Popis a ovládání systému DMC pro automatické řízení pohybu

Text zpracoval prof. Ing. Jaroslav Novák, CSc.

1. Výkonová část systému

U systému pro automatické řízení pohybu DMC je pro pohon osy použit synchronní motor s permanentními magnety. Jedná se o elektrický točivý stroj, který má po konstrukční stránce stejné provedení statoru, jako třífázový asynchronní motor, tj. předpokládá se napájení třífázovou soustavou napětí. Rotor synchronního motoru je opatřen permanentním magnetem, který vytváří magnetický tok. Výhody synchronního motoru pro účely zpětnovazební regulace polohy oproti motoru asynchronnímu jsou zejména vyšší dynamika a menší rozměry motoru pro daný výkon. Oproti stejnosměrným servomotorům má popisovaný motor výhodu v jednodušší stavbě a větší spolehlivosti. Synchronní servomotory s permanentními magnety se běžně používají pro výkony od desítek W do jednotek kW.

Při provozu synchronního motoru v otáčkové či polohové zpětnovazební smyčce je třeba měnit otáčky tohoto stroje. Vzhledem k tomu, že rotor se otáčí synchronně s točivým polem statoru, je nutné měnit frekvenci třífázového napájecího napětí. Zároveň, díky tomu, že je při regulaci nutné měnit i moment a tím i proud motoru, musí existovat možnost měnit i efektivní hodnotu napájecího napětí. Pro napájení střídavých motorů napětím s proměnnou frekvencí a efektivní hodnotou se nejčastěji používají tzv. nepřímé měniče frekvence. Vstupní obvod nepřímého měniče frekvence je tvořen třífázovým můstkovým diodovým usměrňovačem, který je napájen ze sítě 3x400V. Na výstupu usměrňovače je stejnosměrné napětí, avšak s nenulovou střídavou složkou. Tato střídavá složka se filtruje pomocí kondenzátoru. Tento kondenzátor tvoří tzv. stejnosměrný meziobvod. Ze stejnosměrného meziobvodu je napájen třífázový můstkový střídač. Tento obvod je tvořen šesticí tranzistorů, které pracují výhradně ve spínacím režimu, s antiparalelně řazenými tzv. zpětnými diodami. Výstupní třífázové napětí střídače je formováno pomocí šířkově-pulsní modulace, to znamená, že výstupní napětí každé fáze je tvořeno pulsy, jejichž velikost je dána velikostí napětí ve stejnosměrném meziobvodu. Sekvence spínání tranzistorů je taková, že střední hodnota (tedy potažmo první harmonická) fázového i sdruženého napětí má díky šířkově-pulsní modulaci sinusový průběh s takovou frekvencí a efektivní hodnotou, jaká je v daném okamžiku potřebná pro realizaci zpětnovazebního regulačního algoritmu. Zpětné diody jsou do obvodu řazeny kvůli vedení zpětného proudu, který je vyvolán energiemi, akumulovanými v indukčnostech obvodu.

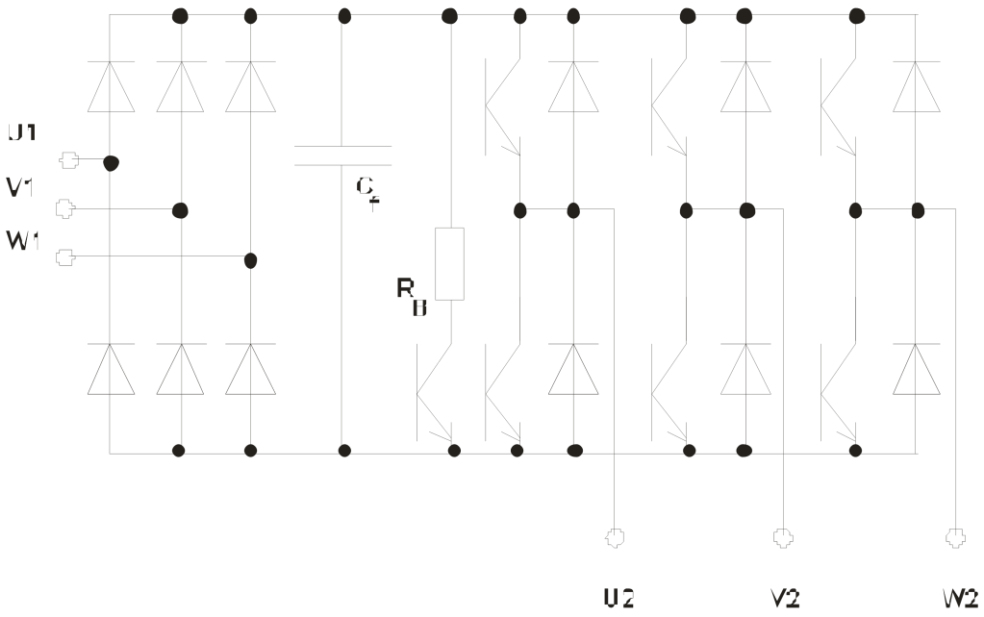
Ve stejnosměrném meziobvodu je ještě řazen paralelně k vyhlazovacímu kondenzátoru odpor spínaný tranzistorem. Je-li synchronní motor v generátorickém režimu, jedná-li se tedy o brzdění pohonu, je brzdná energie přenášena přes střídač, pracující v usměrňovačovém režimu, do stejnosměrného meziobvodu. Tato energie nemůže být dále předána do sítě, neboť ji nelze přenést přes diodový usměrňovač. Proto je tato energie mařena ve spínaném odporu ve stejnosměrném meziobvodu.

