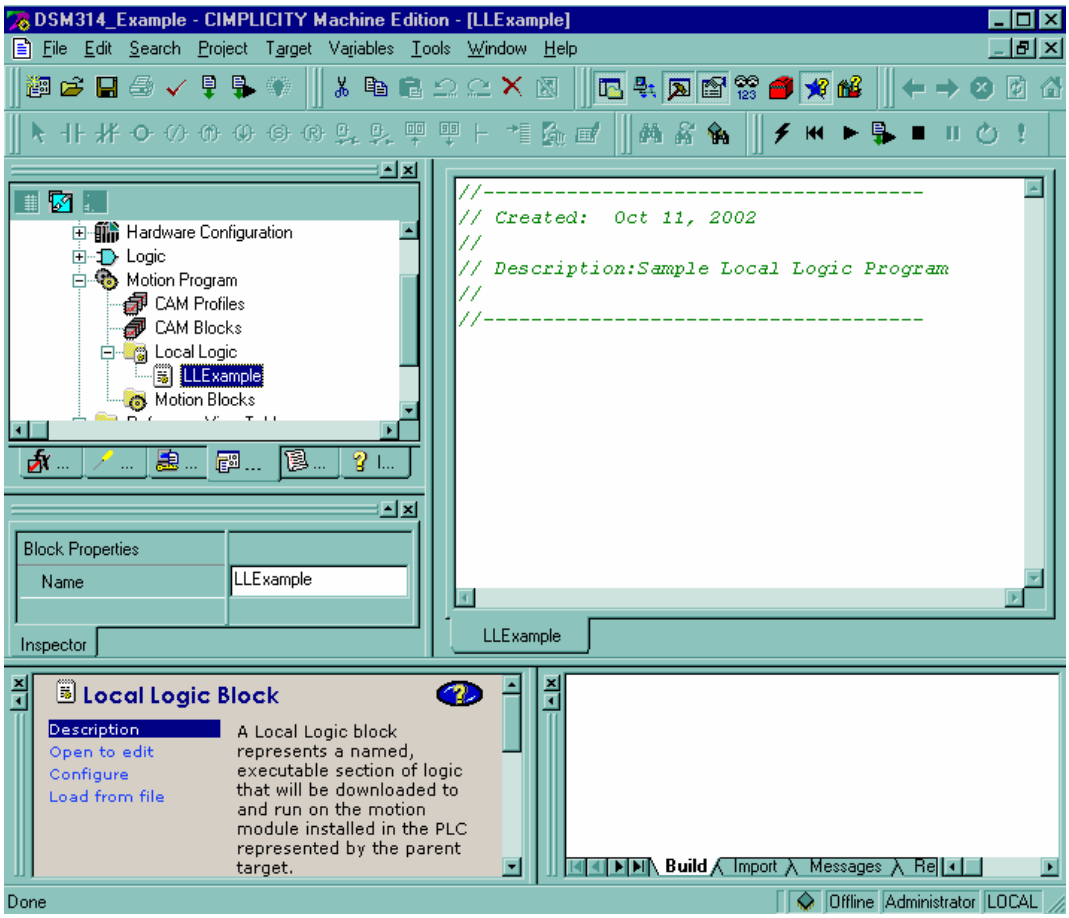
"VersaPro" na straně 10-5

"CIMPLICITY Machine Edition" na straně 10-8

Výsledné zobrazení vypadá přibližně jako na obrázku 10-16 a 10-17.



Obrázek 10-10. Nový program lokální logiky VersaPro



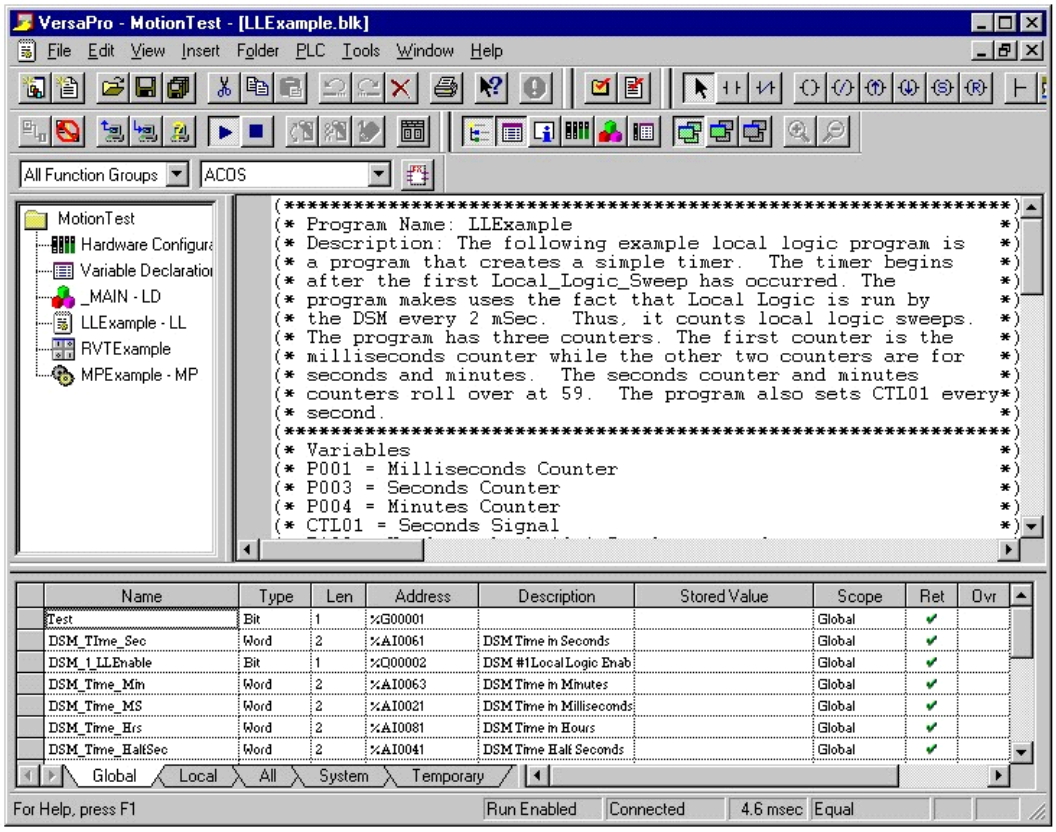
Obrázek 10-11. Nový program lokální logiky CIMPLICITY Machine Edition

Editor lokální logiky je neformátový textový editor, který umožňuje zapisovat programy ve stylu, který vám vyhovuje. Tento příklad je velmi jednoduchý program lokální logiky, který nepředstavuje plně funkční aplikaci, protože je určený pouze pro účely cvičení. Příklad programu je jednoduchá aplikace časovače, který spočívá ve vzorkovací periodě polohové smyčky digitálního serva (2 ms) jako časové základny. Vzorkovací periody polohovací smyčky pro jiné konfigurace najdete v kapitole 1.

Příklad programu lokální logiky

```
(*********************************************************************)
(* Název programu: LLExample
      *)
(* Popis: Následující příklad programu lokální logiky je program,  *)
(* který vytváří jednoduchý časovač. Časovač začíná po výskytu     *)
(* prvního Local_Logic_Sweep. Program využívá skutečnosti,         *)
(* že Lokální logika se vykonává v DSM každé 2 ms.                 *)
(* Proto počítá cykly lokální logiky. Program má tři čítače.       *)
(* První čítač je milisekundový čítač, zatímco druhé dva           *)
(* čítače jsou pro sekundy a minuty. Sekundový čítač a minutový    *)
(* čítač se přetáčejí na 59. Program také nastavuje CTL01 každou   *)
(* sekundu.                                                        *)
(*********************************************************************)
(* Proměnné                                                        *)
(* P001 = Milisekundový čítač                                      *)
(* P003 = Sekundový čítač                                          *)
(* P004 = Minutový čítač                                           *)
(* CTL01 = Sekundový signál                                        *)
(* P100 = Používá se ke kontrole, jestli uplynula 1 sekunda        *)
(* P102 = Používá se ke kontrole, jestli uplynula 1 minuta         *)
(*********************************************************************)
IF First_Local_Logic_Sweep THEN   (* První cyklus vykonávání *)
      P001 := 0;                  (* Inicializace P001 na 0 *)
      P003 := 0;                  (* Inicializace P003 na 0 *)
      P004 := 0;                  (* Inicializace P004 na 0 *)
END_IF;

P001:=P001+2;              (* Čas v milisekundách *)

P100:= P001 MOD 1000;      (* Kontrola, jestli uplynula 1 s (1000 ms)) *)
IF P100 = 0 THEN           (* Zbytek operace MOD =0 uplynula 1 s *)
       P003:=P003+1;       (* Čas v sekundách *)
       CTL01 := 1;
       IF P003 = 60 THEN   (* Pokud sekundy = 60, pak začít znovu na 0 *)
              P003:=0;
       END_IF;
END_IF;
IF P100 <> 0 THEN
       CTL01 :=0;          (* CTL01=0 Když se neinkrementuje sekundový
čítač *)
END_IF;

P101:=P001 MOD 60000;      (* Kontrola, jestli uplynula 1 min (60000 ms)*)
IF P101 = 0 THEN           (* Zbytek operace MOD =0 uplynula 1 Min *)
       P004:=P004+1;       (* Čas v minutách *)
       IF P004 = 60 THEN   (* Pokud minuty = 60, pak začít znovu na 0 *)
              P004:=0;
       END_IF;
END_IF;
```

Jakmile zapíšete výše uvedený program do textového editoru, obrazovka bude vypadat asi jako na Obrázek 10-12. .



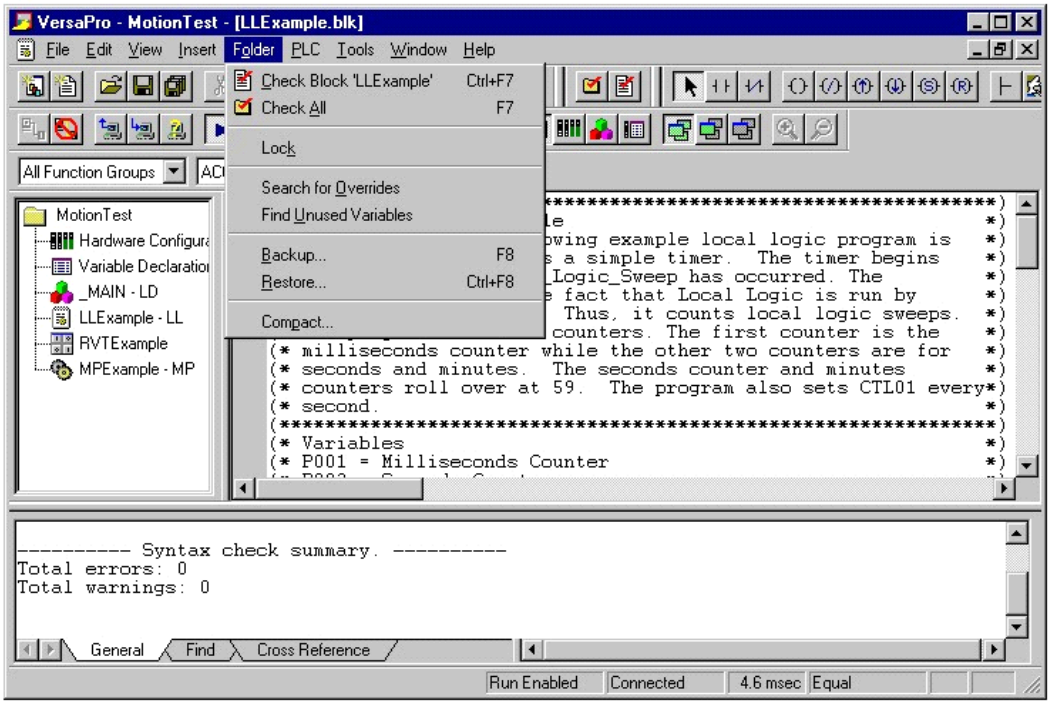
Obrázek 10-12. Lokální logika (LLExample)

Kontrola syntaxe lokální logiky

Nyní byste měli zkontrolovat správnou jazykovou syntaxi.

VersaPro

Chcete-li zkontrolovat jazykovou syntaxi ve VersaPro, z hlavního menu zvolte Folder, pak podmenu Check Block 'LLExample'. Tím se spustí programy pro kontrolu syntaxe zadaného programu lokální logiky. **Poznámka:** Chcete-li zkontrolovat všechny bloky v adresáři, zvolte Check All. (Obrázek 10-13.)



Obrázek 10-13. Povel kontroly syntaxe LLExample

Pokud nebylo zobrazeno dříve, kontrola bloků vyvolá zobrazení informačního okna. Všimněte si, že informační okno je možno zapínat a vypínat stisknutím ikony  na panelu nástrojů.

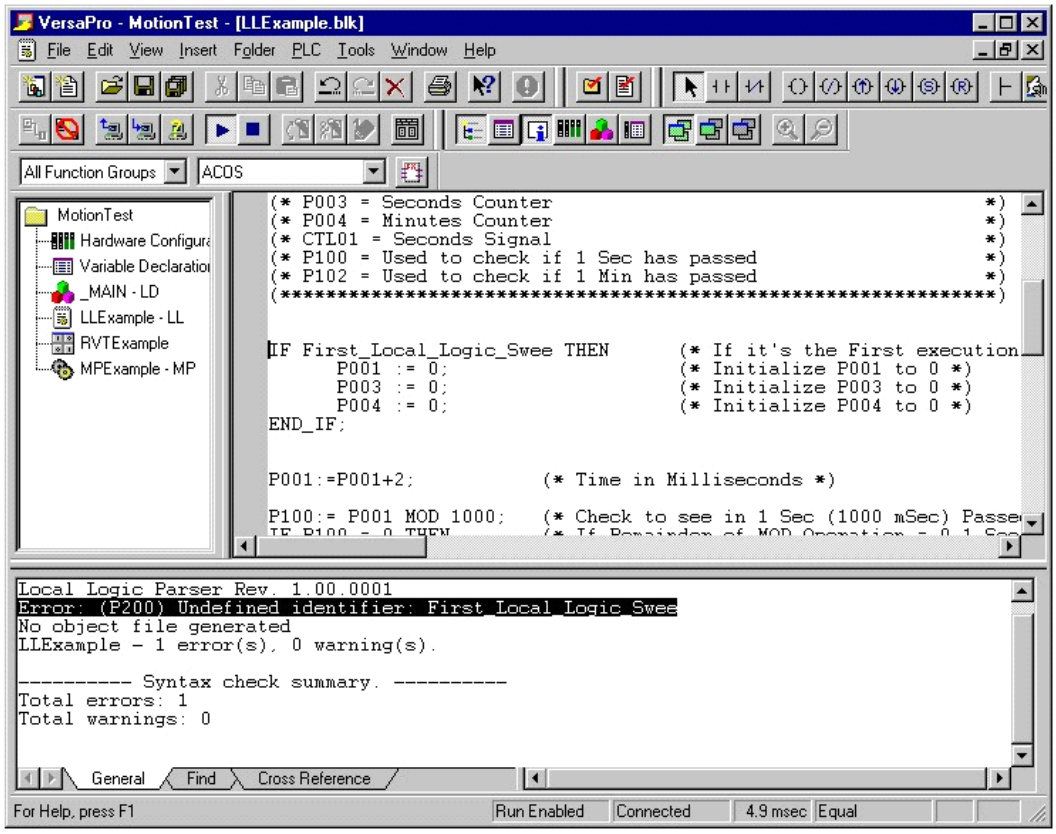
Informační okno zobrazí výstup operace kontroly syntaxe. Pokud vzorový program byl zapsaný správně, měli byste dostat zprávu s nulovým počtem chyb a s žádnou výstrahou.

Pokud informační okno bude indikovat, že se vyskytly chyby syntaxe, informačním oknem můžete rolovat na řádek, který obsahuje chybové hlášení. Když informační okno je zvětšené, klikněte dvakrát na chybové hlášení. Tím okno editoru automaticky přejde na řádek v programu, který způsobil chybu. Pokud například v příkladu programu místo "First_Local_Logic_Sweep" bude nesprávně napsáno "First_Local_Logic_Swee", bude se generovat chyba syntaxe. Asi takto:

```
Error: (P200) Undefined identifier: First_Local_Logic_Swee
```

Pak můžete přejít na chybové hlášení v informačním okně a dvakrát kliknout na řádek. Editor lokální logiky automaticky přejde na začátek řádku, který způsobil chybové hlášení, takže máte možnost chybu opravit tak, jak je znázorněno na obrázku 10-31.

Pokud právě budete editovat program lokální logiky a budete chtít najít přibližné číslo řádku, klikněte na vertikální rolovací lištu na pravé straně editoru lokální logiky. Zobrazí se číslo řádku na horním okraji okna. Pokud kurzor bude několik řádků pod hořejškem okna, buď rolujte oknem, až aktuální řádek bude u horního okraje okna, nebo odpočítejte počet řádků od shora a přičtěte je k aktuálnímu číslu řádku.



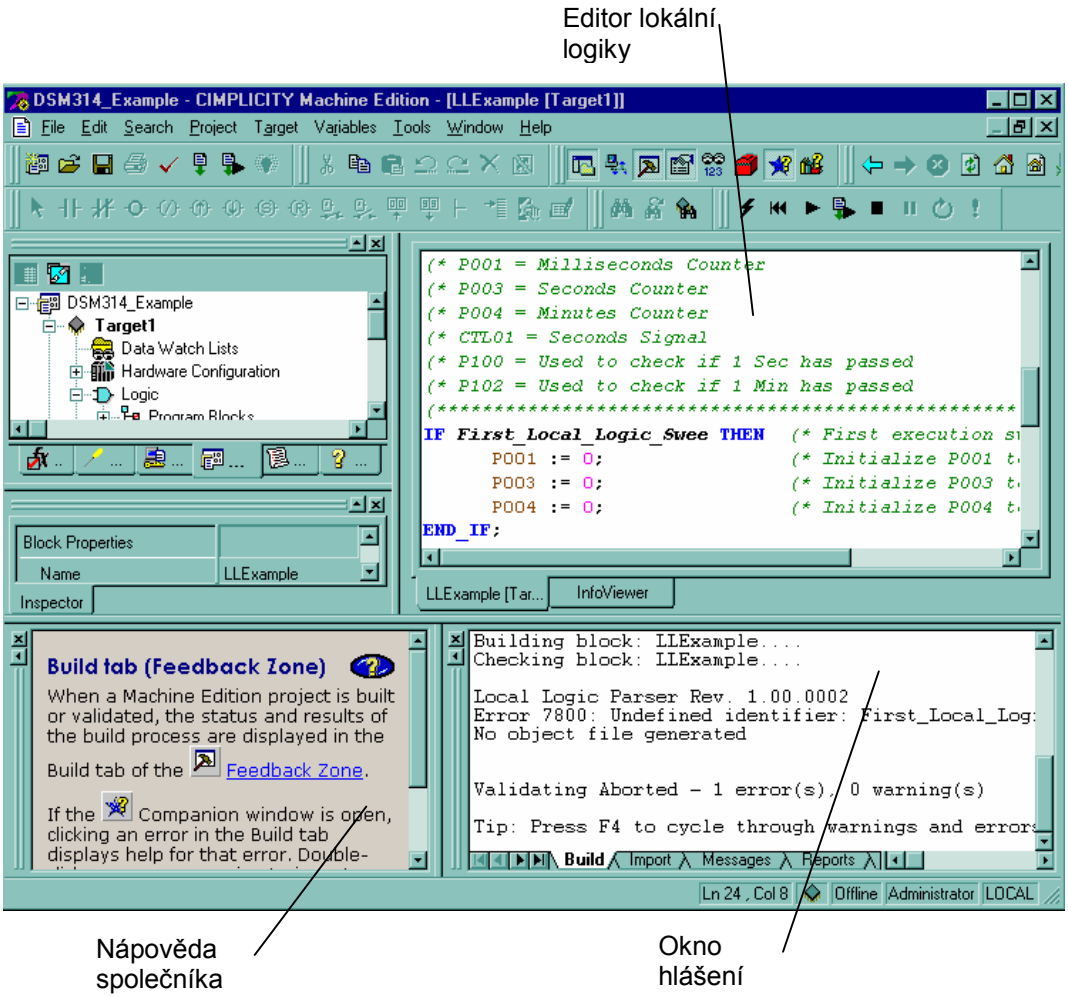
Obrázek 10-14. Chyba kontroly syntaxe LLExample

Kapitola 12 obsahuje další podrobnosti, které uvádějí neúspěšnou kontrolu syntaxe a nápravné kroky.

CIMPLICITY Machine Edition

Chcete-li zkontrolovat jazykovou syntaxi, zvolte Target (cíl), pak Validate <Název cíle>. Také můžete kdekoliv v okně CIMPLICITY Machine Edition stisknout F7. Zkontrolují se všechny bloky logiky aktivního cíle. Výsledky kontroly syntaxe se zobrazí v Okně hlášení. (Pokud Okno hlášení nebude ještě otevřené, otevře se spuštěním procesu Validate.)

V následujícím příkladu je řádek “First_Local_Logic_Sweep” nesprávně napsaný jako “First_Local_Logic_Swee”.



Tipy: Chcete-li si prohlížet výstražná a chybová hlášení v Okně hlášení, stiskněte F4.

Chcete-li přejít na řádek, který způsobil chybu v programu lokální logiky, klikněte dvakrát na popis chyby v Okně hlášení. Zvětšení se posune do okna editoru lokální logiky a kurzor se přemístí na začátek řádku, který obsahuje chybu.

Kapitola 12 obsahuje další podrobnosti a nápravné kroky neúspěšné kontroly syntaxe.

Nastavení hardwarové konfigurace pro lokální logiku

Po úspěšné kontrole syntaxe potřebujete nastavit hardwarovou konfiguraci, která umožní načtení vzorového programu do správného modulu DSM314. Všimněte si, že toto není typické pořadí, ve kterém se tyto kroky provádějí. Většina uživatelů nejdříve nastaví hardwarovou konfiguraci a pak vytvoří programové příkazy. Pořadí v tomto příkladu je však obrácené, aby se lépe ilustrovala vazba mezi hardwarovou konfigurací a názvem programu lokální logiky v hardwarové konfiguraci DSM314.

Podrobnosti, jak provést kroky 1 a 2, najdete v jedné z následujících částí kapitoly 2:

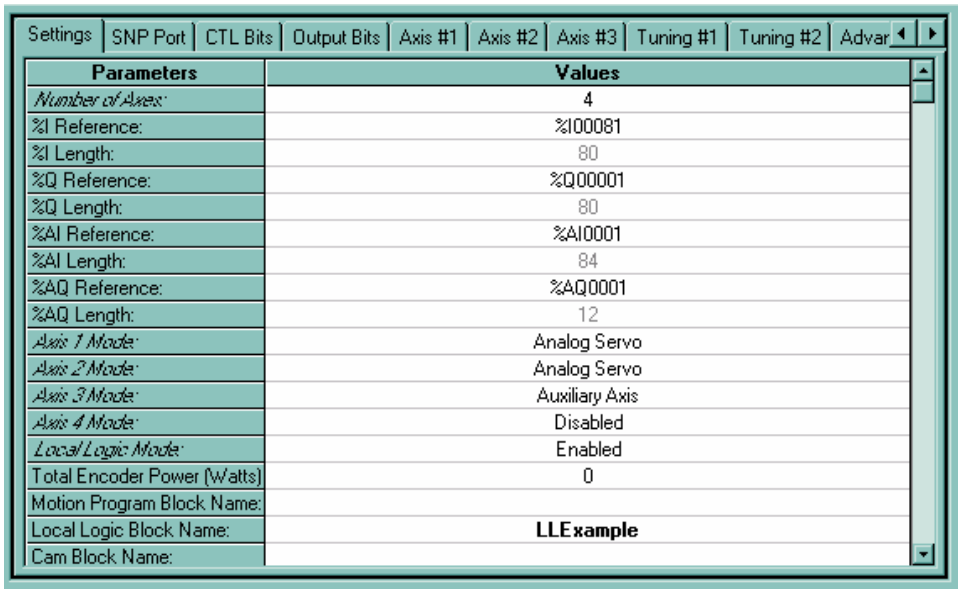
- "Konfigurace VersaPro"
- "Konfigurace CIMPLICITY Machine Edition"

1. Pokud jste to ještě neprovedli, otevřete hardwarovou konfiguraci a ve své aplikaci nakonfigurujte CPU, které podporuje firmware verze 10.0 nebo pozdější a příslušné napájení. Přidejte DSM314 do konfigurace sestavy. Tato operace přidá do sestavy DSM314 a otevře konfigurační obrazovku DSM314, které vám umožní přizpůsobit DSM314 pro vaší konkrétní aplikaci.

Poznámka: Podrobnosti ohledně konfiguračních nastavení DSM4 najdete v kapitole 4.

2. Na záložce "Settings" nastavte parametr "Local Logic Mode" na Enabled a zapište název vzorového programu "LLExample" do pole "Local Logic Block Name". Výsledná obrazovka pro konfiguraci hardwaru bude vypadat jako na Obrázek 10-15.

Poznámka: Tento způsob připojení DSM314 k programu lokální logiky umožňuje snadno specifikovat několik DSM314, které používají stejný program lokální logiky. Tento příklad má pouze jedno DSM314. Pokud však budete mít několik DSM314, které mají mít spuštěný stejný program lokální logiky, v konfiguraci pro každé DSM314 jednoduše zadejte, že má vykonat tento program. To umožní programovacímu zařízení, aby měl jeden zdrojový soubor lokální logiky pro několik DSM314. Také si všimněte, že toto neznemožní, aby DSM314 vykonávala různé programy.



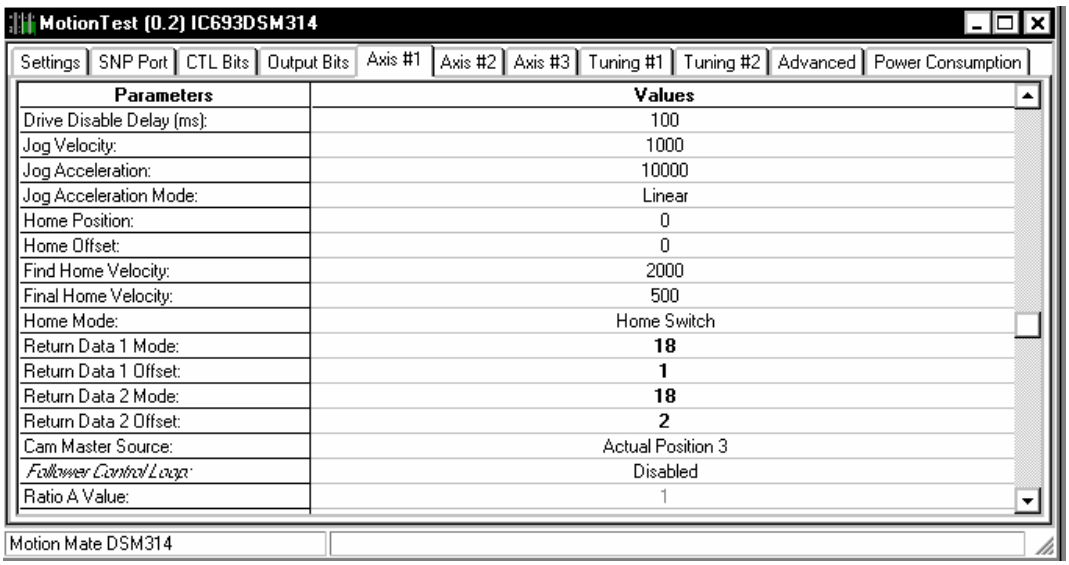
Obrázek 10-15. Záložka nastavení hardwarové konfigurace sestavy 90-30 DSM314

3. Konfigurace dat návratu

   Příklad programu lokální logiky zobrazený na straně 10-14 používá registry parametrů P001, P003 a P004 jako čítače, které obsahují hodnoty představující čas. Chcete-li si prohlédnout registry parametrů v registrech dat návratu DSM, je nutno data návratu nakonfigurovat. Nakonfigurování dat návratu:

   A. Zvolte záložku Axis #1 a v režimu Režim dat návratu 1 zapište 18. To DSM řekne, že chcete vrátit registry parametrů. V případě Offsetu dat návratu 1 zapište 1. To DSM řekne, že chcete vrátit parametr P001.

   Program LLExample vrátí P001, P003 a P004. Seskupení je však lepší, pokud v Axis #2 vrátíte P003 a P004. Proto můžete buď jako výchozí hodnoty ponechat Režim dat návratu 2 a Offset dat návratu 2 nebo do Režimu dat návratu 2 zapsat 18 a do Offsetu dat návratu 2 zapsat 2 a tím říct DSM, že má vrátit P002. Všimněte si, že Volba dat návratu 2 pro osu 1 se vrátí v paměti %AI s offsetem 21 a Data návratu 2 pro osu 1 se vrátí v paměti %AI s offsetem 23.



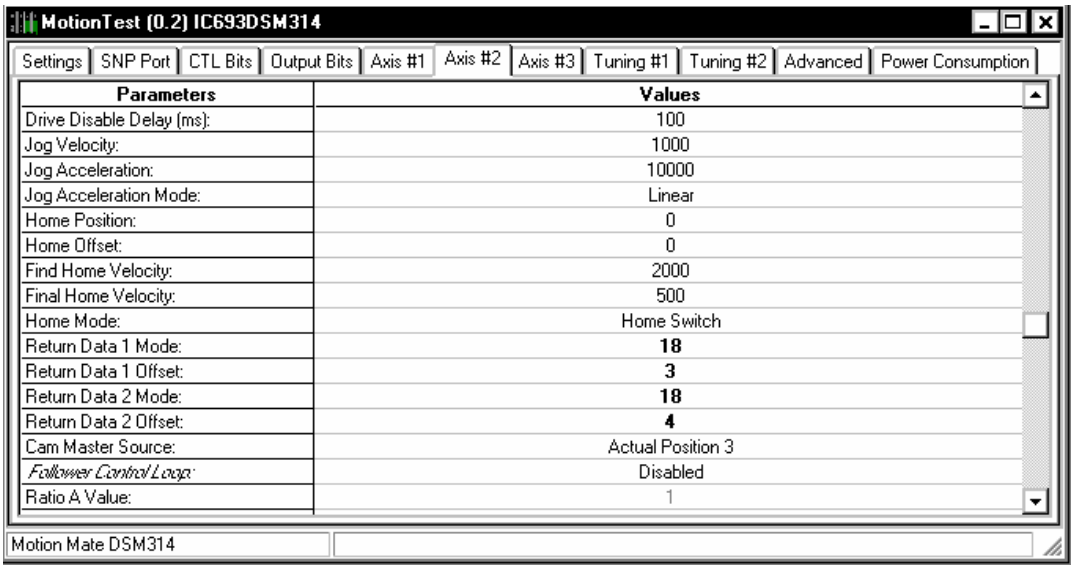
Obrázek 10-16. Záložka hardwarové konfigurace sestavy 90-30 DSM314 Axis#1

Pro P003 a P004 je nutno výše uvedené kroky zopakovat.

   B. Zvolte záložku Axis #2 a v Režimu dat návratu 1 zapište 18. To DSM řekne, že chcete vrátit registry parametrů. V případě Offsetu dat návratu 1 zapište 3. To DSM řekne, že chcete vrátit parametr P003.

   C. Na záložce Axis #2 v Režimu dat návratu 2 zapište 18 a v Offsetu dat návratu 2 zapište 2, což DSM řekne, že má vrátit P004.

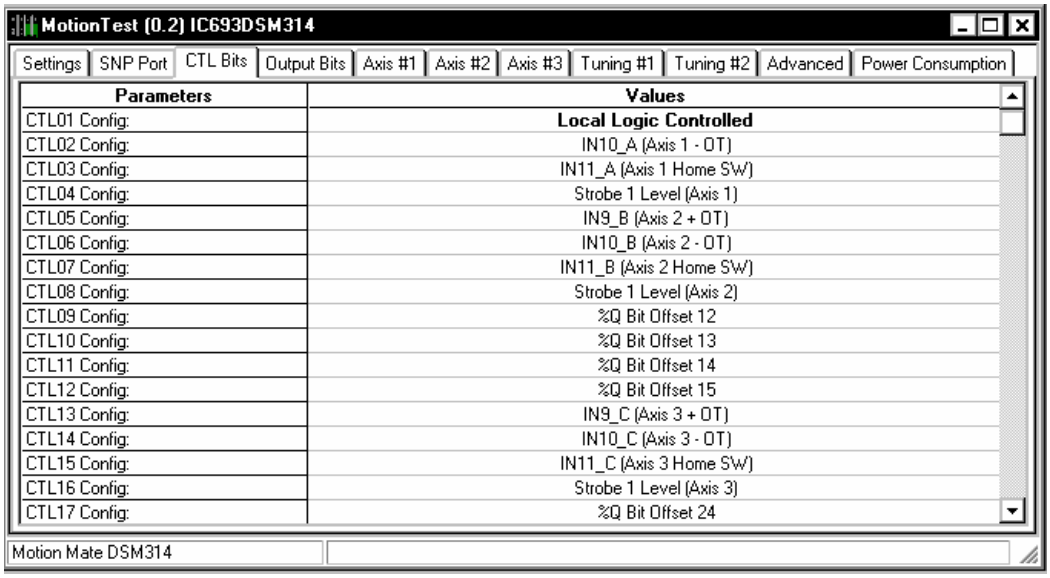
   Poznámka: Volba dat návratu 2 pro osu 2 se vrátí v paměti %AI s offsetem 41 a Data návratu 2 pro osu 2 se vrátí v paměti %AI s offsetem 43.



Obrázek 10-17. Záložka Axis#2 hardwarové konfigurace 90-30 se sestavou DSM314

4. Nakonfigurování bitu CTL.

   Vzorový příklad lokální logiky uvedený na straně 10-14 řídí CTL01, který se používá k signalizaci pohybovému programu, že uplynula jedna sekunda. Bit CTL musí být nakonfigurovaný tak, aby byl řízený lokální logikou. K tomu účelu otevřete záložku CTL Bits v hardwarové konfiguraci. Vyberte "CTL01 Config" a zvolte Local_Logic_Controlled. Výsledná záložka CTL01 je zobrazená na Obrázek 10-18.



Obrázek 10-18. Záložka CTL Bits hardwarové konfigurace sestavy 90-30 DSM314

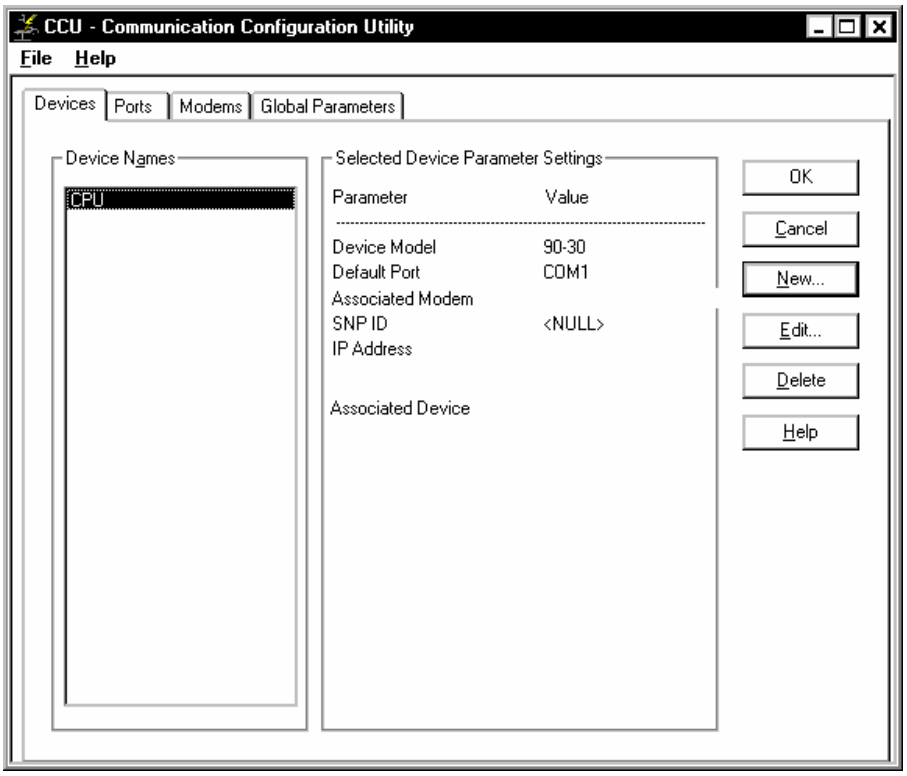
5. Tím se dokončí změny konfigurace potřebné v tomto příkladu. Uzavřete nástroj hardwarové konfigurace a uložte adresář. Spojení mezi příkladem programu lokální logiky a modulem DSM314 je nyní kompletní. Nyní můžete vytvořit libovolnou žebříkovou logiku příček PLC a pak v programech Zkontrolovat všechno.

Načtení programu lokální logiky

Chcete-li provést operaci načtení, nejdříve se přesvědčte, že komunikační port je správně nakonfigurovaný. Chcete-li otevřít nastavení konfigurace, z hlavního menu zvolte Tools a pak zvolte podmenu Communications Setup … . Tato volba menu vyvolá soubor pro konfiguraci komunikace. První dialogové okno požádá o heslo. Výchozí (při dodání) heslo je **netutil** (Obrázek 10-19.).

Obrázek 10-19. Dialogové okno hesla pro nastavení komunikace

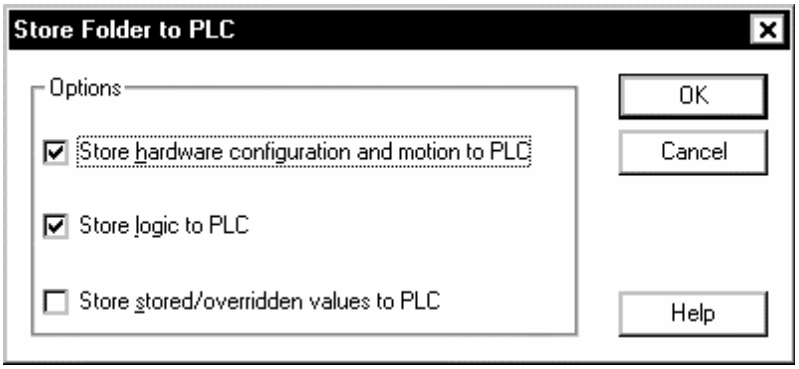
Po zápisu hesla se vyvolá obslužný program pro nakonfigurování komunikace (Obrázek 10-20.). VersaPro k zajištění ovladačů komunikace používá Host Communications Toolkit (HCT). Další podrobnosti týkající se správného použití tohoto obslužného programu čtenář najde v "GFK-1026 – Obslužný program pro nakonfigurování komunikace GE Fanuc". Pro obslužný program je on-line nápověda, která může zodpovědět mnoho obecných otázek.



Obrázek 10-20. Obslužný program pro nakonfigurování komunikace

Po nakonfigurování komunikačního portu je možno do CPU PLC načíst (uložit) program lokální logiky. Chcete-li do PLC uložit aktuální adresář, na liště Menu zvolte PLC a z podmenu zvolte Connect. Po připojení z lišty Menu zvolte PLC a z podmenu zvolte Store. Operace uložení spustí proces přesunu adresáře z programovacího zařízení do CPU PLC. Když uživatel vyvolá operaci uložení, otevře se dialogové okno, které umožní zvolit, co se má uložit do PLC. V tomto případě chceme uložit program lokální logiky, hardwarovou konfiguraci a nějakou logiku PLC. Má-li se provést tato operace, v dialogovém okně zvolte *Store hardware configuration and motion to the PLC* a *Store logic to PLC* (Obrázek 10-21.).

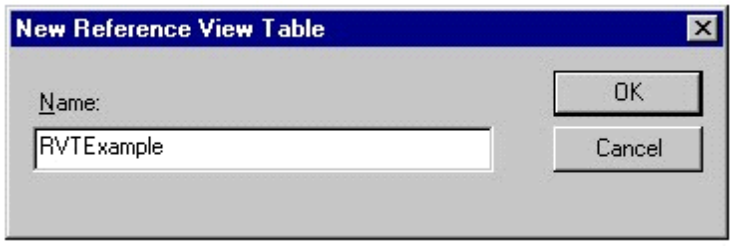
Poznámka: Programy lokální logiky a pohybové programy se přenesou jako součást procesu konfigurace hardwaru. Proto při každém načítání aktualizovaného programu lokální logiky a/nebo pohybového programu v dialogovém okně pro uložení zvolte Store hardware configuration to PLC. (Obrázek 10-21.).



Obrázek 10-21. Dialogové okno pro ukládání VersaPro

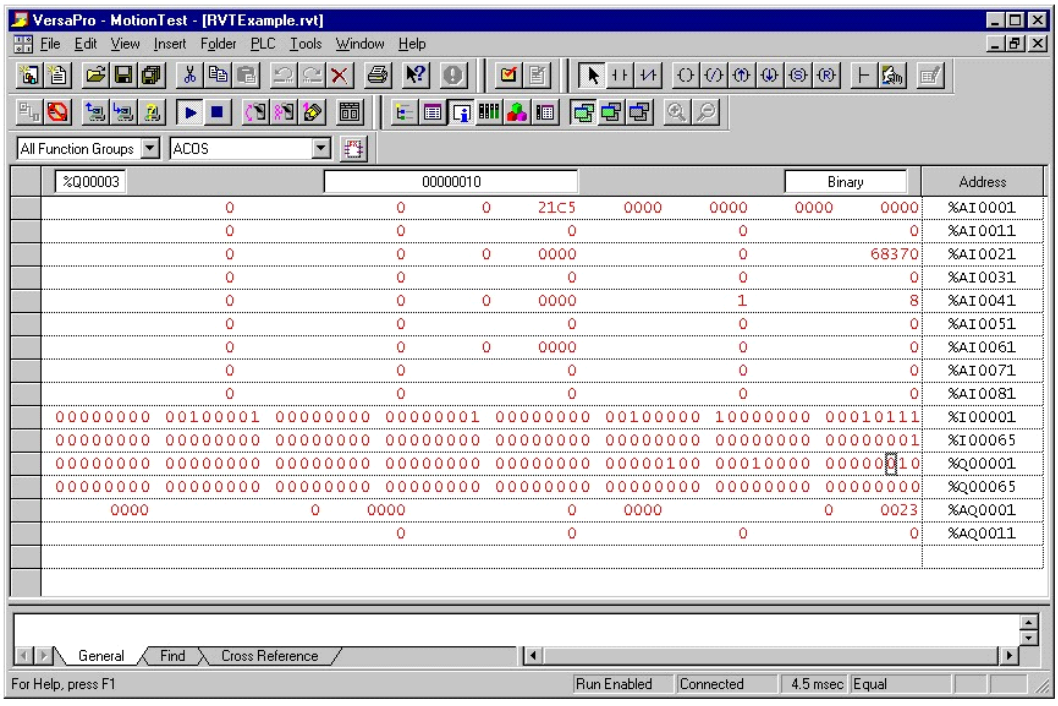
VersaPro pak zkontroluje všechny bloky, které se změnily. Pokud procedura sestavování bude úspěšná, načtou se soubory do PLC. O úspěšném načtení souborů do PLC VersaPro informuje pomocí dialogového okna s textem “Store to PLC Successful”.

Programy se nyní načtou do PLC. Spolu s DSM můžete ověřit, že program lokální logiky pracuje správně. Zobrazí se tabulka zobrazení referencí (RVT), kterou je možno použít pro tuto operaci. Chcete-li vytvořit RVT, otevřete menu File a zvolte menu New Reference View Table. (Obrázek 10-21). Tím se zobrazí dialogové okno New Reference View Table. Dialogové okno vám umožní pojmenovat RVT. V tomto příkladu bylo zvoleno RVTExample.



Obrázek 10-22. Dialogové okno New Reference View Table

Pak je možno vkládat proměnné, určovat formát zobrazování proměnných, přepínat datové body a posílat AQ povely mezi ostatními akcemi. Další podrobnosti o konstrukci RVT najdete v dokumentaci k VersaPro. Jeden příklad, který je užitečný pro tento program, je uvedený na Obrázek 10-23. .



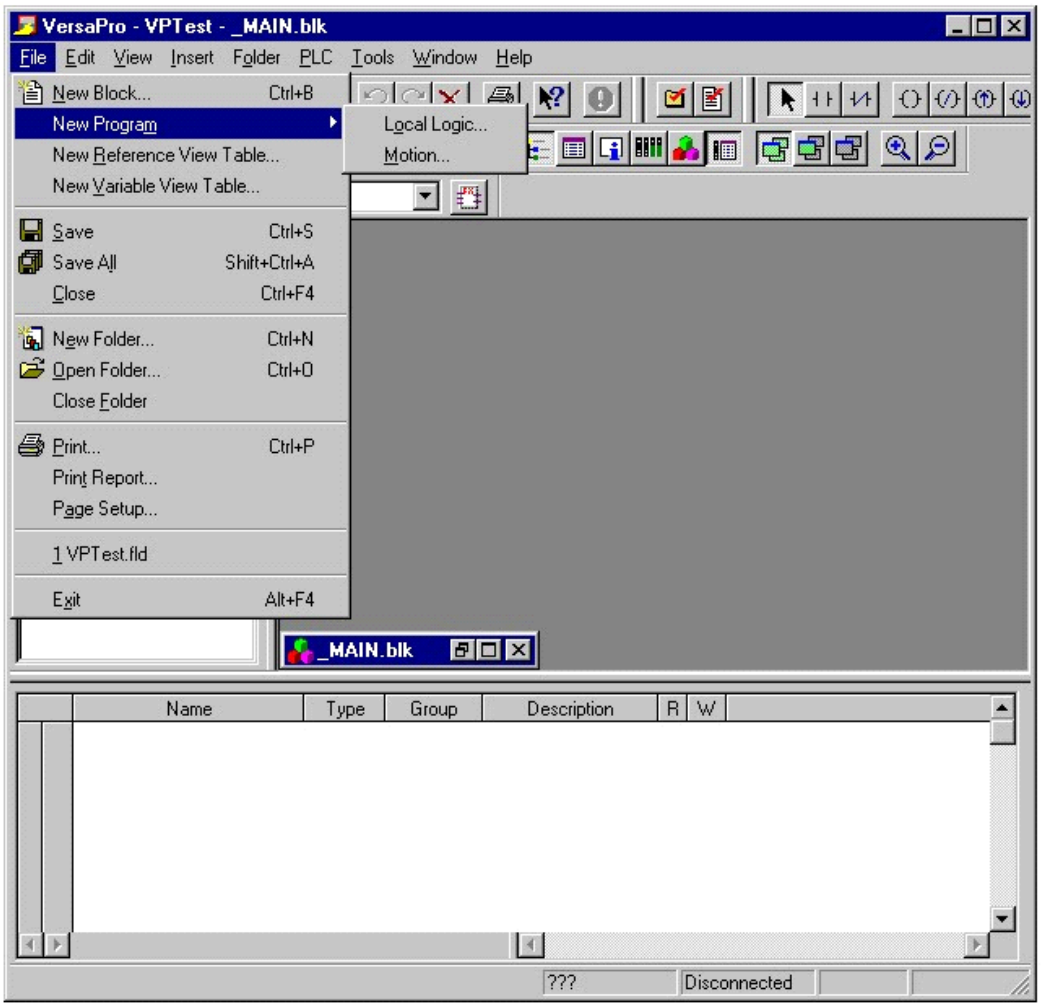
Obrázek 10-23. Reference View Table

Vykonání prvního programu lokální logiky

Jakmile se dokončí operace načítání, modul je připravený k vykonání programu lokální logiky. Má-li DSM vykonat program lokální logiky, z PLC je nutno offset bitu Q nastavit na 1, když se PLC nachází v režimu RUN. Od toho okamžiku bude program lokální logiky aktivní a poběží uvnitř DSM. **Poznámka**: Vzorový program *LLExample* je jednoduchá aplikace čítače. RVT je možno použít k prohlížení předaných parametrů a k ověření, že program je aktivní a funguje správně. Z RVT (Obrázek 10-23.) můžete vidět, že od spuštění lokální logiky uplynula 1 minuta 8 sekund (viz %AI0043 a %AI0041). Kromě toho uplynulo 68 370 milisekund, jak je vidět v %AI0021. Další podrobnosti týkající se rozhraní mezi DSM a PLC jsou uvedené v kapitole 5. Jakmile ověříte, že program pracuje správně, adresář uložte.

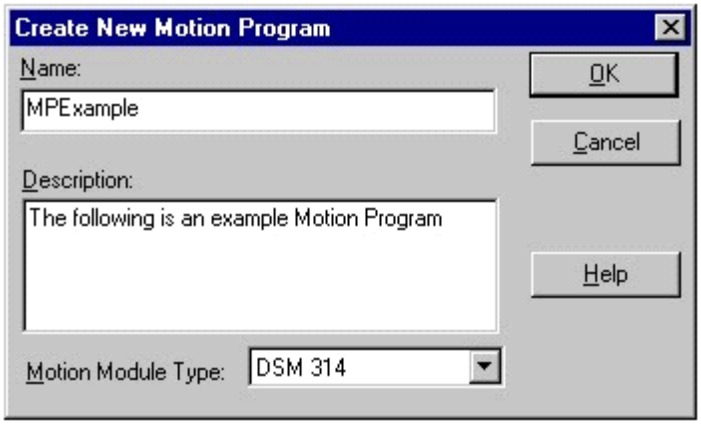
Používání editoru pohybového programu

Nyní, když jste úspěšně rozběhli program lokální logiky, bylo by dobré se připojit k pohybovému programu. Editor pohybového programu se otevře způsobem velmi podobným jako editor lokální logiky. Editor umožňuje snadné vytváření, editování, ukládání a načítání pohybových programů. Chcete-li vytvořit pohybový program, musíte mít otevřený adresář VersaPro. V tomto příkladu otevřený adresář VersaPro musí být adresář vytvořený jako součást příkladu lokální logiky v předchozí části. Název adresáře v tomto příkladu je "MotionTest". V dokumentaci k VersaPro najdete, jak otevřít adresář. Jakmile adresář VersaPro bude otevřený, z menu File zvolte menu New Program a pak zvolte menu Motion… (obrázek 10-23).



Obrázek 10-24. Menu VersaPro pro vytvoření pohybových programů

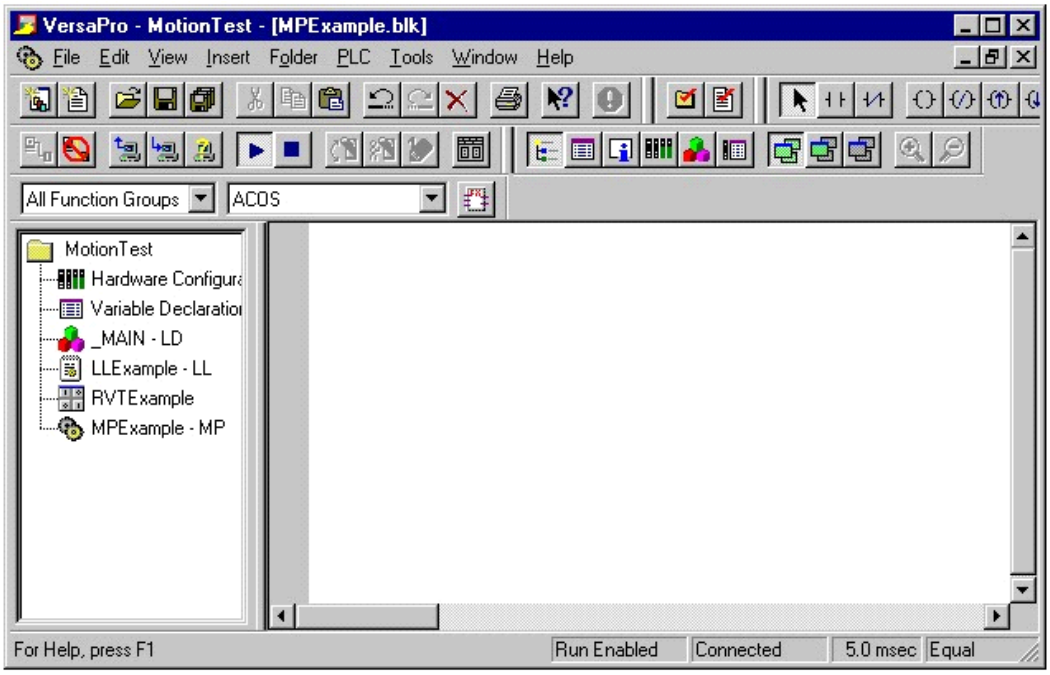
Volby z výše uvedených menu vyvolají dialogové okno, které umožňuje dát pohybovému programu název, popisný komentář a zvolit typ pohybového modulu. V současné době pohybové programy používající tento editor podporuje pouze DSM314. Výchozí volba v Motion Module Type by se neměla měnit. (Viz obrázek 10-24.)



Obrázek 10-25. Dialogové okno Create New Motion Program

Stisknutím tlačítka OK se pak vytvoří pohybový program.

Jakmile se tento krok dokončí, VersaPro vyvolá na obrazovku editor pohybu. (Obrázek 10-26.).



Obrázek 10-26. Prázdný editor pohybového programu VersaPro

Funkce menu souborů byly probrané v předchozích částech a proto v této části nebudou opakované. Tyto informace si proto najděte v předchozích částech. Editor pohybu je neformátový textový editor, který uživateli umožňuje zapisovat programy ve stylu, který mu vyhovuje. Příklad používá velmi jednoduchý program pohybu. Příklad neobsahuje funkční aplikaci a slouží pouze pro účely cvičení. Tento příklad je spojený s programem lokální logiky zapsaným v části krátkého úvodu do lokální logiky. Pro jednoduchost je opakovaný předchozí program lokální logiky:

```
(*********************************************************************)
(* Název programu: LLExample                                         *)
(* Popis: Následující příklad programu lokální logiky je program,    *)
(* který vytváří jednoduchý časovač. Časovač začíná po výskytu       *)
(* prvního Local_Logic_Sweep. Program využívá skutečnosti,           *)
(* že Lokální logika se vykonává v DSM každé 2 ms. Proto počítá      *)
(* cykly lokální logiky.                                             *)
(* Program má tři čítače. První čítač je milisekundový čítač,        *)
(* zatímco druhé dva čítače jsou pro sekundy a minuty. Sekundový     *)
(* čítač a minutový čítač se přetáčejí na 59. Program také           *)
(*nastavuje CTL01 každou sekundu.                                   *)
(*********************************************************************)
(* Proměnné                                                          *)
(* P001 = Milisekundový čítač                                        *)
(* P003 = Sekundový čítač                                            *)
(* P004 = Minutový čítač                                             *)
(* CTL01 = Sekundový signál                                          *)
(* P100 = Používá se ke kontrole, jestli uplynula 1 sekunda          *)
(* P102 = Používá se ke kontrole, jestli uplynula 1 minuta           *)
(*********************************************************************)


IF First_Local_Logic_Sweep THEN  (* První cyklus vykonávání *)
      P001 := 0;                 (* Inicializace P001 na 0 *)
      P003 := 0;                 (* Inicializace P003 na 0 *)
      P004 := 0;                 (* Inicializace P004 na 0 *)
END_IF;


P001:=P001+2;              (* Čas v milisekundách *)

P100:= P001 MOD 1000;      (* Kontrola, jestli uplynula 1 s (1000 ms)) *)
IF P100 = 0 THEN           (* Zbytek operace MOD =0 uplynula 1 s *)
       P003:=P003+1;       (* Čas v sekundách *)
       CTL01 := 1;
       IF P003 = 60 THEN   (* Pokud sekundy = 60, pak začít znovu na 0 *)
              P003:=0;
       END_IF;
END_IF;
IF P100 <> 0 THEN
       CTL01 :=0;          (* CTL01=0 Když se neinkrementuje sekundový
čítač *)
END_IF;


P101:=P001 MOD 60000;      (* Kontrola, jestli uplynula 1 Min (60000 ms)*)
IF P101 = 0 THEN           (* Zbytek operace MOD =0 uplynula 1 Min *)
       P004:=P004+1;       (* Čas v minutách *)
       IF P004 = 60 THEN   (* Pokud minuty = 60, pak začít znovu na 0 *)
              P004:=0;
       END_IF;
END_IF;
```

Program lokální logiky způsobí, že CTL01 každou sekundu přejde z logické 0 do logické 1. U tohoto jednoduchého pohybového programu se hřídel motoru otočí o 1/60 otáčky na každý přechod CTL01. Pohybový program proto přinutí, aby se hřídel motoru chovala jako druhá ručička křemenných hodin. Není to elegantní aplikace, ale znázorňuje několik koncepcí.

Než budete psát pohybový program, musíte určit měřítko osy. První proměnná, kterou je nutno určit, je poměr uživatelských jednotek na počet pulsů. Poměr uživatelských jednotek na jednotku nastaví počet programovacích jednotek na jednotku polohové zpětné vazby. To umožní naprogramovat DSM314 v jednotkách specifických pro aplikaci. Hodnoty uživatelských jednotek a jednotek musí být v rozsahu 1 až 65 535. Poměr uživatelských jednotek na jednotku musí být rozsahu 8:1 až 1:32. Pokud například se na 8192 pulsů zpětné vazby vykoná posuv 1 000 palců, poměr Uživatelská jednotka:Jednotka o hodnotě 1000:8192 bude představovat 1 Uživatelskou jednotku na 0.001 palce.

Chcete-li nastavit poměr uživatelských jednotek na počet pulsů, první požadovaný údaj je počet pulsů na otáčku zpětnovazebního zařízení. V tomto příkladu se používá motor Beta 0.5. Beta 0.5 má rozlišení zpětné vazby 8192 pulsů na otáčku. Nyní provedeme výpočet pro určení poměru. Základní rovnice je:

$$\frac{\text{User Units}}{\text{Counts}} = \left(\frac{\text{Load Movement Per Motor Rotation}}{\text{Desired Resolution}} \right) \cdot \frac{1}{\text{Encoder Counts Per Motor Rotation}}$$

Pro tento příklad:

$$\frac{\text{User Units}}{\text{Counts}} = \left(\frac{1}{\frac{1}{60}} \right) \cdot \frac{1}{8192}$$

$$\frac{\text{User Units}}{\text{Counts}} = \frac{60}{8192}$$

Tento poměr je problematický, protože porušuje pravidlo, že minimální poměr uživatelských jednotek na počet pulsů je $\frac{1}{32}$. Tento problém se snadno napraví: změňte programovací jednotky z 1/60 otáčky na 1/600 otáčky. Tím pádem se 1 programovací jednotka bude rovnat $\frac{1}{600} \cdot \text{revolution}$. Zopakujte výše uvedený výpočet:

$$\frac{\text{User Units}}{\text{Counts}} = \left(\frac{1}{\frac{1}{600}} \right) \cdot \frac{1}{8192}$$

$$\frac{\text{User Units}}{\text{Counts}} = \frac{600}{8192}$$

Aby se tedy motor pohyboval $\frac{1}{60}$ otáčky, v pohybovém programu je nutno zapsat 10 jednotek. Více informací o nastavení poměru uživatelských jednotek na počet pulsů najdete v kapitole 4.

Další údaj, který je nutno určit, je špičková rychlost motoru. To je relativně jednoduchý výpočet.

$$\text{TopSpeed}\cdot\left(\frac{\text{uu}}{\text{sec}}\right)=\text{Motor Top Speed}\left(\frac{\text{rev}}{\text{sec}}\right)\cdot\text{Enc. Counts per Rev}\cdot\left(\frac{\text{cnts}}{\text{rev}}\right)\cdot\frac{\text{User Units}}{\text{Counts}}\cdot\left(\frac{\text{uu}}{\text{cnts}}\right)$$

$$\text{TopSpeed}\cdot\left(\frac{\text{uu}}{\text{sec}}\right)=\frac{3000}{60}\cdot\left(\frac{\text{rev}}{\text{sec}}\right)\cdot 8192\cdot\left(\frac{\text{cnts}}{\text{rev}}\right)\cdot\frac{600}{8192}\cdot\left(\frac{\text{uu}}{\text{cnts}}\right)$$

$$\text{TopSpeed}\cdot\left(\frac{\text{uu}}{\text{sec}}\right)=3000\cdot 10\cdot\left(\frac{\text{uu}}{\text{sec}}\right)$$

$$\text{TopSpeed}\cdot\left(\frac{\text{uu}}{\text{sec}}\right)=30000\cdot\left(\frac{\text{uu}}{\text{sec}}\right)$$

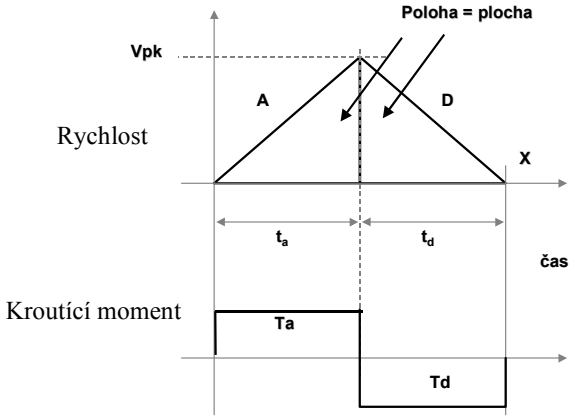
Dále je nutno vypočítat rychlost a zrychlení požadované při pohybu. V tomto příkladu se z důvodu minimalizace času zvolil trojúhelníkový profil rychlosti. Rovnice pro výpočet parametrů jsou následující.

Rovnice:

$$x=\frac{1}{2}V_{pk}(t_a+t_d)$$

$$V_{pk}=\frac{2(x)}{(t_a+t_d)}$$

$$a=\frac{V_{pk}}{t_a}$$



Použitím čísel z tohoto příkladu v rovnicích trojúhelníkové rychlosti dostaneme následující:

Je dáno :

$$t_a = 0.01 \cdot \text{sec}$$

$$t_d = 0.01 \cdot \text{sec}$$

$$x = \frac{1}{60} \cdot \text{rev}$$

$$V_{pk} = \frac{2 \cdot \frac{1}{60} \cdot \text{rev}}{0.01 \cdot \text{sec} + 0.01 \cdot \text{sec}}$$

$$V_{pk} = 1.6667 \cdot \frac{rev}{\text{sec}}$$

$$V_{pk} = 1.6667 \cdot \frac{rev}{\text{sec}} \cdot 8192 \cdot \frac{cnts}{rev} \cdot \frac{600}{8192} \cdot \frac{uu}{cnts}$$

$$V_{pk} = 1000 \cdot \frac{uu}{\text{sec}}$$

$$a = \frac{V_{pk}}{t_a}$$

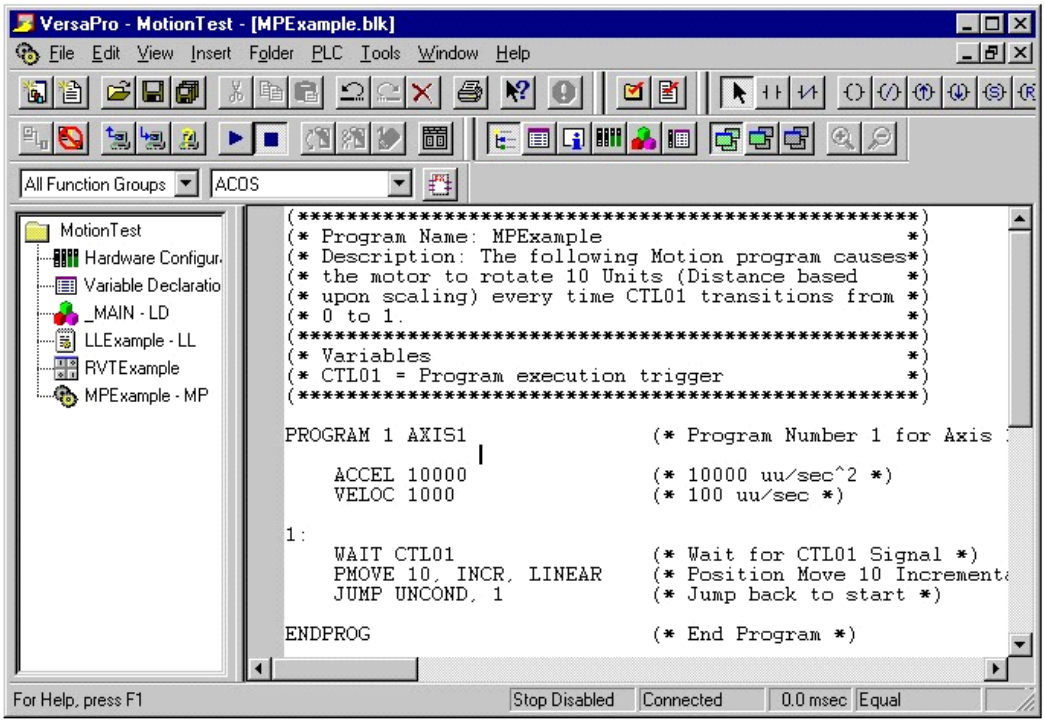
$$a = \frac{1000 \cdot \frac{uu}{\text{sec}}}{0.01 \cdot \text{sec}}$$

$$a = 10000 \cdot \frac{uu}{\text{sec}^2}$$

Nyní jste připraveni napsat pohybový program. Kód pro tento vzorový program bude vypadat následovně.

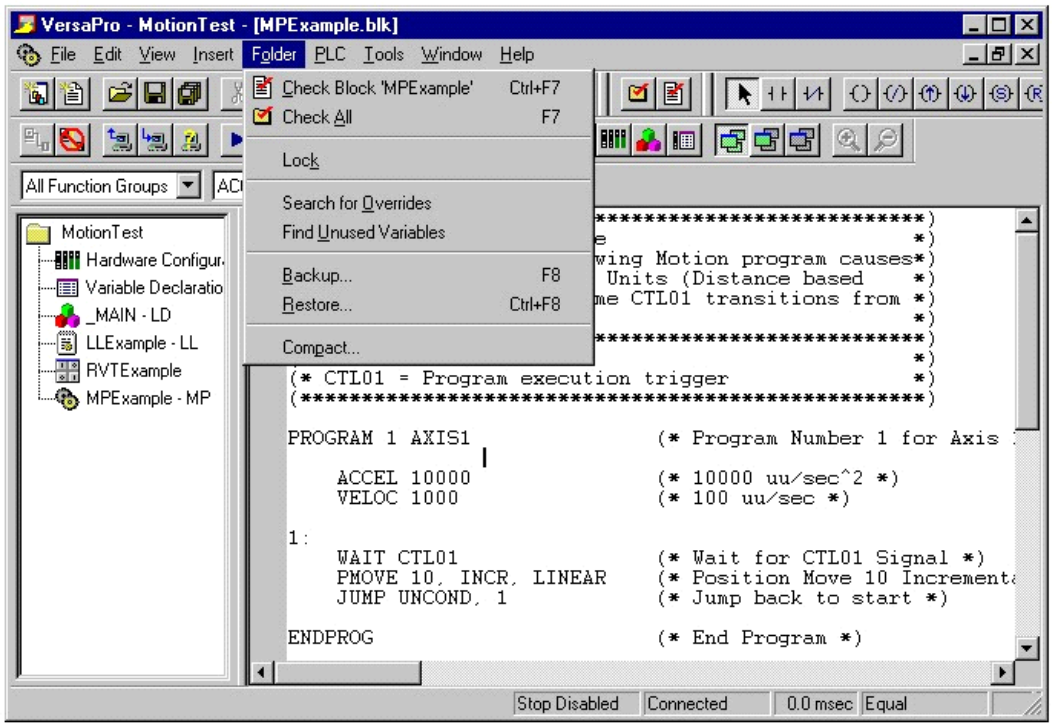
```
(*************************************************************)
(* Název programu: MPExample                                 *)
(* Popis: Následující Pohybový program způsobí,              *)
(* že motor se otočí o 10 jednotek (vzdálenost podle         *)
(* měřítka ) při každém přechodu CTL01 z 0 do 1.             *)
(*************************************************************)
(* Proměnné                                                  *)
(* CTL01 = Aktivace vykonání programu                        *)
(*************************************************************)

PROGRAM 1 AXIS1            (* Program číslo 1 pro osu 1 *)

    ACCEL 10000            (* 10000 uu/sec^2 *)
    VELOC 1000             (* 100 uu/sec *)

1:
    WAIT CTL01                 (* Čekání na signál CTL01 *)
    PMOVE 10, INCR, LINEAR     (* Posuv v poloze 10 inkrementů lineárně *)
    JUMP UNCOND, 1             (* Skok zpátky na start *)

ENDPROG                        (* Konec programu *)
```

Uživatel musí zapsat výše uvedený program do textového editoru. Jakmile toto bude hotové, editor bude vypadat asi jako na Obrázek 10-27. .



Obrázek 10-27. VersaPro editor pohybu MPExample

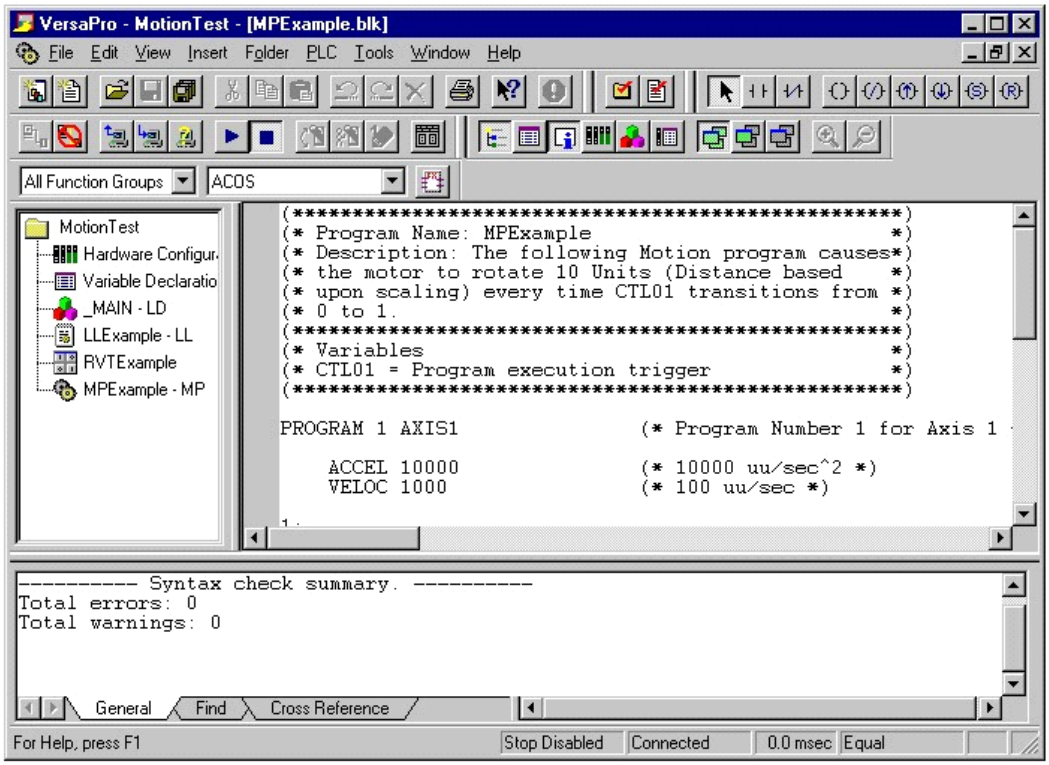
Nyní je nutno zkontrolovat správnou jazykovou syntaxi. Ověření jazykové syntaxe se provede zvolením hlavního menu Folder, pak podmenu Check Block 'MPExample'. Tím se spustí programy pro kontrolu syntaxe zadaného pohybového programu. Poznámka: Také je možno zvolit Check All a tím vybrat všechny bloky v adresáři. (Viz obrázek 10-27.)



Obrázek 10-28. Menu kontroly bloku MPExample

Pokud nebylo zobrazeno dříve, kontrola bloků vyvolá zobrazení informačního okna. Všimněte si, že informační okno je možno zapínat a vypínat stisknutím ikony  na panelu nástrojů.

Informační okno zobrazí výstup operace kontroly syntaxe. Pokud vzorový program byl zapsaný správně, měli byste dostat zprávu s nulovým počtem chyb a s žádnou výstrahou. Poznámka: V informačním okně je možno rolovat. Stisknutím rolovacího tlačítka po pravé straně okna si můžete prohlédnout informace, které se nezobrazí uvnitř viditelné plochy okna.



Obrázek 10-29. Úspěšná kontrola syntaxe MPExample

Pokud informační okno bude indikovat, že se vyskytly chyby syntaxe, informačním oknem můžete rolovat na řádek, který obsahuje chybové hlášení. Když informační okno je zvětšené, klikněte dvakrát na chybové hlášení. Tím okno editoru automaticky přejde na řádek v programu, který způsobil chybu.

Pokud právě budete editovat pohybový program a budete chtít najít přibližné číslo řádku, klikněte na vertikální rolovací lištu na pravé straně editoru pohybového programu. Zobrazí se číslo řádku na horním okraji okna. Pokud kurzor bude několik řádků pod hořejškem okna, buď rolujte oknem, až aktuální řádek bude u horního okraje okna, nebo odpočítejte počet řádků od shora a přičtěte je k aktuálnímu číslu řádku.

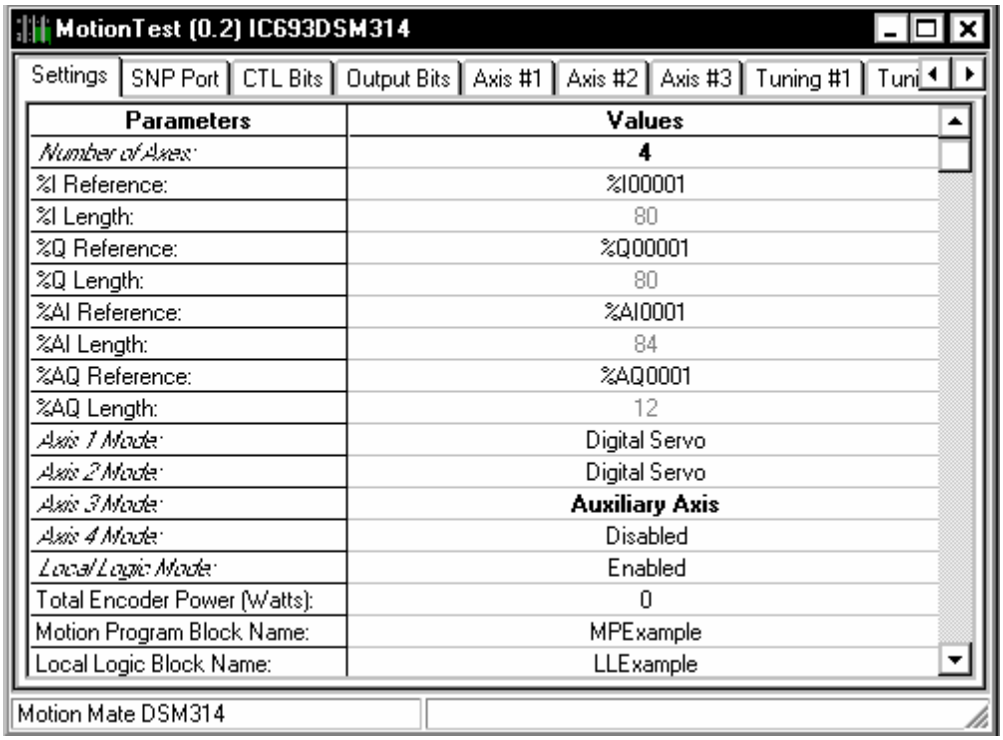
Kapitola 12 obsahuje další podrobnosti, které uvádějí neúspěšnou kontrolu syntaxe a nápravné kroky. Po úspěšné kontrole syntaxe potřebujete nastavit hardwarovou konfiguraci, která umožní načtení vzorového programu do správného modulu DSM314. Pořadí, ve kterém je příklad vytvořený, není pro většinu aplikací typické. Většina uživatelů nejdříve nastaví hardwarovou konfiguraci a pak vytvoří programové příkazy. Tento příklad však má za cíl znázornit pohybové programy a má obrácené pořadí pro lepší názornost vazby mezi hardwarovou konfigurací a názvem pohybového programu v hardwarové konfiguraci DSM314. Proto dalším krokem v tomto příkladu je vyvolat hardwarovou konfiguraci. Existuje několik způsobů, jak vyvolat obrazovky hardwarové konfigurace. (Další podrobnosti najdete v dokumentaci k VersaPro.) Předchozí odstavce této kapitoly uvádějí metodu popsanou krok za krokem, jak aktivovat hardwarovou konfiguraci. Proto tyto informace nebudou v této části opakované. Tato část se zaměřuje na další informace, které je nutno zadat do hardwarové konfigurace, aby pohybový program byl funkční. Další podrobnosti ohledně konfiguračních nastavení DSM4 najdete v kapitole 4. První pole, které je nutno nastavit, je "Motion Program Block Name" na záložce Settings. Toto pole definuje název pohybového

programu DSM314, který se má načíst do modulu. Do tohoto pole zapište název vzorového příkladu “MPExample”.

Poznámka: Tato metoda spojování DSM314 s pohybovým programem byla zvolena proto, aby bylo možno snadno zadat několik DSM314, které všechny používají stejný pohybový program. Tento příklad má pouze jedno DSM314. Pokud však budete mít několik DSM314, které mají mít spuštěný stejný pohybový program, v konfiguraci pro dané DSM314 zadejte, že má vykonat tento program. To umožní programovacímu zařízení, aby měl jeden zdrojový soubor pohybového programu pro několik DSM314. Všimněte si, že toto neznemožní, aby DSM314 vykonávalo různé programy.

Protože v tomto příkladu se používá Beta 0.5, nastavte režim pro osu 1 na digitální servo.

Výsledná obrazovka pro konfiguraci hardwaru bude vypadat jako na Obrázek 10-30. .



Obrázek 10-30. Záložka nastavení hardwarové konfigurace DSM314

Také můžete nakonfigurovat DSM s výše vypočítanými hodnotami pro poměr uživatelských jednotek na počet pulsů a špičkovou rychlost. Příklad také provádí konfiguraci směru osy a horní meze polohy. Ty jsou volitelné. Informace o těchto konfiguračních polích najdete v kapitole 4. Chcete-li přidat tyto hodnoty, na záložce Axis#1 do polí zapište následující.

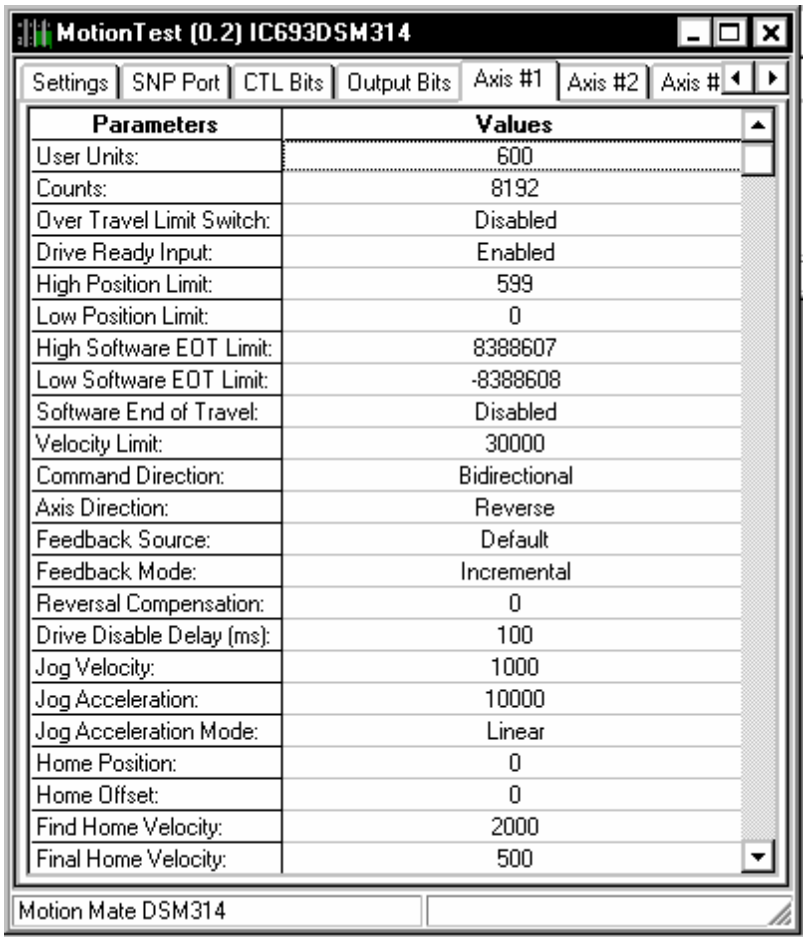
UserUnits: 600

Counts: 8192

High Position Limit: 599 (volitelné, způsobí, že se poloha bude přetáčet při každé otáčce)

Positive Velocity: 30000

Axis Direction: Reverse (volitelné, způsobí, že se osa bude otáčet ve směru hodinových ručiček)



Parameters	Values
User Units:	600
Counts:	8192
Over Travel Limit Switch:	Disabled
Drive Ready Input:	Enabled
High Position Limit:	599
Low Position Limit:	0
High Software EOT Limit:	8388607
Low Software EOT Limit:	-8388608
Software End of Travel:	Disabled
Velocity Limit:	30000
Command Direction:	Bidirectional
Axis Direction:	Reverse
Feedback Source:	Default
Feedback Mode:	Incremental
Reversal Compensation:	0
Drive Disable Delay (ms):	100
Jog Velocity:	1000
Jog Acceleration:	10000
Jog Acceleration Mode:	Linear
Home Position:	0
Home Offset:	0
Find Home Velocity:	2000
Final Home Velocity:	500

Obrázek 10-31. Záložka Axis#1 hardwarové konfigurace DSM314

K dokončení konfigurace je nutno přejít na záložku Tuning#1 a zapsat následující data.

Motor Type: 13

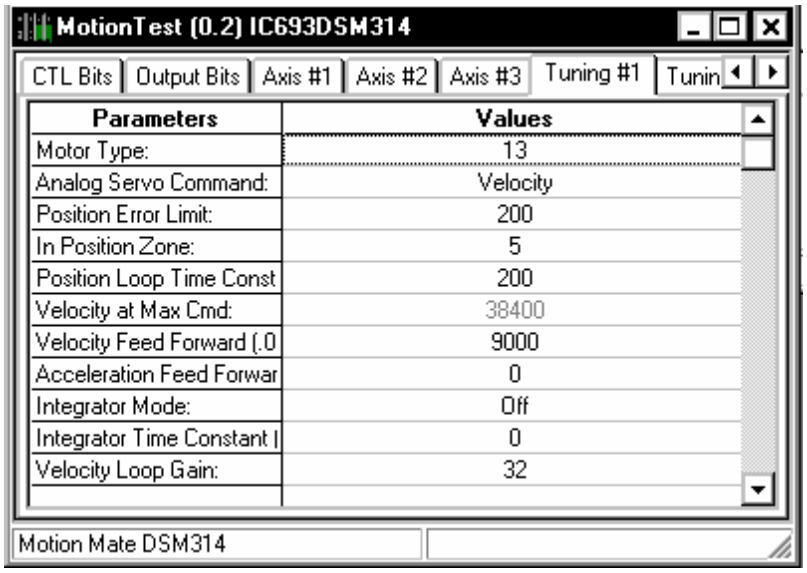
Position Error Limit: 200 (volitelné, další informace najdete v Konfiguračních informacích)

In Position Zone: 5 (volitelné, další informace najdete v Konfiguračních informacích)

Pos Loop Time Const: 200 (Poznámka: Vychází z aplikace/mechaniky, viz kapitola 4 a Dodatek D)

Velocity FeedForward: 9000 (Poznámka: Vychází z aplikace/mechaniky, viz kapitola 4 a Dodatek D)

Výsledné zobrazení vypadá přibližně jako na Obrázek 10-32. .



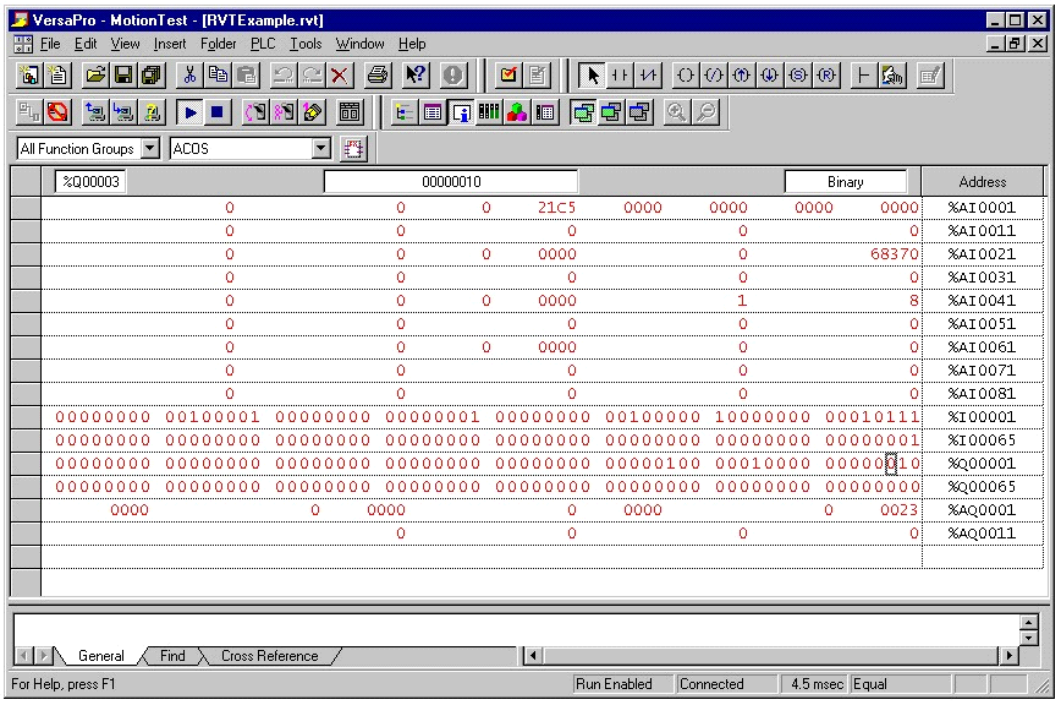
Obrázek 10-32. Záložka hardwarové konfigurace Tuning#1

Nyní je možno uzavřít dialogové okno konfigurace modulu. To je vhodný okamžik k tomu, abyste uložili svou práci. Chcete-li svou práci uložit, z hlavního menu zvolte File a pak z menu souboru zvolte Save All.

Spojení mezi příkladem pohybového programu a modulem DSM314 je nyní kompletní. Vytvořte libovolnou žebříkovou logiku PLC a pak u programů proveďte Zkontrolovat všechno a načtěte je do PLC. Další informace týkající se operace načtení jsou uvedené v dokumentaci k VersaPro GFK-1670 nebo v on-line nápovědě VersaPro 1.1.

Vykonání prvního pohybového programu

Jakmile se dokončí operace načítání, modul je připravený k vykonání pohybového programu a programu lokální logiky. Má-li DSM vykonat program lokální logiky, z PLC je nutno offset bitu Q nastavit na 1, když se PLC nachází v režimu RUN. Tím se aktivuje program lokální logiky v DSM. Další věc, kterou je nutno udělat, je vykonat povel pro nastavení polohy. Tím se nastaví reference modulu a umožní se vykonávat požadovaný pohybový program. Chcete-li vykonat tuto funkci, otevřete RVT (RVTExample) vytvořenou v části Lokální logika a v AQ offsetu 1 zapište 0023 hex. Tím se zapíše povel pro nastavení polohy. Pak do AQ offset 2 zapište 0. Další informace týkající se konfigurace povelů AQ najdete v kapitole 5. Výsledné zobrazení vypadá přibližně jako na obrázku 10-33. .



