VISOKA POSLOVNA ŠKOLA
STRUKOVNIH STUDIJA
ČAČAK

DIPLOMSKI RAD

Izrada hardvera i softvera laboratorijskog radnog mesta za ispitivanje vektorske regulacije asinhronog motora

Mentor:_____ Student: _____
Profesor: _____ Br.Indeksa: _____

Beograd, oktobar 1995.

SADRŽAJ

2. MATEMATIÈKI MODEL VEKTORSKI UPRAVLJANOG ASINHRONOG MOTORA

U ovom poglavlju dat je matematièki model asinhronog motora, èije se upravljanje zahteva i izložene su teoretske osnove indirektne vektorske kontrole, kojom je zahtevano upravljanje realizovano.

2.1. Model asinhronog motora

Prilikom modelovanja asinhronog motora polazne pretpostavke koje se usvajaju su postojanje raspodeljenog trofaznog namota na rotoru i statoru, rotaciona simetrija mašine, sinusoidalna promena magnotopobudne sile u zazoru (odsustvo prostornih harmonika), zanemarenje iviènih efekata, zanemarenje gubitaka u magnetiku usled histerezisa i vihornih struja i zanemarenje efekta potiskivanja struje. Trofazni namoti statora asinhronog motora se priključuju na trofazni, prostoperiodièni naponski izvor, usled èega kroz njih protièu prostoperiodične struje statora. Otpornosti statorskih namotaja su međusobno jednaki kod simetriènih mašina ($R_a=R_b=R_c=R_s$). Jednaèine naponske ravnoteže za stator su:

$$\begin{aligned} u_a &= R_s i_a + \frac{d\Psi_a}{dt} \\ u_b &= R_s i_b + \frac{d\Psi_b}{dt} \\ u_c &= R_s i_c + \frac{d\Psi_c}{dt} \end{aligned} \qquad (2.1)$$

Namoti rotora su kratko spojeni i u njima dolazi do indukovanja rotorskih struja. Takođe važi jednakost između otpornosti rotorskih namota ($R_A=R_B=R_C=R_r$). Za jednaèine naponske ravnoteže rotora možemo pisati:

$$\begin{aligned} 0 &= R_r i_A + \frac{d\Psi_A}{dt} \\ 0 &= R_r i_B + \frac{d\Psi_B}{dt} \\ 0 &= R_r i_C + \frac{d\Psi_C}{dt} \end{aligned} \qquad (2.2)$$

Fluksni obuhvati statorskih i rotorskih namotaja su dati u matriènoj formi:

$$\begin{bmatrix} \Psi_a \\ \Psi_b \\ \Psi_c \\ \Psi_A \\ \Psi_B \\ \Psi_C \end{bmatrix} = \begin{bmatrix} L_a & M_{ab} & M_{ac} & M_{aA} & M_{aB} & M_{aC} \\ M_{ab} & L_b & M_{bc} & M_{bA} & M_{bB} & M_{bC} \\ M_{ac} & M_{bc} & L_c & M_{cA} & M_{cB} & M_{cC} \\ M_{aA} & M_{bA} & M_{cA} & L_A & M_{AB} & M_{AC} \\ M_{aB} & M_{bB} & M_{cB} & M_{AB} & L_B & M_{BC} \\ M_{aC} & M_{bC} & M_{cC} & M_{AC} & M_{BC} & L_C \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \\ i_A \\ i_B \\ i_C \end{bmatrix} \qquad (2.3)$$

gde su: $L_a=L_b=L_c=L_{ss}$ - sopstvene induktivnosti statorskih namota
$M_{ab}=M_{ac}=M_{bc}=M_{ss}$ - međusobne induktivnosti statorskih namota
$L_A=L_B=L_C=L_{rr}$ - sopstvene induktivnosti rotorskih namota

$M_{AB}=M_{AC}=M_{BC}=M_{rr}$ - meðusobne induktivnosti rotorskih namota

$M_{aA}, M_{aB}, M_{aC}, M_{bA}, M_{bB}, M_{bC}, M_{cA}, M_{cB}, M_{cC}$ - meðusobne induktivnosti statorskih i rotorskih namota

Meðusobne induktivnosti M_{xY} u matrici induktivnosti su promenljive i zavise od relativnog položaja statora i rotora. Konstantne vrednosti ovih induktivnosti se mogu dobiti magnetnim rasprezanjem namota statora i rotora, tj. transformacijom položaja namota tako da se nalaze pod uglom od 90^0. Ovim trofazni sistem (a,b,c) svodimo na dvofazni (α,β) sistem, gde je ovaj novi $(\alpha-\beta)$ koordinatni sistem osa vezan za stator. Matrièni prikaz transformacije rasprezanja je:

$$\begin{bmatrix} i_{\alpha s} \\ i_{\beta s} \end{bmatrix} = \frac{2}{3}\begin{bmatrix} 1 & \cos\frac{2\pi}{3} & \cos\frac{4\pi}{3} \\ 0 & \sin\frac{2\pi}{3} & \sin\frac{4\pi}{3} \end{bmatrix}\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = [C]_s[i]_s$$

$$\begin{bmatrix} u_{\alpha s} \\ u_{\beta s} \end{bmatrix} = [C]_s[u]_s \text{èèèèèèèèèèè} \begin{bmatrix} \Psi_{\alpha s} \\ \Psi_{\beta s} \end{bmatrix} = [C]_s[\Psi]_s$$

$$\begin{bmatrix} i_{\alpha r} \\ i_{\beta r} \end{bmatrix} = \frac{2}{3}\begin{bmatrix} \cos(\theta_r) & \cos(\theta_r+\frac{2\pi}{3}) & \cos(\theta_r+\frac{4\pi}{3}) \\ \sin(\theta_r) & \sin(\theta_r+\frac{2\pi}{3}) & \sin(\theta_r+\frac{4\pi}{3}) \end{bmatrix}\begin{bmatrix} i_A \\ i_B \\ i_C \end{bmatrix} = [C]_r[i]_r$$

$$\theta_r = \theta_r(0) + \int_0^t \omega_r dt$$

$$\begin{bmatrix} u_{\alpha r} \\ u_{\beta r} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \text{èèèèèèèèèèè} \begin{bmatrix} \Psi_{\alpha r} \\ \Psi_{\beta r} \end{bmatrix} = [C]_r[\Psi]_r \qquad (2.4)$$