Schéma standardního zapojení silové části nepřímého měniče frekvence je na obr.1.

2. Řídící část systému DMC

Zjednodušené blokové schéma řízení a regulace systému je na obr.2.

Nejvyšší úroveň řídícího algoritmu je zabezpečována programem, který je sestaven uživatelem systému a vychází z konkrétní úlohy, kterou má pohon zabezpečit. Tento program v podstatě zadává potřebnou sekvenci pohybů pohonu v závislosti na potřebách dané aplikace. Tím, že systém plní úlohu i logického řídícího automatu, může být sled pohybů dán nejen časovou sekvencí, ale může docházet ke složitým vazbám mezi ostatními částmi technologie. Tyto vazby mohou být zprostředkovány pomocí

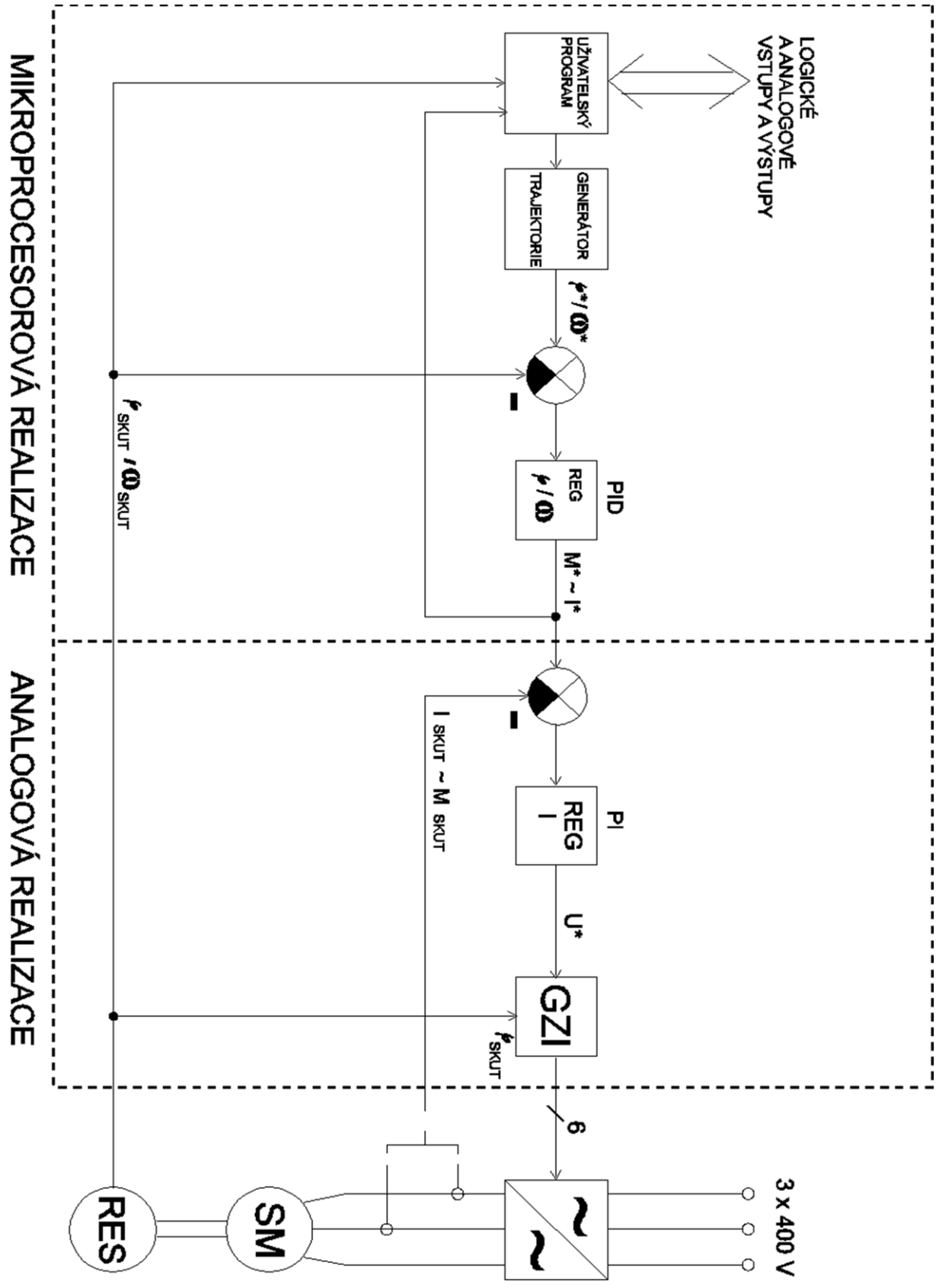


Obr. 1. Schéma zapojení silové části nepřímého měniče frekvence

logických a analogových vstupů a výstupů, prostřednictvím sériové linky nebo snímáním signálů ze snímačů otáček a polohy jiných pohonů, což umožňuje vzájemnou synchronizaci více pohonných

jednotek. Díky zadávání pracovních algoritmů pohonu pomocí programu je zde nesmírně velká variabilita úloh, které
může systém zabezpečovat. Pro realizaci úloh logického automatu lze využít i uživatelsky orientované časovače a přerušení.

Uživatelský program generuje při požadavku na pohyb pohonu (tzn. při požadavku na změnu úhlu natočení osy z jedné polohy do druhé - tzv. polohování) konečný úhel natočení osy, přičemž rovněž, prostřednictvím programově nastavitelných