Obrázek 10-33. Obrazovka RVTExample

Pokud se dosud neobjevily žádné chyby, můžete vykonat pohybový program. Do bitu Q offset 2 (%Q00003) zapište 1 (nebo ho překlopte). Motor by nyní měl začít vykonávat pohybový program a každou sekundu postoupit o 1/60 otáčky.

Další podrobnosti týkající se rozhraní mezi DSM a PLC jsou uvedené v kapitole 5.

Kapitola 11

Výuka lokální logiky

Úvod

Programovací jazyk lokální logiky podporuje přiřazování, podmíněné příkazy, aritmetické, logické a relační operace. Program lokální logiky běží synchronně s polohovou smyčkou pohybového modulu a jako takový je deterministický. Jazyk zahrnuje konstrukce, které umožňují programu lokální logiky předávat informace mezi programem logiky, pohybovým programem a nadřízeným PLC. Výuka se zaměřuje na jazyk lokální logiky a jeho komunikaci s pohybovými programy. Další informace týkající se jazyka zařízení pro programování pohybu najdete v kapitole 7.

Příkazy

Programovací jazyk lokální logiky podporuje příkazy přiřazování a podmíněné příkazy. Příkazy přiřazování povolují, aby aritmetické výsledky a bitové logické operace byly přiřazované proměnným. Podmíněné příkazy povolují vykonávání podmíněného logického kódu. Podmíněné vykonávání vychází z hodnoty konstanty nebo proměnné nebo z výsledku relační nebo bitové logické operace.

Příkazy přiřazování používají operátor ":=". Následující příklad násobí dva registry parametrů a výsledek přiřadí jinému parametru registrů.

```
P001 := P210 * P107;
```

Poznámka: Příkazy přiřazování vyžadují středník jako koncový znak, jak je znázorněno výše.

Podmíněné příkazy používají kombinaci klíčových slov IF-THEN-END_IF. Klíčové slovo END_IF uzavírá podmíněný příkaz. Následující příklad kontroluje hodnotu proměnné Block_1 a podmíněně nastaví hodnotu v registru parametrů. Konkrétně pokud se hodnota proměnné Block_1 bude rovnat 5, pak se hodnota parametru P010 nastaví na 100.

```
IF Block_1 = 5 THEN
        P010 := 100;
END_IF;
```

Klíčová slova IF, THEN a END_IF rozeznávají velká a malá písmena a příkaz END_IF je ukončený středníkem. Příkazy IF je možno vnořovat až do osmi úrovní a tělo příkazu IF může obsahovat jeden nebo více příkazů. Detailní popis k těmto příkazům najdete v kapitole 12.

Komentáře

Komentáře umožňují programátorovi popsat činnost programu nebo do programu vložit jiné informace, které bude chtít. Komentář začíná dvojicí znaků **(*** a končí dvojicí znaků ***)** a může se objevit kdekoliv uvnitř programu. Například:

(* Platná struktura komentáře *)

DSM během vykonávání programu komentáře ignoruje. Při určování délky programu lokální logiky se proto řádky komentáře nepočítají.

Proměnné

Lokální logika dává uživateli pomocí pevné sady proměnných přístup k datům pohybového kontroléru, řídícím a stavovým bitům a parametrům. Jazyk také podporuje dekadické, hexadecimální a binární konstanty. Hexadecimální a dekadické hodnoty mohou být konstanty bez znaménka v programu, ale v matematických výrazech jsou VŽDY interpretované jako dvojkový doplněk. Má-li se proměnné přiřadit znaménko, je nutno zapsat následující

Torque_Limit_1 :=5000; (* Nastavení meze kroutícího momentu osy 1 na 50% *)

nebo v hexadecimálním tvaru

Torque_Limit_1:=16#1388; (* Nastavení meze kroutícího momentu osy 1 na 50% *)

Když proměnné mají přiřazené číselnou hodnotu, automaticky se u nich zkontrolují meze 32-bitové hodnoty se znaménkem. Například následující hodnoty představují největší kladnou a zápornou hodnotu, které jsou přípustné.

P001:=16#7FFFFFFF; (* P001=2147483647 *)

nebo v dekadickém tvaru

P001:=2147483647; (* P001=16#7FFFFFFF *)

Má-li se přiřadit maximální záporná hodnota, je nutno zapsat

P002:=16#80000000; (* P002=-2147483648 *)

nebo v dekadickém tvaru

P002:=-2147483648; (* P002=16#80000000 *)

Pokud byste zapsali číslo, které přesahuje výše uvedené číselné meze, bude se generovat chyba udávající, že konstanta je mimo rozsah.

Proměnné lokální logiky mají "směrový" atribut čtení, zápis nebo čtení/zápis. (Další informace týkající se proměnných a jejich typů najdete v kapitole 13.) Například proměnná Kladný konec posuvu pro osu 1 (Positive_EOT_1) je proměnná pouze pro čtení. Proto následující konstrukce je platná:

P001:=Positive_EOT_1; (* P001 = Kladný konec posuvu osy 1 *)

Avšak následující konstrukce je neplatná:

Positive_EOT_1:=1;

Pokud se program bude snažit zapsat do proměnné pouze pro četní nebo se bude snažit číst z proměnné určené pouze pro zápis, kompilační analyzátor lokální logiky vygeneruje chybu.

Kromě toho proměnné lokální logiky mají atribut velikosti v rozsahu od Booleovské proměnné (1 bit) až po celé číslo s dvojnásobnou délkou (64 bitů). Když Booleovské proměnné bude přiřazena nebooleovská hodnota, kompilační analyzátor logické proměnné bude generovat chybu. Když se objeví toto přiřazení, výstraha upozorní, že v důsledku zaokrouhlení může dojít ke ztrátě dat. Všimněte si, že celočíselné proměnné s dvojnásobnou délkou (64 bitů) je možno použít pouze jako výsledek operace násobení nebo čitatel operace dělení nebo modulo.

Další informace týkající se proměnných lokální logiky najdete v kapitole 13. Kromě toho tabulka proměnných lokální logiky (LLVT) v programovacím softwaru obsahuje informace o velikosti, typu a vlastnosti čtení/zápis proměnných.

Operátory

Lokální logika používá tři třídy operátorů. Jsou to aritmetické, relační a bitové logické operátory. Úvod k jednotlivým operátorům je uvedený níže. Více podrobností o operátorech najdete v kapitole 12.

Aritmetické operátory

Lokální logika dává možnost provádět základní aritmetické operace. Jazyk podporuje 32-bitové celočíselné operace a omezené použití 64/32-bitových operací, kde je nutno dodržovat přesnost. Všechny aritmetické funkce kromě funkce ABS vyžadují dva operandy.

Lokální logika podporuje operace sčítání, odčítání, násobení, dělení, absolutní hodnotu a modulo.

Příklad konstrukce:

```
P010 := Commanded_Velocity_1 - P009; (* P010 = Zadaná rychlost osy 1 – P009*)
```

Všimněte si, že následující zápis je neplatná matematická konstrukce:

```
Commanded_Velocity_1 := P010 - P009; (* Commanded_Velocity_1=P010-P009*)
```

Důvodem, proč je zápis neplatný, je že matematický výraz se snaží přiřadit výsledek (P010-P009) hodnotě Commanded_Velocity_1, což je proměnná pouze pro čtení.

Ukládání mezivýsledků do registru parametrů dává flexibilitu potřebnou k řešení složitých matematických výrazů.

Například následující konstrukce je platná, protože obsahuje více než jednu operaci (násobení a odečítání):

```
P005:= Torque_Limit_1 *( P001 – P010);
```

Aby se dosáhlo stejných výsledků, je nutno zapsat následující:

```
P004:= P001 – P010;
P005:= Torque_Limit_1 * P004;
```

Relační operátory

Relační operátory porovnávají dva operandy v podmíněném výrazu. Platné relační operátory jsou < (menší než), > (větší než), <= (menší nebo rovno), >= (větší nebo rovno), = (rovná se), a <> (nerovná se). Vykonání těla příkazu IF se provede, když bude splněný podmíněný výraz. V příkladu se proměnná Torque_Limit_1 nastaví na 10000, pokud se proměnná Block_1 bude rovnat 3. Pokud se hodnota proměnné Block_1 nebude rovnat 3, pak se výraz vyhodnotí jako nepravdivý a vykonávání programu bude pokračovat za programovým příkazem END_IF.

Příklad:

```
IF Block_1 = 3 THEN
    Torque_Limit_1 := 10000;       (* Nastavení meze kroutícího momentu = 100% @ Blok 3 *)
END_IF;
```

Složité vztahy je možno vyřešit vnořováním příkazů IF. Má-li se například nastavit osa 1 (Torque_Limit_1) na 10000=100% (tj. stejné měřítko jako při zpracování povelu AQ), když je aktivní blok 3 pohybového programu a zadaná rychlost pro osu 1 (Commanded_Velocity_1) je menší než 1000, bude platná následující konstrukce:

```
IF Block_1 = 3 THEN
    IF Commanded_Velocity_1 < 1000 THEN
      Torque_Limit_1 := 10000;     (* Nastavení meze kroutícího momentu = 100% @ Blok 3 *)
    END_IF;
END_IF;
```

Bitové logické operátory

Bitové logické operátory maskují nebo invertují jednotlivé bity nebo skupinu bitů. Operátory BWAND (a), BWOR (nebo), BWXOR (exclusive or) a BWNOT (jedničkový komplement) jsou platné konstrukce. BWAND, BWOR a BWXOR vyžadují dva operandy. Operátor BWNOT vyžaduje jeden operand.

Následující kódový segment vyjme kopii několika bitů ze slova CTL_1_to_32 a přiřadí je registru parametrů.

Pak se testují čtyři nejméně významné bity této hodnoty a pokud některý z nich bude nastavený, do P002 se zapíše hodnota 4985.

```
P001 := CTL_1_to_32 BWAND 16#0000A005;
IF P001 BWAND 16#F THEN
    P002 := 4985;
END_IF;
```

Příkazy konkrétně vykonávají následující operace. První příkaz používá 16#0000A005 jako masku. Maska odpovídá binární hodnotě následujícím způsobem:

16# 0000A005 = 2#0000 0000 0000 0000 1010 0000 0000 0101

Proto příkaz

P001:=CTL_1_to_32 BWAND 16#0000A005

oddělí bity 1,3,14 a 16 z CTL_1_to_32 a výsledek uloží do P001.

Další příkaz provádí testování po jednotlivých bitech, jestli bity v nejméně významném bajtu jsou v jedničce. Testovaná hodnota odpovídá binární hodnotě následujícím způsobem:

16#F = 2#1111

Proto příkaz

IF P001 BWAND 16#F THEN

provede testování jednotlivých bitů nejméně významného bajtu P001 a pokud některý bit z nejméně významného bajtu bude nastavený do logické jedničky (hodnota = 1), pak se vyhodnotí příkaz v bloku IF.

Protože v tomto příkladu CTL_1_to_32 je maskováno v předchozím příkazu, podmínka IF pouze testuje bit 1 a bit 3 z CTL_1_to_32.

Komunikace mezi lokální logikou / PLC / pohybovým programem

Program lokální logiky nebo nadřízené PLC komunikují s pohybovým programem pomocí parametrů, CTL bitů a čísla bloku pohybového programu. Tyto metody se používají pro:

- **Data parametrů** – Data parametrů (P000-P255) jsou přístupná z lokální logiky, nadřízeného PLC a pohybových programů. Data parametrů jsou podobná proměnným v programu. Například pohybový program může vykonat PRODLEVU určitou dobu, která je určena parametrem. Program lokální logiky nebo nadřízené PLC mohou zapsat parametr, který určuje dobu PRODLEVY v pohybovém programu.
- **Bity CTL** – Bity CTL umožňují programu lokální logiky nebo nadřízenému PLC signalizovat pohybovému programu, že má začít děj. Bity CTL se například používají k řízení toku pohybového programu s povelem JUMP.
- **Čísla bloků pohybových programů** – Pohybový program (kde se v pohybových programech používají čísla bloků) dává k dispozici číslo aktuálního bloku pro program lokální logiky nebo nadřízené PLC. Aktuální číslo bloku je možno použít v programu lokální logiky nebo nadřízeného PLC k tomu, aby se činnost provedla pouze během konkrétní části pohybového programu.

Signálové konstrukce mezi programy (PLC, pohyb a lokální logiky) umožňují, aby mezi sebou působily a vykonaly se operace mezi programy. Tyto signální konstrukce jsou důležité pro následující příklady programování. Další informace o komunikace PLC s pohybovým programem a spolupráci programu najdete v kapitole 5 a 7.

Příklady programování lokální logiky

Předchozí části představovaly úvod do základních konstrukcí jazyka lokální logiky. Následující odstavce obsahují příklady programů, které tyto koncepce objasňují. Tyto programy jsou určené pouze pro znázornění a nemusí nezbytně představovat funkční aplikace. Další podrobnosti týkající se použitelných příkazů lokální logiky, proměnných a konstrukcí najdete v kapitolách 12 a 13.

Příklad programu s omezením kroutícího momentu

Následující příklad znázorňuje metodu použití lokální logiky ve shodě s pohybovým programem k omezení kroutícího momentu na základě čísla bloku v pohybovém programu. V příkladu servoosa 1 má matku na hřídeli se závitem. Na začátku osa vykoná pohyb kousek zpět, aby se zlepšil záběr matky a závitu hřídele. Pohyb má nastavenou mez kroutícího momentu na maximální hodnotu. Pak se matka kroutí až do utažení kroutícím momentem omezeným na 30% maximální hodnoty. Během této operace se obvykle nedosáhne cílového bodu zadaného povelem a osa se zastaví, když tření bude větší než mez kroutícího momentu. Pak aby se uvolnilo pnutí v mechanice, kroutící moment se nastaví na 0 a po 0,1 sekundy signál "operace šroubu hotová" přejde do jedničky. Když PLC nastaví signál "kleština matky uvolněná" do jedničky, osa se bude pohybovat do výchozí polohy s plným kroutícím momentem.

Programu lokální logiky s omezením kroutícího momentu.

```
(*PŘIŘAZENÍ ŘÍDICÍCH BITŮ*)

(*CTL01 - start činnosti šroubu – signál z PLC*)
(*CTL02 - smyčka pohybového programu 1*)
(*CTL03 - kleština matky uvolněná – signál z PLC*)
(*CTL04 - operace šroubu hotová – signál do PLC*)

(* PŘIŘAZENÍ DAT PARAMETRŮ*)

(*P001 - zrychlení reverzního pohybu*)
(*P002 - rychlost reverzního pohybu*)
(*P003 - vzdálenost reverzního pohybu*)
(*P004 - zrychlení kroutícího pohyb matky*)
(*P005 - rychlost kroutícího pohybu matky*)
(*P006 - vzdálenost kroutícího pohybu matky*)
(*P007 - čítač používaný pro časovač uvolnění tahu*)

(*KÓD LOKÁLNÍ LOGIKY*)

IF Program_Active_1 = 1 THEN (*pohybový program pro osu 1 aktivní*)
   IF Block_1 = 10 THEN       (*pohybový program čeká na povel start*)
      P007:=0;                (*reset čítače*)
      CTL04:=0;               (*operace šroubu hotová na nulu*)
      Torque_Limit_1:= 10000; (*nastavení meze kroutícího momentu na
                               maximální hodnotu 100%*)
   END_IF;
   IF Block_1=30 THEN         (*hlavní kroutící pohyb matky*)
      Torque_Limit_1:= 3000;  (*nastavení meze krouticího momentu na 30%*)
   END_IF;
   IF Block_1=40 THEN         (*čekání na signál uvolnění *)
      Torque_Limit_1:= 0;     (*nastavení meze krouticího momentu =0*)
      P007:=P007 + 1;         (*aktualizace čítače časovače*)
      IF P007=50 THEN          (*uplynulo 100 ms*)
         CTL04:=1;            (*operace šroubu hotová do jedničky*)
      END_IF;
   END_IF;
   IF Block_1=50 THEN
      Torque_Limit_1:= 10000; (*nastavení meze kroutícího momentu na
                               maximální hodnotu 100%*)
   END_IF;
END_IF;
```

Pohybový program s omezením kroutícího pohybu

```
(* Pohybový program s omezením kroutícího pohybu *)


Program 1 AXIS1

10: WAIT   CTL01              (* Čekání na povel start *)
    ACCEL  P001               (* Nastavení zrychlení pohybu *)
    VELOC  P002               (* Nastavení rychlosti pohybu *)
20: PMOVE  P003,INCR,LINEAR   (* Zpětný pohyb *)
    ACCEL  P004               (* Nastavení zrychlení pohybu *)
    VELOC  P005               (* Nastavení rychlosti pohybu *)
30: CMOVE  P006,INCR,LINEAR   (* Hlavní kroutící pohyb matky *)
40: WAIT  CTL03               (* Čekání na signál uvolnění kleštiny *)
50: PMOVE 0,ABS,LINEAR        (* Pohyb do výchozí polohy *)
60: JUMP  CTL02,10            (* Skok na začátek smyčky, pokud program
                              běží ve smyčce *)

EndProg
```

Příklad programu plánovače zisku

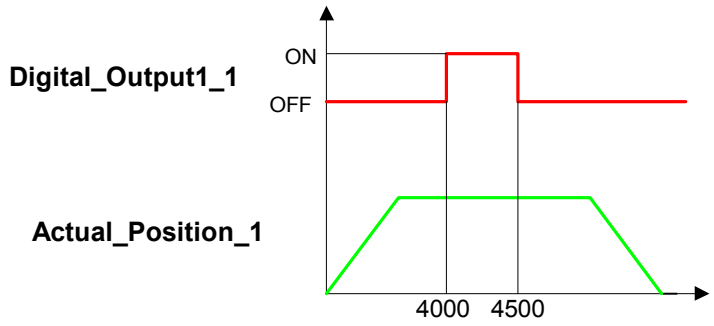
Následující příklad znázorňuje metodu použití lokální logiky k realizaci jednoduchého algoritmu plánování zisku. Při každém použití algoritmu, který dynamicky mění charakteristiky řízení, je nutno dát pozor. V mnoha situacích dynamické měnění řídicích charakteristik může způsobit, že řízený proces bude nestabilní. Všimněte si, že v následujícím programu se ve stejném cyklu může několikrát zapisovat do řídicí proměnné *Veocity_Loop_Gain*. Avšak aktivní hodnota bude konečná hodnota zapsaná v daném cyklu, protože proměnné se aktualizují při skončení vykonávání lokální logiky. Podrobný popis řídicích proměnných a výstupů lokální logiky najdete v kapitolách 12 a 13.

Program lokální logiky plánovače zisku

```
(* Tento příklad ukazuje způsob použití základního algoritmu plánování *)
(* zisku. Základní zisk rychlostní smyčky je nastavený na 12.          *)
(* Avšak zisk rychlostní smyčky se upravuje podle polohy a zadané      *)
(* rychlosti.                                                          *)

Velocity_Loop_Gain_1 := 12;       (*  Základní zisk rychlostní smyčky *)
IF Actual_Position_1  > 500 THEN  (* Pokud okamžitá poloha osy 1 > 500 *)
   IF Commanded_Velocity_1 < 10   (* Pokud zadaná rychlost osy 1 < 10 *)
      Velocity_Loop_Gain_1 := 16; (* Zisk rychlostní smyčky osy 1 = 16 *)
   END_IF;
END_IF;

IF Actual_Position_1  > 1000 THEN    (* Pokud okamžitá poloha osy 1 > 1000 *)
   IF Commanded_Velocity_1 < 10   (* Pokud zadaná rychlost osy 1 < 10 *)
      Velocity_Loop_Gain_1 := 20; (* Zisk rychlostní smyčky osy 1 = 20 *)
   END_IF;
END_IF;

IF Actual_Position_1  > 1500 THEN    (* Pokud okamžitá poloha osy 1 > 1500 *)
   IF Commanded_Velocity_1 < 10 THEN (* Pokud zadaná rychlost osy 1 < 10 *)
      Velocity_Loop_Gain_1 := 24;    (* Zisk rychlostní smyčky osy 1 = 24 *)
   END_IF;
END_IF;
```

Příklad programu programovatelného koncového spínače

Následující příklad znázorňuje metodu použití lokální logiky k provedení funkce programovatelného koncového spínače. Tento konkrétní programovatelný koncový spínač zapne/vypne výstup podle okamžité polohy motoru a bloku v pohybovém programu.



Obrázek 11-1. Příklad programovatelného koncového spínače

Program lokální logiky programovatelného koncového spínače

```
(* Tento příklad ukazuje způsob, jak přimět Digitální výstup # 1 osy 1,*)
(* aby se sepnul, když okamžitá poloha osy 1 bude mezi 4000 a 4500,    *)
(* ale pouze když program bude na bloku #4. To ukazuje funkčnost,      *)
(* která se často realizuje pomocí vysokorychlostního čítače           *)
(* (HSC) a povelů PRESET. Tento příklad dosahuje rozlišení 2 ms.       *)

Digital_Output1_1 := 0;  (* Digitální výstup osy 1 je v nule *)
IF Block_1 = 4 THEN      (* Pokud aktuální číslo bloku pro osu 1 se rovná
                            4 *)
   IF Actual_Position_1 > 4000 THEN     (* Pokud okamžitá poloha osy 1 >
                                           4000 *)
      IF Actual_Position_1 > 4500 THEN  (* Pokud okamžitá poloha osy 1 >
                                           4500 *)
         Digital_Output1_1 := 1;  (* Digitální výstup osy 1 je v jedničce*)
      END_IF;
   END_IF;
END_IF;
```

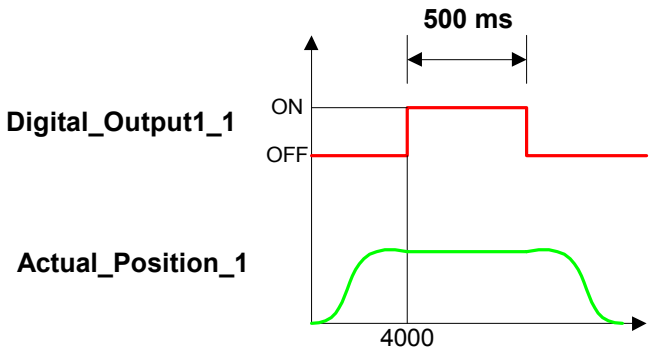
Segment pohybového programu odpovídající výše uvedenému programu lokální logiky je uvedený níže.

Příklad segmentu pohybového program programovatelného koncového spínače

```
Program 1 Axis1              (* Číslo programu 1 pro osu 1 *)
3: PMOVE 0,ABS,LINEAR        (* Pohyb do výchozí polohy *)
4: PMOVE 8000,ABS,LINEAR     (* Pohyb do polohy 8000 *)
5: PMOVE -8000,ABS,LINEAR    (* Pohyb do polohy -8000 *)
EndProg
```

Příklad programu aktivace výstupu na základě polohy

Následující příklad ukazuje způsob použití lokální logiky k aktivaci časovaného výstupu na základě okamžité polohy motoru. Všimněte si, že implementace časovače využívá čítač uvnitř programu. Čítač čítá, kolikrát se program vykoná od posledního resetu čítače. Protože programy lokální logiky se vykonají při každé periodě vzorkování polohové smyčky, doba periody čítače vychází z této periody. Tento příklad používá digitální serva, která mají periodu vzorkování polohové smyčky 2 ms. Proto čítač bude čítat v inkrementech 2 ms. Popis k ostatním konfiguracím periody vzorkování polohové smyčky najdete v kapitole 1. Kromě toho lokální logika umožňuje, aby program zapisoval proměnnou několikrát během programu. Poslední stav, ve kterém se proměnná nachází při dokončení programu, se zapíše na výstup (viz kapitola 12, část Výstupy/povely lokální logiky). To je důležité v následujícím programu. Druhý blok IF-THEN-END_IF nastaví digitální výstup pro osu 1 (Digital_Output1_1) do jedničky, když okamžitá poloha osy 1 (Actual_Position_1) bude větší než 4000, bez ohledu na okamžitou hodnotu časovače (P008). Pokud však časovač překročí 500, poslední blok IF-THEN-END_IF v programu zkontroluje okamžitou hodnotu čítače (P008) a nastaví digitální výstup 1 osy 1 (Digitial_Output_1) na nulu. Aplikace je ukázána na obrázku 11-2.



Obrázek 11-2. Příklad výstupu časovače na základě polohy

Program lokální logiky pro výstup časovače na základě polohy

```
(* Tento příklad ukazuje způsob spouštění časovaného výstupu na    *)
(* základě okamžité polohy osy 1                                    *)
(* Když Actual_Position_1 přesáhne 4000, Digital_Output1_1 se       *)
(* sepne na 500 ms                                                  *)

IF Block_1 = 4 THEN  (* Blok 4 v pohybovém programu resetuje časovač *)
   P008 := 0;        (* Resetování časovače (P008) *)
   Digital_Output1_1 := 0;  (* Přesvědčte se, že výstup je v nule před
                               pohybem *)
   P009 := 0;        (* P009 sleduje stav digitálního výstupu *)
END_IF;

IF Block_1=5 THEN
   IF Actual_Position_1 > 4000 THEN  (*Actual_Position_1 přesahuje 4000 *)
      Digital_Output1_1 := 1;        (* nastavení Digital_Output1_1 do
                                        jedničky *)
      P009 := 1;
   END_IF;
END_IF;

IF P009 = 1 THEN        (* Vždy když Digital_Output1_1 je v jedničce, *)
   P008 := P008 + 2;    (* inkrementuje se časovač o 2 ms *)
   IF P008 > 500 THEN   (* Když časovač překročí 500 ms *)
      Digital_Output1_1 := 0; (* Digital_Output1_1 přejde do nuly *)
      P009 := 0;          (* Nezapomeňte, že pro digitální výstup se
                          vyskytne pouze poslední zápis v cyklu lokální
                          logiky*)
   END_IF;
END_IF;
```

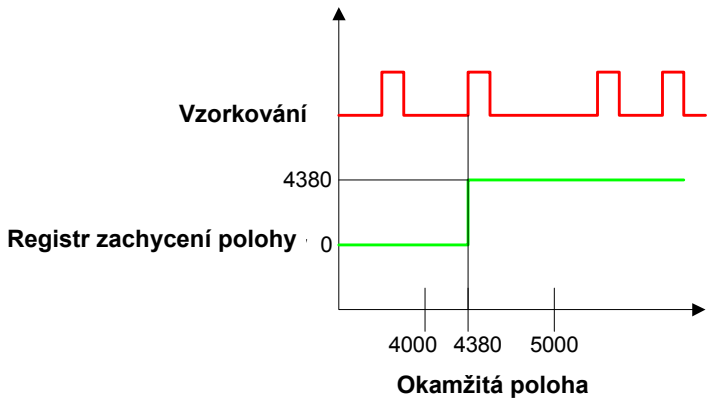
Segment pohybového programu odpovídající výše uvedenému programu lokální logiky je uvedený níže.

Příklad segmentu pohybového programu výstupu časovače na základě polohy

```
Program 1 Axis1           (* Číslo programu 1 pro osu 1 *)
4: PMOVE 0, ABS,LINEAR    (* Pohyb do výchozí polohy *)
5: PMOVE 12000,ABS,S-CURVE  (* Pohyb do polohy 12000 *)
EndProg
```

Příklad programu časových výřezů pomocí vzorkování

Následující příklad znázorňuje použití lokální logiky k provedení funkce časových výřezů (oken) pomocí vzorkování. Příklad bude ignorovat vzorkovací povely, dokud aktuální poloha motoru uvnitř časového výřezu nebude (Okamžitá poloha > 4000 ale menší než 5000). Pokud poloha motoru bude uvnitř výše zmíněného časového výřezu, první výskyt vzorkovacího pulsu způsobí, že se zachytí okamžitá poloha motoru do vzorkovacího registru. Aplikace je ukázána na Obrázek 11-3.



Obrázek 11-3. Příklad časových výřezů pomocí vzorkování

Program lokální logiky časových výřezů pomocí vzorkování

```
(* Časové výřezy pomocí vzorkování - příklad lokální logiky    *)
(* P009 je vzorkovaná poloha načtená z Strobe1_Position_1      *)
(* CTL13 se používá jako příznak výskytu vzorkovacího pulsu - musí být
   nakonfigurovaný pro řízení lokální logiky v hardwarové konfiguraci
   VersaPro                                                    *)

IF Block_1 = 6 THEN
   P009 := 0;    (* Inicializace P009 na 0 *)
   CTL13 := 0;   (* reset příznaku výskytu vzorkovacího pulsu *)
END_IF;
Reset_Strobe1_1 := 0;       (* Bit Reset Strobe nechejte implicitně na
                            nule *)
IF Strobe1_Flag_1 = 1 THEN  (* Pokud se vzorkovací puls vyskytne v rozsahu
                            "okna", *)
   IF Actual_Position_1 > 4000 THEN    (*nastaví se P009 = poloha
                                       vzorkování *)
      IF Actual_Position_1 < 5000 THEN
         P009 := Strobe1_Position_1;
         CTL13 := 1 ;       (* Nastavení příznaku výskytu vzorkovacího
                            pulsu na jedničku. *)
      END_IF;
   END_IF;
   Reset_Strobe1_1 := 1;    (* Pokud se vyskytne vzorkování, nastavit bit
                            Reset Strobe na jedničku. *)
END_IF;
```

Segment pohybového programu odpovídající výše uvedenému programu lokální logiky je uvedený níže.

Segment příkladu pohybového programu časových výřezů pomocí vzorkování

```
Program 1 Axis1             (* Číslo programu 1 pro osu 1 *)
   PMOVE 0,ABS,LINEAR       (* Pohyb do výchozí polohy *)
6: DWELL 10
7: PMOVE 10000,ABS,S-CURVE  (* Pohyb do polohy 10000 *)
EndProg
```

Kapitola 12

Syntaxe jazyka lokální logiky

Úvod

Tato kapitola popisuje syntaxi jazyka, pravidla a prvky jazyka pro programování lokální logiky. Jazyk používá neformátovaný text založený na konstrukcích odvozených z normy pro strukturovaný text IEC 1131. Následující odstavce popisují použitelné povely a jejich syntaxi.

Syntaktické prvky

Popis syntaxe jazyka lokální logiky je uvedený v následujících odstavcích. Syntaxi je snadné se naučit a poskytuje bohatý soubor funkcí, který uživateli umožňuje splnit programovací úlohu. Kapitola 11 obsahuje mnoho příkladů, které čtenáři dále pomůžou při pochopení syntaxe a jejího použití. Ti, kteří jazyk používají poprvé, mohou jako další pomůcku využít odstavec o programu "Vytvoření prvního programu lokální logiky" v kapitole 10 a vzorových příkladů v kapitole 11.

Číselné konstanty

Jazyk lokální logiky podporuje dekadické, hexadecimální a binární konstanty. DSM pracuje se všemi konstantami jako s 32-bitovými dvojkovými doplňky celočíselných hodnot se znaménkem. K zlepšení čitelnosti velkých čísel je možno mezi dvě číslice zařadit jeden znak podtržení (např. 16#7fff_ffff).

Dekadické konstanty musí být v rozsahu –2 147 483 648 až 2 147 483 647. Podporují se pouze celočíselné hodnoty a proto konstanty nemají dekadickou tečku. Proto jako u všech systémů na základě celých čísel jsou desetinné tečky obsažené a pokud je zapotřebí používat zlomky, programátor je musí sledovat.

Příklady:

523	Kladná dekadická konstanta
-1048	Záporná dekadická konstanta
1_745_245	Kladná dekadická konstanta s vloženými znaky podtržení

Hexadecimální konstanty (se základem 16) jsou označené předponou #16 a musí mít hodnotu, kterou je možno zapsat pomocí 32 bitů (8 hexadecimálních číslic). Hexadecimální konstanty nemohou mít předponu znaménka (+/-). Hexadecimální číslice A-F mohou být malá i velká písmena.

Příklady:
16#FFFF Hexadecimální konstanta
16#7fff_ffff Hexadecimální konstanta s vloženým znakem podtržení

Binární konstanty (se základem 2) jsou označené předponou #2 a musí mít hodnotu, kterou je možno zapsat pomocí 32 bitů (32 binárních číslic). Binární konstanty nemohou mít předponu znaménka (+/-).

Příklady:
2#1010 Binární konstanta
2#11111110_11101101_10111110_11101111 Binární konstanta s vloženými znaky podtržení

Program lokální logiky může mít maximálně 50 *jednoznačných* konstant, jejichž hodnota je větší než 2047 nebo menší než –2048. Pokud program lokální logiky bude deklarovat více než 50 jednoznačných konstant, proces sestavování bude hlásit chybu. Většina programů používá méně než 50 konstant, takže toto obecně nepředstavuje nějaké omezení.

Proměnné lokální logiky

Jazyk lokální logiky podporuje mnoho předdefinovaných proměnných, které umožňují přístup k I/O datům, CTL bitům a dalším stavovým a řídicím informacím DSM. Podrobný popis sady proměnných lokální logiky je uvedený v kapitole 13. Každá proměnná má dva atributy, velikost a směr. Hodnoty proměnných lokální logiky jsou v rozsahu 1 bitu (bitové operandy) až 64 bitů.

Všechny registry parametrů lokální logiky mají jeden z následujících typů.

- **Celočíselné proměnné s dvojnásobnou délkou** obsahují 32-bitové hodnoty se znaménkem (-2 147 483 648 až 2 147 483 647). Existuje 256 registrů parametrů (P000-P255).
- **Dlouhé celočíselné proměnné** obsahují 64-bitové hodnoty se znaménkem (+/-9.22 x 10^{18}). Dlouhé celočíselné proměnné jsou jedinečné v tom, že je možno je použít pouze pro výsledek násobení nebo jako čitatel v operacích dělení a dělení modulo. Existuje 8 registrů pro dlouhá celá čísla (D00-D07).

Všechny proměnné lokální logiky mají jeden z následujících směrových atributů.

- **Pouze pro čtení** se nesmí použít jako místo určení operace přiřazování.
- **Pouze pro čtení** se smí použít *pouze* jako místo určení příkazu přiřazení.
- **Čtení-zápis** se smí použít jako zdroj nebo jako místo určení.

V kapitole 13 najdete všechny atributy velikosti a směru proměnných lokální logiky.

Příkazy lokální logiky

Jazyk lokální logiky podporuje dva druhy příkazů: Přiřazovací a podmíněný. Program lokální logiky podporuje 150 příkazů. Pokud se překročí limit 150 řádků, kontrola bloku lokální logiky bude generovat chybové hlášení. Když program lokální logiky překročí 100 řádků, bude se generovat upozornění. Výstražné hlášení je možno vypnout instrukcí #pragma. Další podrobnosti viz odstavce #pragma. Příkazy programu se oddělují pomocí středníků.

Přiřazovací příkazy lokální logiky

Přiřazovací příkazy umožňují provádět jednoduché aritmetické a bitové operace a výsledek se přiřadí do proměnné. Přiřazovací příkaz má následující formát.

<místo určení> := <výraz>;

Operátor <**místo určení>** se může skládat z proměnné pro čtení-zápis nebo pouze pro zápis. **<Výraz>** může být jednoduchá konstanta nebo proměnná, matematická nebo bitová logická operace se dvěma operandy, funkce ABS nebo bitová operace NOT. Proměnné pouze pro zápis nemohou být **výrazem** pro operaci přiřazení.

Příklady:

P032 := Strobe1_Position_1 + 5000; ← *Tato konstrukce je v pořádku.*

P001 := ABS(Analog_Input1_1); ← *Tato konstrukce je v pořádku.*

Reset_Strobe1_1 := BWNOT Strobe1_Flag_1; ← *Tato konstrukce je v pořádku.*

P040 := 2#11111010_1011000; ← *Tato konstrukce je v pořádku.*

P011 := 3 * Strobe1_Position_1 + 20; ← *Tato konstrukce je NEPŘÍPUSTNÁ – příliš mnoho operací.*

Pokud se vyžadují složité operace, proveďte operaci pomocí série kroků, které používají registry parametrů k uložení mezivýsledků.

Příklady:

Má-li se nastavit Velocity_Loop_Gain_2 na hodnotu (1+75000/Actual_Velocity_2), programátor musí použít sérii příkazů podobných následujícímu...

```
P012 := 75000 / Actual_Velocity_2;
Velocity_Loop_Gain_2 := 1 + P012;
```

Pokud se použije booleovská proměnná jako výraz obsahující nebooleovskou proměnnou nebo konstantu, jejíž hodnota je nenulová nebo jednička, proces sestavování bude hlásit výstrahu. Výstraha se generuje proto, že DSM přiřazuje booleovské proměnné hodnotu nejméně významného bitu výrazu.

Podmíněné příkazy lokální logiky

Podmíněné příkazy umožňují vykonání podmíněného kódu na základě jednoduché relační nebo bitové logické operace. Podmíněný příkaz má následující formát.

IF <výraz> THEN

Příkazy lokální logiky

END_IF;

<Výraz> se může skládat z konstanty, proměnné, relační nebo bitové logické operace nebo bitového doplňku konstanty nebo proměnné. Proměnné pouze pro zápis nejsou ve výrazech přípustné. Pokud relační výraz bude pravdivý nebo pokud bitová operace, proměnná nebo konstanta budou mít nenulovou hodnotu, příkazy lokální logiky v těle příkazu IF se vykonají. V těle příkazu IF se může objevit libovolný počet programových příkazů (až do celkového limitu). Každý příkaz IF-THEN musí mít odpovídající END_IF.

Příklady:

IF P226 THEN ← *Tato konstrukce je v pořádku.*

IF CTL_1_to_32 BWAND 2#1010 THEN ← *Tato konstrukce je v pořádku.*

IF Strobe1_Level_1 = TRUE THEN ← *Tato konstrukce je v pořádku.*

IF BWNOT P100 THEN ← *Tato konstrukce je v pořádku.*

IF BWNOT P001 <> P002 THEN ← *Tato konstrukce je NEPŘÍPUSTNÁ – příliš mnoho operací.*

Příkazy IF je možno vnořovat až do hloubky 8 úrovní. Při počítání počtu programových příkazů se příkazy IF-THEN a END_IF počítají jako dva samostatné příkazy.

Tabulka 12-1. Platné operátory

Typ příkazu	Platné operátory	
Podmíněný	Relační	<, >, <=, >=, <>, =
	Bitové logické	BWAND, BWOR, BWXOR, BWNOT
Přiřazení	Aritmetické	+, -, /, *, MOD
	Bitové logické	BWAND, BWOR, BWXOR, BWNOT
	Funkce Abs	ABS ()

Prázdné místo

Mezery, konce řádků a tabulátory se pokládají za prázdné místo. Prázdné místo se ignoruje, kromě případu, kdy se používá k oddělení sousedních syntaktických prvků, a použití odsazení a prázdných řádků se může použít ke zlepšení čitelnosti programu.

Komentáře

Komentáře je možno použít k přidání informací do programu. Nástroj pro vykonání programu lokální logiky je ignoruje. Podporují se dva typy komentářů.

Znaková dvojice **(*** uvádí normální komentář, který je ukončený znakovou dvojicí ***)**. Tyto komentáře se mohou objevit kdekoliv, kde může být prázdné místo, například uvnitř nebo za příkazem programu lokální logiky, samostatně na řádku nebo mohou sahat přes několik řádků. Tyto komentáře se nevnořují.

Znaková dvojice // uvádí jednořádkový komentář. Veškerý text, který následuje za // až do konce řádku, nástroj pro vykonání lokální logiky bude ignorovat.

Poznámka: Je nutno dát pozor na to, že je možno zapsat program lokální logiky a nevhodný komentář mimo kód, který se má vykonat. Pak by běžně mohlo dojít k následujícímu:

```
(* Tento příklad ukazuje způsob, jak přimět Digitální výstup # 1 osy 1, aby      *)
(* se sepnul, když okamžitá poloha osy 1 bude mezi 4000 a 4500,                  *)
(* ale pouze když program bude na bloku #4. To ukazuje                           *)
(* funkčnost, která se často realizuje pomocí vysokorychlostního čítače          *)
(* (HSC) a povelů PRESET. Tento příklad dosahuje rozlišení 2 ms.    )
```

```
Digital_Output1_1 := 0;                    (* Digitální výstup osy 1 je v nule *)
IF Block_1 = 4 THEN                        (* Pokud aktuální číslo bloku pro osu 1 se rovná 1 *)
```

Ve výše uvedeném kódovém segmentu je struktura konce komentáře, řádek zobrazený pro názornost kurzívou/tučně, nesprávný, protože ve struktuře pro uzavření komentáře chybí hvězdička. Chyba bude mít za následek, že se následující řádek bude pokládat za komentář. Proto příkaz Digital_Output_1:=0 se bude pokládat za komentář a nevykoná se. Barevné rozlišování v rámci editoru lokální logiky může být velmi užitečné při hledání těchto typů problémů. Implicitní označování barvou vybarví komentáře jinou barvou než příkazy programu. Uživatel tím tak získá vizuální způsob, jak hledat tyto chyby. V kapitole 2 najdete informace o tom, jak změnit systém barevného označování v editoru.

Instrukce PRAGMA

Instrukce #pragma se používá k nakonfigurování kompilačního analyzátoru lokální logiky. Instrukce k činnosti analyzátoru NENÍ NUTNÁ. Pokud si však uživatel bude chtít výstražná hlášení vypnout, instrukce #pragma to umožňuje. Instrukce #pragma MUSÍ být první řádek v programu. Kromě toho se před instrukcí nesmí vyskytovat žádné prázdné místo.

Chcete-li vypnout VŠECHNY výstrahy lokální logiky, použijte následující povel:

```
#pragma errorsonly 1
   nebo
#pragma errorsonly ON
```

Chcete-li výstražná hlášení opět zapnout, buď instrukci smažte nebo jí změňte následovně:

```
#pragma errorsonly 0
   nebo
#pragma errorsonly OFF
```

Klíčová slova a operátory lokální logiky

Následující klíčová slova a operátory mají v programovacím jazyku lokální logiky zvláštní význam. Klíčová slova rozlišují malá a velká písmena a používejte v nich pouze velká písmena. Ty jsou podrobně popisované v následujících odstavcích.

Tabulka 12-2. Klíčová slova lokální logiky

ABS	TRUE	+	>
BWAND	FALSE	-	<
BWOR	IF	/	>=
BWXOR	THEN	*	<=
BWNOT	END_IF	16#	=
ON	MOD	2#	<>
OFF	;	:=	

Povolení a zákaz lokální logiky

Vykonání lokální logiky se povoluje pomocí PLC bitu Q. Pokud například DSM bude nakonfigurováno s počáteční adresou %Q0001, pak bit pro povolení lokální logiky bude %Q0002 (počáteční adresa + offset 1). Název programu lokální logiky musí být zadaný pomocí softwaru pro hardwarovou konfiguraci a pole pro povolení/zákaz lokální logiky musí být nastaveno na Povoleno. Podrobný popis konfigurace lokální logiky při hardwarové konfiguraci najdete v kapitole 10.

Lokální logika se vykoná, pouze pokud PLC bude v režimu RUN. Pokud PLC bude přepnuté do režimu STOP nebo pokud bit Q lokální logiky bude v nule, vykonání lokální logiky se zastaví a všechny digitální výstupy, řídicí bity (Jog, zastavení posuvu, resety vzorkování, povolení vlečené osy) a bity CTL, které řídí lokální logiky, budou zakázané.

Snaha vykonat lokální logiku v prvním cyklu PLC bude mít za následek hlášení chyby. Například přepnutí z režimu Stop do režimu Run, když bit povolení lokální logiky bude v jedničce, bude mít za následek chybu a program lokální logiky se nevykoná. Aby se program lokální logiky spustil, přepněte bit povolení Q.

Poznámka

Výkonný program lokální logiky nepoběží, pokud některá *uživatelská* funkce lokální logiky bude v hardwarové konfiguraci VersaPro povolená pomocí rozšířených parametrů. Uživatelskou funkci normálně není možno použít a GE Fanuc jí vyvinul pouze pro použití s konkrétní aplikací.

Výstupy/povely lokální logiky

S výstupy povelových bitů DSM (**Jog, Zastavení posuvu, Povolení vlečené osy** a **Resety vzorkování**) se mezi povelem PLC a povelem lokální logiky vykonává funkce **OR**. Proto je může řídit buď PLC nebo lokální logika, tj. výstup povelového bitu je aktivní, pokud ho buď PLC nebo lokální logika nastaví do jedničky.

Povely AQ se přijímají na principu posledního zápisu. Pokud například PLC (%AQ) i lokální logika vydají povel Poměr vlečené osy, bude aktivní poslední zapsaná hodnota.

Digitální výstupy na čelní desce (skutečné výstupy, které spíná DSM) je možno individuálně nakonfigurovat, aby je řídila buď lokální logika nebo PLC, ale ne oba současně. Podrobný popis konfigurace digitálních výstupů najdete v kapitole 14.

Digitální výstupy lokální logiky, okamžité povely a povelové bity se aktualizují na konci každého cyklu lokální logiky (seznam proměnných a proměnných digitálních výstupů najdete v kapitole 13). Proto pokud program lokální logiky zapíše do stejné programové proměnné nebo několikrát zapíše do proměnné digitálního výstupu během jednoho cyklu, platný povel bude naposledy zapsaná hodnota.

Například níže uvedený příklad kódu ukazuje proměnnou *Jog_Plus*, proměnnou *Strobe_Reset* a *Follower_Ratio*, do kterých se zapisuje několikrát během jednoho cyklu. Ve všech případech je aktivní ta hodnota, která byla zapsána nakonec.

Příklad:

```
Jog_Plus_1 := TRUE;   (* Nastavení Jog Plus pro osu 1 do jedničky *)

Strobe_Reset1_3 := 0;   (* Nastavení resetovacího bitu Strobe 1 pro osu 3 do nuly *)

(* Další příklady kódů *)

Follower_Ratio_A_1 := 10; (* Nastavení poměru vlečené osy A pro osu 1 na 10 *)

Jog_Plus_1 := FALSE;     (* Nastavení Jog Plus pro osu 1 do nuly *)

Strobe_Reset1_3 := 1;    (* Nastavení resetovacího bitu Strobe 1 pro osu 3 do
                            jedničky *)

Follower_Ratio_A_1 := 20;  (* Nastavení poměru vlečené osy A pro osu 1 na 20 *)
```

U každého z výše uvedených povelů výkonný program logiky postupuje na konci každého cyklu podle poslední zapsané hodnoty. Proto *Jog_Plus_1* se přepne do nuly, *Strobe_Reset1_3* se přepne do jedničky a *Follower_Ratio_A_1* se nastaví na 20.

Aritmetické operátory lokální logiky

Jazyk lokální logiky obsahuje známé konstrukce k vykonávání celočíselných aritmetických výpočtů se znaménkem. Jazyk podporuje 32-bitovou aritmetiku v programu lokální logiky a omezené použití 64/32-bitové aritmetiky. Všechny operace vyžadují dva operandy kromě funkce ABS, která uloží absolutní hodnotu proměnné nebo číselné konstanty.

Tabulka 12-3. Aritmetické operátory

Operátor	Význam
+	Sčítání
-	Odečítání
*	Násobení
/	Celočíselné dělení
MOD	Modulo
ABS	Absolutní hodnota

Aritmetické výrazy je možno použít pouze v přiřazovacích příkazech s jednou operací na příkaz.

Aritmetické operace nevyžadují funkci konverze typu dat, protože pohybový modul tuto operaci provádí automaticky.

Operátor +

Sečte zdroj1 a zdroj2 a výsledek uloží na místo určení.

Syntaxe

místo určení := zdroj1 + zdroj2;

Syntaxe operátoru + má tyto části:

Část	Popis
Místo určení	Jakákoliv proměnná lokální logiky pro zápis kromě registrů Dxx.
zdroj1	Jakákoliv proměnná/konstanta lokální logiky pro čtení kromě registrů Dxx.
zdroj2	Jakákoliv proměnná/konstanta lokální logiky pro čtení kromě registrů Dxx.

Přetečení – Nastaví se do jedničky, když výsledek sčítání bude větší než 2 147 483 647 nebo menší než –2 147 483 648. Module_Status_Code se nastaví na hodnotu 16#0095, což je pouze stavová chyba.

Operátor -

Odečte zdroj2 od zdroje1 a výsledek uloží na místo určení.

místo určení := zdroj1 - zdroj2;

Syntaxe operátoru - má tyto části:

Část	Popis
Místo určení	Jakákoliv proměnná lokální logiky pro zápis kromě registrů Dxx.
zdroj1	Jakákoliv proměnná/konstanta lokální logiky pro čtení kromě registrů Dxx.
zdroj2	Jakákoliv proměnná/konstanta lokální logiky pro čtení kromě registrů Dxx.

Přetečení – Nastaví se do jedničky, když výsledek odčítání bude větší než 2 147 483 647 nebo menší než -2,147,483,648. Module_Status_Code se nastaví na hodnotu 16#0095, což je pouze stavová chyba.

Poznámky
Operátor - se nesmí použít jako unární operátor s výjimkou dekadické konstanty (základ 10) (např. P001 := -P003; je nepřípustné). Chcete-li negovat proměnnou, odečtěte jí od nuly, např. P001 := 0 – P003;.

Operátor *

Provede násobení zdroje1 a zdroje2 se znaménkem a vytvoří 64-bitový výsledek se znaménkem. Výsledek je možno uložit do 32-bitového nebo 64-bitového místa určení.

Syntaxe 1

místo určení := zdroj1 * zdroj2;

Syntaxe 2

místo určení s dvojnásobnou délkou := zdroj1 * zdroj2;

Syntaxe operátoru * má tyto části:

Část	Popis
Místo určení	Jakákoliv logická proměnná pro zápis
místo určení s dvojitou délkou	Kterákoliv proměnná z 64-bitového parametru lokální logiky (registry Dxx).
zdroj1	Jakákoliv proměnná/konstanta lokální logiky pro čtení kromě registrů Dxx.
zdroj2	Jakákoliv proměnná/konstanta lokální logiky pro čtení kromě registrů Dxx.

Přetečení – Nikdy se nenastaví.

Poznámky
Pokud výsledek bude přiřazený do 32-bitové proměnné, uloží se 32 nejméně významných bitů. Všechno, co je navíc, se ořízne.

Druhou syntaxi je možno použít k operacím násobení, kde výsledek bude spadat mimo rozsah +/- 2 miliard.

Operátor /

Provede celočíselné dělení zdroje1 zdrojem 2 se znaménkem a podíl uloží na místo určení. Registr parametru s dvojnásobnou přesností (64 bitů) je možno použít jako čitatel.

Syntaxe 1

místo určení := zdroj1 / zdroj2;

Syntaxe 2

místo určení := zdroj1 s dvojnásobnou délkou / zdroj2;

Syntaxe operátoru / má tyto části:

Část	Popis
Místo určení	Jakákoliv proměnná lokální logiky pro zápis kromě registrů Dxx.
zdroj1	Jakákoliv proměnná nebo konstanta lokální logiky pro čtení.
zdroj1 s dvojnásobnou délkou	Kterákoliv proměnná z 64-bitového parametru lokální logiky (registry Dxx).
zdroj2	Jakákoliv proměnná/konstanta lokální logiky pro čtení kromě registrů Dxx.

Přetečení – Viz **poznámky** níže.

Poznámky
V případě dělení nulou se Module_Status_Code nastaví na 16#2093. V případě přetečení při dělení se Module_Status_Code nastaví na 16#2094.

Přetečení při dělení nebo dělení nulou jsou chyby ***Rychlého zastavení***. Lokální logika se okamžitě zruší a pohyb se zastaví nastavením povelu rychlosti serva na nulu.

K přetečení při dělení dojde, když podíl operace dělení nelze správně zapsat jako 32-bitovou hodnotu se znaménkem. K tomu může dojít, *pouze* když se jako čitatel používá operand s dvojnásobnou délkou (registry Dxx). K dělení nulou dojde, když jmenovatel dělení má hodnotu nula.

Operátor MOD

Operátor MOD vrátí zbytek po celočíselném dělení se znaménkem zdroje1 zdrojem2. Registr parametru s dvojnásobnou přesností (64 bitů) je možno použít jako čitatel.

Syntaxe 1

místo určení := zdroj1 MOD zdroj2;

Syntaxe 2

místo určení := zdroj1 s dvojnásobnou délkou MOD zdroj2;

Syntaxe operátoru MOD má tyto části:

Část	Popis
Místo určení	Jakákoliv proměnná lokální logiky pro zápis kromě registrů Dxx.
zdroj1	Jakákoliv proměnná nebo konstanta lokální logiky pro čtení.
zdroj1 s dvojnásobnou délkou	Kterákoliv proměnná z 64-bitového parametru lokální logiky (registry Dxx).
zdroj2	Jakákoliv proměnná/konstanta lokální logiky pro čtení kromě registrů Dxx.

Přetečení - Viz **poznámky** níže.

Poznámky
V případě dělení nulou se Module_Status_Code nastaví na 16#2093. V případě přetečení při dělení se Module_Status_Code nastaví na 16#2094.

Modulo (zbytek) se vypočítává vykonáním celočíselného dělení a proto operátor MOD má stejné podmínky chyby jako operátor dělení.

K přetečení při dělení dojde, když podíl operace dělení nelze správně reprezentovat jako 32-bitovou hodnotu se znaménkem. K tomu může dojít, *pouze* když se jako čitatel používá operand s dvojnásobnou délkou. K dělení nulou dojde, když jmenovatel dělení má hodnotu nula.

Přetečení při dělení nebo dělení nulou jsou chyby ***Rychlého zastavení***. Lokální logika se okamžitě zruší a pohyb se zastaví nastavením povelu rychlosti serva na nulu.

Funkce ABS

Funkce ABS vrátí velikost proměnné nebo konstanty bez znaménka.

Syntaxe

místo určení := ABS(parametr);

Syntaxe operátoru ABS má tyto části:

Část	Popis
Místo určení	Jakákoliv proměnná lokální logiky pro zápis kromě registrů Dxx.
Parametr	Jakákoliv proměnná/konstanta lokální logiky pro čtení kromě registrů Dxx.

Přetečení – Nastaví se do jedničky, když operand bude mít hodnotu –2 147 483 648. Module_Status_Code se nastaví na hodnotu 16#0096, což je pouze stavová chyba.

Bitové logické operátory lokální logiky

Všechny logické operace se provádějí na principu bit po bitu, například operace BWAND se skládá z 32 operací *AND* mezi jednotlivými vzájemně odpovídajícími bity operandu. Logické operátory mají předponu 'BW', která zvýrazní skutečnost, že to nejsou booleovské operátory.

Tabulka 12-4. Bitové logické operátory

Operátor	Význam
BWAND	Bitový logický AND
BWOR	Bitový logický OR
BWXOR	Bitový logický Exclusive OR
BWNOT	Bitový logický NOT (jedničkový doplněk)

Výrazy používající bitové logické operátory je možno použít v přiřazovacích nebo podmíněných příkazech. Na jeden výraz je možno použít pouze jeden bitový logický operátor.

Operátor BWAND

Provede bitové *and* zdroje1 a zdroje2.

Syntaxe 1

místo určení := zdroj1 BWAND zdroj2;

Syntaxe 2

IF zdroj1 BWAND zdroj2 THEN

Syntaxe operátoru BWAND má tyto části:

Část	Popis
Místo určení	Jakákoliv proměnná lokální logiky pro zápis kromě registrů Dxx.
zdroj1	Jakákoliv proměnná/konstanta lokální logiky pro čtení kromě registrů Dxx.
zdroj2	Jakákoliv proměnná/konstanta lokální logiky pro čtení kromě registrů Dxx.

Poznámky
Syntaxe 1 se používá pro přiřazování, syntaxe 2 se používá pro vyhodnocování podmínky.

Operátor BWOR

Operátor BWOR vrátí bitové *or* zdroje1 a zdroje2.

Syntaxe 1

místo určení := zdroj1 BWOR zdroj2;

Syntaxe 2

IF zdroj1 BWOR zdroj2 THEN

Syntaxe operátoru BWOR má tyto části:

Část	Popis
Místo určení	Jakákoliv proměnná lokální logiky pro zápis kromě registrů Dxx.
zdroj1	Jakákoliv proměnná/konstanta lokální logiky pro čtení kromě registrů Dxx.
zdroj2	Jakákoliv proměnná/konstanta lokální logiky pro čtení kromě registrů Dxx.

Poznámky
Syntaxe 1 se používá pro přiřazování, syntaxe 2 se používá pro vyhodnocování podmínky.

Operátor BWXOR

Operátor BWXOR vrátí bitový exclusive or zdroje1 a zdroje2.

Syntaxe 1

místo určení := zdroj1 BWXOR zdroj2;

Syntaxe 2

IF zdroj1 BWXOR zdroj2 THEN

Syntaxe operátoru BWXOR má tyto části:

Část	Popis
Místo určení	Jakákoliv proměnná lokální logiky pro zápis kromě registrů Dxx.
zdroj1	Jakákoliv proměnná/konstanta lokální logiky pro čtení kromě registrů Dxx.
zdroj2	Jakákoliv proměnná/konstanta lokální logiky pro čtení kromě registrů Dxx.

Poznámky
Syntaxe 1 se používá pro přiřazování, syntaxe 2 se používá pro vyhodnocování podmínky.

Operátor BWNOT

Operátor BWNOT vrátí jedničkový doplněk zdrojového parametru.

Syntaxe 1

místo určení := zdroj BWNOT ;

Syntaxe 2

IF zdroj BWNOT THEN

Syntaxe operátoru BWNOT má tyto části:

Část	Popis
Místo určení	Jakákoliv proměnná lokální logiky pro zápis kromě registrů Dxx.
zdroj1	Jakákoliv proměnná/konstanta lokální logiky pro čtení kromě registrů Dxx.

Poznámky

Syntaxe 1 se používá pro přiřazování, syntaxe 2 se používá pro vyhodnocování podmínky.

Operátory porovnání

Operátory porovnání vytvoří relační výrok mezi dvěma operandy. Výraz porovnání vyhodnotí podmínku na základě operandů hodnoty celého čísla se znaménkem.

Tabulka 12-5. Relační operátory

Operátor	Význam
<	Menší než
>	Větší než
<=	Menší nebo rovno
>=	Větší nebo rovno
=	Rovná se
<>	Nerovná se

Operátory porovnání je možno použít pouze jako výrazy v podmíněných příkazech a v jednom výrazu je možno použít pouze jeden operátor porovnání.

IF zdroj1 Op_porovnání zdroj2 THEN

Operátory porovnání mají tyto části:

Část	Popis
zdroj1	Jakákoliv booleovská nebo 32-bitová proměnná lokální logiky nebo číselná konstanta určená pro čtení.
zdroj2	Jakákoliv booleovská nebo 32-bitová proměnná lokální logiky nebo číselná konstanta určená pro čtení.
Op_porovnání	Jakýkoliv relační operátor / logický operátor

Poznámky

Následující tabulka obsahuje seznam operátorů porovnání a podmínky, které určí, jestli se výsledek vyhodnotí jako Pravdivý nebo Nepravdivý:

Tabulka 12-6. Operátory porovnání lokální logiky

Relace	Operátor	Pravdivý, když	Nepravdivý když
Menší než	<	zdroj1 < zdroj2	zdroj1 >= zdroj2
Menší nebo rovno	<=	zdroj1 <= zdroj2	zdroj1 > zdroj2
Větší než	>	zdroj1 > zdroj2	zdroj1 <= zdroj2
Větší nebo rovno	>=	zdroj1 >= zdroj2	zdroj1 < zdroj2
Rovnost	=	zdroj1 = zdroj2	zdroj1 <>zdroj2
Nerovnost	<>	zdroj1 <> zdroj2	zdroj1 = zdroj2
Prázdný operátor (IF *Proměnná* THEN….)	žádný	*Proměnná* je nenulová	*Proměnná* je nulová
Bitové logické operátory	BWAND, BWOR,BWXOR, BWNOT	Výsledek je nenulový	Výsledek je nulový

Chyby lokální logiky během vykonávání

Stav přetečení

Některé aritmetické operace mohou mít výsledky, které nelze správně reprezentovat jako 32-bitovou hodnotu se znaménkem. V následujícím kódovém segmentu je ukázaný příklad.

```
P001 := 16#7FFF_FFFF;  // (1) P001 ← 2 147 483 647 (max. kladná 32-bitová hodnota)

P003 := P001 + 1;      // (2) ! Přetečení !
```

Na prvním řádku se do P001 načte 2 147 483 647, největší hodnota, kterou lze zapsat jako 32-bitovou hodnotu dvojkového doplňku se znaménkem. Na druhém řádku se k této hodnotě přičte jednička. Součet zapsaný v hexadecimálním tvaru je 16#8000_0000. Tato hodnota interpretovaná jako 32-bitové číslo ve dvojkovém doplňku se znaménkem představuje zápornou hodnotu –2 147 483 648 a ne kladnou hodnotu 2 147 483 648! V mnoha situacích by tento výsledek byl neočekávaný a měl nežádoucí účinky v následujících programových příkazech.

K detekci přetečení v programu lokální logiky existují tři proměnné. Proměnná *Přetečení* je booleovská proměnná pro zápis-čtení, která je k dispozici pouze pro program lokální logiky (viz kapitola 13, část "Systémové proměnné lokální logiky"). Když dojde k přetečení a proměnná Přetečení se nevynuluje před koncem cyklu lokální logiky, slovo stavového kódu %AI modulu DSM (proměnná lokální logiky *Module_Status_Code*) se nastaví do jedničky. Chybový kód udává typ přetečení a bit %I Výskyt chyby modulu (proměnná lokální logiky *Module_Error_Present*) se nastaví do jedničky. Slovo %AI stavového kódu modulu a bit %I Výskyt chyby modulu se nenastaví do jedničky, dokud se nedokončí vykonávání aktuálního cyklu lokální logiky. Naopak, proměnná *Přetečení* se nastaví do jedničky hned po instrukci, která přetečení způsobila. Program lokální logiky může proměnnou *Přetečení* vynulovat tak, že do ní zapíše nulu. Stavový kód modulu musí vynulovat PLC nastavením bitu %Q Vynulování chyby.

Chyby přetečení a výpočtu jsou chyby *pouze stavové* se dvěma výjimkami. Dělení nulou nebo přetečení při dělení (podíl nelze vyjádřit ve 32 bitech) jsou chyby *Rychlého zastavení*. V případě *pouze stavových* chyb vykonávání lokální logiky a generování dráhy bude pokračovat normálně. Chyba *rychlého zastavení* způsobí, že vykonávání lokální logiky se přeruší před postupem na další instrukci a všechny pohyby se zruší nastavením povelu rychlosti serva na nulu. **Poznámka:** Vynulování proměnné Přetečení nemá žádný vliv na chyby Rychlého zastavení.

Seznam všech chybových kódů při vykonávání lokální logiky najdete v tabulce 12-10.

Dělení nulou

Výkonný program logiky označí operace Dělení nulou jako chyby Rychlého zastavení, protože výsledek operace je nedefinovaný. Vykonávání lokální logiky a pohybu serva se zastaví. Pokud pohony byly povolené, ve stavovém kódu modulu se bude hlásit chybový kód 16#2093 a chybové kódy 16#2x9A na jednotlivé osy.

Výstraha / chyba překročení času hlídacího obvodu

Programy lokální logiky musí dokončit své vykonávání během 300 mikrosekund ve výkonném programu logiky. Důvodem je, aby modul měl dostatek času na zpracování při vygenerování dráhy a na další úkoly. Podrobný seznam dob vykonávání pro všechny platné operace lokální logiky najdete v Dodatku E. Pomocí datových tabulek uvedených v Dodatku E si uživatel si může vypočítat doby vykonávání požadované daným programem. Pokud vykonávání bude trvat déle než 275 mikrosekund, ale méně než 300 mikrosekund, výkonný program logiky předá chybový kód upozornění (Pouze stav). Chyba Rychlého zastavení se bude generovat, když doba vykonávání programu překročí 300 mikrosekund. Seznam kódů výstrah a chyb během vykonávání najdete v Tabulce 12-10.

Poznámka: Vykonávání lokální logiky se zastaví, pokud se vyskytnou nějaké chyby lokální logiky *Rychlého zastavení* (viz tabulka 12-10) a všechny digitální výstupy, řídicí bity (Jog, Zastavení posuvu, Resety vzorkování, Povolení vlečené osy) a bity CTL, které řídí lokální logiky, budou zakázané. Vykonávání lokální logiky se obnoví od začátku, když uživatel tuto chybu vynuluje (pomocí bitu %Q vynulování chyby).

Chybová hlášení lokální logiky

Chybová hlášení při sestavování lokální logiky

Proces sestavování programu lokální logiky předává stav sestavování přes okno editoru protokolu chyb lokální logiky. V případě výskytu chyby proces sestavování chybu nahlásí a bude snažit v procesu sestavování pokračovat.

Chybová hlášení generovaná procesem sestavování lokální logiky spadají do tří kategorií: syntaktické chyby, chyby kompilačního analyzátoru a výstrahy kompilačního analyzátoru.

Chybová hlášení kompilačního analyzátoru mají několik společných prvků.

Název souboru (řádek): [závažnost] [chybové hlášení]

Název souboru je název aktuálně sestavovaného souboru.

Řádek je číslo řádku v souboru, ve kterém byla zjištěna chyba.

Závažnost popisuje závažnost chyby. Chyby nedovolí vytvoření binárního kódu. Výstrahy jsou pro informaci.

Chybové hlášení je krátký všeobecný popis chyby.

Syntaktické chyby lokální logiky

Proces sestavování vynucuje syntaxi lokální logiky. Pokud zdrojový program nebude splňovat toto kritérium, proces sestavování bude hlásit chybu syntaxe. Chybové hlášení identifikuje chybu jako chybu syntaxe. Chybové hlášení obsahuje nalezený typ syntaktického prvku následovaný jedním nebo několika syntaktickými prvky, které kompilační analyzátor očekával. Je běžné, že syntaktické chyby se aktuálně hlásí na řádku následujícím za řádkem s aktuální chybou. Typickým příkladem jsou chybějící středníky.

Příklad:

scratch.llp (3): Error :syntax error

actual: IF expecting: ;

V tomto případě na řádku 2 chybí středník. Protože středník může ve skutečnosti následovat na jiném řádku, kompilační analyzátor nehlásí chybu dřív, než si prohlídne významný syntaktický prvek, který není středník.

Vzhledem ke své povaze jedna chyba syntaxe může způsobit "kaskádování chyb". Oprava jedné chyby může odstranit několik hlášení chyby syntaxe. Aby při ladění programů lokální logiky se syntaktickými chybami nedocházelo k omylu, opravte první chybu a před pokračováním sestavte program znovu, aby se obnovil seznam chyb.

Chyby kompilačního analyzátoru lokální logiky

Chyby kompilačního analyzátoru se vyskytnou, když syntaxe programu bude správná, ale objeví se sémantický problém. Například proměnné s dvojnásobnou přesností nelze přiřadit jinou hodnotu než výsledek operace násobení.

Příklady:

```
Error (P203) Invalid assignment to Double precision var: D00
(Chyba - neplatné přiřazení proměnné s dvojnásobnou přesností)
```

V tomto případě za chybovým hlášením bude následovat řetězec, který identifikuje symbol, který způsobil chybu. Seznam chyb kompilačního analyzátoru a jejich typické příčiny jsou uvedené v následující tabulce:

Tabulka 12-7. Chyby kompilačního analyzátoru lokální logiky

Číslo chyby	Popis chyby
(P200)	Nedefinovaný identifikátor Program obsahuje neznámou proměnnou nebo klíčové slovo. Zkontrolujte hláskování a syntaxi povelu.
(P201)	Registr parametrů musí být v rozsahu P255 - P255 Tato chyba se generuje, když program zadá nedefinovaný registr parametrů, například P278.
(P202)	Proměnná CTL musí být v rozsahu CTL01 - CTL32 Tato chyba se generuje, když program zadá nedefinovanou proměnnou CTL, například CTL35.
(P203)	Neplatné přiřazení proměnné s dvojnásobnou přesností Program se snaží o neplatné použití jednoho z registrů s dvojnásobnou přesností. Do registrů s dvojnásobnou přesností je možno přiřadit pouze hodnoty výsledku operace násobení.
(P204)	Neplatné použití proměnné s dvojnásobnou přesností Program se snaží o neplatné použití jednoho z registrů s dvojnásobnou přesností. Registry s dvojnásobnou přesností ve výrazu je možno použít pouze jako dělenec v operaci dělení nebo MOD.
(P205)	Zápis do proměnné pouze pro čtení Program se snaží přiřadit hodnotu do proměnné pouze pro čtení.
(P206)	Snaha číst proměnnou pouze pro zápis Program se snaží použít proměnnou pouze pro čtení jako jeden z operandů v aritmetickém, logickém nebo relačním výrazu.
(P207)	Indexované proměnné nejsou podporované Proměnné ve tvaru Data_Table_Int[xx] nejsou podporované. Operace datové tabulky vyžadují použít proměnnou Data_Table_Ptr.
(P208)	Název identifikátoru je delší než 50 znaků Program se snaží adresovat proměnnou s identifikátorem delším než 50 znaků.
(P209)	Registr s dvojnásobnou přesností musí být v rozsahu D00 - D07. Tato chyba se generuje, když program zadá nedefinovaný registr s dvojnásobnou přesností, například D08.
(P220)	Hexadecimální konstanty musí být v rozsahu 16#0 - 16#FFFFFFFF Program definoval hexadecimální konstantu, kterou nelze vyjádřit pomocí 32 bitů.

Číslo chyby	Popis chyby
(P221)	Binární konstanty musí být v rozsahu 0 až (2^{31})-1 Program definoval binární konstantu, kterou nelze vyjádřit pomocí 32 bitů.
(P222)	Celočíselné konstanty musí být v rozsahu -2 147 483 648 až 2 147 483 647 Program definoval dekadickou konstantu, kterou nelze vyjádřit pomocí 32 bitů.
(P223)	Přetečení tabulky konstant Program může obsahovat maximálně 50 jednoznačných konstant větších než 2047 nebo menších než –2048 (tj. čísla, která nelze vyjádřit méně než 12 bity). Program může obsahovat libovolný počet konstant v rozsahu –2048 až 2047.
(P230)	Překročeno 8 úrovní vnořování IF. Příkazy IF se nesmí vnořovat do větší úrovně než 8.
(P231)	Nepřípustný termín v příkazu IF Program má aritmetický operátor v příkazu IF.
(P232)	Chybí příkaz END_IF Existuje příkaz IF, u kterého chybí odpovídající příkaz END_IF. Tato chyba se zjistí pouze na konci programu.
(P233)	Zjištěný nepárovaný END_IF Byl zjištěný příkaz END_IF, který k sobě nemá odpovídající příkaz IF.
(P240)	Přiřazení konstantě Program se snaží použít konstantu jako místo určení příkazu přiřazení.
(P241)	Neplatný operátor, očekávalo se přiřazení Zjistil se jiný operátor, kde se očekával operátor přiřazení (:=).
(P242)	Relační operátor je nepřípustný v příkazu přiřazení Snaha provést operaci porovnání v příkazu přiřazení. Přiřazení na základě relace je možno provést přiřazením booleovské proměnné v příkazu IF.
(P260)	Neplatný logický operátor. Použijte BWAND, BWOR, BWXOR nebo BWNOT - <operátor> Program použil klíčové slovo AND, OR, XOR nebo NOT místo BWAND, BWOR, BWXOR respektive BWNOT.
(P280)	Překročení meze instrukce, maximálně 150 Program lokální logiky může mít délku maximálně 150 příkazů. Tato chyba se bude hlásit, když program přesáhne tuto délku.
(P290)	Adresa mimo rozsah paměti s přímým přístupem Proměnná paměti s přímým přístupem zadala neplatný offset.
(P291)	Neplatná proměnná paměti s přímým přístupem Byla zadána neplatná proměnná paměti s přímým přístupem.
(P292)	Proměnná paměti s přímým přístupem vyžaduje index Program adresoval proměnnou paměti s přímým přístupem bez zadání ofsetu.
(P293)	Překročen maximální počet chyb. Kompilační analyzátor lokální logiky bude hlásit maximálně 30 chyb. Když se překročí tato mez, zobrazí se toto hlášení a další chyby se už hlásit nebudou.
298-(P299)	Interní chyba. Kontaktujte GE Fanuc Technical Support. Pokud kompilační analyzátor bude u uživatelského programu hlásit chybu 298 nebo 299, uvědomte o tom technickou podporu GE Fanuc (1-800-GE FANUC). Předejte kopii programu a protokol chyb.
(P300)	Instrukce syntaktické analýzy musí předcházet před vykonatelnými příkazy Instrukce #pragma se musí v programovém bloku lokální logiky objevit před jakýmkoliv vykonatelným příkazem.

Číslo chyby	Popis chyby
(P301)	Neplatný výběr instrukce Kompilační analyzátor lokální logiky nezná zadanou instrukci #pragma.
(P302)	Neplatný parametr instrukce Byl zadaný neplatný argument do instrukce #pragma pouze chyby. Argument musí být 1, ON, 0 nebo OFF.

Výstrahy kompilačního analyzátoru lokální logiky

Výstrahy kompilačního analyzátoru se generují u stavů, které mohou mít neočekávané výsledky nebo které indikují možné opomenutí v programu lokální logiky.

Tabulka 12-8. Výstrahy kompilačního analyzátoru lokální logiky

Číslo chyby	Popis chyby
(P400)	Zápis do binární proměnné může vést ke ztrátě dat. Toto hlášení se generuje, když booleovská proměnná bude přiřazená z nebooleovské proměnné nebo konstanty nebo z výrazu obsahujícího nebooleovské proměnné.
(P410)	Kontrola doby vykonávání instrukce Toto hlášení se generuje u programů, které mají více než 100 příkazů. Protože maximální počet příkazů je 150, je možné napsat program lokální logiky, který by byl příliš dlouhý na vykonávání. Doby instrukcí najdete v Dodatku A v *Referenční příručce pro CPU Series 90-30,* GFK-0467.
(P481)	Zastaralá syntaxe: parametr funkce vyžaduje závorky Parametr volání funkce ABS není uzavřený v závorkách.
(P482)	Neočekávaný konec programu: neuzavřený komentář Komentář otevřený dvojicí znaků "(*" nebyl při zjištění konce programu uzavřený.
(P483)	Vnořované komentáře Toto hlášení se generuje, když program lokální logiky definoval text komentáře v jiném komentáři.
(P490)	Program neobsahuje žádné vykonatelné příkazy. Program obsahuje pouze prázdné místo a/nebo komentáře.

Chybová hlášení při načítání lokální logiky

Když se program lokální logiky bude načítat do modulu, ve stavovém kódu modulu se mohou hlásit následující chyby.

Tabulka 12-9. Chybové kódy konfigurace lokální logiky

Chybový kód (hexadecimálně)	Odezva	Popis	Typ chyby
0A	Systémová chyba	Neplatná konfigurace digitálního výstupu	Modul
0B	Systémová chyba	Neplatná konfigurace bitu CTL	Modul

Poznámka: Podrobný popis konfigurace bitů CTL a digitálních výstupů lokální logiky najdete v kapitole 14.

Tabulka 12-10. Chybové kódy předzpracování lokální logiky

Chybový kód (hexadecimálně)	Odezva	Popis	Typ chyby
F0A0	Systémová chyba	Chyba záhlaví programu lokální logiky	Modul
F0A1	Systémová chyba	Chyba zakončujícího znaku programu lokální logiky	Modul
F0A2	Systémová chyba	Chyba záhlaví konstanty programu lokální logiky	Modul
F0A3	Systémová chyba	Chyba zakončujícího znaku konstanty programu lokální logiky	Modul
F0A4	Systémová chyba	Chyba ukazatele konstanty programu lokální logiky	Modul
F0A5	Systémová chyba	Překročení limitu kompilovaného kódu programu lokální logiky	Modul
F0A6	Systémová chyba	Chyba nepárovaného IF_THEN programu lokální logiky	Modul
F0A7	Systémová chyba	Chyba nepárovaného END_IF programu lokální logiky	Modul
F0A8	Systémová chyba	Překročení limitu vnořování programu lokální logiky	Modul
F0A9	Systémová chyba	Chyba čtení programu lokální logiky	Modul
F0AA	Systémová chyba	Chyba vyhrazené třídy programu lokální logiky	Modul
F0AB	Systémová chyba	Neplatný registr parametrů programu lokální logiky	Modul
F0AC	Systémová chyba	Neplatný registr s dvojnásobnou přesností programu lokální logiky	Modul
F0AD	Systémová chyba	Chyba digitálního výstupu programu lokální logiky	Modul
F0AE	Systémová chyba	Chyba bitu CTL programu lokální logiky	Modul
F0B0	Systémová chyba	Neplatný primární operátor programu lokální logiky	Modul
F0B1	Systémová chyba	Neplatný sekundární operátor programu lokální logiky	Modul
F0B2	Systémová chyba	Neplatný sekundární zdroj lokální logiky	Modul
F0B3	Systémová chyba	Neplatný primární zdroj programu lokální logiky	Modul
F0B4	Systémová chyba	Neplatný zdroj programu lokální logiky	Modul
F0B5	Systémová chyba	Chyba zdroje pouze pro zápis programu lokální logiky	Modul
F0B6	Systémová chyba	Chyba adresy přímého přístupu do paměti programu lokální logiky	Modul
F0B7	Systémová chyba	Neplatné místo určení programu lokální logiky	Modul
F0B8	Systémová chyba	Místo určení pouze pro čtení programu lokální logiky	Modul

Chyby během vykonávání lokální logiky

Následující chyby a upozornění se mohou hlásit, když se v modulu vykonává program lokální logiky.

Tabulka 12-11. Chybové kódy lokální logiky během vykonávání

Chybový kód (hexadecimálně)	Odezva	Popis	Typ chyby
91	Rychlé zastavení	Zadané zastavení systému programu lokální logiky	Modul
92	Rychlé zastavení	Překročení limitu doby vykonávání lokální logiky	Modul
93	Rychlé zastavení	Dělení nulou lokální logiky	Modul
94	Rychlé zastavení	Přetečení při dělení lokální logiky	Modul
95	Pouze stav	Přetečení při sčítání/odčítání lokální logiky	Modul
96	Pouze stav	Přetečení absolutní hodnoty (ABS) lokální logiky	Modul
97	Pouze stav	Výstraha překročení limitu doby vykonávání lokální logiky	Modul
98	Pouze stav	Chyba prvního cyklu vykonávání lokální logiky	Modul
99	Pouze stav	Neplatný nebo chybějící název programu lokální logiky v hardwarové konfiguraci	Modul
9A	Rychlé zastavení	Chyba zastavení lokální logiky	Pro danou osu

Kapitola 13

Proměnné lokální logiky

Typy proměnných lokální logiky

Lokální logika vykonává přístup k proměnným kontroléru pohybu a registrům parametru pomocí předem definovaných názvů proměnných. Úplný seznam všech proměnných lokální logiky najdete v Tab. 13-1 až Tab. 13-6.

Příklady:

```
IF Actual_Position_2 > 5000 THEN ...;

IF Strobe1_Level_2 = ON THEN ...;
```

Uložení hodnot do proměnných se provádí pomocí operátoru přiřazení ":=".

Příklady:

```
Torque_Limit_2 := 8500;       (* Nastavení meze krouticího momentu na 85% * )

Position_Loop_TC_1 := 2500 / Actual_Velocity_1;
```

Proměnné lokální logiky se dělí na dvě kategorie: globální proměnné a proměnné pro danou osu. Existují čtyři soubory proměnných osy (Axis1 - Axis4). Každý soubor proměnných se dále dělí na **Řídicí proměnné**, **Stavové proměnné** a **I/0 na čelní desce** (viz Tab. 13-1 až Tab. 13-6). Popis termínů používaných v tabulkách proměnných je uvedený níže.

Atribut proměnné

Atribut pro každou proměnnou lokální logiky je uvedený v Tab. 13-1 až Tab. 13-6. Proměnné mohou být **Pouze pro čtení**, **Pouze pro zápis** nebo **Pro čtení i zápis**. Pokud se uživatel bude snažit zapisovat do proměnné pouze pro čtení nebo číst z proměnné pouze pro zápis, kompilační analyzátor bude hlásit chybu.

Velikost proměnné

Proměnné lokální logiky mají velikost v rozsahu od 1 bitu (bitový operand) až 64 bitů (pro registry s dvojnásobnou přesností Dxx). Seznam velikostí jednotlivých proměnných lokální logiky najdete v Tab. 13-1 až Tab. 13-6. Snaha zapsat hodnotu větší než je velikost dané proměnné bude mít za následek hodnotu, která bude oříznutá. Pokud například výsledek matematické operace bude mít délku 32 bitů a bude zapisovaný do 16-bitové proměnné, uloží se pouze dolních 16 bitů. Pokud se použije bitový operand jako místo určení proměnné v nebooleovské matematické operaci, kompilační analyzátor vydá upozornění (uloží se pouze nejméně významný bit výsledku).

Poznámka: Proměnné povelu AQ (Mez krouticího momentu, Zisk rychlostní smyčky, Poměr vlečené osy, Polohový inkrement a Časová konstanta polohové smyčky) mohou mít přípustný rozsah, který je menší než je velikost proměnné lokální logiky. Pokud se program bude snažit zapsat hodnotu mimo platný rozsah povelu AQ, modul bude hlásit výstražný chybový kód a odmítne všechny neplatné hodnoty. Popis přípustných rozsahů povelů %AQ najdete v kapitole 14.

Znaménko proměnné

Proměnné lokální logiky, které mají délku menší než 32 bitů, jsou buď se znaménkem nebo bez znaménka (kromě bitových operandů, které jsou vždy bez znaménka). Všechny matematické/logické operace v nástroji logiky jsou 32-bitové operace se znaménkem (kromě 64-bitových operací dělení a dělení modulo). U proměnných se znaménkem, které mají délku menší než 32 bitů, se znaménko automaticky prodlouží na 32 bitů při načtení do nástroje logiky. U proměnných bez znaménka se znaménko neprodlouží. Proto nástroj logiky provádí všechny převody dat a automaticky kontroluje jejich meze.

Systémové proměnné lokální logiky

Proměnné *First_Local_Logic_Sweep*, *Přetečení* a *System_Halt* se používají výhradně v nástroji logiky a jejich popis je uvedený níže.

Proměnná First_Local_Logic_Sweep

Proměnná *First_Local_Logic_Sweep* je bitový operand **Pouze pro čtení** (viz Tab. 13-5). Nastavuje jí nástroj logiky během prvního vykonání cyklu, když se povolí lokální logika a PLC je v režimu RUN. Vynuluje se v následujících cyklech. Proto jí je možno použít v lokální logice pro výchozí nastavení některých proměnných. Například níže uvedený kód nastaví některé registry parametrů a řídicí proměnné v prvním cyklu pomocí proměnné *First_Local_Logic_Sweep*.

```
IF First_Local_Logic_Sweep THEN    (* Pokud to je první vykonání cyklu, pak *)
    P001 := 0;                     (* inicializace P001 na 0 *)
    P015 := 1000;                  (* inicializace P015 na 1000 *)
    Velocity_Loop_Gain_1 := 20;    (* Nastavení zisku rychlostní smyčky osy 1 na 20 *)
END_IF;
```

Proměnná přetečení

Proměnná *Přetečení* je operand **Pro čtení a pro zápis** (viz Tab. 13-5). Nastavuje jí nástroj logiky, když se vyskytne přetečení při sčítání, odčítání nebo absolutní hodnotě (ABS). Pokud bude nastavený příznak Přetečení a vyskytne se chyba přetečení (viz kapitola 12), stavový kód modulu bude hlásit výstražný chybový kód. Všimněte si, že je možno zakázat hlášení výstrah přetečení při sčítání/odčítání/ABS nastavením proměnné *Přetečení* na nulu na konci programu lokální logiky. Obdobně je možno zjišťovat chyby Přetečení v programu lokální logiky přečtením proměnné *Přetečení* a vykonáním některého příslušného kroku. Proměnná *Přetečení* se vynuluje za následujících okolností:

1) Vynuluje se automaticky při spuštění lokální logiky před vykonáním prvního cyklu.

2) Může jí vynulovat uživatel v programu lokální logiky (nastavením *Overflow := 0;*).

3) Vynuluje se, když uživatel překlopí bit vynulování chyby Q.

Proměnná System_Halt

Proměnná *System_Halt* je bitový operand **Pouze pro zápis** (viz Tab. 13-5). Pokud program lokální logiky zapíše 1 do proměnné *System_Halt*, pohyb serva a vykonání lokální logiky se zastaví. Ve stavovém kódu modulu se bude také hlásit chybový kód (viz kapitola 12). Proto proměnnou *System_Halt* je možno použít k zachycení stavu fatální chyby a provést odstranění chyby. Níže uvedený příklad kódu ukazuje možný scénář, ve kterém je možno použít proměnnou *System_Halt*:

```
IF Overflow THEN                (* Zachycení přetečení *)
   System_Halt := TRUE;         (* Zastavení vykonávání lokální logiky a pohybu serva *)
END_IF;
```

64-bitové registry pro dvojnásobnou přesnost

Kromě 255 registrů s délkou 32 bitů lokální logika má osm 64-bitových registrů (D00-D07) (viz Tab. 13-5). To umožní uživateli uložit výsledek násobení dvou 32-bitových čísel do registru Dxx a pak s výsledkem provést operaci dělení/dělení modulo. Proto je možno 64-bitové registry použít za následujících okolností:

1. Jako registr místa učení pro operaci násobení.
2. Jako dělenec (čitatel) v operaci dělení/dělení modulo.

Pokud se použije v jiných operacích, kompilační analyzátor bude hlásit chybu. Příklad kódu pro použití registrů Dxx je uvedený níže:

```
D01:= P001 * 2147483647;  (* Provede operaci násobení a uložení do 64-bitového registru *)
P010:= D01 / 12500;       (* Vydělí výsledek a uloží ho do 32-bitového registru *)
```

Všimněte si, že výše uvedený scénář může vést k **Přetečení při dělení,** pokud se výsledek nevejde do 32-bitového registru. Přetečení při dělení zastaví vykonávání lokální logiky a pohybu serva, protože výsledek operace bude nedefinovaný (viz kapitola 12). Ve stavovém kódu modulu se bude také hlásit chybový kód.

Poznámka

Obsah 64-bitových datových registrů (D00-D07) a 32-bitových datových registrů pro zápis (P000-P255) se automaticky inicializuje lokální logikou při jejím spuštění. Uživatel musí všechny ostatní proměnné inicializovat pomocí samostatného programu lokální logiky nebo proměnné *First_Local_Logic_Sweep* nebo PLC žebříkové logiky.

Tabulka uživatelských dat lokální logiky

Lokální logika má **kruhovou paměť s 8192 bajty**, kterou program lokální logiky může použít k uložení a načtení dat. Viz Tab. 13-5 se seznamem proměnných *Data_Table*. Do tabulky dat je přístup pomocí nepřímého adresování paměti. Proměnná *Data_Table_Ptr* ("Ukazatel") se používá k označení správného paměťového bajtu v kruhové paměti s 8192 bajty. Proto velikost proměnné *Data_Table_Ptr* je 13 bitů (přípustný rozsah 0-8191). Ukazatel se automaticky inkrementuje, když se hodnota načte nebo se zapíše do kruhové paměti. Velikost, o kterou se ukazatel inkrementuje, závisí na velikosti adresované proměnné. *Data_Table_Ptr* se při spuštění lokální logiky před vykonáním prvního cyklu automaticky inicializuje na 0. Proto proměnné z tabulky dat je možno použít pro přístup k velkému bloku předem načtených dat v programu lokální logiky. Pro přístup do kruhové paměti se používají následující proměnné.