Jednaèine elektriènog podsistema u $\alpha-\beta$ sistemu koordinata sada dobijaju oblik:

$$u_{\alpha s} = R_s i_{\alpha s} + \frac{d\Psi_{\alpha s}}{dt}$$

$$u_{\beta s} = R_s i_{\beta s} + \frac{d\Psi_{\beta s}}{dt}$$

$$0 = R_r i_{\alpha r} + \frac{d\Psi_{\alpha r}}{dt} + \omega_r \Psi_{\beta r} \qquad (2.5)$$

$$0 = R_r i_{\beta r} + \frac{d\Psi_{\beta r}}{dt} - \omega_r \Psi_{\alpha r}$$

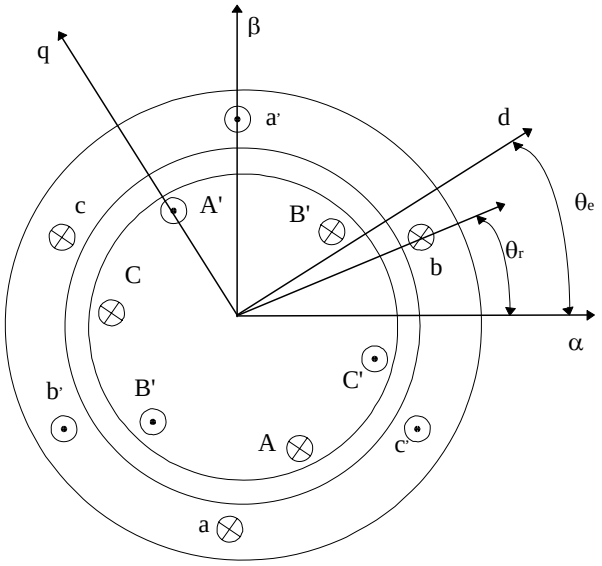
$$\omega_r = \frac{d\theta_r}{dt}.$$

Ugao θ_r je ugao koji zaklapaju osa faze *a* statora, uzeta za α-osu koordinatnog sistema, i osa faze *A* rotora i prikazan je na slici 2.1. Veza flukseva i struja je:

$$\begin{aligned}\Psi_{\alpha s} &= L_s i_{\alpha s} + M i_{\alpha r}\\ \Psi_{\beta s} &= L_s i_{\beta s} + M i_{\beta r}\\ \Psi_{\alpha r} &= L_r i_{\alpha r} + M i_{\alpha s}\\ \Psi_{\beta_r} &= L_r i_{\beta r} + M i_{\beta s r}\end{aligned} \qquad (2.6)$$

gde su: L_s - induktivnost statorskog namota,
M - meðusobna induktivnost statora i rotora i
L_r - induktivnost rotorskog namota.

Velièine u $\alpha - \beta$ stacionarnom koordinatnom sistemu su promenljive i u tranzijentnom i u stacionarnom režimu rada. Pri konstantnoj brzini i optereæenju ove velièine su sinusoidalne i uèestanost im je jednaka ugaonoj brzini obrtnog polja (ω_e).



Sl. 2.1. Ilustracija rotacione transformacije

Ukoliko bi promenljive iz stacionarnog α–β sistema transformisali u d-q koordinatni sistem koji rotira sinhrono sa obrtnim poljem, vrednosti napona, struje i fluksa bi u stacinarnom stanju postale konstantne. Položaj d-q koordinatnog sistema u odnosu na α–β koordinatni sistem određen je uglom θ_e (slika 2.1.). Transformacija rotacije vektora napona $[u]_{\alpha\beta}$ u vektor $[u]_{dq}$ je:

$$[u]_{dq} = \begin{bmatrix} \cos\theta_e & \sin\theta_e \\ -\sin\theta_e & \cos\theta_e \end{bmatrix} [u]_{\alpha\beta}$$

$$\theta_e = \theta_e(0) + \int_0^t \omega_e dt \qquad (2.7)$$

Jednaèine elektriènog podsistema u d-q koordinatnom sistemu imaju izgled:

$$
\begin{aligned}
u_d &= R_s i_d + \frac{d\Psi_d}{dt} - \omega_e \Psi_q \\
u_q &= R_s i_q + \frac{d\Psi_q}{dt} + \omega_e \Psi_d \\
0 &= R_r i_D + \frac{d\Psi_D}{dt} - (\omega_e - \omega_r)\Psi_Q \\
0 &= R_r i_Q + \frac{d\Psi_Q}{dt} + (\omega_e - \omega_r)\Psi_D
\end{aligned}
\qquad (2.8)
$$

Indeksi *d* i *q* oznaèavaju promenljive statora , dok se *D* i *Q* indeksi nalaze uz promenljive rotora. Uz pretpostavku da su koeficijenti induktivnosti konstantni, veze izmeðu flukseva i struja su:

$$
\begin{aligned}
\Psi_d &= L_s i_d + M i_D \\
\Psi_q &= L_s i_q + M i_Q \\
\Psi_D &= L_r i_D + M i_d \\
\Psi_Q &= L_r i_Q + M i_q
\end{aligned}
\qquad (2.9)
$$

Elektromagnetni moment se može odrediti kao vektorski proizvod vektora struje i vektora fluksa statorskog namotaja. Tako određen moment, izražen preko transformisanih vrednosti fluksa i struje u d-q koordinatnom sistemu, dobija oblik:

$$T_e = \Psi_d i_q - \Psi_q i_d \qquad (2.10)$$

Korekcija izraza elektromagnetnog momenta se vrši jer transformacije nisu invarijantne po snazi, a i da bi važio za mašine sa proizvoljnim brojem pari polova (p - broj pari polova):

$$T_e = \frac{3}{2} \cdot \frac{p}{2} (\Psi_d i_q - \Psi_q i_d) \qquad (2.11)$$

Jednaèina koja kompletira model asinhronog motora opisuje mehanièki podsistem:

$$J \frac{d\omega_r}{dt} + K_t \omega_r = T_e - T_l \qquad (2.12)$$

gde je: J - moment inercije,
K_t - koeficijent trenja i
T_l - moment optereæenja.