parametrů, definuje i tvar trajektorie pohybu tj. zejména velikost zrychlení a zpomalení (nemusí být vždy konstantní) a maximální rychlost pohybu. Tyto informace vstupují do generátoru trajektorie. Tento blok vypočítá závislost rychlostí a poloh na čase během pohybu (polohování) z výchozí do požadované cílové polohy, přičemž uvedené závislosti se uchovají ve formě tabulky s hustotou vzorkování 1 ms. Tento blok potom během pohybu zadává po 1ms žádané hodnoty rychlosti a polohy do paralelního PID regulátoru. Skutečná hodnota rychlosti a polohy se snímá resolverem, což je v podstatě elektrický stroj, který generuje dvě sinusová napětí vzájemně posunutá o 90°. Tato napětí se během periody vzorkují a v závislosti na velikosti napětí v daném okamžiku se usuzuje na úhel natočení osy. Derivace vzorkovaného napětí v daném bodě udává rychlost. Sinusová napětí, vzájemně posunutá, musí být dvě, aby se rozlišil směr otáčení. Resolvery se dodávají dvoupólové, kdy jedné otáčce osy odpovídá jedna perioda výstupní sinusovky resolveru, čtyřpólové se dvěma periodami sinusovky na otáčku osy či šestipólové se třemi periodami výstupní sinusovky na otáčku osy. Řídící mikropočítač provádí přepočet údaje o poloze osy na počet tzv. inkrementů, tj. v podstatě pulsů za periodu jedné sinusovky z resolveru. Na jednu periodu připadá 8192 inkrementů, tzn. v případě dvoupólového resolveru odpovídá tento počet počtu inkrementů za otáčku. V případě