Data_Table_Ptr	: Ukazatel datové tabulky - platný rozsah 0-8191.	
Data_Table_sint	: 8 bitů se znaménkem	(Ukazatel se při čtení/zápisu automaticky inkrementuje o 1)
Data_Table_usint	: 8 bitů bez znaménka	(Ukazatel se při čtení/zápisu automaticky inkrementuje o 1)
Data_Table_in	: 16 bitů se znaménkem	(Ukazatel se při čtení/zápisu automaticky inkrementuje o 2)
Data_Table_uint	: 16 bitů bez znaménka	(Ukazatel se při čtení/zápisu automaticky inkrementuje o 2)
Data_Table_dint	: 32 bitů bez znaménka	(Ukazatel se při čtení/zápisu automaticky inkrementuje o 4)

Následující příklad kódu ukazuje, jak je možno adresovat konkrétní paměťové místo v kruhové paměti:

```
Data_Table_Ptr := 100;      (* Ukazatel na bajt s offsetem 100 v kruhové paměti *)
P001 := Data_Table_int;     (* Načtení 16-bitového čísla se znaménkem z paměti *)
                            (* Data_Table_Ptr se automaticky inkrementuje na 102 *)
Data_Table_sint := -120;    (* Zápis 8-bitového čísla se znaménkem do bajtu 102 *)
Data_Table_Ptr := 0;        (* Ukazatel na bajt s offsetem 0 v kruhové paměti *)
```

Digitální výstupy / proměnné CTL

8 digitálních výstupů v modulu (2 na osu) je možno nakonfigurovat samostatně tak, že budou buď

řízené PLC (**výchozí**) nebo řízené lokální logikou (DSM). Pokud program lokální logiky zapisuje do konkrétní proměnné *Digital_Output* (viz Tab. 13-1 až Tab. 13-6), musí být nakonfigurovaný na řízení z DSM. Modul DSM odmítne programy lokální logiky, které jsou načtené s nesprávnou konfigurací digitálního výstupu. Podrobný popis konfigurace digitálních výstupů najdete v kapitole 14.

CTL01-CTL24 je možno také nakonfigurovat samostatně tak, aby měly jiné vstupní zdroje. Podrobný popis možností konfigurace najdete v kapitole 14. CTL25 až CTL32 nejsou konfigurovatelné a jsou vždy řízené lokální logikou. Modul DSM odmítne programy lokální logiky, které jsou načtené s nesprávnou konfigurací CTL. Pokud například program lokální logiky bude mít příkaz, který zapisuje do CTL16 (např. *CTL16 := 1*;), pak CTL16 musí být v hardwarové konfiguraci VersaPro nakonfigurovaný jako "řízený lokální logikou". CTL01 až CTL32 a čísla bloků pohybového programu (proměnné *Block_1, Block_2, Block_3, Block_4*) je možno použít k synchronizaci pohybového programu a programu lokální logiky.

Tab. 13-1. Proměnné osy 1

Název proměnné lokální logiky	Atribut	Velikost
I/O na čelní desce		
Strobe1_Level_1	Pouze čtení	Bitový operand
Strobe2_Level_1	Pouze čtení	Bitový operand
Positive_EOT_1	Pouze čtení	Bitový operand
Negative_EOT_1	Pouze čtení	Bitový operand
Home_Switch_1	Pouze čtení	Bitový operand
Digital_Output1_1 (1)	Pouze zápis	Bitový operand
Digital_Output3_1 (1)	Pouze zápis	Bitový operand
Analog_Input1_1	Pouze čtení	16 bitů se znaménkem
Analog_Input2_1	Pouze čtení	16 bitů se znaménkem
Řídicí proměnné		
Velocity_Loop_Gain_1	Čtení/zápis	8 bitů bez znaménka
Position_Loop_TC_1	Pouze zápis	16 bitů bez znaménka
Torque_Limit_1	Pouze zápis	16 bitů bez znaménka
Follower_Ratio_A_1	Pouze zápis	16 bitů se znaménkem
Follower_Ratio_B_1	Pouze zápis	16 bitů se znaménkem
Position_Increment_Cts_1 (2)	Pouze zápis	16 bitů se znaménkem
Reset_Strobe1_1	Pouze zápis	Bitový operand
Reset_Strobe2_1	Pouze zápis	Bitový operand
Enable_Follower_1	Pouze zápis	Bitový operand
Jog_Plus_1	Pouze zápis	Bitový operand
Jog_Minus_1	Pouze zápis	Bitový operand
FeedHold_1	Pouze zápis	Bitový operand
Stavové proměnné		
Error_Code_1	Pouze čtení	16 bitů bez znaménka
Block_1	Pouze čtení	16 bitů bez znaménka
Actual_Position_1	Pouze čtení	32 bitů
Commanded_Position_1	Pouze čtení	32 bitů
Position_Error_1	Pouze čtení	32 bitů
Strobe1_Position_1	Pouze čtení	32 bitů
Strobe2_Position_1	Pouze čtení	32 bitů
Actual_Velocity_1	Pouze čtení	32 bitů
Commanded_Velocity_1	Pouze čtení	32 bitů
Commanded_Torque_1	Pouze čtení	32 bitů
User_Selected_Data1_1	Pouze čtení	32 bitů
User_Selected_Data2_1	Pouze čtení	32 bitů
UnAdjusted_Actual_Position_Cts_1	Pouze čtení	32 bitů
UnAdjusted_Strobe1_Position_Cts_1	Pouze čtení	32 bitů
UnAdjusted_Strobe2_Position_Cts_1	Pouze čtení	32 bitů
Axis_OK_1	Pouze čtení	Bitový operand
Position_Valid_1	Pouze čtení	Bitový operand
Strobe1_Flag_1	Pouze čtení	Bitový operand
Strobe2_Flag_1	Pouze čtení	Bitový operand
Drive_Enabled_1	Pouze čtení	Bitový operand
Program_Active_1	Pouze čtení	Bitový operand
Moving_1	Pouze čtení	Bitový operand
In_Zone_1	Pouze čtení	Bitový operand
Position_Error_Limit_1	Pouze čtení	Bitový operand
Torque_Limited_1	Pouze čtení	Bitový operand

Název proměnné lokální logiky	Atribut	Velikost
Servo_Ready_1	Pouze čtení	Bitový operand
Follower_Enabled_1	Pouze čtení	Bitový operand
Follower_Velocity_Limit_1	Pouze čtení	Bitový operand
Follower_Ramp_Active_1	Pouze čtení	Bitový operand

Poznámky:

(1) Aby mohla lokální logika do těchto digitálních výstupů zapisovat, musí v hardwarové konfiguraci VersaPro být nakonfigurované na řízení lokální logikou.

(2) Proměnná Position_Increment_Cnts_n má maximální rozsah ±1023 jednotek.

Tab. 13-2. Proměnné osy 2

Název proměnné lokální logiky	Atribut	Velikost
I/O na čelní desce		
Strobe1_Level_2	Pouze čtení	Bitový operand
Strobe2_Level_2	Pouze čtení	Bitový operand
Positive_EOT_2	Pouze čtení	Bitový operand
Negative_EOT_2	Pouze čtení	Bitový operand
Home_Switch_2	Pouze čtení	Bitový operand
Digital_Output1_2 (1)	Pouze zápis	Bitový operand
Digital_Output3_2 (1)	Pouze zápis	Bitový operand
Analog_Input1_2	Pouze čtení	16 bitů se znaménkem
Analog_Input2_2	Pouze čtení	16 bitů se znaménkem
Řídicí proměnné		
Velocity_Loop_Gain_2	Čtení/zápis	8 bitů bez znaménka
Position_Loop_TC_2	Pouze zápis	16 bitů bez znaménka
Torque_Limit_2	Pouze zápis	16 bitů bez znaménka
Follower_Ratio_A_2	Pouze zápis	16 bitů se znaménkem
Follower_Ratio_B_2	Pouze zápis	16 bitů se znaménkem
Position_Increment_Cts_2 (2)	Pouze zápis	16 bitů se znaménkem
Reset_Strobe1_2	Pouze zápis	Bitový operand
Reset_Strobe2_2	Pouze zápis	Bitový operand
Enable_Follower_2	Pouze zápis	Bitový operand
Jog_Plus_2	Pouze zápis	Bitový operand
Jog_Minus_2	Pouze zápis	Bitový operand
FeedHold_2	Pouze zápis	Bitový operand
Stavové proměnné		
Error_Code_2	Pouze čtení	16 bitů bez znaménka
Block_2	Pouze čtení	16 bitů bez znaménka
Actual_Position_2	Pouze čtení	32 bitů
Commanded_Position_2	Pouze čtení	32 bitů
Position_Error_2	Pouze čtení	32 bitů
Strobe1_Position_2	Pouze čtení	32 bitů
Strobe2_Position_2	Pouze čtení	32 bitů
Actual_Velocity_2	Pouze čtení	32 bitů
Commanded_Velocity_2	Pouze čtení	32 bitů
Commanded_Torque_2	Pouze čtení	32 bitů

Název proměnné lokální logiky	Atribut	Velikost
User_Selected_Data1_2	Pouze čtení	32 bitů
User_Selected_Data2_2	Pouze čtení	32 bitů
UnAdjusted_Actual_Position_Cts_2	Pouze čtení	32 bitů
UnAdjusted_Strobe1_Position_Cts_2	Pouze čtení	32 bitů
UnAdjusted_Strobe2_Position_Cts_2	Pouze čtení	32 bitů
Axis_OK_2	Pouze čtení	Bitový operand
Position_Valid_2	Pouze čtení	Bitový operand
Strobe1_Flag_2	Pouze čtení	Bitový operand
Strobe2_Flag_2	Pouze čtení	Bitový operand
Drive_Enabled_2	Pouze čtení	Bitový operand
Program_Active_2	Pouze čtení	Bitový operand
Moving_2	Pouze čtení	Bitový operand
In_Zone_2	Pouze čtení	Bitový operand
Position_Error_Limit_2	Pouze čtení	Bitový operand
Torque_Limited_2	Pouze čtení	Bitový operand
Servo_Ready_2	Pouze čtení	Bitový operand
Follower_Enabled_2	Pouze čtení	Bitový operand
Follower_Velocity_Limit_2	Pouze čtení	Bitový operand
Follower_Ramp_Active_2	Pouze čtení	Bitový operand

Poznámky:

(1) Aby mohla lokální logika do těchto digitálních výstupů zapisovat, musí v hardwarové konfiguraci VersaPro být nakonfigurované na řízení lokální logikou.

(2) Proměnná Position_Increment_Cnts_n má maximální rozsah ±1023 jednotek.

Tab. 13-3. Proměnné osy 3

Název proměnné lokální logiky	Atribut	Velikost
I/O na čelní desce		
Strobe1_Level_3	Pouze čtení	Bitový operand
Strobe2_Level_3	Pouze čtení	Bitový operand
Positive_EOT_3	Pouze čtení	Bitový operand
Negative_EOT_3	Pouze čtení	Bitový operand
Home_Switch_3	Pouze čtení	Bitový operand
Digital_Output1_3 *	Pouze zápis	Bitový operand
Digital_Output3_3 *	Pouze zápis	Bitový operand
Analog_Input1_3	Pouze čtení	16 bitů se znaménkem
Analog_Input2_3	Pouze čtení	16 bitů se znaménkem
Řídicí proměnné		
Reset_Strobe1_3	Pouze zápis	Bitový operand
Reset_Strobe2_3	Pouze zápis	Bitový operand
Stavové proměnné		
Error_Code_3	Pouze čtení	16 bitů bez znaménka
Actual_Position_3	Pouze čtení	32 bitů
Strobe1_Position_3	Pouze čtení	32 bitů
Strobe2_Position_3	Pouze čtení	32 bitů
Actual_Velocity_3	Pouze čtení	32 bitů
Axis_OK_3	Pouze čtení	Bitový operand
Position_Valid_3	Pouze čtení	Bitový operand
Strobe1_Flag_3	Pouze čtení	Bitový operand
Strobe2_Flag_3	Pouze čtení	Bitový operand

***Poznámka:** Aby mohla lokální logika do těchto digitálních výstupů zapisovat, musí v hardwarové konfiguraci VersaPro být nakonfigurované na řízení lokální logikou.

Poznámka

V případě osy 3 DSM314 verze 2.0 podporuje pouze proměnné v tabulce 13-3.

Tab. 13-4. Proměnné osy 4

Název proměnné lokální logiky	Atribut	Velikost
I/O na čelní desce		
Strobe1_Level_4	Pouze čtení	Bitový operand
Strobe2_Level_4	Pouze čtení	Bitový operand
Positive_EOT_4	Pouze čtení	Bitový operand
Negative_EOT_4	Pouze čtení	Bitový operand
Home_Switch_4	Pouze čtení	Bitový operand
Digital_Output1_4 *	Pouze zápis	Bitový operand
Digital_Output3_4 *	Pouze zápis	Bitový operand
Analog_Input1_4	Pouze čtení	16 bitů se znaménkem
Analog_Input2_4	Pouze čtení	16 bitů se znaménkem

***Poznámka**: Aby mohla lokální logika do těchto digitálních výstupů zapisovat, musí v hardwarové konfiguraci VersaPro být nakonfigurované na řízení lokální logikou.

Poznámka

V případě osy 4 DSM314 verze 2.0 podporuje pouze proměnné v tabulce 13-4.

Tab. 13-5. Globální proměnné

Název proměnné lokální logiky	Atribut	Velikost
Přetečení (1)	Čtení / zápis	Bitový operand
System_Halt (1)	Pouze zápis	Bitový operand
Data_Table_Ptr (2)	Čtení / zápis	13 bitů
Data_Table_sint (2)	Čtení / zápis	8 bitů se znaménkem
Data_Table_usint (2)	Čtení / zápis	8 bitů bez znaménka
Data_Table_int (2)	Čtení / zápis	16 bitů se znaménkem
Data_Table_uint (2)	Čtení / zápis	16 bitů bez znaménka
Data_Table_dint (2)	Čtení / zápis	32 bitů
Module_Error_Present	Pouze čtení	Bitový operand
New_Configuration_Received	Pouze čtení	Bitový operand
First_Local_Logic_Sweep (1)	Pouze čtení	Bitový operand
Module_Status_Code	Pouze čtení	16 bitů bez znaménka
CTL_1_to_32 (3)	Pouze čtení	32 bitů
P000-P255	Čtení / zápis	32 bitů
D00-D07 (4)	Čtení / zápis	64 bitů

Poznámky:

(1)- Viz odstavec “Systémové proměnné lokální logiky”.

(2)- Viz odstavec “Tabulka uživatelských dat lokální logiky ”.

(3)- Proměnnou *CTL_1_to_32* je možno použít k načtení všech 32 CTL bitů do registru.

(4)- Viz odstavec ”64-bitové registry s dvojnásobnou přesností”.

Tab. 13-6. Bity CTL

Název proměnné lokální logiky	Atribut	Velikost
CTL01 **	Čtení / zápis	Bitový operand
CTL02 **	Čtení / zápis	Bitový operand
CTL03 **	Čtení / zápis	Bitový operand
CTL04 **	Čtení / zápis	Bitový operand
CTL05 **	Čtení / zápis	Bitový operand
CTL06 **	Čtení / zápis	Bitový operand
CTL07 **	Čtení / zápis	Bitový operand
CTL08 **	Čtení / zápis	Bitový operand
CTL09 **	Čtení / zápis	Bitový operand
CTL10 **	Čtení / zápis	Bitový operand
CTL11 **	Čtení / zápis	Bitový operand
CTL12 **	Čtení / zápis	Bitový operand
CTL13 **	Čtení / zápis	Bitový operand
CTL14 **	Čtení / zápis	Bitový operand
CTL15 **	Čtení / zápis	Bitový operand
CTL16 **	Čtení / zápis	Bitový operand
CTL17 **	Čtení / zápis	Bitový operand
CTL18 **	Čtení / zápis	Bitový operand
CTL19 **	Čtení / zápis	Bitový operand
CTL20 **	Čtení / zápis	Bitový operand
CTL21 **	Čtení / zápis	Bitový operand
CTL22 **	Čtení / zápis	Bitový operand
CTL23 **	Čtení / zápis	Bitový operand
CTL24 **	Čtení / zápis	Bitový operand
CTL25	Čtení / zápis	Bitový operand
CTL26	Čtení / zápis	Bitový operand
CTL27	Čtení / zápis	Bitový operand
CTL28	Čtení / zápis	Bitový operand
CTL29	Čtení / zápis	Bitový operand
CTL30	Čtení / zápis	Bitový operand
CTL31	Čtení / zápis	Bitový operand
CTL32	Čtení / zápis	Bitový operand

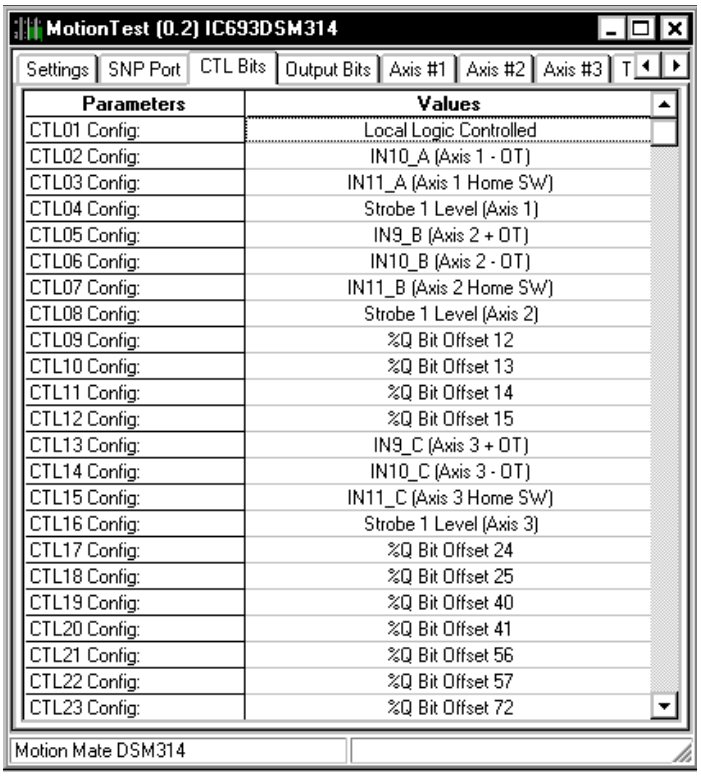
****Poznámka**: Bity CTL 1 až 24 je možno samostatně nakonfigurovat v hardwarové konfiguraci VersaPro (viz kapitola 14 “Konfigurace lokální logiky”). Lokální logika může do CTL01-24 zapisovat, pouze pokud budou v hardwarové konfiguraci nakonfigurované jako “řízené lokální logikou”.

Kapitola 14

Konfigurace lokální logiky

Konfigurace bitu CTL

Programovací prostředí VersaPro umožňuje uživateli nakonfigurovat vstupní zdroj bitů CTL (CTL01-CTL24) pomocí obrazovky hardwarové konfigurace. Na obrazovce hardwarové konfigurace zvolte modul DSM314, který chcete nakonfigurovat. Správnou posloupnost menu/ tlačítek panelu pro přístup k hardwarové konfiguraci najdete v kapitole 10. Konfigurační obrazovky DSM314 obsahují záložku s názvem CTL Bits. Po zvolení této záložky bude obrazovka vypadat přibližně jako na Obr. 14-1.



Obr. 14-1. Dialogové okno pro konfiguraci bitů CTL

Konfigurační obrazovka umožňuje uživateli zvolit konfigurační bit CTL, který odpovídá pohybovému programu a programu lokální logiky. Následující odstavce uvádějí další informace týkající se procesu konfigurace bitu CTL.

Nové bity CTL CTL01-CTL32

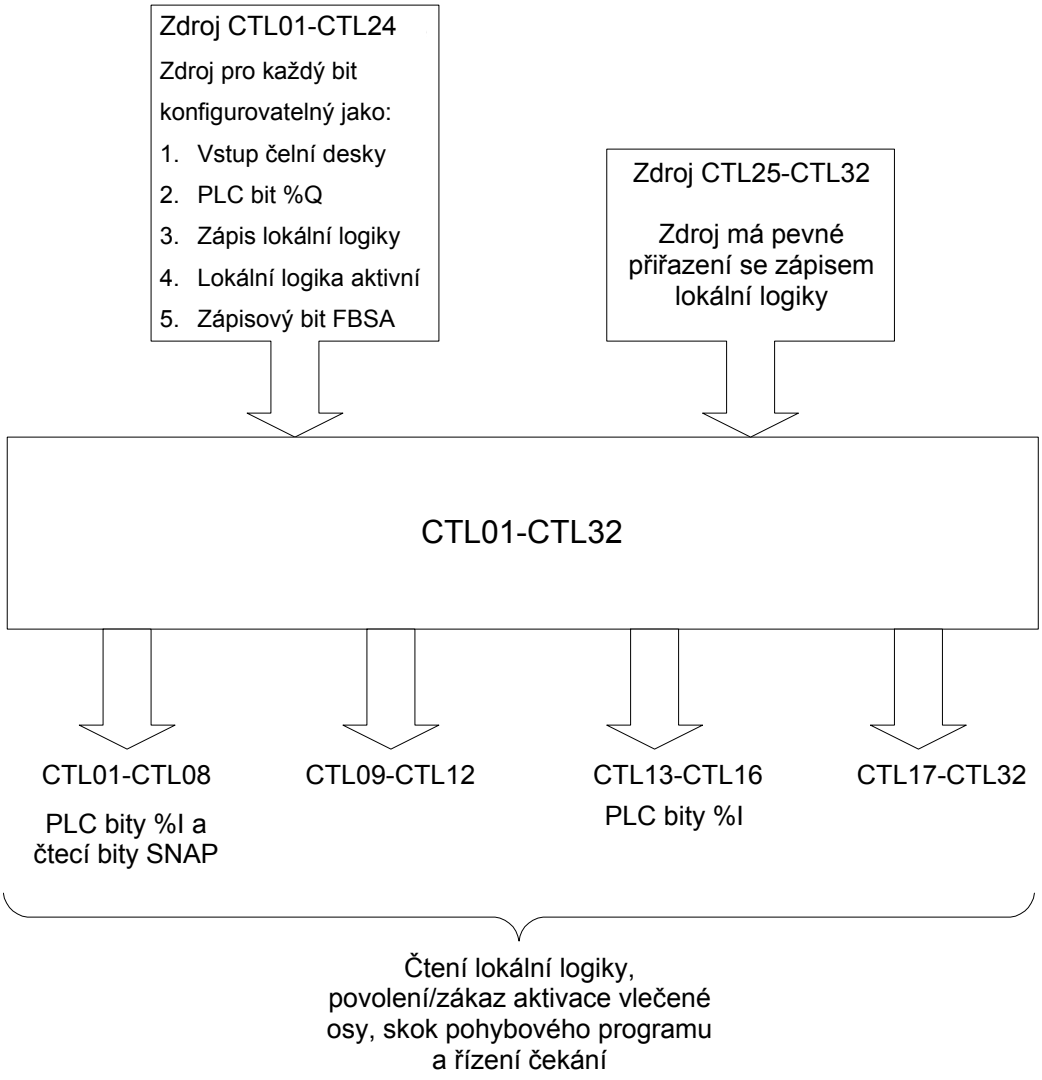
- CTL01 - CTL24 jsou konfigurovatelné bity CTL.
- CTL25-CTL32 jsou nekonfigurovatelné bity CTL umožňující čtení lokální logiky a zápis lokální logiky.

Tabulka 14-1. Přehled bitů CTL

Přehled bitů CTL01-CTL32 pro DSM314							
Označení	%I Bit	Vstupy na čelní desce	%Q bit	Čtení lokální logiky	Zápis lokální logiky	FBSA[1] Zápis	FBSA[1] Čtení
CTL01-CTL08	X	Konfigurovatelné	Konfigurovatelné	X	Konfigurovatelné	Konfigurovatelné	X
CTL09-CTL12		Konfigurovatelné	Konfigurovatelné	X	Konfigurovatelné	Konfigurovatelné	
CTL13-CTL16	X	Konfigurovatelné	Konfigurovatelné	X	Konfigurovatelné	Konfigurovatelné	
CTL17-CTL24		Konfigurovatelné	Konfigurovatelné	X	Konfigurovatelné	Konfigurovatelné	
CTL25-CTL32				X	X		

[1]FBSA znamená Fast Backplane Status Access (Přístup ke stavu rychlé vnitřní sběrnice). Podrobnosti viz GFK-0467L (nebo pozdější verze).

Následující obrázek znázorňuje zdroje, které zapisují do bitů CTL, a cílová místa, která čtou bity CTL:



Obr. 14-2. Zdroj/cílové místo bitu CTL

Volba konfigurace bitů CTL01-CTL24

Každý z bitů CTL01-CTL24 je možno nakonfigurovat nezávisle. CTL17-CTL22 jsou implicitně řídící bity digitálního výstupu %Q pro osu 1 – osu 3. CTL23-CTL24 jsou implicitně zápisové bity Fast Backplane Status Access (FBSA). Volby konfigurací jsou uvedené v následující tabulce.

Tabulka 14-2. Volba konfigurace bitů CTL

Bity CTL	Přípustné konfigurační hodnoty bitového zdroje	Popis
CTL01-CTL24	IN9_A	Přejetí (+) osy 1
	IN10_A	Přejetí (-) osy
	IN11_A	Spínač výchozí polohy osy 1
	IN9_B	Přejetí (+) osy 2
	IN10_B	Přejetí (-) osy
	IN11_B	Spínač výchozí polohy osy 2
	IN9_C	24 V vstup osy 3 na čelní desce
	IN10_C	24 V vstup osy 3 na čelní desce
	IN11_C	Spínač výchozí polohy osy 3
	IN9_D	24 V vstup osy 4 na čelní desce
	IN10_D	24 V vstup osy 4 na čelní desce
	IN11_D	24 V vstup osy 4 na čelní desce
	Úroveň vzorkování 1 osy 1	Vstupní úroveň vzorkování 1 osy 1
	Úroveň vzorkování 2 osy 1	Vstupní úroveň vzorkování 2 osy 1
	Úroveň vzorkování 1 osy 2	Vstupní úroveň vzorkování 1 osy 2
	Úroveň vzorkování 2 osy 2	Vstupní úroveň vzorkování 2 osy 2
	Úroveň vzorkování 1 osy 3	Vstupní úroveň vzorkování 1 osy 3
	Úroveň vzorkování 2 osy 3	Vstupní úroveň vzorkování 2 osy 3
	IN5_D	5 V vstup osy 4 na čelní desce
	IN6_D	5 V vstup osy 4 na čelní desce
	Zápis lokální logiky	Bit CTL řízený lokální logikou
	Příznak aktivní lokální logiky	Program lokální logiky aktivní
	FBSA zápisový 1	Sériový Non-Acknowledge protokol (FBSA) Bit 1
	FBSA zápisový 2	Sériový Non-Acknowledge protokol (FBSA) Bit 2
	FBSA zápisový 3	Sériový Non-Acknowledge protokol (FBSA) Bit 3
	FBSA zápisový 4	Sériový Non-Acknowledge protokol (FBSA) Bit 4
	Bit %Q Offset 12	CTL09 řízení programu
	Bit %Q Offset 13	CTL10 řízení programu
	Bit %Q Offset 14	CTL11 řízení programu
	Bit %Q Offset 15	CTL12 řízení programu
	Bit %Q Offset 24	24 V výstup řízení osy 1 na čelní desce (OUT1_A)
	Bit %Q Offset 25	5 V výstup řízení osy 1 na čelní desce (OUT3_A)
	Bit %Q Offset 40	24 V výstup řízení osy 2 na čelní desce (OUT1_B)
	Bit %Q Offset 41	5 V výstup řízení osy 2 na čelní desce (OUT3_B)
	Bit %Q Offset 56	24 V výstup řízení osy 3 na čelní desce (OUT1_C)
	Bit %Q Offset 57	5 V výstup řízení osy 3 na čelní desce (OUT3_C)

FBSA funkce a přiřazení bitu CTL

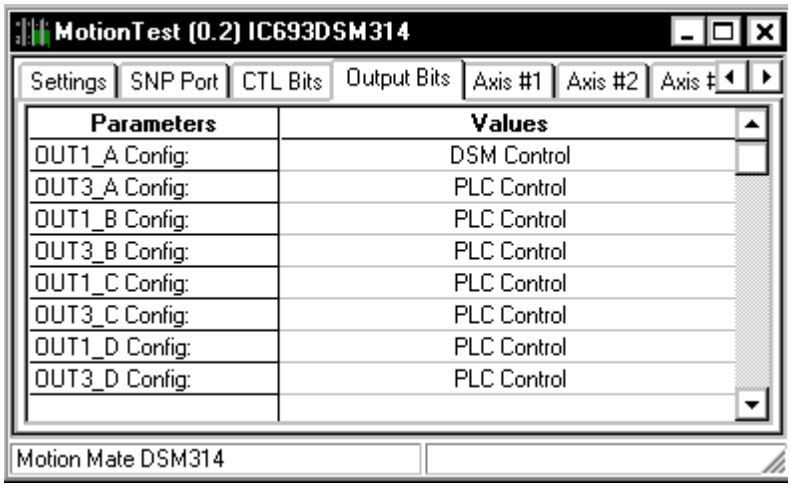
Funkce propojovací roviny Fast Backplane Status Access (FBSA) provede zápis 4 bitů do DSM a čtení 8 bitů. Funkce FBSA má rozložení podle následující tabulky. Informace FBSA service request najdete v GFK-0467L (nebo pozdější verze), *Referenční příručka instrukční sady CPU PLC Series 90-30/20/Micro*.

Tabulka 14-3. Přiřazení FBSA bitů CTL

Přiřazení FBSA bitů		
FBSA Čtení	CTL01-CTL08	CTL01-CTL08 mají nezávislý konfigurovatelný zdroj, který zahrnuje lokální logiku nebo libovolný vstup čelní desky DSM. Bity je možno vždy číst jako PLC bity %I a FBSA vstupy.
FBSA Zápis	CTL01-CTL24 (konfigurovatelné)	FBSA zápisové bity 1-4 je možno nakonfigurovat jako zdroj pro kterýkoliv bit CTL01-CTL24. FBSA zápisové bity 1-2 jsou implicitní zdroje pro CTL23-24.

Konfigurace výstupních bitů čelní desky

Programovací prostředí VersaPro umožňuje uživateli pomocí hardwarové konfigurace nakonfigurovat digitální výstupy čelní desky DSM314 buď že jsou řízené programem lokální logiky nebo programem PLC. Přístup na konfigurační obrazovku výstupních bitů získáte vyvoláním hardwarové konfigurace z VersaPro. Jakmile budete v hardwarové konfiguraci, zvolte modul DSM314, který chcete nakonfigurovat. Správnou posloupnost menu/tlačítek panelu pro přístup k hardwarové konfiguraci najdete v GFK-1670 nebo on-line nápovědě VersaPro 1.1. Konfigurační obrazovky DSM314 obsahují záložku (Output Bits). Po zvolení této záložky bude obrazovka vypadat přibližně jako na Obr. 14-3.



Obr. 14-3. Konfigurace výstupních bitů

Následující tabulka uvádí výstupy čelní desky, které je možno řídit z lokální logiky nebo PLC.

Tabulka 14-4. Popis výstupních bitů čelní desky

Název signálu	Popis
OUT1_A	24 V (SSR) výstup osy 1 na čelní desce
OUT3_A	5 V výstup osy 1 na čelní desce
OUT1_B	24 V (SSR) výstup osy 2 na čelní desce
OUT3_B	5 V výstup osy 2 na čelní desce
OUT1_C	24 V (SSR) výstup osy 3 na čelní desce
OUT3_C	5 V výstup osy 3 na čelní desce
OUT1_D	24 V (SSR) výstup osy 4 na čelní desce
OUT3_D	5 V výstup osy 4 na čelní desce

Kapitola 15

Používání VersaPro s DSM314

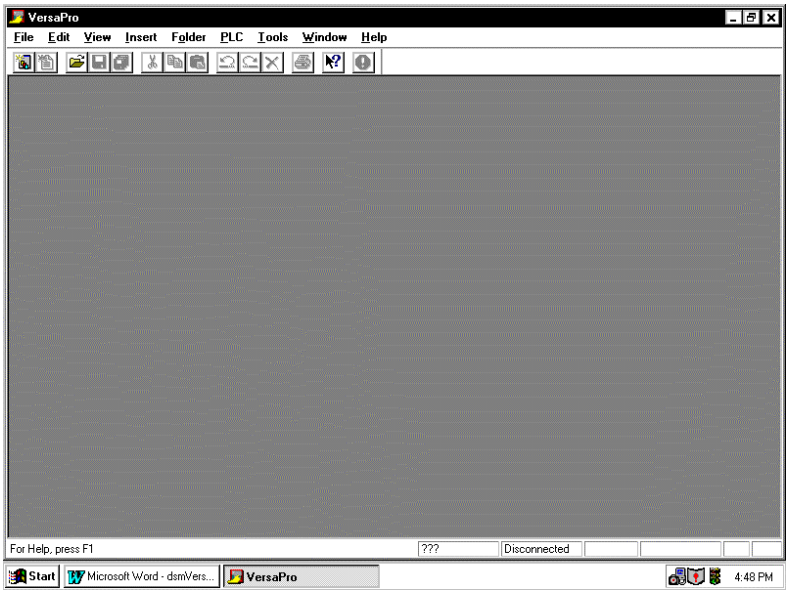
Konfigurace popisované v této kapitole se týkají konkrétně programovacího softwaru VersaPro. Uživatelé CIMPLICITY Machine Edition najdou podrobné instrukce k nakonfigurování kontroléru DSM314 v on-line nápovědě.

Krátký úvod

Poznámka: Pro použití s DSM314 se vyžaduje VersaPro verze 1.1 nebo pozdější. Tento dokument popisuje, jak používat software VersaPro pro přístup ke konfiguraci DSM314, programování pohybu a programovacím obrazovkám lokální logiky. Neuvádí konkrétně, jaké hodnoty se mají nakonfigurovat nebo jaké povely se mají použít v programech pohybu nebo programech lokální logiky. Tyto informace jsou uvedené jinde v tomto manuálu. Další informace k VersaPro je možno nalézt v *Uživatelské příručce k programovacímu softwaru VersaPro*, GFK-1670, a v on-line nápovědě VersaPro.

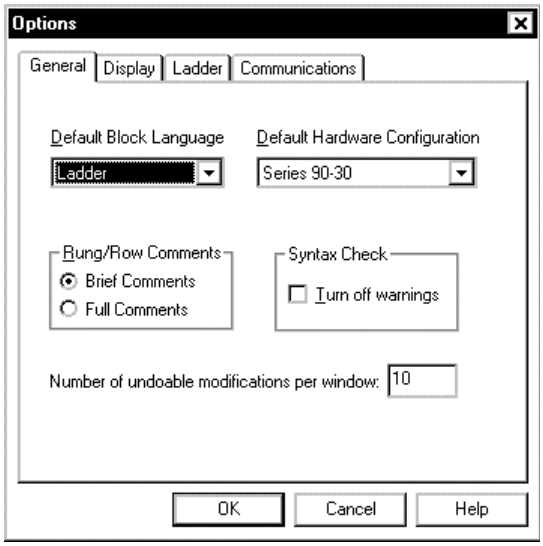
Spuštění VersaPro

Software se spustí dvojnásobným kliknutím na ikonu VersaPro na pracovní ploše Windows. VersaPro se spustí s prázdnou obrazovkou nazývanou “Pracovní plocha”.



Obrázek 15-1. “Pracovní plocha” obrazovky VersaPro po spuštění

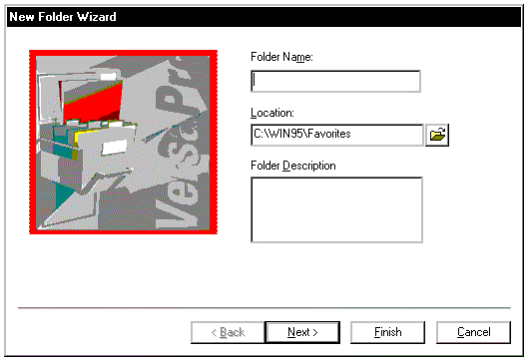
- Než budete vytvářet soubor, zkontrolujte výchozí nastavení Pracovní plochy, abyste se přesvědčili, že jako výchozí PLC je nastaveno Series 90-30. K tomu účelu klikněte na Tools na panelu menu (viz obrázek 15-4) a pak klikněte na volbu Options. Zobrazí se dialogové okno Options zobrazené níže:



Obrázek 15-2. Kontrola výchozího nastavení VersaPro v dialogovém okně Tools/Options

- Přesvědčte se, že v poli Default Hardware Configuration se zobrazuje Series 90-30, pak klikněte na tlačítko OK.

- Adresář otevřete kliknutím na File na panelu menu, potom kliknutím na Open Folder buď otevřete existující adresář nebo klikněte na New Folder a vytvořte adresář nový. Také můžete naimportovat existující adresář Series 90-30, který byl původně vytvořený v Logicmasteru nebo Control. Podrobnosti viz odstavec "Práce s adresářem" v kapitole 2 v uživatelské příručce VersaPro. V tomto příkladu klikněte na New Folder. Objeví se dialogové okno New Folder Wizard, které je ukázáno na dalším obrázku.

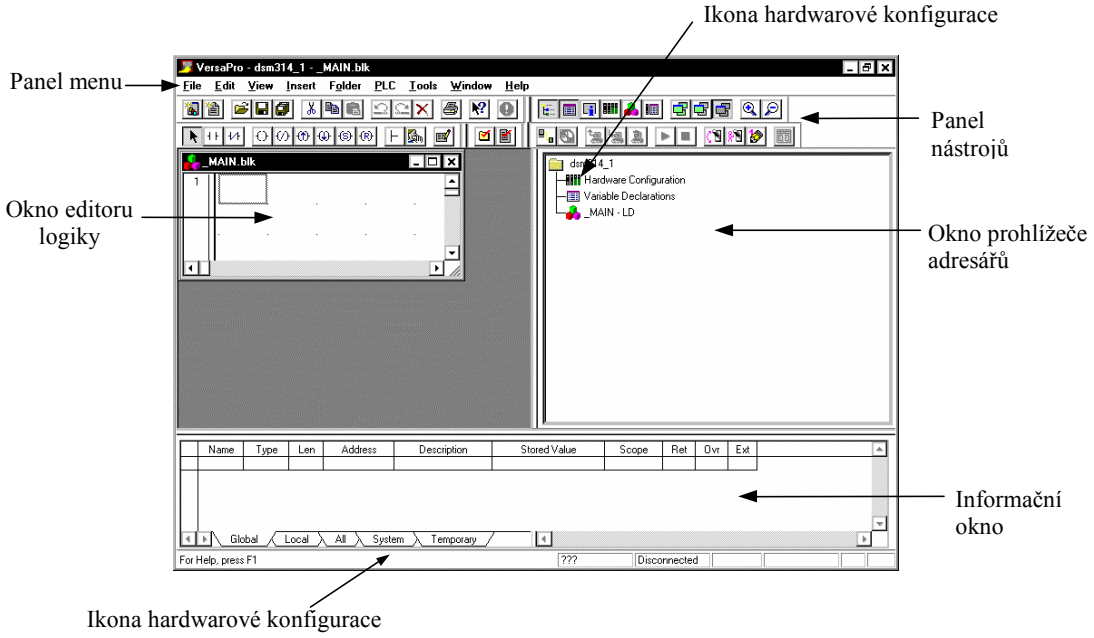


Obrázek 15-3. Dialogové okno New Folder Wizard

- Zapište název svého adresáře. I když pro název adresáře můžete použít až 255 znaků, jako název adresáře v PLC se použije pouze posledních 7 znaků. Posledních 7 znaků z Folder Name se nazývá Přezdívka adresáře. V takovém případě asi budete chtít dávat název svému adresáři uvážlivě tak, aby přezdívka měla smysl. Pokud například v poli Folder Name bude název "Pumphouse_Number_1", přezdívka uložená v PLC bude "umber_1", což asi nebude to, co byste chtěli. Lepší název může být například "PHouse1" nebo "PH1". V kapitole 2 uživatelské

příručky VersaPro (GFK-1670) je odstavec, který uvádí pravidla vytváření názvů adresářů včetně přípustných znaků.

- Pokud budete chtít adresář umístit na jiné místo, můžete změnit cestu umístění adresáře z výchozí cesty zobrazené v poli Location. Také tu je pole s popisem (Description), které umožňuje zapsat až 64 znaků popisných informací. Když skončíte se zápisem informací v tomto dialogovém okně, klikněte na tlačítko Finish. (Pokud budete chtít naimportovat adresář Logicmaster nebo Control, klikněte na Next, čímž se vyvolá další dialogové okno s volbami importu.) Nyní se zobrazí obrazovka Main LD (žebříková logika).



Obrázek 15-4. Obrazovka hlavní žebříkové logiky (LD) VersaPro

Změna uspořádání obrazovky VersaPro

Pokud se vám uspořádání obrazovky a VersaPro nebude líbit, máte možnost ho změnit. Někteří uživatelé například dávají přednost, když okno Prohlížeč adresářů je na levé straně obrazovky, nebo jiní mají radši, když okno Editoru lokální logiky je větší než je jeho výchozí velikost. Změnu uspořádání obrazovky je možno v zásadě provést dvěma základními způsoby: přemístěním a změnou velikosti.

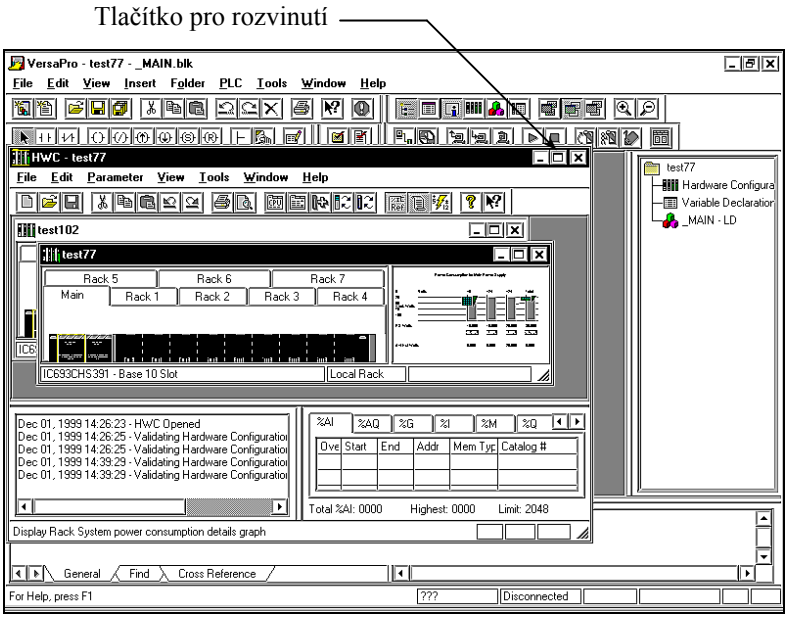
Přemístění okna: Klikněte levým tlačítkem myši **přímo uvnitř** hranice okna. Na obrazovce se objeví pravoúhlý rámeček. Držte tlačítko myši, táhněte rámečkem na požadované místo a pak tlačítko myši uvolněte v místě, kam ho chcete umístit. Někdy se při přemístění okna změní jeho velikost. Pokud budete chtít, můžete velikost změnit na jiný tvar, jak je popsáno dále.

Změna velikosti okna: Umístěte kurzor myši **na** obrys okna, u kterého chcete změnit velikost. Ukazatel myši se změní na krátkou dvojitou čáru se šipkami: ⇳

Stiskněte a držte levé tlačítko myši a táhnutím změňte velikost okna.

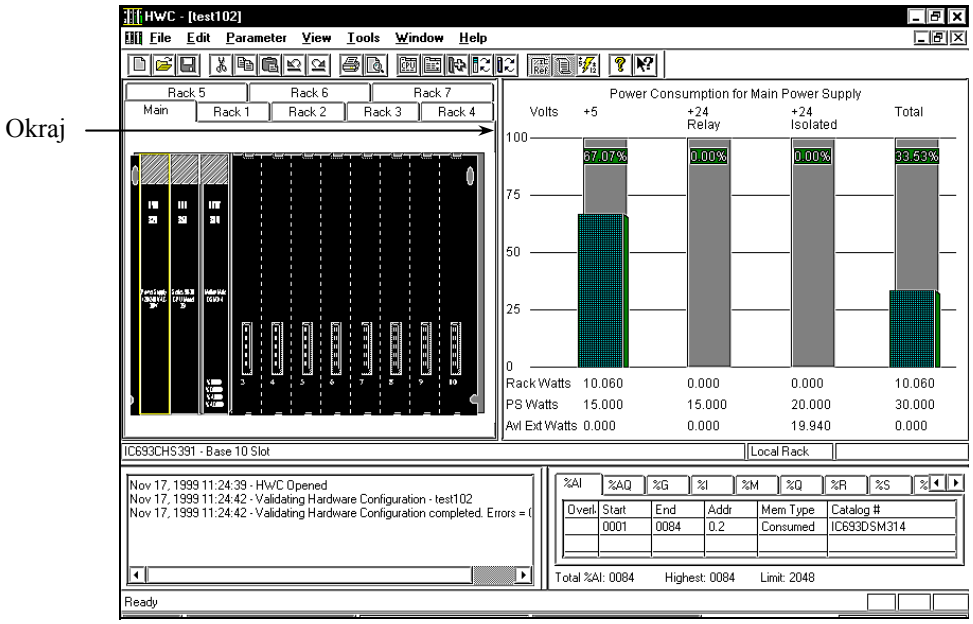
Spuštění procesu konfigurace

Konfigurátor je vlastně samostatný program, který můžete spustit z hlavní obrazovky (ukázána na předchozím obrázku). Chcete-li začít, dvojitým kliknutím na ikonu hardwarové konfigurace spustíte program HWC (Hardware Configuration). Obrazovka HWC se může objevit jako okno v horní části pracovní plochy VersaPro, jak je ukázáno níže. V tom případě kliknutím na tlačítko Expand toto okno roztáhněte na plnou velikost. (Také můžete v menším okně kliknout na tlačítko Expand a tím ho roztáhnout.)



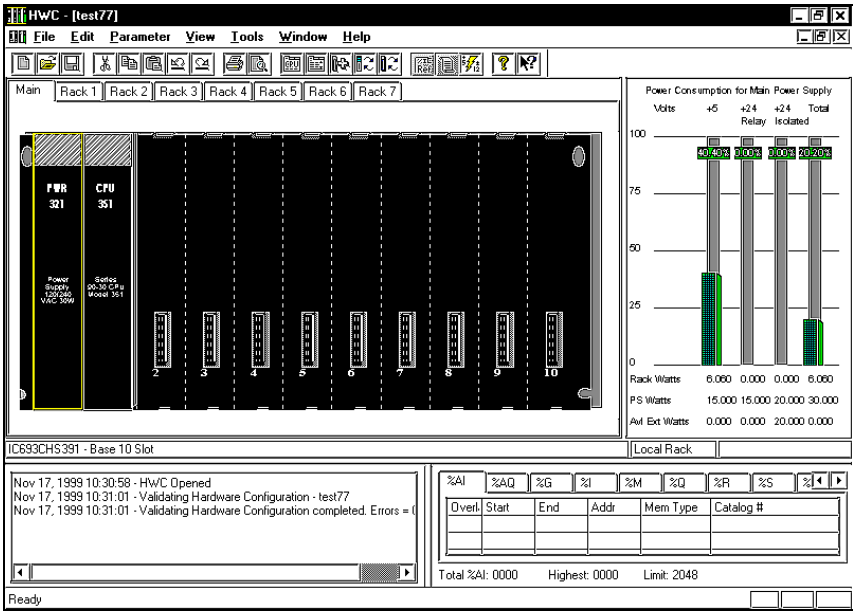
Obrázek 15-5. Úvodní obrazovka hardwarové konfigurace (HWC)

Konfigurační okno se rozvine do plné velikosti:



Obrázek 15-6. Rozvinutá obrazovka konfigurace hardwaru

- Pokud nebudete schopni jasně přečíst čísla modulů, můžete zvětšit levé okno tažením za jeho okraj (zmíněno na předchozím obrázku) doprava. Když to provedete, vaše obrazovka by měla vypadat asi jako na následujícím obrázku:



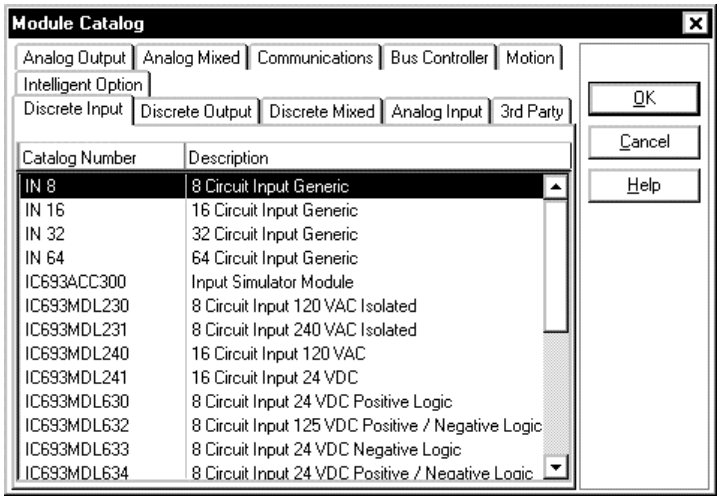
Obrázek 15-7. Zvětšená obrazovka konfigurace hardwaru

Nyní jste již připraveni začít proces konfigurace popsaný v následující části “Konfigurace DSM314”.

Konfigurace DSM314

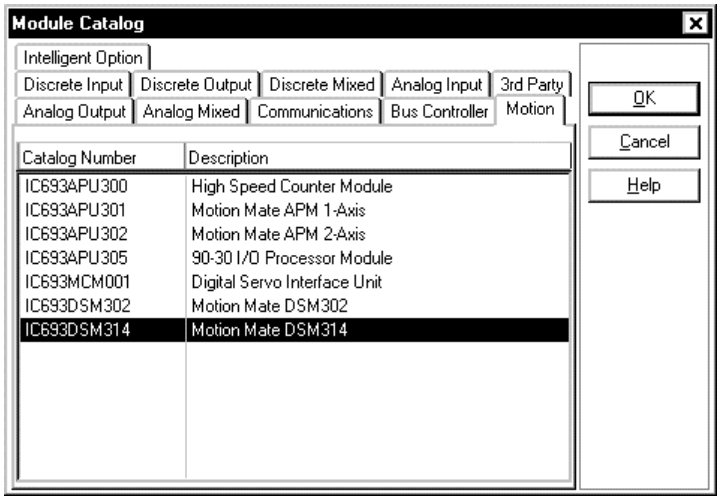
Následující informace uvádějí, jak nakonfigurovat DSM314. Budete-li chtít provést konfiguraci jiného hardwaru, postupujte prosím podle *Uživatelské příručky VersaPro*, GFK-1670, a on-line nápovědy VersaPro.

- Když bude otevřené okno konfigurace, jak je znázorněno na předchozím obrázku, klikněte dvakrát na prázdnou pozici, kde má být nainstalované DSM314. Objeví se okno Module Catalog se seznamem kategorií modulů:



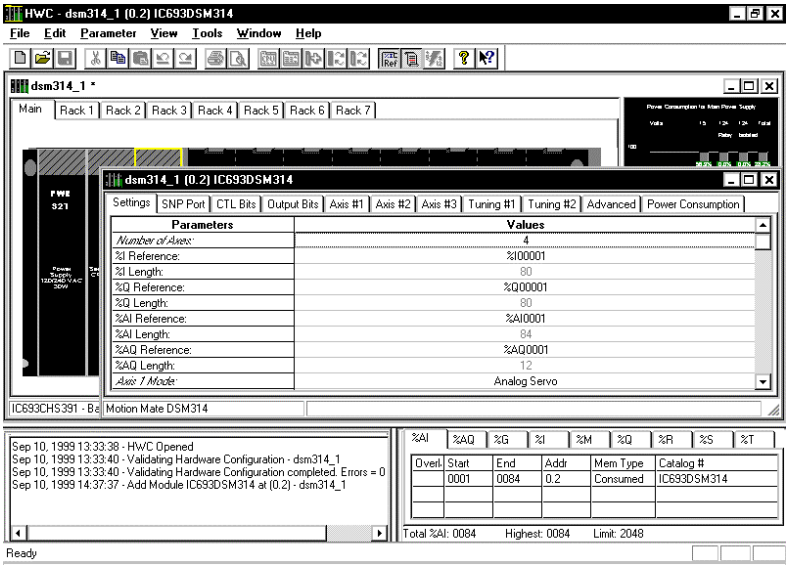
Obrázek 15-8. Okno seznamu modulů pro konfiguraci hardwaru

- Kliknutím na záložku Motion se otevře přístup k volbě pohybových modulů:



Obrázek 15-9. Záložka Motion pro konfiguraci hardwaru

- Klikněte dvakrát na IC693DSM314 nebo ho zvýrazněte, jak je znázorněno, a klikněte na tlačítko OK. Do sestavy se na obrazovce přidá DSM314 a objeví se jeho konfigurační okno.



Obrázek 15-10. Okno hardwarové konfigurace DSM314

Výše uvedený obrázek ukazuje výchozí nastavení konfigurace DSM314. Zobrazí se pouze 11 záložek pro volbu. Další záložky, které se nezobrazují, se zobrazí, pokud budou zvolené s nimi související parametry. Podrobnosti jednotlivých konfiguračních nastaveních najdete v kapitole 4. Zde je přehled záložek:

Tabulka 15-1. Volitelné záložky v okně hardwarové konfigurace DSM314

Název záložky	**Popis**
Settings	Obsahuje přiřazení a délku adres PLC, nastavení osy DSM a další globální data.
SNP Port	Nastavení pro port SNP na předním panelu DSM (označený COMM).
CTL Bits	Konfigurace 24 řídicích bitů používaných uvnitř DSM.
Output bits	Konfigurace 8 digitálních výstupů čelní desky DSM.
Axis #1	Konfigurace parametrů osy, například meze polohy, nalezení rychlosti nájezdu do výchozí polohy a zrychlení jogu.
Axis #2	
Axis #3	
Axis #4	
Tuning #1	Konfigurace údajů ladění servosmyčky, například typ motoru, časová konstanta polohové smyčky a parametrů rychlostní zpětné vazby.
Tuning #2	
Tuning #3	
Tuning #4	
Advanced	Umožňuje zápis uživatelských parametrů ladění pro každou osu
Power Consumption	Uvádí odběr DSM požadovaný z napájení propojovací roviny (4.0 W + výkon snímače polohy).

- Když budete s konfigurací modulu hotovi, klikněte na uzavírací tlačítko okna konfigurace DSM314 (tlačítko v horním pravém rohu okna s označením X) a vraťte se do "Prohlížení sestavy". Konfigurační nastavení zatím není uloženo na disk. Je pouze uloženo v energeticky závislé paměti RAM počítače.

Uložení konfiguračního nastavení na disk

- Klikněte na File na panelu menu a pak na rozbalovacím menu File klikněte na Save. Konfigurační nastavení se zapíše do příslušného souboru ve vašem programovém adresáři. Jakmile bude soubor uložený, volba Save v menu File bude neaktivní (změní se z černé barvy na šedou). Pokud pak provedete další změny konfigurace, volba Save v menu File se vrátí do aktivního stavu (a její barva se změní zpět na černou).
- Po uložení konfiguračního souboru klikněte na File na panelu menu, pak kliknutím na Exit se vraťte na obrazovku Main LD.

Připojení a uložení konfigurace do PLC

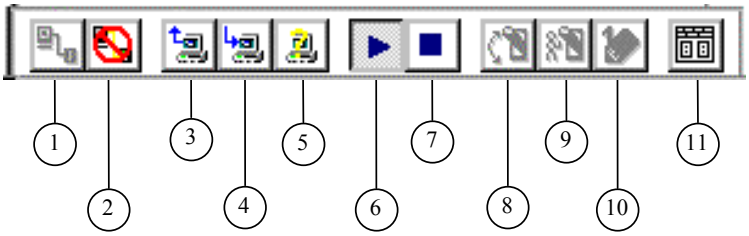
Poznámka

Konfigurační soubor nelze uložit do PLC z konfiguračního programu. Aby bylo možno provést uložení do PLC, je nutno být na VersaPro obrazovce Main LD.

Užitečné ikony na panelu nástrojů

V následujících krocích se použije několik ikon z panelu nástrojů k vyvolání operací jako Připojení, Zastavení PLC a Uložení. Následující obrázek identifikuje tyto ikony panelu nástrojů:

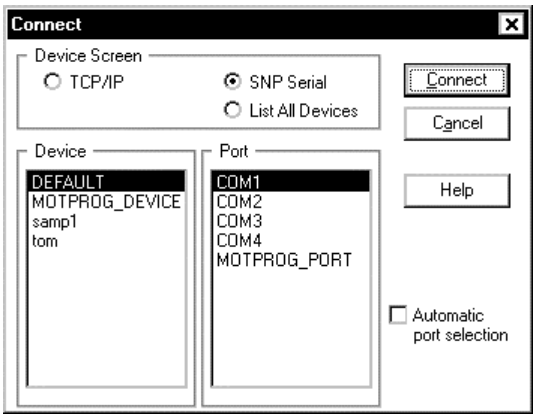
1. Připojit
2. Odpojit
3. Uložit do PLC
4. Načíst z PLC
5. Ověřit s PLC
6. Spustit PLC
7. Zastavit PLC
8. Přepnout adresu
9. Přepsat adresu
10. Zapsat hodnotu do adresy
11. Prohlížení stavu PLC



Obrázek 15-11. Ikony panelu nástrojů VersaPro

Připojení k PLC

- Na hlavní obrazovce VersaPro klikněte na ikonu Připojit na panelu nástrojů. Zobrazí se dialogové okno Connect (připojit).

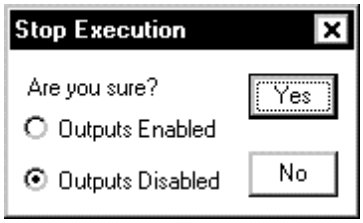


Obrázek 15-12. Dialogové okno Connect

- Pokud se budete připojovat přímo k portu programovacího zařízení PLC ze sériového portu COM1 na svém počítači, použijte nastavení DEFAULT ukázané na obrázku výše.
- Přesvědčte se, že mezi počítačem a sériovým portem na PLC je připojený sériový kabel. Pak kliknutím na tlačítko Connect v dialogovém okně Connect spusťte připojení PLC. Ve spodní části obrazovky VersaPro se zobrazí hlášení "Connecting" (připojování) s horizontálním grafem. Jakmile se připojení provede, stavové hlášení se změní z Odpojeno na Připojeno.

Zastavení PLC

- Aby bylo možno uložit konfigurační soubory, PLC musí být zastaveno, takže na panelu nástrojů klikněte na ikonu Zastavit. Zobrazí se dialogové okno Stop Execution.

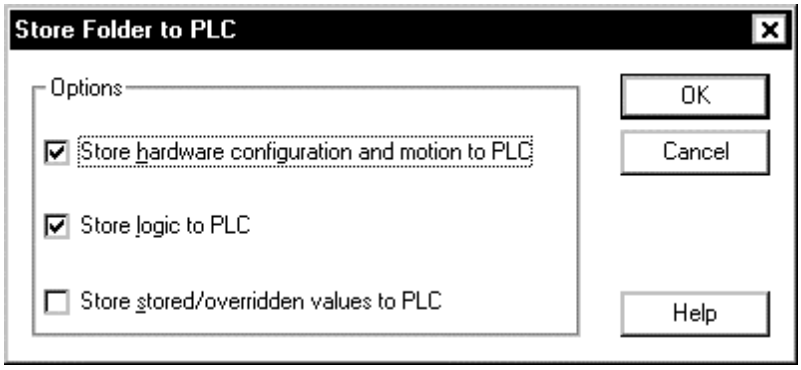


Obrázek 15-13. Dialogové okno Stop Execution

- Kliknutím na Yes PLC zastavte. Stavové hlášení ve spodní části obrazovky se změní z Run Enabled na Stop Disabled.

Operace uložení

- Klikněte na ikonu Uložit do PLC na panelu nástrojů. Zobrazí se dialogové okno Store Folder to PLC (uložení adresáře do PLC).



Obrázek 15-14. Dialogové okno uložení adresáře do PLC

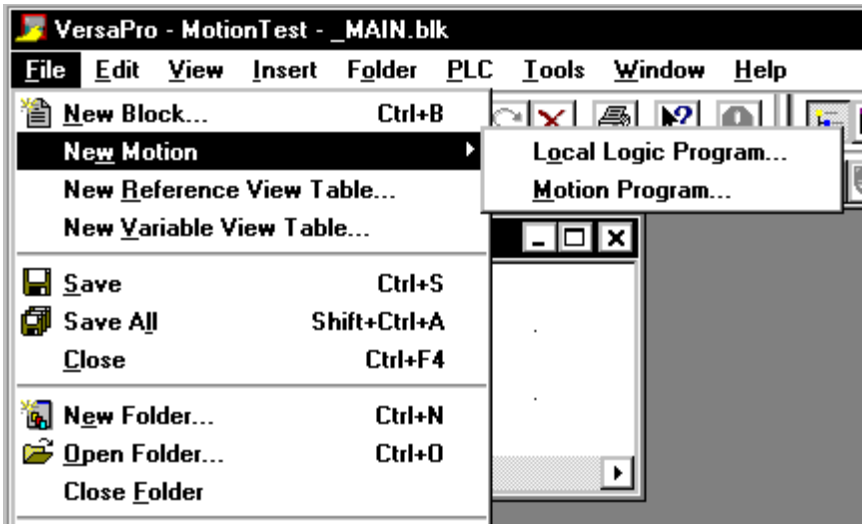
- Přesvědčte se, že je označeno pole "Store hardware configuration and motion to PLC" (Uložit hardwarovou konfiguraci a pohyb do PLC), pak kliknutím na tlačítko OK proveďte uložení do PLC. Jakmile se uložení dokončí, stavové hlášení ve spodní části obrazovky se změní z Not Equal na Equal.

Používání editoru pohybu

Otevření obrazovky editoru pohybu

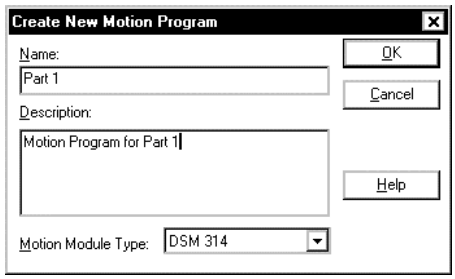
Editor pohybu i editor lokální logiky se otevírají z okna prohlížeče adresářů VersaPro. Po vytvoření a uložení se však pohybové programy a programy lokální logiky stanou součástí hardwarové konfigurace CPU PLC a uloží se do PLC s ostatními konfiguračními informacemi.

- Na obrazovce Main LD klikněte na File na panelu menu a pak zvolte New Motion. Pak na postraním menu klikněte na Motion Program (viz následující obrázek).



Obrázek 15-15. Vytvoření nového pohybového programu z menu File

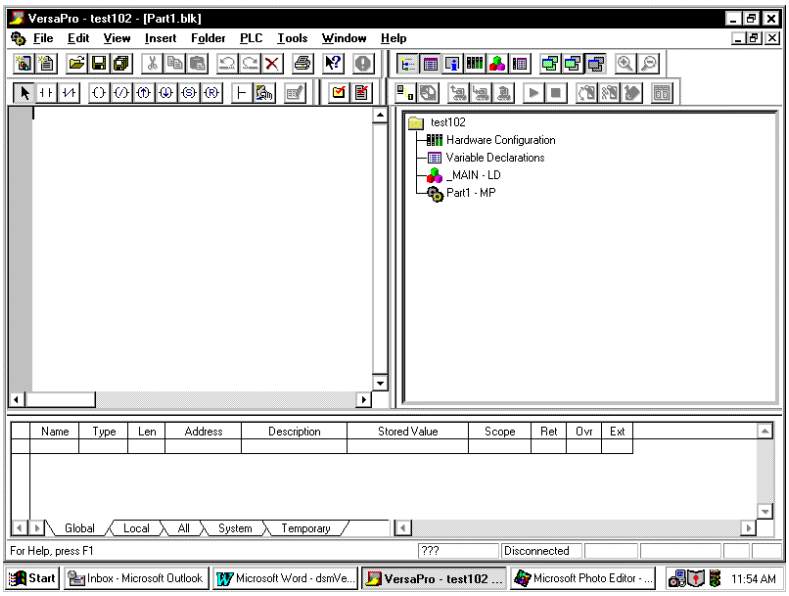
- Zobrazí se dialogové okno Create New Motion Program.



Obrázek 15-16. Dialogové okno vytvoření nového pohybového programu

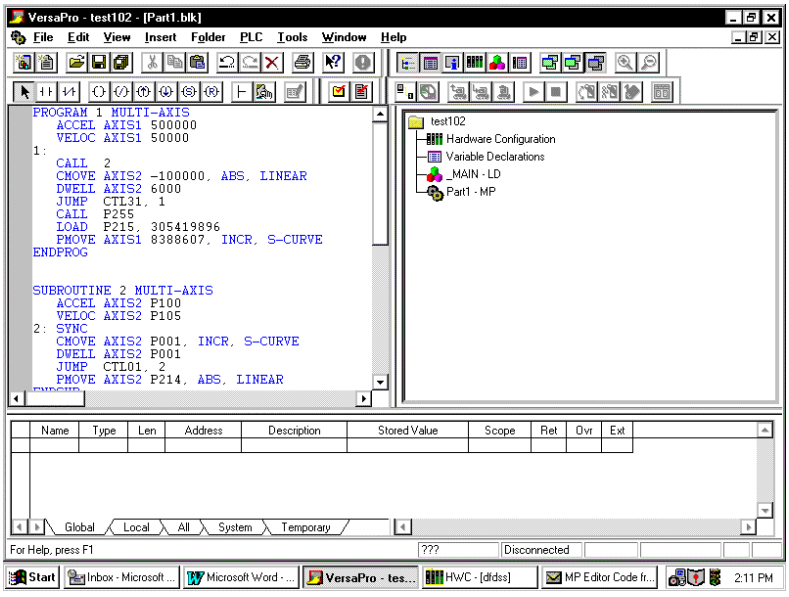
- Zapište název (Name) a popis (Description) nového programu, pak klikněte na tlačítko OK (ponechejte pole Motion Module Type box nastavené na výchozí DSM314). Otevře se okno pro blok nového pohybového programu. Jak je znázorněno na následujícím obrázku, název okna odpovídá názvu adresáře, v tomto případě Test102, a názvu pohybového programu, v

tomto případě Part1. Všimněte si také na následujícím obrázku, že v okně prohlížeče adresářů se objeví ikona pro nový pohybový program s názvem “Part1 – MP” (Motion Program).



Obrázek 15-17. Okno editace nového pohybu

- V okně editoru pohybu se vytvářejí pohybové programy a podprogramy ve tvaru textu, jak je ukázáno na následujícím obrázku. Ve stejném okně je možno naprogramovat až 10 pohybových programů a 40 podprogramů oddělených jejich identifikačními záhlavími (například “PROGRAM 1 MULTI-AXIS”) a ty se uloží do stejného souboru. Podrobnosti o povelech pohybových programů a syntaxi najdete v kapitole 7.



Obrázek 15-18. Okno editoru pohybu s naprogramovaným kódem

Uložení pohybového programu

- Když budete připraveni uložit soubor svého pohybového programu/podprogramu na pevný disk počítače, buď klikněte na ikonu Uložit na panelu nástrojů (vypadá jako disketa) nebo klikněte na File z panelu menu a pak klikněte na Save.

Uložení pohybových programů a podprogramů do PLC

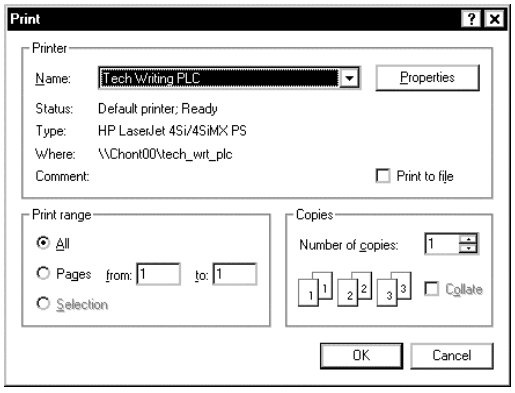
Protože soubor pohybových programů/podprogramů se pokládá za součást skupiny konfiguračního souboru, použijte postup uvedený v odstavci "Připojení a uložení konfigurace do PLC" výše.

Vytisknutí pohybových programů a podprogramů

V menu File jsou dvě volby pro tisk: Print a Print Report.

Print:

- Tato položka popisuje, jak vytisknout celý soubor pohybového programu (blok). Když bude editor pohybu aktivní, klikněte na File na panelu menu a zvolte Print. Zobrazí se dialogové okno Printer. Proveďte všechny změny nastavení tiskárny a pak klikněte na tlačítko OK.



Obrázek 15-19. Dialogové okno pro tisk

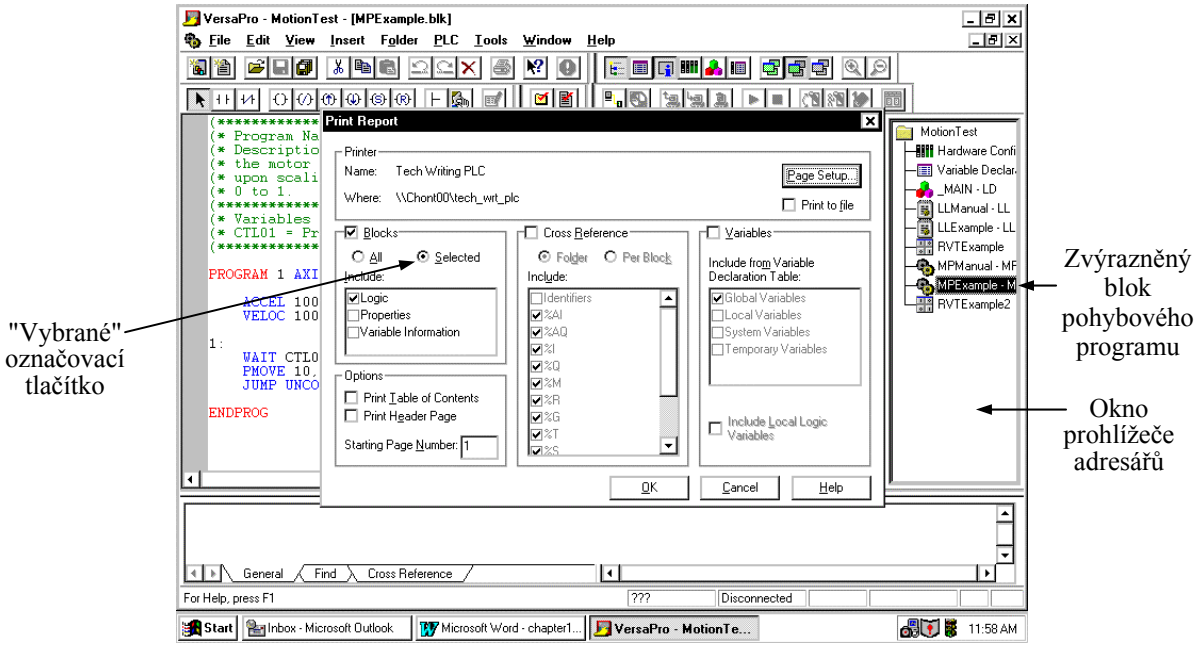
- Tato položka popisuje, jak vytisknout pouze zvolenou část souboru pohybového programu/podprogramu. V okně editoru pohybu použijte myš a zvolte část, kterou chcete vytisknout, klikněte na File na panelu menu a pak zvolte Print. V dialogovém okně Print (zobrazeno výše) se přesvědčte, že je označeno výběrové tlačítko v části Print range (rozsah tisku) (má tečku uprostřed). Klikněte na tlačítko OK.

Print Report:

- Chcete-li vytisknout všechny bloky pohybového programu (pokud jich máte víc než jeden) jako součást výkazu s dalšími informacemi v adresáři, klikněte na File na panelu menu a pak zvolte Print Report. Zobrazí se dialogové okno Print Report. V dialogovém okně Print Report klikněte na označovací pole Blocks. Přesvědčte se, že je zvoleno výběrové tlačítko All. (Pro tento výkaz také můžete zvolit další položky a funkce, například obsah, křížové

odkazy, proměnné, atd.) Kliknutím na tlačítko OK spusťte tisk. Jako součást tohoto výkazu se vytisknou bloky Pohybový program, Lokální logiky a Žebříková logika.

- Chcete-li vytisknout pouze zvolené bloky, zvýrazněte je v okně Prohlížeče adresářů. Klikněte na File na panelu menu a zvolte Print Report. Klikněte na označovací pole Blocks, pak zvolte výběrové tlačítko Selected. Tím se omezí výkazy pouze na ty, které jste zvýraznili v okně Prohlížeče adresářů. Příklad je uvedený na následujícím obrázku.

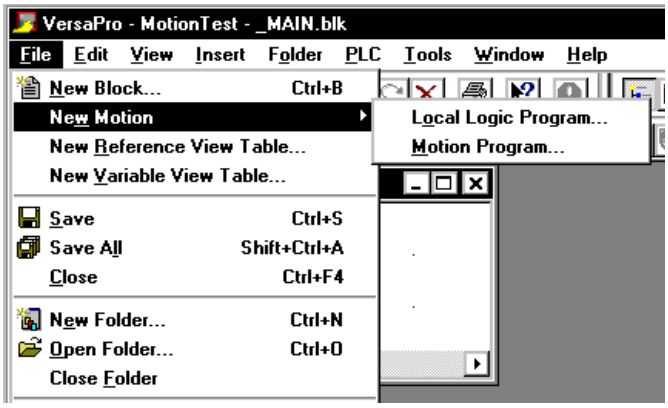


Obrázek 15-20. Dialogové okno tisku výkazu

Otevření obrazovky editoru lokální logiky

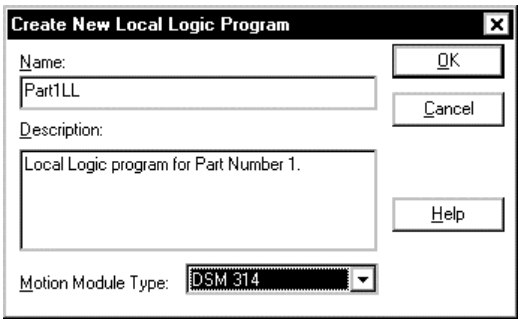
Editor pohybu i editor lokální logiky se otevírají z okna prohlížeče adresářů VersaPro. Po vytvoření a uložení se však pohybové programy a programy lokální logiky stanou součástí hardwarové konfigurace CPU PLC a uloží se do PLC s ostatními konfiguračními informacemi.

- Na obrazovce Main LD klikněte na File na panelu menu a pak zvolte New Motion. Pak na postraním menu klikněte na Local Logic Program.



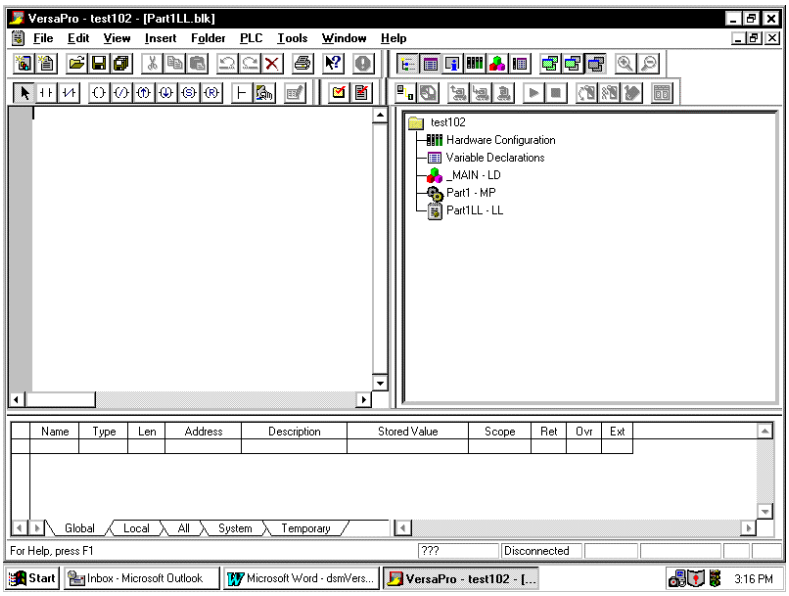
Obrázek 15-21. Vytvoření nového programu lokální logiky

Zobrazí se dialogové okno Create New Local Logic.



Obrázek 15-22. Dialogové okno vytvoření nové lokální logiky

- Zapište název (Name) a popis (Description) nové lokální logiky, pak klikněte na tlačítko OK (ponechejte pole Module Type box nastavené na výchozí DSM314). Otevře se okno pro blok nové lokální logiky. Jak je znázorněno na následujícím obrázku, název okna odpovídá jak názvu adresáře, v tomto případě Test102, tak i názvu programu lokální logiky, v tomto případě Part1LL. Program lokální logiky v tomto příkladu nelze nazvat "Part1", protože tento název se již použil jako název bloku pohybového programu. Všimněte si také na následujícím obrázku, že v okně prohlížeče adresářů se po pravé straně obrazovky objeví ikona pro nový pohybový program s názvem “Part1LL – LL” (Lokální logika).



Obrázek 15-23. Okno nového programu lokální logiky

Program lokální logiky na bázi textu se vytvoří v levé části okna (okno editoru lokální logiky). Podrobnosti o povelech lokální logiky a syntaxi najdete v kapitolách 10 - 14.

Uložení programu lokální logiky

Když budete připraveni uložit soubor lokální logiky na pevný disk počítače, buď klikněte na ikonu Uložit na panelu nástrojů (vypadá jako disketa) nebo klikněte na File z panelu menu a pak klikněte na Save.

Uložení programu lokální logiky do PLC

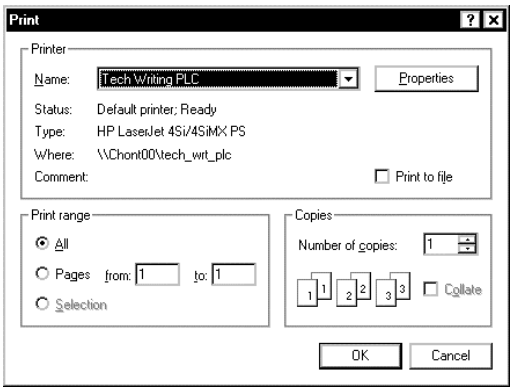
Protože soubor lokální logiky se pokládá za součást skupiny konfiguračního souboru, použijte postup uvedený v odstavci "Připojení a uložení konfigurace do PLC" výše v této kapitole.

Vytisknutí programu lokální logiky

V menu File jsou dvě volby pro tisk: Print a Print Report.

Print:

- **Vytisknutí celého souboru lokální logiky (bloku):** Když bude editor lokální logiky aktivní, klikněte na File na panelu menu a zvolte Print. Zobrazí se dialogové okno Printer. Proveďte všechny změny nastavení tiskárny a pak klikněte na tlačítko OK.



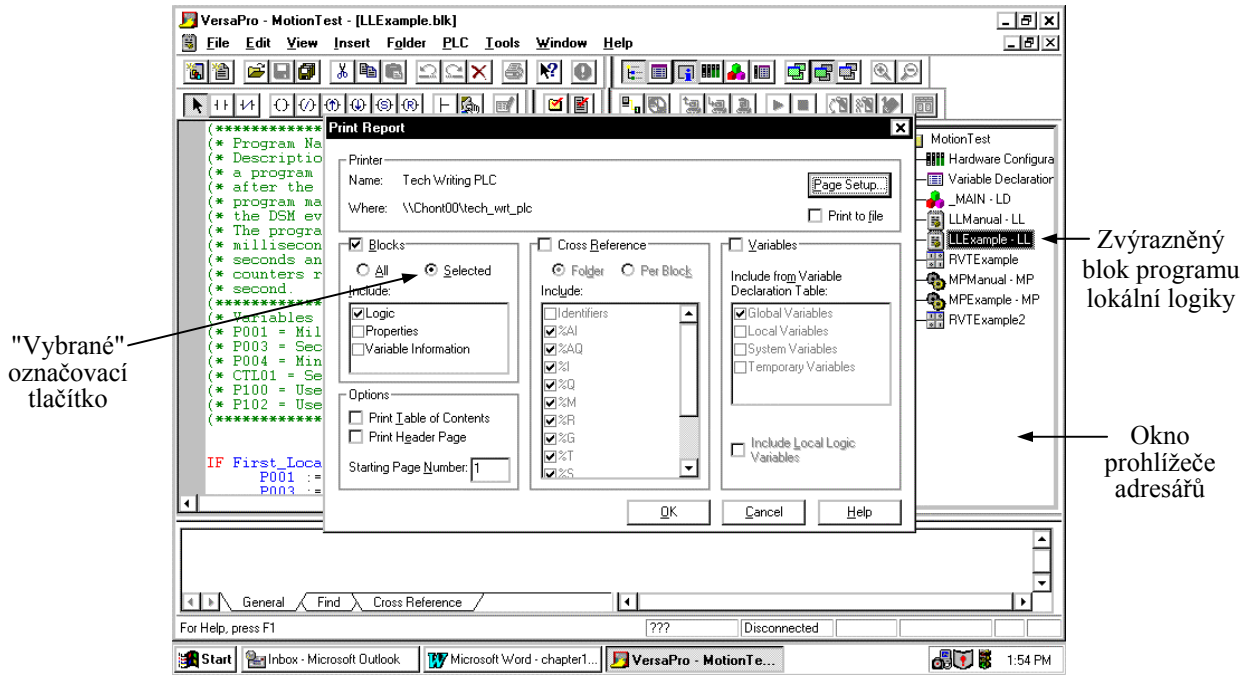
Obrázek 15-24. Dialogové okno pro tisk

- **Vytisknutí pouze vybrané části souboru lokální logiky:** V okně editoru lokální logiky použijte myš a zvolte část, kterou chcete vytisknout, klikněte na File na panelu menu a pak zvolte Print. V dialogovém okně Print (zobrazeno výše) se přesvědčte, že je označeno výběrové tlačítko v části Print range (rozsah tisku) (má tečku uprostřed). Klikněte na tlačítko OK.

Print Report:

- **Chcete-li vytisknout všechny bloky lokální logiky jako součást výkazu s** dalšími informacemi v adresáři, klikněte na File na panelu menu a pak zvolte Print Report. Zobrazí se dialogové okno Print Report. V dialogovém okně Print Report klikněte na označovací pole Blocks. Přesvědčte se, že je zvoleno výběrové tlačítko All. (Pro tento výkaz také můžete zvolit další položky a funkce, například obsah, křížové odkazy, proměnné, atd.) Kliknutím na tlačítko OK spusťte tisk. Jako součást tohoto výkazu se vytisknou bloky Pohybový program, Lokální logiky a Žebříková logika.

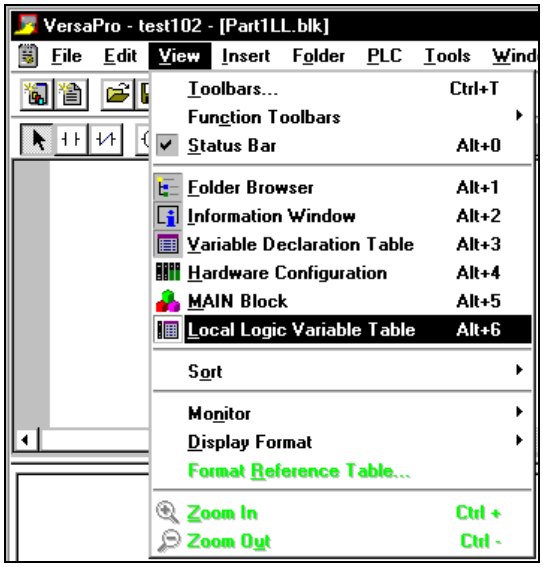
- **Chcete-li vytisknout pouze zvolené bloky lokální logiky jako součást výkazu**, zvýrazněte požadované bloky lokální logiky v okně prohlížeče adresářů. Klikněte na File na panelu menu a zvolte Print Report. Klikněte na označovací pole Blocks, pak zvolte výběrové tlačítko Selected. Tím se omezí výkazy pouze na ty, které jste zvýraznili v okně Prohlížeče adresářů. Příklad je uvedený na následujícím obrázku.



Obrázek 15-25. Dialogové okno tisku výkazu

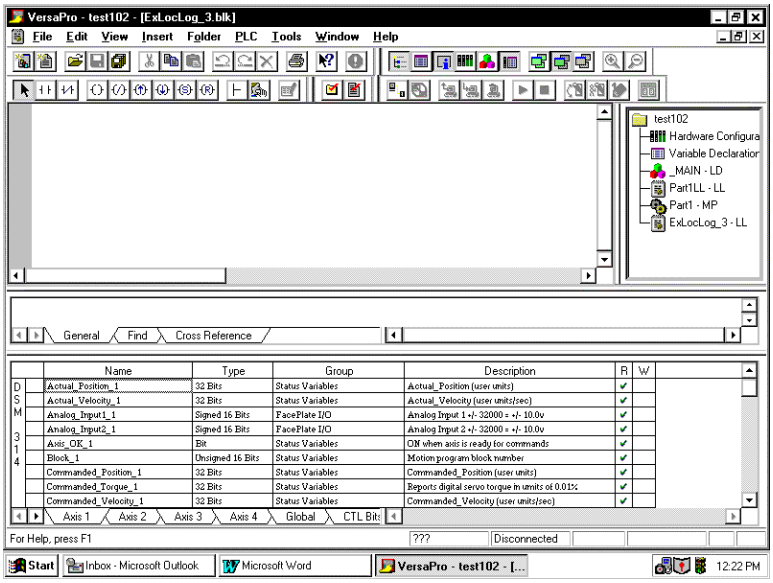
Prohlížení tabulky proměnných lokální logiky

Tabulka proměnných lokální logiky se objeví na obrazovce na ploše informačního okna a obsahuje informace o proměnných používaných v programu lokální logiky. Chcete-li použít tuto funkci, musí existovat blok lokální logiky. Pokud žádný neexistuje, vytvořte nový. Tabulku zobrazíte tak, že kliknete na View na panelu menu a z rozbalovacího menu zvolíte Local Logic Variable Table, jak je znázorněno na následujícím obrázku:



Obrázek 15-26. Volba tabulky proměnných lokální logiky z menu View

Jakmile se tabulka proměnných lokální logiky objeví ve spodní části obrazovky VersaPro, můžete táhnout za její horní okraj nebo okraje sloupce a změnit její velikost podle potřeby. Viz následující obrázek.



Obrázek 15-27. Prohlížení tabulky proměnných lokální logiky na spodním okraji obrazovky

Tato tabulka je užitečná při vytváření programu lokální logiky, protože umožňuje do programu kopírovat a vkládat názvy proměnných, například “Actual_Position_1”. Tím se eliminuje potřeba si zapamatovat přesné názvy proměnných, vyhledávat je v tomto manuálu nebo je zapisovat.

Kapitola 16

Používání funkce elektronické vačky

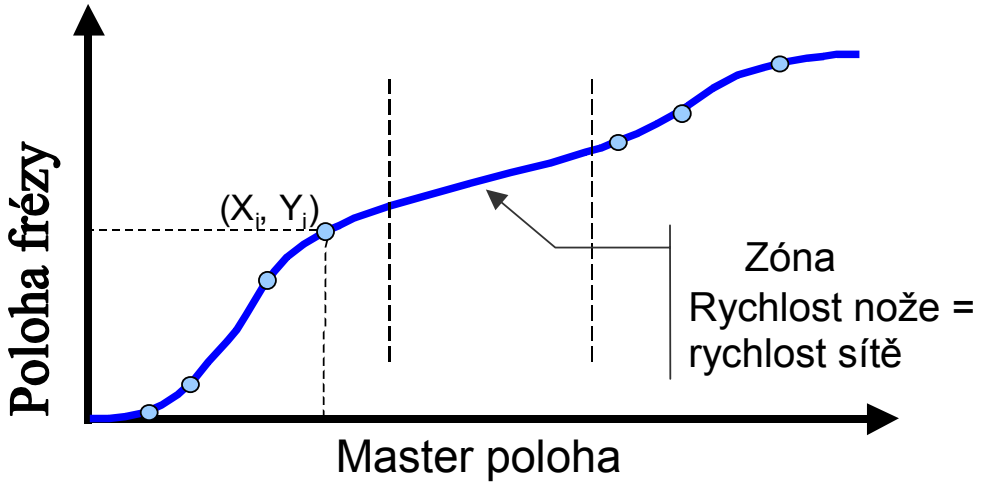
Část 1: Úvod

Tato kapitola popisuje funkci elektronické vačky (CAM) DSM314 verze 2.0. Elektronická vačka je analogická mechanické vačce. Ve většině případů elektronická vačka nejen že může nahradit tradiční mechanickou vačku, ale vykonává mnoho funkcí, které nelze dosáhnout mechanickou předlohou. Například elektronická vačka nikdy nepodléhá mechanickému opotřebení.