2.2. Indirektna vektorska kontrola

Kontrolom amplitude vektora statorske struje i njegove orjentacije u odnosu na vektor rotorskog fluksa se ostvaruje raspregnuto upravljanje fluksom i momentom asinhronog motora. Za ovakvo upravljanje potrebno je poznavati orjentaciju vektora rotorskog fluksa. Ugaoni pomeraj θ_e (slika 2.1.) između vektora rotorskog fluksa i ose α stacionarnog $\alpha-\beta$ koordinatnog sistema izraèunava se u matematièkom modelu rotorskog kola. Ulazne velièine u model rotorskog kola su struje statora i ugaona brzina rotora. Strujno regulisani naponski invertor, iz koga se napaja motor, omoguæava manipulaciju strujnim vektorom nezavisno od optereæenja i brzine. Uz njegovu pomoæ u motor se injektuje statorska struja određene amplitude i pomeraja u odnosu na vektor fluksa. Komponenta ove struje paralelna sa vektorom rotorskog fluksa određuje amplitudu fluksa, dok komponenta normalna na fluks daje moment. Transformacijom promenljivih na d-q rotacioni sistem koordinata odabran tako da se d-osa poklapa sa vektorom rotorskog fluksa, dobija se:

$$\vec{\Psi}_R = \vec{d}\cdot\Psi_D + \vec{q}\cdot 0 \Rightarrow \Psi_Q = 0 \qquad (2.13)$$

Ukoliko struje rotora i fluks statora izrazimo preko struja statora i fluksa rotora:

$$\begin{aligned}
\Psi_Q &= L_r i_Q + M i_q \Rightarrow i_Q = -\frac{M}{L_r} i_q \\
\Psi_D &= L_r i_D + M i_d \Rightarrow i_D = -\frac{\Psi_D - M i_d}{L_r} \\
\Psi_q &= L_s i_q + M i_Q = (L_s - \frac{M^2}{L_r}) i_q = L_\sigma i_q \Rightarrow L_\sigma = L_s - \frac{M^2}{L_r} \\
\Psi_d &= L_s i_d + M i_D = \frac{M}{L_r}(\frac{L_s L_r}{M} i_d + L_r i_D + M i_d - M i_d) = \frac{M}{L_r}\Psi_d + L_\sigma i_d
\end{aligned} \qquad (2.14)$$

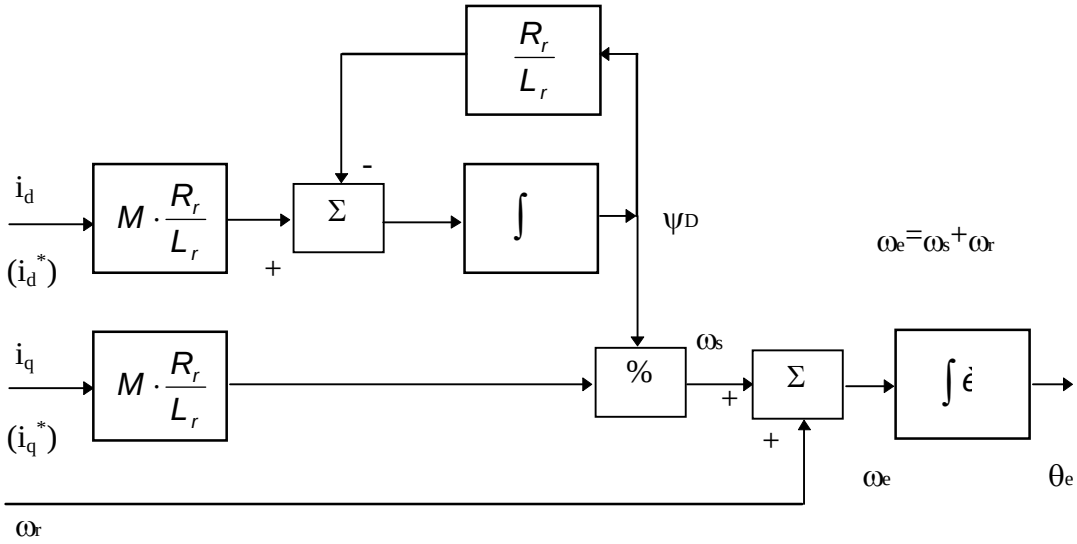
Jednaèine elektriènog podsistema su:

$$\begin{aligned}
u_d &= R_s i_d + L_\sigma \frac{di_d}{dt} + \frac{M}{L_r}\frac{d\Psi_D}{dt} - \omega_e L_\sigma i_q \\
u_q &= R_s i_q + L_\sigma \frac{di_q}{dt} + \omega_e \frac{M}{L_r}\Psi_D + \omega_e L_\sigma i_d \\
u_D &= \frac{R_r}{L_r}\Psi_D + \frac{d\Psi_D}{dt} - R_r \frac{M}{L_r} i_d = 0 \\
u_Q &= -R_r \frac{M}{L_r} i_d + (\omega_e - \omega_r)\Psi_D = 0
\end{aligned} \qquad (2.15)$$

Izraz za elektromagnetni moment sada postaje:

$$T_e = \frac{3}{2}\frac{p}{2}(\Psi_d i_q - \Psi_q i_d) = \frac{3}{2}\frac{p}{2}\Psi_D i_q \qquad (2.16)$$

Sada imamo raspregnuto upravljanje fluksom i momentom, èime se vektorski kontrolisani asinhroni motor izjednaèava sa motorom jednosmerne struje u pogledu regulacionih karakteristika. Jednaèine d-ose asinhronog motora analogne su jednaèinama pobudnog namota motora jednosmerne struje, dok struje q-ose odgovaraju struji armature. Ovo je omoguæeno poznavanjem položaja vektora rotorskog fluksa, što se izraèunava u modelu rotorskog kola. Na osnovu prethodno navedenih jednaèina rotorskog kola, moguæe je odrediti blok dijagram modela rotorskog kola u d-q koordinatnom sistemu, dat na slici 2.2.