Obr. 2 Blokové schéma řízení a regulace systému DMC

čtyřpólového resolveru připadá na otáčku 2x8192 inkrementů a v případě šestipólového resolveru připadá na otáčku 3x8192 inkrementů.
V případě popisovaného systému je resolver zabudován ve společném pouzdře se synchronním motorem.
V závislosti na vzájemné velikosti žádaných a skutečných hodnot polohy a rychlosti je PID regulátorem generována žádaná hodnota momentu. Této hodnotě je úměrná i žádaná hodnota proudu. Až do tohoto bodu jsou řídící funkce realizovány mikroprocesorem 80196. Následující část je náročná na rychlost vykonávání a je realizována analogově. Žádaná hodnota proudu je vedena přes D/A převodník a je porovnána se skutečnou hodnotou proudu, která je vyhodnocena snímači proudu. Regulační odchylka proudu se zavádí do PI regulátoru proudu, jehož výstup generuje signál úměrný požadované efektivní hodnotě výstupního napětí. Tento signál, společně se signálem o poloze rotoru z resolveru, se zavádí do bloku generátoru zapalovacích impulsů (GZI) a na základě informace o poloze rotoru a o požadované efektivní hodnotě výstupního napětí je generována šestice spínacích pulsů pro výkonové tranzistory střídače. Časové rozložení těchto pulsů realizuje šířkově-pulsní modulaci s potřebným průběhem první harmonické napájecího napětí.

Popsaná regulační struktura je velmi podobná regulačnímu schématu, které se často používá u stejnosměrných motorů s cizím buzením. Proto se někdy takovýto servopohon se synchronním motorem označuje jako pohon se stejnosměrným elektronicky komutovaným motorem. Rovněž je někdy možné se setkat s označením ventilový pohon.

3. Provedení systému DMC po stránce konstrukční a hardwarové

Systém DMC je určen pro průmyslové aplikace a tomu odpovídá jeho provedení jak po stránce mechanické, tak po stránce elektrické.

Systém sestává ze dvou celků a to napáječe a motoru. Motor má vestavěný resolver, tepelnou termistorovou ochranu, která chrání motor před tepelným přetížením tím, že vyšle signál napáječi, který motor odpojí od napájení, a dále mechanickou brzdu, která je ovládána elektromagnetem. Krytí motoru je IP54, případně IP65.

Napáječ obsahuje řídící část a výkonové obvody frekvenčního měniče. Má vestavěnou tepelnou ochranu silové části, která v případě tepelného přetížení vyvolá vypnutí měniče. Konstrukčně je napáječ koncipován pro montáž do rozvaděče.

Provedení napáječe i propojení napáječe s motorem zabezpečuje jednak odolnost zařízení proti rušení a zároveň minimalizuje rušivé vlivy vlastní silové části na jiná zařízení.

Zařízení je vybaveno deseti uživatelsky využitelnými logickými vstupy a šesti logickými výstupy. Tyto logické vstupy a výstupy pracují s úrovněmi 0 a +24V, což jsou standardní úrovně pro nasazení v průmyslovém prostředí. Dále jsou k dispozici dva analogové vstupy a dva analogové výstupy.

Pro komunikaci lze využít sériovou linku RS232 nebo RS422. Novější verze zařízení umožňují zapojení do soustavy, komunikující pomocí systému CAN.

4. Tvorba a ladění uživatelských programů

Pro tvorbu uživatelských programů se používá jazyk PL2. Programy napsané v tomto jazyce jsou lehce srozumitelné a jazyk jako takový je podobný jazyku BASIC.

Zdrojový program v jazyce PL2 se píše v integrovaném prostředí editoru, překladače a ladícího prostředí. Jedná se o aplikaci spustitelnou ve WINDOWS. Po sestavení a úspěšném překladu programu se kód programu přenese po sériové lince do řídícího systému napáječe volbou položky menu

DOWNLOAD. Po přenesení tohoto programu je možno jej spustit a pomocí komunikačního okna TERMINAL na PC řídit, monitorovat a modifikovat vykonávání algoritmu. Konkrétně, jedná-li se o výpis nebo změnu obsahu registrů programu, se při požadavku na vypsání obsahu určitého registru do terminálového okna zadá příkaz:

```
disp    PG.APos
```

kde PG.APos je název registru. Poté se na obrazovce PC vypíše obsah uvedeného registru. Při požadavku na přepsání obsahu registru se zadá příkaz ve tvaru:

```
REG.DGain,20
```

kde REG.DGain je název registru a 20 je hodnota, která se do něj má přepsat.

Při rozpojení sériové linky mezi napáječem a PC se uživatelský program vykonává dále. Při opětném připojení sériové linky je možné se systémem pomocí PC dále komunikovat a řídit běh uživatelského programu.