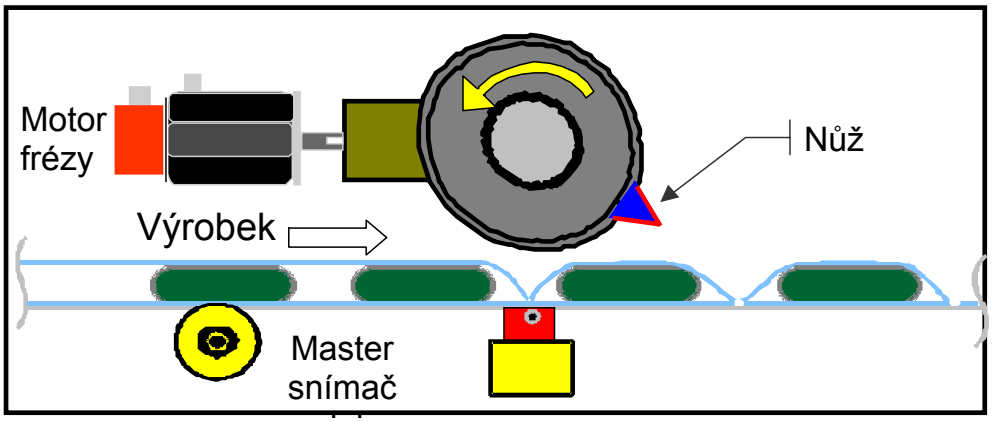
Přehled elektronické vačky

Elektronické vačky se používají v průmyslu mechaniky k vykonávání složitých pohybů, které vyžadují úzkou koordinaci mezi osami. Existuje mnoho příkladů, které spadají do těchto požadavků. Mezi některé příklady patří aplikace jednoduchého rotačního nože zobrazeného na Obr. 16-1 a

Obr. 16-2. U této aplikace poloha pásu dopravníku slouží jako master poloha, zatímco řezný nůž je slave. Protože poloha nože je svázána s master polohou, nůž bude vždy sledovat master polohu, i když linka bude zrychlovat nebo zpomalovat.

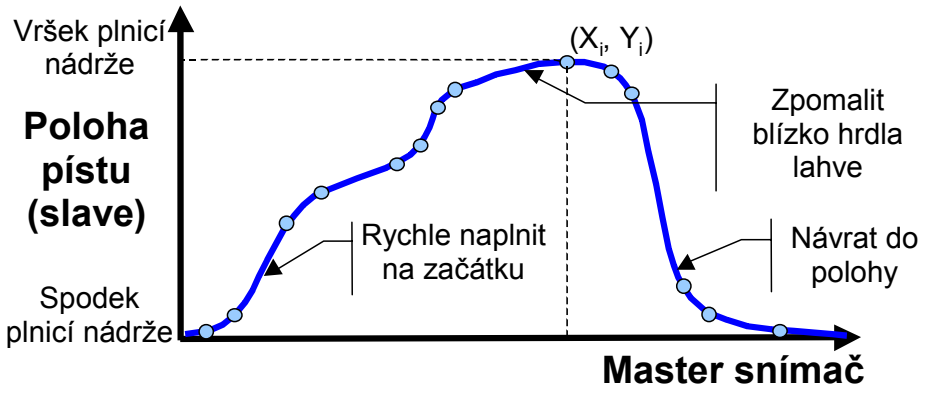


Obr. 16-1. Poloha rotačního nože k polohovacímu stolu

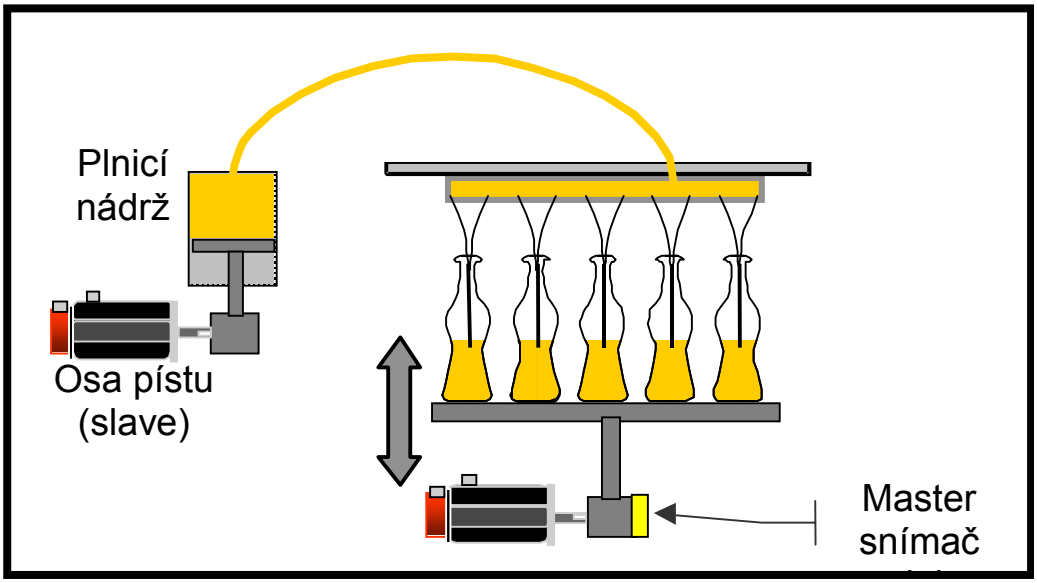


Obr. 16-2. Aplikace rotačního nože

Jiným příkladem aplikace je linka na plnění lahví (viz Obr. 16-3 a Obr. 16-4). V tomto případě výtah, který přemisťuje lahve nahoru a dolů, slouží jako CAM master. Slave je píst, který vtlačuje kapalinu do láhve. V tomto příkladu láhve mají zakřivený tvar. Proto se rychlost plnění musí měnit tak, aby se bral ohled na tento tvar.



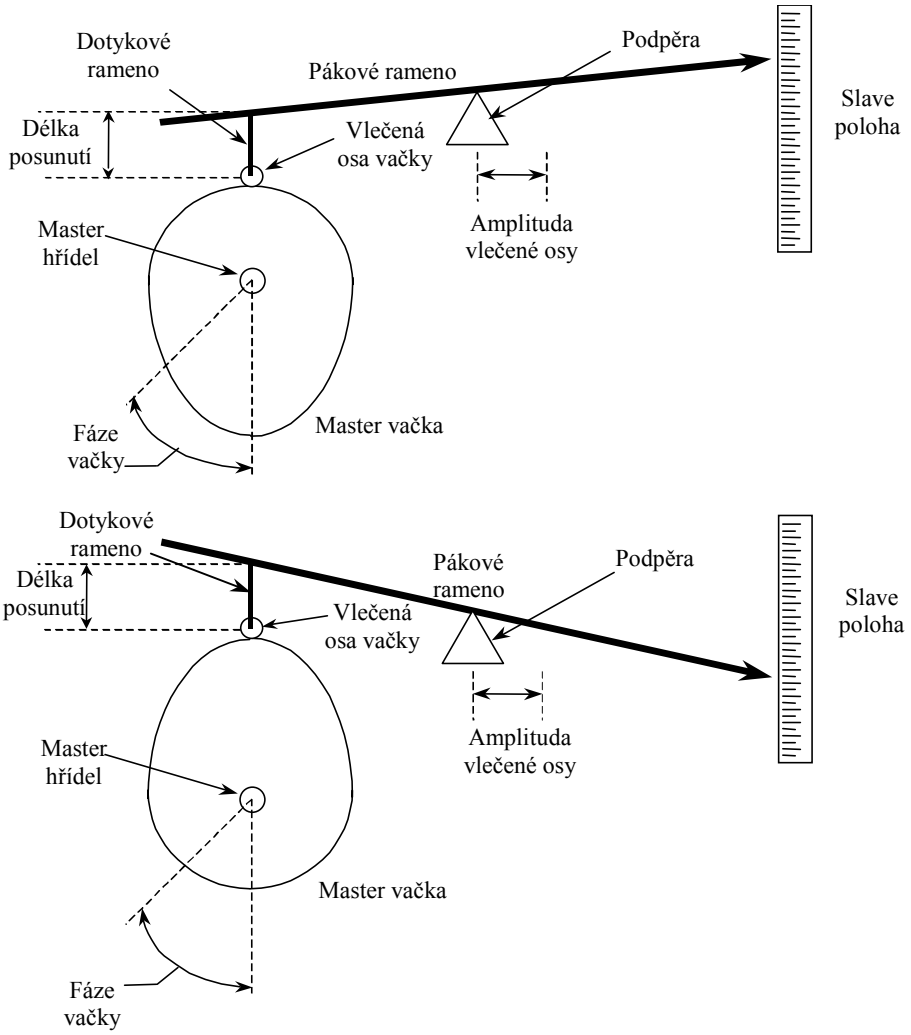
Obr. 16-3. Poloha k polohovacímu stolu v aplikaci plnění



Obr. 16-4. Aplikace plnění

Základní tvary vačky/definice

Elektronické vačky kopírují chování svých mechanických předloh. Následující obrázek znázorňuje prvky základního mechanického vačkového systému a ukazuje slave polohu pro dvě polohy master vačky. Jak se master hřídel otáčí, master vačka, která je připevněna k master hřídeli, se otáčí také. Vlečená osa vačky (která je namontovaná přes kuličkové ložisko k dotykovému ramenu) se odvaluje na master vačce tak, jak se master vačka otáčí. Vlečená osa vačky buď vytlačuje nahoru nebo tlačí dolů dotykové rameno v závislosti na poloze master vačky. Pákové rameno, které je spojeno s dotykovým ramenem, se pak pohybuje nahoru a dolů a otáčí se na podpěře podle toho, jak se pohybuje dotykové rameno. Kromě tvaru master vačky, další parametry ovlivňující slave pohyb, je hodnota fáze vačky (velikost, o kterou je poloha master vačky posunutá od polohy master hřídele), délka posunutí od dotykového ramena a amplituda vlečené osy (vycházející z polohy podpěry). Všechny tyto mechanické parametry mají protějšky elektronické vačky.



Obr. 16-5. Model vačky

Část 2: Syntaxe vačky

Tato část popisuje některé kritické vlastnosti funkce vačky a uvádí příkazy pohybového programu a chybové kódy vačky.

Typy vačky

Důležitá věc týkající se funkce vačky jsou jejich různé použitelné typy. Profily vačky mohou mít některý z následujících typů:

1) Necyklická vačka

2) Lineární cyklická vačka

3) Kruhová cyklická vačka

Následující odstavce popisují jednotlivé typy těchto vaček.

Necyklická vačka

Necyklická vačka má jednoznačný neopakující se profil pro celý rozsah hodnot master polohy. Vačka se ukončí, když se dosáhne některého okraje profilu vačky. Vačka se také může ukončit, pokud externí děj bude nakonfigurovaný na aktivaci podmíněného skoku. Poměr uživatelských jednotek k počtu pulsů zadaný pro master osu a slave osu při konfiguraci necyklické vačky musí souhlasit se zadaným poměrem uživatelských jednotek : počet pulsů pro odpovídající osy v hardwarové konfiguraci. Také maximální a minimální hodnoty polohy slave a master osy musí ležet v mezích horní/dolní polohy zadané v hardwarové konfiguraci pro odpovídající osy.

Lineární cyklická vačka

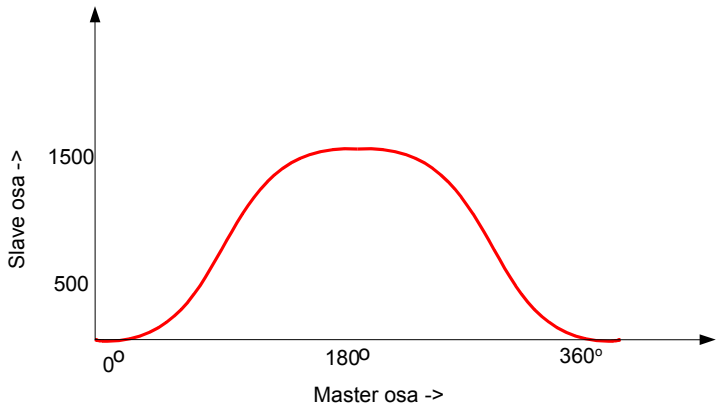
Lineární cyklická vačka má profil, který udržuje opakování, dokud ho nějaký děj neukončí. Kromě toho číselný a fyzický koncový bod vačkové slave osy je stejný jako počáteční bod cyklu. Příkladem lineární cyklické vačky je kliková hřídel pohybující se vpřed a zpět. Poměr uživatelských jednotek na počet pulsů zadaný pro master a slave osu při konfiguraci profilu vačky musí souhlasit s hodnotou poměru uživatelských jednotek : počet pulsů pro odpovídající osy v hardwarové konfiguraci. Příklad použití lineární cyklické vačky je ukázaný na obrázku 16-1 a 16-2.

Omezení: První a poslední slave bod musí být pro lineární cyklickou vačku stejný. Pokud data v tabulce vačky toto omezení splnění, editor vačky nezobrazí volbu “Linear Cyclic” v poli “Cam Type”.

Poznámky:

1. Pro každou cyklickou vačku musí být nastavené meze horní/dolní polohy v hardwarové konfiguraci podle vratných bodů masteru v profilu vačky. Dolní mez master osy se musí rovnat první master poloze v profilu. Mez horní polohy masteru se musí rovnat (poslední master poloze - 1) v uživatelských jednotkách. Je to z toho důvodu, že první a poslední bod profilu jsou na fyzickém zařízení stejné.

2. U lineární cyklické vačky se master osa vrací v koncových bodech profilu, ale slave osa ne. Maximální a minimální hodnoty profilu slave a master osy musí ležet v horní/dolní mezích zadaných v hardwarové konfiguraci pro odpovídající osu.



Obr. 16-6. Lineární cyklická vačka

Kruhová cyklická vačka

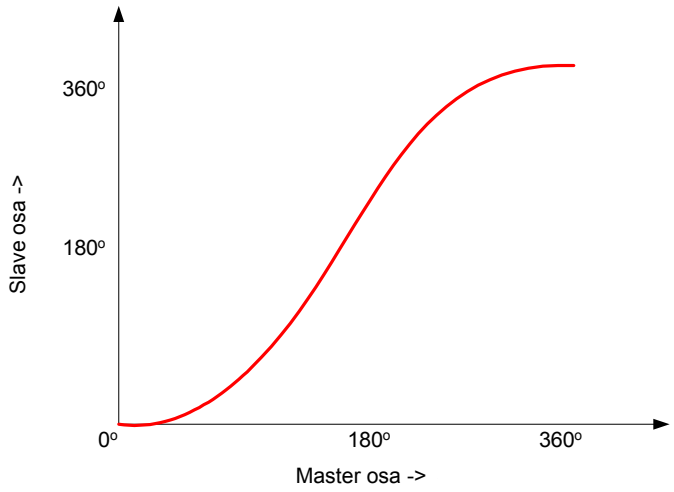
Kruhová cyklická vačka má profil, který udržuje opakování, dokud ho nějaký děj neukončí. Kromě toho kruhová cyklická vačka má odlišné číselné počáteční a koncové polohy osy (viz Obr. 16-7). Master i slave osa se vrací v koncovém bodě profilu. Příkladem kruhové cyklické vačky může být rotační nůž.

Omezení: **Celý** slave profil (včetně interpolovaných hodnot) musí ležet mezi *minimální* a *maximální* mezí polohy, kde *minimální* a *maximální* slave meze jsou definované následovně:

Minimální slave hodnota	Maximální slave hodnota	Podmínka
První slave bod	Poslední slave bod	Poslední bod > první bod
Poslední slave bod	První slave bod	Poslední bod < první bod

Poznámky

1. Pokud nebude splněno výše uvedené omezení, editor nezobrazí volbu “Circular Cyclic” v poli “CAM Type”.
2. Pro cyklické vačky musí být nastavené meze horní/dolní polohy v hardwarové konfiguraci podle vratných bodů masteru v profilu vačky. Dolní mez master osy se musí rovnat první master poloze v profilu. Mez horní polohy masteru se musí rovnat (poslední master poloze - 1) v uživatelských jednotkách. Je to z toho důvodu, že první a poslední bod profilu jsou na fyzickém zařízení stejné. (například 0° a 360° na kruhovém noži).
3. U kruhové cyklické vačky se master osa i slave osa vrací v koncových bodech profilu. Horní a dolní mez polohy pro osu slave jsou nastavené (v hardwarové konfiguraci) následovně:
   - Pokud minimální slave poloha je první bod profilu a maximální slave poloha je poslední bod profilu, nastavte mez dolní polohy na hodnotu prvního slave bodu a horní mez polohy na (hodnotu posledního bodu – 1).
   - Pokud minimální slave poloha je poslední bod profilu a maximální slave poloha je první bod profilu, nastavte mez dolní polohy na hodnotu (posledního slave bodu + 1) a horní mez polohy na hodnotu prvního bodu.



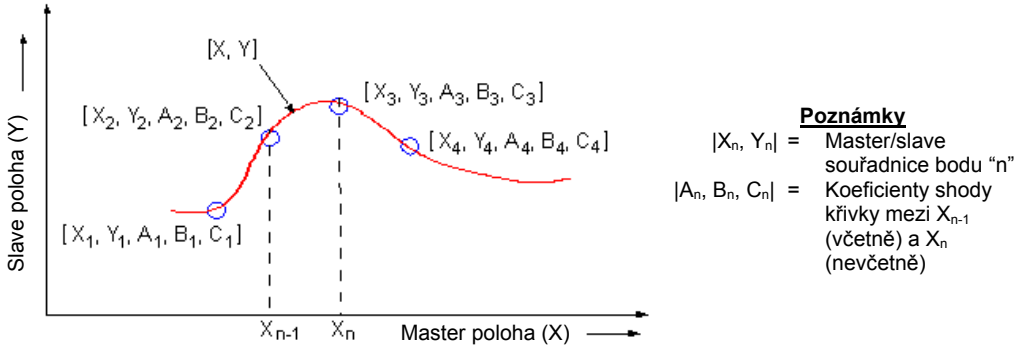
Obr. 16-7. Kruhová cyklická vačka

Interpolace a vyhlazení

Jednou z klíčových funkcí vačky je interpolační schéma používané k definování profilů vačky. Následující odstavce jsou převzaté ze systému nápovědy editoru vačky. Jsou zahrnuté do této části nejen z důvodu uvedení těchto důležitých principů, ale také jako námět pro uživatele k tomu, aby hledal další informace v on-line nápovědě editoru vačky.

Editor vačky používá hladkou polynomickou interpolaci k definování oblastí profilu, které spadají mezi uživatelem definované body. Tento přístup snižuje nároky na paměť požadovanou pro ukládání profilu do cílového modulu pohybu a přitom vytváří přesnou a hladkou trajektorii pohybu. Bez tohoto interpolačního schématu by k definování každého profilu byl zapotřebí velký počet datových bodů a tudíž velký paměťový prostor.

Profil vačky je definovaný s minimálním počtem aktuálních datových bodů. Po definování jsou tyto body zařazené do sektorů; profil se skládá z jednoho nebo více sektorů. Pro každý sektor se zadává řád shody křivky (1, 2 nebo 3). Čím vyšší řád, tím vyšší je hladkost shody křivky. Řád shody křivky je řád polynomických křivek používaných k definování oblastí sektoru, který není zadaný uživatelem definovanými body. Jednoznačné polynomické koeficienty shody křivky se generují pro každý segment sektoru (tj. mezi každým párem uživatelem definovaných bodů). Koeficienty polynomických členů se vypočítávají tak, aby obsahovaly uživatelem definované body a souhlasily se sklonem profilu na obou stranách uživatelem definovaného bodu (kromě sektorů 1. řádu).



Polynomické křivky pro profil polohy jsou popsané následujícími funkcemi:

$$Y(X) = A_{n-1}(X_n-X_{n-1})^3 + B_{n-1}(X_n-X_{n-1})^2 + C_{n-1}(X_n-X_{n-1}) + Y_{n-1}$$

kde:

Y = hodnota slave polohy pro master polohu X.

X_{n-1} = hodnota master polohy v bodě n-1.

A_{n-1}, B_{n-1}, C_{n-1} = koeficienty shody křivky v bodě n-1.

Poznámky:

- Pro danou master polohu X, která leží mezi X_{n-1} and X_n, se volí koeficienty A, B a C pro bod odpovídající X_{n-1}.
- Pro shodu křivky druhého řádu je koeficient A vždy nula a pro shodu křivky prvního řádu jsou vždy v nule koeficienty A i B.

Navazování sektorů

Proces použitý k navazování sousedních sektorů závisí na jejich řádu shody křivky. Následující popis uvádí možné scénáře.

1. řád na 1. řád

K vyhlazení přechodu mezi po sobě jdoucími sektory (to znamená s řádem shody křivky 1) se nevykoná žádný krok. Profil prostě spojí koncový bod jednoho sektoru s počátečním bodem druhého sektoru přímkou.

1. řád na 2. řád

Když kvadratický sektor (2. řádu) bude následovat po lineárním sektoru (1. řádu), polynomické koeficienty pro první segment kvadratického sektoru se vypočítají tak, aby sklon profilu byl shodný na obou stranách počátečního bodu tohoto sektoru. To znamená, že výchozí sklon kvadratického sektoru je nastavený tak, aby se shodoval s konečným sklonem lineárního sektoru.

2. řád na 1. řád

Když lineární sektor (1. řádu) bude následovat za kvadratickým sektorem (2. řádu), k vyhlazení přechodu se neprovede žádný krok. Tento typ přechodu se nedoporučuje (pokud se mu lze vyhnout), protože může vést na velké změny rychlosti nebo zrychlení řízeného serva.

2. řád na 2. řád

Když kvadratický sektor (2. řádu) bude následovat za jiným kvadratickým sektorem (2. řádu), polynomické koeficienty pro první segment druhého kvadratického sektoru se vypočítají tak, aby sklon profilu byl shodný na obou stranách počátečního bodu tohoto sektoru. To znamená, že výchozí sklon druhého kvadratického sektoru je nastavený tak, aby se shodoval s konečným sklonem prvního kvadratického sektoru.

2. řád na 3. řád

Když kvadratický sektor (3. řádku) bude následovat za kvadratickým sektorem (2. řádu), polynomické koeficienty prvního segmentu kubického sektoru se vypočítají tak, aby sklon profilu byl shodný na obou stranách počátečního bodu tohoto sektoru. To znamená, že počáteční sklon kubického sektoru bude nastavený shodně s koncovým sklonem kvadratického sektoru.

3. řád na 2. řád

Když kvadratický sektor (2. řádu) bude následovat po kubickém sektoru (3. řádu), polynomické koeficienty prvního segmentu kvadratického sektoru se vypočítají tak, aby sklon profilu byl shodný na obou stranách počátečního bodu tohoto sektoru. To znamená, že počáteční sklon kvadratického sektoru bude nastavený shodně s koncovým sklonem kubického sektoru.

3. řád na 3. řád

Když budou sousedit dva kubické sektory (3. řádu), sklony profilů před a za bodem, kde se setkávají, se nastaví shodně. Také 2. derivace profilu před a za bodem, kde se sektory setkávají, budou nastavené shodně. Polynomické koeficienty shody křivky dvou sousedních segmentů se vypočítávají současně.

Podmínky rozhraní

U necyklických profilů je nutné definovat určitou podmínku na začátku a konci profilu pro účely výpočtu polynomických koeficientů shody křivky. Podmínka počátku nebo konce může:

- být číselná hodnota 1. derivace (sklonu) profilu.
- být číselná hodnota 2. derivace profilu.
- vycházet z výchozího výpočtu.

Výchozí podmínky jsou následující:

- **Počáteční rozhraní.** Sklon v počátečním bodě profilu se vypočítá přechodným přizpůsobením polynomické křivky na první tři (sektor 2. řádu) nebo čtyři body (sektor 3. řádu) a vypočítáním sklonu přechodného polynomu v prvním bodě.
- **Koncové rozhraní.** Sklon v koncovém bodě profilu se vypočítá přechodným přizpůsobením polynomické křivky na poslední čtyři (sektor 3. řádu) a vypočítáním sklonu přechodného polynomu v koncovém bodě.

Vzájemné ovlivňování programů pro vačku

Vačkový pohyb se vyvolá v DSM314 pomocí instrukcí v pohybovém programu. Pro podporu programování vačkového pohybu se vyžadují následující nové instrukce pohybu:

1) **CAM**: Používá se v pohybovém program ke spuštění vačkového pohybu a k zadání podmínek ukončení .
2) **CAM-LOAD**: Používá se k načtení registru parametrů s počáteční polohou vačkové osy slave. Povel PMOVE je možno použít ve spojení s povelem CAM-LOAD k přemístění slave osy do počátečního bodu.
3) **CAM-PHASE**: Používá se k zadání fáze pro povely vačky. Hodnotu fáze je možno zadat buď přes registr parametrů nebo jako konstantu.

Následující odstavce popisují podrobněji syntaxi a funkci každé instrukce uvedené výše. Pravidla používaná k zadání syntaxe povelu jsou následující:

'<>' závorky – označují povinné pole.

'[ ]' závorky – označují volitelné pole.

'{}' závorky – označují pole, které se vyžaduje pro programy a podprogramy pro více os, ale je nepřípustné pro programy a podprogramy pro jednu osu.

Povel CAM

Povel CAM se používá k naprogramování vačkového pohybu pomocí zadaného profilu vačky.

Syntaxe:

CAM <"název profilu CAM ">, <vzdálenost>, <master režim>, [Podmínka cyklického ukončení]

Parametr	Popis
<"Název profilu CAM">	Název profilu vačky z knihovny (profil musí být navázaný s blokem načtení vačky). Délka názvu je omezená maximálně na 20 znaků (viz také poznámku 3 níže). **Všimněte si, že uvozovky kolem názvu jsou povinné.**
<Vzdálenost>	Maximální vzdálenost, kterou master osa může ujet, jakmile je vačka aktivní (v uživatelských jednotkách). Vzdálenost může být konstanta, parametr nebo klíčové slovo NONE. Přípustný rozsah: –MaxPosn …. (MaxPosn-1) uu/cts
<Master režim>	Master režim může být deklarovaný jako ABS (absolutní) nebo INCR (inkrementální); udává, jak se master poloha bude interpretovat. V režimu ABS se k určení slave hodnoty z tabulky CAM použije absolutní poloha master osy. V režimu INCR se předpokládá, že master hodnota v počátečním bodě povelu CAM se rovná hodnotě "CAM-fáze" a slave hodnoty vypočítané během vačkového pohybu jsou vztažené k této počáteční master hodnotě.
[Podmínka cyklického ukončení]	Udává podmínku ukončení **Cyklických** vaček. Přípustný rozsah je CTL01-CTL32. Pokud se bit CTL vyhodnotí jako Pravda, vačka se ukončí na konci aktuálního cyklu. **Všimněte si, že tento parametr se nesmí použít pro profil necyklické vačky.**

Poznámky:

1. Instrukce CAM není přípustná v pohybovém programu pro více os.
2. Povel CAM se počítá jako dvě instrukce vzhledem k maximálnímu počtu 1000 instrukcí na pohybový program.
3. Názvy profilů **UDT_CAM_1**, **UDT_CAM_2**, **UDT_CAM_3** a **UDT_CAM_4** jsou vyhrazené pro použití v budoucnu.

Argument **<vzdálenost>** v povelu CAM se používá k definování maximální vzdálenosti, kterou se master může pohybovat podél profilu před ukončením. U cyklických vaček vzdálenost může být buď větší nebo menší než délka tabulky vačky (definované jako absolutní rozdíl mezi první a poslední master polohou tabulky vačky). Pokud vzdálenost bude menší než délka tabulky, povel vačky skončí, jakmile se vzdálenost ujede. Pokud vzdálenost bude větší než délka tabulky, vačka bude cyklovat tabulkou vačky, dokud se nedosáhne zadané vzdálenosti. Tak je možno zadat, kolikrát se má cyklická vačka vykonat. Například vzdálenost 2,5 násobku délky master polohy z tabulky vačky bude mít za následek, že profil vačky se vykoná dva a půlkrát a pak se ukončí. U necyklických vaček zadaná vzdálenost nesmí přesáhnout délku tabulky.

Vzdálenost se také může vyjádřit jako “NONE”. V případě cyklické vačky to bude mít za následek souvislý vačkový pohyb, dokud bit CTL nevyvolá ukončení nebo dokud se pohyb nepřeruší. U necyklických vaček zadání "NONE" znamená, že vačka se ukončí, když dosáhne buď minimální nebo maximální master polohy profilu.

[**Master režim**] se používá k zadání, jestli master osa bude pracovat v absolutním nebo inkrementálním režimu. Master osa může pracovat v absolutním nebo inkrementálním režimu u cyklické i necyklické vačky. V absolutním režimu master polohy v tabulce reprezentují absolutní polohy master osy. V inkrementálním režimu polohy slave osy v tabulce jsou vztažené k poloze master osy, když se vyvolá instrukce CAM.

[Podmínka cyklického ukončení] se používá k zadání podmínky ukončení profilu cyklické vačky. Pokud se podmínka CTL vyhodnotí jako TRUE, vačka se ukončí na konci aktuálního cyklu.

Obousměrné a **jednosměrné** vačky je možno definovat pomocí parametrů meze rychlosti master osy +Vlim a -Vlim. V případě jednosměrné činnosti musí být pro směr, ve kterém je pohyb zakázaný, příslušná mez rychlosti nastavená na nulu. Pokud například bude zakázaný pohyb v záporném směru, pak -Vlim musí být nastavený na nulu.

Povel CAM-LOAD

Povel CAM-LOAD se používá k načtení polohy slave osy do registru parametrů. K posunu slave osy do načtené polohy pak je možno použít normální povel PMOVE. Povel CAM-LOAD používá název profilu CAM, okamžitou polohu a fázi masteru (zadanou pomocí povelu CAM-PHASE) k určení počátečního bodu osy slave.

Syntaxe:

CAM-LOAD <"název profilu vačky">, <číslo parametru>, <master režim>

Parametr	Popis
<"Název profilu vačky">	Název profilu vačky z knihovny (profil musí být navázaný s blokem načtení vačky). Délka tohoto názvu je omezená maximálně na 20 znaků (viz také poznámka 3 níže). **Všimněte si, že uvozovky kolem názvu jsou povinné.**
<číslo parametru>	Udává číslo načítaného parametru.
<master režim>	Master režim může být deklarovaný jako ABS (absolutní) nebo INCR (inkrementální); udává, jak se master poloha bude interpretovat při výpočtu počáteční polohy slave. V režimu ABS se absolutní poloha master osy používá k určení odpovídající hodnoty počáteční polohy slave z tabulky vačky. V režimu INCR se předpokládá, že master hodnota se rovná fázi vačky ve výpočtech.

Poznámky:

1. Pro účely maximálního počtu 1000 instrukcí na pohybový program se povel CAM-LOAD počítá jako dvě instrukce.
2. Když se vykoná povel CAM-LOAD, provede se následující sekvence kroků:
   A. Načte se aktuální poloha masteru.
   B. Pomocí master polohy, hodnoty fáze vačky, tabulky profilu vačky a konfigurační tabulky vačky se vypočítá příslušná poloha slave osy a načte se do určeného registru parametrů.
   C. Pohybový program může použít instrukci PMOVE k vykonání pohybu slave osy do polohy vypočítané v kroku B.
3. Názvy **UDT_CAM_1**, **UDT_CAM_2**, **UDT_CAM_3** a **UDT_CAM_4** jsou vyhrazené pro použití v budoucnu a nelze je použít pro názvy profilu vačky.

Povel CAM-PHASE

Povel CAM-PHASE se používá k zadání fáze povelů vačky. Tento povel umožňuje provést posunutí nebo posunout fázový poměr mezi master polohou a polohou vlečené osy. Hodnotu fáze je možno zadat buď přes registr parametrů nebo jako konstantu. Všimněte si, že hodnota fáze je aktivní pro všechny instrukce CAM, které následují, dokud nebude změněná jiným povelem, CAM-PHASE. Výchozí hodnota fáze vačky pro pohybový program je 0.

Syntaxe:

CAM-PHASE <fáze>

Parametr	Popis
<fáze>	Hodnota fáze vačky zadaná jako konstanta nebo jako parametr registrů. Přípustný rozsah konstanty: –MaxPosn …. (MaxPosn-1)

Instrukce CAM a MOVE

Řada povelů CAM se může vykonat bez prodlevy nebo přerušení. Aby se získal hladký pohyb, je nutno zajistit, aby počáteční bod každého následujícího profilu vačky byl stejný, jako koncový bod předcházejícího profilu vačky. Tím se zajistí souvislá poloha a rychlost trajektorie. Aby se pro sekvenci necyklických vaček získaly hladké přechody, počáteční a koncový bod je možno přizpůsobit v editoru vačky. Přechody mezi povely CAM a MOVE během pohybu slave osy nejsou přípustné. Proto slave osa musí mít v bodě přechodu mezi povelem CAM a MOVE počáteční rychlost 0. Když se povel CAM ukončí a nenásleduje za ním hned další povel CAM, osa ke zpomalení a zastavení použije naprogramovanou velikost zrychlení.

Pohyb vačky na bázi času

Použití profilu vačky na bázi času používá stejný mechanismus jako obyčejná vačka (master na bázi polohy). Má-li se naprogramovat profil vačky na bázi času, master zdroj vačky je nutno v hardwarové konfiguraci modulu DSM314 nakonfigurovat jako “Zadanou polohu” osy 3 s režimem osy 3 nastaveným jako “Pomocná osa”. Pro osu 3 se pak vyvolá povel konstantní rychlosti. Aktivní časová osa pohybu vačky se pak stanoví v hardwarové konfiguraci nastavením měřítka masteru v souboru zdroji profilu a koeficientem převodu uživatelských jednotek na pulsy. Povel pohybu vačky na bázi času je možno vykonat současně v několika osách.

Editor měřítka vačky a hardwarová konfigurace

DSM modul umožňuje nastavit měřítko rozlišení polohového zpětnovazebního zařízení vzhledem k programovacím jednotkám modulu. Předpokládejme například, že 1 otáčka motoru odpovídá posunutí poháněného břemena o 1 palec. V tomto příkladu motor připojený k poháněnému břemenu má snímač polohy, který dává 8192 pulsů na otáčku motoru. Pak se 8192 zpětnovazebních pulsů bude rovnat 1 palci posunutí břemena. Někomu se bude zdát jednodušší naprogramovat pohyb v palcích místo v jednotkách zpětné vazby. V takovém případě je možno nastavit měřítko pohybového programu v tisícinách palce. Aby se získal tento výsledek, v hardwarové konfiguraci DSM nastavte poměr uživatelských jednotek na počet pulsů jako poměr 1000 ku 8192. Další informace o zadání těchto hodnot najdete na v kapitole 5.

Funkce vačky také podporuje jednotky specifické pro danou aplikaci. Musíte však ručně přesunout hodnoty zapsané v hardwarové konfiguraci do příslušné paměti v editoru vačky.

Poznámka: Tyto hodnoty je nutno přesunout pro master i slave osu. Když budeme vycházet z předchozího příkladu, předpokládejme, že master a slave osa mají stejný motor. Proto obě zařízení zpětné vazby mají stejný počet pulsů 8192 na otáčku. Avšak pro master se jedna otáčku motoru rovná 1 palci pohybu břemena, zatímco pro slave se jedna otáčka motoru rovná 5 palcům pohybu břemena. Aby bylo jednodušší pochopit programy, je nutno master i slave naprogramovat ve stejných jednotkách. V tomto příkladu se používají jednotky 0.001 palce. Abychom dostali stejný výsledek, nejdříve je nutno pro master i slave stanovit správné poměry uživatelských jednotek na počet pulsů.

Pro stanovení poměru platí následující rovnice.

$$\frac{\text{Uživ. jednotky}}{\text{Počet pulsů}} = \left(\frac{\text{Pohyb břemena na otáčku motoru}}{\text{Požadované rozlišení}} \right) \cdot \frac{1}{\text{Počet pulsů zpětné vazby na otáčku motoru}}$$

Pro master osu v tomto příkladu platí:

$$\frac{\text{Uživ. jednotky}}{\text{Počet pulsů}} = \left(\frac{1 \text{ in}}{\frac{1}{1000}} \right) \cdot \frac{1}{8192 \text{ pulsů}}$$

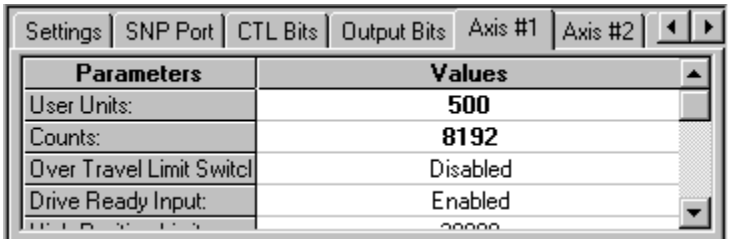
$$\frac{\text{Uživ. jednotky}}{\text{Počet pulsů}} = \left(\frac{1000}{8192} \right) \cdot \frac{\text{in}}{\text{Počet pulsů}}$$

Pro podřízenou osu v tomto příkladu platí:

$$\frac{\text{Uživ. jednotky}}{\text{Počet pulsů}} = \left(\frac{.5 \text{ in}}{\frac{1}{1000}} \right) \cdot \frac{1}{8192 \text{ pulsů}}$$

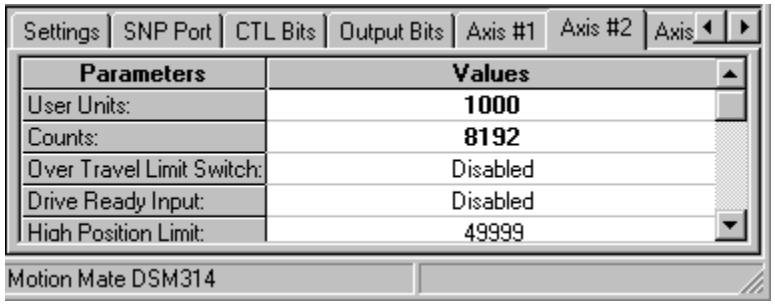
$$\frac{\text{Uživ. jednotky}}{\text{Počet pulsů}} = \left(\frac{500}{8192} \right) \cdot \frac{\text{in}}{\text{Počet pulsů}}$$

Pak může být nutné v hardwarové konfiguraci tyto hodnoty zapsat na různých místech. V tomto příkladu osa #2 je master a osa #1 je slave. Proto v hardwarové konfiguraci zapište hodnoty uživatelských jednotek a počtu pulsů pro slave, jak je znázorněno v Obr. 16-8.



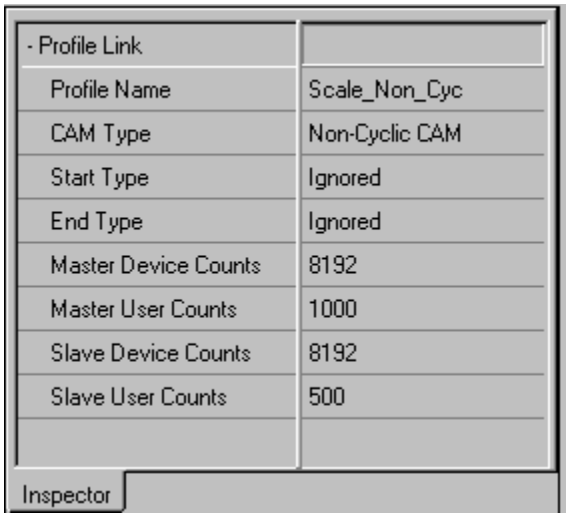
Obr. 16-8. Hardwarová konfigurace uživatelských jednotek/počet pulsů slave

Hodnotu uživatelských jednotek/počet pulsů masteru je možno zadat jako na Obr. 16-9.



Obr. 16-9. Hardwarová konfigurace uživatelských jednotek/počet pulsů masteru

To řekne modulu správné měřítko, které má použít při spuštění pohybových programů. Avšak editor vačky také potřebuje znát správné měřítko pro provedení řádného převodu uživatelských jednotek na pulsy. V tomto příkladu je nutno data zapsat do všech profilů vačky, které se vykonávají na těchto osách. Příklad je uvedený na Obr. 16-10.



Obr. 16-10. Editor vačky uživatelských jednotek/počet pulsů pro slave a master

Doporučuje se provést právě diskutované operace nastavení měřítka před programováním vačky, které nemají měřítko jedna ku jedné, Důvodem je vyhnout se nutnosti opakovaného zápisu dat. Editor vačky zobrazuje data master/slave v uživatelských jednotkách. Proto pokud nedefinujete své

měřítko a zapíšete všechna data vačky implicitně, zvolili jste měřítko 1 ku 1. Když konečně opravíte tuto chybu a zapíšete správné měřítko, všimnete si, že se všechna nenulová čísla v datové tabulce vačky změní tak, aby se vyjádřily hodnoty nových uživatelských jednotek.

Následující část pojednává o tom, jak editor vačky při zápisu dat zaokrouhluje hodnoty. Tato funkce se vykonává automaticky a není nutno žádným speciálním způsobem editor spustit nebo nakonfigurovat. Tato část je uvedena pouze pro úplnost předmětu nastavení měřítka uvedeného v předchozích odstavcích.

Všimněte si, že DSM interně pracuje v původních jednotkách zpětné vazby a převod původních jednotek na uživatelské jednotky provádí automaticky. Modul vykonává tuto operaci proto, aby využil celou použitelnou rozlišitelnost zpětné vazby. To zahrnuje i případ, kdy si uživatel zvolí programovat pohyb v uživatelských jednotkách, které nemají plnou rozlišitelnost zpětnovazebního zařízení. Editor vačky se také snaží pro nastavený poměr daného motoru/zpětné vazby udržet celou použitelnou rozlišitelnost (aniž by se vykazovala chybná rozlišitelnost). Proto když se v editoru vačky zadá měřítko, editor v některých případech do datové tabulky přidá desetinná místa. Dále automaticky zaokrouhlí čísla na hodnoty, které je možno vyjádřit jako celá čísla v počtu zpětnovazebních jednotek. Příklad vačky uvedený na obrázku 16-1 vychází z předchozího příkladu. Všimněte si, že editor vačky zobrazuje hodnoty v uživatelských jednotkách, ale vždy je zaokrouhlí na celočíselnou hodnotu v počtech pulsů.

Tabulka 16-1. Příklad dat vačky v palcích

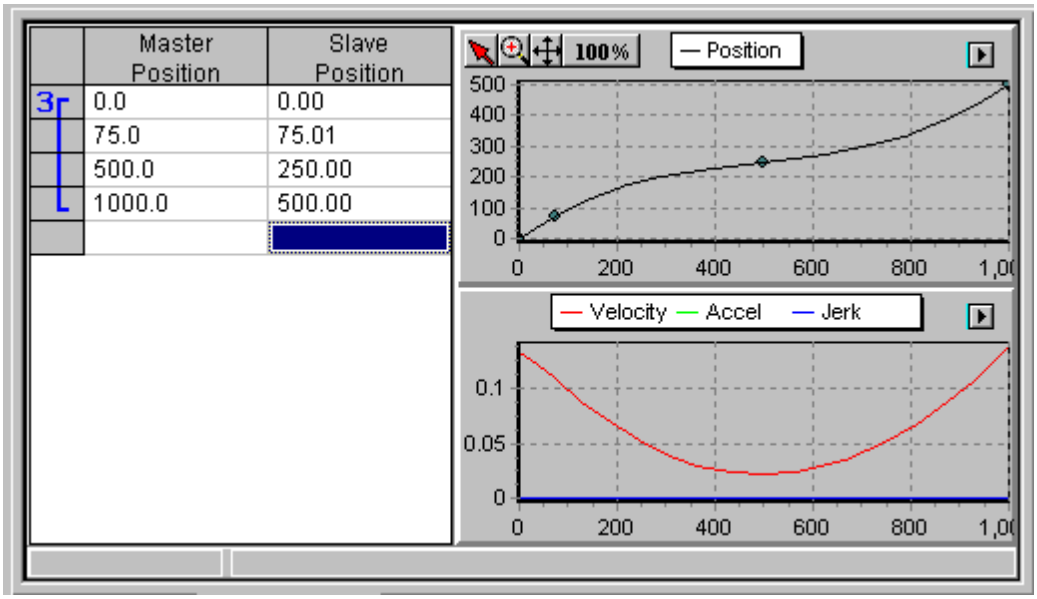
Master poloha (palce)	Slave poloha (palce)
0	0
0.075	0.075
0.5	0.25
1	0.5

Výše uvedená tabulka ukazuje hodnoty v palcích. V tomto příkladu je vačka naprogramovaná v jednotkách 1/1000 palce. Proto tyto hodnoty převeďte a zapište data do editoru vačky, jak je znázorněno v Tabulka 16-2.

Tabulka 16-2. Příklad dat vačky v jednotkách 0,001 palce

Master poloha (1/1000 palce)	Slave poloha (1/1000 palce)
0 palce =0 tisícin palce	0 palce =0 tisícin palce
0,075 palce =75 tisícin palce	0,075 palce =75 tisícin palce
0,5 palce =500 tisícin palce	0,25 palce =250 tisícin palce
1 palec =1000 tisícin palce	0,5 palce =500 tisícin palce

Když zapíšete data do editoru vačky (Obr. 16-11), editor vačky některé hodnoty automaticky změní.



Obr. 16-11. Příklad uživatelských jednotek z datové tabulky vačky

Editor vačky automaticky změní hodnoty tak, aby odpovídaly celému číslu počtu zpětnovazebních pulsů. Editor také zaokrouhlí zobrazené hodnoty tak, aby se omezil zmatek v tabulce. Všimněte si, že editor zachovává přesnost proměnných a zaokrouhluje se pouze zobrazení. Funkce, které editor vačky provádí automaticky, jsou zobrazené níže. Nejdříve určete celý počet zpětnovazebních pulsů, který je nejblíže požadovaným hodnotám. V tomto příkladu nelze přesně vyjádřit pouze dvě čísla. Jsou to master poloha 75 a slave poloha 75. Nejbližší celočíselná hodnota počtu pulsů k těmto hodnotám je 614 pulsů pro master a 1229 pulsů pro slave. Vztah mezi master polohou a slave polohou vzhledem k uživatelským jednotkám a počtu pulsů je ukázaný v Tabulka 16-3.

Master poloha (Uživatelské jednotky)	Master poloha (Počet pulsů)	Slave poloha (Uživatelské jednotky)	Slave poloha (počet pulsů)
0	0	0	0
74.951171875	614	75.01000000000001	1229
500	4096	250	4096
1000	8192	500	8192

Tabulka 16-3 Vztah mezi uživatelskými jednotkami a počtem pulsů v příkladu nastavení měřítka

To souhlasí s funkcemi, které automaticky vykonal editor. Všimněte si, že master polohu 74.951171875 editor zaokrouhlí na 75.0. Slave polohu 75.01000000000001 editor zaokrouhlí na 75.01.

Synchronizace pohybu vačky s externími událostmi.

Následující mechanismy umožňují programátorovi synchronizovat pohyb vačky s externími událostmi:

- Start pohybu vačky je možno synchronizovat s externí událostí pomocí povelu WAIT, která je v pohybovém programu.
- Cyklickou vačku je možno synchronizovat se vzorkováním pomocí proměnných lokální logiky. Další podrobnosti ohledně lokální logiky najdete v kapitole 11-14.

Chybové kódy DSM týkající se vačky

Tabulka 16-4. Chybové kódy týkající se vačky

Chybové kódy programu vačky				
Chybový kód (hex)	**Odezva**	**Popis**	**Typ chyby**	**Možná příčina**
2A	Normální zastavení	CTL podmínka ukončení cyklické vačky zadaná pro necyklickou vačku	Osa	Podmínka ukončení v CTL je přípustná pouze pro cyklické vačky. Pohybový program obsahuje instrukci necyklické vačky s CTL podmínkou ukončení.
2B	Normální zastavení	Fáze vačky mimo rozsah	Osa	Hodnota CAM PHASE je mimo rozsah polohy osy.
Chybové kódy konfigurace vačky				
2D	Normální zastavení	Chyba konfigurace vačky master osy – master profil nesouhlasí s konfigurací master osy	Osa	Poměr uživatelských jednotek : počet pulsů pro master osu v editoru a v hardwarové konfiguraci nejsou kompatibilní a/nebo Mez horní/dolní polohy zadaná pro master osu v hardwarové konfiguraci není kompatibilní s profilem. Podrobný popis nastavení mezí horní/dolní polohy najdete v části o typech vaček.
2E	Normální zastavení	Chyba konfigurace vačky slave osy – slave profil nesouhlasí s konfigurací slave osy	Osa	Poměr uživatelských jednotek : počet pulsů pro slave osu v editoru a v hardwarové konfiguraci nejsou kompatibilní a/nebo Mez horní/dolní polohy zadaná pro slave osu v hardwarové konfiguraci není kompatibilní s profilem. Podrobný popis nastavení mezí horní/dolní polohy najdete v části o typech vaček.
2F	Normální zastavení	Režim SW EOT vačky slave osy nelze povolit pro cyklickou kruhovou vačku	Osa	

Chybové kódy parametrů konfigurace				
Chybový kód (hex)	**Odezva**	**Popis**	**Typ chyby**	**Možná příčina**
1D	Normální zastavení	Snaha použít povely CAM, CAM-Load nebo CAM-Phase v režimu vlečené osy	Osa	Když budete používat vačku, přesvědčte se, že není nakonfigurovaný režim vlečené osy (režim vlečené osy nelze použít, když se používá vačka). Pokud budete používat režim vlečené osy, zajistěte, aby se v pohybových programech nevyskytovaly povely vačky (vačku nelze použít, když bude nakonfigurovaný režim vlečené osy).
Chybové kódy vykonávání vačky				
66	Normální zastavení	Profil vačky nenalezen v načteném bloku vačky	Osa	Profil vačky není v editoru vačky napojený na blok načtení vačky a/nebo v hardwarové konfiguraci nebyl zadaný název bloku načtení vačky.
67	Normální zastavení	Vzdálenost ukončení vačky je mimo rozsah (necyklické vačky)	Osa	Vzdálenost ukončení pro necyklické vačky je větší než je modul vačky.
68	Pouze stav	(Korekce povolena) Povel rychlosti je omezený v důsledku porušení meze rychlosti nebo porušení meze polohové odchylky.	Osa	
68	Normální zastavení	(Korekce zakázána) Povel rychlosti vačky je větší než nakonfigurovaná mez rychlosti osy.	Osa	
6A	Normální zastavení	Porušení meze polohové odchylky vačky (se zakázanou korekcí)	Osa	
6B	Pouze stav	Zadaná poloha vačky při ukončení je jiná než hodnota profilu vačky v důsledku meze polohové odchylky nebo rychlosti.	Osa	
6C	Normální zastavení	Master hodnota vačky je mimo master rozsah profilu necyklické vačky (povely CAM a CAM-LOAD)	Osa	
6D	Normální zastavení	Absolutní režim vačky po inkrementálním režimu vačky za sebou	Osa	
6F	Rychlé zastavení	Chyba výpočtu trajektorie vačky	Osa	Spojte se s GE Fanuc Automation

Část 3: Základy programování elektronické vačky

Tato část obsahuje úvod do základů koncepce programování elektronické vačky. Funkce lokální logiky a pohybové programy se v této části neprobírají podrobně, protože byly probírané v jiných kapitolách tohoto manuálu.

Požadavky

Editor lokální logiky, editor vačky a editor pohybového programu jsou zakomponované do prostředí programovacího softwaru. Budete potřebovat některý z následujících softwarových balíků. Návod k instalaci najdete v dokumentaci k softwaru.

DSM314 vyžaduje následující konfigurační/programovací softwarové balíky:

- CIMPLICITY Machine Edition Logic Developer – PLC verze 2.1 nebo pozdější
- VersaPro verze 1.1 nebo pozdější

Nastavení funkce DSM314 vyžaduje CPU 90-30 s firmwarem verze 10.0 nebo pozdější. Funkce vačky také vyžaduje firmware DSM314 verze 2.0 nebo pozdější.

Úvod do programování elektronické vačky

Funkce elektronické vačky pracuje ve spojení s pohybovým programem DSM314, programem lokální logiky DSM314 a programovým prostředím PLC a tím se vytváří flexibilní programovací prostředí. Funkce elektronické vačky konkrétně umožňuje zadávat přesné polohové vztahy mezi master osou a slave osu. Tato schopnost je kritická pro mnoho aplikací, kde těsná synchronizace mezi osami je absolutním požadavkem.

DSM funkce elektronické vačky umožňuje zapsat tyto polohové vztahy pomocí integrovaného softwarového programovacího nástroje. Editor vačky umožňuje zapsat tyto vztahy grafiky, v tabulkovém tvaru nebo v kombinaci obou možností. Tyto profily vačky se pak uloží do DSM modulu, kde k nim pak mají přístup pohybové programy DSM. Základní koncepce vačky jsou jednoduché a nejlépe se vysvětlí pomocí jednoduchého příkladu.

Vytvoření příkladu aplikace vačky

Základní kroky

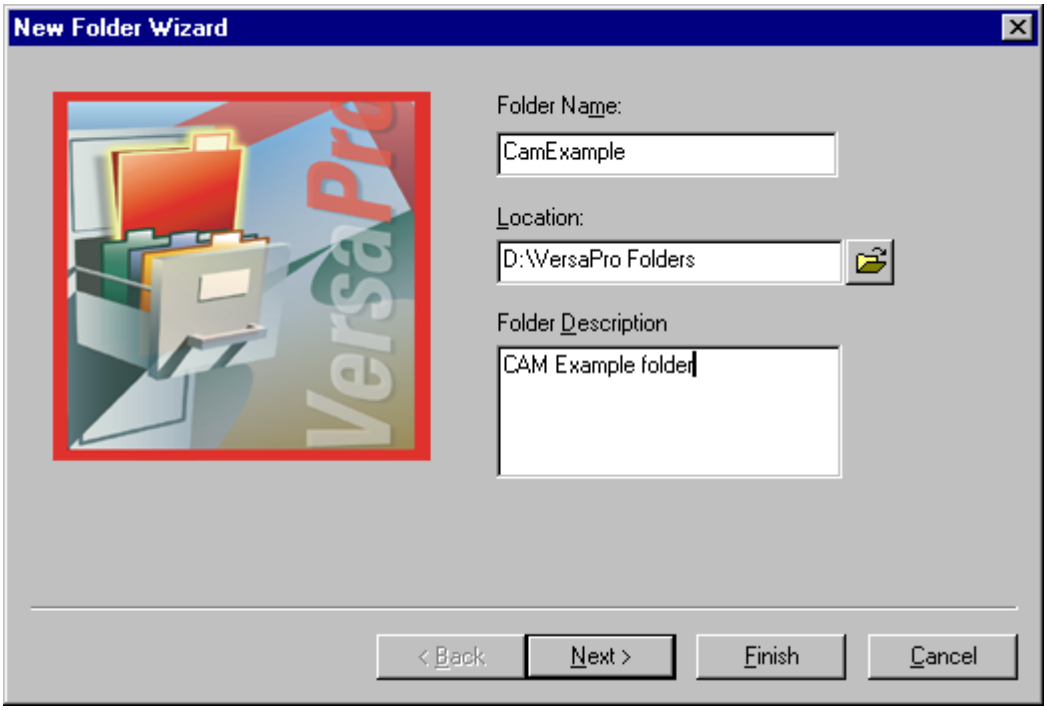
1. Otevřete adresář projektu nebo vytvořte nový adresář
2. Vytvořte blok vačky
3. Vytvořte profil vačky
4. Navažte profil vačky s blokem vačky
5. Proveďte konfiguraci profilu vačky

6. Zadejte typ vačky
7. Zadejte vlastnost Korekce
8. Uložte profil vačky
9. Vygenerujte pohybový program a program lokální logiky
10. Proveďte nastavení hardwarové konfigurace v konfiguračním/programovacím softwaru
11. Vykonejte (vyzkoušejte) aplikaci

VersaPro

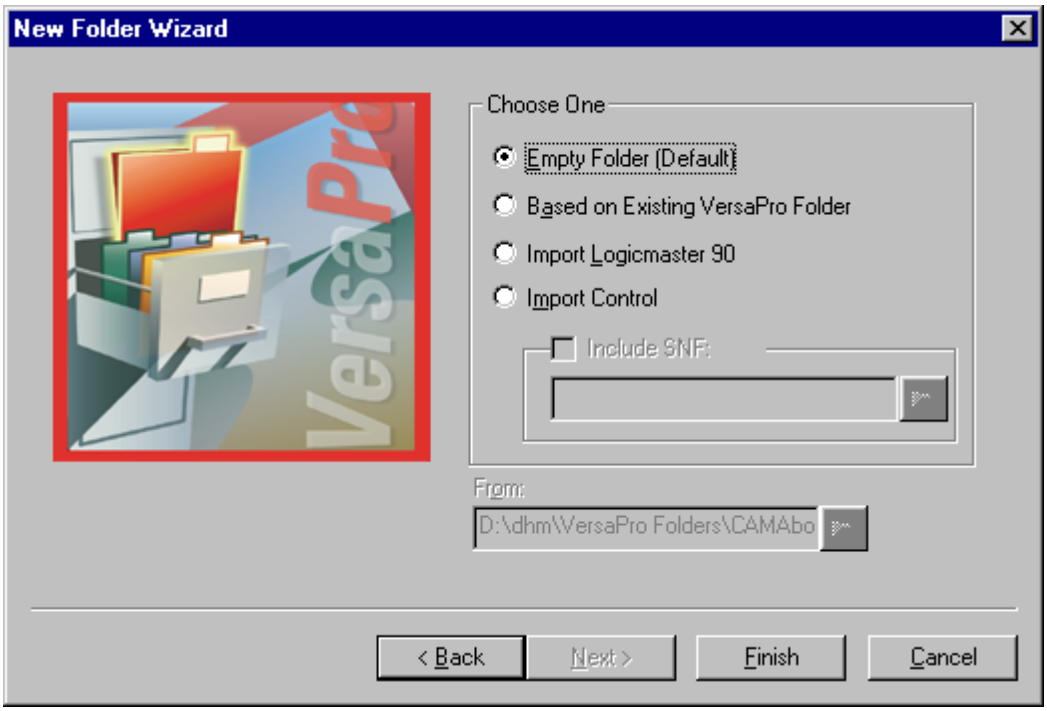
Krok 1: Vytvoření nového adresáře

Měli byste začít otevřením VersaPro. K tomu potřebný způsob je popsaný v kapitole 15 tohoto manuálu, v on-line nápovědě VersaPro a v manuálu VersaPro, GFK-1670. Jakmile bude VersaPro otevřený, vytvořte nový adresář otevřením menu **File** a zvolením podmenu **New Folder**. Objeví se dialogové okno, které umožní vytvořit nový adresář. V tomto příkladu adresář pojmenujte "CAMExample," jak je ukázáno na Obr. 16-12.



Obr. 16-12. Stránka 1 obrazovky New Folder Wizard

Kliknutím na tlačítko Next se vás VersaPro zeptá, jestli chcete vytvořit adresář (pokud neexistuje). Jakmile adresář bude vytvořený, objeví se další dialogové okno, které jako zdroj nového adresáře nabídne několik voleb. Tento příklad začíná prázdným adresářem, který je výchozí (Obr. 16-13).

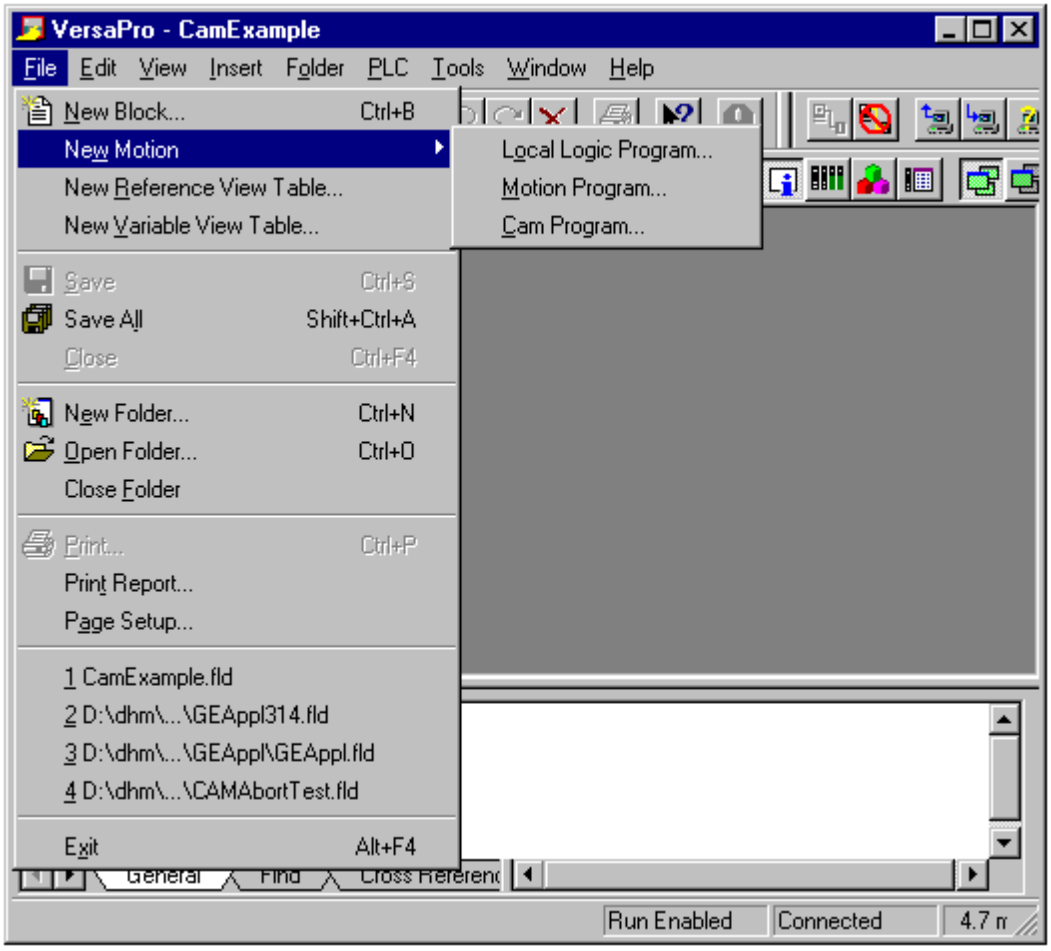


Obr. 16-13. Stránka 2 obrazovky New Folder Wizard

Kliknutí na tlačítko Finish způsobí, že VersaPro vytvoří adresář a otevře hlavní program.

Krok 2: Vytvoření bloku vačky pomocí editoru vačky

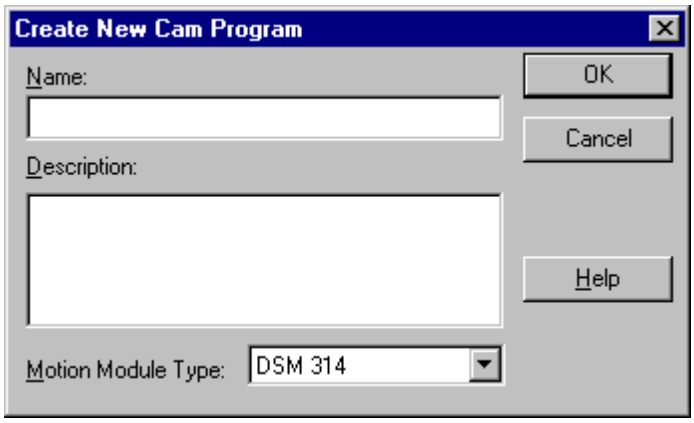
Editor CAM je součástí prostředí VersaPro. Editor umožňuje snadné vytváření, editování, ukládání a načítání bloků vačky. Chcete-li vytvořit blok vačky, je nutno otevřít nebo vytvořit nový adresář VersaPro (viz krok 1). Informace, jak vytvořit nebo otevřít adresář, najdete v kapitole 15. Jakmile adresář VersaPro bude otevřený, zvolte menu File, pak zvolte New Motion a potom menu CAM Program… (Obr. 16-14).



Obr. 16-14. Vytvoření programu vačky

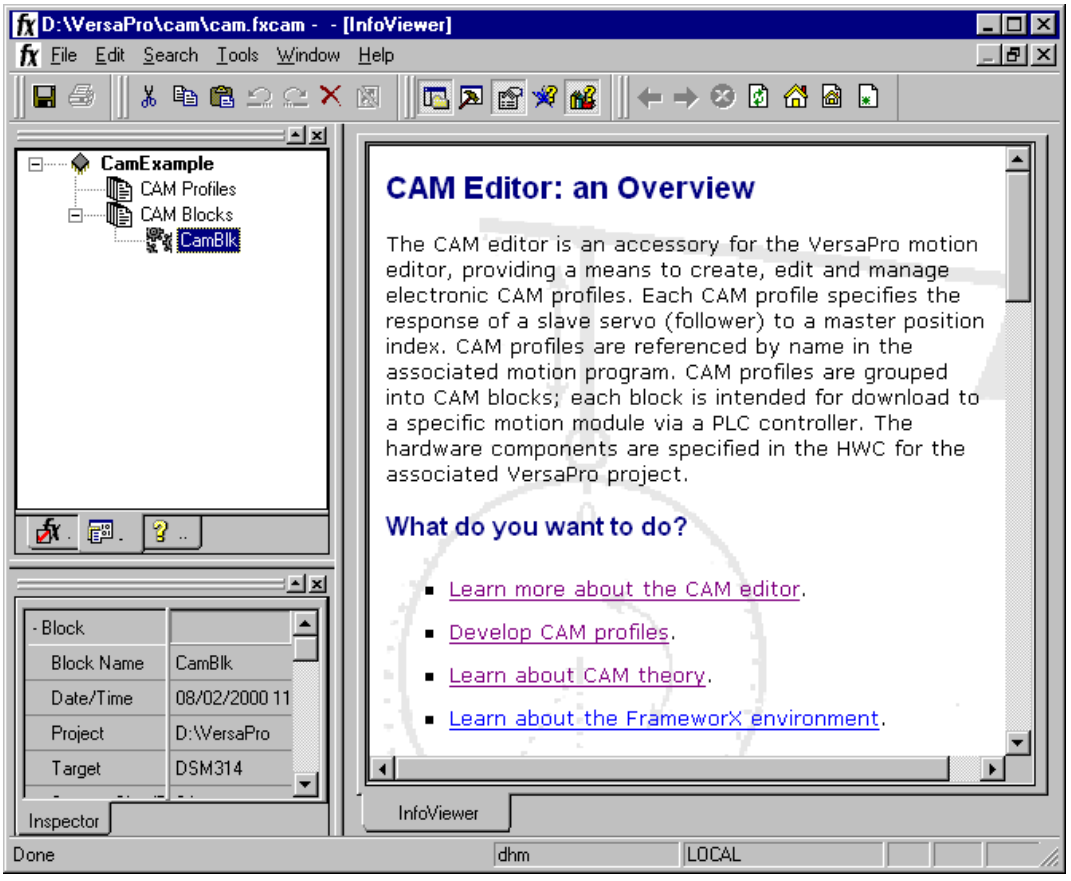
Objeví se dialogové okno “Create New Program”. Pojmenujte blok vačky a přidejte popisný komentář. V současné době funkci vačky podporuje DSM314 (verze 2.0 nebo pozdější). Proto by se výchozí volba v Motion Module Type neměla měnit (Obr. 16-15). Pravidla pro název bloku vačky jsou:

- Jsou přípustné pouze znaky A-Z, a-z, 0-9 a _ (znak podtržítka). Po sobě jdoucí podtržítka nejsou přípustná.
- Název bloku musí začínat písmenem nebo znakem podtržítka.
- Blok nesmí mít stejný název jako jiný blok existující v otevřeném adresáři.
- Název bloku vačky smí obsahovat maximálně dvacet znaků.



Obr. 16-15. Vytvoření nového programu vačky

V dialogovém okně “Create New CAM Program” zapište data, pak kliknutím na tlačítko OK vytvořte blok vačky. VersaPro spustí editor programu vačky. (Obr. 16-16)

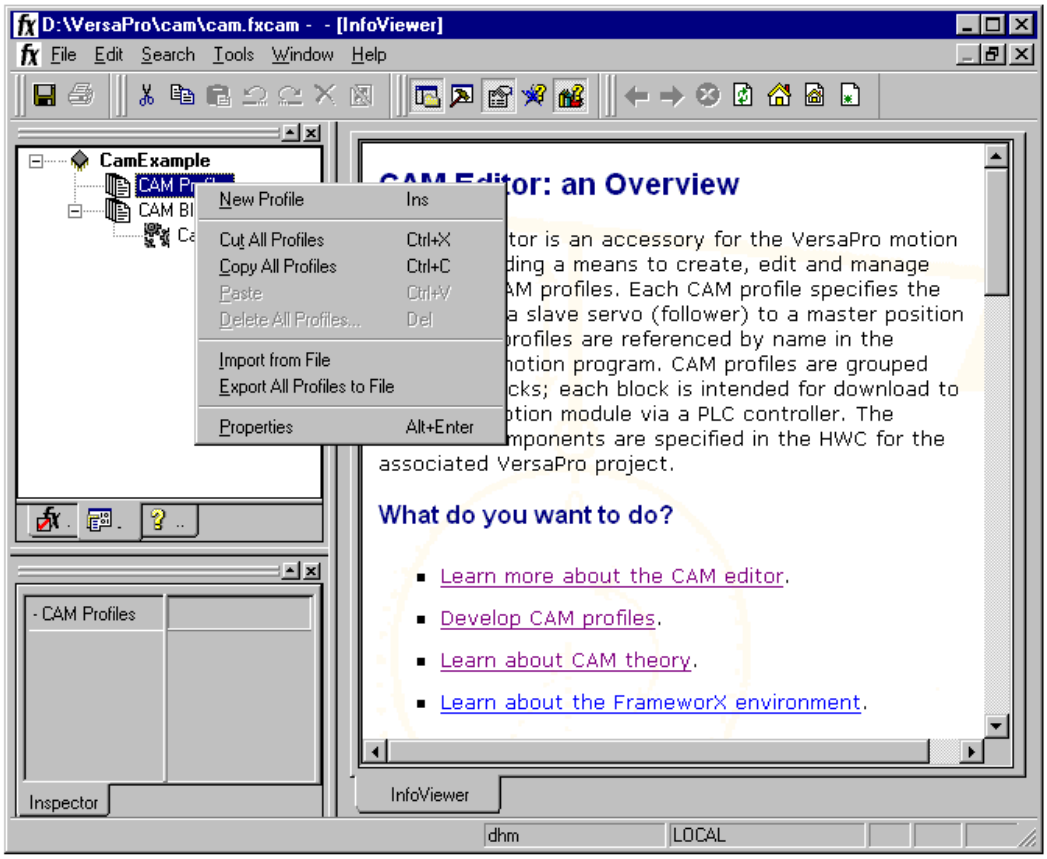


Obr. 16-16. Výchozí editor vačky s obrazovkou InfoViewer

Editor vačky obsahuje rozsáhlou on-line hypertextovou nápovědu. Tento manuál se pouze snaží uvést některé tyto koncepce. Nesnaží se pokrýt všechny vlastnosti editoru, takže se rozhodně doporučuje si zopakovat on-line materiál programovacího softwaru.

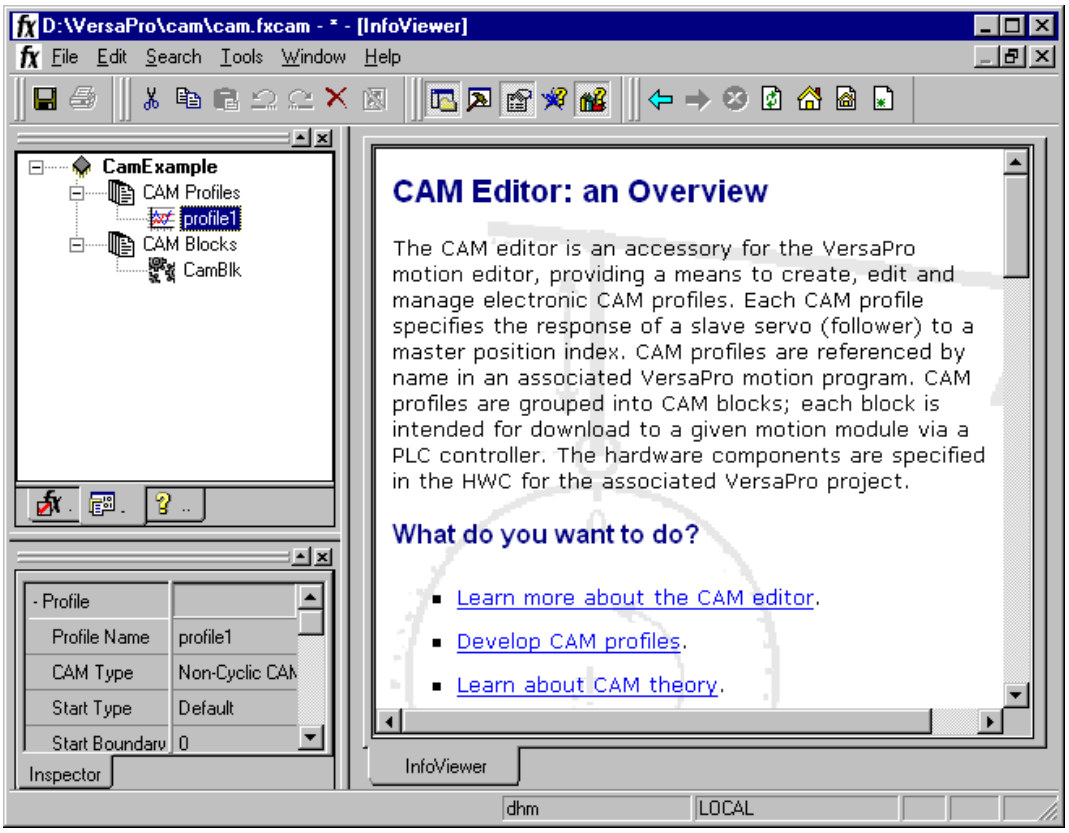
Krok 3: Vytvoření profilu vačky

Další krok je vytvoření jednoduchého profilu vačky v editoru vačky. Editor vačky má knihovnu profilů, který si vytváří uživatel. Profily vačky v knihovně se pak navazují na bloky vačky. Další informace o tomto navazování najdete v on-line nápovědě. V tomto příkladu je nutno nejdříve profil v knihovně vytvořit. Jeden způsob, jak provést tento krok, je kliknout pravým tlačítkem myši na ikonu “CAM Profiles” v okně Navigátor. Objeví se klávesová zkratka menu. Zvolte New Profile, jak je ukázáno na Obr. 16-17.



Obr. 16-17. Vytvoření nového profilu vačky

Tím se do knihovny vloží nový profil s názvem "profile1", jak je ukázáno na Obr. 16-18.

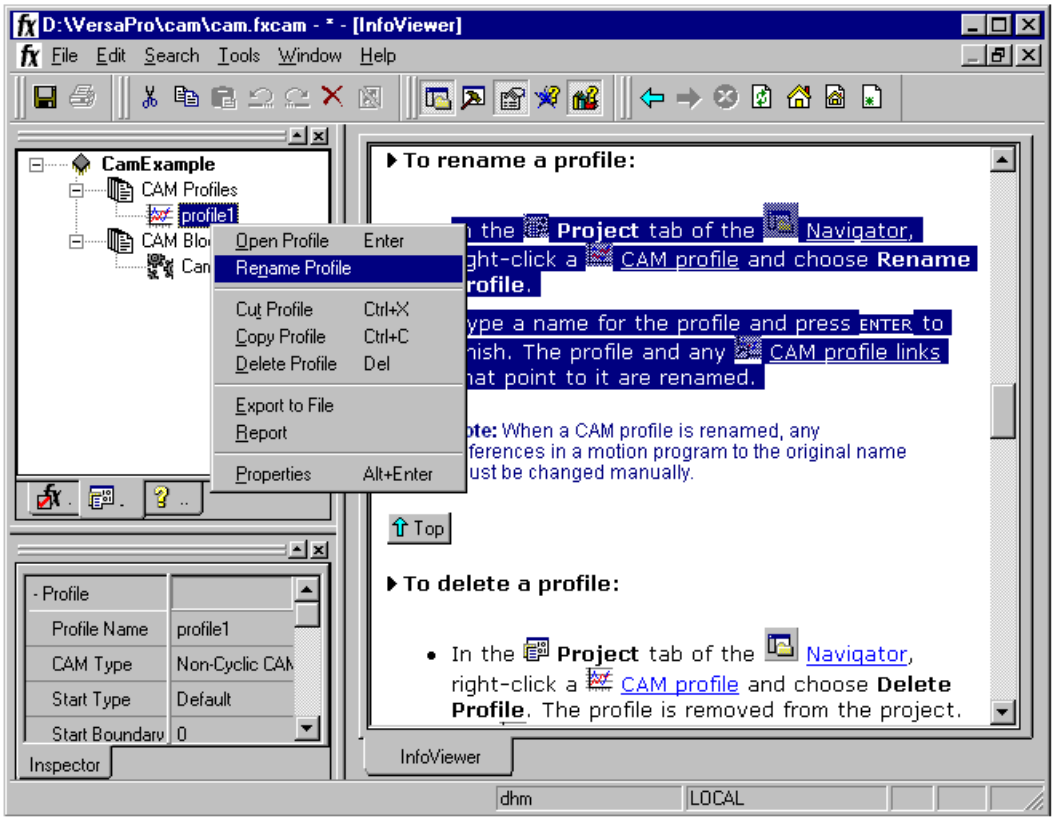


Obr. 16-18. Vytvoření nového profilu

Pak, pokud budete chtít, můžete tomuto profilu dát nový název, který bude pro aplikaci vhodnější. Pravidla pro přejmenování jsou:

- Je možno použít libovolný alfanumerický znak nebo znak podtržítka (_).
- První znak názvu profilu musí být písmeno.
- Název profilu nesmí být delší než 20 znaků.
- Profil se v pohybovém programu VersaPro adresuje podle názvu. POZNÁMKA: VersaPro při adresování názvu profilu nebere ohled na velikost písmen.

Jeden způsob, jak přejmenovat profil, je kliknout pravým tlačítkem myši na název profilu v okně Navigátoru a z menu klávesových zkratek zvolit Rename Profile (Obr. 16-19). Zapište název profilu a stisknutím ENTER zápis ukončete. Profil a všechny k němu připojené profily vačky se přejmenují. U tohoto příkladu se název přejmenoval na ExCamProfile. Další informace najdete v on-line nápovědě.



Obr. 16-19. Přejmenování profilu

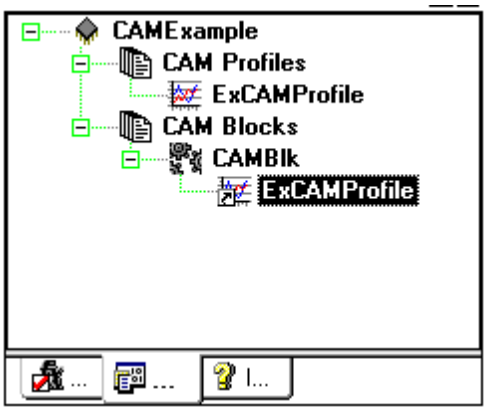
Krok 4: Navázání profilu vačky s blokem vačky

- Profily vačky musí být navázané se souvisejícím blokem vačky. Blok vačky může obsahovat mnoho profilů vačky. DSM má dvě omezení, která se dotýkají počtu profilů. Maximální velikost bloku je 50K a maximální počet navázaných profilů na jeden blok je 100. Knihovna profilů vačky je omezená pouze volným prostorem na disku na nadřízeném počítači. Blok vačky se pak naváže na DSM přes zápis bloku vačky v hardwarové konfiguraci.

  I když existuje více než jeden způsob navázání profilu vačky s blokem vačky, nejjednodušším způsobem je jednoduše kliknout na požadovaný profil vačky, pak ho přetáhnout a pustit na příslušný blok vačky. Výsledek je znázorněný na následujícím obrázku.

Poznámka

VersaPro má omezenou celkovou velikost načítaného bloku (kombinace pohyb, lokální logika a CAM) na 32K.



Obr. 16-20. Navázání profilu na blok vačky

Krok 5: Konfigurace datových bodů profilu vačky

Jakmile budou tyto operace hotové, je nutno nakonfigurovat profil vačky. Profil vačky je vztah mezi master polohou a slave polohou. Profil vačky se skládá ze série bodů. Každý bod je definovaný dvěma souřadnicemi. Z pohledu grafické reprezentace master souřadnice představují horizontální osu a slave souřadnice představují vertikální osu, jak je znázorněno na následujícím obrázku.

Začněte dvojitým kliknutím na profil a tím otevřete okno editoru profilu (viz následující obrázek), který má dva editory:

- **Editor tabulky** je podobný tabulkovému procesoru. V tabulce každý bod má svou řadu se dvěma sloupci, jeden pro master polohu a jeden pro odpovídající slave polohu. Když se otevře nový profil, implicitně v něm jsou pouze dva body, počáteční a koncový bod. Počáteční bod je horní bod tabulky a koncový bod je dole.
- **Chcete-li editovat body pomocí grafického editoru**, klikněte na bod na grafu a přetáhněte ho na požadované místo. (POZNÁMKA: Data bodu v tabulkovém editoru se aktualizují na novou polohu.) Chcete-li v grafickém editoru provést další úlohy, klikněte v grafu pravým tlačítkem myši a z menu klávesových zkratek zvolte příslušnou úlohu.

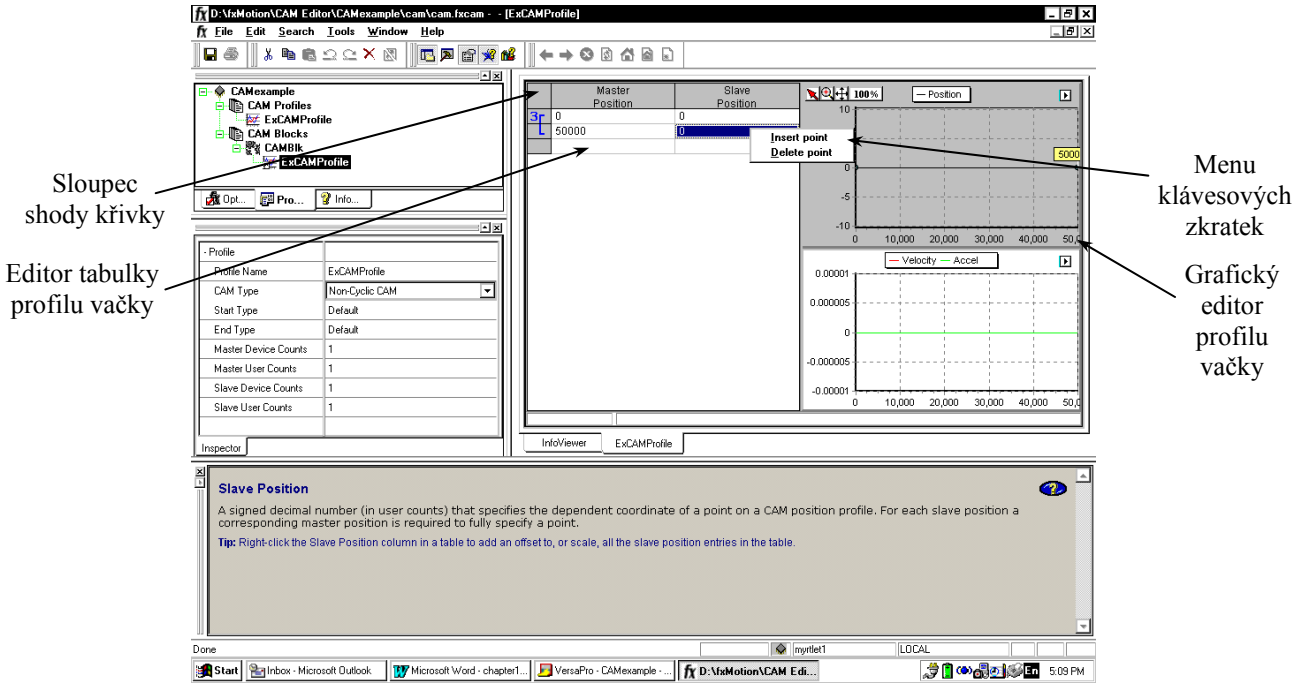
Dalším krokem je editování koncového bodu (dolního bodu v tabulce) pro master a slave. V tabulkovém editoru klikněte na master sloupec koncového bodu a zapište hodnotu 50 000; pak klikněte na slave sloupec koncového bodu a zapište hodnotu 0. (POZNÁMKA: Když se v tabulkovém editoru přidá nebo změní bod, graf v grafickém editoru se odpovídajícím způsobem aktualizuje.)

Dále do editoru tabulky vložte další bod. Klikněte pravým tlačítkem myši na master sloupec koncového bodu a z menu klávesových zkratek zvolte Insert Point (ukázáno na následujícím obrázku). Nad řádek koncového bodu se přidá nový řádek, který specifikuje nový bod s master a slave hodnotami, implicitně uprostřed (25 000 a 0) mezi hodnotami stávajících sousedních bodů (nahoře a dole). Změňte hodnoty pro tento bod na 47 500 pro master a 11 000 pro slave. Chcete-li změnit hodnotu bodu, klikněte na ní, zapište nové číslo a pak buď stiskněte tlačítko Enter nebo klikněte mimo tabulku.

Chcete-li změnit řád shody křivky, klikněte na sloupec Curve-Fit (shoda křivky) a pak v okně Property inspector (inspektor vlastností) zvolte řád shody křivky. Profil je také možno rozdělit na více částí nebo více částí je možno sloučit do jednoho kliknutím pravého tlačítka myši na zobrazení shody křivky a zvolením Curve-Fit z menu klávesových zkratek.

Poznámka

Pokud profil vačky obsahuje sektory druhého a třetího řádu, je omezený na 400 bodů. Pokud profil vačky obsahuje pouze sektory prvního řádu, je omezený na 5000 bodů.