Sl. 2.2. Model rotorskog kola u sinhronom d-q koordinatnom sistemu sa ilustracijom indirektnog određivanja orjentacije rotorskog fluksa θ_e

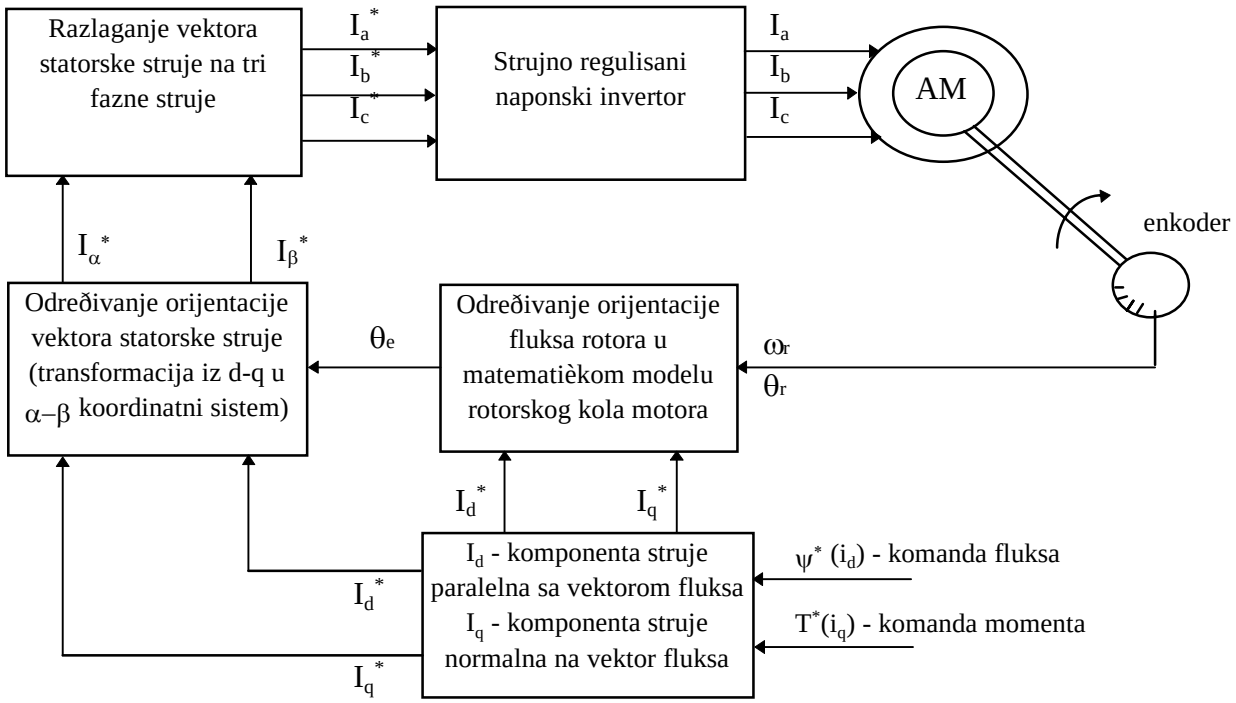
Neophodni parametri rotorskog kola za izraèunavanje vektora rotorskog fluksa su induktivnost magnetizacije i vremenska konstanta rotora, odnosno karakteristika magneæenja mašine i vrednost otpornosti rotorskog namota. Struje i_d i i_q se određuju na osnovu merenja faznih struja motora, što je moguæe izbeæi u sluèaju napajanja iz strujno regulisanog naponskog invertora sa karakteristikama bliskim idealnim. Tada se razlike između komandovanih i aktuelnih vrednosti struja mogu zanemariti i kao ulazne veliène uzeti komandovane vrednosti struja i_d^* i i_q^*.

Na osnovu svega do sada izloženog, u moguænosti smo da nacrtamo principijelnu šemu indirektnog vektorskog upravljanja, što je i uèinjeno na slici 2.3.

Ulogu povratne sprege po stanju u vektorskom kontroleru ima orjentacija rotorskog fluksa θ_e izraèunata u modelu rotorskog kola. Od taènosti modela rotorskog kola, tj. odstupanja parametara modela od parametara motora, zavisiæe i performanse ovakvog pogona. Do odstupanja može doæi usled efekta potiskivanja struje u provodnicima rotora ka periferiji na višim uèestanostima, zasiæenja magnetnog kola glavnog fluksa i zasiæenja na putu rasipnog fluksa i temperaturnih promena otpornosti rotorskih namotaja.

U konkretnom sluèaju se radi o standardnom niskonaponskom èetvoropolnom asinhronom motoru snage 750 W. Kontrola brzine ovog motora se vrši do njegove

nominalne vrednosti i nije predviđen za rad u oblasti slabljenja polja, niti se amplituda fluksa prilagođava optereæenju radi minimizacije gubitaka. Amplituda rotorskog fluksa se održava konstantnom, zadavanjem konstantne vrednosti struje magnetizacije i_d. U tom sluèaju se induktivnost L_m ($L_m = \Psi_m / i_m$) ne menja, jer je nivo magnetnog kola konstantan.



Sl. 2.3. Principijelna šema indirektnog vektorskog upravljanja.