Uživatelský program je po přenosu z PC uložen v paměti RAM. V této paměti program zůstává uchován i při vypnutí zařízení, neboť paměť RAM je zálohována baterií. Po opětném zapnutí zařízení je automaticky spuštěn posledně přenesený uživatelský program i v případě, není-li spojena sériová linka s PC. Kód uživatelského programu může mít maximální délku 16kB.

5. Stručný popis nejzákladnějších vlastností jazyka PL2 s ohledem na programování systému DMC

Základní typy operandů jazyka PL2 jsou:

- šestnáctibitová konstanta se znaménkem v rozsahu -32768 až 32767
- dvaatřicetibitová konstanta se znaménkem v rozsahu -2147483648 až 2147483647
- dvaatřicetibitový registr - přístupný přímým (jako **R**n) nebo nepřímým (jako **R**(**R**n)) adresováním (n je číslo registru)
- šestnáctibitový registr - zpravidla se jedná o systémové proměnné, které jsou v některých případech přístupné jen pro čtení

Nejpoužívanější direktivy:

- .LIST - povolení/zakázání generování výpisu programu po překladu do souboru
  Příklad použití:
  .LIST ON
  .LIST OFF
  Přednastavená hodnota: ON
- .DEBUGINFO - povolení/zakázání generování informací pro ladění ve spustitelném souboru
  Příklad použití:
  .DEBUGINFO ON
  .DEBUGINFO OFF
  Přednastavená hodnota: ON
- .LISTMACRO - povolení/zakázání generování výpisu makra ve výpisu programu po

překladu

Příklad použití:
.LISTMACRO ON
.LISTMACRO OFF
Přednastavená hodnota: ON

- .DEFINE - definuje jméno konstanty či registru (obdobně jako v jazyce C)
  Příklad použití:
  .DEFINE dvacitka = 20
  .DEFINE pomreg = *registr*
- .REGISTER - deklarace uživatelského registru
  Příklad použití:
  .REGISTER *název proměnné*
- .INCLUDE - připojení souboru
  Příklad použití:
  .INCLUDE *jméno souboru*

Řádkové příkazy použitelné při komunikaci napáječe s PC:

LIST *(řádka 1(,řádka 2))* - výpis programu, popř. při uvedení jednoho čísla řádky výpis jedné řádky, při uvedení dvou čísel řádky výpis od řádky 1 do řádky 2

HLIST - stejná funkce jako LIST, pouze se před řádku vypíše kód HEX příslušného řádku

DEL *řádka 1(,řádka 2)* - vymaže uvedenou řádku, v případě uvedení čísel dvou řádek vymaže program od řádky 1 do řádky 2

RUN *(řádka)* - spustí program od instrukce na uvedené řádce

NEW - vymazání kódu uživatelského programu v paměti programu RAM

CONT *(řádka)* - pokračování ve vykonávání programu od určité řádky po jeho předchozím zastavení

Operátory v jazyce PL2:

Jazyk PL2 pracuje se všemi základními operátory. Operandem je buď registr nebo konstanta. Jeden z operandů musí být zpravidla registr. Nejčastější zápis výrazů je ve tvaru:

Operand 1 Operátor *Operand 2*

Aritmetické operátory jsou:
+, – , *, / ,MOD (zbytek po celočíselném dělení)
Logické operátory jsou:
AND, OR, EXOR

Operátory pro operace posunu:
>> (binární posun operandu 1 vpravo - počet posunů udává operand 2), << (binární posun operandu 1 vlevo - počet posunů udává operand 2)

Relační operátory:
<, < =, < >, =, >, > =

Výběr nejpoužívanějších příkazů jazyka PL2:

NOP - prázdná operace

STOP - zastavení vykonávání uživatelského programu. Program může být opětně spuštěn pomocí řádkového příkazu CONT.

END - konec programu. Hlášení o ukončení programu je zobrazeno na obrazovce.

DISP *registr* - zobrazení obsahu registru na obrazovce

GOTO *návěští* - příkaz nepodmíněného skoku

GOSUB *návěští* - příkaz volání podprogramu

RETURN - návrat z podprogramu

IRETURN *proměnná* - návrat z obslužného podprogramu přerušení. *Proměnná* označuje typ přerušení a její uvedení zabezpečí opětné povolení tohoto přerušení po návratu z jeho obslužného programu.