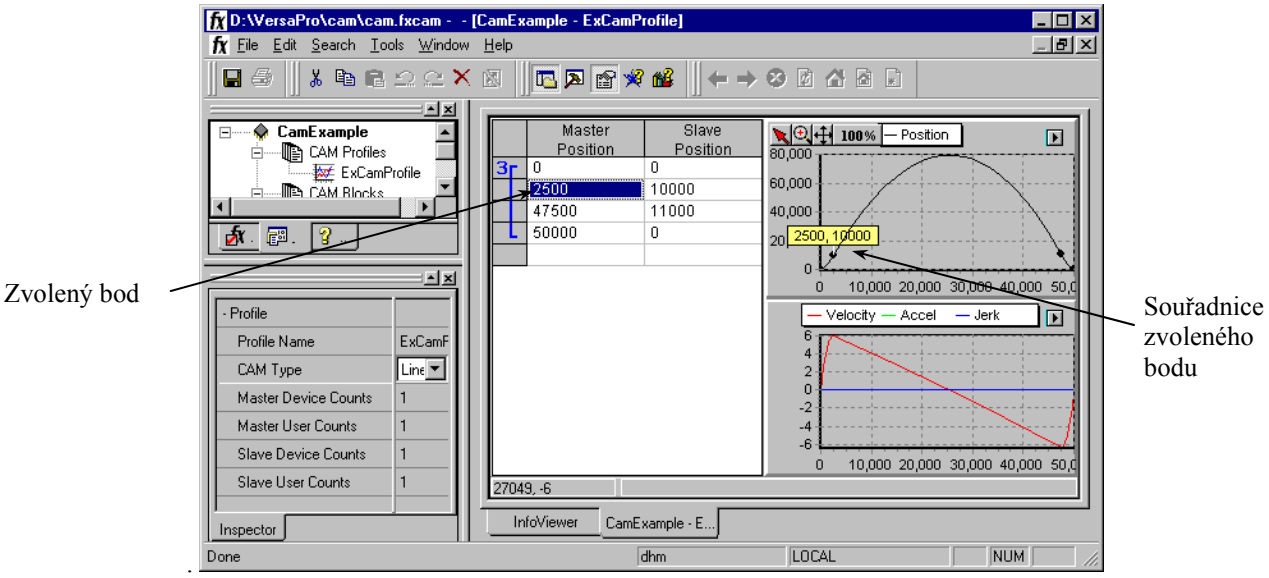
Obr. 16-21. Vložení bodu v okně editoru profilu

Protože koncový bod slave polohy má stejnou hodnotu (0) jako počáteční bod slave polohy, tato vačka splňuje požadavky na lineární cyklickou vačku. (Pokud budete potřebovat, více informací o jiných typech vačky najdete v části 2.) Všimněte si, že editor vačky má několik "chytrých" editovacích polí, která POUZE zobrazí volby, které jsou platné pro daný soubor dat. Protože například požadavek pro lineární cyklickou vačku je, aby počáteční bod slave polohy a koncový bod slave polohy byly stejné, editor dovolí zvolit lineární cyklickou vačku, pouze pokud toto kritérium bude splněné.

Dále do profilu vložte nový bod a pak tento bod editujte. Bod je možno editovat buď v tabulce profilu nebo graficky v diagramu. Vložte bod, jak je znázorněno výše na Obr. 16-22 a Obr. 16-23 kliknutím pravého tlačítka myši na bod pod polohou vložení a z menu zvolte Insert Point. Pak změňte výchozí hodnoty na 2 500 pro master a 10 000 pro slave.

	Master Position	Slave Position
3	0	0
	2500	10000
	47500	11000
	50000	0

Obr. 16-22. Tabulka dat profilu vačky



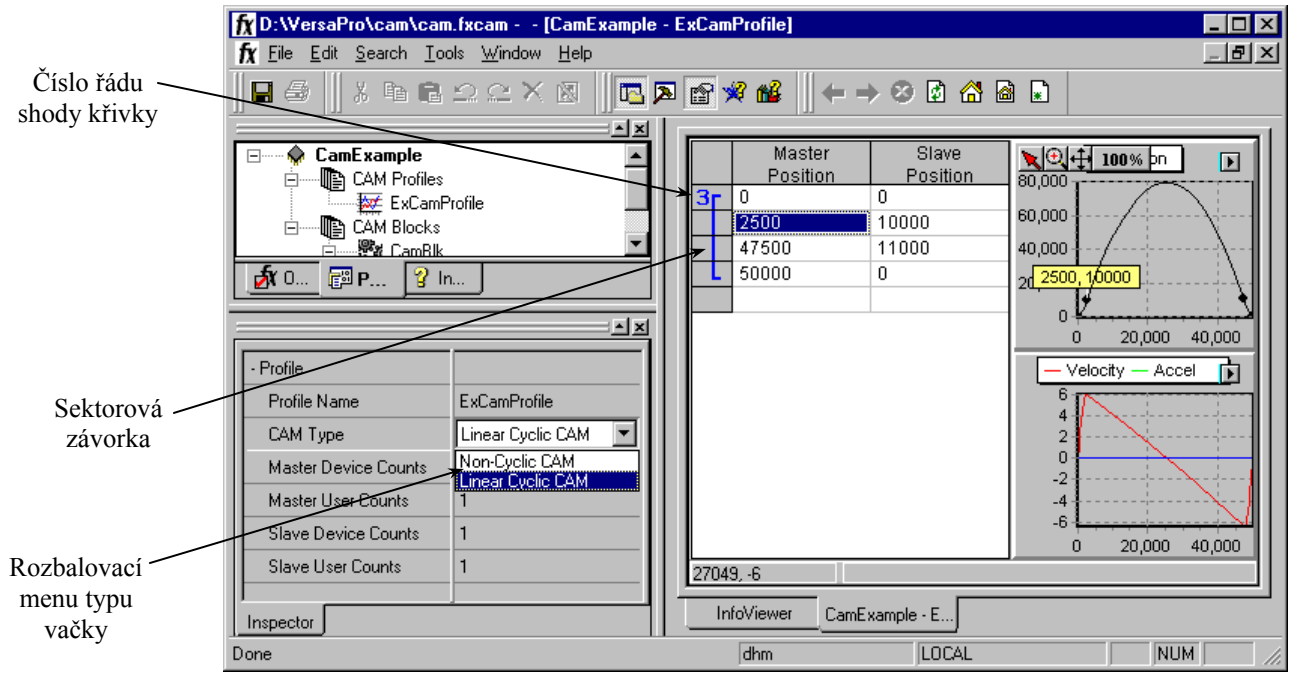
Obr. 16-23. Příklad editoru vačky

V editoru je mnoho dalších jiných funkcí. Mezi ně patří možnost definovat další sektory, u kterých každý bude mít jinou metodu shody křivky. Tyto funkce editoru jsou probírané v on-line nápovědě programovacího softwaru. Další informace si vyhledejte v tomto zdroji.

Krok 6: Zadání typu vačky

Jak bylo uvedeno dříve, v tomto příkladu je vačka lineární cyklická. Použijte následující postup:

- Na záložce Project v okně Navigátor klikněte pravým tlačítkem myši na profil vačky. Objeví se klávesová zkratka menu.
- Z menu klávesových zkratek zvolte Properties. Otevře se Inspektor zobrazující vlastnosti profilu vačky.
- V okně Inspektoru klikněte na šipku v poli CAM Type. Objeví se rozbalovací seznam typů vačky.
- Ze seznamu zvolte ‘Linear Cyclic CAM’ (Obr. 16-24).

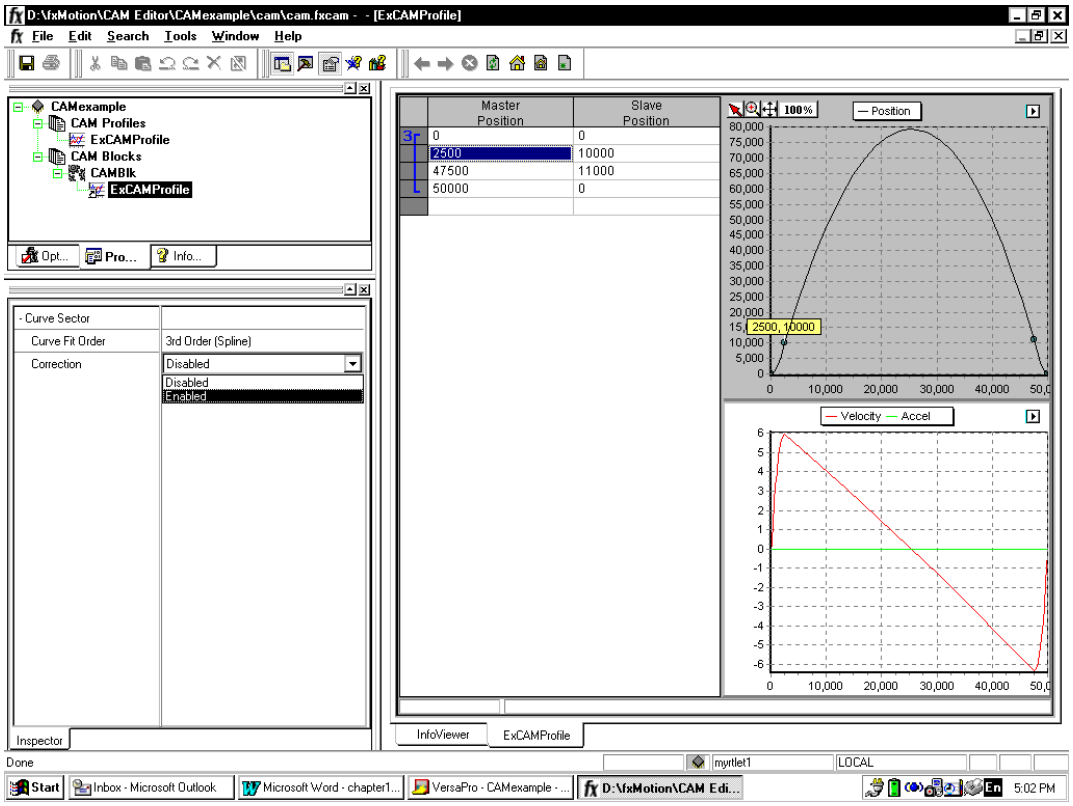


Obr. 16-24. Volba typu vačky v editoru vačky

Krok 7: Zadání vlastnosti Korekce

Poslední položka, kterou je nutno zadat v tomto příkladu, je stav korekcí. Vlastnost Korekce určuje, jestli pohybový modul pro konkrétní sektor povolí on-line korekci. Sektor je oblast profilu vačky definovaná posledními dvěma sousedními uživatelem definovanými body. Sektor zahrnuje uživatelem definované body, křivku, která je propojuje, a také, ale ne včetně, první bod definovaný pro následující sousední sektor. Body zahrnuté v daném sektoru jsou označené sektorovou závorkou, ukázanou na obrázku výše. Každý sektor má přiřazené číslo řádu shody křivky, také ukázané na obrázku výše. Segmenty profilu mezi uživatelem definovanými body jsou definované polynomy zadaného řádu shody křivky. Pro interpolaci mezi každým párem sousedních uživatelem definovaných bodů se používá jednoznačný polynom. I když koeficienty aktuálního polynomu mohou být pro každý segment odlišné, řád shody křivky je v celém sektoru stejný. Sektor je v tabulce profilu vačky označený jako čára spojující hodnoty uživatelem definované master polohy zahrnuté v sektoru. Z počátku jsou všechny body definované v sektoru zahrnuté v jednom sektoru. Tento jeden výchozí sektor je možno podle potřeby dále rozdělit tak, aby se profil vačky vyhladil. Když bude vlastnost Korekce povolená, pohybový modul bude hlásit výstrahu, pokud se překročí mez rychlosti. Když vlastnost Korekce bude nastavená na Zakázáno, pohybový modul bude hlásit chybu pro tato překročení a slave osu zastaví.

U tohoto příkladu korekce musí být povolená. Chcete-li korekci povolit, zvolte kliknutím sektor z tabulky profilu vačky. To způsobí, že okno Inspektoru zobrazí vlastnosti sektoru a umožní jejich editování. Zvolte rozbalovací pole Correction a vyberte Enabled (Obr. 16-25).



Obr. 16-25. Povolení korekcí v editoru vačky

Krok 8: Uložení profilu vačky

Nyní je definovaný jednoduchý profil vačky. Aby se bloky/profily vačky uložily, v hlavním menu zvolte položku File a pak zvolte podmenu Save Project. Také můžete zvolit Exit, což vyvolá automatické uložení. Editor vačky má mnoho dalších vlastností a funkcí. Podrobný popis těchto funkcí najdete v dokumentaci k on-line nápovědě.

Krok 9: Vygenerování pohybového programu a programu lokální logiky

Dalšími položkami, které je nutno vygenerovat, jsou pohybový program a program lokální logiky, které budou pracovat s tímto profilem vačky. U tohoto příkladu logika musí pracovat s DSM314, které řídí dvě osy. Osa #1 bude slave a osa #2 bude master. Proto zde budou dva pohybové programy. Program pro osu 1, pro slave, provede určitou základní inicializaci, načtení slave počátečního bodu pro daný profil vačky a pak vykoná povel CAM. Program pro osu 2, pro master zdroj, je jednoduchý program, který provede inicializaci a pak počká, až slave bude připravený. Pak vykoná sérii pohybů. Program zastaví v bodech popsaných v master vačce tak, aby bylo snadné ověřit, jestli slave osa správně vykonává profil vačky. Tento příklad také vyžaduje program lokání logiky. V tomto příkladu program lokální logiky má úlohu dohlížení nad pohybovými programy slave vačky a master vačky. Proto lokální logika tyto dva programy synchronizuje.

Další podrobnosti k těmto funkcím najdete v příslušných kapitolách tohoto manuálu. Pohybový program a program lokální logiky pro tento příklad budou vypadat následovně:

// Pohybový program pro příklad bloku vačky

// Osa slave

```
        Program 1 AXIS1
        VELOC 10000                          // Nastavení rychlosti
        ACCEL 10000                          // Nastavení zrychlení
100:    WAIT CTL01                           // Čekání, až LL řekne, že master je
                                             připravený
110:    CAM-LOAD "ExCamProfile", P006, ABS   // Načtení registru parametrů se slave
                                             bodem odpovídajícím aktuální master
                                             poloze
120:    PMOVE P006, ABS, LINEAR              // Pohyb slave osy do polohy odpovídající
                                             počátku tabulky
130:    CAM "ExCamProfile", 50000, ABS       // Vykonání příkazu CAM
140:    PMOVE 0,ABS,S-CURVE                  // Pohyb zpět na nulu
150:
        ENDPROG
```

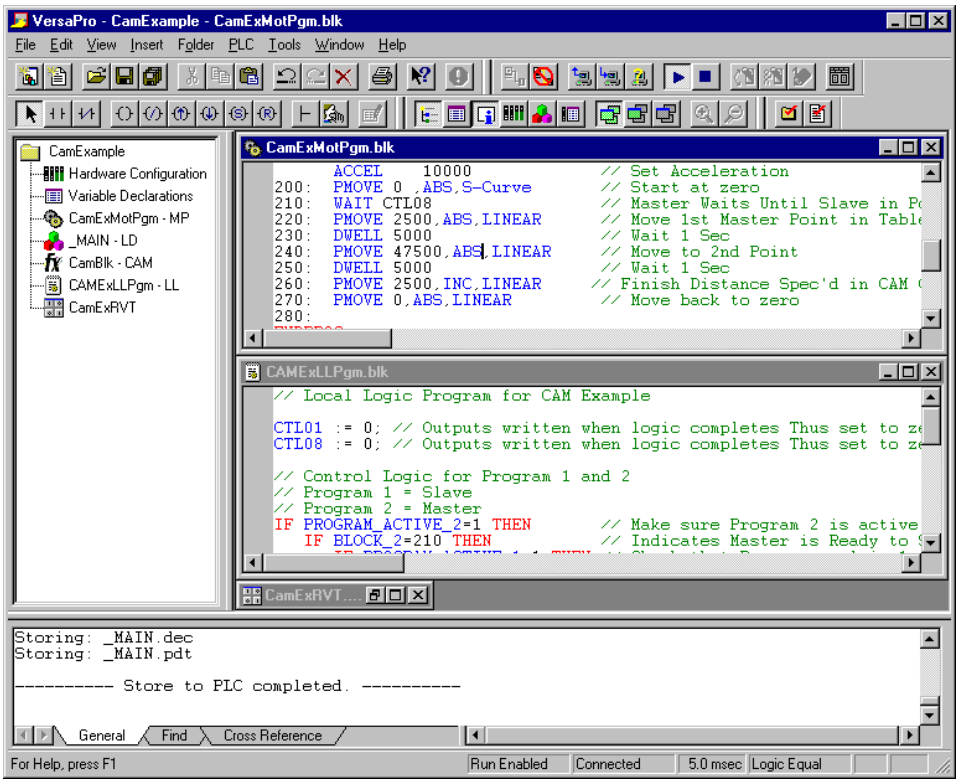
// /Program master osy

```
Program 2 AXIS2
        VELOC 10000                  // Nastavení rychlosti
        ACCEL 10000                  // Nastavení zrychlení
200:    PMOVE 0 ,ABS,S-Curve         // Start na nule
210:    WAIT CTL08                   // Master čeká, až slave bude v poloze
220:    PMOVE 2500,ABS,LINEAR        // Pohyb do 1. master bodu v tabulce
230:    DWELL 5000                   // Čekání 5 sekund
240:    PMOVE 47500,ABS,LINEAR       // Pohyb do 2. bodu
250:    DWELL 5000                   // Čekání 5 sekund
260:    PMOVE 2500,INC,LINEAR        // Dokončení vzdálenosti zadané v povelu CAM, 1,
                                     vačka kompletní
270:    PMOVE 0,ABS,LINEAR           // Pohyb zpět na nulu
280:
        ENDPROG
```

// Příklad programu lokální logiky nebo vačky

```
CTL01 := 0;                          // Výstupy zapsané, když logika dokončí
                                     inicializaci na nulu, aby se umožnilo přepnutí na
                                     jedničku
CTL08 := 0;                          // Výstupy zapsané, když logika dokončí
                                     inicializaci na nulu, aby se umožnilo přepnutí na
                                     jedničku
// Řídicí logiky pro programy 1 a 2
// Program 1 = Slave
// Program 2 = Master
IF PROGRAM_ACTIVE_2=1 THEN           // Přesvědčte se, že program 2 je aktivní
   IF BLOCK_2=210 THEN               // Udává, že master je připravený spustit vačku
      IF PROGRAM_ACTIVE_1=1 THEN     // Přesvědčte se, že program osy 1 je aktivní
         IF BLOCK_1=100 THEN         // Blok udává, že slave je připravený pro sekvenci
                                     CAM-Load
            CTL01:=1;                // Signál pro slave k vykonání sekvence načtení
                                     vačky
         END_IF;
         IF BLOCK_1=130 THEN         // Blok udává, že slave dokončil inicializaci a
                                     sekvenci načtení vačky
            CTL08:=1;                // Signál pro master a slave, že obě osy jsou
                                     připravené spustit sekvenci vačky
         END_IF;
      END_IF;
   END_IF;
END_IF;
// Konec řídicí logiky programu 1 a 2
```

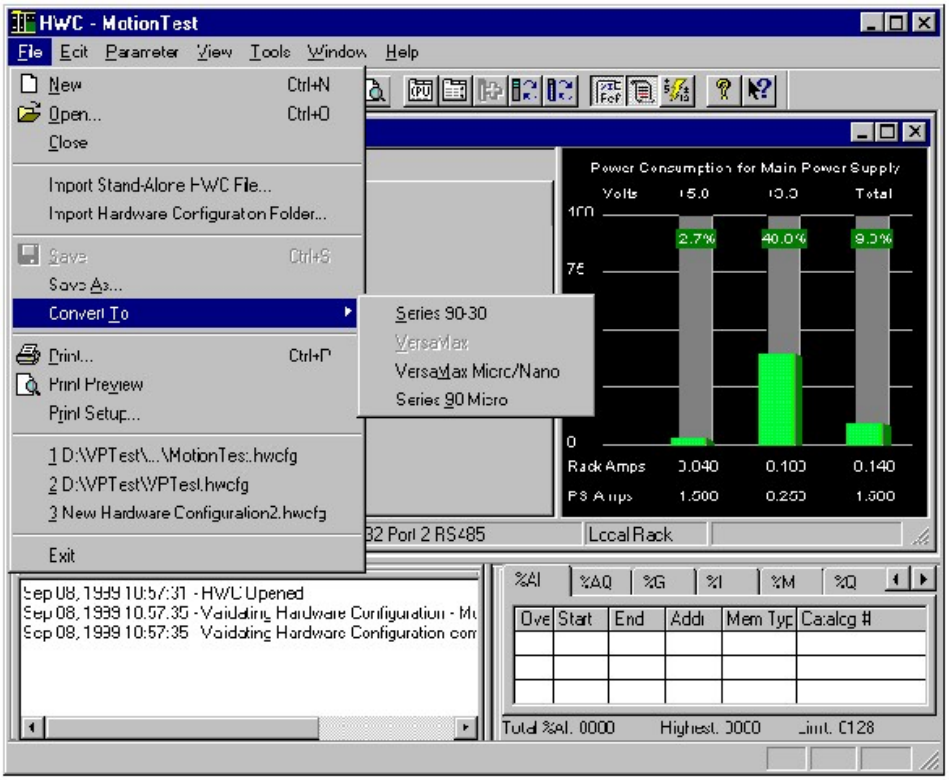
Po dokončení zápisu programu bude výsledná obrazovka VersaPro vypadat přibližně jako na obrázku níže (Obr. 16-26).



Obr. 16-26. Obrazovky VersaPro s příkladem vačky

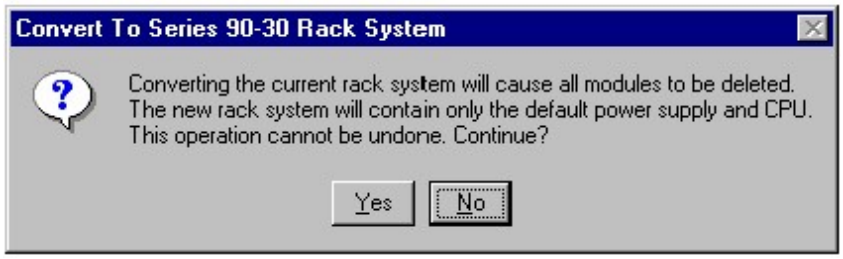
Krok 10: Nastavení hardwarové konfigurace ve VersaPro

Jakmile bude dokončená úspěšná kontrola syntaxe programu lokální logiky a pohybového programu, je nutno nastavit hardwarovou konfiguraci, která umožní načíst program do modulu DSM314. Sekvence kroků v tomto příkladu není pro většinu instalací typická. Většina uživatelů nejdříve nastaví hardwarovou konfiguraci a pak vytvoří programové příkazy. Příklad se však snaží znázornit funkci vačky. Proto pořadí je obrácené, aby se lépe znázornila vazba mezi hardwarovou konfigurací a názvem bloku vačky v hardwarové konfiguraci DSM314. Dalším krokem v tomto příkladu je spuštění hardwarové konfigurace. Je několik možností, jak to provést. Zde jsou uvedené dva způsoby (další podrobnosti najdete v dokumentaci k VersaPro). Chcete-li provést tyto kroky, buď (1) proveďte volbu View z hlavního menu a pak zvolte Hardware Configuration nebo (2) klikněte na ikonu Hardwarové konfigurace na liště nástrojů. Tím se spustí nástroj pro konfiguraci hardwaru. Výchozí obrazovky konfigurace hardwaru jsou pro výrobek VersaMax. Proto první operací musí být volba Series 90-30. K tomu účelu na liště menu zvolte File, z menu File zvolte Convert To a pak Series 90-30 z menu Convert To, jak je znázorněno na Obr. 16-27.



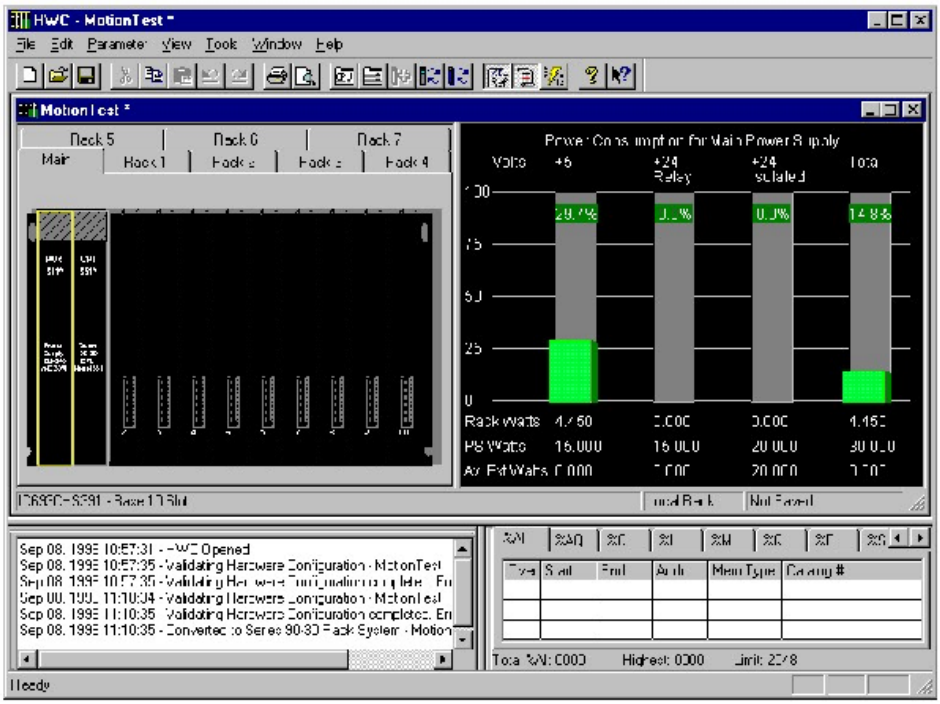
Obr. 16-27. Volba konfigurace sestavy hardwaru

Objeví se dialogové okno, které vás upozorní, že informace budou smazané. Adresář vytvořený pro tento příklad je nový; proto nedoje ke ztrátě žádných informací. Pokud nevytváříte nový adresář, dejte pozor na to, že vykonáním této operace dojde ke ztrátě konfiguračních informací. Doporučuje se, abyste pro tento příklad použili nový prázdný adresář. V dialogovém okně odpovězte Yes, jak je znázorněno na Obr. 16-28.



Obr. 16-28. Dialogové okno převodu konfigurace sestavy hardwaru

Jakmile bude tato operace hotová, budete mít prázdnou sestavu 90-30, kterou je nutno nakonfigurovat. Výsledná obrazovka pro konfiguraci hardwaru bude vypadat asi jako na Obr. 16-29.

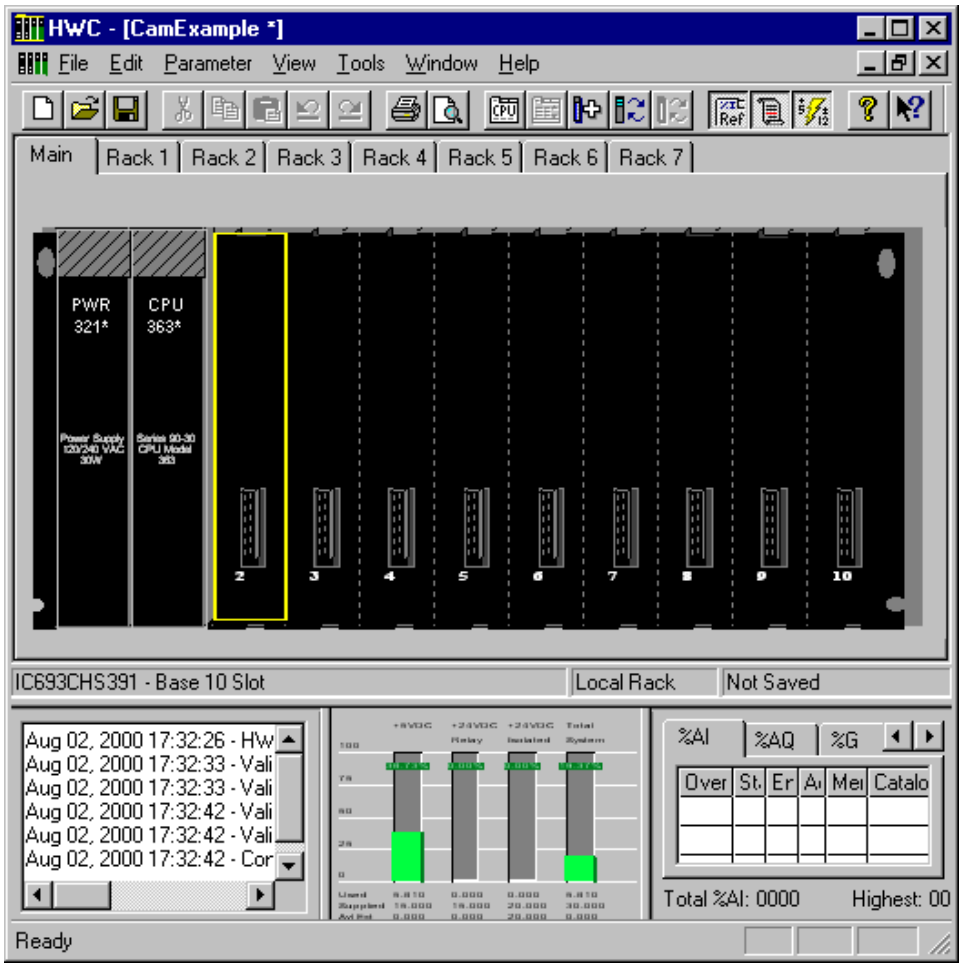


Obr. 16-29. Hardwarová konfigurace sestavy 90-30 s CPU

Dále je nutno zvolit napájecí zdroj a CPU, které jsou vhodné pro vaší instalaci. Všimněte si, že lokální logika vyžaduje CPU s firmwarem verze 10.00 nebo vyšší. Výchozí CPU, "CPU351", nepodporuje firmware verze 10.00; proto je nutno CPU změnit například na model CPU363, který verzi 10.00 podporuje. Kroky jsou následující:

- Klikněte pravým tlačítkem myši na CPU na obrazovce hardwarové konfigurace a v menu klávesových zkratek zvolte Replace CPU. Objeví se dialogové okno pro volbu modulu.
- Klikněte na položku IC693CPU363, pak klikněte na tlačítko OK.
- Zobrazí se dvě okna s hlášením upozorňující, že (1) záměnu CPU nelze provést a (2) že SNP a ostatní parametry se přenášejí vždy, atd. Pokračujte tím, že u obou hlášení kliknete na Yes.
- Objeví se konfigurační okno CPU363. Proveďte všechny požadované změny a pak konfigurační okno zavřete.

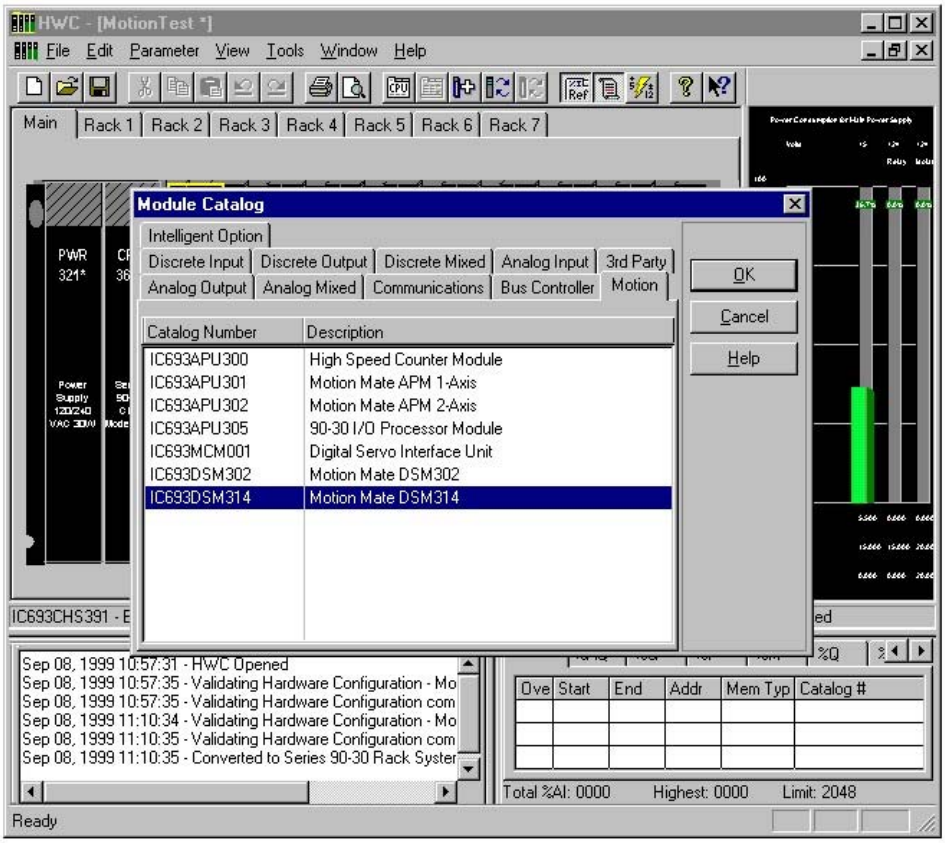
Pokud budete potřebovat, podívejte se do dokumentace k VersaPro nebo do on-line nápovědy, kde najdete více podrobností. Výsledné zobrazení bude vypadat jako na Obr. 16-30.



Obr. 16-30. Hardwarová konfigurace sestavy 90-30 s CPU363

Nyní je nutné do sestavy přidat DSM314. Chcete-li provést tento krok, zvolte pozici v sestavě, do které se má DSM314 nainstalovat. V tomto příkladu se DSM314 nainstaluje do pozice číslo 2. Přidání modulů do pozice sestavy je možno provést několika způsoby. Zde jsou uvedené dva způsoby přidání modulu (další podrobnosti a postupy najdete v dokumentaci k VersaPro).

- Můžete buď (1) dvakrát kliknout na požadovanou pozici, v tomto případě pozici číslo 2, nebo (2) kliknout pravou myší na požadovanou pozici (číslo 2) a pak zvolit Add Module.
- Pak se otevře dialogové okno, které umožní zvolit typ modulu. V dialogovém okně zvolte záložku "Motion".
- Ze seznamu zvolte modul DSM314. Výsledné zobrazení bude vypadat jako na Obr. 16-31.

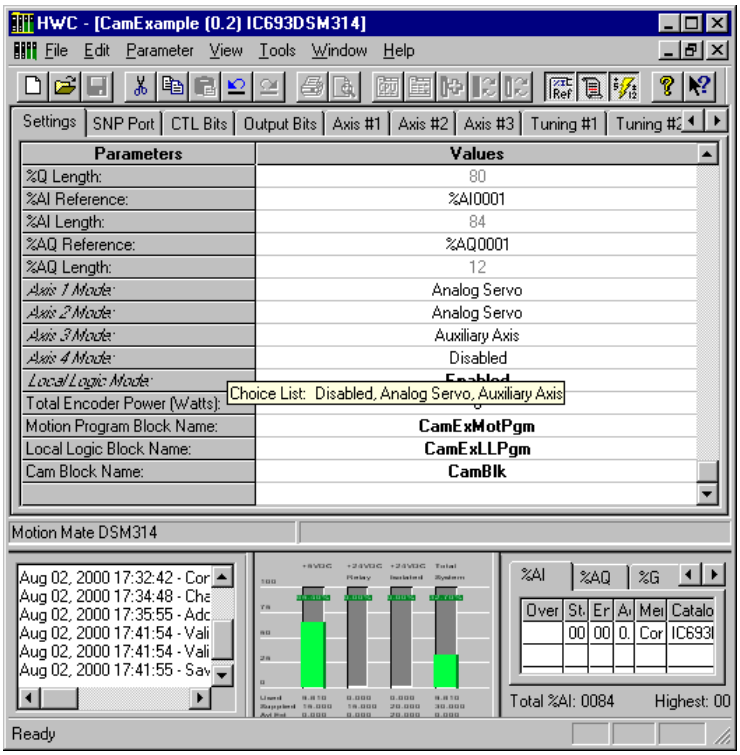


Obr. 16-31. Volba hardwarové konfigurace sestavy 90-30 s DSM314

Tato operace přidá do sestavy DSM314 a vyvolá obrazovky pro konfiguraci DSM314. To umožní přizpůsobit DSM314 vaší konkrétní aplikaci. Podrobnosti ohledně konfiguračních nastavení DSM314 najdete v kapitole 4. V tomto příkladu se změní název bloku lokální logiky, název bloku pohybového programu, název bloku vačky a režim lokální logiky. Tato pole se nacházejí na záložce "Settings". Režim nástroje lokální logiky musí být Povolený. V poli “Local Logic Block Name” zapište název vzorového programu “CamExLLPgm". V poli CAM Block Name zapište “CamBlk” a v poli Motion Program Block Name zapište “CamExMotPgm”.

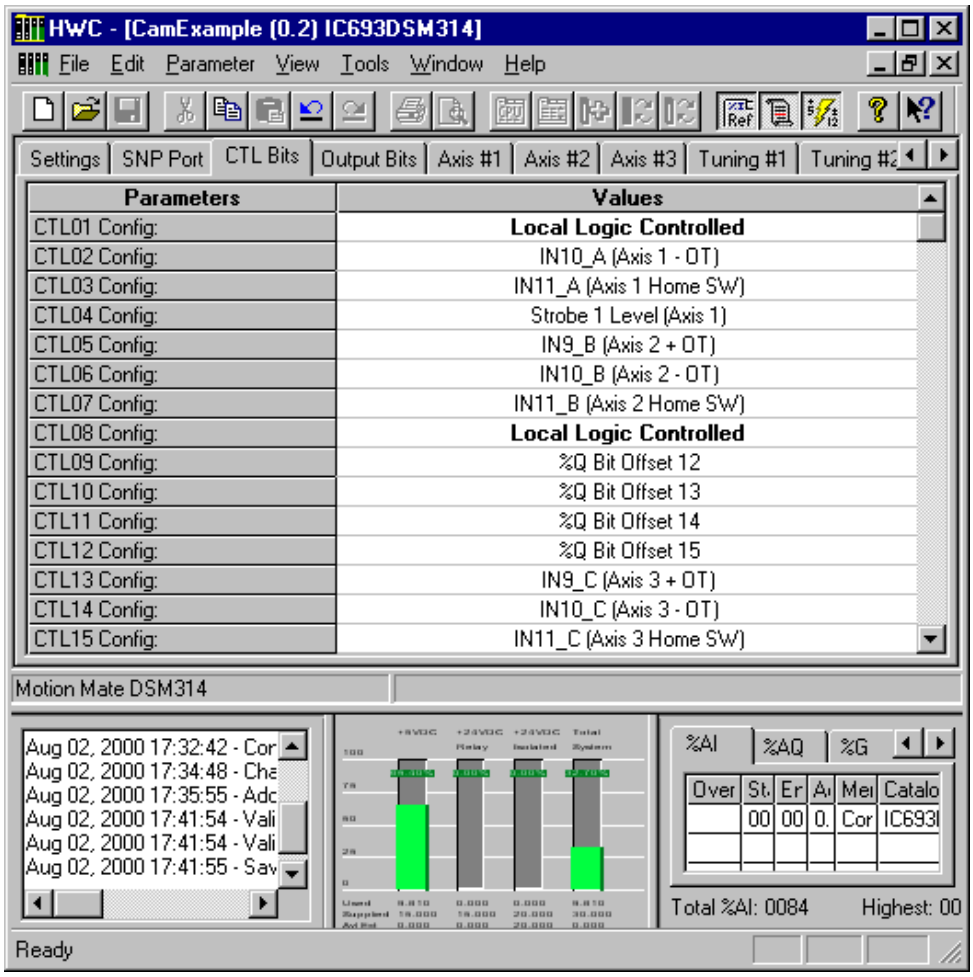
Poznámka

Tento způsob navázání DSM314 s bloky programu a bloky vačky byl vybraný proto, aby bylo možno snadno zadat více modulů DSM314, které používají stejné bloky. Tento příklad používá pouze jeden DSM314. Pokud budete mít více DSM314, které budou potřebovat mít spuštěný stejný program lokální logiky, pohybové programy nebo bloky vačky, v konfiguraci pro každé DSM314 zadejte, že má vykonat tento program. To umožní, aby více modulů DSM314 mělo jeden zdrojový soubor. Všimněte si, že toto nezabrání tomu, aby DSM314 vykonávala různé programy. Proto DSM314 bude vykonávat soubory (program vačky, program lokální logiky a pohybový program) označený v konfiguraci. Výsledná obrazovka pro konfiguraci hardwaru bude vypadat jako na Obr. 16-32.



Obr. 16-32. Záložka Settings hardwarové konfigurace 90-30 s DSM314

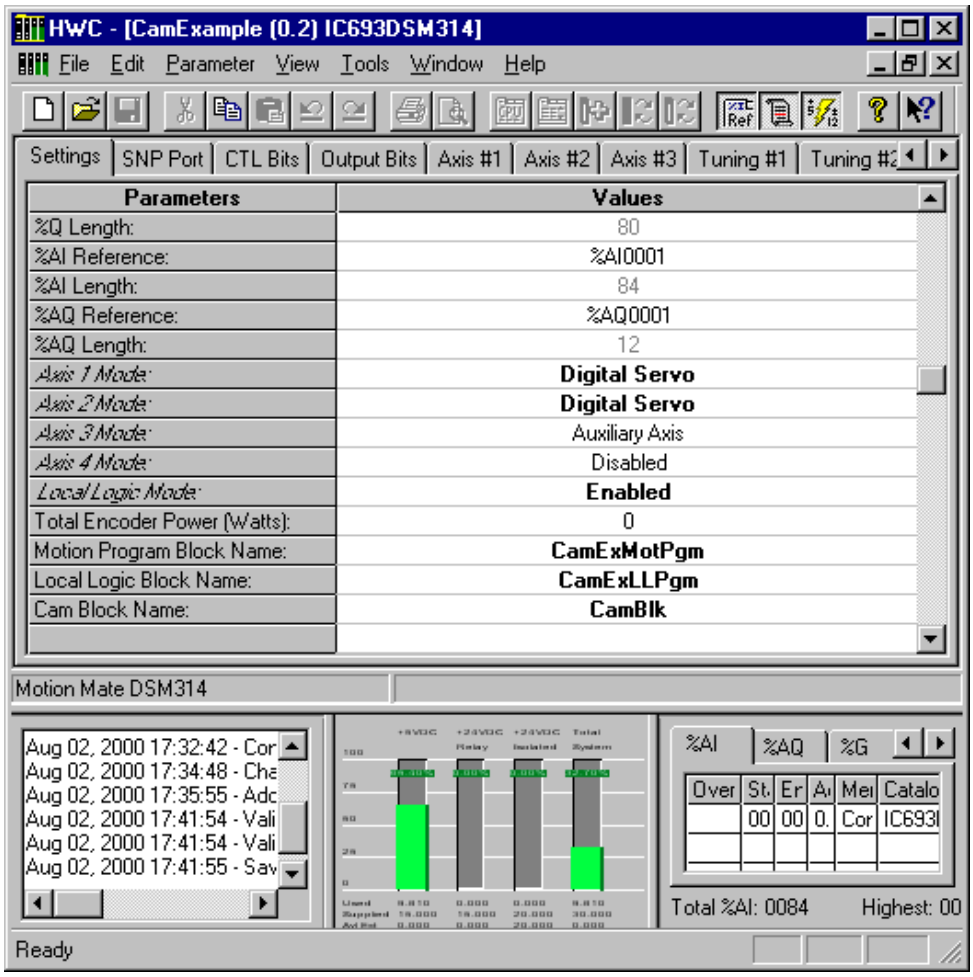
V tomto příkladu program lokální logiky bude řídit CTL01 a CTL08. Protože CTL01 a CTL08 se používají k signalizování pohybových programů, je nutno tyto CTL bity nakonfigurovat tak, aby byly řízené lokální logikou. K tomu účelu otevřete záložku CTL Bits v hardwarové konfiguraci VersaPro. Vyberte “CTL01 Config” a zvolte Local_Logic_Controlled. Zopakujte postup pro CTL08. Výsledná obrazovka pro konfiguraci hardwaru bude vypadat jako na Obr. 16-33.



Obr. 16-33. Volba hardwarové konfigurace 90-30 se sestavou DSM

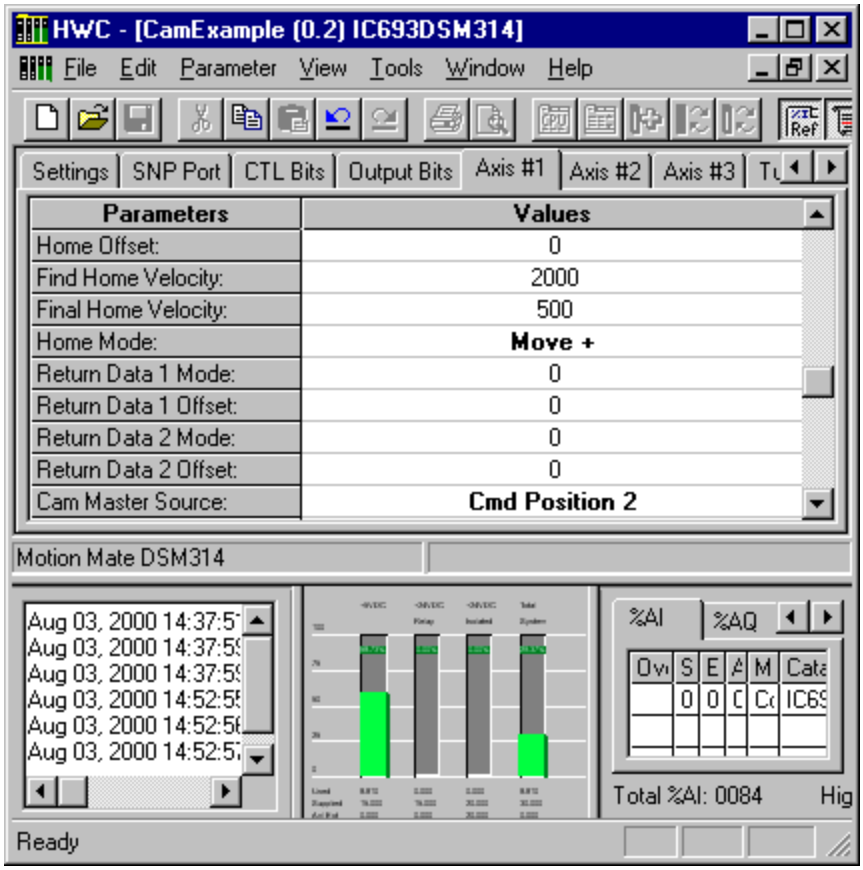
Protože tento příklad používá digitální servo Beta 0.5, režim osy 1 a osy 2 je nutno nastavit na digitální servo.

Výsledná obrazovka pro konfiguraci hardwaru bude vypadat jako na Obr. 16-34.



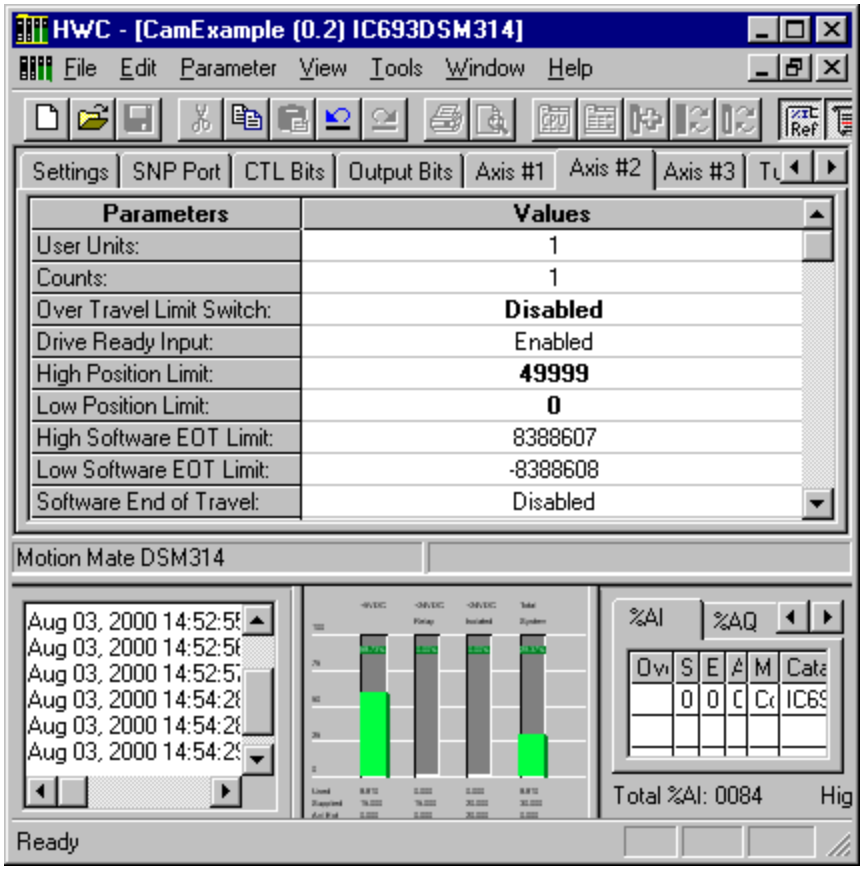
Obr. 16-34. Záložka nastavení hardwarové konfigurace DSM314

Také je nutno ose č.1 sdělit, že bude používat zadanou polohou osy č.2 jako svůj master zdroj vačky. K tomu účelu v hardwarové konfiguraci zvolte záložku Axis #1. Přejděte na zápisové pole CAM Master Source. Z rozbalovacího seznamu zvolte Cmd Position 2. Tím se nakonfiguruje, že osa č.1 bude používat zadanou polohu osy č.2 jako svůj master zdroj vačky (Obr. 16-35). Když už budete na této záložce, změňte Home Mode: na Move + a OverTravel Switch změňte na Disabled.



Obr. 16-35. Volba master zdroje pro slave vačku

Také je nutno ose č.2 sdělit body přetočení referenční polohy master osy. K tomu účelu v hardwarové konfiguraci zvolte záložku Axis #2. Do pole High Position Limit zapište 49 999 a do pole Low Position Limit zapište 0 (Obr. 16-36). Všimněte si, že protože toto je cyklická vačka, horní mez master zdroje na základě definice musí být o jedničku menší než poslední bod v datové tabulce masteru. V tomto příkladu to je bod 50 000. Proto se horní mez bude rovnat 49 999. Jedním ze způsobů, jak čelit tomuto principu, je si představit cyklickou master vačku jako souvislý kruhový pásek, kde první bod pásku je stejný jako poslední bod pásku. Proto v tomto příkladu 50 000 je stejný bod jako nula. Když už budete na této záložce, změňte Home Mode: na Move + a OverTravel Switch změňte na Disabled.

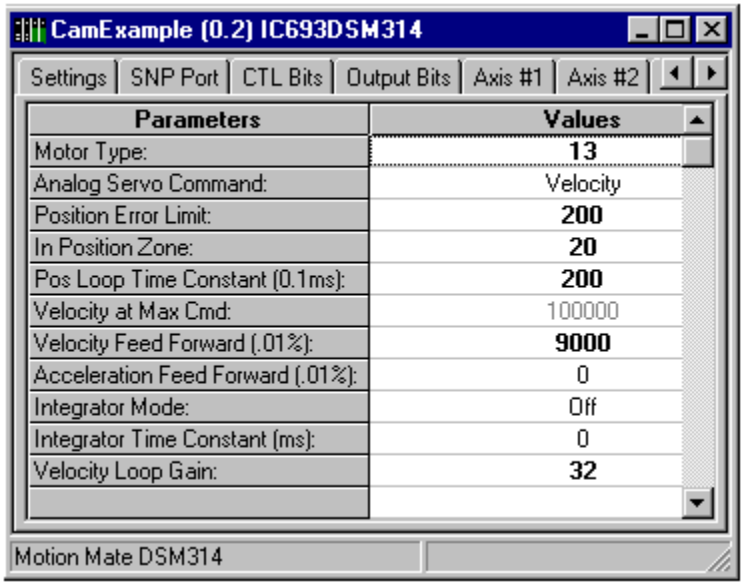


Obr. 16-36. Nastavení měřítka master osy vačky

K dokončení konfigurace je nutno přejít na záložku Tuning#1 a Tuning #2 a zapsat následující data.

- **Typ motoru:** 13
- **Mez polohové odchylky:** 200 (volitelné; další informace najdete v konfiguračních informacích)
- **Zóna dosažení polohy:** 20 (volitelné; další informace najdete v konfiguračních informacích)
- **Časová konstanta polohové smyčky:** 200 (Poznámka: Vychází z aplikace/mechaniky, viz kapitola 4 a Dodatek D)
- **Posuv rychlostí:** 9000 (Poznámka: Vychází z aplikace/mechaniky, viz kapitola 4 a Dodatek D)
- **Zisk rychlostní smyčky:** 32 (Poznámka: vychází ze setrvačnosti připojené k motoru. Typický případ Beta Demo má připojený indikační kotouč, který představuje tuto setrvačnost přibližně jako Beta 0.5)

Výsledné zobrazení vypadá přibližně jako na Obr. 16-37.



Obr. 16-37. Záložka hardwarové konfigurace Tuning#1

Záložku Tuning pro osu #1 je také nutno nastavit tak, jak je uvedeno pro osu č.1 na obrázku 2-26.

Nyní je možno uzavřít dialogové okno konfigurace modulu a práci uložit. Chcete-li svou práci uložit, z hlavního menu zvolte File a pak z menu souboru zvolte Save All.

Tím se dokončí změny konfigurace potřebné v tomto příkladu. Uzavřete nástroj pro hardwarovou konfiguraci a adresář uložte tak, že z hlavního menu zvolíte File a pak z menu souboru zvolíte Save All.

Propojení mezi ukázkovým blokem vačky, pohybovým programem a modulem DSM314 je nyní kompletní. Vytvořte libovolný požadovaný program žebříkové logiky PLC a pak u programů proveďte Check All (Zkontrolovat všechno) a načtěte je do PLC. Další informace týkající se operace načtení jsou uvedené v manuálu k VersaPro GFK-1670 nebo v on-line nápovědě VersaPro 1.5.

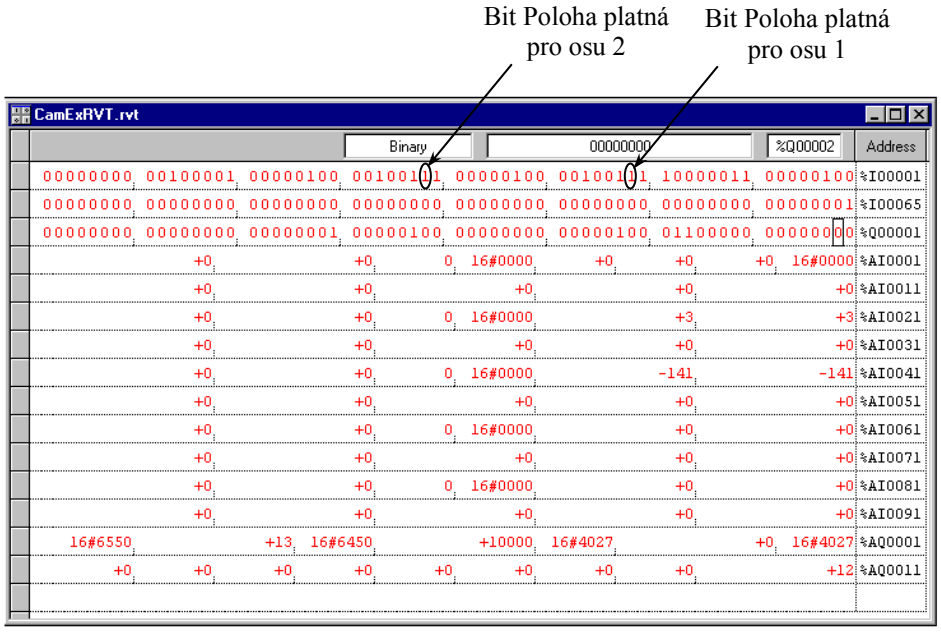
Krok 11: Vykonání (vyzkoušení) pohybového programu vačky

Výstraha

Než budete testovat aplikaci na skutečném stroji, nejdříve si ověřte, že to je bezpečné. To zahrnuje zajištění, že všechna zařízení jsou bezpečně namontovaná, veškerá bezpečnostní zařízení jsou nainstalovaná a funkční a že pracovníci v okolí jsou vyrozuměni. Pokud neprovedete všechna opatření související s bezpečností, může dojít ke zranění obsluhy a k poškození zařízení.

Jakmile se dokončí operace načítání, modul je připravený k vykonání bloků vačky, pohybových programů a programu lokální logiky. Použijte následující postup:

1. PLC nastavte do režimu běhu.
2. Povolte servopohony. Povolení osy č.1 proveďte nastavením bitu %Q offset 18 do jedničky. Povolení osy č.2 proveďte nastavením bitu %Q offset 34 do jedničky. V závislosti na aktuálním chybovém stavu modulu můžete také spustit rutinu pro vynulování chyby nastavením bitu %Q offset 0 do jedničky.
3. V obou osách vykonejte rutinu nájezdu do výchozího bodu nastavením bitu %Q offset 19 (nalezení výchozí poloha osy č.1) a bitu %Q offset 35 (nalezení výchozí polohy osy č.2). Nyní obě osy vykonají cyklus nalezení výchozí polohy. Počkejte, až se operace v obou osách dokončí a bity %I Poloha platná se nastaví do jedničky. Bit %I Poloha platná pro osu 1 je bit %I offset 17 (18. bit %I) a pro osu 2 to je bit %I offset 33 (34. bit %I). Výsledné zobrazení bude vypadat jako na Obr. 16-38.



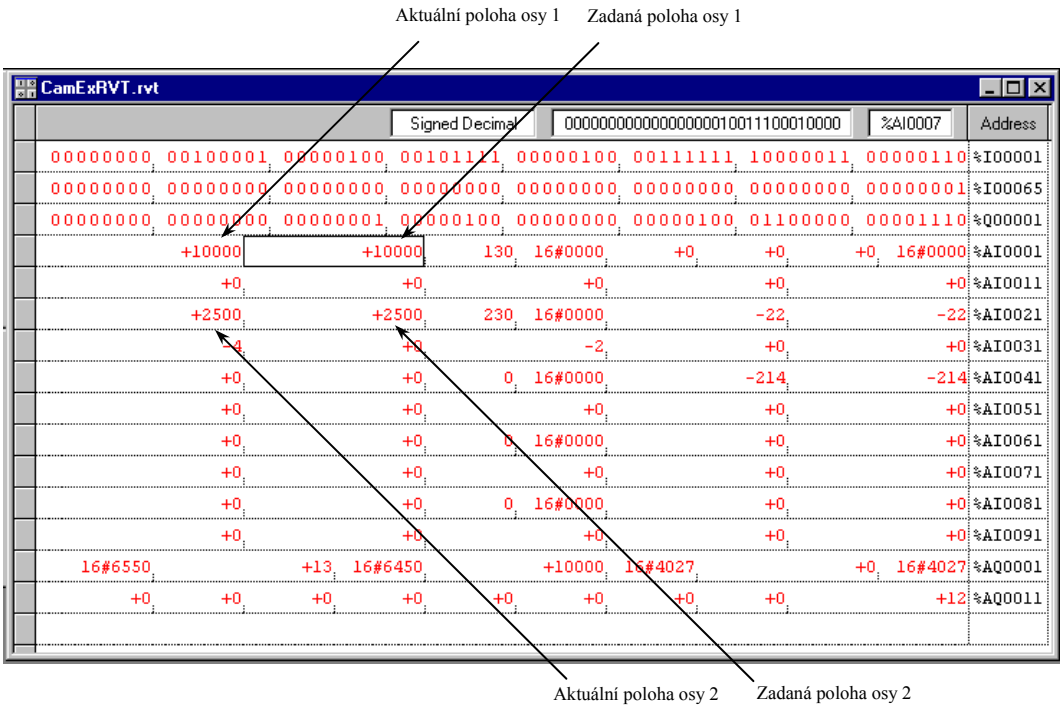
Obr. 16-38. Obrazovka RVTExample

4. Povolte lokální logiky tak, že z PLC nastavíte bit %Q offset 1 do jedničky. Pokud se nebudou vyskytovat žádné chyby, pohybové programy můžete vykonat.

5. Program 1 vykonejte nastavením bitu %Q offset 2 do jedničky. Motor připojený k ose č.1 by měl začít vykonávat pohybový program č.1.

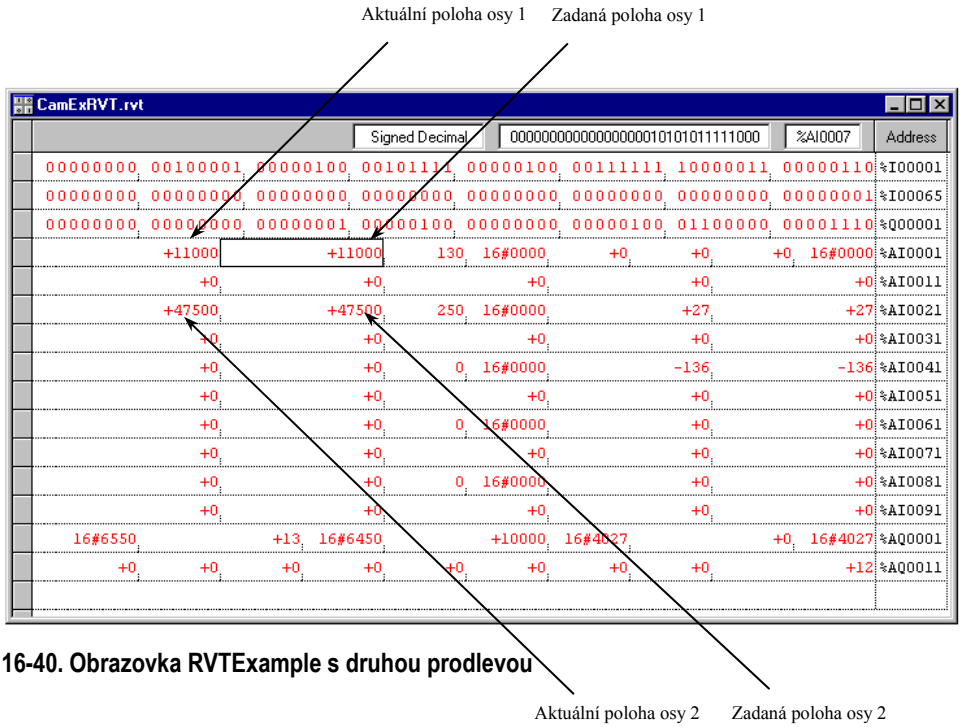
6. Program 2 vykonejte nastavením bitu %Q offset 3 do jedničky. Motor připojený k ose č.2 by měl začít vykonávat pohybový program č.2.

7. Motory budou vykonávat příkazy, dokud nedosáhnou prvního příkazu DWELL, kde můžete vizuálně ověřit, že správně sledovaly profil vačky. Zobrazení vypadá přibližně jako na Obr. 16-39. Všimněte si, že zadaná poloha pro osu č.2 se rovná 2500, zatímco zadaná poloha pro slave odpovídá tabulce vačky a má hodnotu 10 000.



Obr. 16-39. Obrazovka RVTExample s první prodlevou

Jakmile skončí čas prodlevy, motory budou pokračovat ve vykonávání příkazů, dokud se nedosáhne druhého příkazu DWELL, kde můžete vizuálně ověřit, že sledování je správné. Zobrazení vypadá přibližně jako na obrázku 16-40. Všimněte si, že zadaná poloha pro osu č.2 se rovná 47 500, zatímco zadaná poloha pro slave odpovídá tabulce vačky a má hodnotu 11 000.



Obr. 16-40. Obrazovka RVTExample s druhou prodlevou

Když master osa dosáhne 50 000 (47 500 +2 500), povel CAM se ukončí, osa slave provede zpomalení naprogramovanou velikostí zrychlení a zastaví se a obě osy se vrátí na nulu.

Detaily o paměti %AI, %AQ, %I a %Q DSM314 najdete v kapitole 5.

Dodatek A

Hlášení chyb

Chybové kódy DSM314

DSM314 hlásí chybové kódy v těchto paměťových místech %AI:

Paměťové místo %AI	Hlášená data	Použití
00	Kód stavu modulu	Chyby nevztahující ke konkrétní ose
04	Chybový kód osy 1	Chyby vztahující se k ose 1
24	Chybový kód osy 2	Chyby vztahující se k ose 2
44	Chybový kód osy 3	Chyby vztahující se k ose 3
64	Chybový kód osy 4	Chyby vztahující se k ose 4

Každý chybový kód je hexadecimální slovo, které popisuje udávanou chybu, když bude nastavený stavový bit %I *Výskyt chyby modulu*.

Kódové slovo stavu modulu

Stavové slovo %AI ***Stavový kód modulu*** hlásí následující dvě kategorie chyb:

- Chyby modulu, které se nevztahují ke konkrétní ose. Příkladem těchto chyb může být hardwarová závada zjištěná při vlastním testu nebo požadavek vykonat prázdný nebo neplatný program. Nový *Kód stavu modulu* nenahradí předchozí *Kód stavu modulu*, dokud nový *Kód stavu modulu* nebude mít prioritu Rychlého zastavení nebo Systémové chyby. Ty je možno vynulovat pomocí bitu %Q Vynulování chyby.
- Systémové stavové chyby. Mají formát Dxxx, Exxx a Fxxx. Pokud se vyskytne některý z těchto kódů, modul nebude pracovat a bit %Q Vynulování chyby chybu nevynuluje. Podrobnosti viz odstavec "Systémové stavové chyby" dále v tomto dodatku.

Kódová slova chyby osy

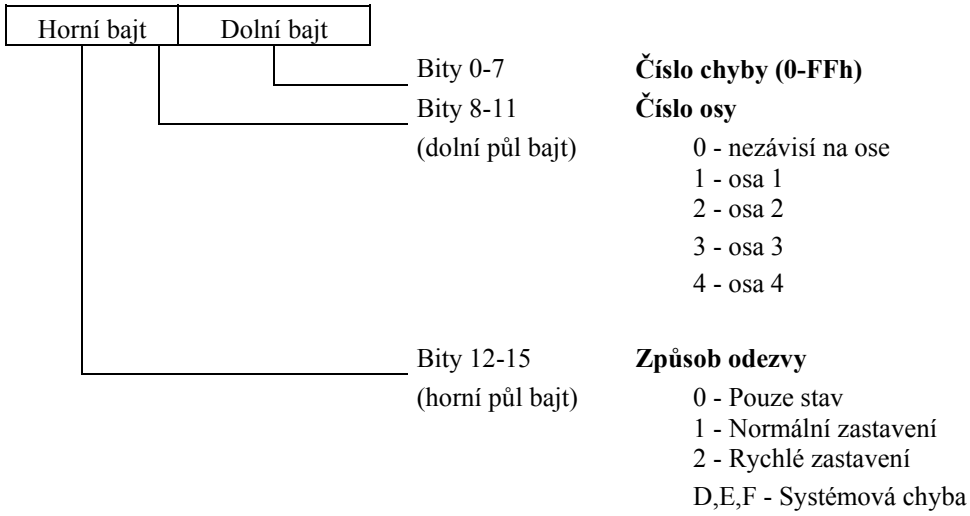
Všechny chyby vztahující se k pohybu osy se hlásí v příslušném stavovém slově %AI ***Chybový kód osy***. Kdykoliv bit %I *Výskyt chyby modulu* bude v jedničce, je nutno zkontrolovat hlášené chyby ve všech chybových slovech (včetně *Stavového kódu modulu)*. Nový *Chybový kód osy* nahradí předchozí *Chybový kód osy*, pokud má v porovnání s předchozím *Chybovým kódem osy* stejnou nebo vyšší prioritu (Výstraha, Normální zastavení, Rychlé zastavení).

Chybové kódy, které zastaví osu, vynulují bit %I *Osa OK* pro tuto osu. Uživatelská logika, která pošle povely %Q nebo %AQ do osy, musí být potvrzena příslušným bitem %I *Osa OK*. Pokud bit *Osa OK* bude v nule, osa neodpoví na žádný povel %Q nebo %AQ kromě *Vynulování chyby* nebo *Načtení parametru*. Bit %Q *Vynulování chyby* vždy vynuluje *Chybový kód osy*; pokud však bude stále existovat stav, který tuto chybu způsobil, chyba se bude okamžitě hlásit znovu.

Poznámka: Dioda STAT LED na čelní desce modulu bude blikat pomalu (čtyřikrát/sekundu) v případě pouze stavových chyb a rychle (osmkrát/sekundu) v případě chyb, které způsobí zastavení serva. V případě zjištění fatální hardwarové chyby při zapnutí napájení dioda STAT LED bude blikáním udávat kód chyby, který je nutno hlásit GE Fanuc. Více podrobností najdete v části "Kontrolky LED" dále v této kapitole.

Formát chybového kódu

Všechny chybové kódy se předávají jako hexadecimální data v následujícím formátu:



Obrázek A-1. Organizace stavového kódu

Způsoby odezvy

1. **Chyby Pouze stav:** Nastaví bit %I *Výskyt chyby modulu* a *slovo %AI Stavový kód modulu* nebo *Chybový kód osy*, ale nemá vliv na pohyb.

   Poznámka: Pokud nebude uvedeno jinak, bude se ignorovat každý povel, který způsobí chybu Pouze stav.

2. **Chyby Normální zastavení:** Provedou interní zrušení všech aktuálních pohybů s použitím aktuálního ***Zrychlení jogu*** a ***Režimu zrychlení jogu*** (LINEAR nebo S–CURVE). Oba bity %I *Pohon povolen* a *Osa povolena* přejdou do nuly po nakonfigurované ***Prodlevě zákazu pohonu***.
3. **Chyby Rychlé zastavení:** Okamžitě zruší všechny pohyby nastavením povelu rychlosti serva na nulu. Oba bity %I *Pohon povolen* a *Osa povolena* přejdou do nuly po nakonfigurované ***Prodlevě zákazu pohonu***.

Systémové chyby (zobrazené pouze ve stavovém kódu modulu): DSM se zakáže a nebude odpovídat na povely řízení PLC. Systémové chyby nelze vynulovat, dokud se do DSM nepošle nová konfigurace.

Tabulka A-1. Chybové kódy DSM314

Chybový kód (hex)	Odpověď	Popis	Typ chyby	Možná příčina
00	Žádná	Žádná chyba	Všechny	
		Chyby konfigurace		
02	Pouze stav	Měřítko dat příliš velké, použije se maximální hodnota rozsahu	Osa	Zkontrolujte konfiguraci osy DSM v HWCFG
03	Pouze stav	Výchozí poloha > kladné EOT, použije se kladné EOT	Osa	Zkontrolujte konfiguraci osy DSM v HWCFG
04	Pouze stav	Výchozí poloha < záporné EOT, použije se záporné EOT	Osa	Zkontrolujte konfiguraci osy DSM v HWCFG
05	Pouze stav	Neplatný parametr ladění v řádku 1; data se budou ignorovat	Osa	Zkontrolujte konfiguraci ladění DSM na záložce HWCFG Advanced, řádek 1
06	Pouze stav	Neplatný parametr ladění v řádku 2; data se budou ignorovat	Osa	Zkontrolujte konfiguraci ladění DSM na záložce HWCFG Advanced, řádek 2
07	Pouze stav	Neplatný parametr ladění v řádku 3-16; data se budou ignorovat	Osa	Zkontrolujte ladění DSM na záložce HWCFG Advanced. Jeden nebo více parametrů na řádku 3-16 je neplatných
0A	Systémová chyba	Výstup zapsaný lokální logikou není nakonfigurovaný pro řízení lokální logikou	Modul	Název bloku lokální logiky je zadaný v konfiguraci a hardwarová konfigurace (záložka Output Bits – výstupní bity) nemá požadovaný výstup nakonfigurovaný na řízení lokální logikou.
0B	Systémová chyba	Bit CTL zapsaný lokální logikou není nakonfigurovaný pro řízení lokální logikou	Modul	Název bloku lokální logiky je zadaný v konfiguraci a název bloku lokální logiky je zadaný v konfiguraci a hardwarová konfigurace (záložka CTL Bits – bity CTL) nemá požadovaný CTL bit nakonfigurovaný na řízení lokální logikou.
		Chyba konfiguračního parametru		
17	Pouze stav	Chyba nastavení EOT	Osa	Softwarový konec posuvu je povolený v konfiguraci a hodnoty horní nebo dolní softwarového konce posuvu jsou mimo horní nebo dolní mez. Konfiguraci je nutno změnit buď zakázáním softwarového konce posuvu nebo nastavením hodnot konce posuvu uvnitř mezí.
18	Pouze stav	(Pouze pomocná) Modul počtu pulsů EOT není celé číslo	Osa	Zkontrolujte konfiguraci osy DSM v HWCFG
19	Pouze stav	Horní/dolní mez počtu pulsů EOT není celé číslo	Osa	Zkontrolujte konfiguraci osy DSM v HWCFG
1C	Pouze stav	Nepodporovaný režim povelu AQ	Osa	Povely AQ, které konfigurují proměnné režimu kroutícího momentu, nelze použít v Analogovém rychlostním režimu.
1D	Normální zastavení	Snaha použít povely CAM, CAM-Load nebo CAM-Phase v režimu vlečené osy	Osa	Když budete používat vačku, přesvědčte se, že není nakonfigurovaný režim vlečené osy (režim vlečené osy nelze použít, když se používá vačka). Pokud budete používat režim vlečené osy, zajistěte, aby se v pohybových programech nevyskytovaly povely vačky (vačku nelze použít, když bude nakonfigurovaný režim vlečené osy).
1E	Pouze stav	Okamžitý povel Rychlost jogu je mimo rozsah, povel se ignoruje	Osa	Poslaný okamžitý povel AQ Rychlost jogu je příliš velký Zapište povel znovu s menší hodnotou
1F	Pouze stav	Okamžitý povel Zrychlení jogu je mimo rozsah, povel se ignoruje	Osa	Poslaný okamžitý povel AQ Zrychlení jogu je příliš velký. Zapište povel znovu s menší hodnotou
		Chyby programu		
20	Pouze stav	Naprogramované zrychlení je mimo rozsah, zrychlení se nastaví na maximální hodnotu	Osa	Naprogramované zrychlení v aktuálně vykonávaném pohybovém programu je příliš velké. V pohybovém programu se použije maximální hodnota (1 073 741 823 puls/sec/sec při měřítku 1:1).
21	Pouze stav	Naprogramované zrychlení je příliš malé, nastaví se na výchozích 32 puls/sec/sec	Osa	Naprogramované zrychlení v aktuálně vykonávaném pohybovém programu je příliš malé. V pohybovém programu se použije výchozí hodnota (32 puls/sec/sec).
22	Pouze stav	Rychlost nastavená podle měřítka je větší než 1 milion pulsů/sec, použije se 1 milion puls/sec	Osa	Zkontrolujte měřítko v konfiguraci, rychlost v programu
23	Pouze stav	Naprogramovaná rychlost je nula, nastavte minimální hodnotu 1 puls/sekundu.	Osa	Naprogramovaná rychlost v aktuálně vykonávaném pohybovém programu je nula. Použije se minimální hodnota (1 puls/sekundu).
24	Pouze stav	Rychlost pohybového programu > nakonfigurovaná mez rychlosti, použije se mezní hodnota	Osa	Naprogramovaná rychlost v aktuálně vykonávaném programu je větší než je mez rychlosti nastavená v konfiguraci osy.

25		Vyhrazeno – nepoužívá se v DSM314	Osa	
26	Normální zastavení	Chyba masky skoku	Osa	Spojte se s GE Fanuc Automation
27	Normální zastavení	Chyba masky čekání	Osa	Spojte se s GE Fanuc Automation
28	Normální zastavení	Parametrická poloha je příliš velká	Osa	Poloha obsažená v parametru adresovaném aktuálním povelem PMOVE nebo CMOVE je větší než maximální rozsah polohy (-536 870 912 až +536 870 911 při měřítku 1:1)
29	Pouze stav	Doba prodlevy je větší než 60 sekund, použije se 5 sekund	Osa	Při vykonávání pohybového programu se zjistil příkaz DWELL, kde doba prodlevy DWELL je větší než 60 sekund. Tato hodnota je větší než přípustná. Doba DWELL použitá v programu je 5 sekund. Otevřete pohybový program a opravte dobu příkazu DWELL tak, aby byla menší než 60 sekund. Pokud je nutná větší doba DWELL, použijte více příkazů DWELL
2A	Normální zastavení	CTL podmínka ukončení cyklické vačky zadaná pro necyklickou vačku	Osa	Podmínka ukončení v CTL je přípustná pouze pro cyklické vačky. Pohybový program obsahuje instrukci necyklické vačky s CTL podmínkou ukončení.
2B	Normální zastavení	Fáze vačky mimo rozsah	Osa	Hodnota CAM PHASE je mimo rozsah polohy osy.
Chyby inkrementu polohy				
2C	Pouze stav	Chyba překročení rozsahu inkrementu polohy, inkrement se ignoruje	Osa	Inkrement polohy v povelu AQ musí být v rozsahu -128 až 127 uživatelských jednotek
2D	Normální zastavení	Chyba konfigurace vačky master osy – master profil nesouhlasí s konfigurací master osy	Osa	1) Poměr uživatelské jednotky : počet pulsů pro master osu v editoru a v hardwarové konfiguraci nejsou kompatibilní a/nebo 2) Mez horní/dolní polohy zadaná pro master osu v hardwarové konfiguraci není kompatibilní s profilem. Podrobný popis nastavení mezí horní/dolní polohy najdete v části o typech vaček.
2E	Normální zastavení	Chyba konfigurace vačky slave osy – slave profil nesouhlasí s konfigurací master osy	Osa	1) Poměr uživatelské jednotky : počet pulsů pro slave osu v editoru a v hardwarové konfiguraci nejsou kompatibilní a/nebo 2) Mez horní/dolní polohy zadaná pro slave osu v hardwarové konfiguraci není kompatibilní s profilem. Podrobný popis nastavení mezí horní/dolní polohy najdete v části o typech vaček.
2F	Normální zastavení	Režim SW EOT vačky slave osy nelze povolit pro cyklickou kruhovou vačku	Osa	
Chyby nájezdu do výchozí polohy				
30	Pouze stav	Chyba Nalezení výchozí polohy při nepovoleném pohonu	Osa	Povel Nalezení výchozí polohy se vykonal, když bit Povolení pohonu nebyl v jedničce. Povolte pohon a vykonejte povel znovu.
31	Pouze stav	Chyba Nalezení výchozí polohy při zvoleného programu	Osa	Povel Nalezení výchozí polohy se vykonal, když se vykonával pohybový program. Pohybový program je nutno zastavit (bit %I Program aktivní v nule) před povelem Nalezení výchozí polohy.
32	Pouze stav	Nalezení výchozí polohy během Vynucené rychlosti digitálního serva nebo Vynuceného analogového výstupu	Osa	Povel Nalezení výchozí polohy se vykonal, když se odesílal AQ povel Vynucené rychlosti digitálního serva (34h) nebo Vynuceného analogového výstupu (24h). Tento povel je nutno zrušit před vykonáním povelu Nalezení výchozí polohy.
33	Pouze stav	Chyba Nalezení výchozí polohy během jogu	Osa	Vykonal se povel Nalezení výchozí polohy, když servo provádělo jog. Povel Jog je nutno zastavit před vykonáním povelu Nalezení výchozí polohy.
34	Pouze stav	(1) Chyba Nalezení výchozí polohy, když se vykonával pohyb rychlostí, nebo (2) Nalezení výchozí polohy, když je aktivní jiný cyklus Nalezení výchozí polohy	Osa	Vykonává se povel Nalezení výchozí polohy, (1) když se vykonával AQ povel Pohyb rychlostí (22h) nebo (2) když probíhal jiný cyklus Nalezení výchozí polohy. V případě (1) zastavte operaci Pohyb rychlostí (bit I Pohyb v nule) před vykonáním povelu Nalezení výchozí polohy. V případě (2) ověřte, že osa je v poloze a nepohybuje se před vykonáním povelu Nalezení výchozí polohy.
35	Pouze stav	Nalezení výchozí polohy při povolené vlečené ose	Osa	Vykonal se povel Nalezení výchozí polohy, když je povolená funkce vlečené osy. Vlečenou osu je nutno zakázat (bit I Vlečená osa povolena v nule) před vykonáním povelu Nalezení výchozí polohy.

36	Pouze stav	Chyba Nalezení výchozí polohy, když bit Zrušit je v jedničce	Osa	Vykonal se povel Nalezení výchozí polohy, když bit Zrušit je nastavený do jedničky. Bit Zrušit je nutno nastavit na nulu před vykonáním povelu Nalezení výchozí polohy.
37	Pouze stav	Chyba Nalezení výchozí polohy při prvním cyklu PLC.	Osa	Bit Q Nalezení výchozí polohy byl v jedničce během prvního cyklu PLC. PLC program je nutno opravit tak, aby se tento povel neposlal při prvním cyklu PLC.
		Chyby Pohybu rychlostí		
38	Pouze stav	Chyba Pohyb rychlostí při prvním cyklu PLC	Osa	Povel Pohyb rychlostí (22h) byl poslaný během prvního cyklu PLC. PLC program je nutno opravit tak, aby se tento povel neposlal při prvním cyklu PLC.
39	Pouze stav	Chyba Pohyb rychlostí při nepovoleném pohonu	Osa	Povel Pohyb rychlostí (22h) byl poslaný, když bit Povolení pohonu nebyl v jedničce. Povolte pohon a vykonejte povel znovu.
3A	Pouze stav	Chyba Pohyb rychlostí při zvoleném programu	Osa	Povel Pohyb rychlostí (22h) byl poslaný, když pohybový program byl zvolený k vykonání. Pohybový program je nutno zastavit (bit %I Program aktivní v nule) před posláním povelu Pohyb rychlostí.
3B	Pouze stav	Chyba Pohyb rychlostí, když je aktivní cyklus výchozí polohy	Osa	Povel Pohyb rychlostí (22h) byl poslaný, když modul vykonával cyklus výchozí polohy. Před posláním povelu Pohyb rychlostí je nutno buď zrušit cyklus výchozí polohy nebo počkat, až se cyklus výchozí polohy dokončí.
3C	Pouze stav	Chyba Pohyb rychlostí během jogu	Osa	Povel Pohyb rychlostí (22h) byl poslaný, když byl aktivní bit Q Jog. Povel Jog je nutno zastavit před posláním povelu Pohyb rychlostí.
3D	Pouze stav	Chyba Pohyb rychlostí, když je nastavený bit Zrušit všechny pohyby	Osa	Povel Pohyb rychlostí (22h) byl poslaný, když bit Q Zrušit všechny pohyby je v jedničce. Bit Zrušit je nutno vynulovat před posláním povelu Pohyb rychlostí.
3E	Pouze stav	Data pohybu rychlostí větší než 8 388 607 uživatelských jednotek/sekundu	Osa	Poslal se povel Pohyb rychlostí (22h), kde zadaná rychlost je větší než 8 388 607 uživatelských jednotek/sekundu. Před dalším vykonáním povelu zadanou rychlost nastavte na menší hodnotu.
3F	Pouze stav	Chyba Data pohybu rychlostí větší než 1 milion pulsů/sekundu	Osa	Vykonal se povel Pohyb rychlostí (22h), kde zadaná rychlost je větší než 1 milion pulsů/sekundu. Před dalším vykonáním povelu zadanou rychlost nastavte na menší hodnotu. Zkontrolujte nastavení měřítka.
		Chyby jogu		
40	Pouze stav	Chyba Jog během Nalezení výchozí polohy	Osa	Vykonal se Jog, když modul vykonával funkci Nalezení výchozí polohy. Před vykonáním funkce Jog buď zrušte funkci Nalezení výchozí polohy nebo počkejte, až se dokončí.
41	Pouze stav	Chyba Jog když se vykonával pohyb rychlostí	Osa	Bit Q Jog byl nastavený do jedničky, když se vykonával povel Pohyb rychlostí (22h). Před vykonáním Jogu se akce Pohyb rychlostí musí zastavit.
42	Pouze stav	Chyba Jog během Vynucené rychlosti digitálního serva	Osa	Bit Q Jog byl nastavený do jedničky, když se vykonával povel AQ Vynucená digitální rychlost (34h) nebo Vynucený analogový výstup (24h). Před vykonáním jogu se povel AQ musí odstranit.
43	Pouze stav	Chyba Jog během Zvoleného programu a není Zastavení posuvu	Osa	Pokud program běží, DSM může vykonat Jog, pouze pokud bit Q Zastavení posuvu bude v jedničce.
		Chyby vynucené rychlosti digitálního serva		
47	Pouze stav	Chyba Vynucené rychlosti digitálního serva nebo Vynuceného analogového výstupu během jogu	Osa	Povel Vynucené rychlosti digitálního serva (34h) nebo Vynuceného analogového výstupu (24h), když modul vykonával funkci Jog. Před vykonáním Vynucené rychlosti digitálního serva nebo Vynuceného analogového výstupu je nutno funkci Jog zastavit.
48	Pouze stav	Chyba Vynucené rychlosti digitálního serva nebo Vynuceného analogového výstupu během Pohybu rychlostí	Osa	Povel Vynucené rychlosti digitálního serva (34h) nebo Vynuceného analogového výstupu (24h), když modul vykonával funkci Pohyb rychlostí. Před vykonáním Vynucené rychlosti serva nebo Vynuceného výstupu je nutno povel Pohyb rychlostí zastavit.

49	Pouze stav	Chyba Vynucené rychlosti digitálního serva nebo Vynuceného analogového výstupu během Zvoleného programu	Osa	Povel Vynucené rychlosti digitálního serva (34h) nebo Vynuceného analogového výstupu (24h), když modul vykonával pohybový program. Před vykonáním Vynucené rychlosti digitálního serva nebo Vynuceného analogového výstupu je nutno pohybový program zastavit.
4A	Pouze stav	Chyba Vynucené rychlosti digitálního serva nebo Vynuceného analogového výstupu během povolení vlečené osy	Osa	Povel Vynucené rychlosti digitálního serva (34h) nebo Vynuceného analogového výstupu (24h), když je povolená vlečená osa. Před vykonáním Vynucené rychlosti digitálního serva nebo Vynuceného analogového výstupu je nutno vlečenou osu zakázat (bit I Vlečená osa povolena v nule).
4B	Pouze stav	Vynucený analogový výstup během režimu analogového kroutícího momentu	Osa	Vykonává se povel AQ Vynucený D/A (24h), když servo je nakonfigurované na režim analogového kroutícího momentu. Vynucený analogový výstup se v režimu analogového kroutícího momentu nepodporuje.
Chyby nastavení polohy				
50	Pouze stav	Chyba Nastavení polohy při zvoleném programu	Osa	Vykonal se povel Nastavení polohy, když byl k vykonání zvolený pohybový program. Pohybový program je nutno zastavit (bit %I Program aktivní v nule) před povelem Nastavení polohy.
51	Pouze stav	Chyba Data nastavení polohy mimo rozsah	Osa	Vykonal se povel Nastavení polohy s hodnotou větší než je maximální rozsah polohy (-536 870 912 až +536 870 911 při měřítku 1:1)
52	Pouze stav	Nastavení polohy Během pohybu nebo Nastavení polohy, když není pohyb a není V poloze a rychlost > 100 puls/sec.	Osa	Nastavení polohy je nepřípustné, pokud bit %I Pohyb bude v jedničce. Pokud bit Pohyb bude v nule a bit %I V poloze bude také v nule, Okamžitá rychlost musí být < 100 pulsů/sekundu.
53	Pouze stav	Snaha inicializovat polohu před tím, než digitální snímač polohy projde referenčním bodem.	Osa	Po prvním zapnutí napájení absolutní digitální snímač polohy neprošel nulovým referenčním bodem. Než je možno v absolutním režimu povolit Nastavení polohy, snímač je nutno protočit přes referenční bod (alespoň 1 otáčku).
54	Pouze stav	Neplatná poloha digitálního snímače polohy, nutno použít Nalezení výchozí polohy nebo Nastavení polohy.	Osa	1. Absolutní snímač polohy nebyl inicializovaný po prvním zapnutí napájení. 2. Konfigurace režimu snímače polohy byla změněna z inkrementální na absolutní. 3. Konfigurace směru osy (normální nebo reverzní) byla změněna. 4. Rozlišitelnost snímače polohy (nastavená v parametru na konfigurační záložce Advanced) byla změněna. 5. Vyskytnul se alarm snímače polohy.
55	Pouze stav	Digitální snímač polohy při vypnutém napájení vykonal pohyb příliš daleko	Osa	Digitální absolutní snímač polohy při vypnutém napájení vykonal pohyb o více než 16 383 otáček.
Chyby konce posuvu a mezního počtu pulsů				
56	Pouze stav	Zadaná poloha > kladný konec pohybu nebo horní mez počtu pulsů	Osa	Vykonal se povel, který má za následek zadanou polohu serva přesahující kladný konec posuvu nebo horní mez počtu pulsů. Buď upravte povel tak, aby byl menší než tyto hodnoty, nebo v hardwarové konfiguraci tyto hodnoty zvyšte.
57	Pouze stav	Zadaná poloha < záporný konec pohybu nebo dolní mez počtu pulsů	Osa	Vykonal se povel, který má za následek zadanou polohu serva přesahující záporný konec posuvu nebo dolní mez počtu pulsů. Buď upravte povel tak, aby byl větší než tyto hodnoty nebo v hardwarové konfiguraci tyto hodnoty nastavte na zápornější hodnotu.
58	Pouze stav	Poloha absolutního snímače polohy > horní mez softwarového EOT	Osa	Tato chyba se hlásí při zapnutí napájení nebo nové konfiguraci, pokud se absolutní digitální snímač polohy posunul za horní mez softwarového EOT.
59	Pouze stav	Poloha absolutního snímače polohy < dolní mez softwarového EOT	Osa	Tato chyba se hlásí při zapnutí napájení nebo nové konfiguraci, pokud se absolutní digitální snímač polohy posunul za dolní mez softwarového EOT.
Chyby zákazu pohonu				
5B	Normální zastavení	Pohon zakázaný během pohybu	Osa	Bit Q Povolit pohon přešel do nuly, když servo vykonávalo Jog nebo Pohyb rychlostí (bit I Pohyb v jedničce). PLC program je nutno opravit, aby nedocházelo k této chybě. Zvažte použití bitu Pohyb v logice, která zakáže pohon.