Kao što smo videli, ugao θ_e se određuje integracijom sume ugaonih brzina rotora (ω_r) i klizanja ($\omega_s = \omega_e - \omega_r$). Zahtev za korektnim određivanjem orjentacije fluksa je ekvivalentan zahtevu za korektnim izraèunavanjem klizanja ω_s koordinatnog d-q sistema u odnosu na rotor. Kako vektorski kontroler zadaje konstantnu vrednost fluksa ψ_{D0}, određenu konstantnom vrednošæu struje i_{d0}, jednaèine rotorskog kola mašine postaju:

$$\frac{1}{T_r}\Psi_{d0} = \frac{1}{T_r} M i_{d0} \quad (M = \frac{\Psi_m}{i_m},\ T_r = \frac{L_r}{R_r})$$
$$\omega_s \Psi_{D0} = \frac{1}{T_r} M i_q \quad (\Psi_{q0} = 0,\ \Psi_{d0} = const). \qquad (2.17)$$

Na osnovu datih jednaèina, klizanje raèunamo kao:

$$\omega_s = \frac{1}{T_r} \cdot \frac{i_q}{i_{d0}} \qquad (2.18)$$

Nepoznanicu nam predstavlja vremenska konstanta rotora T_r, koju određujemo iz poznatih vrednosti struja i klizanja za nominalni radni režim i dobijenu vrednost uvrštavamo u model rotorskog kola.

Kao zaključak možemo reæi da na osnovu poznate pozicije rotora dobijene preko inkrementalnog davaèa i klizanja izraèunatog preko komandovanih vrednosti struja u modelu rotorskog kola, dobijamo položaj vektora rotorskog fluksa θ_e koji sa strujama i_{do} (konstantna komponenta struje paralelna sa vektorom fluksa, kojom se kontroliše fluks) i

i_q (komponenta normalna na vektor fluksa za kontrolu momenta) daje komponentu vektora statorske struje u $\alpha-\beta$ koordinatnom sistemu:

$$\begin{aligned} i_\alpha &= i_{d0} \cdot \cos\theta_e - i_q \cdot \sin\theta_e \\ i_\beta &= i_{d0} \cdot \sin\theta_e + i_q \cdot \cos\theta_e . \end{aligned} \qquad (2.19)$$

Struje i_α i i_β su izlazne velièine programa INDVEK.ASM, koji je dat na kraju ovog rada, i koje se preko D/A konvertora dovode na strujno regulisani naponski invertor za koga su one sada upravljaèke velièine. U poglavljima koja slede detaljno je obraðena konkretna realizacija dosadašnjeg izlaganja.

3. HARDVER

U ovom poglavlju dat je detaljan opis hardverske osnove ovoga rada. Podela hardvera je izvršena na mikroprocesorsku karticu, koja omoguæuje komunikaciju mikroprocesora sa spoljnim ureðajima, sekciju za merenje brzine i pozicije, koja služi za obradu signala sa enkodera, digitalno-analognu konverziju, gde se vrši konverzija komandovanih vrednosti struja i strujno regulisani naponski invertor, kao konkretnu vezu sa asinhronim motorom. Na kraju poglavlja su priložene detaljne šeme realizacije opisanog hardvera.

3.1. Mikroprocesorska kartica

Na osnovu teorije izložene u prethodnom poglavlju, zakljuèujemo da su za realizaciju vektorske kontrole asinhronog motora neophodna brza i složena izraèunavanja koja nam omoguæava mikroprocesor. Ulaznu velièinu za ova izraèunavanja predstavlja podatak o položaju rotora, dok na izlazu dobijamo komandovane vrednosti struja statora. Zahtev za dvosmernom komunikacijom izmeðu mikroprocesora i ostalih delova opreme realizuje se preko mikroprocesorske kartice, tj. sklopa za prilagoðavanje, èije je rešenje predloženo u uputstvu za *PC - "IBM Technical Reference"*. Ovo rešenje zadovoljava zahteve koji se postavljaju po pitanju komunikacije mikroprocesora sa enkoderom (ulazni podaci) i sa D/A konvertorima (izlazni podaci). Šema realizovane mikroprocesorske kartice data je na kraju poglavlja.

Kartica za prilagoðavanje omoguæava adresiranje elemenata preko kojih se komunicira sa spoljnim hardverom, kao i prenos podataka u oba smera. Za prenos podataka izmeðu spoljnih ureðaja i procesora koristi se kolo 74LS245 (*Octal Bus Transceiver*) èija je uloga osmo-linijska asinhrona dvosmerna komunikacija izmeðu magistrala podataka. Pinom 1 ovog kola, oznaèenim na šemi sa DIR (*DIRection Input*), kontroliše se smer prenosa podataka u zavisnosti od logièkog nivoa pina, dok se pinom 19, na slici oznaèen sa $\overline{G}$, vrši izolovanje magistrale podataka. Komunikacija transivera sa magistralama podataka se obavlja preko pinova oznaèenih sa A1 do A8, sa strane mikroprocesora i sa B1 do B8, sa druge strane. Smer prenosa podataka ređuje se pinovima $\overline{G}$ i DIR prema tabeli:

Tabela 3.1. Ređivanje smera prenosa podataka transivera

$\overline{G}$	DIR	
0	0	podaci idu prema procesoru
0	1	podaci idu od procesora
1	X	magistrale podataka su izolovane

Za adresiranje elemenata preko kojih se komunicira koriste se dva kola 74LS244 (*Octal Buffer*/*Line Driver*). Bafer ostvaruje tri stanja na izlaznim pinovima: nula, jedan i stanje visoke impedanse, u zavisnosti od ulaza $\overline{G}1$ i $\overline{G}2$ na naèin prikazan tabelom:

Tabela 3.2. Određivanje stanja izlaza bafera

$\overline{G}1, \overline{G}2$	D	Izlaz
0	0	0
0	1	1
1	X	(Z)

Na pinove 2 i 4 prvog bafera dovode se signali za èitanje, odnosno upisivanje u ili iz spoljnog uređaja ($\overline{RD}$ i $\overline{WR}$), dok se na pin 11 dovodi RESET signal. Pinovi 13, 15 i 17 prvog bafera, kao i pinovi 2, 4, 6, 8, 11, 13 i 15 drugog bafera spojeni su sa adresnom magistralom. Selektovanje elemenata preko kojih se komunicira sa spoljnim hardverom, ostvaruje se signalom $\overline{CS}$ (*Chip Select*), koji generiše kolo 74LS138 (*Decoder/Demultiplexer*). Pomoæu tri adresne linije (A2, A3 i A4) koje dolaze na ulaze ovoga kola (oznaèene sa A, B i C) moguæe je adresirati osam perifernih uređaja. Kao što se vidi na šemi, na kartici se nalaze èetiri kola koja su povezana sa spoljnim uređajima: tri PPI-e 8255 (*Programmable Peripheral Interface*) i jedan tajmer 8253 (*Programmable Interval Timer*).