Příklad použití:

```
IRETURN    IN1        ; návrat z externího přerušení, jehož
                      ; signál je čten z logického vstupu
                  ; č. 1 (IN1)
```

LOOP *registr, řádka* - je-li obsah registru větší než nula, jeho obsah se dekrementuje a provede se skok na uvedenou řádku. V opačném případě program pokračuje na další řádce.

IF *výraz* THEN *řádka* - příkaz podmíněného skoku

LET *registr* = *výraz* - provedení operace ve výrazu s přiřazením

ADD *registr, registr* nebo *konstanta* - sčítání

SUB *registr, registr* nebo *konstanta* - odčítání

ABS *registr, registr* nebo *konstanta* - jestliže je druhý operand kladný, potom se číslo v prvním registru převede do absolutní hodnoty. V opačném případě se před absolutní hodnotu čísla v prvním registru přiřadí znaménko mínus.

CLR *registr* - vynulování registru

AND *registr, registr* nebo *konstanta* - logický součin bit po bitu

OR *registr, registr* nebo *konstanta* - logický součet bit po bitu

PUSH *registr* nebo *konstanta* - uložení hodnoty do zásobníku

POP *registr* - vyzvednutí hodnoty ze zásobníku

CONNECT *analog. vstup či výstup* TO *registr* - přenos hodnoty analog. vstupu/výstupu do/z registru

Příklad použití:

```
CONNECT   ANA.IN1  TO  R20
```

Hodnota výsledku A/D převodu z analogového vstupu 1 se zapíše do registru R20

WAIT *výraz* - program pokračuje za tímto příkazem až po dosažení nenulové hodnoty výrazu - ve výrazu figuruje systémový registr

Komentáře se v jazyce PL2 oddělují středníkem.

Poznámka:

V jazyce PL2 se u aritmetických a logických operací a u příkazů pro přesun dat může používat jak zápis se syntaxí obdobnou syntaxi ve vyšších jazycích (např. R2=R2+5 či R4 = 6), tak i zápis připomínající formu assembleru (ADD R2,5 či R4,6).

Systémové registry související s řízením pohybu a s funkcemi logického automatu:
Veškeré systémové registry jsou uspořádány do několika skupin:
- skupiny RD1 a RD2 - registry vztahující se k resolverům
- skupina PG - registry související se zadáváním trajektorie pohybu
- skupina STACK - zásobník
- skupina MOTOR - registry pro nastavení parametrů motorů
- skupina REG - registry pro nastavení parametrů regulátoru
- skupina GEAR - registry pro realizaci mechanického převodu elektrickou vazbou
- skupina TMR - registry pro řízení časovačů
- skupina SYSIO - systémové registry mikropočítačového řídícího systému
- skupina INT - registry pro řízení přerušení
- skupina IN - logické vstupy
- skupina OUT - logické výstupy
- skupina VECTOR - vektory přerušení
- skupina ANA - registry související s analogovými vstupy a výstupy
- skupina COMM - registry pro řízení sériové komunikace

První část jména konkrétního systémového registru je stejná pro všechny registry téže skupiny, druhá část je specifická pro každý registr. Dvě části jména registru jsou odděleny tečkou. Dále bude uveden přehled základních registrů z několika skupin.

Skupina RD1:
RD1.Pos - skutečná okamžitá poloha osy v inkrementech určená pomocí resolveru
RD1.Speed - skutečná okamžitá rychlost osy v inkrementech za sekundu určená pomocí resolveru

Skupina PG:
PG.Acc - nastavení zrychlení v inkrementech za sekundu na druhou
PG.APos - okamžitá žádaná poloha v inkrementech
PG.ASpeed - okamžitá žádaná rychlost v inkrementech za sekundu
PG.Speed - žádaná rychlost při polohování v inkrementech za sekundu
PG.PosSpeed - maximální přípustná rychlost při polohování
PG.Decel - nastavení zpomalení v inkrementech za sekundu na druhou
PG.ADecel - okamžité žádané zpomalení v inkrementech za sekundu na druhou
PG.DPos - registr, do kterého se zadává cílová poloha v inkrementech při požadavku na zahájení polohování
PG.Rdy - registr příznaku konce polohování - není-li ukončeno polohování, obsah registru je 0
PG.Mode - při nastaveni hodnoty 1 se povolí spuštění polohování