5C	Normální zastavení	Pohon zakázaný během aktivního programu	Osa	Bit Q Povolit pohon přešel do nuly, když servo vykonávalo pohybový program (bit I Program aktivní v jedničce). PLC program je nutno opravit, aby nedocházelo k této chybě. Zvažte použití bitu Program aktivní v logice, která zakáže pohon.
		Softwarové chyby		
5F	Pouze stav	Softwarová chyba (zavolejte servis GE Fanuc)	Osa	Spojte se s GE Fanuc Automation
60	Pouze stav	Chyba výpočtu rotační polohy absolutního snímače polohy	Osa	Spojte se s GE Fanuc Automation
		Chyby programu a podprogramu		
61	Normální zastavení	Neplatné číslo podprogramu	Osa	Pohybový program vyvolal podprogram, který se nenachází v programové paměti modulu. Pokud instrukce volání bude adresovat parametr., který obsahuje číslo podprogramu, přesvědčte se, že data parametru jsou správná.
62	Normální zastavení	Chyba volání (podprogram již je v ose aktivní)	Osa	Podprogram pohybu volá sebe sama nebo volá jiný podprogram, který vyvolal původní podprogram.
63	Normální zastavení	V programu nalezen povel Konec podprogramu	Osa	Pohybový program obsahuje neplatný povel konce podprogramu uvnitř hlavního pohybového programu (Program 1-10). Upravte pohybový program tak, aby neobsahoval tento výrok.
64	Normální zastavení	V podprogramu nalezen povel Konec programu	Osa	Pohybový podprogram obsahuje neplatný povel konce programu uvnitř pohybového podprogramu (podprogram 1-40). Upravte podprogram tak, aby neobsahoval tento výrok.
65	Normální zastavení	Synchronizační podprogram zjistil nesynchronizační program	Osa	Pohybový program zjistil synchronizační blok v programu, který není pro více os a je nastavený pro synchronizační bloky.
66	Normální zastavení	Profil vačky v načteném bloku vačky nenalezen	Osa	Profil vačky není v editoru vačky napojený na blok načtení vačky a/nebo v hardwarové konfiguraci nebyl zadaný název bloku načtení vačky Konfigurace.
67	Normální zastavení	Vzdálenost ukončení vačky je mimo rozsah (necyklické vačky)	Osa	Vzdálenost ukončení pro necyklické vačky je větší než je modul vačky.
68	Pouze stav	(Korekce povolena) Povel rychlosti je omezený v důsledku porušení meze rychlosti nebo porušení meze polohové odchylky.	Osa	
69	Normální zastavení	(Korekce zakázána) Povel rychlosti vačky je větší než nakonfigurovaná mez rychlosti osy.	Osa	
6A	Normální zastavení	Překročení meze polohové odchylky vačky (se zakázanou korekcí)	Osa	
6B	Pouze stav	Zadaná poloha vačky při ukončení je jiná než hodnota profilu vačky v důsledku meze polohové odchylky nebo rychlosti.	Osa	
6C	Normální zastavení	Master hodnota vačky je mimo master rozsah profilu necyklické vačky (povely CAM a CAM-LOAD)	Osa	
6D	Normální zastavení	Absolutní režim vačky po inkrementálním režimu vačky v sekvenci	Osa	
6F	Rychlé zastavení	Chyba výpočtu trajektorie vačky	Osa	Spojte se s GE Fanuc Automation
		Chyby při vykonávání programu		
70	Pouze stav	Vykonání programu při prvním cyklu PLC	Modul	Bit Q Vykonání programu byl v jedničce během prvního cyklu PLC. PLC program je nutno opravit tak, aby tyto bity Q nebyly v jedničce při prvním cyklu PLC.
71	Pouze stav	Příliš mnoho programů požadovaných ve stejném cyklu PLC	Modul	Počet bitů Q Vykonání programu, které přešly do jedničky v 1 cyklu, je větší než nakonfigurovaný počet os.
72	Pouze stav	Vykonání programu pro osu 1 nebo 2 při aktivním programu pro více os	Modul	Bit Q Vykonání programu byl nastavený do jedničky pro program osy 1 nebo osy 2, když program pro více os již byl aktivní.
73	Pouze stav	Vykonání programu pro osu nakonfigurovanou jako Omezená pomocná osa	Modul	Pohybové programy nelze vykonat s osou nakonfigurovanou jako Omezená pomocná osa. Omezená pomocná osa vykonává pouze zpracování polohové zpětné vazby a nemá interní generátor dráhy pohybu.
74		Vyhrazeno – nepoužívá se v DSM314		
75	Pouze stav	Požadovaný prázdný nebo neplatný program	Modul	Bit Q Vykonání programu byl nastavený do jedničky pro číslo programu nedefinované v nakonfigurovaném bloku pohybového programu. Zkontrolujte, jestli je správně nakonfigurovaný název bloku pohybového programu. Přesvědčte se, že požadované číslo programu je definované v nakonfigurovaném programovém bloku.

76	Pouze stav	Zadaná poloha povelu AQ Pohyb mimo rozsah	Osa	Poslaný povel AQ Pohyb (27h) s hodnotou polohy větší než maximální rozsah polohy. (-536 870 912 až +536 870 911 při měřítku 1:1)
77	Pouze stav	Povel pohybu AQ při prvním cyklu PLC	Osa	Povel AQ Pohyb (27h) byl zadaný během prvního cyklu PLC. PLC program je nutno opravit tak, aby se povel AQ Pohyb neposlal při prvním cyklu.
Chyby podmínek vykonávání programu				
80	Pouze stav	Vykonání programu, když je aktivní cyklus výchozí polohy	Osa	PLC nastavil do jedničky bit Q Vykonání programu, když modul vykonával cyklus výchozí polohy. Buď je nutno počkat, až se cyklus výchozí polohy dokončí, nebo cyklus výchozí polohy před vykonáním pohybového programu zrušit.
81	Pouze stav	Vykonání programu během jogu	Osa	PLC nastavil do jedničky bit Q Vykonání programu, když modul vykonával operaci jogu. Bity jogu (z PLC nebo z lokální logiky) je nutno před vykonáním pohybového programu nastavit do nuly.
82	Pouze stav	Vykonání programu během pohybu rychlostí	Osa	PLC nastavil bit Q Vykonání programu do jedničky, když se vykonával povel Pohyb rychlostí (22h). Před vykonáním pohybového programu je nutno povel Pohyb rychlostí zastavit.
83	Pouze stav	Vykonání programu během Vynucené rychlosti digitálního serva nebo Vynuceného analogového výstupu	Osa	PLC nastavil bit Q Vykonání programu do jedničky, když se vykonával povel Vynucená digitální rychlost (34h) nebo Vynucený analogový výstup (24h). Před vykonáním pohybového programu je nutno odstranit povel Vynucená rychlost digitálního serva nebo Vynucený analogový výstup.
84	Pouze stav	Vykonání programu během aktivního programu	Osa	PLC nastavil bit Q Vykonání programu do jedničky pro osu, která již vykonávala pohybový program. Před vykonáním dalšího programu pro stejnou osu se nejdříve musí dokončit aktuální program (bit I Program aktivní do nuly).
85	Pouze stav	Vykonání programu, když bit Zrušit všechny pohyby je v jedničce	Osa	PLC nastavil do jedničky bit Q Vykonání programu, když modul vykonával Zrušit všechny pohyby. Před vykonáním programu musí být bit Q Zrušit, bit I Pohyb a bit I Program aktivní v nule.
86	Pouze stav	Vykonání programu, když Poloha platná není v jedničce	Osa	PLC nastavil do jedničky bit Q Vykonání programu, když bit I Poloha platná byl v nule. Poloha platná se musí nastavit do jedničky cyklem Nalezení výchozí polohy nebo povelem Nastavení polohy.
87	Pouze stav	Vykonání programu, když Pohon povolen není v jedničce	Osa	PLC nastavil do jedničky bit Q Vykonání programu, když pohon nebyl povolený (bit I Pohon povolen je v nule). Aby se povolil pohon, bit Q Povolit pohon musí být v jedničce.
Chyby synchronního programového bloku				
8C	Pouze stav	Chyba synchronního bloku během CMOVE	Osa	Vykonávání programu zjistilo CMOVE identifikovaný synchronním blokem, i když druhá osa k synchronnímu bloku ještě nedospěla.
8D	Pouze stav	Chyba synchronního bloku během skoku	Osa	Vykonávání programu skočilo na CMOVE nebo PMOVE identifikovaný synchronním blokem, i když druhá osa k synchronnímu bloku ještě nedospěla.
Chyby EEPROM				
90	Pouze stav	Chyba programování paměti Flash EEPROM	Modul	Spojte se s GE Fanuc Automation
Chyby lokální logiky				
91	Rychlé zastavení	Systémové zastavení lokální logiky	Modul	Program lokální logiky vykonal příkaz, který zapsal do proměnné System_Halt (např. System_Halt := 1;)
92	Rychlé zastavení	Chyba překročení doby lokální logiky	Modul	Program lokální logiky překročil dobu 300 mikrosekund vyhrazenou pro vykonávání. Zkraťte dobu vykonávání lokální logiky snížením počtu příkazů lokální logiky nebo úpravou struktury programu. Více podrobností o době vykonávání lokální logiky najdete v Dodatku E.
93	Rychlé zastavení	Chyba lokální logiky dělení nulou	Modul	Program lokální logiky vykonal dělení nulou nebo modulo nula. Zkontrolujte příkazy dělení programu lokální logiky, jestli zde není chyba. Možným zdrojem těchto chyb můžou být registry parametrů, které obsahují nulu.

94	Rychlé zastavení	Chyba přetečení lokální logiky dělení/modulo	Modul	Program lokální logiky vykonal dělení (nebo modulo) 64-bitového celého čísla a výsledek se nevejde do 32-bitového celého čísla. Zkontrolujte příkazy dělení programu lokální logiky, jestli zde není chyba.
95	Pouze stav	Upozornění na přetečení při sčítání/odčítání lokální logiky	Modul	Program lokální logiky sečetl nebo odečetl dvě čísla, která způsobila přetečení. Přípustný rozsah je -2 147 483 648 až +2 147 483 647. Změňte program lokální logiky tak, aby nedocházelo k přetečení nebo na konci každého cyklu lokální logiky nastavte proměnnou Přetečení na 0.
96	Pouze stav	Upozornění na přetečení absolutní hodnoty (ABS) lokální logiky	Modul	Program lokální logiky se snaží vykonat operaci ABS s číslem -2 147 483 648, která vede na přetečení.
97	Pouze stav	Upozornění na překročení doby lokální logiky	Modul	Doba vykonávání programu lokální logiky je blízko (větší než 275 mikrosekund) maximální přípustné doby vykonávání (300 mikrosekund). Zkraťte dobu vykonávání lokální logiky snížením počtu příkazů lokální logiky nebo úpravou struktury programu. Více podrobností o době vykonávání lokální logiky najdete v Dodatku E.
98	Pouze stav	Chyba prvního cyklu vykonávání lokální logiky	Modul	Snaha vykonat lokální logiku při prvním cyklu (např. bit Q Povolení lokální logiky je v jedničce, když PLC přešlo z režimu Stop do režimu Run).
99	Pouze stav	Neplatný nebo chybějící název programu lokální logiky v hardwarové konfiguraci	Modul	Název programu lokální logiky zadaný v hardwarové konfiguraci je neplatný (nebo prázdný) nebo lokální logika nebyla v hardwarové konfiguraci povolená.
9A	Rychlé zastavení	Chyba zastavení lokální logiky (pro danou osu)	Osa	Vyskytla se chyba rychlého zastavení lokální logiky (chybové kódy 91-94).
		Chyby hardwarového koncového spínače		
A0	Rychlé zastavení	Chyba koncového spínače (+)	Osa	Koncový spínač kladného přejetí je v nule. Pokud se nepoužívají koncové spínače přejetí, konfiguraci koncového spínače přejetí nastavte na Zakázáno.
A1	Rychlé zastavení	Chyba koncového spínače (-)	Osa	Koncový spínač záporného přejetí je v nule. Pokud se nepoužívají koncové spínače přejetí, konfiguraci koncového spínače přejetí nastavte na Zakázáno.
		Hardwarové chyby		
A8	Rychlé zastavení	Chyba porušení synchronizace	Osa	Polohová odchylka překročila mez polohové odchylky. Možné zdroje této chyby jsou: 1. Mez polohové odchylky je pro aplikaci nastavená příliš nízko. 2. Zpětnovazební zařízení je odpojeno nebo prokluzuje na řízeném zařízení. 3. Nesprávné zapojení zpětnovazebního zařízení. (tj. kladné otáčení se zpětnovazebním zařízením indikuje jako záporné)
A9	Pouze stav	Ztráta polohové zpětné vazby	Osa	U snímače s fázovým posuvem se zjistila chyba fázového posunutí. Zkontrolujte zapojení snímače polohy a přesvědčte se, že snímač polohy nepracuje nad jmenovitou rychlostí.
B0-BE		Viz tabulka A-3		Alarmy digitálního serva jsou zdokumentované v tabulce A-3.

Alarmy snímače polohy				
C0	Rychlé zastavení	Servo nepřipraveno	Osa	U analogových serv vstup Pohon připraven na čelní desce musí být nastavený (0 voltů) během 1 sekundy po přechodu bitu Q Pohon povolen do jedničky. Pokud se vstup Pohon připraven u analogových serv nebude používat, konfigurace vstupu musí být nastavena na Zakázáno. U digitálních serv FANUC může být aktivován vstup E–Stop zesilovače , nebo se může vyskytnout chyba zesilovače.
C1	Pouze stav	Vybitá baterie sériového snímače polohy	Osa	Napětí baterie sériového snímače polohy je nízké. Baterii je nutno vyměnit nebo snímač polohy je možno nakonfigurovat na inkrementální provoz (místo absolutního).
C2	Normální zastavení	Porucha baterie sériového snímače polohy	Osa	Baterie sériového snímače polohy má poruchu. Baterii je nutno vyměnit nebo snímač polohy je možno nakonfigurovat na inkrementální provoz (místo absolutního).
C3	Normální zastavení	Překročení teploty servomotoru	Osa	Servomotor nebo řídicí firmware hlásí stav překročení teploty. Je nutno zkontrolovat pohybový program a přesvědčit se, že není překročený činitel využití motoru. Také je nutno zkontrolovat montáž motoru, jestli je dostatečný odvod tepla a odpovídající ventilace.
C4	N/A	Nepoužívá se.	N/A	
C5	Rychlé zastavení	Ztráta snímače polohy	Osa	Modul není schopný komunikovat se snímačem polohy. Přesvědčte se, že servozesilovač je zapnutý. Zkontrolujte zapojení snímače polohy a přesvědčte se, že je zapojený kabel. Dále zkontrolujte jestli je správné uzemnění.
C6	Rychlé zastavení	Chyba detekce pulsů snímače polohy	Osa	Obvod detekce pulsů snímače polohy zjistil chybu. Přesvědčte se, že motor je řádně uzemněný. Pokud se chyba bude vyskytovat nadále, spojte se výrobním závodem.
C7	Rychlé zastavení	Chyba čítače snímače polohy	Osa	Obvod čítače pulsů snímače polohy zjistil chybu. Přesvědčte se, že motor je řádně uzemněný. Pokud se chyba bude vyskytovat nadále, spojte se výrobním závodem.
C8	Rychlé zastavení	LED snímače polohy je odpojená	Osa	Dioda LED snímače polohy je odpojená. Spojte se s výrobním závodem.
C9	Rychlé zastavení	Chyba CRC kontrolního součtu snímače polohy	Osa	Obvod komunikace snímače polohy zjistil chybu CRC. Zkontrolujte, jestli možná příčina chyby není v zapojení země snímače polohy a uzemnění motoru. Zkontrolujte další zdroje elektrického šumu v místě zapojení motoru a snímače polohy. Pokud možno, oddělte zdroje od zapojení motoru/snímače polohy. Pokud se chyba bude vyskytovat nadále, spojte se výrobním závodem.
CA	Rychlé zastavení	Nepodporovaný typ snímače polohy, lineární typ nebo typ A	Osa	Snímač polohy motoru připojený k modulu není podporovaný. Motor buď není podporovaný modulem DSM nebo modul má k motoru připojený nesprávný snímač polohy. Zkontrolujte štítek motoru a ověřte, že model motoru je podporovaný. Pokud se problémy budou vyskytovat nadále, spojte se výrobním závodem.
CB	Rychlé zastavení	Nepodporovaný snímač polohy, typ C	Osa	Snímač polohy motoru připojený k modulu není podporovaný. Motor buď není podporovaný modulem DSM nebo modul má k motoru připojený nesprávný snímač polohy. Zkontrolujte štítek motoru a ověřte, že model motoru je podporovaný. Pokud se problémy budou vyskytovat nadále, spojte se výrobním závodem.
CC	Normální zastavení	Zmeškaný puls DZ, když DS přešel z 1 do 0	Osa	Data polohy mohou být nesprávná. Proveďte vypnutí a zapnutí napájení motoru a zesilovače. Pokud se problém bude vyskytovat nadále, spojte se výrobním závodem.
Alarmy DSP				
D1	Rychlé zastavení	Zjištěný nadměrný proud	Osa	Firmware pro řízení motoru zjistil stav nadměrného proudu. Možné zdroje této chyby jsou : - Nesprávně zvolený typ motoru v hardwarové konfiguraci - Motor je nadměrně zpětně poháněn strojem - Stav překročení činitele využití
D2	N/A	Nepoužívá se		

D3	Rychlé zastavení	Zjištěno nadměrné zrychlení	Osa	Firmware řízení motoru zjistil hodnotu zrychlení, která překračuje přípustné hodnoty. Tato chyba se za normálních provozních podmínek nemůže objevit. Možné příčiny chyby jsou porucha snímače polohy, prokluzování snímače polohy, nesprávná poloha hlášená ze snímače polohy. Pokud se chyba nevysvětlí fyzickým hardwarem, spojte se s výrobním závodem.
D4	Rychlé zastavení	Zjištěna příliš velká rychlost	Osa	Firmware řízení motoru zjistil hodnotu rychlosti, která překračuje přípustné hodnoty. Tato chyba se za normálních provozních podmínek nemůže objevit. Možné příčiny chyby jsou porucha snímače polohy, prokluzování snímače polohy, nesprávná poloha hlášená ze snímače polohy. Pokud se chyba nevysvětlí fyzickým hardwarem, spojte se s výrobním závodem.
D5	Pouze stav	Zisk rychlostní smyčky Kp je příliš velký	Osa	Proporcionální zisk rychlostní smyčky překračuje přípustné hodnoty. Hodnoty jsou omezené platným rozsahem. Tato chyba se během normálního provozu nemůže objevit. Možné příčiny chyby jsou nesprávný typ motoru zvolený v hardwarové konfiguraci nebo hodnoty zisku rychlostní smyčky, které jsou příliš velké. Pokud je typ motoru v hardwarové konfiguraci správný, pak snižte zisk rychlostní smyčky. Pokud problém bude i nadále nebo zisk rychlostní smyčky bude pro danou aplikaci příliš malý, spojte se s výrobním závodem.
D6	Pouze stav	Integrační zisk je příliš velký	Osa	Integrační zisk rychlostní smyčky překračuje přípustné hodnoty. Hodnoty jsou omezené platným rozsahem. Tato chyba se během normálního provozu nemůže objevit. Možné příčiny chyby jsou nesprávný typ motoru zvolený v hardwarové konfiguraci nebo hodnoty zisku rychlostní smyčky, které jsou příliš velké. Pokud je typ motoru v hardwarové konfiguraci správný, pak snižte zisk rychlostní smyčky. Pokud problém bude i nadále nebo zisk rychlostní smyčky bude pro danou aplikaci příliš malý, spojte se s výrobním závodem.
D7	Pouze stav	Přetečení G.S. při výpočtu alfa	Osa	Interní výpočet rychlostní smyčky přesáhnul přípustné hodnoty. Hodnoty jsou omezené platným rozsahem. Tato chyba by se v prodávaném systému neměla vyskytnout. Snižte zisk rychlostní smyčky. Pokud se problémy budou vyskytovat nadále, spojte se výrobním závodem.
D8	Pouze stav	Přetečení při výpočtu zisku integrátoru	Osa	Integrační zisk proudové smyčky překračuje přípustný rozsah. Výpočet je omezený platným rozsahem. Tato chyba by se v prodávaném systému neměla vyskytnout. Pokud se chyba bude vyskytovat nadále, spojte se výrobním závodem.
D9	Pouze stav	Přetečení výpočtu Kp	Osa	Proporcionální zisk proudové smyčky překračuje přípustný rozsah. Výpočet je omezený platným rozsahem. Tato chyba by se v prodávaném systému neměla vyskytnout. Pokud se chyba bude vyskytovat nadále, spojte se výrobním závodem.
DA	Rychlé zastavení	Zjištěna chyba FPGA	Osa	Byla zjištěna chyba, když se prováděla inicializace hradlového pole programovatelného v poli (FPGA). Tato chyba se během normálního provozního stavu nemůže objevit. Pokud se chyba bude vyskytovat nadále, spojte se výrobním závodem.
		Chyby zvláštního účelu		
E2	Rychlé zastavení	Chyba přerušení DSP	Modul	Spojte se s GE Fanuc Automation
		Chyby náběhu vlečené osy		
E8	Pouze stav	Registrační vzdálenost vlečené osy (z registru parametrů) je mimo přípustný rozsah - vlečená osa přestane používat náběh zrychlení.	Osa	Když Zákaz činnosti vlečené osy = Inkrementální poloha, inkrementální vzdálenost (registrační vzdálenost) uvedená v souvisejícím registru parametrů musí být větší, než je vzdálenost zastavení. Vzdálenost zastavení závisí na aktuální rychlosti osy slave a náběhu zrychlení vlečené osy. **Záporné rychlosti osy slave vyžadují záporné registrační vzdálenosti.**
E9		Vyhrazeno – nepoužívá se v DSM314		
EA	Pouze stav	Master rychlost větší než 0.8*mez rychlosti – žádná korekce vzdálenosti	Osa	Mater rychlost při převodu na jednotky osy slave je větší než 0,8 * nakonfigurovaná mez rychlosti. Mez rychlosti se musí zvýšit nebo master rychlost se musí snížit.
EB	Rychlé zastavení	Chyba při výpočtu během zrychlování vlečené osy	Osa	Spojte se s GE Fanuc Automation

EC	Pouze stav	Doba úpravy vlečené osy není dostatečně dlouhá	Osa	Nakonfigurovaná doba úpravy náběhu je příliš malá, takže skutečná doba úpravy je delší. Doba úpravy náběhu zrychlení vlečené osy je nutno zvýšit.
ED	Pouze stav	Překročení meze rychlosti během náběhu vlečené osy	Osa	Úprava náběhu vlečené osy vyžaduje rychlost větší než 0,8 * nakonfigurovaná mez rychlosti osy, takže skutečná doba úpravy je delší než nakonfigurovaná hodnota. Zvyšte mez rychlosti, dobu úpravy nebo náběh zrychlení.
EE	Pouze stav	Překročení mezní doby během sektoru zrychlování korekce vzdálenosti vlečené osy	Osa	Úprava náběhu vyžaduje zrychlení > 64000 doba vzorkování polohové smyčky. Náběh zrychlení vlečené osy je nutno zvýšit.
		Chyby polohové smyčky		
F1	Pouze stav	Zjištěna mez polohové odchylky vlečené osy	Osa	Polohová odchylka dosáhla meze a smyčka vlečené osy již není polohově svázaná s master osou. Mez polohové odchylky je nutno zvýšit nebo je nutno použít posuv rychlostí.
F2	Pouze stav	Zjištěn stav meze rychlosti	Osa	Součet všech vstupních povelů (interní povely + master vlečená osa + lokální logika) do polohové smyčky přesahuje nakonfigurované meze rychlosti. Osa už nebude polohově svázaná s povely. Rychlosti povelů je nutno snížit nebo je nutno zvýšit mez rychlosti.
F3	Pouze stav	Hodnota poměru B vlečené osy = 0	Osa	Hodnoty poměru B vlečené osy < 0 jsou nepřípustné.
F4	Pouze stav	Hodnota poměru B vlečené osy < 0	Osa	Hodnota poměru B vlečené osy = 0 je nepřípustná.
F5	Pouze stav	Poměr B vlečené osy A:B > 32:1 nebo < 1:10000	Osa	Hodnoty poměru A / poměru B musí představovat poměr A/B v rozsahu 32:1 až 1:10000.
		Interní chyby		
FB	Pouze stav	Doba vykonávání smyčky řízení > 500 mikrosekund	Osa	Spojte se s GE Fanuc Automation
FC	Pouze stav	Vykonávání smyčky řízení > 400 mikrosekund více než 5 krát za sebou	Osa	Spojte se s GE Fanuc Automation
FD	Rychlé zastavení	Chyba systémového softwaru	Osa	Spojte se s GE Fanuc Automation
FE	Rychlé zastavení	Neznámá chyba, nepodporuje se	Osa	Chyba může indikovat vadný kabel snímače polohy – zkontrolujte kabel. Pokud kabel bude v pořádku, spojte se s GE Fanuc Automation.

Kódy systémových chyb

Pokud DSM zjistí chyby v konfiguraci, pohybovém programu nebo bloku lokální logiky, do registru Stavový kód modulu umístí chybový kód (první slovo AI). Když se vyskytne systémová chyba, DSM neprovede žádnou aktualizaci bitů %I nebo dat %AI a nebude odpovídat na žádný bit %Q nebo žádné povely %AQ.

To znamená, že bit %Q Vynulování chyby nemá žádný vliv na systémovou chybu. Systémovou chybu je možno vynulovat pouze posláním nové konfigurace do DSM.

Následující kódy systémových chyb indikují, že v konfiguračním/programovacím softwaru byla zapsaná neplatná konfigurace DSM. Pokud se vyskytne některá z těchto chyb, je nutno změnit konfiguraci a novou konfiguraci uložit do PLC. Všechny jiné chyby formátu **Dxxx**, **Exxx** nebo **Fxxx**, které nejsou uvedené v tabulce, jsou neočekávané a je nutno je hlásit do GE Fanuc Automation.

Tabulka A-2. Kódy systémových chyb

Chybový kód (hex) (x = číslo osy)	**Typ systémové chyby**	**Popis**
D008	Modul	Osa 4 není zakázaná, když Osa 1,2 = digitální servo
Dx65	Osa	Zdroj zpětné vazby je neplatný nebo není podporovaný
Dx68	Osa	Zákaz činnosti vlečené osy není podporovaný
Dx69	Osa	Režim náběhu úpravy vlečené osy není podporovaný
Dx71	Osa	Neplatný typ digitálního servomotoru
Dx81	Osa	Režim povelu analogového serva (režim krouticího momentu) není podporovaný **Poznámka:** DSM314 verze 3.0 nebo pozdější nepodporuje režim krouticího momentu.

Alarmy digitálních serv DSM (B0–BE)

Digitální servosystémy GE Fanuc α a β mají vestavěnou detekci a obvod bezpečnostního zastavení pro mnoho potenciálně nebezpečných stavů. Následující tabulka bere v potaz to, že s DSM je možno použít tři různé modely servozesilovačů, β Series, α Series SVU a α Series SVM. Tato tabulka uvádí alarmy, které konkrétní servozesilovač podporuje, a odpovídající chybový kód DSM. *Tabulkové položky, které jsou ve sloupcích zesilovače prázdné, udávají alarmy zesilovače, které konkrétní řada zesilovače nepodporuje.* Má-li se vynulovat alarm serva, je nutno provést reset zesilovače vypnutím a zapnutím napájení. Kromě toho se k vynulování chybového kódu DSM vyžaduje diskrétní povel %Q "Vynulování chyby". Alarmy zesilovače, které se nevynulují vypnutím napájení, se budou dále hlásit do DSM modulu. Krátká pasáž s lokalizací chyb servozesilovačů je uvedená na konci tabulky chybových hlášení.

Tabulka A-3. Alarmy digitálního serva DSM

Číslo chyby (Hexadecimálně)	Název alarmu serva	Popis	Zobrazení alarmu serva		
			SVM 7 SEG	SVU 7 SEG	β ALM LED
B0	HV	Přepětí DC LINK	07†	1	ON
B1	LV	Nízkonapěťové řídicí napájení	06†	2	
B2	DBRLY	Porucha obvodu dynamické brzdy		7	
		† SVM PSM DC LINK nízké nabíjení	05†		
B3	LVDC	Nízkonapěťové DC LINK	04†	3	ON
B4	OH	Přehřátí zesilovače	03†		ON
B5	FAL	Porucha chladicího ventilátoru	02†		ON
B6		† SVM PSM IPM Alarm nebo nadměrný proud	01†		
B7	DCSW	Regenerační obvod - alarm poruchy	08†	4	ON
	DCOH	Regenerační obvod - alarm vybití		5	
B9	LV5V	Není +5 V SVM servomodulu	2		
BA	IPML	IPM nadměrný proud, vysoká teplota nebo nízké napětí	8.	8.	
	IPMM		9.	9.	
	IPMN		A.	A.	
	IPMLM		b.	b.	
	IPMMN	(osa L, osa M, osa N, osy L & M, osy M & N, osy N & L nebo osy L & M & N)	C.	C.	
	IPMNL		d.	d.	
	IPMLMN		E.	E.	
BB	LVDC	Nízké DC LINK servomodulu SVM	5		
BD	FAL	Porucha ventilátoru servomodulu SVM	1		
BE	HCL	Abnormálně vysoký proud motorem	8	8	ON
	HCM		9	9	
	HCN		A	A	
	HCLM	(osa L, osa M, osa N, osy L & M, osy M & N, osy N & L nebo osy L & M & N)	b	b	
	HCMN		C	C	
	HCNL		d	d	
	HCLMN		E	E	

† Dvousegmentový displej na modulu napájení SVM (PSM) indikuje alarm napájení.

Lokalizace chyb digitálního serva:

Níže uvedené postupy jsou určené jako pomůcka při hledání problémů souvisejících s různými alarmy serva. Pokud se tyto návody nebudou hodit pro daný případ nebo alarm neřeší, vyměňte servozesilovač nebo požádejte o pomoc Hotline GE Fanuc. Příslušný návod pro údržbu nebo popisný manuál zesilovače a motoru budou obsahovat podrobnější postupy lokalizace chyb.

HV (vysoké napětí) alarm: Tento alarm se objeví, když stejnosměrná úroveň vysokého napětí (DC LINK) bude abnormálně vysoká.

1. Střídavé napětí přiváděné do zesilovače může být vyšší než je jmenovité vstupní napětí. U zesilovače β Series by třífázové napájecí napětí mělo být v rozmezí 200 V stř. až 240 V stř.
2. Externí regenerační odpor může být zapojený nesprávně. Pečlivě zkontrolujte zapojení regeneračního odporu k zesilovači. Přesvědčte se, že hodnota regeneračního odporu je v rozmezí 20% jmenovité hodnoty. Pokud odpor nebude v toleranci, regenerační jednotku vyměňte.
3. Regenerační odpor možná není schopný odvádět nadměrné vygenerované teplo. Ověřte výpočty pro volbu regenerační vybíjecí jednotky a podle potřeby vyměňte odpor za vyšší výkon. Snížení hodnoty zrychlení a zisky polohové smyčky (větší hodnota *časové konstanty polohové smyčky*) dále sníží úroveň regenerovaného napětí.

LVDC (nízké napětí DC Link): Tento alarm se objeví, když stejnosměrná úroveň vysokého napětí (DC LINK) bude abnormálně nízká.

1. Střídavé napětí přiváděné do zesilovače může chybět nebo mít nižší hodnotu než je jmenovité vstupní napětí. U zesilovače β Series by třífázové napájecí napětí mělo být v rozmezí 200 V stř. až 240 V stř. Ověřte, že na vstupních přívodech zesilovače (L1, L2 a L3) je správná úroveň střídavého napětí..

DCOH nebo DCSW (alarm regenerace): Alarm DCOH se objeví, když teplota regeneračního odporu bude příliš vysoká. Alarm DCSW indikuje problémy ve spínací části regeneračního obvodu.

1. Pokud se externí regenerační odpor *nepoužívá*, přesvědčte se, že vstup čidla teploty do zesilovače je zkratovaný nebo přemostěný. Na konektoru CX11-6 musí být nainstalovaná propojka T604 zesilovače β Series.
2. Externí regenerační odpor může být zapojený nesprávně. Pečlivě zkontrolujte zapojení regeneračního odporu k zesilovači. Přesvědčte se, že hodnota regeneračního odporu teplotního čidla je při pokojové teplotě blízko nuly. Pokud teplotní čidlo bude indikovat rozpojený stav, regenerační odpor vyměňte.
3. Regenerační odpor možná není schopný odvádět nadměrné vygenerované teplo. Ověřte výpočty pro volbu regenerační vybíjecí jednotky a podle potřeby vyměňte odpor za vyšší výkon. Snížení hodnoty zrychlení a zisky polohové smyčky (větší hodnota *časové konstanty polohové smyčky*) dále sníží úroveň regenerovaného napětí.

OH (alarm přehřátí): Teplota chladiče zesilovače je příliš vysoká nebo je nadměrná teplota motoru.

1. Okolní teplota může být příliš vysoká, zvažte instalaci chladicího ventilátoru servomotoru. GE Fanuc dodává sadu pro většinu motorů FANUC.
2. Motor může pracovat při porušení omezení činitele využití. Vypočítejte potřebnou dobu chlazení na základě křivek činitele využití uvedených pro daný motor.
3. Motor může být přetížený. Zkontrolujte nadměrné tření nebo sevření ve stroji.
4. U všech výše uvedených problémů nechejte zesilovač chladit alespoň deset minut při minimálním nebo žádném zatížení motoru a pak proveďte reset zesilovače vypnutím a zapnutím napájení.

FAL (alarm ventilátoru): Porucha chladícího ventilátoru.

1. Zkontrolujte překážky nebo znečištění ventilátoru. Při vypnutém napájení zkuste ručně otáčet ventilátorem.
2. U systému se zesilovačem typu SVM modul napájení (PSM) a modu servozesilovače mají chladicí ventilátor. Kód alarmu indikuje, která jednotka má poruchu.
3. Některé zesilovače mají jednotky, které je možno vyměnit v poli. Pokud ventilátorová jednotka na výměnu není k dispozici, vyměňte zesilovač.

HC, HCL, atd. (alarm vysokého proudu): Nadměrný proud motorem. U zesilovačů α Series přípona (L, M, N, atd.) indikuje, ve které ose se vyskytnul alarm.

1. Zapojení napájení motoru (U, V a W) může být zkratované na zem nebo připojené s nesprávnou fází. Zkontrolujte zapojení a připojení. Zkontrolujte, jestli servomotor nemá zkrat na kostru motoru. V případě zkratu ho vyměňte.
2. Může být nakonfigurovaný nesprávný kód typu motoru nebo v parametrech ladění mohou být nadměrné hodnoty. Ověřte, že je nakonfigurovaný správný motor a nižší hodnoty zisku.
3. Manuál údržby zesilovače popisuje postup pro sledování proudových signálů motoru (IR a IS). Pokud tvar průběhu bude nenormální, zesilovač vyměňte. Pokud zjistíte nadměrnou hlučnost, zkontrolujte zemnění a zejména uzemnění stínění povelových kabelů (K1) k zesilovači.
4. Motor může pracovat při porušení omezení činitele využití. Vypočítejte potřebnou dobu chlazení na základě křivek činitele využití uvedených pro daný motor.
5. Motor může být přetížený. Zkontrolujte nadměrné tření nebo sevření ve stroji.
6. U všech výše uvedených problémů nechejte zesilovač chladit alespoň deset minut při minimálním nebo žádném zatížení motoru a pak proveďte reset zesilovače vypnutím a zapnutím napájení.

LV (alarm nízkého napětí řídicího napájení): Řídicí napětí používané k ovládání nízkonapěťového obvodu v zesilovači je příliš nízké.

1. Zesilovače typu α Series SVU se dodávají z výroby se zkratovacími propojkami nastavenými k použití jednofázového napájení zesilovače napětím 220 V stř. Zkratovací propojky je možno odstranit a připojit řídicí napájení 220 V stř. odděleně. Přesvědčte se, že na svorkách L1C a L2C je minimální napětí 200 V stř. pro implicitní instalaci nebo na konektoru CX3 (Y Key) pro oddělené řídicí napájení.
2. Zkontrolujte pojistku zesilovače. Pokud pojistka bude přepálená, po zkontrolování napětí řídicího napájení jí vyměňte za novou. Pokud se přepálí i tato druhá pojistka, vyměňte zesilovač.

DBRLY (porucha relé dynamické brzdy): Tento alarm indikuje, že kontakty brzdného relé jsou spečené. Zesilovač okamžitě vyměňte.

IPML, IPMM, atd. (alarm IPM): Inteligentní napájecí modul (IPM) je vysokonapěťové spínací zařízení v zesilovači. IPM může zjistit stav nadměrného proudu nebo nízkého napětí ve spínacím obvodu napájení. Přípona (L, M, N, atd.) indikuje, ve které ose se vyskytnul alarm.

1 Zapojení napájení motoru (U, V a W) může být zkratované na zem nebo připojené s nesprávnou fází. Zkontrolujte zapojení a připojení. Zkontrolujte, jestli servomotor nemá zkrat na kostru motoru. V případě zkratu ho vyměňte.

2 Může být nakonfigurovaný nesprávný kód typu motoru nebo v parametrech ladění mohou být nadměrné hodnoty. Ověřte, že je nakonfigurovaný správný motor a nižší hodnoty zisku.

3 Manuál údržby zesilovače popisuje postup pro sledování proudových signálů motoru (IR a IS). Pokud tvar průběhu bude nenormální, zesilovač vyměňte. Pokud zjistíte nadměrnou hlučnost, zkontrolujte zemnění a zejména uzemnění stínění povelových kabelů (K1) k zesilovači.

4 Motor může pracovat při porušení omezení činitele využití. Vypočítejte potřebnou dobu chlazení na základě křivek činitele využití uvedených pro daný motor.

5 Motor může být přetížený. Zkontrolujte nadměrné tření nebo sevření ve stroji.

6 U všech výše uvedených problémů nechejte zesilovač chladit alespoň deset minut při minimálním nebo žádném zatížení motoru a pak proveďte reset zesilovače vypnutím a zapnutím napájení.

LED kontrolky

Na modulu DSM314 je sedm kontrolek LED, která udávají stav. Tyto kontrolky LED jsou následující:

STAT Normálně SVÍTÍ. BLIKÁNÍ indikuje chybu provozu. Bliká *pomalu* (čtyřikrát/sekundu) v případě pouze stavových chyb. Bliká *rychle* (osmkrát/sekundu) v případě chyb, které způsobí zastavení serva.

SVÍTÍ: Když LED bude trvale SVÍTIT, DSM314 funguje správně. Normálně by tato kontrolka LED měla vždy SVÍTIT.

NESVÍTÍ: Když tato LED bude ZHASNUTÁ, DSM314 nefunguje. Je to výsledkem hardwarové nebo softwarové závady, která brání, aby bylo možno modul zapnout.

BLIKÁ: Když kontrolka LED bude BLIKAT, signalizuje se chybový stav.

<u>*Konstantní rychlost, LED kontrolka CFG SVÍTÍ*</u>:

LED bliká pomalu (čtyřikrát/sekundu) v případě pouze stavových chyb a rychle (osmkrát/sekundu) v případě chyb, které způsobí zastavení serva. Stavový bit %I *Chyba modulu* bude v jedničce. Chybový kód (hexadecimální formát) se uloží do slova %AI *Stavový kód modulu* nebo některého slova %AI *Chybový kód osy*.

<u>*Konstantní rychlost, LED kontrolka CFG bliká*</u>:

Pokud LED kontrolky STAT a CFG blikají **společně** konstantní rychlostí, modul DSM314 je v bootovacím režimu a čeká na načtení nového firmwaru. Pokud obě LED kontrolky STAT a CFG blikají **střídavě** konstantní rychlostí, firmware DSM314 zjistil překročení doby softwarového hlídacího obvodu v důsledku chyby hardwaru nebo softwaru.

<u>*Nepravidelná rychlost, LED kontrolka CFG NESVÍTÍ*</u>:

Pokud k tomuto dojde hned po zapnutí napájení, pak se zjistila chyba hardwaru nebo softwaru. LED kontrolka modulu STAT bude blikáním zobrazovat dvě chybová čísla oddělené krátkou prodlevou. Čísla jsou určená počtem blikání v obou sekvencích. Poznamenejte si tato čísla a vyžádejte si u GE Fanuc informace, jakým způsobem problém opravit.

OK LED kontrolka OK indikuje aktuální stav modulu DSM314.

SVÍTÍ: Když LED bude trvale SVÍTIT, DSM314 funguje správně. Normálně by tato kontrolka LED měla vždy SVÍTIT.

NESVÍTÍ: Když tato LED bude ZHASNUTÁ, DSM314 nefunguje. Je to výsledkem hardwarové nebo softwarové závady, která brání, aby bylo možno modul zapnout.

CFG Tato kontrolka LED bude SVÍTIT, když konfigurace modulu přišla z PLC.

EN1 Když tato LED bude SVÍTIT, výstup relé povolení pohonu osy 1 je aktivní.

EN2 Když tato LED bude SVÍTIT, výstup relé povolení pohonu osy 2 je aktivní.

EN3 Když tato LED bude SVÍTIT, výstup relé povolení pohonu osy 3 je aktivní.

EN4 Když tato LED bude SVÍTIT, výstup relé povolení pohonu osy 4 je aktivní.

Dodatek

B

Instrukce požadavku na komunikaci DSM314

Tento dodatek popisuje dva typy žebříkových instrukcí požadavků na komunikaci (v tomto dodatku zkracováno jako COMM REQ) používaných v DSM314:

- **Typ načtení parametru:** Používá se k načtení paměti parametrů DSM. Výhodou instrukce COMM REQ je, že každá z nich může načíst až 16 parametrů, a v jednom cyklu PLC je možno použít více instrukcí COMM REQ. Pro porovnání, každý okamžitý povel načtení parametrů může načíst pouze jeden parametr na každý cyklus PLC s jedním až čtyřmi okamžitými povely načtení parametrů přípustnými na cyklus PLC v závislosti na počtu slov %AQ (který naopak závisí na počtu nakonfigurovaných os – viz tabulka 5-8). Proto COMM REQ je nejvhodnější k načtení několika nebo mnoha parametrů a okamžitý povel načtení parametrů je nejvhodnější, pokud je zapotřebí načíst málo parametrů (jeden až čtyři).

- **Typ tabulky uživatelských dat (UDT):** Používá se pro přístup tabulky uživatelských dat lokální logiky DSM314. Tabulka uživatelských dat je paměťová oblast o velikosti 8192 bajtů, kterou programy lokální logiky mohou použít pro uložení a načtení dat. UDT COMM REQ může kopírovat data buď z paměti slov PLC do UDT nebo z UDT do paměti slov PLC.

COMM REQ se v zásadě používá v žebříkovém programu PLC Series 90-30 pro komunikaci s různými inteligentními moduly. Tento dodatek nejdříve probírá instrukci COMM REQ obecně v části 1 a 2, pak v části 3 - 5 probírá, jak ho konkrétně použít v modulu DSM314. Tento dodatek je rozdělený na následující části:

- Část 1: Úvod do požadavku na komunikaci
- Část 2: Žebříková instrukce COMM REQ
- Část 3: COMM REQ tabulky uživatelských dat (UDT)
- Část 4: COMM REQ načtení parametrů
- Část 5: Příklad žebříkové logiky COMM REQ (používá COMM REQ načtení parametrů)

Část 1: Úvod do požadavku na komunikaci

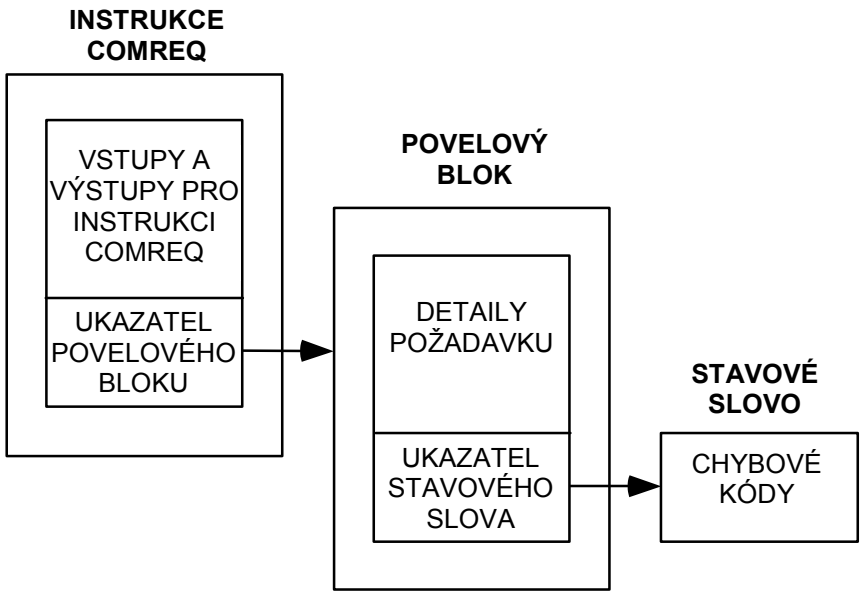
Požadavek na komunikaci používá parametry *žebříkové instrukce COMM REQ* a související *Povelový blok* k definování charakteristik požadavku. Související *Stavové slovo* hlásí výsledky každého požadavku.

Struktura požadavku na komunikaci

Požadavek na komunikaci se skládá ze tří hlavních částí:

- Žebříkové instrukce COMM REQ
- Povelového bloku, což je blok paměti PLC (obvykle paměť %R, která obsahuje instrukce a data pro COMM REQ
- Stavového slova, což je slovo paměti, do kterého se zapisují stavové/chybové kódy.

Následující obrázek znázorňuje vztah mezi těmito částmi:



Obrázek B-1. Struktura COMM REQ

Žebříková instrukce COMM REQ: Žebříková instrukce COMM REQ je hlavní struktura používaná k zapsání konkrétních informací o požadavku na komunikaci. Tyto informace zahrnují sestavu a umístění pozice modulu DSM souvisejícího s požadavkem a parametr, který ukazuje na počáteční adresu povelového bloku. Všimněte si, že při programování této instrukce je nutno data povelového bloku inicializovat v žebříkovém programu před tím, než se vykoná příčka obsahující instrukci COMM REQ.

Povelový blok: Povelový blok se skládá z několika slov paměti PLC, která obsahuje doplňující informace o požadavku na komunikaci. Tyto informace zahrnují časovací parametry, ukazatel na Stavové slovo, Blok dat, typy a velikosti paměti a konkrétní povelový kód. Blok dat udává směr přesunu dat (přes povelový kód) a umístění a typ dat, která se mají přesunout.

Stavové slovo: Stavové slovo je jedno místo v datové paměti PLC, do kterého CPU hlásí výsledek požadavku na komunikaci. Adresu stavového slova zadává v povelovém bloku uživatel. Následující tabulka uvádí stavové kódy hlášené ve stavovém slově:

Tabulka B-1. Kódy stavového slova COMM REQ DSM

Kódy stavového slova COMM REQ DSM			
Název kódu	**Číslo kódu**	**Popis**	**Co dělat**
IOB_SUCCESS	1	Všechny komunikace pokračují normálně.	Nevyžaduje se.
IOB_PARITY_ERR	-1	Vyskytla se chyba parity během komunikace s přídavnou sestavou.	Zkuste znovu. Zkontrolujte hardware – expanzní kabely, DSM modul, atd.
IOB_NOT_COMPL	-2	Po skončení komunikace modul neindikoval, že komunikace skončila.	Zkuste znovu. Ověřte parametry COMM REQ.
IOB_MOD_ABORT	-3	Modul přerušil komunikaci.	Zkuste znovu. Ověřte parametry COMM REQ.
IOB_MOD_SYNTAX	-4	Modul indikoval, že poslaná data nebyla ve správné sekvenci.	Ověřte parametry COMM REQ.
IOB_NOT_RDY	-5	Bit RDY ve stavu modulu nebyl aktivní.	Zkuste znovu. Zkontrolujte DSM modul.
IOB_TIMEOUT	-6	Uplynula maximální doba odezvy, aniž by z modulu přišla odezva.	Zkontrolujte DSM modul. Ověřte parametry COMM REQ.
IOB_BAD_PARAM	-7	Některý z předaných parametrů je neplatný.	Ověřte parametry COMM REQ.
IOB_BAD_CSUM	-8	Kontrolní součet, který přišel z modulu protokolu DMA, nesouhlasí s přijatými daty.	Zkuste znovu. Zkontrolujte, jestli instalace je správně uzemněná, stíněná, jestli je potlačený šum, atd.
IOB_OUT_LEN_CHGD	-9	Výstupní délka modulu se změnila, takže normální zpracování záznamu odpovědi by se nemělo provést.	Ověřte parametry COMM REQ.

Nápravný krok

Typ nápravného kroku, který je nutno provést, závisí na aplikaci. Pokud se chyba vyskytne během fáze spouštění nebo ladění vývoje žebříkové logiky, nejvhodnější radou je "Ověřte parametry COMM REQ". Totéž platí, pokud se chyba vyskytne hned po úpravě programu. Pokud se ale chyba vyskytne na vyzkoušené aplikaci, která běžela úspěšně, problém nejspíš bude v hardwaru. Je nutno projít tabulku chyb PLC a zjistit další informace při lokalizaci chyb podle stavového slova.

Monitorování stavového slova

Detekce a ošetření chyb

Jak je uvedeno v tabulce výše, do stavového slova se vrátí hodnota 1, pokud komunikace bude pokračovat normálně, ale pokud se zjistí nějaký chybový stav, vrátí se záporná hodnota. Pokud budete v žebříkovém program vyžadovat detekci chyb, můžete použít instrukci porovnání Menší než (LT) ke zjištění, jestli hodnota stavového slova je záporná (menší než nula). Příklad je uveden níže na následujícím obrázku. Pokud se objeví chyba, výstup funkce Menší než (Q) přejde do jedničky. Cívku řízenou výstupem je možno použít k povolení logiky ošetření chyby nebo hlášení chyby.

Výstup FT z COMM REQ popisovaný dále v tomto dodatku při určitých chybách přejde do jedničky a je možno ho také použít k detekci chyb. Kromě toho logika hlášení může monitorovat stavové slovo pro zobrazení na zařízení rozhraní obsluhy, kdy kód stavového slova odpovídá příslušnému chybovému hlášení, které se zobrazí na obrazovce obsluhy. Pokud se například ve stavovém slově zjistí -1, zobrazí se hlášení, které bude sdělovat něco jako "Chyba komunikace s DSM modulem v expanzní sestavě".

Chcete-li dynamicky kontrolovat stavové slovo, před každým vykonáním souvisejícího COMM REQ do stavového slova zapište nevýznamné kladné číslo (obvykle se používá 0 nebo 99). Pokud se pak instrukce vykoná úspěšně, CPU sem zapíše číslo 1. Tento způsob vám sdělí, že pokud zde je 1, poslední COMM REQ se určitě vykonal úspěšně a že 1 není pouze “pozůstatkem” z předchozího vykonání. V příkladu uvedeném na konci tohoto dodatku se číslo 99 přemístilo do stavového slova (%R0195) v příčce před příčkou, která obsahuje instrukci COMM REQ.

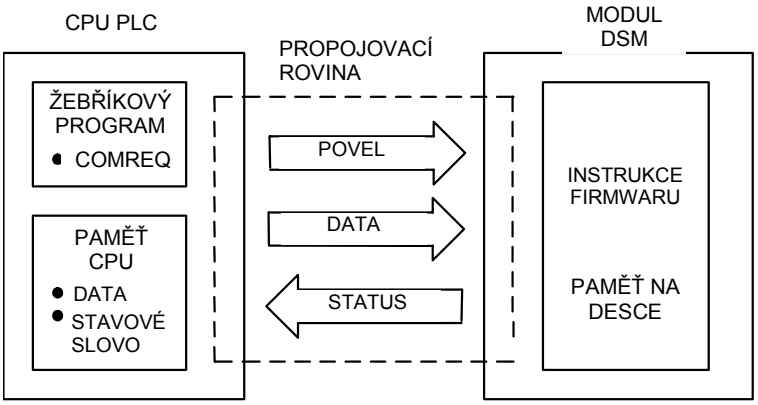
Když se bude používat více COMM REQ pro DSM, doporučuje se ověřit úspěšnou komunikaci u každé instrukce před povolením další. Monitorování stavového slova je jedním ze způsobů, jak to provést.

Ověření, že DSM obdrželo správná data

U kritických aplikací může být radno ověřit, že hodnoty určitého parametru byly předané správně do modulu DSM před tím, než činnost bude smět pokračovat. K tomu účelu nejdříve naprogramováním okamžitého povelu %AQ *Volba dat návratu* zadejte číslo parametru DSM, který se má načíst do příslušného dvojitého slova˘%AI *Uživatelem volitelná data* (na každou osu je jedno dvojité slovo %AI *Uživatelem volitelná data*). **Všimněte si, že než v PLC budou k dispozici nová *Uživatelem volitelná data*, musí uplynout minimálně tři cykly PLC nebo 20 milisekund, podle toho co je delší.** Aby bylo zaručeno, že tento požadavek bude splněný, je nutné naprogramovat nějakou logiku časové prodlevy. Pak naprogramujte instrukci Rovná se typu celého čísla s dvojnásobnou délkou pro porovnání hodnoty vrácené ve slově dvojnásobné délky *Uživatelem volitelná data* s poslanou hodnotou. V části 5 tohoto dodatku je ukázaný příklad, jak to provést. Více informací o slově *Uživatelem volitelná data* a povelu *Volba dat návratu* najdete v kapitole 5.

Činnost požadavku na komunikaci

Následující obrázek znázorňuje tok informací z CPU PLC do modulu DSM:



Obrázek B-2. Činnost požadavku na komunikaci DSM

Požadavek na komunikaci se vyvolá, když se během čtení PLC aktivuje žebříková instrukce COMM REQ. Pak se detaily požadavku na komunikaci skládající se z povelu a dat pošlou z CPU PLC do modulu DSM.

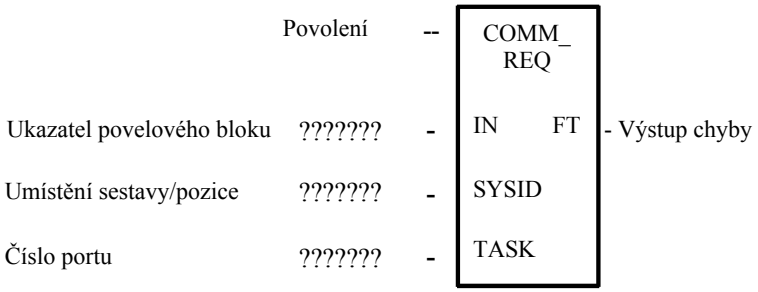
- V případě COMM REQ pro načtení parametrů data povelu udávají, že se data mají načíst z paměti PLC a zkopírovat do určených paměťových míst parametru DSM.
- V případě COMM REQ pro UDT data povelu buď udávají, že se data mají načíst z paměti PLC a zkopírovat do určeného paměťového segmentu UDT nebo načíst z konkrétního paměťového segmentu UDT a zkopírovat do paměti PLC.

Pořadí, ve kterém se tyto instrukce posílají, je kritické, takže Povelový blok pro každý typ instrukce COMM REQ je nutno naprogramovat přesně tak, jak je uvedeno dále v tomto dodatku. Na obrázku výše je ukázaný modul DSM v sestavě CPU a komunikace se odehrává přes propojovací rovinu PLC. Pokud modul DSM bude umístěný v expanzní nebo vzdálené sestavě, povely a data se budou posílat přes propojovací rovinu sestavy CPU, přes expanzní nebo vzdálený kabel do sestavy obsahující modul DSM a přes propojovací rovinu této sestavy do DSM.

Na závěr každého požadavku CPU PLC hlásí stav požadavku do stavového slova, což je místo v paměti PLC, které je určeno ukazatelem stavového slova v povelovém bloku.

Část 2: Žebříková instrukce COMM REQ

Tato část popisuje instrukci COMM REQ obecně. Více informací najdete v *Referenční příručce instrukční sady CPU PLC Series 90-30/20/Micro*, GFK-0467L nebo pozdější. Požadavek na komunikaci začíná, když se aktivuje žebříková instrukce COMM REQ. Žebříková instrukce COMM REQ má čtyři vstupy a jeden výstup:



Obrázek B-3. Žebříková instrukce COMM REQ

Popis každého vstupu a výstupu je uvedený dále. Je důležité si uvědomit, že ukazatel povelového bloku ukazuje na jiné paměťové místo, kde musíte zapsat další informace o požadavku na komunikaci.

Povolení vstupu: Musí být v logické 1, aby instrukce COMM REQ byla povolená. Doporučuje se, aby logika povolení byl kontakt přechodové ("jednorázové") cívky.

IN: Paměťové místo prvního slova povelového bloku. Může to být libovolná adresa v paměti typu slova (%R, %AI nebo %AQ).

SYSID: Hexadecimální hodnota, která představuje umístění sestavy a pozice modulu, který COMM REQ adresuje. Horní bajt (první dvě číslice hexadecimálního čísla) obsahuje číslo sestavy a dolní bajt obsahuje číslo pozice. Následující tabulka ukazuje některé příklady:

Příklady SYSID

Sestava	***Pozice***	***Hodnota hexadecimálního slova***
0	4	0004h
3	4	0304h
2	9	0209h

TASK: Pro modul DSM by zde měla být vždy zapsaná 0.

Výstup chyby: Výstup funkce FT (chyba) dává výstup na volitelnou logiku, která může ověřit úspěšné dokončení požadavku na komunikaci. Výstup FT může mít tři stavy:

Tabulka B-2. Pravdivostní tabulka výstupu FT instrukce COMM REQ

Výstup FT		
Stav vstupu povolení	**Existuje chyba?**	**Výstup FT**
Aktivní	Ne	Nula
Aktivní	Ano	Jednička
Neaktivní	Bez vykonání	Nula

- Výstup FT se nastaví do jedničky, když:
    - Zadaná adresa cíle neexistuje (například zadání sestavy 1, když systém používá pouze sestavu 0).
    - Zadané číslo úlohy je pro zařízení neplatné (číslo TASK musí být pro DSM vždy 0).
    - Délka dat je nastavená na 0.

Požadavky a doporučení pro programování COMM REQ DSM

- Doporučuje se, aby instrukce COMM REQ pro DSM byla povolená pomocí kontaktu přechodové cívky.
- Pokud v žebříkovém programu budete používat více než jeden COMM REQ pro DSM, před vykonáním další instrukce se přesvědčte, že předchozí COMM REQ se vykonal úspěšně. To je možno provést zkontrolováním stavového slova a výstupu FT (chyba), jak bylo vysvětleno dříve v tomto dodatku v odstavci s názvem "Monitorování stavového slova".
- Jak je vidět z tabulky výše, výstup FT bude v nule, pokud Vstup povolení nebude aktivní. To znamená, že pokud COMM REQ bude povolený přechodovým (jednorázovým) kontaktem a vyskytne se chyba, výstup FT bude v jedničce pouze po dobu jednoho čtení PLC. Proto chcete-li "zachytit" chybu, můžete naprogramovat výstup chyby jako SET cívku, která se na konci čtení neresetuje automaticky. Další logika pak bude zapotřebí k resetování chybového výstupu cívky po potvrzení chyby.
- Programovací zařízení, například SET cívka, na výstupu FT instrukce COMM REQ je volitelné.
- Všimněte si, že COMM REQ Series 90-30 (na rozdíl od mnoha jiných instrukcí PLC Series 90-30) nemá výstup OK.
- Data povelového bloku je nutné inicializovat před vykonáním instrukce COMM REQ. Protože pořadí normálního cyklu PLC je od spodu, inicializace povelového bloku v dřívější příčce (nebo příčkách), která obsahuje COMM REQ, tento požadavek usnadní. Viz příklad na konci tohoto dodatku.
- Doporučení: Pokud budete používat instrukce MOVE k načtení hodnot do registrů povelového bloku, k načtení hexadecimálního čísla použijte MOVE typu slova a k načtení dekadického čísla použijte MOVE typu celého čísla. Použití tohoto uvidíte v příkladu na konci tohoto dodatku pro COMM REQ načtení parametru, kde kód E501h se načte pomocí instrukce MOVE typu slova a zbývající dekadické hodnoty se načtou pomocí instrukcí MOVE typu celého čísla.

Část 3: COMM REQ tabulky uživatelských dat (UDT)

DSM314 má paměťovou oblast v délce 8192 bajtů nazývanou Tabulka uživatelských dat (UDT), která je určena pro použití s programy lokální logiky (LL). Programy LL můžou adresovat celou tuto paměť nebo její část k ukládání a načítání dat. UDT je užitečná pro ukládání a načítání velkých objemů dat, například dávky dat nastavení.

CPU PLC může zapisovat nebo číst z UDT přes tabulku uživatelských instrukce požadavku na komunikaci (COMM REQ pro UDT) v žebříkovém programu PLC. Jeden COMM REQ pro UDT přečte nebo zapíše najednou 2048 bajtů paměti. Proto UDT je logicky rozdělená na čtyři segmenty po 2048 bajtech nazývané segmenty 1-4, které COMM REQ pro UDT může adresovat samostatně. Pro každý ze čtyř segmentů, celkem 8 možných povelů COMM REQ pro UDT, existuje jednoznačný povel pro čtení a jednoznačný povel pro zápis.

Vlastnosti a informace o použití tabulky uživatelských dat COMM REQ

- Provádí čtení a zápis 2 K (2048) bajtů najednou do tabulky uživatelských dat lokální logiky. Jiná hodnota je nepřípustná.
- Pracuje pouze s modulem DSM314 (nebude fungovat s DSM302)
- Nelze použít k načtení dat parametrů do DSM314
- Tato instrukce prodlouží dobu čtení PLC (cyklu) asi o 15 ms na jedno čtení, pokud parametr řízení cyklu komunikačního okna PLC bude nastavený na DOKONČENO (úplný). Pokud parametr řízení cyklu komunikačního okna PLC bude nastavený na OMEZENÝ, COMM REQ se vykoná v několika čteních s menším dopadem na dobu čtení. COMM REQ se však pravděpodobně nevykoná opakovaně – vykoná se, pouze když je nutno změnit data. Proto pokud byl poslaný při prvním čtení nebo během nastavování úlohy, nebude mít dopad, když aplikace bude běžet.
- Aby se předešlo konfliktům při přístupu do paměti, doporučuje se nepoužívat cyklický podprogram během doby, kdy je tento COMM REQ aktivní.
- Tento COMM REQ nepodporuje diskrétní paměť jako svůj typ dat PLC.

Povelový blok COMM REQ pro UDT

Tabulka B-3. Tabulka uživatelských dat povelového bloku

Tabulka uživatelských dat povelového bloku COMM REQ pro modul DSM314		
Popis	**Offset adresy**	**Číslo a hodnota slova**
Délka záhlaví bloku dat	Adresa + 0	Slovo 1, vždy nastaveno na 4
Příznak ČEKAT/NEČEKAT	Adresa + 1	Slovo 2, vždy nastaveno na 0
Paměť stavového slova (viz typy kódů paměti stavového slova v tabulce níže)	Adresa + 2	Slovo 3, volí uživatel (viz tabulku kódů typů paměti níže)
Offset ukazatele stavu	Adresa + 3	Slovo 4, volí uživatel
Hodnota prodlevy při nečinnosti	Adresa + 4	Slovo 5, vždy nastaveno na 0
Maximální doba komunikace	Adresa + 5	Slovo 6, vždy nastaveno na 0
Kód povelu	Adresa + 6	Slovo 7, viz tabulku kódů povelů
Velikost dat parametrů, v bajtech	Adresa + 7	Slovo 8, vždy nastaveno na 2048
Typ paměti pro data PLC	Adresa + 8	Slovo 9, volí uživatel (viz tabulku kódů typů paměti níže)
Počátek dat PLC (offset dat)	Adresa + 9	Slovo 10, volí uživatel

Délka bloku dat (slovo 1): Délka části záhlaví bloku dat povelového bloku. Musí být nastavena na 4. Záhlaví bloku dat je uloženo ve slovech 7 až 10 povelového bloku.

Příznak ČEKAT/NEČEKAT (slovo 2): Pro DSM se vždy musí nastavit na logickou **nulu**.

Typ paměti stavového slova (slovo 3): Toto slovo udává typ paměti, která se použije pro stavové slovo. Každý typ paměti má své specifické číslo kódu uvedené v tabulce kódů typů paměti níže. Pokud například budete chtít pro stavové slovo použít paměť %R, do tohoto slova vložte buď dekadické číslo 8 nebo hexadecimální číslo kódu 08h.

Všimněte si, že pokud zvolíte diskrétní typ paměti (%I nebo %Q), stavovému slovu se přiřadí 16 po sobě jdoucích bitů počínaje adresou uvedenou ve slově offsetu ukazatele stavového slova popsaného níže.

Tabulka B-4. Kódy typů paměti stavového slova

Kódy typů paměti stavového slova COMM REQ			
Zkratka typu paměti	**Typ paměti**	**Číslo zapisovaného kódu**	
		Dekadicky	**Hexadecimálně**
%I	Tabulka diskrétních vstupů	70	46h
%Q	Tabulka diskrétních výstupů	72	48h
%R	Paměť registrů	8	08h
%AI	Tabulka analogových vstupů	10	0Ah
%AQ	Tabulka analogových výstupů	12	0Ch

Offset ukazatele stavového slova (slovo 4): Toto slovo obsahuje offset uvnitř zvoleného typu paměti. ***Poznámka: Offset ukazatele stavového slova je číslo začínající na nule.*** To znamená, že

od čísla adresy, kterou chcete zadat, je nutno odečíst jedničku. Například chcete-li zvolit %R0001, zapište nulu (1 - 1 = 0). Nebo pokud chcete zadat %R0100, zapište 99 (100 - 1 = 99). Všimněte si, že typ paměti, v tomto příkladu %R, je zadaný předchozím slovem (viz vysvětlení "Typ paměti ukazatele stavového slova" výše).

Hodnota prodlevy při nečinnosti (slovo 5): Protože DSM používá vždy režim NEČEKAT (příznak ČEKAT/NEČEKAT se vždy nastaví na nulu), tento parametr hodnoty prodlevy při činnosti se nepoužívá pro DSM. Nastavte ho na nulu.

Maximální doba komunikace (slovo 6): Protože DSM používá vždy režim NEČEKAT (příznak ČEKAT/NEČEKAT se vždy nastaví na nulu), tento parametr maximální doby komunikace se pro DSM nepoužívá. Nastavte ho na nulu.

Kód povelu (slovo 7): Použijte jeden z osmi kódů povelu z následující tabulky. Kódy povelů jsou uvedené jako hexadecimální čísla.

Tabulka B-5. Kódy povelů COMM REQ pro UDT

Povely tabulky uživatelských dat COMM REQ (UDT)	
Kód povelu	**Popis povelu**
D001h	Zápis do UDT segmentu 1
D101h	Zápis do UDT segmentu 2
D201h	Zápis do UDT segmentu 3
D301h	Zápis do UDT segmentu 4
D804h	Čtení z UDT segmentu 1
D904h	Čtení z UDT segmentu 2
DA04h	Čtení z UDT segmentu 3
DB04h	Čtení z UDT segmentu 4

Velikost datového segmentu UDT (slovo 8): Udává velikost paměti, v bajtech, adresovaného segmentu UTP. Tato hodnota musí být vždy 2048 bajtů (800h hexadecimálně).

Typ datové paměti PLC (slovo 9): Toto slovo udává typ paměti, která se použije pro data PLC. Každý typ paměti má specifické číslo kódu uvedené v tabulce kódů typů paměti níže. Pokud například budete chtít zadat paměť %R, do tohoto slova vložte buď dekadické číslo 8 nebo hexadecimální číslo 08h.

Poznámka

COMM REQ pro UDT nepodporuje diskrétní paměť (%I nebo %Q) pro typ datové paměti PLC.

Tabulka B-6. Kódy typu datové paměti PLC pro COMM REQ pro UDT

Kódy typu datové paměti PLC pro COMM REQ pro UDT			
Zkratka typu paměti	**Typ paměti**	**Číslo zapisovaného kódu**	
		Dekadicky	**Hexadecimálně**
%R	Paměť registrů	8	08h
%AI	Tabulka analogových vstupů	10	0Ah
%AQ	Tabulka analogových výstupů	12	0Ch

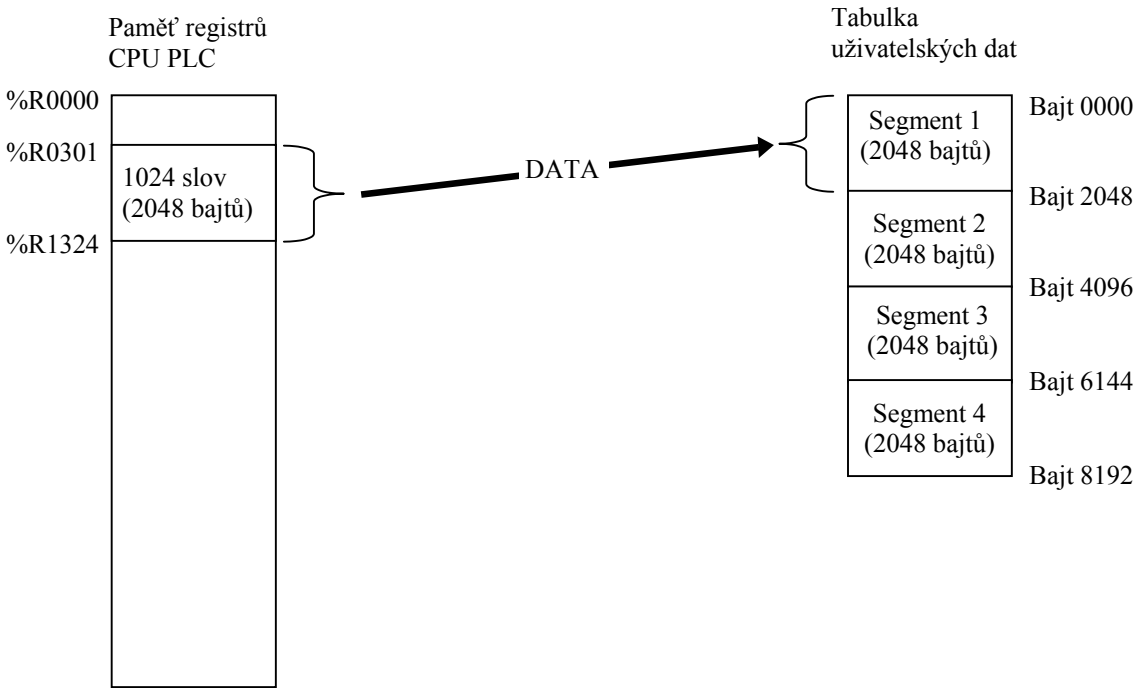
Offset ukazatele počátku dat (slovo 10): Toto slovo obsahuje offset uvnitř typu paměti zvolené ve slově typu datové paměti PLC (slovo 9). ***Poznámka: Offset ukazatele počátku dat PLC je číslo začínající na nule.*** To znamená, že od čísla adresy, kterou chcete zadat, je nutno odečíst jedničku. Chcete-li například zvolit %R0001 jako umístění počátku dat PLC, zapište nulu (1 - 1 =

0). Nebo chcete-li zvolit %R0100, zapište 99 (–100-1 = 99). Všimněte si, že typ paměti %R v tomto příkladu je zadaný v předchozím slově. Počáteční adresa určená tímto slovem bude prvních 1024 souvislých slov paměti PLC používané v COMM REQ.

Příklad tabulky uživatelských dat COMM REQ

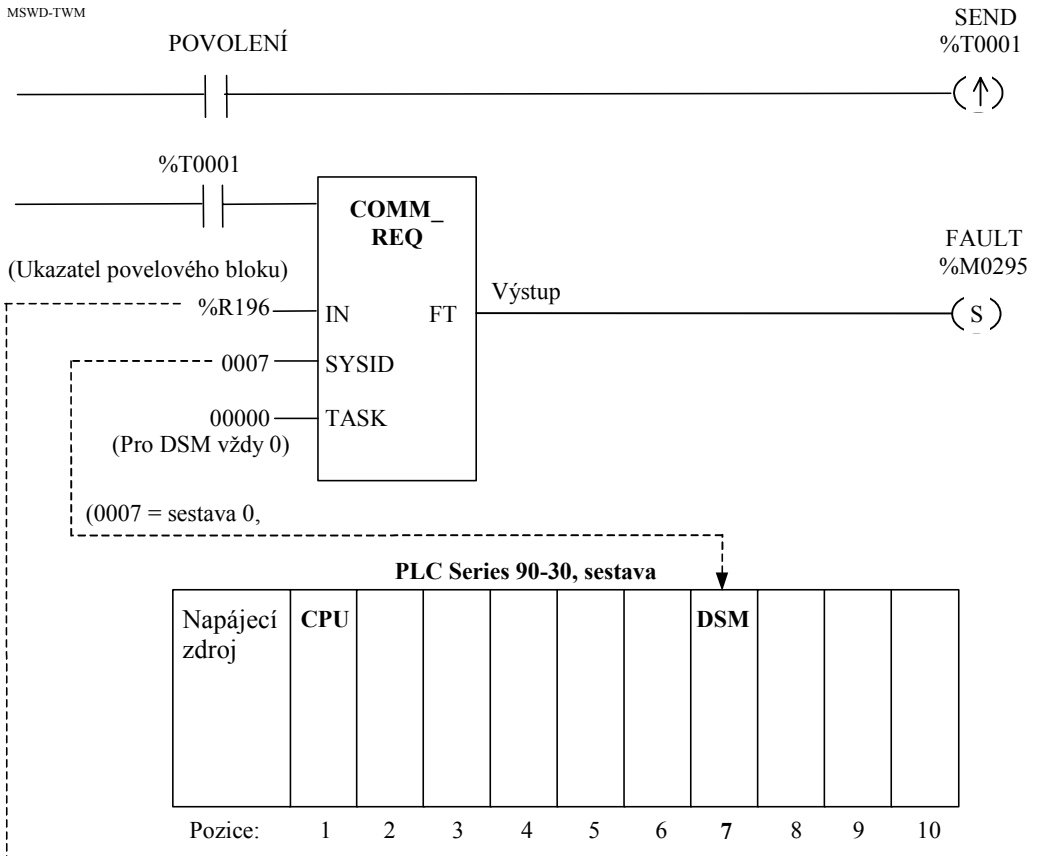
V tomto příkladu jsou dané následující specifikace:

- Modul DSM314 je namontovaný v sestavě 0, pozici 7 PLC.
- Počáteční adresa povelového bloku je %R0196.
- Stavové slovo je umístěno na %R0195.
- Výstup FT (chyba) instrukce COMM REQ řídí SET cívku.
- Segment 1 tabulky uživatelských dat DSM314 je segment, do kterého se zapisuje. To je zadáno v kódu povelu D001 ve slově 7 povelového bloku.
- Data v části paměti registrů PLC s délkou 1024 slov (2048 bajtů) %R0301 až %R1324 se zkopírují a zapíšou do segmentu 1 (2048 bajtů) tabulky uživatelských dat. (Všimněte si, že každé slovo %R má délku dva bajty.) Tento přesun dat je znázorněný na dalším obrázku:



Obrázek B-4. Přesun dat pro kód povelu D001 (zápis do segmentu 1)

Příklad tabulky uživatelských dat COMM REQ



Povelový blok pro DSM COMM_REQ			
Adresa paměti	**Hodnota**	**Popis**	
%R196	4	Délka záhlaví bloku dat ve slovech (vždy 4)	
%R197	0	Příznak ČEKAT/NEČEKAT. Pro DSM vždy 0	
%R198	8	8 = Paměť registrů (%R) pro adresu stavového slova	%R195: Registr stavového slova
%R199	194	Specifikuje registr 195 pro adresu stavového slova	
%R200	0	Pro DSM vždy 0	
%R201	0	Pro DSM vždy 0	
(začátek sekce "záhlaví BlokU dat")			
%R202	D001h	Zápis do segmentu 1 tabulky uživatelských dat	Záhlaví bloku dat (4 slova)
%R203	2048	Velikost segmentu UDT, vždy 2048 bajtů	
%R204	8	8 = Paměť registrů (%R) pro paměť PLC	
%R205	300	Specifikuje registr 301 začátku dat PLC	

Obrázek B-5. Příklad COMM REQ pro UDT DSM314

Část 4: COMM REQ pro načtení parametrů

Povelový blok

Povelový blok obsahuje podrobnosti požadavku na komunikaci. První adresa povelového bloku je zadána ve vstupu IN žebříkové instrukce COMM REQ. Tato adresa může být slovně orientovaná oblast paměti (%R, %AI nebo %AQ). Struktura povelového bloku se může umístit do určené paměti pomocí příslušné programovací instrukce (doporučuje se instrukce BLOCK MOVE). Povelový blok DSM má následující strukturu:

Tabulka B-7. DSM povelový blok COMM REQ pro načtení parametrů

Povelový blok instrukce COMM REQ pro načtení parametrů pro modul DSM			
Popis	**Adresa + Offset**	**Číslo a hodnota slova**	
Délka záhlaví bloku dat	Adresa + 0	Slovo 1, vždy nastaveno na 4	
Příznak ČEKAT/NEČEKAT	Adresa + 1	Slovo 2, vždy nastaveno na 0	
Typ paměti ukazatele stavu (viz tabulka kódů typů paměti níže)	Adresa + 2	Slovo 3, volí uživatel (viz tabulku kódů typů paměti níže)	Specifikace adresy slova
Offset ukazatele stavu	Adresa + 3	Slovo 4, volí uživatel	
Hodnota prodlevy při nečinnosti	Adresa + 4	Slovo 5, vždy nastaveno na 0	
Maximální doba komunikace	Adresa + 5	Slovo 6, vždy nastaveno na 0	
Kód povelu	Adresa + 6	Slovo 7, vždy E501 (hex.)	Záhlaví bloku dat
Velikost bloku dat parametrů v bajtech (musí obsahovat 4 bajty pro slova specifikování parametrů)*	Adresa + 7	Slovo 8 (velikost závisí na hodnotě slova 12)	
Typ paměti dat parametrů (pro slovo 11)	Adresa + 8	Slovo 9, volí uživatel (viz tabulku kódů typů paměti níže)	Specifikace adresy slova 11
Offset dat parametrů (pro slovo 11)	Adresa + 9	Slovo 10, volí uživatel	
Počáteční číslo parametru (0 - 255)	Adresa xyz (adresa zadaná ve slovech 9 a 10)	Slovo 11, volí uživatel	Slova specifikace parametru
Počet načítaných parametrů	Adresa xyz + 1	Slovo 12, volí uživatel	
První data parametrů	Adresa xyz + 2/3	Slovo 13 a slovo 14	Data parametrů
Druhá data parametrů	Adresa xyz + 4/5	Slovo 15 a slovo 16	
...	Adresa xyz + …	...**	
16. data parametru (4 bajty)	Adresa xyz + 32/33	Slovo 43 a slovo 44 **	Parametr bloku dat

* Velikost bloku dat parametru musí být 4 bajty pro slova specifikování parametru plus 4 bajty pro každý parametr.

** Použití těchto slov závisí na hodnotě slova 12.

Délka bloku dat (slovo 1): Délka části záhlaví bloku dat povelového bloku. Pro DSM musí být nastaveno na 4. Záhlaví bloku dat je uloženo ve slovech 7 až 10 povelového bloku.

Příznak ČEKAT/NEČEKAT (slovo 2): Pro DSM se vždy musí nastavit na logickou **nulu**.

Typ paměti ukazatele stavového slova (slovo 3): Toto slovo udává typ paměti, která se použije pro stavové slovo. Každý typ paměti má své specifické číslo kódu uvedené v tabulce kódů typů paměti níže. Pokud například budete chtít pro stavové slovo použít paměť %R, do tohoto slova vložte buď dekadické číslo 8 nebo hexadecimální číslo 08h.

Všimněte si, že pokud zvolíte diskrétní typ paměti (%I nebo %Q), stavovému slovu se přiřadí 16 po sobě jdoucích bitů počínaje adresou uvedenou ve slově offsetu ukazatele stavového slova popsaného níže.

Offset ukazatele stavového slova (slovo 4): Toto slovo obsahuje offset uvnitř zvoleného typu paměti. ***Poznámka: Offset ukazatele stavového slova je číslo začínající na nule.*** To znamená, že od čísla adresy, kterou chcete zadat, je nutno odečíst jedničku. Například chcete-li zvolit %R0001, zapište nulu (1 - 1 = 0). Nebo pokud chcete zadat %R0100, zapište 99 (100 - 1 = 99). Všimněte si, že typ paměti, v tomto příkladu %R, je zadaný předchozím slovem (viz vysvětlení "Typ paměti ukazatele stavového slova" výše).

Hodnota prodlevy při nečinnosti (slovo 5): Protože DSM používá vždy režim NEČEKAT (příznak ČEKAT/NEČEKAT se vždy nastaví na nulu), tento parametr hodnoty prodlevy při činnosti se pro DSM nepoužívá. Nastavte ho na nulu.

Maximální doba komunikace (slovo 6): Protože DSM používá vždy režim NEČEKAT (příznak ČEKAT/NEČEKAT se vždy nastaví na nulu), tento parametr maximální doby komunikace se pro DSM nepoužívá. Nastavte ho na nulu.

Kód povelu (slovo 7): Pro DSM to je vždy E501 (hexadecimálně). Chcete-li tuto hodnotu zapsat přímo jako hexadecimální číslo, použijte instrukcí MOVE slovního typu. Protože tato hodnota je 58 625 dekadicky, instrukce MOVE typu celého čísla (omezená na maximální dekadickou hodnotu 32 767, protože bit 16 se používá pro znaménko) nemá kapacitu na její uložení. Instrukce MOVE typu slova může obsahovat dekadické číslo až 65 535 (FFFF hex.).

Velikost dat parametrů (slovo 8): Udává velikost dat parametrů v bajtech. Tato hodnota závisí na hodnotě slova 12, které udává počet načítaných parametrů. Tato hodnota může být v rozmezí 8 a 68. Rovná se 4 bajtům (pro první dvě slova sekce dat parametrů) plus 4 další bajty pro každý načtený parametr. Pokud například budete chtít načíst 16 parametrů (maximum na COMM REQ), vynásobte 4 krát 16 a dostanete 64. Přičtěte 4 k 64 a dostanete celkem 68 bajtů.

Typ paměti dat parametrů (pro slovo 9): Toto slovo udává typ paměti, která se použije pro data parametrů. Každý typ paměti má specifické číslo kódu uvedené v tabulce kódů typů paměti níže. Pokud například budete chtít zadat paměť %R, do tohoto slova vložte buď dekadické číslo 8 nebo hexadecimální číslo 08h.

Všimněte si, že pokud zvolíte diskrétní typ paměti (%I nebo %Q), bude se vyžadovat skupina 32 po sobě jdoucích bitů na každý parametr a pro slova 11 a 12 se bude vyžadovat skupina 16 po sobě jdoucích bitů.

Offset ukazatele počátku dat parametru (slovo 10): Toto slovo obsahuje offset uvnitř typu paměti zvolené v parametru typu datové paměti. ***Poznámka: Offset ukazatele počátku dat PLC je číslo začínající na nule.*** To znamená, že od čísla adresy, kterou chcete zadat, je nutno odečíst jedničku. Chcete-li například zvolit %R0001 jako umístění počátku dat parametru, zapište nulu (1 - 1 = 0). Nebo chcete-li zvolit %R0100, zapište 99 (– 100-1 = 99). Všimněte si, že typ paměti %R v tomto příkladu je zadaný v předchozím slově.

Počáteční číslo parametru (slovo 11): Udává číslo prvního načítaného parametru. Platné hodnoty jsou 0 – 255. Toto číslo však musí vzít v úvahu hodnotu ve slově 12. Pokud například slovo 12 udává, že se má načíst 10 parametrů, počáteční číslo parametru musí být menší než 247; jinak číslo posledního načítaného parametru bude mimo rozsah (bylo by větší než 255).

Počet parametrů k poslání (slovo 12): Udává, kolik parametrů se má načíst. Platné hodnoty jsou 1 – 16. Hodnota v tomto slově má vliv na hodnotu slova 8.

Data parametrů (slova 13 – 44): Velikost této oblasti dat parametrů závisí na hodnotě ve slově 12 (Počet parametrů k poslání). Na každý parametr se požadují dvě slova (4 bajty) dat. Protože platný počet parametrů s dvojnásobnou délkou celého čísla je 1 až 16, oblast dat parametrů může být mezi 2 a 32 slovy.

Kódy typů paměti COMM REQ: Kódy v následující tabulce se používají ve slově 3 (typ paměti ukazatele stavového slova) a ve slově 9 (typ paměti dat parametrů).