PPI 8255 sastoji se od kontrolnog registra i tri *data buffer* registra koje uobièajeno nazivamo portovima A, B i C. Portovi su osmobitni i svaki posebno ostvaruje komunikaciju sa perifernim uređajima, koja može biti bidirekciona. Naèin adresiranja portova i smer protoka podataka između mikroprocesora i porta, u zavisnosti od vrednosti ulaznih signala, dat je u tabeli:

Tabela 3.3. Adresiranje portova i smer protoka signala

A1	A0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
0	0	0	1	0	sa porta A na *data bus*
0	1	0	1	0	sa porta B na *data bus*
1	0	0	1	0	sa porta C na *data bus*
0	0	1	0	0	sa *data bus* na port A
0	1	1	0	0	sa *data bus* na port B
1	0	1	0	0	sa *data bus* na port C
1	1	1	0	0	upis u kontrolni registar ako je D7=1, a ako je D7=0 onda "reset"
X	X	X	X	1	D7-D0 su u stanju visoke impedanse
1	1	0	1	0	nedozvoljena kombinacija
X	X	1	1	0	D7-D0 su u stanju visoke impedanse

Kontrolni registar je takoðe osmobitni i u njega je moguæe upisati podatak sa magistrale, ali ne i èitati ga. Sadržaj ovog registra određuje naèin rada (mod) grupa A i B i definiše smer podataka na svim portovima. Format kontrolnog registra je:

Tabela 3.4. Format kontrolnog registra

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Znaèenje bita D_7 objašnjeno je u prethodno datoj tabeli (1 - set, 0 - reset). Bitovi D_6 i D_5 govore o modu rada grupe A (00 - mod 0, 01 - mod 1 i 1X - mod 2), dok bit D2 određuje mod rada grupe B (0 - mod 0 i 1 - mod 1). Bitom D_4 se definiše smer porta A, a bitom D_1 smer porta B, pri

èemu 1 znaèi da je port ulazni, a 0 da je izlazni. Ista logika važi i za bitove D_3 i D_0 koji se odnose na gornji i donji deo porta C.

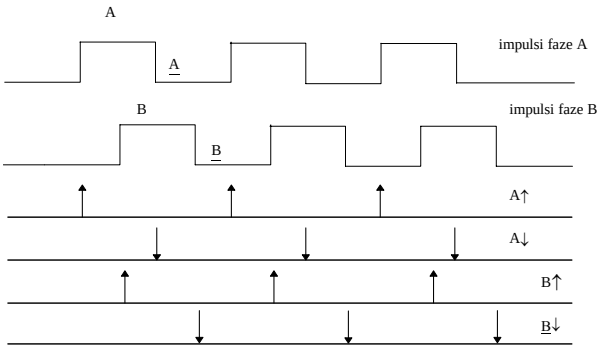
U konkretnom sluèaju svi portovi PPI#1 su definisani kao ulazni, osim donjeg dela porta C, èiji pinovi PC0 i PC1 dolaze na ulaze flip-flopova datih na šemi enkodera. PPI#2 je cela posveæena D/A konverziji, tako da su svi njeni portovi izlazni. Portovi A i B su povezani sa D/A konvertorima prikazanim na šemi D/A konverzije, dok je port C povezan sa dodatnim D/A konvertorom za praæenje prelaznih procesa. Još jedan D/A konvertor je dodat za monitoring i sa njim je povezan port C PPI#3. Portovi A i B ove PPI-e su definisani kao ulazni i povezani su sa leèevima sa kojih se upisuje podatak o trenutnom položaju rotora.

Prikazani tajmer 8253 je iskorišæen za merenje širine impulsa i broja signala inkrementalnog davaèa u kombinovanoj metodi određivanja brzine i njegove adrese su od 30CH do 30FH.

Adrese PPI-ja su od 300H do 303H za PPI#1 (port A ima najnižu, a kontrolni registar najvišu adresu), 304H do 307H za PPI#2 i od 308H do 30BH za PPI#3.

3.2. Sekcija za merenje brzine i pozicije

Ulazne velièine u model rotorskog kola mašine, u kome se određuje ugao vektora rotorskog fluksa u odnosu na α-osu stacionarnog $\alpha-\beta$ koordinatnog sistema, su struje statora i ugaona brzina rotora. Ugaona brzina se izraèunava na osnovu poznate trenutne pozicije rotora, koja se dobija sa inkrementalnog optièkog enkodera ugrađenog u laboratorijski model pogona. Enkoder svoju poziciju i brzinu saopštava preko dva signala, tj. dve povorke impulsa faza A i B. Impulsi faza su međusobno fazno pomereni za 90^0, što omoguæava određivanje smera obrtanja. Detekcijom uzlaznih i silaznih ivica faznih impulsa i njihovom kombinacijom sa logièkim nivoima istih, moguæe je rezoluciju enkodera uveæati èetiri puta. Ovaj postupak ilustrovan je sledeæom slikom:



$$\text{UP} = AB\uparrow + BA\downarrow + \overline{A}\,B\downarrow + \overline{B}\,A\uparrow$$
$$\text{DOWN} = AB\downarrow + \overline{B}\,A\downarrow + \overline{A}\,B\uparrow + BA\uparrow$$