Skupina REG:
REG.PGain - proporcionální konstanta regulátoru úhlu natočení

REG.IGain - integrační konstanta regulátoru úhlu natočení
REG.DGain - derivační konstanta regulátoru úhlu natočení
REG.PosErr - okamžitý rozdíl mezi žádanou a skutečnou polohou tj. regulační odchylka úhlu
REG.PosIErr - integrovaný rozdíl mezi žádanou a skutečnou polohou tj. integrál regulační odchylky úhlu
REG.SpdErr - okamžitý rozdíl mezi žádanou a skutečnou rychlostí tj. regulační odchylka rychlosti
REG.Torque - žádaná hodnota momentu (tj. proudu) - jmenovitému momentu odpovídá hodnota 8191

Skupina TMR:

TMR.ABs - absolutní čas od spuštění zařízení v milisekundách - obsah tohoto dvaatřicetibitového registru lze pouze číst
TMR.CycInt - šestnáctibitový registr pro nastavení frekvence generování periodického přerušení od časovače - pakliže je přerušení povoleno, je generováno s periodou danou obsahem tohoto registru v ms
TMR.T0 - obsah časovače 0 - jeho obsah je zvětšován současně s hodnotou absolutního času. Při změně obsahu tohoto dvaatřicetibitového registru se automaticky změní i obsah registru TMR.A0.
TMR.A0 - obsah tohoto dvaatřicetibitového registru udává posun časového údaje časovače T0 oproti absolutnímu času systému
TMR.T1, TMR.T2, TMR.T3 - časovače T1, T2, T3 -význam registrů stejný jako u časovače 0
TMR.A1, TMR.A2, TMR.A3 - analogie registru TMR.A0

Skupina INT:

INT.Pend - příznaky žádosti o externí přerušení - nastavené bity 0 až 9 tohoto registru odpovídají žádostem o externí přerušení signálem přivedeným přes logické vstupy 1 až 10
INT.Mask - povolení externího přerušení - nastavení bitů 0 až 9 tohoto registru povoluje externí přerušení přivedená přes logické vstupy 1 až 10
INT.Level - určení aktivní hrany externího přerušení - bity 0 až 9 odpovídají externím přerušením od logických vstupů 1 až 10, přičemž nastavený bit znamená přerušení od vzestupné hrany, nulovaný bit přerušení od sestupné hrany
INT.SysPend - nastavený bit 0 je příznakem žádosti o přerušení od časovače generujícího přerušení periodicky se zadanou frekvencí
INT.SysMask - nastavením bitu 0 se povolí periodické přerušení od časovače

Skupina IN:

IN.Di1 až IN.Di10 - obsahují logickou hodnotu na logických vstupech 0 až 10

Skupina OUT:

OUT.Do1 až OUT.Do6 - provádí se zápis logických hodnot vysílaných na logické výstupy 0 až 6

Skupina VECTOR

VECTOR.CycInt - do tohoto registru se zadá adresa (návěští) obslužného podprogramu přerušení cyklicky generovaného od časovače
VECTOR.Di1 až VECTOR.Di10 - do těchtoto registrů se zadají adresy (návěští) obslužných podprogramů externích přerušení od logických vstupů 0 až 10

Skupina ANA:

Výsledky A/D převodu je možno reprezentovat v potřebném rozsahu. Přepočet do tohoto rozsahu se provede automaticky na základě zadaných parametrů podle vztahu:

výsledek = ((výstup A/D převodníku)/16384) ∗ ANA.InRange + ANA.InOffset

U čísla, vysílaného na D/A převodník rovněž dojde k automatickému přepočtu na základě zadaných parametrů podle vztahu:

$$\text{výstup} = \text{ANA.Out} * 2^{\text{ANA.Outsf}} + \text{ANA.OutOffs}$$

ANA.In1, ANA.In2 - hodnoty úměrné napětím na analogových vstupech 1 a 2
ANA.In1Range, ANA.In2Range - parametry pro přepočet výsledků A/D převodu
ANA.In1Offs, ANA.In1Offs - parametry pro přepočet výsledků A/D převodu
ANA.Out1, ANA.Out2 - vstupní hodnoty pro D/A převodník
ANA.Out1sf, ANA.Out2sf - parametry pro přepočet vstupní hodnoty D/A převodu
ANA.Out1Offs, ANA.Out2Offs - parametry pro přepočet vstupní hodnoty D/A převodu
ANA.ConnTMR - do tohoto registru se zadává čas v ms mezi dvěma přístupy k A/D či D/A převodníku