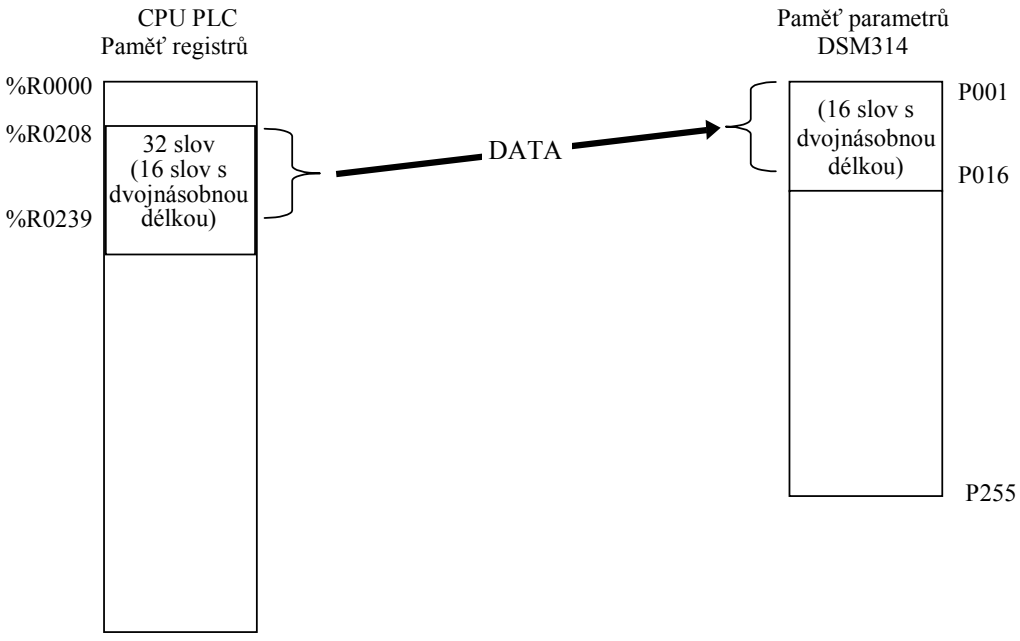
Tabulka B-8. Kódy typů paměti COMM REQ pro načtení parametrů

Kódy typů paměti COMM REQ pro načtení parametrů			
Zkratka typu paměti	**Typ paměti**	**Číslo zapisovaného kódu**	
		Dekadicky	**Hexadecimálně**
%I	Tabulka diskrétních vstupů	70	46h
%Q	Tabulka diskrétních výstupů	72	48h
%R	Paměť registrů	8	08h
%AI	Tabulka analogových vstupů	10	0Ah
%AQ	Tabulka analogových výstupů	12	0Ch

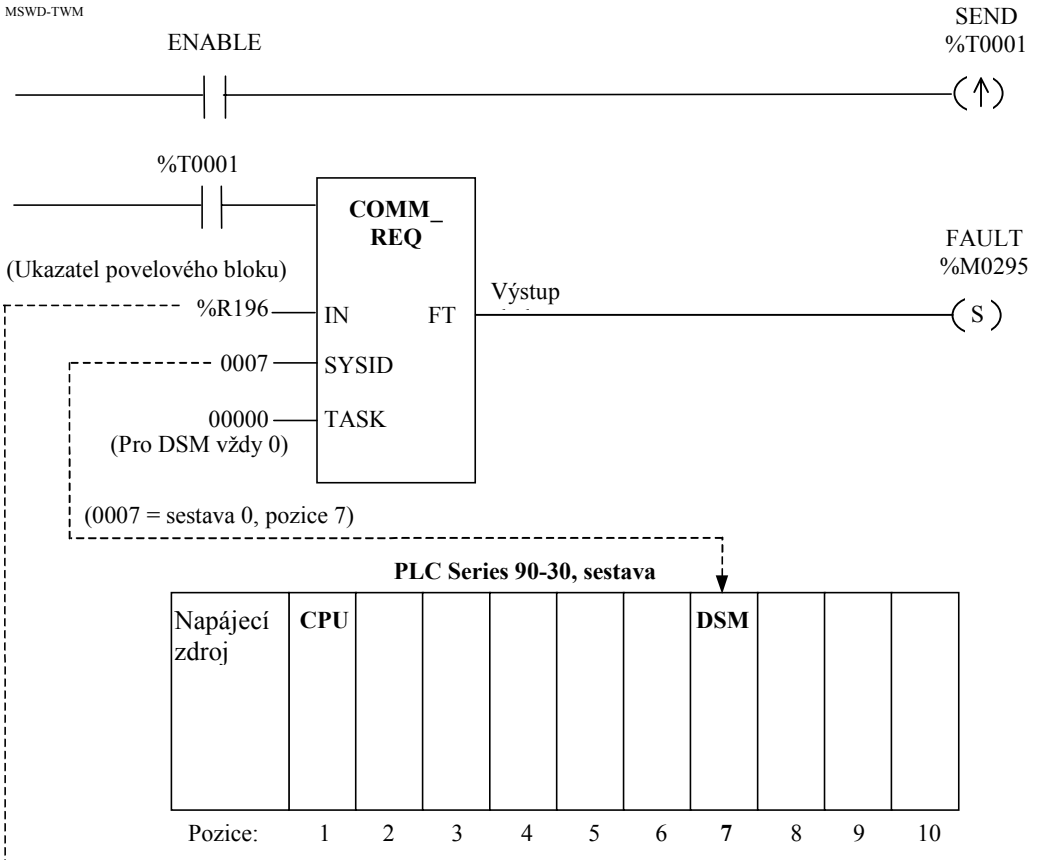
Příklad COMM REQ pro načtení parametru DSM

Tento příklad se používá jako základ pro následující část, "Část 5: Příklad žebříkové logiky COMM REQ". V tomto příkladu jsou dané následující specifikace:

- Modul DSM je namontovaný v PLC v sestavě 0, pozici 7.
- Počáteční adresa povelového bloku je %R0196.
- Stavové slovo je umístěno na %R0195.
- Má se poslat 16 parametrů.
- Výstup FT (chyba) instrukce COMM REQ řídí SET cívku.
- DSM parametr 1 se v tomto příkladu aplikace pokládá za kritický. Poslední dvě příčky "Příkladu žebříkové logiky COMM REQ" (viz část 5) ověří, že parametr 1 pomocí COMM REQ přijal správnou hodnotu.
- Data ve 32 slovech (16 slov s dvojnásobnou délkou) paměti PLC, %R0208 až %R0239, se zkopírují do 16 registrů parametrů s dvojnásobnou délkou slov P001 až P016 v paměti parametrů DSM314. Tento přesun dat je znázorněný na dalším obrázku:



Obrázek B-6. Příklad přesunu dat pro COMM REQ pro načtení parametru



Povelový blok pro DSM COMM_REQ		
Adresa paměti	**Hodno**	**Popis**
%R196	4	Délka datového bloku záhlaví ve slovech (vždy 4)
%R197	0	Příznak ČEKAT/NEČEKAT. Pro DSM vždy 0
%R198	8	8 = Paměť registrů (%R) pro adresu stavového slova
%R199	194	Specifikuje registr 195 pro adresu stavového slova
%R200	0	Pro DSM vždy 0
%R201	0	Pro DSM vždy 0
(začátek sekce "Blok dat" povelového bloku)		
%R202	E501	Pro DSM vždy E501
%R203	68	Velikost dat parametrů, v bajtech
%R204	8	8 = Paměť registrů (%R) pro data parametrů bloku dat
%R205	205	Specifikuje registr 206 začátku Data parametrů
(začátek sekce “Data parametrů” bloku dat)		
%R206	1	Počáteční číslo parametru
%R207	16	Počet parametrů k poslání
%R208 - %R239	xxxxx	Data parametrů pro 16 po sobě jdoucích parametrů s dvojnásobnou délkou

%R195: Registr stavového slova

Záhlaví bloku dat (4 slova)

Data parametrů bloku dat (68 bajtů)

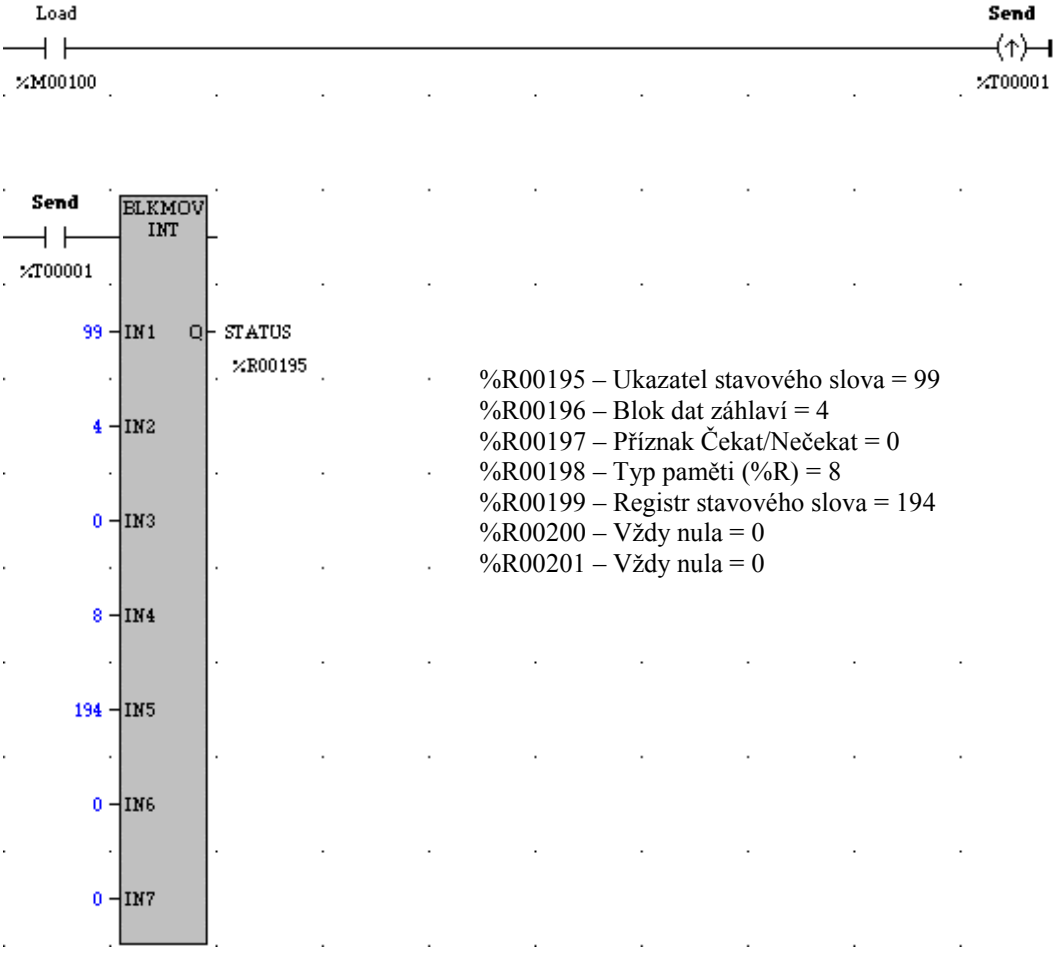
Obrázek B-7. Celkový přehled příkladu COMM REQ pro načtení parametru

Část 5: Příklad žebříkové logiky COMM REQ

Následující příklad žebříkové logiky vychází z příkladu COMM REQ načtení parametru v předchozí části. Seznam povelových bloků najdete v tabulce na předchozí stránce.

Nastavení hodnot povelového bloku COMM REQ

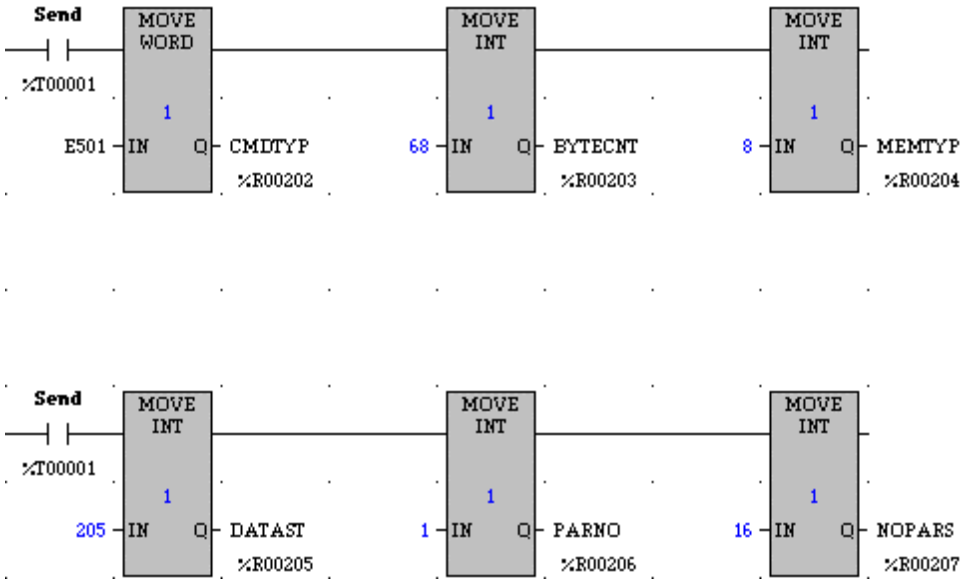
Následující dvě příčky načtou příslušné hodnoty do prvních sedmi slov povelového bloku COMM REQ.



%R00195 – Ukazatel stavového slova = 99
%R00196 – Blok dat záhlaví = 4
%R00197 – Příznak Čekat/Nečekat = 0
%R00198 – Typ paměti (%R) = 8
%R00199 – Registr stavového slova = 194
%R00200 – Vždy nula = 0
%R00201 – Vždy nula = 0

V následujících dvou příčkách se načtou zbývající data povelového bloku. Tato data jsou:

%R00202 – Povel. Pro DSM to je vždy = E501 (hex)
%R00203 – Velikost dat parametru v bajtech = 68
%R00204 – Kód typu paměti pro paměť %R = 8
%R00205 – Počáteční registr pro data parametru (offset o jedničku) = 205
%R00206 – Počáteční číslo parametru = 1
%R00207 – Počet posílaných parametrů = 16
%R00208 – %R00239 – Posílaná data parametru



Logika pro data parametru (nezobrazeno)

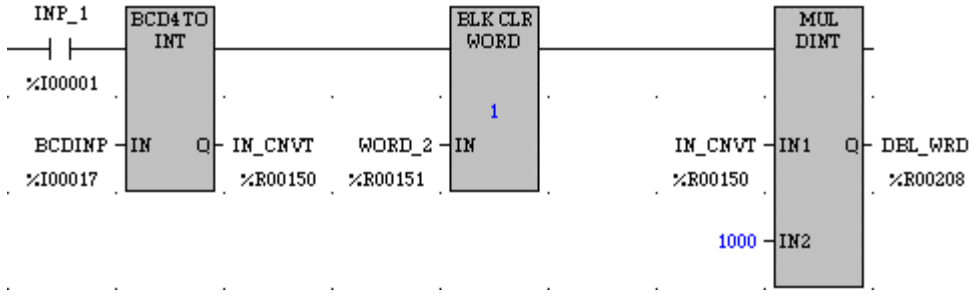
Další logika bude nutná k načtení dat do registrů %R00208 – %R00239, aby bylo možno je poslat do parametrů DSM314. (Hodnota ve dvojnásobném slově %R00208/%R00209 se pošle do parametru 1, hodnota v %R00210/%R00211 se pošle do parametru 2, a tak dál, až konečně hodnota v %R00238/%R00239 se pošle do parametru 16.) Metoda použitá k načtení dat do těchto registrů závisí na vaší aplikaci. Pokud se hodnoty dat nebudou měnit, konstanty je možno přemístit do registrů pomocí instrukcí Přesunutí bloku a/nebo Přesunutí. Pokud se hodnoty mají měnit, je možno je přemístit do registrů ze zařízení rozhraní operátora.

Práce s hodnotami parametrů s dvojnásobnou délkou celého čísla a nastavení měřítka vstupní hodnoty

Data v registru s jednoduchou přesností (16 bitů) je nutno převést na tvar celého čísla s dvojnásobnou délkou (32 bitů), protože parametry DSM mají velikost celého čísla s dvojnásobnou délkou. To se pohodlně provede, když pomocí instrukce násobení celého čísla s dvojnásobnou délkou (MUL DINT) přemístíte vstupní data do registrů PLC, jejichž obsah se pošle do DSM. Tento přístup má dvě možné výhody:

- Je to snadný způsob převodu celého čísla s jednoduchou délkou na dvojnásobnou délku.
- Umožní snadno nastavit měřítko vstupních hodnot, pokud je zapotřebí. Termín nastavení měřítka se vztahuje k násobení a/nebo dělení hodnoty, kterým se vytvoří nová hodnota, která je úměrná původní hodnotě. Například násobení vstupní hodnoty dvěma a pak dělení 3 dá výstupní hodnotu, která bude vždy 2/3 velikosti vstupní hodnoty. Nastavení měřítka je často nutné v servosystému k přizpůsobení aktuální vzdálenosti pohybu se zadanou vzdáleností. Při tom umožňuje funkci "elektronické převodovky". Je možno ho použít k nastavení převodového poměru, stoupání kuličkového šroubu, rozlišitelnosti snímače polohy a preference uživatelské vstupní hodnoty.

V následujícím příkladu se celočíselná hodnota ze vstupního operátorského zařízení (4-místný BCD otočný přepínač) násobí koeficientem 1000, pak se umístí do celočíselného slova s dvojnásobnou délkou %R00208/%R00209 (pro parametr 1).



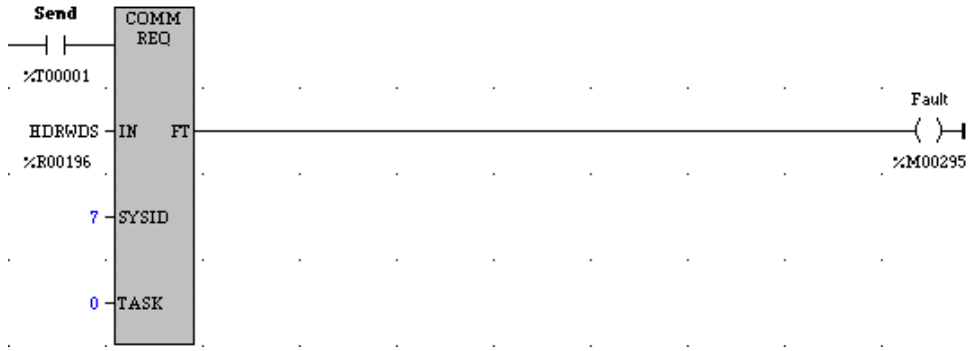
Když v předchozím příkladu bude spínač %I00001 sepnutý, dvojkově kódovaná desítková hodnota (BCD) v BCDINP (%I00017 – %I00032) z BCD operátorského vstupního zařízení se převede na celočíselnou hodnotu (pomocí instrukce BDC4 TO INT) a celočíselná hodnota se umístí do registru %R00150. Dále se %R00151 vynuluje pomocí instrukce BLK CLR. Všimněte si, že na výstupu instrukce BCD4 TO INT bude jednoduchá celočíselná hodnota %R00150. Když se však %R00150 použije jako vstup (IN1) pro instrukci násobení celého čísla s dvojnásobnou délkou (MUL DINT), CPU ho automaticky spojí s další adresou %R (%R00151) a vytvoří celočíselnou hodnotu s dvojnásobnou délkou. Slovo %R00150 se stane nejméně významným slovem a %R00151 se stane nejvýznamnějším slovem tohoto celočíselného slova s dvojnásobnou délkou. Instrukce MUL DINT vynásobí hodnotu v %R00150/%R00151 hodnotou 1000 a výsledek uloží do slova s dvojnásobnou délkou %R00208/%R00209 (DBL_WRD).

Při použití tímto způsobem se %R00151 nazývá “implikovaná adresa”, protože se nezobrazuje na obrazovce. Nezapomeňte, že %R00151 nesmíte použít pro žádný jiný účel (musí zůstat na nule); jinak hodnota uložená do %R00150 z instrukce BDC4 TO INT by se změnila. Stejný princip platí v případě slova s dvojnásobnou délkou %R00208/R00209. Zde se použití %R00209 předpokládá proto, že %R00208 se zobrazí jako výstup instrukce násobení celého čísla s dvojnásobnou délkou (MUL DINT). Proto %R00209 je nutno vyhradit pouze pro toto použití.

Na této příčce instrukce MUL DINT vykonává dvě funkce: (1) převádí hodnotu v %R00150 z tvaru jednoduchého celého čísla na celé číslo s dvojnásobnou délkou a (2) nastavuje měřítko v %R00150/%R00151 vynásobením hodnotou 1000. Pokud nastavení měřítka bude nežádoucí, místo hodnoty 1000 v IN2 instrukce MUL DINT je možno použít hodnotu 1; tím se provede převod na celé číslo s dvojnásobnou délkou bez změny (nastavení měřítka) hodnoty.

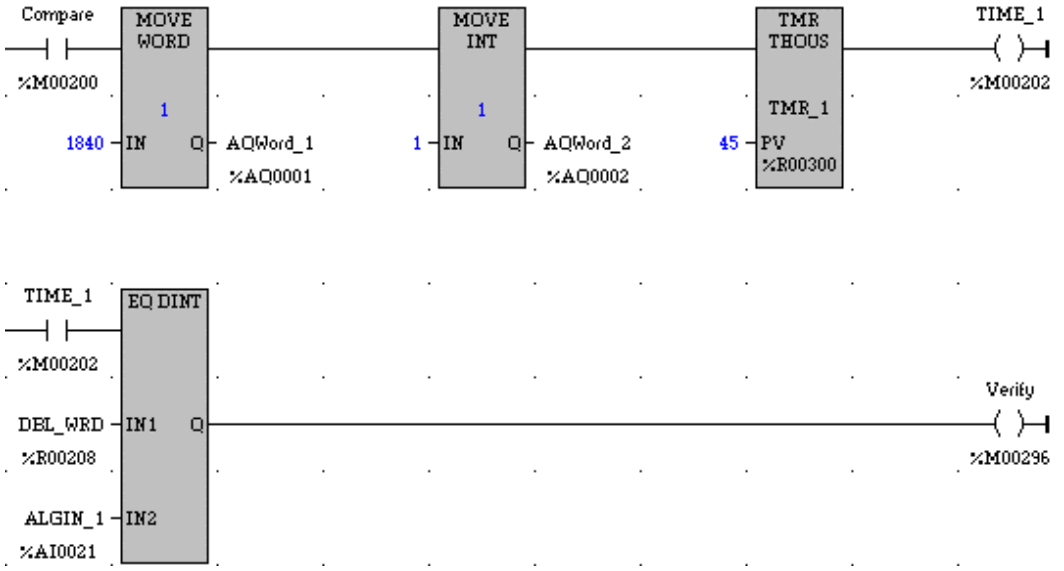
Instrukce požadavku na komunikaci

Následující obrázek ukazuje instrukci požadavku na komunikaci (COMM REQ). Vstup IN obsahuje adresu prvního slova povelového bloku. Vstup SYSID obsahuje číslo sestavy a pozice (sestava 00, pozice 07) DSM314 adresovaného touto instrukcí COMM REQ. Vstup TASK je pro DSM314 vždy nula. Výstup FT připojuje cívku (%M00295), která se nabudí, když se zjistí chyba.



Ověření dat poslaných do parametru 1

V tomto příkladu je hodnota DSM parametru 1 kritická, protože specifikuje vzdálenost pohybu, která, pokud bude nesprávná, může vést na poškození stroje. Proto logika na dvou následujících příčkách ověřuje, že do parametru 1 byla uložena správná hodnota. Pokud hodnota nebude správná, kontakty (nezobrazené) z výstupní cívky "VERIFY" na druhé příčce zabrání, aby DSM vygeneroval pohyb.



První příčka: Instrukce MOVE WORD přesune hexadecimální číslo 1840 do %AQ00001, prvního slova okamžitého povelu. Hodnota dolního bajtu (40) tohoto čísla zadává *Volbu dat návratu* okamžitého povelu. Hodnota horního bajtu (18) zadává volbu režimu pro *Data parametrů.*

Instrukce MOVE INT přemístí dekadickou hodnotu 1 udávající parametr 1 do %AQ00002. Tím se zadá, že hodnota v DSM parametru 1 se zapíše do dvojnásobného slova *Uživatelem volitelná data* pro osu 1, v tomto příkladu %AI00021/AI00022.

> **Poznámka:** Skutečné adresy %AI používané v jednotlivých modulech DSM se určí, když se provádí konfigurace modulu.

Instrukce časovače TMR THOUS (tisíciny) vytvoří prodlevu 45 milisekund po poslání okamžitého povelu *Volba dat návratu*. To je nutné proto, že *Volba dat návratu* není v žebříku k dispozici, dokud neuplynou alespoň 3 cykly nebo 20 milisekund (podle toho, co je delší) po poslání okamžitého povelu *Volba dat návratu*. Protože doba cyklu v tomto příkladu je 14 milisekund, tato prodleva 45-milisekund zajistí, že data parametru 1 budou ve slově s dvojnásobnou délkou *Uživatelem volitelná data* před tím, než se na další příčce vykoná instrukce Equal. Všimněte si, že kontakt %M00200 musí zůstat ve stavu ON dostatečně dlouho, aby časovač TMR vykonal časovou prodlevu a povolil druhou příčku.

Druhá příčka: Po uplynutí prodlevy 45 milisekund v předchozí příčce se kontakt %M00202 sepne a povolí tuto příčku. V této příčce instrukce celého čísla s dvojnásobnou délkou EQUAL porovná hodnotu v %R00208/R00209 (zdroj hodnoty poslané instrukcí COMM REQ do DSM parametru 1) s hodnotou vrácenou z parametru 1 v %AI00021/AI00022. Pokud hodnoty budou stejné, cívka “Verify” se sepne.

Dodatek C

Zařízení polohové zpětné vazby

S DSM314 fungují čtyři modely digitálního sériového snímače polohy α a β GE Fanuc:

Tabulka C-1. Rozlišitelnost digitálního sériového snímače polohy

8K	(8 192 pulsů/ot.)	- U nových motorů se už nepoužívá
32K	(32 768 pulsů/ot.)	- Standardně u motorů β Series
64K	(65 536 pulsů/ot.)	- Standardně u motorů α Series
1000K	(1 048 576 pulsů/ot.)	- Volitelně u motorů α Series

Poznámka: Starší sériové snímače polohy s milionem pulsů "A" nebo "C" Series nebudou s DSM314 fungovat. Pokud tyto snímače polohy připojíte, bude se hlásit chyba.

Pro účely řízení polohy DSM314 všechny snímače polohy implicitně bere jako snímače polohy s 8192 pulsy/ot. K zajištění hladké činnosti při nízkých rychlostech se v digitálních kontrolérech rychlosti serva stále používají snímače polohy 32K, 64K a 1000K. Chcete-li použít vyšší rozlišení polohové zpětné vazby, přečtěte si odstavec Parametry ladění v kapitole Konfigurace.

Režimy digitálního sériového snímače polohy

Digitální sériové snímače polohy GE Fanuc je možno provozovat buď v **Inkrementálním** režimu nebo v **Absolutním** režimu. Režim se nakonfiguruje zvolením ***Režimu zpětné vazby*** v konfiguračním softwaru. Řádná činnost **absolutního** režimu vyžaduje externí baterii, která musí být připojená k servozesilovači. Informace o volbě a instalaci baterie najdete v příslušném manuálu k zesilovači.

Poznámky k inkrementálnímu režimu snímače polohy

Digitální sériový snímač polohy je možno použít jako inkrementální snímač polohy, který dává 8192 pulsů na otáčku hřídele, kde při vypnutí a zapnutí napájení se neudržuje počet pulsů na otáčku. Ekvivalent *značkovacího* pulsu se objeví jednou na každou otáčku hřídele motoru. Všechny ***Režimy výchozí polohy*** (Spínač výchozí polohy, Přesunutí +, Přesunutí –) a %AQ povely *Nastavení polohy* se vztahují k ose a po úspěšném dokončení nastaví %I bit *Poloha platná* do jedničky. Nakonfigurovaná ***Horní mez polohy*** a ***Dolní mez polohy*** je platná a %AI stavové slovo *Okamžitá poloha* předávané z DSM314 se přetočí z hodnoty horní meze na dolní mez nebo z dolní meze na horní mez. To je vynikající režim pro souvislé aplikace, které umožňují činnost přes inkrementální pohyby ve stejném směru. Konfigurační údaje ***Offset výchozí polohy*** a ***Výchozí poloha*** umožňují jednoduché adresování požadovaného místa.

Poznámky k režimu absolutního snímače polohy

Digitální sériový snímač polohy GE Fanuc je možno použít jako absolutní snímač polohy přidáním baterie, která bude udržovat polohu serva během vypnutí napájení. Cyklus Nalezení výchozí polohy nebo %AQ povel *Nastavení polohy* se musí provést předem nebo vždy, když dojde ke ztrátě napětí baterie snímače polohy a servozesilovač má také vypnuté napájení. Aby snímač polohy fungoval správně s baterií, v konfiguračním softwaru je nutno ***Režim zpětné vazby*** nastavit na ABSOLUTNÍ.

Absolutní snímač polohy – první použití nebo použití po ztrátě napětí baterie snímače polohy

Absolutní snímač polohy přechodně dává inkrementální data během prvního použití nebo po obnovení napětí baterie snímače polohy. Ke ztrátě inkrementálních dat dojde, když otočení hřídele motoru způsobí, že snímač polohy projde referenčním bodem (podobný značkovacímu signálu) během jedné otáčky hřídele motoru. Po novém připojení baterie absolutního režimu k zesilovači se digitální absolutní snímač polohy musí otočit až o celou jednu otáčku. Snímač polohy během jedné otáčky nastaví svou referenci a nahlásí do DSM314 stav nastavení reference.

Režim absolutního snímače polohy - inicializace polohy

Když se systém poprvé zapne v režimu absolutního snímače polohy, u snímače polohy je nutno nastavit posunutí polohy. To lze provést pomocí cyklu %Q *Nalezení výchozí polohy* nebo povelu %AQ *Nastavení polohy*.

Cyklus Nalezení výchozí polohy – Režim absolutního snímače polohy

Režim nalezení výchozí polohy je možno nakonfigurovat na operaci Pohyb (+), Pohyb (–) nebo Spínač výchozí polohy. Další podrobnosti k operaci cyklu výchozí polohy najdete v kapitole 4. Konfigurační údaje ***Offset výchozí polohy*** a ***Výchozí poloha*** fungují stejně jako v inkrementálním režimu snímače polohy. Po dokončení cyklu výchozí polohy se %AI stavové slovo *Okamžitá poloha* nastaví na nakonfigurovanou hodnotu ***Výchozí poloha***. DSM314 interně vypočítá absolutní offset zpětné vazby snímače polohy k vytvoření nakonfigurované ***Výchozí polohy*** při dokončení cyklu výchozí polohy. Tento offset zpětné vazby se okamžitě uloží do energeticky nezávislé paměti DSM314 (zálohované kondenzátorem).

Jakmile bude absolutní poloha zavedena úspěšným dokončením cyklu Nalezení výchozí polohy, DSM314 automaticky provede inicializaci %AI stavového slova *Okamžitá poloha* po zapnutí napájení a %I bit *Poloha platná* nastaví do jedničky.

Poznámka: Pokud %I bit Poloha platná bude nastavený v jedničce před vyvoláním cyklu výchozí polohy, tento cyklus vynuluje bit Poloha platná a pak ho nastaví znovu po dokončení cyklu. Pokud se cyklus výchozí polohy přeruší bitovým povelem %Q bit Zrušit všechny pohyby, bit Poloha platná zůstane v nule. Vypnutí a zapnutí napájení však bude mít za následek, že se obnoví okamžitá poloha a Poloha platná se nastaví do jedničky.

Povel Nastavení polohy – Režim absolutního snímače polohy

Povel %AQ *Nastavení polohy* funguje stejným způsobem jako v inkrementálním režimu snímače polohy. Po dokončení operace *Nastavení polohy* se *Okamžitá poloha* nastaví na hodnotu *Nastavení polohy*. DSM314 interně vypočítá absolutní offset zpětné vazby snímače polohy potřebný k vytvoření zadané hodnoty *Nastavení polohy*. Tento offset zpětné vazby se okamžitě uloží do energeticky nezávislé paměti DSM314 (zálohované kondenzátorem).

Pokud AQ povel Nastavení polohy přijde před tím, než se nastaví reference snímače polohy, bude se hlásit Chybový kód 53 (hex) "Snaha inicializovat polohu před průchodem digitálního snímače polohy referenčním bodem". Tato chyba se hlásí, pouze pokud režim zpětné vazby bude nastavený na **Absolutní**. Sériové snímače polohy nakonfigurované na **Inkrementální** režim toto omezení nemají.

Jakmile bude absolutní poloha zavedena povelem *Nastavení polohy*, DSM314 automaticky provede inicializaci *Okamžité polohy* po zapnutí napájení a %I bit *Poloha platná* nastaví do jedničky.

Režim absolutního snímače polohy – zapnutí napájení DSM314

Baterie připojená k podsystému serva udržuje napájení logiky čítače snímače polohy. Jakmile bude nastavená reference snímače polohy po prvním zapnutí napájení, snímač polohy bude automaticky udržovat okamžitou polohu, i když během ztráty napájení serva dojde k pohybu osy. Snímač polohy sleduje stav baterie a hlásí ztrátu napětí baterie nebo nízké napětí baterie do DSM314.

DSM314 dokončí diagnostiku po zapnutí napájení a když bude nakonfigurovaný na absolutní režim snímače polohy, dotáže se na referenční stav digitálního sériového snímače polohy. Platný referenční stav ze snímače polohy signalizuje do DSM314, že má načíst absolutní polohu snímače. DSM314 nahlásí %AI stav *Okamžitá poloha* jako součet polohy snímače a absolutního offsetu zpětné vazby nastavené výchozím cyklem Nalezení výchozí polohy nebo %AQ povelem *Nastavení polohy*.

Inkrementální snímač polohy s fázovým posuvem

Inkrementální snímače polohy s fázovým posuvem dávají tři výstupní signály do DSM314: Kanál A, Kanál B a Značka. Signály Kanál A a Kanál B se mění při otáčení snímače polohy, což umožňuje, aby DSM314 čítal počet přechodů signálu a vypočítal nejnovější změnu polohy snímače a směr otáčení.

Inkrementální snímač polohy s fázovým posuvem jsou inkrementální zpětnovazební zařízení; nedávají souvislou indikaci absolutního úhlu hřídele, když se vstupní hřídel otáčí. Z toho důvodu %AI stavové slovo DSM314 *Okamžitá poloha* se musí inicializovat známou fyzickou polohou před povolením řízení polohy. Toto sladění polohy je možno provést pomocí %AQ okamžitého povelu *Nastavení polohy* nebo pomocí cyklu %Q *Nalezení výchozí polohy*. Cyklus výchozí polohy používá značkovací kanál snímače polohy, což je jeden puls na otáčku vytvořený ve známém úhlu hřídele snímače polohy. Úspěšné dokončení cyklu %Q *Nalezení výchozí polohy* nebo %AQ povelu *Nastavení polohy* bude mít za následek, že DSM314 nastaví %I bit osy *Poloha platná* do jedničky. *Poloha platná* musí být nastavená před tím, než se pohybové programy smí vykonat. *Poloha platná* se vynuluje pouze chybou fázového posuvu snímače polohy (Kanál A a Kanál B se přepnou ve stejném okamžiku) nebo současným nastavením %Q bitů *Nalezení výchozí polohy* a *Zrušení* do jedničky.

Poznámka: V digitálním režimu se pro master osu režimu vlečené osy podporují pouze inkrementální snímače polohy s fázovým posuvem.

Dodatek D

Spuštění a ladění digitálních a analogových servosystémů GE Fanuc

Tento dodatek uvádí postup pro spuštění a ladění digitálních nebo analogových servosystémů GE Fanuc. U digitálních servosystémů jsou v DSM314 dvě řídící smyčky, které je nutno naladit, rychlostní smyčka a polohová smyčka. Vždy začněte konfigurací modulu, pak pokračujte nastavením rychlostní smyčky a nakonec nastavte polohovou smyčku. U analogových servosystémů existuje několik postupů spouštění.

Informace ke spuštění a ladění digitálních servosystémů

Jsou zde popsané tři hlavní části:

- Potvrzení spínače výchozí polohy, vstupů přejetí a směru motoru.
- Ladění rychlostní smyčky.
- Ladění polohové smyčky.

Potvrzení spínače výchozí polohy, vstupů přejetí a směru motoru

1. Připojte motor, zesilovač a modul DSM314 podle postupu v kapitole 2.
2. Pokud se budou používat spínače Meze přejetí (v konfiguraci ***Koncový spínač přejetí*** = povolený), zapojte je na správné body 24 V svorkovnice (viz kapitola 3). Vstupy přejetí se používají bezpečným způsobem, tj. je nutno použít normálně rozpínací zařízení nebo zařízení typu PNP. Do vstupu je nutno přivádět proud, aby se na vstupu udržovala úroveň logické 1, když osa NENÍ v poloze přejetí, jinak by se hlásil chybový stav (Chyba A9). V opačném případě je ***Koncový spínač přejetí*** nutno pomocí konfiguračního softwaru nastavit na Zakázaný.
3. Pokud se bude používat Spínač výchozí polohy (v konfiguraci Režim výchozí polohy = Spínač výchozí polohy), zapojte ho na správné body 24 V svorkovnice (viz kapitola 3). Spínač výchozí polohy musí být zapojený a ovládaný tak, aby byl VŽDY V JEDNIČCE (sepnutý), když se osa bude nacházet na záporné straně výchozí polohy, a VŽDY V NULE (rozepnutý), když se osa bude nacházet na kladné straně výchozí polohy. Spínač výchozí polohy se normálně montuje na nebo v blízkosti jednoho konce posuvu osy. Před vykonáním cyklu výchozí polohy je důležité ověřit činnost spínače výchozí polohy. Může být nutné obrátit směr motoru (v konfiguraci modulu Motor1 nebo Motor2 Dir = POS/NEG).

4. K nastavení požadovaných koeficientů měřítka a ostatních konfigurovatelných parametrů použijte konfigurační software. Ve výchozím nastavení konfigurace je NUTNO změnit následující položky:

Položka konfigurace	**Nastavení**
Režim osy 1	Digitální servo
Typ motoru:	Zvolte z tabulky v kapitole 4
Časová konstanta polohové smyčky:	60 ms
Zisk rychlostní smyčky:	(Setrvačnost břemena / Setrvačnost motoru) * 16
Uživatelské jednotky: pulsy (Pouze standardní režim)	Viz kapitola 3
Mez polohové odchylky:	30000 × Uživatelské jednotky / pulsy

Konfigurační parametry nastavte ve výše uvedeném pořadí.

5. Uložte konfiguraci do PLC.
6. Smažte program z PLC, vynulujte všechny %Q bity DSM314 a PLC přepněte do režimu RUN. Sledujte %I bity CTL, jestli reagují na Spínač výchozí polohy, (+) přejetí a (-) přejetí, a přesvědčte se, že každý bit odpovídá správnému spínači (definice %I bitů najdete v kapitole 5).
7. Nastavte %Q bit *Povolit pohon* do jedničky a ověřte, že servozesilovač je povolený. Pokud se na servomotoru používá brzda, měla by být uvolněná.
8. Pošlete kód povelu %AQ pro *Vynucenou rychlost digitálního serva* 100 (ot./min). Přesvědčte se, že se motor pohybuje v požadovaném KLADNÉM směru a *Okamžitá rychlost* se hlásí v tabulce %AI jako KLADNÁ. Pokud se motor bude pohybovat v nesprávném směru, pomocí parametru ***Směr osy*** v konfiguračním softwaru zaměňte kladný a záporný směr osy.
9. Z tabulky %AQ odstraňte povel *Vynucené rychlosti digitálního serva*. Během konfigurace použijte nízkou ***Rychlost jogu*** a ***Zrychlení jogu***, hodnoty je možno zvýšit později. Nastavte %Q bit *Jog Plus* do jedničky. Přesvědčte se, že se servo pohybuje ve správném směru a že *Okamžitá rychlost*, kterou hlásí DSM314 v tabulce %AI, souhlasí s nakonfigurovanou ***Rychlostí jogu***. Pokud pohybové programy používají zrychlení vyšší než ***Zrychlení jogu***, může být nutné zvýšit ***Zrychlení jogu*** tak, aby operace *Zrušit všechny pohyby* a Normální zastavení pracovaly očekávaným způsobem.
10. Při konfiguraci modulu použijte nízkou hodnotu pro ***Rychlost pro nalezení výchozí polohy*** a ***Konečnou rychlost výchozí polohy***, hodnoty je možno zvýšit později. Zkontrolujte správnou činnost cyklu *Nalezení výchozí polohy* krátkodobým nastavením %Q bitu *Nalezení výchozí polohy* do jedničky (%Q bit *Pohon povolen* musí být také v jedničce). Osa by se měla pohybovat směrem ke spínači výchozí polohy nakonfigurovanou ***Rychlostí pro nalezení výchozí polohy***, pak vyhledat referenční bod snímače polohy nakonfigurovanou ***Konečnou rychlostí výchozí polohy***. Pokud bude nutné, upravte rychlosti a polohu spínače výchozí polohy pro správnou činnost. Poslední spínač výchozí polohy se MUSÍ přepnout alespoň 10 milisekund před tím, než snímač polohy najde referenční bod. Fyzické umístění ***Výchozí polohy*** je možno upravit změnou hodnoty ***Offsetu výchozí polohy*** pomocí konfiguračního softwaru.
11. Sledujte výkon serva a použijte %Q bity *Jog Plus* a *Jog Minus* k vykonání pohybu servomotoru v obou směrech. Vložením správného kódu povelu do tabulky % AQ je možno přechodně pozměnit *Časovou konstantu polohové smyčky*. U většiny systémů je možno ***Časovou konstantu polohové smyčky*** snižovat, dokud se nezjistí určitá nestabilita serva, pak

se zvýší na hodnotu přibližně o 50% vyšší. Jakmile bude určena správná časová konstanta, konfiguraci DSM314 je nutno aktualizovat pomocí konfiguračního softwaru. ***Posuv rychlostí*** je možno také nastavit na nenulovou hodnotu (typicky 90 – 100 %) pro optimální charakteristiku serva. Viz *Ladění digitálního serva GE Fanuc*, kde najdete informace o nastavení digitálního serva ***Zisku rychlostní smyčky***.

12. Pokud se bude používat režim vlečené osy s inkrementálnín snímačem polohy s fázovým posuvem, přesvědčte se, že *Okamžitá poloha* (Pomocná osa 3) představuje polohu snímače polohy. **Přesvědčte se, že pomocí konfiguračního softwaru je naprogramovaný požadovaný poměr slave : master vlečené osy jako poměr A:B**.

Rady pro vyhledávání chyb při spouštění digitálního servosystému

1. DSM314 vyžaduje firmware PLC verze 10.0 nebo vyšší a některý z následujících konfiguračních/programovacích softwarových balíků:
   - CIMPLICITY Machine Edition Logic Developer – PLC verze 2.1 nebo pozdější
   - VersaPro verze 1.1 nebo pozdější
2. DSM podpora pro motory Beta M1 a Beta M0.5 vyžaduje DSM firmware verze 3.0 nebo pozdější.
3. Výchozí konfigurace DSM314 pro vstupy ***koncového spínače přejetí*** je nastavena na POVOLENO. Proto na vstupy přejetí musí být přivedeno 24 V ss, jinak DSM314 nebude pracovat. Pokud se vstupy přejetí nebudou používat, konfiguraci vstupů DSM314 ***koncového spínače přejetí*** je nutno nastavit na ZAKÁZÁNO.

   Pokud %I bit *Osa povolena* bude nastavený na nulu, osa nebude odpovídat na žádné %Q bity nebo %AQ povely. Když se servomotor nebude používat se servoosou, ***Typ motoru*** musí být nastavený na 0 nebo *Osa povolena* musí zůstat v nule. ***Typ motoru*** ve stavu 0 zakáže zpracování servosmyčky osy a nastaví *Osa povolena do jedničky*, což umožní, aby osa přijímala povely, jako například *Okamžité načtení parametru* a *Nastavení režimu analogového výstupu*.
4. **%Q řídicí bit *Povolit pohon* musí být nastavený trvale do jedničky, jinak nebude povolený žádný jiný pohyb než Jog.** Pokud se nevyskytnou žádné chyby STOP, %I stavový bit *Pohon povolen* bude zrcadlit stav %Q bitu *Povolit pohon*. Chyba STOP nastaví *Pohon povolen* do nuly, i když *Povolit pohon* bude stále v jedničce. Aby se pohon znovu povolil, chybový stav je nutno opravit a %Q řídicí bit *Vynulování chyby* nastavit do jedničky na dobu jednoho cyklu PLC.
5. Pokud %I stavový bit *Výskyt chyby modulu* bude v jedničce a %I stavové bity *Osa povolena* a *Pohon povolen* budou v nule, pak se vyskytla chyba STOP (stavová LED bude blikat rychle). **V tomto stavu servoosa nebude odpovídat na žádné jiné %Q bity nebo %AQ povely než na %Q bit *Vynulování chyby*.**
6. %Q řídicí bit *Vynulování chyby* používá jednorázovou akci. Aby se chyba vynulovala, při každém vygenerování chyby se bit musí nastavit do nuly a pak do jedničky alespoň na jeden cyklus PLC.
7. LED dioda CFG OK musí svítit, jinak DSM314 nebude odpovídat na povely PLC. Pokud LED nebude svítit, pak z PLC nepřišla platná konfigurace DSM314 nebo se zjistila chyba konfigurace. Zkontrolujte %AI slova chybového kódu, jestli neobsahují chyby Dxxx, které jsou popsané v části “Kódy systémových chyb” v Dodatku A. Také zkontrolujte tabulku chyb PLC, jestli neobsahuje nahlášené chyby konfigurace.

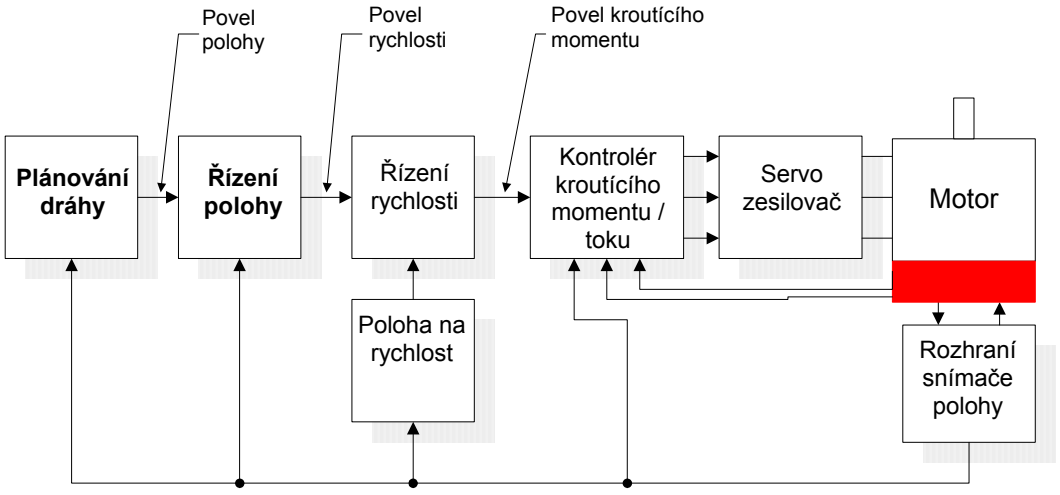
Ladění digitálního servopohonu GE Fanuc

Následující stránky uvádějí úvod do základů požadovaných pro ladění digitálního servopohonu GE Fanuc. Tento úvod ukazuje jednu metodu ladění servopohonu. Tato metoda nefunguje se všemi aplikacemi a přístup je nutno jí upravit podle aplikace. Aby se zobrazovaly a měřily potřebné průběhy signálů, analogové výstupy DSM314 musí být připojené k osciloskopu. Bez osciloskopu pro měření signálů ladění servopohonu s následujícím přístupem nelze provést. Povel %AQ *Režim volby analogového výstupu* (47h) se používá ke zvolení dat, která se mají poslat na analogové výstupy během ladění serva. Popis tohoto povelu %AQ je uvedený v kapitole 5.

Požadavky na ladění

Modul má tři hlavní parametry, které se nastavují během ladění. Tyto parametry jsou ***Časová konstanta polohové smyčky***, ***Zisk rychlostního posuvu***, a ***Zisk rychlostní smyčky***. ***Časová konstanta polohové smyčky integrátoru*** dává polohové smyčce další stupeň volnosti, ale v typických aplikacích se nevyžaduje.

Řídicí smyčky se ladí tak, že se nejdříve provede ladění vnitřních řídicích smyček. V tomto příkladu vnitřní řídicí smyčka, která vyžaduje ladění, je rychlostní smyčka. Jak je znázorněno na následujícím obrázku, polohová smyčka je vnější smyčka a posílá rychlostní povely do rychlostní smyčky.



Obrázek D-1. Blokové schéma řídicích smyček

Ladění rychlostní smyčky *DIGITÁLNÍ REŽIM*

Správný způsob ladění rychlostní smyčky je oddělit rychlostní smyčku od polohové smyčky. Aby se dosáhlo tohoto oddělení, metoda se musí použít na přímo posílané rychlostní povely, aniž by se použilo řízení polohové smyčky. Modul DSM má několik režimů, které umožňují poslat rychlostní povel přímo do rychlostní smyčky. Tyto dva způsoby jsou:

Způsob č.1:

Okamžitý povel %AQ *Vynucená rychlost digitálního serva* (34h) pošle rychlostní povel přímo do rychlostní smyčky. Tento povel se liší od povelu *Pohyb rychlostí,* který k vygenerování povelu používá polohovou smyčku. To je důležité, protože polohová smyčka se nesmí v tomto bodě procesu ladění vzájemně ovlivňovat s rychlostní smyčkou. Povel %AQ *Vynucená rychlost digitálního serva* umožňuje vygenerovat krokovou změnu rychlosti. Krok rychlostního povelu se pak použije k vygenerování krokové odezvy rychlostní smyčky. Všimněte si, že když se provede kroková změna rychlostního povelu, zrychlení bude omezené pouze na šířku pásma rychlostní smyčky. U některých aplikací to může způsobit poškození řízeného zařízení v důsledku vysokého zrychlení.

Způsob č.2:

U některých aplikací metoda č.1 řízenému zařízení dává příliš velký náraz. V takových případech je pro vygenerování rychlostního povelu zapotřebí jiná metoda. Tato metoda vyžaduje nastavit polohovou smyčku na konfiguraci otevřené smyčky. Polohová smyčka se nastaví na otevřenou smyčku nastavením ***časové konstanty polohové smyčky*** na nulu a ***Zisku posuvu rychlostí*** na 100 procent. K vygenerování rychlostních povelů do servopohonu pak je možno použít *Povel Pohyb rychlostí* nebo pohybový program.

První parametr, který je nutno nastavit, je ***Zisk rychlostní smyčky***. Tento parametr nastaví šířku pásma rychlostní smyčky. Jako výchozí bod použijte následující vztah (také se podívejte do kapitoly Zisk rychlostní smyčky):

Rovnice 1

$$\text{Zisk rychlostní smyčky} = \frac{J_l}{J_m} \cdot 16$$

Kde :

J_l = Setrvačnost břemena

J_m = Setrvačnost motoru

Zisk rychlostní smyčky vypočítaný v rovnici 1 není nutno měnit. Avšak vzhledem k aplikaci (například rezonance stroje) může být nutné hodnotu upravit. Pro ladění ***Zisku rychlostní smyčky*** je možno použít následující postup:

1. Zvolte způsob zavedení rychlostního povelu do rychlostní smyčky. Příkladem provedení tohoto úkolu může být metoda č.1 a metoda č.2 (výše).

2. Připojte osciloskop k analogovým výstupům pro zpětnou vazbu rychlosti motoru a povelu kroutícího momentu. Instrukce pro nakonfigurování analogového výstupu najdete v části 4.25 kapitoly 5.

3. Zisk rychlostní smyčky nastavte na nulu. To je konzervativní přístup. Pokud budete vědět, že aplikace nemá rezonanční frekvence v rozmezí nula až přibližně 250 Hz, můžete začít s vyšší hodnotou, ale nepřekročte hodnotu vypočítanou v rovnici 1.

4. Vygenerujte krokovou změnu rychlostního povelu. V tomto bodě kroková změna musí být relativně malá v porovnání s plnou rychlostí stroje. Deset až 20 % jmenovité rychlosti stroje je rozumným začátkem.

5. Na osciloskopu sledujte Rychlost motoru a Povel kroutícího momentu. Cílem je dosáhnout kritické odezvy tlumené rychlostní smyčky. Zejména dávejte pozor na oscilace, které se vyskytují v signálu rychlostní zpětné vazby.

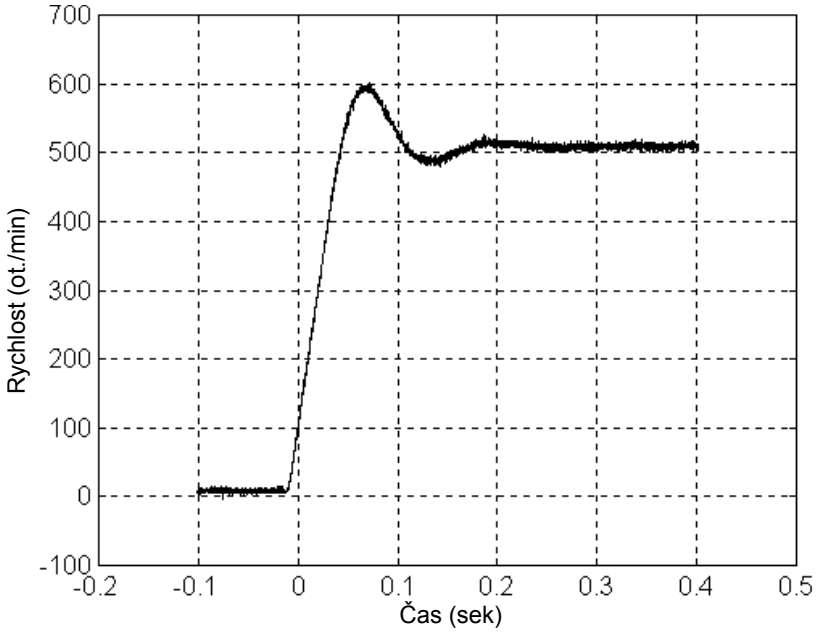
6. Zvyšujte ***Zisk rychlostní smyčky*** v malých krocích a opakujte kroky 4 a 5, dokud nezjistíte nestabilitu signálu rychlostní zpětné vazby motoru. Jakmile dosáhnete tohoto bodu, snižte ***Zisk rychlostní smyčky*** alespoň o 15 %. Obecně platí, že čím nižší je hodnota ***Zisku rychlostní smyčky***, která splňuje požadavky systému, tím robustnější je řízení. Zpětnovazební rychlostní signál je nutno pečlivě sledovat. U některých aplikací použití hodnoty ***Zisku rychlostní smyčky*** dostatečně vysoké k vytvoření nestability může způsobit poškození stroje. Pokud budete mít pochyby, nastavte ***Zisk rychlostní smyčky*** tak, aby nebyl větší než hodnota vypočítaná v rovnici 1. Pokud zjistíte oscilace ve zpětnovazebním signálu rychlosti motoru před tímto bodem, snižte ***Zisk rychlostní smyčky*** a pokračujte krokem 7 níže.

7. Nyní je rychlostní smyčka naladěná. Je však nutno zkontroloval odolnost smyčky. Chcete-li provést tuto zkoušku, zadávejte kroky rychlostního povelu v inkrementech 20% jmenovité rychlosti stroje, 40% jmenovité rychlosti stroje, 60% jmenovité rychlosti stroje, 80% jmenovité rychlosti stroje a 100% jmenovité rychlosti stroje. Na osciloskopu sledujte, jestli se neobjeví nestabilita signálů Rychlost motoru a Povel kroutícího momentu. Pokud zjistíte nestabilitu, snižte ***Zisk rychlostní smyčky*** a test zopakujte.

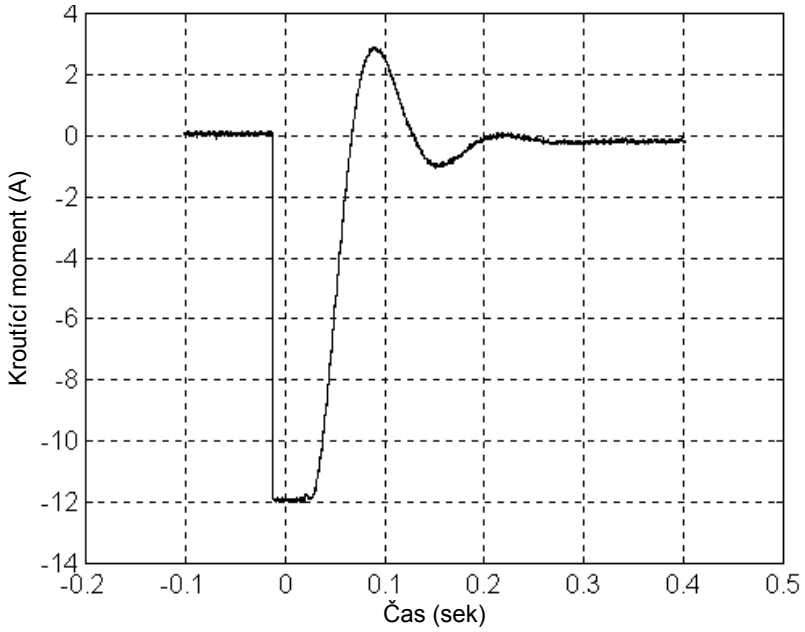
Poznámka: U digitálních serv povel %AQ Vynucený analogový výstup může dát Povel kroutícího momentu nebo Zadanou rychlost motoru. (Rychlost = 750 ot./min/volt a %TqCmd = (100/1.111111 Volt) *X Volt nebo Povel kroutícího momentu = 100% Povel kroutícího momentu = 1.111 V, 100%TqCmd = MaxCur zesilovače. Například: Beta 0.5 MaxCurAmp = 12 A => 1.111111 V = 12 A.

Příklad ladění rychlostní smyčky *DIGITÁLNÍHO REŽIMU*

Příklad ladění rychlostní smyčky je ukázaný v následujících grafech.

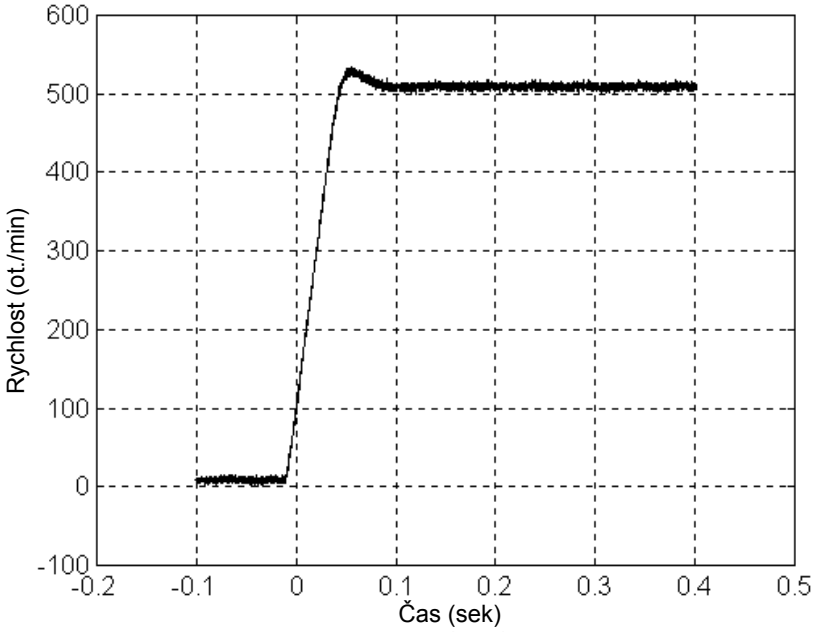


Obrázek D-2. Kroková odezva rychlosti rychlostní smyčky jako funkce času VLGN = 0

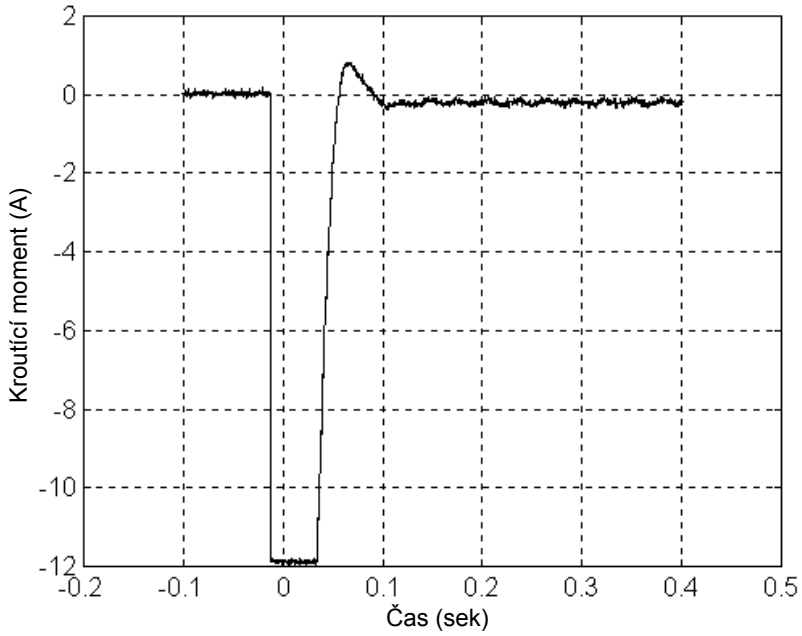


Obrázek D-3. Kroková odezva povelu kroutícího momentu rychlostní smyčky jako funkce času VLGN = 0

Všimněte si, že na obrázcích D-2 a D-3 systém nemá dostatečné tlumení. V takovém případě kontrolér nemá požadovanou šířku pásma a ***Zisk rychlostní smyčky*** je nutno zvýšit.

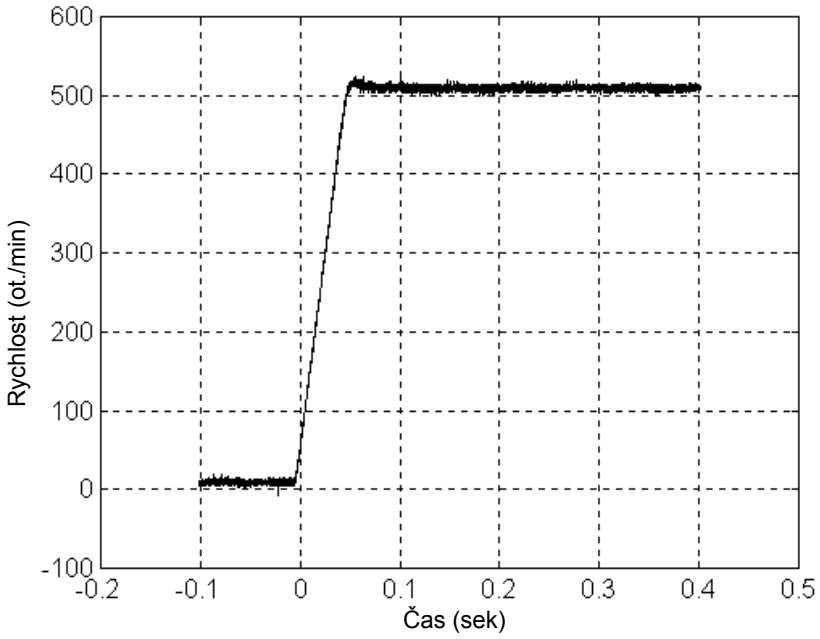


Obrázek D-4. Kroková odezva rychlosti rychlostní smyčky jako funkce času VLGN = 24

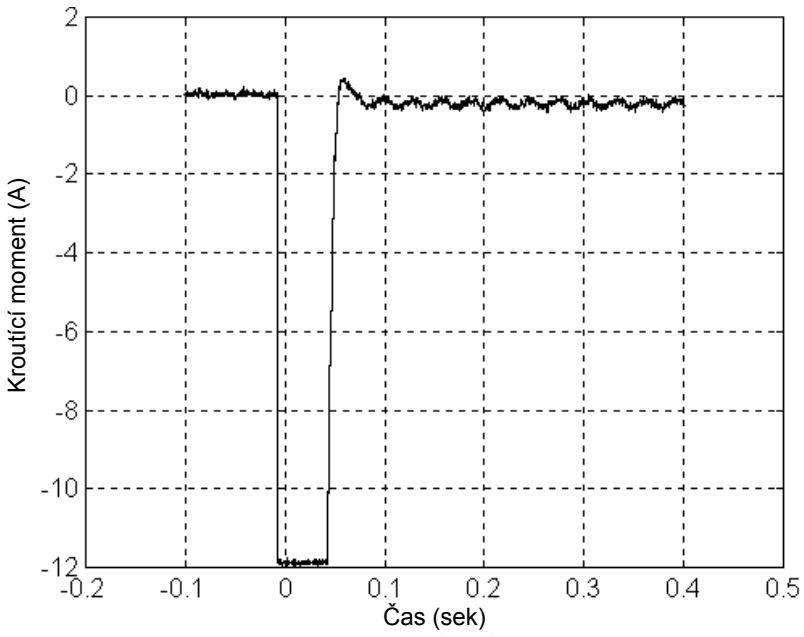


Obrázek D-5. Kroková odezva povelu kroutícího momentu rychlostní smyčky jako funkce času VLGN = 24

Všimněte si, že na obrázcích D-4 a D-5 systém už začíná vypadat přijatelně. Jediným problémem je překmitnutí rychlosti.

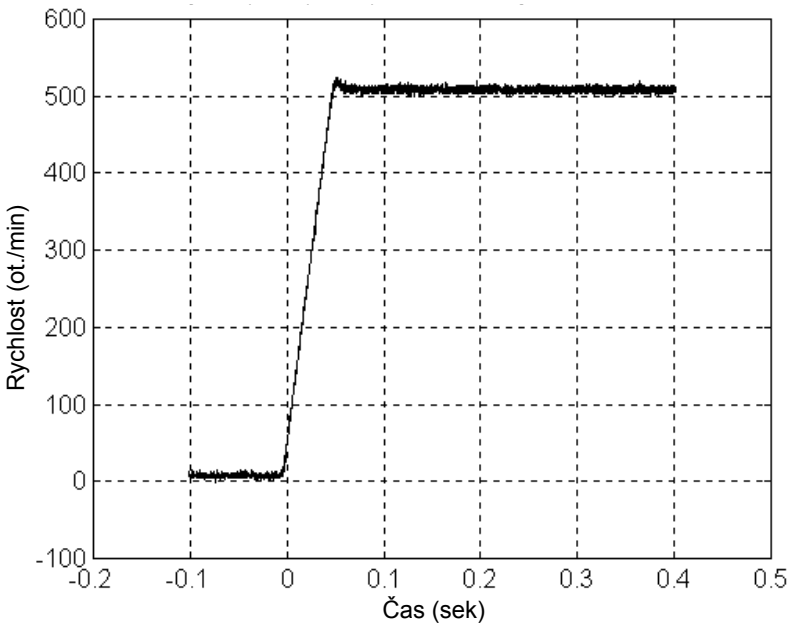


Obrázek D-6. Kroková odezva rychlosti rychlostní smyčky jako funkce času VLGN = 48

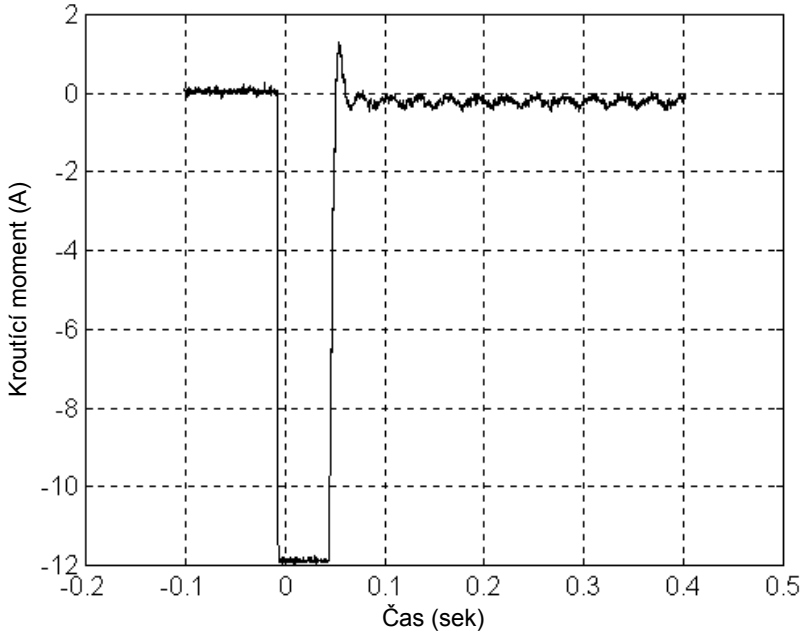


Obrázek D-7. Kroková odezva povelu kroutícího momentu rychlostní smyčky jako funkce času VLGN = 48

Odezva na obrázcích D6 a D7 je dobrá.

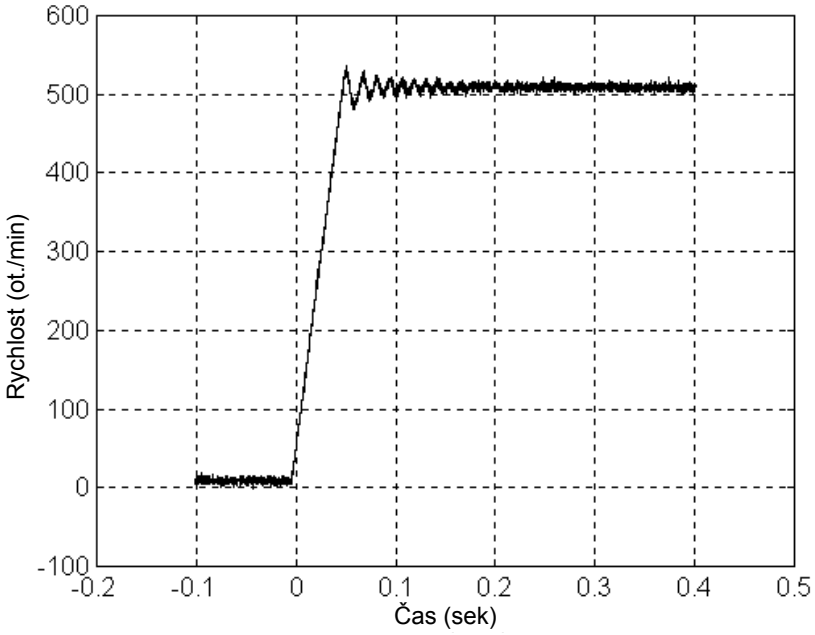


Obrázek D-8. Kroková odezva rychlosti rychlostní smyčky jako funkce času VLGN = 64

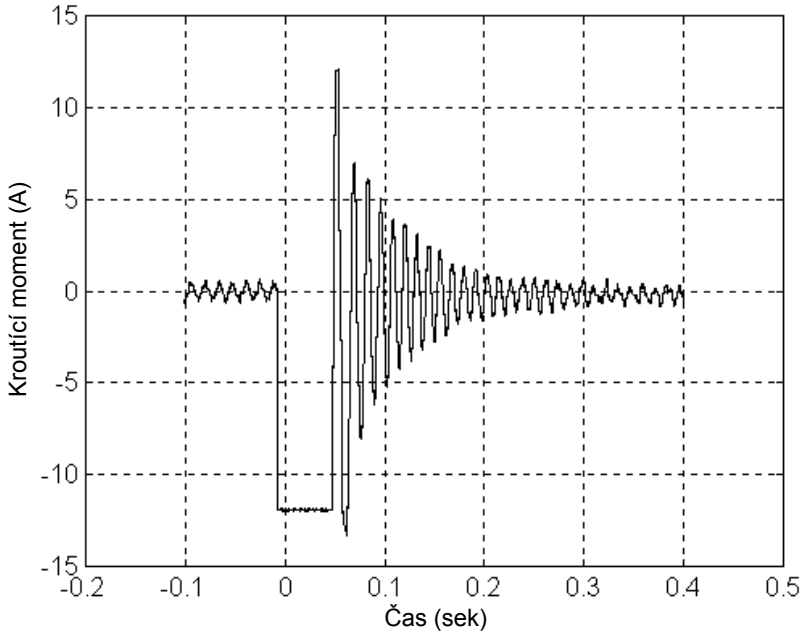


Obrázek D-9. Kroková odezva povelu kroutícího momentu rychlostní smyčky jako funkce času VLGN = 64

Odezva na obrázcích D8 a D9 je vyhovující.



Obrázek D-10. Kroková odezva rychlosti rychlostní smyčky jako funkce času VLGN = 208



Obrázek D-11. Kroková odezva povelu kroutícího momentu rychlostní smyčky jako funkce času VLGN = 208

Odezva ukázaná na obrázcích D-10 a D-11 je kriticky stabilní a v mnoha aplikacích by byla nevyhovující. Grafy jsou uvedené pouze pro orientaci.

Ladění polohové smyčky

Zcela prvním krokem při ladění polohové smyčky je zajistit, aby rychlostní smyčka byla stabilní a měla odezvu vhodnou pro aplikaci. Způsoby nastavení rychlostní smyčky najdete v předchozí části.

Předběžné nastavení polohové smyčky při procesu ladění

1. Pokud budete používat nastavení řídicí smyčky ve standardním režimu, nastavte Uživatelské jednotky a Počet pulsů na hodnoty odpovídající mechanické konfiguraci osy. Podrobnosti viz popis a příklady v kapitole o konfiguraci.
2. Nastavte hodnotu rychlosti při 10 V podle popisu v kapitole o konfiguraci.
3. Volbu Režimu integrátoru nastavte na "OFF".
4. Posuv % nastavte na nulu.
5. Mez polohové odchylky nastavte na hodnotu blízko maxima. Maximum je 60 000 (Uživatelských jednotek / Počet pulsů).

Nastavení zisku polohové smyčky

Polohová řídicí smyčka je především algoritmus "PI" (Proporcionálně-integrační) s volitelným posuvem. Ladění polohové smyčky začněte nastavením proporcionálního zisku (Pos Loop TC), aby se zajistila stabilní odezva s dostatečným ziskem (šířkou pásma), který bude splňovat požadavky profilu pohybu. Nastavení Režimu integrátoru na "OFF" podle popisu v předchozí části vytvoří pouze proporcionální řídicí smyčku. Pro nastavení proporcionálního zisku (Pos Loop TC) jsou dva způsoby.

Metoda 1 proporcionálního zisku polohové smyčky

Výpočet proporcionálního zisku polohové smyčky předpokládá, že mechanické provedení stroje bude mít dostatečnou šířku pásma, aby zůstal stabilní, a že všechny rezonanční kmitočty budou vyšší než šířka pásma požadovaná pro profil pohybu.

Terminologie

Velký nesoulad mezi břemenem a setrvačností motoru může způsobit **REZONANCI** v systému. Rezonance je oscilační chování způsobené mechanickým omezením a zhoršené mrtvým chodem nebo torzním kmitáním mechanických členů, například spojek nebo hřídelí. Rezonance se omezí zlepšením mechaniky, snížením nesouladu břemena/setrvačností motoru nebo snížením zisku serva (snížením výkonu).

ŠÍŘKA PÁSMA je citlivost používaná k porovnání řídicího systému nebo mechanického výkonu. S rostoucí frekvencí povelů odezva systému začne mít prodlevu. Šířka pásma je definovaná jako frekvenční rozsah, ve kterém odezva systému (zisk) je alespoň 70% (-3 decibely) požadovaného povelu.

Velká šířka pásma

- Umožňuje, aby servo přesněji reprodukovalo požadovaný pohyb
- Umožňuje přesnější sledování ostrých rohů po dráze pohybu a při vysokých opakováních strojních cyklů.
- Potlačuje poruchy kroutícího momentu od mechanických nebo vnějších vlivů a zlepšuje tak přesnost systému

- Může odhalit rezonance stroje, které se vyskytují při frekvencích v blízkosti nebo pod šířkou pásma

Odezva pouze proporcionálního systému, která se nastaví nastavením *Režimu integrátoru* na "OFF", má exponenciální náběh. Časová konstanta exponenciální křivky představuje 68% zbývajícího náběhu. Když se například začne nulovou rychlostí, odezva polohové smyčky na změnu v povelu bude k dosažení 68% zadané rychlosti vyžadovat jednu časovou konstantu. Druhá časová konstanta sníží 68% zbývajícího povelu. Následující časové konstanty sníží 68% zbývajícího povelu. Například 100% - 68% (jedna časová konstanta) = 32%, 32% (68%) =21.8%, 68% (první časová konstanta) + 21.8% (druhá časová konstanta) = 89.8%. Dvě časové konstanty budou eliminovat 89.8% potřebného povelu. Tři časové konstanty vykonají 96.7% náběhu povelu. Čtyři časové konstanty vykonají 98.9% náběhu povelu. *Pro většinu aplikací obvykle postačují tři časové konstanty.*

Znalost časových konstant můžete využít k odhadu požadované charakteristiky systému. Pokud se například největší požadované zrychlení v pohybovém profilu musí vykonat během 200 ms, odezva v délce 200 ms na změnu v povelu bude během třech časových konstant z 98,9% kompletní. Vydělením 200 ms třemi dá časovou konstantu asi 67 ms. **Konfigurační pole *Pos Loop TC* představuje jednu časovou konstantu v ms.** V předchozím příkladu jedna časová konstanta je 67 ms.

Metoda 2 proporcionálního zisku polohové smyčky

Je podobná výše uvedené metodě ladění rychlostní smyčky. Použijte osciloskop a postupně snižuje hodnotu Pos Loop TC (zvyšování zisku). Sledujte analogový výstup *Rychlost motoru*, jestli charakteristiky výkonu jsou odpovídající.

Spuštění a informace ladění pro analogové servosystémy

Jsou popsané zde dvě hlavní části;

- Potvrzení spínače výchozí polohy, vstupů přejetí a směru motoru.
- Rychlost při Max Cmd, Časová konstanta polohové smyčky a určení Posuvu rychlostí

Postupy spouštění systémového rychlostního rozhraní analogového režimu

Postupy spouštění

1. Připojte motor k analogovému servozesilovači rychlostního rozhraní podle doporučení výrobce.
2. Připojte výstup relé **Povolit pohon** DSM314 a výstupy **Rychlostních povelů** k servozesilovači. Připojte zařízení polohové zpětné vazby (inkrementální snímač polohy s fázovým posuvem) ke vstupům snímače polohy Motion Mate DSM314.

   Poznámka: Pokud tato připojení budou nesprávná nebo bude docházet k prokluzu ve spojce zpětnovazebního zařízení, při zadání pohybu se může objevit chyba ***porušení synchronizace***.

3. Připojte výstup servozesilovače Připraven (pokud existuje) ke vstupu Pohon připraven (IN_4) na DSM314. Když je zesilovač připravený řídit servo, tento signál se musí přepnout na 0 V. **DSM začne kontrolovat vstup Pohon připraven jednu sekundu po sepnutí relé Povolení pohonu jako odezva na bit %Q Povolit pohon.** Pokud servozesilovač nemá vhodný výstup Připraven, tento vstup DSM314 se musí připojit na 0 V nebo vstup Pohon připraven je možno v konfiguraci modulu zakázat. Pokud se spínač **Výchozí poloha** použije (24 V ss), připojte ho ke správnému vstupu DSM314. Spínač **Výchozí poloha** musí být zapojený tak, aby byl VŽDY V JEDNIČCE, když se osa bude nacházet na záporné straně výchozí polohy, a VŽDY V NULE (rozepnutý), když se osa bude nacházet na kladné straně výchozí polohy.
4. K nastavení požadovaných konfiguračních parametrů použijte konfigurační software. Uložte konfiguraci do PLC.
5. Bit %Q *Pohon povolit* nastavte do jedničky a kód povelu pro *Vynucený D/A výstup* v tabulce %AQ nastavte na 0. Přesvědčte se, že servozesilovač je povolený (motor musí mít přídržný kroutící moment). Pokud se motor bude pohybovat, upravujte povelový offset zesilovače, dokud se motor nezastaví. Poznámka: Aby povel *Vynucený D/A výstup* mohl fungovat, bit %Q *Pohon povolit* musí zůstat v jedničce.
6. Pošlete povelový kód pro *Vynucený D/A výstup* nastavený na +3200 (+1.0 V). Přesvědčte se, že se motor pohybuje v požadovaném KLADNÉM směru (podle nastavení konfiguračního parametru ***Směr osy***) a *Okamžitá rychlost* hlášená do %AI tabulky DSM314 bude KLADNÁ. Pokud se motor bude pohybovat v nesprávném směru, proveďte opravu podle instrukcí výrobce zesilovače. K prohození kladného a záporného směru osy je možno také použít parametr ***Směr osy*** v softwarové konfiguraci. Pokud se motor bude pohybovat v KLADNÉM

směru, ale DSM314 bude hlásit, že *Okamžitá rychlost* je ZÁPORNÁ, pak je nutno zaměnit vstupy snímače polohy kanál A a kanály B.

7. Zaznamenejte aktuální rychlost motoru hlášenou z Motion Mate DSM314 pomocí 1.0 voltového rychlostního povelu. Vynásobte tuto rychlost 10 a pokud bude nutné, aktualizujte záznam ***Rychlost při Max Cmd*** v konfiguraci DSM314. Z počátku nastavte konfigurační parametr ***Časová konstanta polohové smyčky (0.1 ms)*** na vysokou hodnotu (v konfiguraci typicky 100 ms nebo hodnotu 1000).

8. Nastavte bit %Q *Jog Plus* do jedničky. Přesvědčte se, že se servo pohybuje ve správném směru a že *Okamžitá rychlost*, kterou hlásí Motion Mate DSM314 v tabulce %AI, souhlasí s nakonfigurovanou ***Rychlostí jogu***. Pokud pohybové programy používají zrychlení vyšší než ***Zrychlení jogu***, může být nutné zvýšit ***Zrychlení jogu*** tak, aby operace *Zrušit všechny pohyby* a Normální zastavení pracovaly očekávaným způsobem.

9. Když bit %Q *Pohon povolen* bude v jedničce a nebude zadaný žádný pohyb, nastavte offset povelu servopohonu pro nulovou *polohovou odchylku*. Během tohoto procesu musí být integrátor vypnutý.

10. Zkontrolujte správnou činnost cyklu *Nalezení výchozí polohy* krátkodobým nastavením %Q bitu *Nalezení výchozí polohy* do jedničky (%Q bit Pohon povolen *musí být také v jedničce). Osa by se měla pohybovat směrem ke spínači výchozí polohy nakonfigurovanou* ***Rychlostí pro nalezení výchozí polohy***, pak vyhledat značku snímače polohy nakonfigurovanou ***Konečnou rychlostí výchozí polohy***. Pokud bude nutné, upravte rychlosti a polohu **spínače výchozí** polohy pro správnou činnost. Poslední přechod **spínače výchozí polohy** se MUSÍ provést minimálně 10 ms před nalezením značkovacího pulsu snímače polohy. Fyzické umístění ***Výchozí polohy*** je možno upravit změnou hodnoty ***offsetu výchozí polohy*** v konfiguračním softwaru.

11. Sledujte výkon serva a použijte %Q bit *Jog Plus* a *Jog Minus* k vykonání pohybu analogového servomotoru v obou směrech. ***Časovou konstantu polohové smyčky*** je možno přechodně upravit vložením správného povelového kódu do tabulky %AQ. U většiny systémů je možno ***Časovou konstantu polohové smyčky*** snižovat, dokud se nezjistí určitá nestabilita serva, pak se zvýší na hodnotu přibližně o 50% vyšší. Jakmile bude určena správná časová konstanta, konfiguraci DSM314 je nutno aktualizovat pomocí konfiguračního softwaru. ***Posuv rychlostí*** je možno také nastavit na nenulovou hodnotu (typicky 90 – 100 %) pro optimální odezvu serva.

Poznámka: Aby servo fungovalo správně, konfigurační vstup ***Rychlost při Max Cmd*** MUSÍ být nastavený na aktuální rychlost serva (v Uživatelských jednotkách/sec) vyvolanou rychlostním povelem 10 V do zesilovače.

Postupy spouštění systémového rozhraní kroutícího momentu analogového režimu

Postupy spouštění

1. Připojte motor k analogovému rozhraní kroutícího momentu servozesilovače podle doporučení výrobce. **Poznámka:** Zesilovač musí být nakonfigurovaný tak, aby přijímal napětí (±10 V), které odpovídá krouticímu momentu motoru.
2. Připojte výstup relé **Povolit pohon** DSM314 a výstupy **Povelů kroutícího momentu** k servozesilovači. Připojte zařízení polohové zpětné vazby (inkrementální snímač polohy s fázovým posuvem) ke vstupům snímače polohy Motion Mate DSM314.

   Poznámka: Pokud tato připojení budou nesprávná nebo bude docházet k prokluzu ve spojce zpětnovazebního zařízení, při zadání pohybu se může objevit chyba ***porušení synchronizace***.

3. Připojte výstup servozesilovače Připraven (pokud existuje) ke vstupu Pohon připraven (IN_4) na DSM314. Když je zesilovač připravený řídit servo, tento signál se musí přepnout na 0 V. **DSM začne kontrolovat vstup Pohon připraven jednu sekundu po sepnutí relé Povolení pohonu jako odezva na bit %Q Pohon povolit.** Pokud servozesilovač nemá vhodný výstup Připraven, tento vstup DSM314 se musí připojit na 0 V nebo vstup Pohon připraven je možno v konfiguraci modulu zakázat. Pokud se spínač **Výchozí poloha** použije (24 V ss), připojte ho ke správnému vstupu DSM314. Spínač **Výchozí poloha** musí být zapojený tak, aby byl VŽDY V JEDNIČCE, když se osa bude nacházet na záporné straně výchozí polohy, a VŽDY V NULE (rozepnutý), když se osa bude nacházet na kladné straně výchozí polohy.
4. K nastavení požadovaných konfiguračních parametrů použijte konfigurační software. Uložte konfiguraci do PLC. Konkrétní parametry, které je nutno adresovat, jsou následující:

   Povel analogového serva – V konfiguraci musí být nastavený na Krouticí moment. To není výchozí hodnota. Tento konfigurační parametr nastavuje modul, tak aby na analogovém výstupu vygeneroval povel kroutícího momentu. **Poznámka:** Aby fungoval režim analogového kroutícího moment, vyžaduje se firmware DSM verze 3.0 nebo pozdější.

   Rychlost při Max Command – Konfigurační nastavení rychlosti při maximálním povelu určuje maximální rychlost, která bude zadaná pro pohyb serva. Během prvních kroků ladění se doporučuje tuto hodnotu nastavit relativně malou. To umožní ladit systém v několika krocích. Jakmile bude ověřená základní činnost a ladění, maximální hodnotu je možno zvýšit na hodnotu, která je určená buď mezemi procesu nebo nastavením zesilovače/motoru.

   Mez krouticího momentu – Hodnota meze krouticího momentu určuje maximální povel analogového kroutícího momentu, který se pošle do servozesilovače. Během prvních kroků ladění se doporučuje tuto hodnotu nastavit relativně malou. Tato mez kroutícího momentu se nastavuje pomocí povelu %AQ. Informace k tomuto povelu najdete v kapitole 5. Jakmile bude ověřená základní činnost, hodnotu meze krouticího momentu je pak možno nastavit na hodnotu potřebnou pro aplikaci.

   Rozšířené parametry ladění: Sekce rozšířených parametrů ladění obsahuje mnoho parametrů, které se používají k nakonfigurování režimu kroutícího momentu tak, aby pracoval

správně. Rozšířené parametry ladění jsou podrobně popsané v kapitole o konfiguraci. Úplný popis najdete v této kapitole. Parametry, které je nutno naladit, jsou následující:

Parametr ladění 6: Nastavuje parametr rozlišení snímače polohy. Parametr se používá pouze v režimu krouticího momentu. Pro správnou činnost v režimu kroutícího momentu tato hodnota musí být nastavena na počet pulsů snímače polohy s fázovým posuvem generovaných zpětnovazebním zařízením motoru na otáčku. Uživatel může určit hodnotu ze specifikace zpětnovazebního zařízení. Jako dvojnásobnou kontrolu je možno připojit zpětnovazební zařízení k DSM a ručně otočit hřídel motoru o jednu otáčku. Údaj %AI dat DSM pro okamžitou polohu musí přesně souhlasit (odchylky jsou způsobené nepřesným otočením hřídele o jednu otáčku) s hodnotou zapsanou v tomto parametru. Přípustný rozsah je 100 - 32 767 pulsů/otáčku. Výchozí hodnota je 4096 pulsů na otáčku

Parametr ladění 7: Nastaví proporcionální zisk regulátoru rychlosti. Parametr se používá pouze v režimu krouticího momentu. Proporcionální zisk se násobí odchylkou rychlosti (povel rychlosti - rychlostní zpětná vazba) a vygeneruje se část povelu kroutícího momentu odpovídající proporcionální hodnotě. Správné nastavení této hodnoty určuje, jak dobře regulátor rychlosti bude provádět řízení systému. V následujících odstavcích se popisuje, jak nastavit tuto hodnotu.