Sl. 3.1. Impulsi faza enkodera

U konkretnom sluèaju, enkoderu sa 1250 impulsa po obrtaju izvršeno je udvostruèavanje impulsa na naèin koji je prikazan na šemi enkodera (brojaèi), èime mu je rezolucija poveæana na 2500 impulsa po obrtaju. Na izlazima kola IC7 i IC6 (oba su 74LS54) dobijeni su signali $\text{DOWN} = AB\downarrow + \overline{A}\,B\uparrow$ i $\text{UP} = AB\uparrow + \overline{A}\,B\downarrow$, koji detektuju smer obrtanja. Ovi signali se dovode na ulaze UP (*Count Up*) i DN (*Count Down*) èetvorobitnog binarnog brojaèa 74LS193 oznaèenog na šemi sa IC8. Izlazi ovoga brojaèa CO (*Carry Output*) i BO (*Borrow Output*) dolaze na ulaze još dva ovakva brojaèa, što

omoguæava zapis dvanaestobitnog binarnog broja. Izlazi ovih brojaèa su dovedeni na ulaze èetvorobitnih D leèeva 74LS75 èiji su izlazi povezani sa portovima A i B PPI#3. Kako se podaci na izlazima leèeva ne mogu èitati u toku tranzicije, nije moguæe oèitavati podatak o položaju rotora tokom prenosa informacije zbog moguæe greške. Zbog toga se javlja potreba za signalom ENABLE, èije nastajanje je prikazano na šemi enkodera (flip-flopovi). Slanjem jedinice na pinove 0 i 1 porta C PPI#1, ovaj signal postaje nula i zadržava trenutno stanje na izlazima sva tri leèa, èime obezbeđuje siguran prenos informacije. Ponovni protok podataka sa ulaza na izlaze leèeva omoguæuje se slanjem nula na pinove 0 i 1 porta C PPI#1.

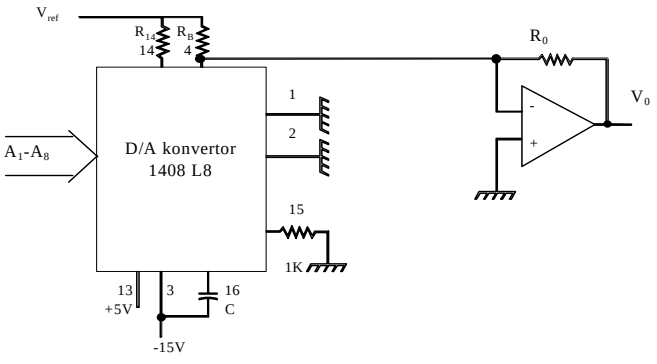
Nakon što disk enkodera napravi jedan obrt u UP ili DOWN smeru, generišuæi pri tom 2500 impulsa, javlja se zahtev za postavljanjem novih stanja brojaèa (reset i set). U sluèaju kada dolaze UP signali na brojaèe, pri èemu se njihova stanja inkrementiraju, potrebno je izvršiti reset brojaèa u trenutku kada izlazi na njima dostignu vrednost 2499. Potrebnu informaciju o dostignutoj vrednosti nam obezbeđuje kolo IC14, koje preko flip-flopa (kolo 74LS74, obeleženo sa IC17) generiše signal jedan na ulazu MR (*Master Reset Input*), èime se postavljaju nule na izlazima svih brojaèa.

Obrnuta logika važi za DOWN smer, pri èemu se sada detektuje stanje koje odgovara nuli na izlazima brojaèa. Izlazi brojaèa su povezani sa ulazima kola 74LS27, koja u trenutku kada se pojave sve nule na njihovim ulazima, preko kola IC3, IC5 i IC16 u kombinaciji sa signalom faze B enkodera, generišu nulu na izlazu $\overline{Q}$ flip-flopa (IC17). Ovaj signal èini aktivnim ulaz $\overline{PL}$ (*Parallel Load Input*) svih brojaèa, èime se postavlja novo stanje brojaèa koje je prisutno na njihovim paralelnim ulazima. To novo stanje odgovara broju 2499, èime obezbeđujemo poèetak novog ciklusa odbrojavanja za smer DOWN.

Potreba za galvanskim odvajanjem elektriènih kola izvedena je pomoæu optokaplera između kola ICZ, na koje dolaze signali sa enkodera i kola ICX. Ovim smo završili opis sekcije za merenje brzine i pozicije i prelazimo na opis digitalno-analogne konverzije.

3.3. Digitalno-analogna konverzija

Složena izraèunavanja koja se obavljaju u mikroprocesoru i koja rezultuju izraèunatim vrednostima komandovanih struja u $\alpha-\beta$ koordinatnom sistemu u digitalnom zapisu, potrebno je konvertovati u analogni oblik signala neophodan za strujnu regulaciju naponskog invertora. Digitalno-analognu konverziju vrši D/A konvertor (DAC) 1408L8 èiji je raspored pinova i naèin vezivanja prikazan na slici 3.2 .



Sl. 3.2. Digitalno-analogni konvertor 1408L8

Binarni ulazi ovog konvertora su pinovi 5 (najveæe težine - MSB) do 12 (najniži - LSB), na koje dolazi digitalna reè koju treba konvertovati. Analogni napon na izlazu operacionog pojaèavaèa imaæe vrednost određenu jednaèinom po kojoj se vrši konverzija, a u zavisnosti od stanja na ulaznim pinovima:

$$V_0 = \frac{V_{ref}}{R_{14}} \cdot R_0 \cdot \left[\frac{A_1}{2} + \frac{A_2}{4} + \frac{A_3}{8} + \frac{A_4}{16} + \frac{A_5}{32} + \frac{A_6}{64} + \frac{A_7}{128} + \frac{A_8}{256} \right] - \frac{V_{ref}}{R_B} \cdot R_0 \quad (3.1)$$

Referentni napon koji figuriše u jednaèini ima vrednost od +5V, dok su vrednosti otpornika R_{14}, R_0 i R_B : 1kΩ, 2kΩ i 4kΩ respektivno. Vrednost izlaznog napona V_0 kreæe se u opsegu od -10V do +10V za vrednosti digitalne reèi od 0 do 255. Kako je prihvatni registar DAC-a dovoljne dužine, tako da prihvata bez odsecanja sve digitalne vrednosti ulaznog signala, analogni napon na izlazu ostaje sa zateèenom konstantnom vrednošæu sve dok se ulazni digitalni signal ne promeni.