Na závěr kapitoly o systému DMC je pro ilustraci uveden příklad uživatelského programu v jazyce PL2. Algoritmus programu je následující: Servopohon nejprve po zvolené trajektorii vyhledá nulovou polohu. Potom se po aktivaci logického vstupu DI1 otočí o počet otáček daný obsahem registru r20. V okamžiku dosažení žádané polohy nastaví logický výstup DI2.

```
;-------------------- Program pro relativni polohovani ----------------------------------

        .include “Makra.pl2“                ;vlozeni souboru s makry
        .include “MotRes.pl2“               ;vlozeni souboru s programem pro nastaveni
                                    ;parametru motoru a resolveru
        .include “SetReg.pl2“               ;vlozeni souboru s programem pro nastaveni
                                    ;parametru regulatoru


        .register       Poloha              ;deklarace registru
        .register       Posun               ;deklarace registru

        Vector.DI1,@POL                     ;adresa obsluzneho programu externiho preruseni
                                    ;od logickeho vstupu DI1
        Int.Level = Int.Level or 1          ;nastavení externiho preruseni od vzestupne
                            ;hrany
;------------------- Nastaveni nulove polohy ---------------------------------------------------
```

```
        Pg.Acc,200000                   ;zadani velikosti zrychleni
        Pg.Decel,100000                 ;zadani velikosti zpomaleni
        Pg.PosSpeed,1000                ;maximalni rychlost pri polohovani
        Pg.Dpos,0                       ;zadani zadane polohy 0
        Pg.Mode,1                       ;povoleni spusteni polohovani
        Pg.Speed = Pg.PosSpeed          ;nastaveni maximalni rychlosti pri polohovani
        wait    Pg.Rdy              ;cekani na konec polohovani
        Dwell  100,rDelay               ;pauza 100ms
        disp    Rd1.Pos                 ;vypis skutecne hodnoty uhlu na obrazovce PC
        Out.DO1 = 1                     ;hlaseni pripravenosti serva nastavenim
                                ;logickeho vystupu DO1
        Int.Mask = Int.Mask or 1        ;povoleni externiho preruseni od logickeho
                                ;vstupu DI1
        r20,0                           ;nulovani registru r20

;------------------------------ Hlavni cekaci smycka ----------------------------------------------------

SMYCKA:
        Poloha = r20 * 8192             ;poloha = r20 otacek motoru
        Ana.ConnTmr,1                   ;nastaveni casu mezi obsluhou D/A a A/D kanalu
        connect  Ana.Out1 To Reg.Torque ;vyslani signalu umerneho momentu motoru na
                                ;D/A vystup
        goto    SMYCKA                  ;skok na zacatek cyklu

;---------------------------- Obsluzny podprogram externiho preruseni od DI1 ---------------------

POL:
        Out.DO2 = 0                     ;nulovani vystupu DO2
        Posun = Rd1.Pos                 ;zapamatovani vychozi polohy
        Pg.Acc,50000                    ;zadani velikosti zrychleni
        Pg.Decel,50000                  ;zadani velikosti zpomaleni
        Pg.PosSpeed,8192                ;maximalni rychlost pri polohovani
        Pg.Dpos,Rd1.Pos + Poloha        ;zadani cilove polohy
        Pg.Mode,1                       ;povoleni spusteni polohovani
        Pg.Speed = Pg.PosSpeed          ;nastaveni maximalni rychlosti pri polohovani
CEKAT:
        Ana.ConnTmr,1                   ;nastaveni casu mezi obsluhou D/A a A/D kanalu
        connect  Ana.Out1 To Reg.Torque ;vyslani signalu umerneho momentu motoru na
                                ;D/A vystup
        if  Pg.Rdy = 0 then CEKAT       ;testovani ukonceni polohovani
        Dwell  500,rDelay               ;pauza 500ms
        Posun = Rd1.Pos - Posun         ;vypocet skutecne velikosti posunu
        disp    Posun                   ;zobrazeni velikosti skutecneho posunu na PC
        Out.DO2 = 1                     ;signalizace ukonceni polohovani nastavenim DO2
        Int.Pend = 0                    ;zruseni zadosti o preruseni
        ireturn IN1                     ;navrat z obsluzneho podprogramu preruseni
```