Parametr ladění 8: Nastaví integrální zisk regulátoru rychlosti. Parametr se používá pouze v režimu krouticího momentu. Integrální zisk je hodnota násobená plochou rychlostní odchylky (povel rychlosti - rychlostní zpětná vazba), která vygeneruje část povelu kroutícího momentu odpovídající integrální hodnotě. Správné nastavení této hodnoty určuje, jak dobře regulátor rychlosti bude provádět řízení systému. V následujících odstavcích se popisuje, jak nastavit tuto hodnotu.

Poznámka: Aby servo fungovalo správně, konfigurační vstup ***Rozlišitelnost snímače polohy*** MUSÍ být nastavený na správnou hodnotu pro nastavený servozesilovač/motor. Pokud tato hodnota nebude nastavená správně, může dojít k nestabilitě.

5. Bit %Q *Pohon povolit* nastavte do jedničky a kód povelu pro *Vynucená rychlost serva* v tabulce %AQ nastavte na 0. Přesvědčte se, že servozesilovač je povolený (motor musí mít přídržný kroutící moment). Pokud se motor bude pohybovat, upravte nastavení zesilovače, dokud se motor nezastaví.

6. Přesvědčte se, že při prvním provádění následující operace je hřídel motoru připojená k břemenu. Nyní je nutno ověřit základní funkci řízení. Pošlete kód povelu pro *Vynucenou rychlost serva* rovnající se 10 ot./min. Přesvědčte se, že se motor pohybuje v požadovaném KLADNÉM směru (podle nastavení konfiguračního parametru ***Směr osy***) a *Okamžitá rychlost* hlášená do %AI tabulky DSM314 bude KLADNÁ. Pokud se motor bude pohybovat v nesprávném směru, proveďte opravu podle instrukcí výrobce zesilovače. K prohození kladného a záporného směru osy je možno také použít parametr ***Směr osy*** v softwarové konfiguraci. Pokud se motor bude pohybovat v KLADNÉM směru, ale DSM314 bude hlásit, že *Okamžitá rychlost* je ZÁPORNÁ, pak je nutno zaměnit vstupy snímače polohy kanál A a kanály B.

7. Když bit %Q *Pohon povolen* bude v jedničce a nebude zadaný žádný pohyb, nastavte servopohon tak, aby se negeneroval žádný pohyb. Aby se tento krok řádně dokončil, integrační zisk rychlostní smyčky MUSÍ být nastavený na 0.

8. Jakmile se dosáhne správné základní činnosti, je nutno provést ladění rychlostní smyčky. Část "Ladění rychlostní smyčky režimu kroutícího momentu" obsahuje základní postup při ladění smyčky. POZNÁMKA: Postup ladění pro rychlostní regulátory režimu kroutícího momentu

jsou ODLIŠNÉ od postupů pro rychlostní regulátory digitálního režimu. Proto se NESMÍ pokračovat laděním polohové smyčky, dokud nebude dokončeno ladění rychlostní smyčky.

9. Jakmile bude dokončeno ladění rychlostních regulátorů, je možno provést ladění a nastavení polohové smyčky. Z počátku nastavte konfigurační parametr ***Časová konstanta polohové smyčky (0.1 ms)*** na vysokou hodnotu (v konfiguraci typicky 100 ms nebo hodnotu 1000).

10. Nastavte bit %Q *Jog Plus* do jedničky. Přesvědčte se, že se servo pohybuje ve správném směru a že *Okamžitá rychlost*, kterou hlásí Motion Mate DSM314 v tabulce %AI, souhlasí s nakonfigurovanou ***Rychlostí jogu***. Pokud pohybové programy používají zrychlení vyšší než ***Zrychlení jogu***, může být nutné zvýšit ***Zrychlení jogu*** tak, aby operace *Zrušit všechny pohyby* a Normální zastavení pracovaly očekávaným způsobem.

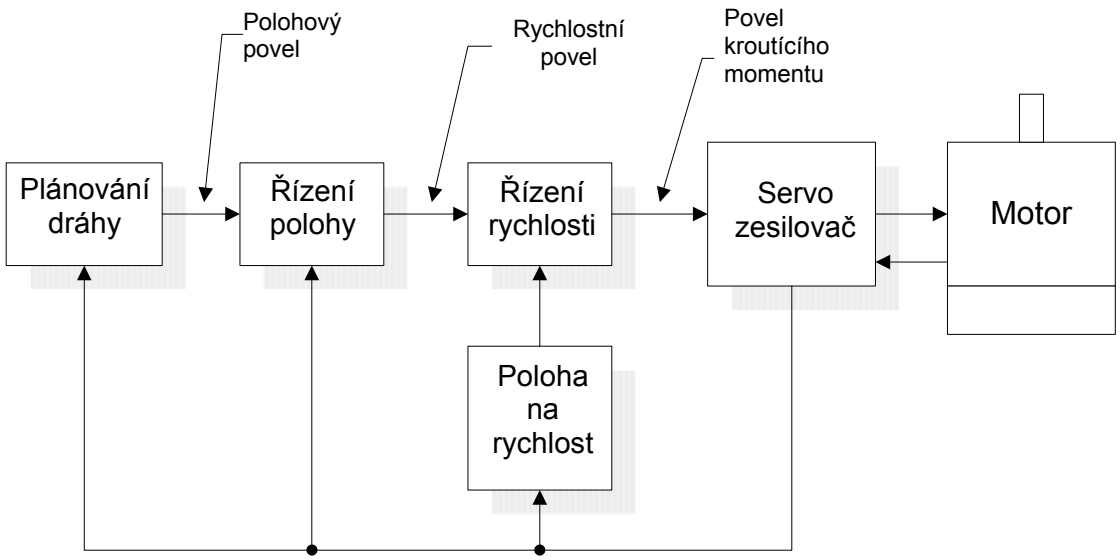
11. Zkontrolujte správnou činnost cyklu *Nalezení výchozí polohy* krátkodobým nastavením %Q bitu *Nalezení výchozí polohy* do jedničky (%Q bit Pohon povolen *musí být také v jedničce).* Osa by se měla pohybovat směrem ke spínači výchozí polohy nakonfigurovanou ***Rychlostí pro nalezení výchozí polohy***, pak vyhledat značku snímače polohy nakonfigurovanou ***Konečnou rychlostí výchozí polohy***. Pokud bude nutné, upravte rychlosti a polohu **spínače výchozí** polohy pro správnou činnost. Poslední **Spínač výchozí polohy** se MUSÍ přepnout alespoň 10 milisekund před tím, než snímač polohy najde značkovací puls. Fyzické umístění ***Výchozí polohy*** je možno upravit změnou hodnoty ***offsetu výchozí polohy*** v konfiguračním softwaru.

12. Sledujte výkon serva a použijte %Q bitu *Jog Plus* a *Jog Minus* k vykonání pohybu analogového servomotoru v obou směrech. ***Časovou konstantu polohové smyčky*** je možno přechodně upravit vložením správného povelového kódu do tabulky %AQ. U většiny systémů je možno ***Časovou konstantu polohové smyčky*** snižovat, dokud se nezjistí určitá nestabilita serva, pak se zvýší na hodnotu přibližně o 50% vyšší. Jakmile bude určena správná časová konstanta, konfiguraci DSM314 je nutno aktualizovat pomocí konfiguračního softwaru. ***Posuv rychlostí*** je možno také nastavit na nenulovou hodnotu (typicky 90 – 100 %) pro optimální odezvu serva.

Poznámka: Aby servo fungovalo správně, konfigurační vstup ***Rychlost při Max Cmd*** MUSÍ být nastavený na maximální rychlost serva (v Uživatelských jednotkách/sec), kterou systém nebo proces umožňuje.

Ladění rychlostní smyčky režimu kroutícího momentu

Správný způsob ladění rychlostní smyčky je oddělit rychlostní smyčku od polohové smyčky. Aby se dosáhlo tohoto oddělení, metoda se musí použít na přímo posílané rychlostní povely, aniž by se použilo řízení polohové smyčky. Modul DSM má několik režimů, které umožňují poslat rychlostní povel přímo do rychlostní smyčky. Tyto dva způsoby jsou:



Obrázek D-12. Blokové schéma rozhraní řídicích smyček kroutícího momentu v analogovém režimu

Způsob č.1:

Okamžitý povel %AQ *Vynucená rychlost serva* (34h) pošle rychlostní povel přímo do rychlostní smyčky. Tento povel se liší od povelu *Pohyb rychlostí,* který k vygenerování povelu používá polohovou smyčku. To je důležité, protože polohová smyčka se nesmí v tomto bodě procesu ladění vzájemně ovlivňovat s rychlostní smyčkou. Povel %AQ *Vynucená rychlost serva* umožňuje vygenerovat krokovou změnu rychlosti. Krok rychlostního povelu se pak použije k vygenerování krokové odezvy rychlostní smyčky. Všimněte si, že když se provede kroková změny rychlostního povelu, zrychlení bude omezené pouze na šířku pásma rychlostní smyčky. U některých aplikací to může způsobit poškození řízeného zařízení v důsledku vysokého zrychlení.

Způsob č.2:

U některých aplikací metoda č.1 řízenému zařízení dává příliš velký náraz. V takových případech je pro vygenerování rychlostního povelu zapotřebí jiná metoda. Tato metoda vyžaduje nastavit polohovou smyčku na konfiguraci otevřené smyčky. Polohová smyčka se nastaví na otevřenou smyčku nastavením ***časové konstanty polohové smyčky*** na nulu a ***Zisku posuvu rychlostí*** na 100 procent. K vygenerování rychlostních povelů do servopohonu pak je možno použít *Povel Pohyb rychlostí* nebo pohybový program.

1. Regulátor rychlosti se naladí následujícím postupem. Doporučuje se, na začátku toto provést, když motor NENÍ připojený k řízenému břemenu. Ladění související s břemenem se provede v následujícím kroku. První parametr, který je nutno nastavit, je ***Proporcionální zisk rychlostní smyčky***. Proporcionální zisk rychlostní smyčky se násobí odchylkou rychlosti (povel rychlosti - zpětná vazba rychlosti) a vygeneruje se část povelu kroutícího momentu odpovídající proporcionální části. Proporcionální část musí být na začátku procesu nastavená na nízkou hodnotu. V závislosti na šířce pásma řízeného servozesilovače může výchozí hodnota 1500 představovat vhodný počáteční bod. Pokud však servozesilovač bude mít malou šířku pásma nebo bude příliš citlivý na změny v povelu kroutícího momentu, může být nutné výchozí hodnotu snížit. Postup ladění umožní se pomocí iterací dostat na konečnou hodnotu. Proto pokud budete mít obavy, začněte s velmi nízkou hodnotou (například 100).

2. Zvolte způsob zavedení rychlostního povelu do rychlostní smyčky. Příkladem provedení tohoto úkolu může být metoda č.1 a metoda č.2 (výše).

3. Připojte osciloskop k analogovým výstupům pro rychlost motoru ze servozesilovače.

4. Na základě předchozího popisu nastavte počáteční hodnotu proporcionálního zisku rychlostní smyčky.

5. Vygenerujte krokovou změnu rychlostního povelu. V tomto bodě kroková změna musí být relativně malá v porovnání s plnou rychlostí stroje. Deset až 20 % jmenovité rychlosti stroje je rozumným začátkem.

6. Na osciloskopu sledujte Rychlost motoru. Cílem je dosáhnout kritické odezvy tlumené rychlostní smyčky. V tomto bodě nejspíš bude ustálená odchylka rychlosti. To se v tomto bodě procesu ladění očekává. Aby se tato odchylka zrušila, integrální část rychlosti se nastaví v následujících krocích. Dávejte pozor zejména na 1. špičku, která se vyskytne, a na všechny oscilace, které se vyskytnou v signálu rychlosti.

7. Zvyšujte ***Proporcionální zisk rychlostní smyčky*** v malých krocích a opakujte kroky 5 a 6, dokud nedosáhnete požadované odezvy. V závislosti na aplikaci to může být kriticky tlumený systém nebo může mít lehké překmitnutí. Obecně platí, že čím nižší je hodnota ***Proporcionálního zisku rychlostní smyčky***, která splňuje požadavky systému, tím robustnější je řízení. Zpětnovazební rychlostní signál je nutno pečlivě sledovat. U některých aplikací použití hodnoty ***Zisku rychlostní smyčky*** dostatečně vysoké k vytvoření nestability může způsobit poškození stroje. Pokud zjistíte oscilace ve zpětnovazebním signálu rychlosti motoru před tímto bodem, snižte ***Proporcionální zisk rychlostní smyčky***.

8. Další parametr, který je nutno nastavit, je ***Integrální zisk rychlostní smyčky***. Integrální zisk rychlostní smyčky je člen násobený plochou rychlostní odchylky (povel rychlosti - zpětná vazba rychlosti) a vygeneruje se část povelu kroutícího momentu odpovídající integrální části. Integrální zisk se typicky používá ke kompenzaci ustálené odchylky rychlosti. Proces ladění začněte tak, že ***Integrální zisk rychlostní smyčky*** bude nastavený na nulu. Při procesu ladění se tato hodnota bude pomalu zvyšovat, dokud se ustálená odchylka neodstraní, aniž by se tím vyvolal velký překmit nebo nadměrné doznívání v odezvě.

9. Zvolte způsob zavedení rychlostního povelu do rychlostní smyčky. Příkladem provedení tohoto úkolu může být metoda č.1 a metoda č.2 (výše).

10. Připojte osciloskop k analogovým výstupům pro rychlost motoru ze servozesilovače.

11. Na základě předchozího popisu nastavte počáteční hodnotu ***Integrální zisk rychlostní smyčky***.

12. Vygenerujte krokovou změnu rychlostního povelu. V tomto bodě kroková změna musí být relativně malá v porovnání s plnou rychlostí stroje. Deset až 20 % jmenovité rychlosti stroje je rozumným začátkem.

13. Na osciloskopu sledujte Rychlost motoru. Cílem je eliminovat ustálenou odchylku, aniž by docházelo k nadměrnému překmitu nebo doznívání. Když budete ladit integrální část, dávejte velký pozor na všechny oscilace, které se vyskytnou v odezvě. Nadměrné oscilace indikují nestabilitu v řídicí smyčce v důsledku příliš velkého integrálního zisku.

14. Zvyšujte ***Integrální zisk rychlostní smyčky*** v malých krocích a opakujte kroky 12 a 13, dokud nedosáhnete požadované odezvy. V závislosti na aplikaci to může být kriticky tlumený systém nebo může mít lehké překmitnutí. Obecně platí, že čím nižší je hodnota ***Integrálního zisku rychlostní smyčky***, která splňuje požadavky systému, tím robustnější je řízení. Zpětnovazební rychlostní signál je nutno pečlivě sledovat. U některých aplikací použití hodnoty ***Integrálního zisku rychlostní smyčky*** dostatečně vysoké k vytvoření nestability může způsobit poškození stroje. Pokud zjistíte oscilace ve zpětnovazebním signálu rychlosti motoru před tímto bodem, snižte ***Integrálního zisk rychlostní smyčky***. Základní rychlostní smyčka je nyní naladěná. Dalším krokem je připojení motoru k břemenu a nastavení parametru ***Zisk rychlostní smyčky*** podle zatížení motoru.

15. Když bude naladěná základní rychlostní smyčka, připojte motor k břemenu. Parametr ***Zisk rychlostní smyčky*** upravuje odezvu rychlostní smyčky pro kompenzaci zátěže. Konkrétně parametr ***Zisk rychlostní smyčky*** upravuje šířku pásma rychlostní smyčky. Jako výchozí bod použijte následující rovnici.

Rovnice 2

$$\text{Zisk rychlostní smyčky} = \frac{J_l}{J_m} \cdot 16$$

Kde:

J_l = Setrvačnost břemena

J_m = Setrvačnost motoru

Výše vypočítaný ***Zisk rychlostní smyčky*** není nutno v mnoha případech měnit. Avšak vzhledem k aplikaci (například rezonance stroje) může být nutné hodnotu upravit. Pro ladění ***Zisku rychlostní smyčky*** je možno použít následující postup:

16. Zvolte způsob zavedení rychlostního povelu do rychlostní smyčky. Příkladem provedení tohoto úkolu může být metoda č.1 a metoda č.2 (výše).

17. Připojte osciloskop k analogovým výstupům ze servozesilovače pro rychlostní zpětnou vazbu motoru.

18. Zisk rychlostní smyčky nastavte na nulu. To je konzervativní přístup. Pokud budete vědět, že aplikace nemá rezonanční frekvence v rozmezí nula až přibližně 250 Hz, můžete začít s vyšší hodnotou, ale nepřekročte hodnotu vypočítanou v rovnici 2.

19. Vygenerujte krokovou změnu rychlostního povelu. V tomto bodě kroková změna musí být relativně malá v porovnání s plnou rychlostí stroje. Deset až 20 % jmenovité rychlosti stroje je rozumným začátkem.

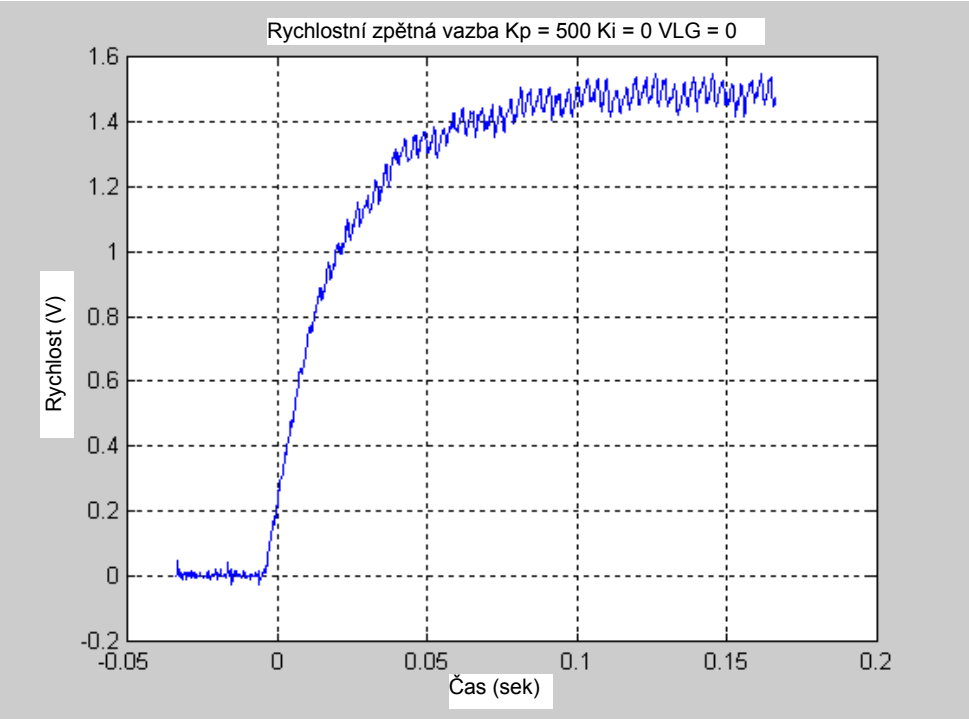
20. Na osciloskopu sledujte Rychlost motoru. Cílem je dosáhnout kritické odezvy tlumené rychlostní smyčky. Zejména dávejte pozor na oscilace, které se vyskytují v signálu rychlostní zpětné vazby.

21. Zvyšujte ***Zisk rychlostní smyčky*** v malých krocích a opakujte kroky 19 a 20, dokud nezjistíte nestabilitu signálu rychlostní zpětné vazby motoru. **Poznámka:** V tomto kroku je nutno dát pozor, aby nestabilita nezpůsobila poškození stroje. Jakmile dosáhnete tohoto bodu, snižte ***Zisk rychlostní smyčky*** alespoň o 15 %. Obecně platí, že čím nižší je hodnota ***Zisku rychlostní smyčky***, která splňuje požadavky systému, tím robustnější je řízení. Zpětnovazební rychlostní signál je nutno pečlivě sledovat. U některých aplikací použití hodnoty ***Zisku rychlostní smyčky*** dostatečně vysoké k vytvoření nestability může způsobit poškození stroje. Pokud budete mít pochyby, nastavte ***Zisk rychlostní smyčky*** tak, aby nebyl větší než hodnota vypočítaná v rovnici 1. Pokud zjistíte oscilace ve zpětnovazebním signálu rychlosti motoru před tímto bodem, snižte ***Zisk rychlostní smyčky*** a pokračujte krokem 22 níže.

22. Nyní je rychlostní smyčka naladěná. Je však nutno zkontroloval odolnost smyčky. Chcete-li provést tuto zkoušku, zadávejte kroky rychlostního povelu v inkrementech 20% jmenovité rychlosti stroje, 40% jmenovité rychlosti stroje, 60% jmenovité rychlosti stroje, 80% jmenovité rychlosti stroje a 100% jmenovité rychlosti stroje. Na osciloskopu sledujte, jestli se neobjeví nestabilita signálů Rychlost motoru a Povel kroutícího momentu. Pokud zjistíte nestabilitu, snižte ***Zisk rychlostní smyčky*** a test zopakujte.

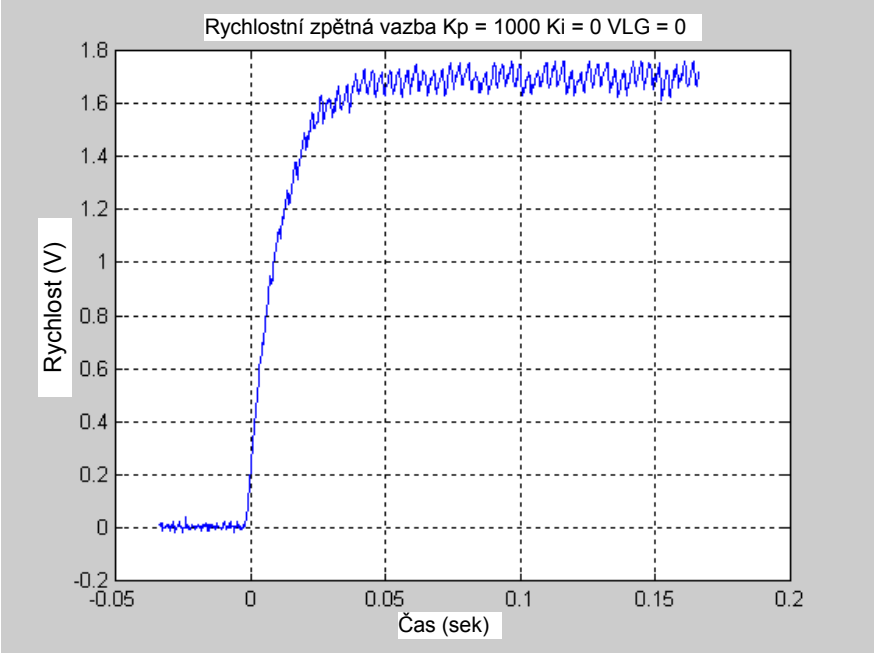
Příklad ladění rychlostní smyčky

Příklad ladění rychlostní smyčky je ukázaný v následujících grafech. Proces se začne laděním ***Proporcionálního zisku rychlostní smyčky***.



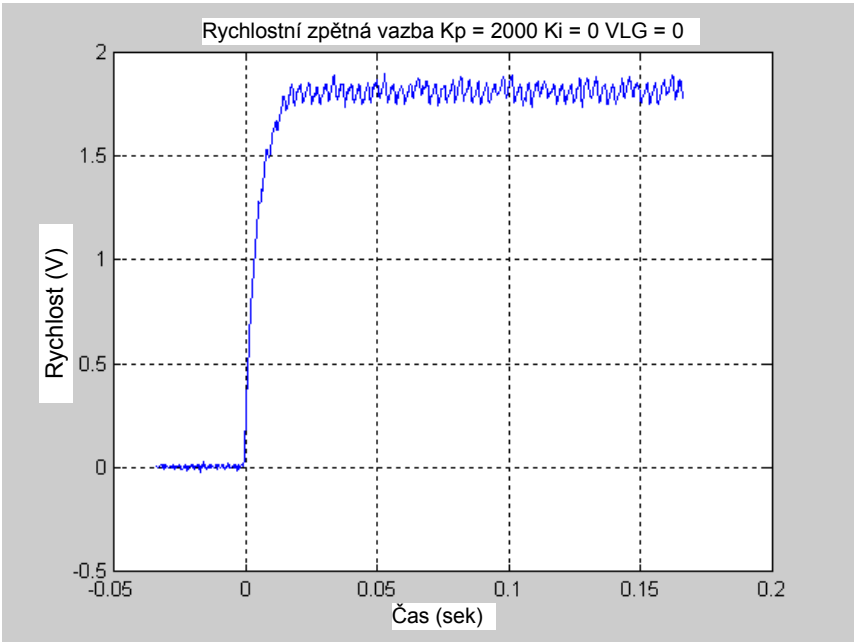
Obrázek D-13. Kroková odezva rychlostní zpětné vazby jako funkce času Kp=500 Ki=0 VLGN = 0

Všimněte si, že systém již má relativně pomalou odezvu. Podle požadované rychlosti zde také je ustálená odchylka. V tomto případě je možno ***Proporcionální zisk rychlostní smyčky*** zvýšit, aby se generovala rychlejší odezva.



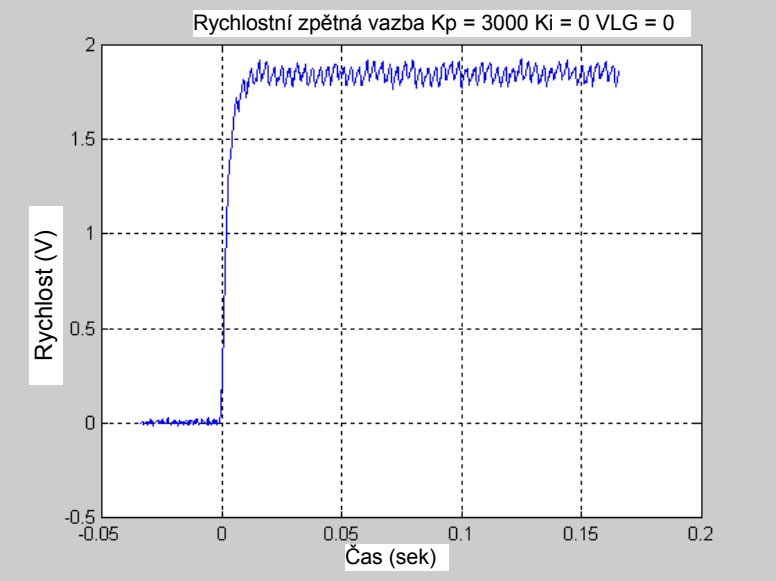
Obrázek D-14. Kroková odezva rychlostní zpětné vazby jako funkce času Kp=1000 Ki=0 VLGN = 0

V obrázku výše se ***Proporcionální zisk rychlostní smyčky*** zvýšil. Doba náběhu se snížila. Systém však je stále možno zlepšit přidáním dalšího ***Proporcionálního zisku rychlostní smyčky***. Stále tu však ale bude ustálená odchylka.



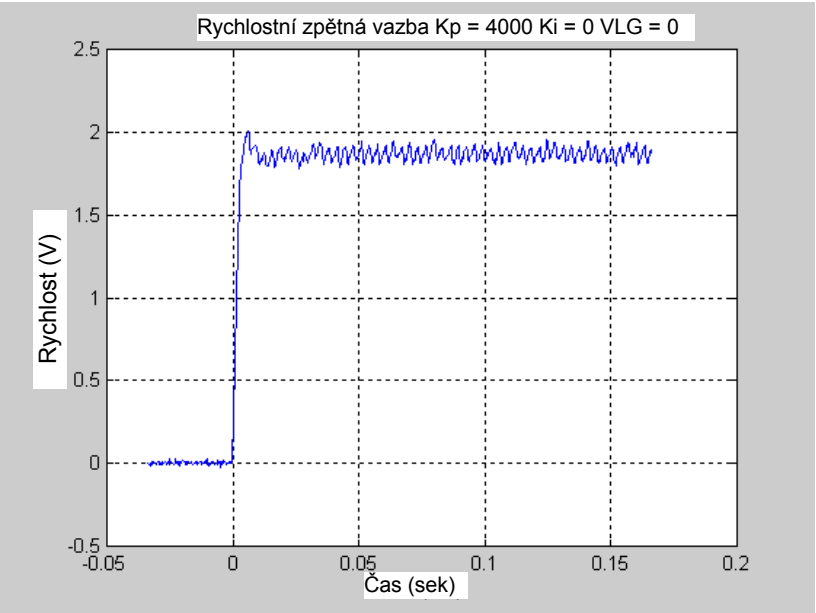
Obrázek D-15. Kroková odezva rychlostní zpětné vazby jako funkce času Kp=2000 Ki=0 VLGN = 0

Proporcionální zisk rychlostní smyčky se opět zvýšil. Zobrazená odezva už začíná vypadat dosti přijatelně. Dobu náběhu je ale možno dále zlepšit.



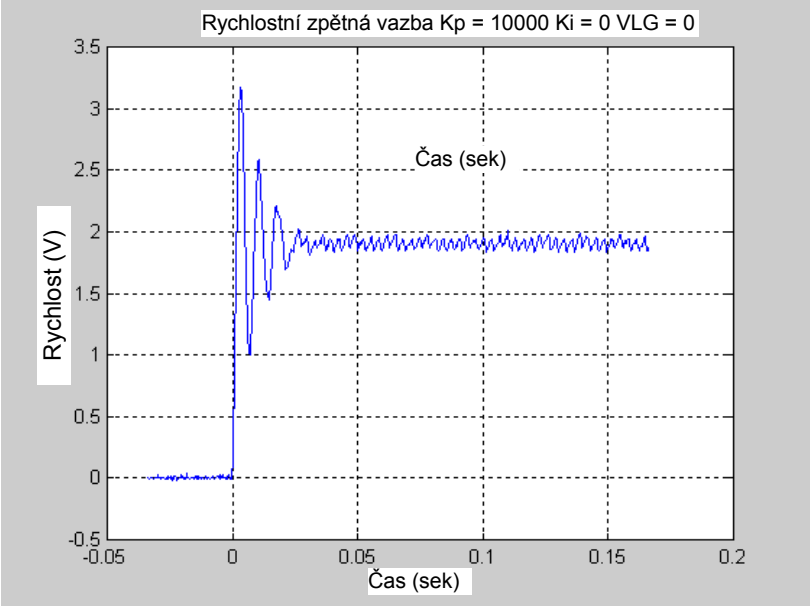
Obrázek D-16. Kroková odezva rychlostní zpětné vazby jako funkce času Kp=3000 Ki=0 VLGN = 0

Výše zobrazená odezva vypadá velmi dobře. Všimněte si malé špičky v odezvě. Abychom experimentovali s odezvou, ***Proporcionální zisk rychlostní smyčky*** dále zvýšíme.



Obrázek D-17. Kroková odezva rychlostní zpětné vazby jako funkce času Kp=4000 Ki=0 VLGN = 0

Odezva zobrazená na obrázku má malý překmit. Tato nebo předchozí odezva by v mnoha aplikacích byla rozhodně vyhovující. Avšak ladění musí být určeno podle schopností stroje. Grafy jsou uvedené pouze pro orientaci.

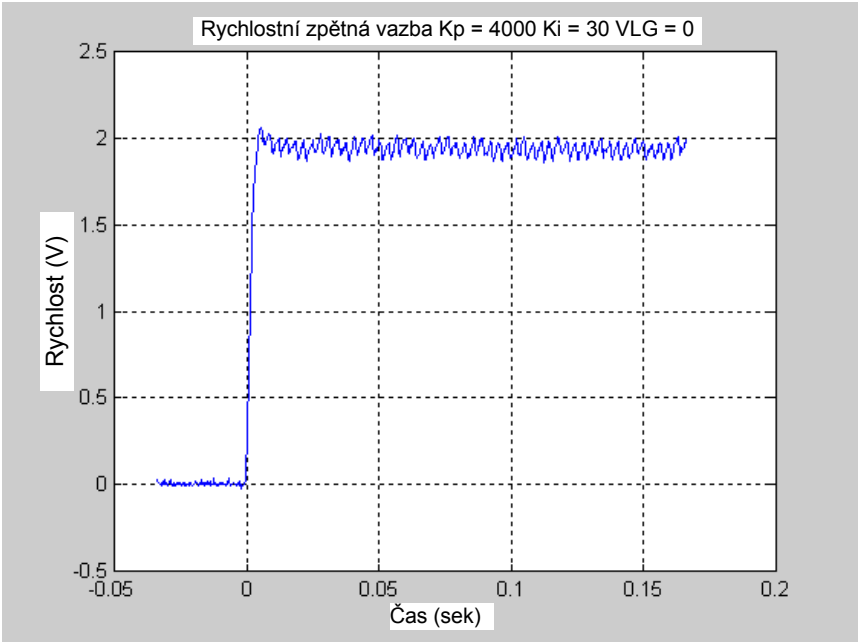


Obrázek D-18. Kroková odezva rychlostní zpětné vazby jako funkce času Kp=10000 Ki=0 VLGN = 0

Výše zobrazený graf znázorňuje nepřijatelnou odezvu. Smyčka vykazuje známky nestability. Všimněte si překmitu a zákmitů za první špičkou. ***Proporcionální zisk rychlostní smyčky*** by se měl výrazně snížit tak, aby se dosáhla odezva s větší stabilitou.

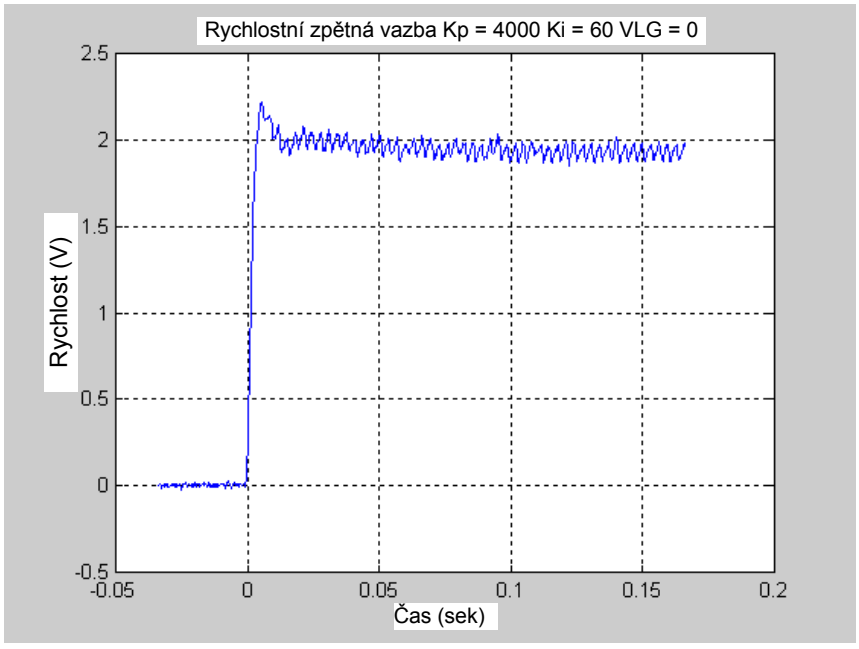
Pro účely tohoto cvičení se odezva odpovídající na ***Proporcionální zisk rychlostní smyčky*** (Kp) = 4000 pro systém zvolí jako požadovaná odezva. Tuto hodnotu je nutno použít, když se provádí ladění ***Integrálního zisku rychlostní smyčky***.

Integrální zisk rychlostní smyčky je z počátku na nule. Můžete provádět malé změny hodnoty a sledovat odezvu.



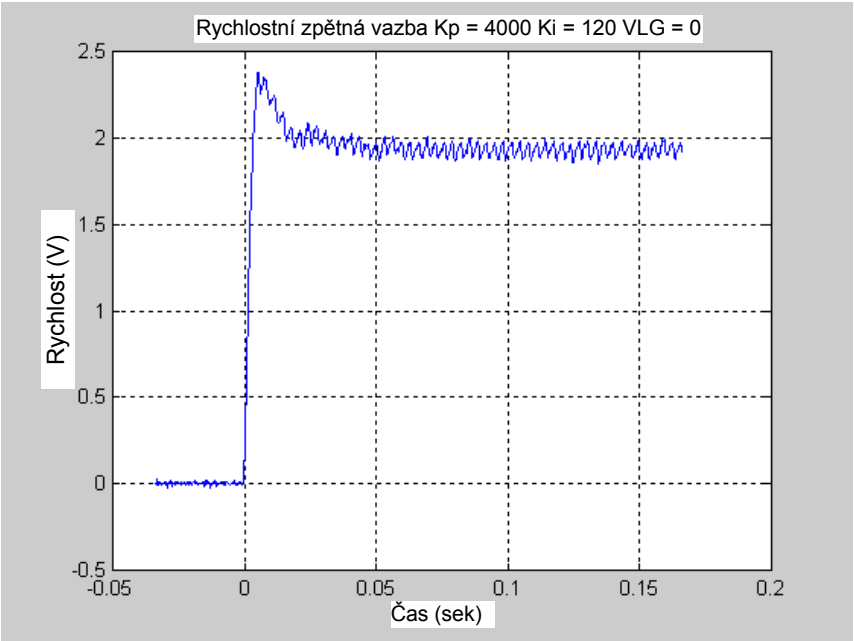
Obrázek D-19. Kroková odezva rychlostní zpětné vazby jako funkce času Kp=4000 Ki=30 VLGN = 0

Odezva výše ukazuje, že ***Integrální zisk rychlostní smyčky*** vede na více vyhovující odezvu. Konkrétně se sníží ustálená odchylka.



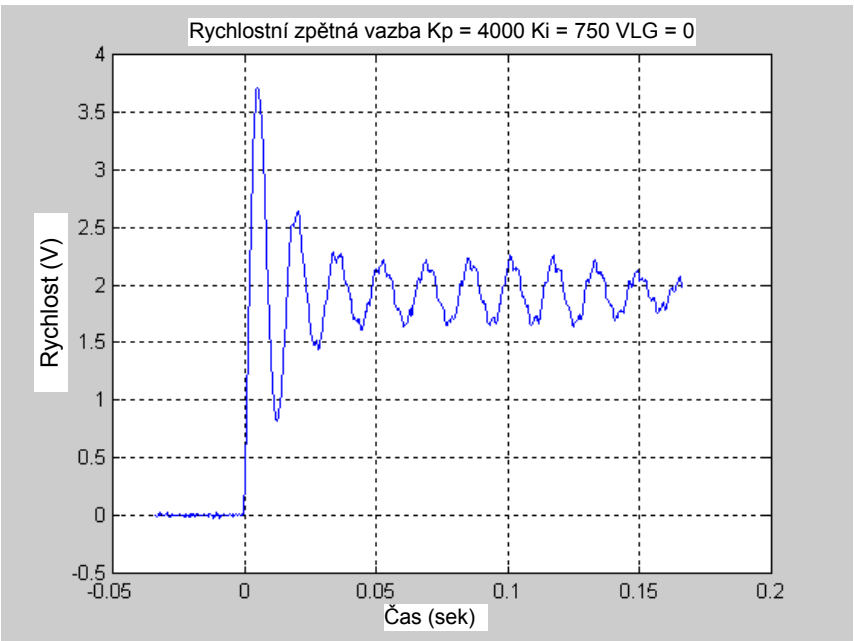
Obrázek D-20. Kroková odezva rychlostní zpětné vazby jako funkce času Kp=4000 Ki=60 VLGN = 0

Když budete zvyšovat dále ***Integrální zisk rychlostní smyčky***, uvidíte začátek překmitu způsobeného integrálním ziskem. Avšak obě odezvy na předchozích dvou obrázcích jsou přijatelné. Konečné zvolené hodnoty závisí na vlastnostech řízeného břemena. Obecně platí, že čím nižší je hodnota ***Proporcionálního zisku rychlostní smyčky*** a ***Integrálního zisku rychlostní smyčky***, která splňuje požadavky systému, tím robustnější je řízení.



Obrázek D-21. Kroková odezva rychlostní zpětné vazby jako funkce času Kp=4000 Ki=120 VLGN = 0

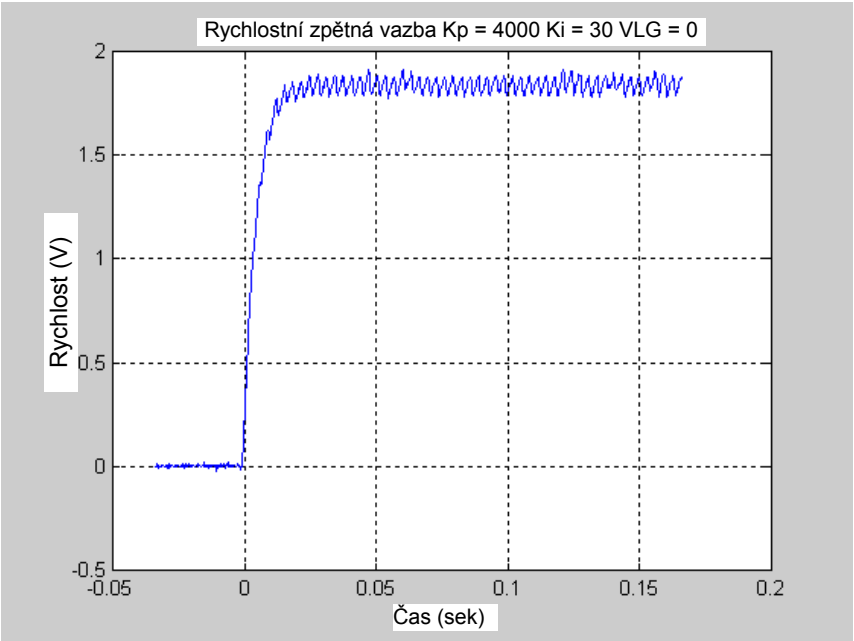
Výše zobrazená odezva znázorňuje příliš velký ***Integrální zisk rychlostní smyčky*** a u většiny aplikací by bude pokládána jako nevyhovující.



Obrázek D-22. Kroková odezva rychlostní zpětné vazby jako funkce času Kp=4000 Ki=7500 VLGN = 0

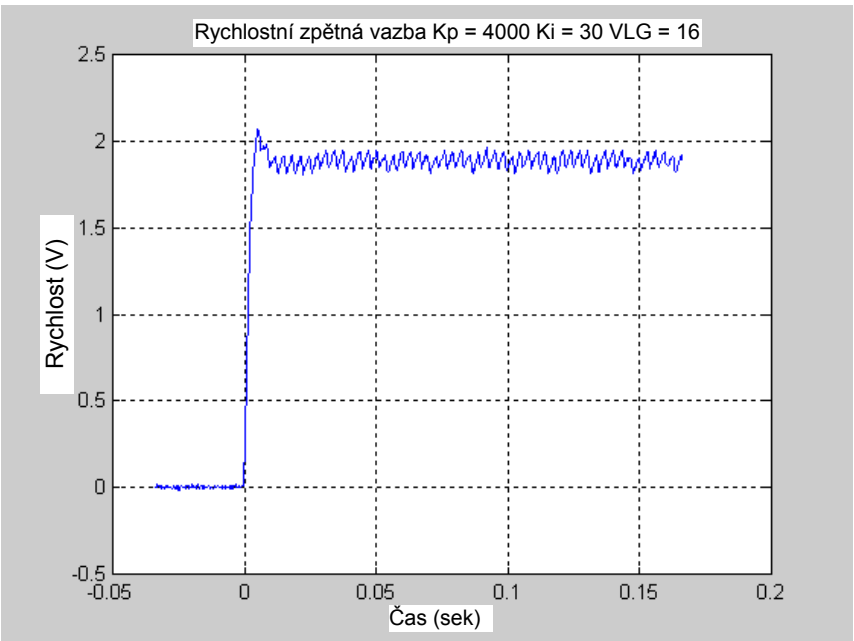
Výše uvedený výsledek představuje hraničně stabilní systém. U této odezvy se nevyskytuje pouze výrazný překmit, ale také zákmity v odezvě rychlosti, které pomalu odeznívají. Tato odezva je nevyhovující.

Dalším krokem procesu ladění je připojit motor k břemenu a pak nastavit řízení tak, aby se dosáhlo požadovaného výkonu. Parametr Zisk rychlostní smyčky umožňuje upravit parametry kontroléru tak, aby se vzalo v úvahu zatížení motoru. Obdobně jako u předchozí procedury začněte se ***Ziskem rychlostní smyčky*** na nule.



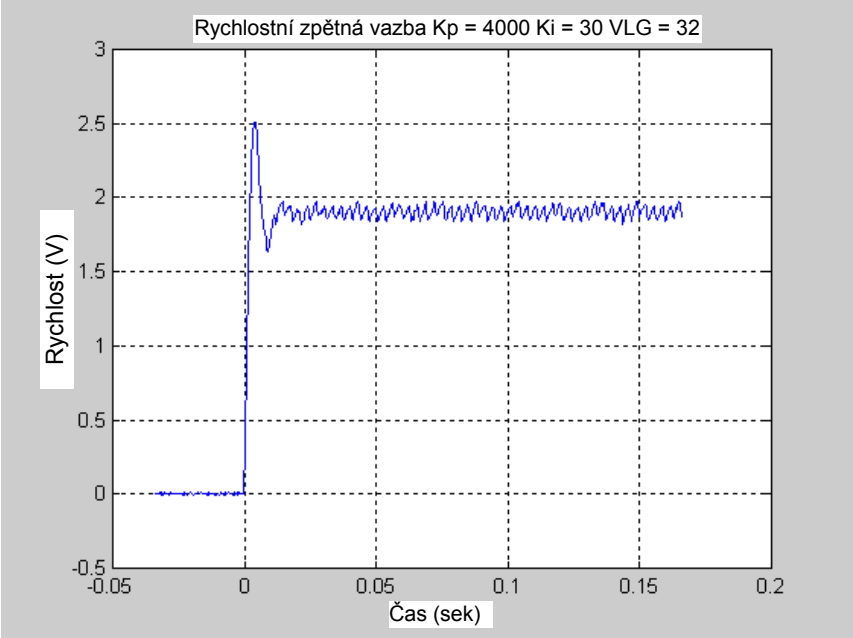
Obrázek D-23. Kroková odezva rychlostní zpětné vazby jako funkce času Kp=4000 Ki=30 VLGN = 0

Výše uvedený obrázek znázorňuje odezvu rychlosti motoru se zátěží připojenou k motoru a motorem naladěným podle předchozího cvičení. Výkon je vyhovující, ale zvýšením ***Zisku rychlostní smyčky*** je možno snížit dobu náběhu.



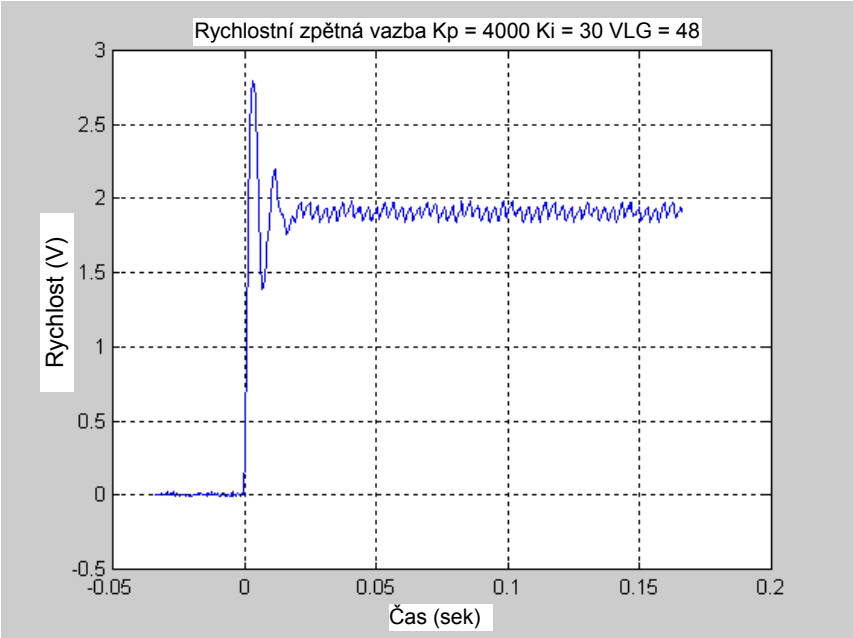
Obrázek D-24. Kroková odezva rychlostní zpětné vazby jako funkce času Kp=4000 Ki=30 VLGN = 16

Výše zobrazená odezva je vyhovující. Odezva má malý překmit, ale bez oscilací a zákmitů.



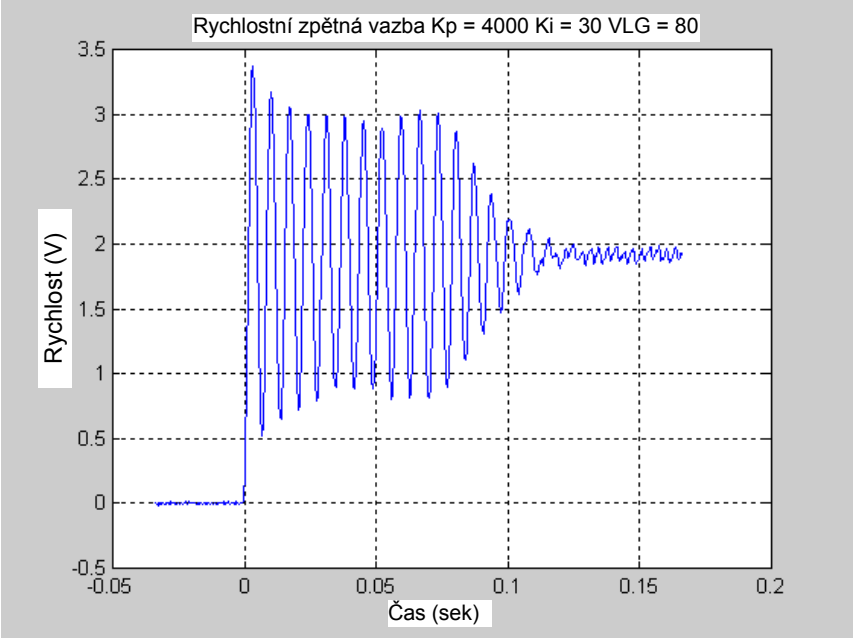
Obrázek D-25. Kroková odezva rychlostní zpětné vazby jako funkce času Kp=4000 Ki=30 VLGN = 32

Odezva zobrazená výše má dost velký překmit, nevyskytují se v ní však žádné nepříznivé vlivy za počátečním překmitem a oscilacemi. Překmit ukazuje, že možná bude dobré snížit ***Zisk rychlostní smyčky***.



Obrázek D-26. Kroková odezva rychlostní zpětné vazby jako funkce času Kp=4000 Ki=30 VLGN = 48

Výše zobrazená odezva vykazuje odezvu s překmitem a patrnými zákmity. Tato odezva začíná indikovat, že zisk rychlostní smyčky je větší než je zapotřebí.



Obrázek D-27. Kroková odezva rychlostní zpětné vazby jako funkce času Kp=4000 Ki=30 VLGN = 80

Výše uvedená odezva představuje hraničně stabilní systém. ***Zisk rychlostní smyčky*** je závažně příliš velký. Všimněte si výrazného překmitu a zákmitů v odezvě. Tato odezva by byla nevyhovující.

Tipy při lokalizaci chyb systému (Analogový režim)

1. DSM314 vyžaduje firmware PLC verze 10.0 nebo vyšší a některý z následujících konfiguračních/programovacích softwarových balíků:
   - CIMPLICITY Machine Edition Logic Developer – PLC verze 2.1 nebo pozdější
   - VersaPro verze 1.1 nebo pozdější
2. Funkce Krouticího momentu vyžaduje firmware DSM verze 3.0 nebo vyšší.
3. Pokud vstup Pohon připraven bude v konfiguraci modulu povolený, vstup musí být připojený k 0 V během 1 sekundy po tom, co relé Povolení pohonu sepne, jinak Motion Mate DSM314 nebude pracovat. Nesprávná konfigurace nebo zapojení vstupu Pohon připraven bude mít za následek, že se v %AI datech *Chybový kód osy* bude hlásit Chybový kód C0h.
4. Řídicí bit %Q POVOLIT POHON musí být trvale nastavený do **1**, jinak nebude povolený žádný jiný pohyb než Jog. Pokud se nevyskytnou žádné chyby STOP (chybové kódy viz Dodatek A), stavový bit %I POHON POVOLEN bude zrcadlit stav bitu %Q POVOLIT POHON. Chyba STOP vynuluje výstupní bit POHON POVOLEN, i když vstupní bit POVOLIT POHON bude stále v **1**. Aby se pohon znovu povolil, chybový stav je nutno opravit a řídicí bit %Q VYNULOVÁNÍ CHYBY nastavit do jedničky na dobu jednoho cyklu PLC.
5. Pokud stavový bit %I CHYBA bude v **1** a %I stavové bity OSA POVOLENA a POHON POVOLEN budou v **0**, pak se vyskytla chyba STOP (stavová LED bude blikat rychle). V tomto stavu DSM314 nebude odpovídat na žádné jiné povely než na řídicí bit %Q VYNULOVÁNÍ CHYBY.
6. Řídicí bit VYNULOVÁNÍ CHYBY používá jednorázovou operaci. Aby se chyba vynulovala, při každém vygenerování chyby se bit musí nastavit **0** a pak do **1** alespoň na jeden cyklus PLC.
7. LED dioda *CFG OK* musí svítit, jinak DSM314 nebude odpovídat na povely PLC. Pokud LED nebude svítit, pak z PLC nepřišla platná konfigurace DSM314 nebo se zjistila chyba konfigurace. Zkontrolujte %AI slova chybového kódu, jestli neobsahují chyby Dxxx, které jsou popsané v části “Kódy systémových chyb” v Dodatku A. Také zkontrolujte tabulku chyb PLC, jestli neobsahuje nahlášené chyby konfigurace.
8. PLC logika nesmí poslat následující bitové povely %Q do DSM314 při prvním cyklu PLC: *Nalezení výchozí polohy, Vykonání pohybového programu, Vykonání lokální logiky*. Pokud by se tyto povely poslaly při prvním cyklu PLC, bude se hlásit chyba a akce se neprovede.
9. PLC logika nesmí do DSM314 při prvním cyklu PLC poslat následující povely %AQ: *Pohyb rychlostí, Povel pohybu*. Pokud by se tyto povely poslaly při prvním cyklu PLC, bude se hlásit chyba a akce se neprovede.

Dodatek

E

Doba vykonávání lokální logiky

Tento dodatek obsahuje informace potřebné ke stanovení doby vykonávání programu lokální logiky.

Data časování vykonávání lokální logiky

Program lokální logiky v DSM se musí kompletně vykonat během **300 mikrosekund**. Překročení doby vykonávání bude mít za následek překročení doby hlídacího obvodu a hlášení chyby. Překročení doby hlídacího obvodu zastaví pohyb os a vykonávání lokální logiky. Data časování uvedená v následující tabulce umožňují vypočítat nejhorší dobu vykonávání programu. Všimněte si, že níže uvedená data představují doby **vykonávání** a ne doby odezvy. Například doba vykonávání potřebná k zápisu hodnoty do proměnných poměru vlečené osy je 0,30 mikrosekundy, avšak doba potřebná ke sledování výsledné změny v pohybu os bude v řádu 2 až 5 milisekund. Obdobně u digitálních vstupů při výpočtu doby odezvy je nutno vzít v úvahu prodlevy hardwarového filtru. **Poznámka:** Pokud doba vykonávání programu bude v rozmezí 300 až 350 mikrosekund, překročení doby hlídacího obvodu se v závislosti na zatížení modulu úlohou nemusí objevit. Aby program běžel bez překročení času, dobu vykonávání programu je nutno udržovat v rozmezí 300 mikrosekund.

Následující tabulky je možno použít pro výpočet nejhorší doby vykonávání a tak zajistit, že program nezpůsobí překročení času hlídacího obvodu. Příklady níže uvádějí výpočet doby vykonávání programu.

Příklad 1

```
P001 := P002 + 3500;                    (* Instrukční řádek 1 *)
IF P001 > 5000 THEN                     (* Instrukční řádek 2 *)
    Torque_Limit_1 := 75;               (* Instrukční řádek 3 *)
     Jog_Plus_1 := Strobe1_Level_1;     (* Instrukční řádek 4 *)
END_IF;                                 (* Instrukční řádek 5 *)
```

Doba vykonávání instrukčního řádku 1 =>

(Doba pro načtení P002) + (Doba pro načtení konstanty) + (Doba pro vykonávání) + (Doba pro zápis P001)

=> 0.60 (z Tabulka E-7) + 0.50 (z Tabulka E-7) + 0.90 (z Tabulka E-1) + 0.60 (z Tabulka E-7)

=> **2.60 mikrosekundy**

Doba vykonávání instrukčního řádku 2 =>

(Doba pro načtení P001) + (Doba pro načtení konstanty) + (Doba pro vykonávání > podmínka)

=> 0.60 (z Tabulka E-7) + 0.50 (z Tabulka E-7) + 2.50 (z Tabulka E-2)

=> **3.60 mikrosekundy**

Doba vykonávání instrukčního řádku 3 (za předpokladu vyhodnocení podmínky jako PRAVDA) =>

(Doba pro načtení konstanty) + (Doba pro zápis Torque_Limit_1)

=> 0.50 (z Tabulka E-7) + 0.30 (z Tabulka E-3)

=> 0.80 mikrosekundy

Doba vykonávání instrukčního řádku 4 (za předpokladu vyhodnocení podmínky jako PRAVDA)=>

(Doba pro načtení Strobe1_Level_1) + (Doba pro zápis Jog_Plus_1)

=> 1.40 (z Tabulka E-3) + 1.70 (z Tabulka E-3)

=> **3.10 mikrosekundy**

Doba vykonávání instrukčního řádku 5=> 0,0 mikrosekundy (z Tabulka E-2)

Celková doba vykonávání => 2.60 + 3.60 + 0.80 + 3.10 +0.0 = **10.10 mikrosekundy**

Příklad 2

```
D00 := P100 * 1000;                          (* Instrukční řádek 1 *)
P101 := D00 / 55;                            (* Instrukční řádek 2 *)
Enable_Follower_1 := CTL01 BWAND CTL02;      (* Instrukční řádek 3 *)
Follower_Ratio_A_1 := P101;                  (* Instrukční řádek 4 *)
```

Doba vykonávání instrukčního řádku 1 =>

(Doba pro načtení P100) + (Doba pro načtení konstanty) + (Dona pro násobení) +(Doba pro zápis D00)

=> 0.60 (z Tabulka E-7) + 0.50 (z Tabulka E-7) + 1.30 (z Tabulka E-1) + 0.70 (z Tabulka E-7)

=> **3.10 mikrosekundy**

Doba vykonávání instrukčního řádku 2 =>

(Doba pro načtení D00) + (Doba pro načtení konstanty) + (Doba pro vykonání dělení) + (Doba pro zápis P101)

=> 0.70 (z Tabulka E-7) + 0.50 (z Tabulka E-7) + 2.90 (z Tabulka E-1) + 0.60 (z Tabulka E-7)

=> 4.70 mikrosekundy

Doba vykonávání instrukčního řádku 3 =>

(Doba pro načtení CTL01) + (Doba pro načtení CTL02) + (Doba pro vykonání BWAND) + (Doba pro uložení Enable_Follower_1)

=> 1.40 (z Tabulka E-7) + 1.40 (z Tabulka E-7) + 0.20 (z Tabulka E-1) + 1.70 (z Tabulka E-3)

=> **4.70 mikrosekundy**

Doba vykonávání instrukčního řádku 4 =>

(Doba pro načtení P101) + (Doba pro zápis Follower_Ratio_A_1)

=> 0.60 (z Tabulka E-7) + 0.30 (z Tabulka E-3)

=> 0.90 mikrosekundy

Celková doba vykonávání => 3.10 + 4.70 + 4.70 + 0.90 = **13.40 mikrosekundy**

Tabulka E-1. Doby vykonávání matematických/logických operací lokální logiky

Matematické a logické operace lokální logiky (Přiřazení, :=)	Doba vykonávání lokální logiky (v mikrosekundách)
Sčítání (+)	0.90**
Odčítání (-)	0.90**
Násobení (*)	1.30
Dělení (/)	2.90
Dělení modulo (MOD)	2.90
Absolutní hodnota (ABS)	1.70**
BWAND	0.20
BWOR	0.30
BWXOR	0.20
BWNOT	0.50

****Poznámka:** Doby vykonávání pro sčítání, odčítání a absolutní hodnotu (ABS) předpokládají, že nedochází k přetečení.

Tabulka E-2. Doby vykonávání podmíněných operací lokální logiky

Podmíněné operace lokální logiky (IF…THEN)	Doba vykonávání lokální logiky (v mikrosekundách)
Větší než (>)	2.50
Menší než (<)	2.50
Větší/Rovno (>=)	2.50
Menší/rovno (<=)	2.50
Rovná se (=)	2.30
Nerovná se (<>)	2.30
BWAND	1.40
BWOR	1.40
BWXOR	1.40
BWNOT	1.60
Prázdný operátor (***IF var THEN***)	1.10
END_IF	0.00

Poznámka: Doba vykonávání pro podmínky je pro případ, kdy se operace IF…THEN vyhodnotí jako NEPRAVDA. To představuje nejhorší případ doby vykonávání, protože doba vykonávání požadovaná pro vyhodnocení podmínky, která je PRAVDA, je menší. Všimněte si, že instrukce END_IF nevyžaduje dobu vykonávání.

Tabulka E-3. Doby vykonávání proměnných lokální logiky pro osu 1

X - Nepoužívá se

Název proměnné lokální logiky	Doba vykonávání lokální logiky (v mikrosekundách)	
	Čtení	Zápis
Strobe1_Level_1	1.40	X
Strobe2_Level_1	1.40	X
Positive_EOT_1	1.40	X
Negative_EOT_1	1.40	X
Home_Switch_1	1.40	X
Digital_Output1_1	X	1.80
Digital_Output3_1	X	1.80
Analog_Input1_1	0.80	X
Analog_Input2_1	0.80	X
Position_Loop_TC_1	X	0.30
Follower_Ratio_A_1	X	0.30
Follower_Ratio_B_1	X	0.30
Torque_Limit_1	X	0.30
Position_Increment_Cts_1	X	0.30
Velocity_Loop_Gain_1	0.80	0.20
Reset_Strobe1_1	X	1.70
Reset_Strobe2_1	X	1.70
Enable_Follower_1	X	1.70
Jog_Plus_1	X	1.70
Jog_Minus_1	X	1.70
FeedHold_1	X	1.70
Error_Code_1	0.80	X
Actual_Position_1	0.70	X
Strobe1_Position_1	0.80	X
Strobe2_Position_1	0.80	X
Actual_Velocity_1	0.80	X
Block_1	0.90	X
Commanded_Position_1	0.60	X
Position_Error_1	0.60	X
Commanded_Velocity_1	0.60	X
User_Selected_Data1_1	0.60	X
User_Selected_Data2_1	0.60	X
UnAdjusted_Actual_Position_Cts_1	0.80	X
UnAdjusted_Strobe1_Position_Cts_1	0.80	X
UnAdjusted_Strobe2_Position_Cts_1	0.80	X
Commanded_Torque_1	0.80	X
Axis_OK_1	1.40	X
Position_Valid_1	1.40	X
Strobe1_Flag_1	1.40	X
Strobe2_Flag_1	1.40	X
Drive_Enabled_1	1.40	X
Program_Active_1	1.40	X
Moving_1	1.40	X
In_Zone_1	1.40	X
Position_Error_Limit_1	1.40	X
Torque_Limited_1	1.40	X
Servo_Ready_1	1.40	X
Follower_Enabled_1	1.40	X
Follower_Ramp_Active_1	1.40	X
Follower_Velocity_Limit_1	1.40	X

Tabulka E-4. Doby vykonávání proměnných lokální logiky pro osu 2

X - Nepoužívá se

Název proměnné lokální logiky	Doba vykonávání lokální logiky (v mikrosekundách)	
	Čtení	Zápis
Strobe1_Level_2	1.40	X
Strobe2_Level_2	1.40	X
Positive_EOT_2	1.40	X
Negative_EOT_2	1.40	X
Home_Switch_2	1.40	X
Digital_Output1_2	X	1.80
Digital_Output3_2	X	1.80
Analog_Input1_2	0.80	X
Analog_Input2_2	0.80	X
Position_Loop_TC_2	X	0.30
Follower_Ratio_A_2	X	0.30
Follower_Ratio_B_2	X	0.30
Torque_Limit_2	X	0.30
Position_Increment_Cts_2	X	0.30
Velocity_Loop_Gain_2	0.80	0.20
Reset_Strobe1_2	X	1.70
Reset_Strobe2_2	X	1.70
Enable_Follower_2	X	1.70
Jog_Plus_2	X	1.70
Jog_Minus_2	X	1.70
FeedHold_2	X	1.70
Error_Code_2	0.80	X
Actual_Position_2	0.70	X
Strobe1_Position_2	0.80	X
Strobe2_Position_2	0.80	X
Actual_Velocity_2	0.80	X
Block_2	0.90	X
Commanded_Position_2	0.60	X
Position_Error_2	0.60	X
Commanded_Velocity_2	0.60	X
User_Selected_Data1_2	0.60	X
User_Selected_Data2_2	0.60	X
UnAdjusted_Actual_Position_Cts_2	0.80	X
UnAdjusted_Strobe1_Position_Cts_2	0.80	X
UnAdjusted_Strobe2_Position_Cts_2	0.80	X
Commanded_Torque_2	0.80	X
Axis_OK_2	1.40	X
Position_Valid_2	1.40	X
Strobe1_Flag_2	1.40	X
Strobe2_Flag_2	1.40	X
Drive_Enabled_2	1.40	X
Program_Active_2	1.40	X
Moving_2	1.40	X
In_Zone_2	1.40	X
Position_Error_Limit_2	1.40	X
Torque_Limited_2	1.40	X
Servo_Ready_2	1.40	X
Follower_Enabled_2	1.40	X
Follower_Ramp_Active_2	1.40	X
Follower_Velocity_Limit_2	1.40	X

Tabulka E-5. Doby vykonávání proměnných lokální logiky pro osu 3

X - Nepoužívá se

Název proměnné lokální logiky	Doba vykonávání lokální logiky (v mikrosekundách)	
	Čtení	Zápis
Strobe1_Level_3	1.40	X
Strobe2_Level_3	1.40	X
Positive_EOT_3	1.40	X
Negative_EOT_3	1.40	X
Home_Switch_3	1.40	X
Digital_Output1_3	X	1.80
Digital_Output3_3	X	1.80
Analog_Input1_3	0.80	X
Analog_Input2_3	0.80	X
Reset_Strobe1_3	X	1.70
Reset_Strobe2_3	X	1.70
Error_Code_3	0.80	X
Actual_Position_3	0.70	X
Strobe1_Position_3	0.80	X
Strobe2_Position_3	0.80	X
Actual_Velocity_3	0.80	X
Axis_OK_3	1.40	X
Position_Valid_3	1.40	X
Strobe1_Flag_3	1.40	X
Strobe2_Flag_3	1.40	X

Tabulka E-6. Doby vykonávání proměnných lokální logiky pro osu 4

X - Nepoužívá se

Název proměnné lokální logiky	Doba vykonávání lokální logiky (v mikrosekundách)	
	Čtení	Zápis
Strobe1_Level_4	1.40	X
Strobe2_Level_4	1.40	X
Positive_EOT_4	1.40	X
Negative_EOT_4	1.40	X
Home_Switch_4	1.40	X
Digital_Output1_4	X	1.80
Digital_Output3_4	X	1.80
Analog_Input1_4	0.80	X
Analog_Input2_4	0.80	X

Tabulka E-7. Celkové doby vykonávání proměnných lokální logiky

X - Nepoužívá se

Název proměnné lokální logiky	Doba vykonávání lokální logiky (v mikrosekundách)	
	Čtení	Zápis
Konstanty programu lokální logiky	0.50	X
Overflow	2.40	1.30
System_Halt	X	1.80
Data_Table_Ptr	0.60	0.70
Data_Table_sint	2.10	1.70
Data_Table_usint	1.80	1.70
Data_Table_int	2.30	2.20
Data_Table_uint	2.30	2.20
Data_Table_dint	3.80	4.00
Module_Error_Present	1.40	X
New_Configuration_Received	1.40	X
First_Local_Logic_Sweep	1.40	X
Module_Status_Code	0.50	X
CTL_1_to_32	0.50	X
P000-P255	0.60	0.60
D00-D07	0.70	0.70
CTL01-CTL32	1.40	1.80

Dodatek

F

Aktualizace firmwaru v DSM314

Firmware pro činnost DSM314 je uložený na desce ve FLASH paměti. Aktualizace firmwaru se provádí přes disketu. Obslužný program PC Loader řídí načtení nového firmwaru z diskety do FLASH paměti DSM314 a je programem na bázi DOS. Winloader je k dispozici také pro Windows. PC Loader vyžaduje IBM AT/PC kompatibilní počítač s minimální pamětí 640K RAM, disketovou jednotkou, MS-DOS®3.3 (nebo vyšší) a jedním sériovým portem RS-232. Aby tento obslužný program běžel v MS-DOS pod Windows® 3.1, Windows® 95 nebo Windows NT®, procesor musí být minimálně Pentium™ 133. Pokud nebude, počítač je nutno rebootovat v režimu MS-DOS. PC Loader funguje optimálně s minimálním paměťovým prostorem 1 MB. Obslužný program pro aktualizaci Winloaderu vyžaduje Windows 95, Windows NT nebo Windows 98. Hardware vyžadovaný pro spuštění těchto operačních systémů by měl stačit také pro spuštění Winloaderu. Winloader vyžaduje asi 500 Kbajtů volného místa na pevném disku.

Výstraha

Před otevřením režimu Bootování se uživatel MUSÍ přesvědčit, že PC je připojené k DSM (a ne CPU PLC nebo jinému modulu, který podporuje upgrade FLASH firmwaru). Jinak může dojít ke ztrátě programu a konfigurace CPU PLC.

Chcete-li nainstalovat nový firmware, vykonejte následující kroky:

1. Před vykonáním funkce aktualizace uložte nebo si udělejte zálohu všech programů nebo dat nacházejících se v modulu.
2. PLC přepněte do režimu STOP/NOIO. (Vynulujte všechny chyby.)
3. Přesvědčte se, že přenosová rychlost sériového portu SNP modulu je nastavená na 19200 baud.
4. Pomocí kabelu k propojení Station Manager s PC, IC693CBL316, připojte příslušný sériový port svého počítače (master) k modulu DSM302, který se má aktualizovat (slave).

Aktualizace pro DOS

Poznámka: Tato část platí pouze pro ty, kteří spouští DOS program Loader z DOSu. Pokud používáte software Windows, postupujte podle dalšího odstavce “Aktualizaci Windows”.

5. (DOS) Vložte disketu se štítkem do jednotky A: nebo B:. Přesvědčte se, že disketa není chráněná proti zápisu. Po požadavku "Unzip to folder:" (rozbalit do adresáře) spusťte samorozbalovací archív zadáním A: nebo B: jako cíl.
6. (DOS) Na výzvu C:\> zapište A:install (nebo B:install, pokud vaše disketová jednotka je B:). Instalace programu nakopíruje několik souborů na pevný disk a pak vyvolá PC Loader. Pokud na pevném disku nebude dost místa, instalaci je také možno spustit přímo z diskety. V tomto případě zapište A:\> nebo B:\>.

7. (DOS) Pokud kabel nebude připojený k COM1, z hlavního menu stisknutím tlačítka F3 nakonfigurujte správný sériový port. Stisknutím tlačítka TAB můžete přepínat mezi různými volbami a stisknutím ENTER se zobrazená volba přijme.
8. (DOS) Z hlavního menu stisknutím F1 se provede připojení ke slave zařízení DSM302.
9. (DOS) Jakmile bude slave zařízení připojené, objeví se menu bootovacího režimu – stisknutím F1 vyvolejte REŽIM BOOT a stisknutím 'Y' operaci potvrďte. Diody STAT a CFG na přední straně modulu by nyní měly blikat jednotně.
10. (DOS) Když budete v bootovacím režimu, stisknutím tlačítka F1 proveďte načtení nového firmwaru.
11. (DOS) Stisknutím tlačítka **Y** operaci potvrďte. Načtení by mělo trvat asi 4 minuty. Pokud se načtení neprovede, postupujte podle odstavce *Restartování přerušené aktualizace firmwaru*.
12. (DOS) Když se načtení vykoná úplně, PC Loader vás požádá o vypnutí a zapnutí modulu. Nyní tedy proveďte vypnutí a zapnutí modulu. Pokud modul bude nainstalovaný v expanzní nebo vzdálené sestavě, je nutno také provést vypnutí a zapnutí hlavní sestavy.
13. (DOS) Označte jednotku nainstalovanou verzí firmwaru. Pokud firmware bude Beta nebo technická verze, uveďte to na štítku.

Aktualizace pro Windows (pro Windows 95/NT/98, NE Windows 3.1)

Poznámka: Tato část platí pro ty, kdo používají pro aktualizaci software Winloader s Windows 95, NT nebo 98. Pokud budete používat operační systém DOS, postupujte podle odstavce "Aktualizace pro DOS".

5. (WIN) Vložte disketu se štítkem do jednotky A: nebo B:. Přesvědčte se, že disketa není chráněná proti zápisu. Po požadavku "Unzip to folder:" (rozbalit do adresáře) spusťte samorozbalovací archív zadáním A: nebo B: jako cíl.
6. (WIN) Vyvolejte software Winloader dvojnásobným kliknutím na jeho ikonu umístěnou na jednotce A: nebo B: (v závislosti na označení jednotky pro 3.5" disketu) ve Windows Exploreru nebo prostě ho vykonejte tak, že z menu start zvolíte RUN. V okně RUN zapište A (nebo B):winloader.exe.
7. (WIN) Ukládání softwaru začněte jedním kliknutím na tlačítko "Update".
8. (WIN) Po skončení aktualizace se "objeví" okno indikující stav aktualizace. Pokud aktualizace byla úspěšná, proveďte vypnutí a zapnutí PLC a kliknutím na "NO" udejte jiné zařízení, které se NEMÁ aktualizovat. Pokud aktualizace nebyla úspěšná, zkuste změnit Nastavení/Rychlost přenosu z 38400 na 19200 a postup zopakujte.

Restartování přerušené aktualizace firmwaru

A. Připojte všechny kabely podle popisu v kroku 4 popisu výše.
B. Proveďte vypnutí a zapnutí sestavy obsahující modul. Pokud se provedlo částečné nebo chybné načtení, na modulu po zapnutí budou shodně blikat LED diody STAT a CFG.
C. Pokud na PC budete mít stále spuštěný PC Loader nebo Winloader, přeskočte na krok D níže; jinak pokračujte krokem 5 a 6 výše.
D. Pokračujte krokem 7 výše. Všimněte si, že se automaticky přepnete do REŽIMU BOOT.
E. Pokračujte kroky 9 až 12 výše.
F. Pokud aktualizace bude stále neúspěšná, zopakujte proces s nižší přenosovou rychlostí.
G. Označte jednotku nainstalovanou verzí firmwaru.

Dodatek G

Výpočty přesnosti vzorkování

Přesnost hodnoty polohy vzorkování je možno obecně vyjádřit jako +/- 2 pulsy s přídavným kolísáním 100 mikrosekund. Skutečná přesnost hodnoty polohy vzorkování však může být lepší v závislosti na konfiguraci osy, zrychlování motoru během vzorkování a počtu pulsů na otáčku použitého snímače polohy. První otázkou je, jestli osa je nakonfigurovaná jako digitální nebo analogová.

Analogový režim

Když se v analogovém režimu vyskytne vzorkování, hodnota fázového posunutí čítače se okamžitě zachytí do přídržného registru. To znamená, že nepřesnosti zachycení polohy vycházejí především z filtrace vstupu a prodlevy vzorkování na vzorkovacím vstupu, což může dát celkem až 10 mikrosekund (nebo počet pulsů, které se mohou objevit během 10 mikrosekund). Všimněte si, že hodnota může být o jeden puls menší v závislosti na tom, jestli se vzorkování uskutečnilo v okamžiku, kdy se změnila hodnota počtu pulsů.

Digitální režim

V digitálním režimu se snímač polohy čte jako sériová data. Protože tato data se vyžadují pouze jednou za 250 mikrosekund, zachycení hodnoty polohy načtené ze snímače polohy umožňuje pouze přesnost 250 mikrosekund. Aby se obešlo toto omezení, událost vzorkování má časovou značku vztaženou k poslední načtené poloze snímače polohy, která se vyskytla v DSM314. Tato hodnota se používá k odhadu polohy osy v okamžiku, kdy se vyskytla událost vzorkování v závislosti na okamžité rychlosti servoosy v čase vzorkování. Rychlost použitá pro výpočet je odvozená z rozdílu dvou čtení snímače polohy kolem události vzorkování (viz vzorec níže).

$$\text{Rychlost} = \frac{\text{(Vzorek polohy po vzorkování) - (Vzorek polohy před vzorkováním)}}{\text{250 mikrosekund}}$$

Proto se změny rychlosti (tj. zrychlení nebo zpomalení motoru) mezi vzorky polohy nebere v úvahu a tím se způsobí nepřesnosti v zachycení hodnoty vzorkované polohy. Pro události vzorkování, které se vyskytnou, když rychlost během periody vzorkování je konstantní, algoritmus interpolace bude přesný s odchylkou jednoho pulsu a nepřesnost zachycení polohy bude primárně určena filtrací a prodlevami při vzorkování.

Následující příklad je možno použít k výpočtu nejhoršího případu nepřesnosti v důsledku zrychlení konkrétního servomotoru:

V tomto příkladu jsou dané následující hodnoty/konstanty:

Rozlišitelnost snímače polohy = 8192 pulsů/ot.

A = Zrychlení/zpomalení během události vzorkování, které je 250 000 000 pulsů/sec^2 (předpokládá se konstantní během celé periody 250μs; větší hodnoty zrychlení zvýší velikost chyby výpočtu).

T_p = Periody vzorkování polohy, která je 250 mikrosekund.

V_I = Počáteční rychlost těsně před událostí vzorkování, která v tomto příkladu je 0.

Změnu počtu pulsů snímače polohy (Cnts) za danou dobu (t) je možno vypočítat pomocí následujícího vztahu:

$$P_{act} = V_I\ t + \tfrac{1}{2} A\ t^2$$

Proto celkový počet pulsů, které se vyskytnou během vzorkovací periody pro tento příkladu, je přibližně 8 pulsů (skutečně vypočtená hodnota je 7.8125) nebo 0.343 stupňů otáčky motoru.

Průměrná rychlost pro vzorkovací periodu s danou změnou polohy bude:

$$V_{avg} = \frac{\text{Změna počtu pulsů}}{\text{Perioda vzorkování}} = \frac{7.8125 \text{ pulsů}}{250\ \mu\text{sec}} = 31250 \text{ pulsů/sec}$$

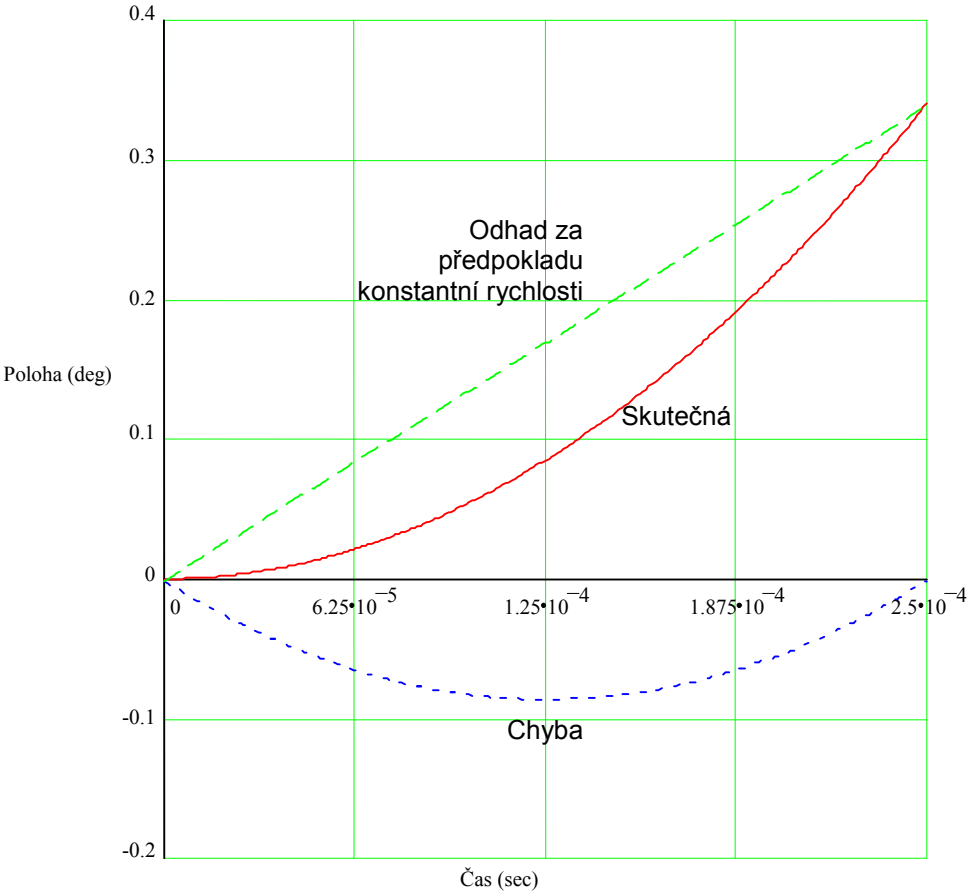
Následující vztah je možno použít k odhadu polohy vzorkování pomocí výše odvozené rychlosti:

$$P_{est} = V_I\ t + V_{avg}\ t$$

Proto chyba mezi odhadovanou polohou vzorkování a skutečnou polohou vzorkování bude:

$$\text{Chyba} = P_{act} - P_{est}$$

Níže uvedený diagram obsahuje graf skutečné polohy, odhadované polohy a výslednou chybu počtu pulsů polohy vzorkování za vzorkovací periodu 250 mikrosekund. Graf ukazuje, že největší chyba počtu pulsů se vyskytne uprostřed (tj. 125 mikrosekund) periody.



Obrázek G-1: Příklad chyby zachycení polohy osy v důsledku zrychlení

Protože počáteční rychlost se rovná 0, vztah pro výpočet P_{act} je možno změnit tak, aby určil čas, kdy se puls skutečně vyskytne při (T_{act}), následujícím způsobem:

$$T_{act} = \sqrt{\frac{2Pact}{A}}$$

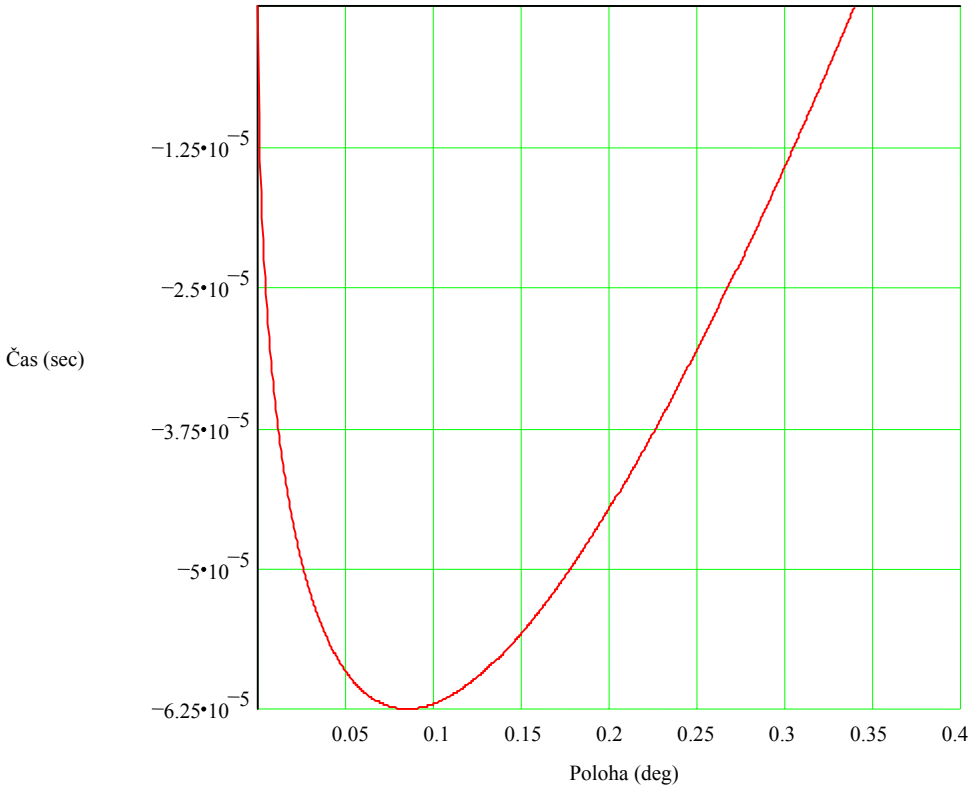
Obdobně je možno stejně řešit vztah pro odhad polohy vzorkování (P_{est}) pro čas (T_{est}) (za předpokladu, že počáteční rychlost je 0):

$$Test = Pest / Vavg$$

Pomocí těchto vztahů je možno následujícím způsobem vypočítat rozdíl času mezi tím, kdy se vyskytlo vzorkování a kdy se hlásí počet pulsů (tj. efektivní prodleva):

$$\text{Efektivní prodleva} = T_{est} - T_{act}$$

Efektivní prodleva pro maximální chybu polohy vzorkování (tj. při 125 mikrosekundách) se rovná -62,5 mikrosekund. Tato hodnota je záporná, protože odhadovaná/hlášená poloha vzorkování se vyskytla před skutečnou polohou, když nastala událost vzorkování. Následující graf představuje efektivní prodlevu, která bude vidět během změny polohy pro vzorkovací periodu v tomto příkladu.



Obrázek G-2: Efektivní prodleva doby odezvy

Proto ve výše uvedeném příkladu nejhorší případ chyby v důsledku zrychlení/zpomalení je možno vyjádřit jako +/- 0.086 stupňů (přibližně 2 pulsy) polohy nebo jako 62.5 mikrosekundy prodlevy (za předpokladu, že počáteční rychlost je 0). ***Všimněte si, že DSM nemůže pracovat se zlomky a proto se chyba zaokrouhlí na nejbližší počet pulsů nebo uživatelských jednotek.***

Vztahy pro stanovení chyby vzorkování v důsledku zrychlení/zpomalení u digitální osy jsou následující:

Počet chybných pulsů =

$$\text{Efektivní prodleva} = \frac{Tp(V_I + A\ Tp)}{4(2\ V_I + A\ Tp)}$$

kde:

A = Zrychlení/zpomalení během události vzorkování

T_p = Perioda vzorkování polohy, která je 250 mikrosekund.

V_I = Rychlost těsně před vzorkováním

Všimněte si, že výše uvedené vzorce předpokládají konstantní zrychlení během celé periody vzorkování. Vztahy pro určení chyby pro případy, kdy zrychlení během periody vzorkování není konstantní, jsou pro kontext tohoto manuálu příliš složité.

Všimněte si, že přídavná chyba až 10 mikrosekund (nebo počet stupňů nebo polohových pulsů, které se vyskytnou během 10 mikrosekund) se může také vyskytnout v důsledku filtrace vstupu/prodlev při vzorkování v hardwaru.

Výstraha

Všimněte si, že uživatelské zapojení a typ použitého zařízení pro vstup vzorkování může také způsobit nepřesnosti hodnoty vzorkování.

#

%

1

2

3

6

A

B

C

Č

D

E

F

G

H

Ch

I

J

K

L

M

N

O

P

R

Ř

S

Š

T

U

V

Z

Ž