Digitalna reè na ulaze DAC-a dolazi preko dva invertora i optokaplera, što je prikazano na šemi D/A konverzije na. Optokaplerom je izvršeno galvansko odvajanje mikroprocesora od ostalog dela upravljaèkog kola. Postojanje dva D/A konvertora 1408L8 je uslovljeno zadavanjem struja po α i β–osi stacionarnog koordinatnog sistema. Izraèunate vrednosti ovih struja se izbacuju na portove A i B PPI#2, prikazane na šemi PC kartice, koji su na pomenuti naèin povezani sa konvertorima. Izlazne vrednosti èitave D/A konverzije su analogni signali komandovanih struja i to struje po α-osi sa DAC1 i njegovog operacionog pojaèavaèa, a struje po β–osi sa DAC2 i njemu pripadajuæeg operacionog pojaèavaèa. Ovako dobijene analogne vrednosti struja dovodimo na ulaze kartice strujne regulacije.

3.4. Strujno regulisani naponski invertor

Za raspregnuto upravljanje momentom i fluksom kod vektorske kontrole asinhronog motora potrebno je odrediti i orjentaciju i amplitudu vektora magnetopobudne sile. Prostorna raspodela magnetopobudne sile zavisi od struja, pa ukoliko je moguæe kontrolisati struje statorskog namota i magnetopobudna sila se može usmeriti u proizvoljan položaj uz željenu amplitudu. Zato je jedan od zahteva koji se postavlja pred pretvaraè za napajanje motora i moguænost brze izmene amplitude i faze statorskih struja. Strujni invertor ispunjava ovaj zahtev u pogledu kontrole faznog stava, ali je brzina promene amplitude struje ogranièena velièinom prigušnice u jednosmernom međukolu pretvaraèa. Naponski invertor uz adekvatne strujne regulatore omoguæava brze promene amplitude i orjentacije vektora statorske struje, zbog èega se uobièajeno koristi za napajanje asinhronih motora u pogonima visokih performansi.

Naponski invertor sa strujnom regulacijom primenjen u ovom radu možemo funkcionalno i organizacijski podeliti na sledeæe delove međusobno povezane: energetsko kolo, strujni regulator, prekostrujnu i prenaponsku zaštitu, pojaèavaèe impulsa i blok za napajanje elektronike, što je i prikazano na strani

Šema energetskog kola pretvaraèa prikazana je na kraju poglavlja. Prekidaèkom akcijom trofaznog tranzistorskog mosta, jednosmerni napon U_c se konvertuje u trofazni sistem naizmeniènih napona U_a, U_b i U_c, koji imaju diskretan karakter. Ovi izlazni naponi se po potencijalu mogu izjednaèiti sa pozitivnom ili negativnom oblogom kondezatora C_1, u zavisnosti od upravljanja prekidaèima Q_1 do Q_6. Pri tome linijski naponi U_{ab}, U_{bc} i U_{ca}

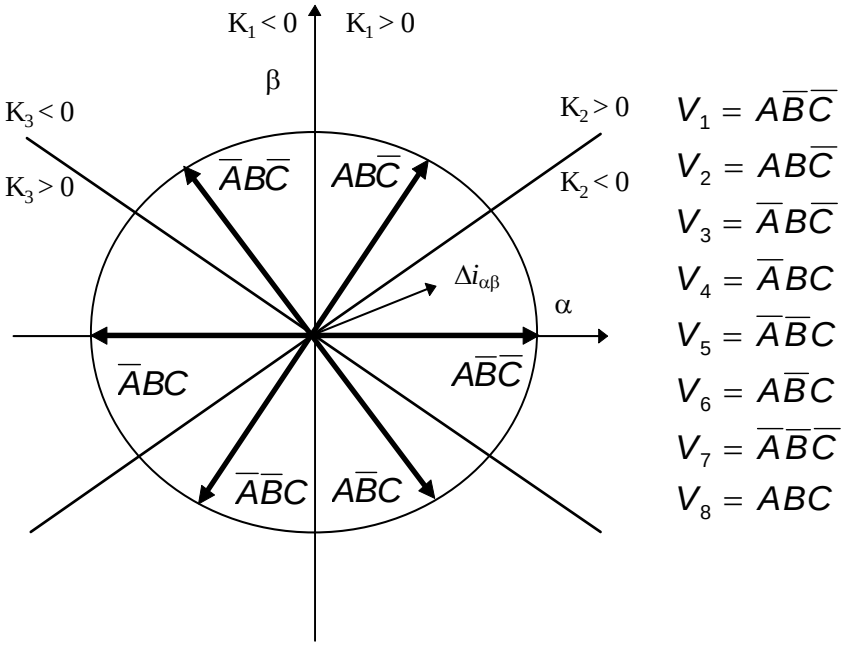
uzimaju vrednosti $+U_c$, $-U_c$ ili 0. Neidealnosti invertora se ogledaju u tome što poluprovodnièki prekidaèi imaju konaèan pad napona, što prouzrokuje zavisnost izlaznih napona od amplitude i polariteta struje. Pored toga prelazni procesi u poluprovodnièkim prekidaèima zahtevaju manju ili veæu pauzu između voðenja gornjeg i donjeg prekidaèa u svakoj od grana (Q_1-Q_2, Q_3-Q_4 i Q_5-Q_6), pa se javljaju intervali vremena (*dead time*) kada su izlazni naponi van kontrole. U neidealnosti se ubraja i konaèno vreme kašnjenja između promene stanja upravljaèkih signala B_1 do B_6 i promene stanja prekidaèa invertorskog mosta. Upravljanjem prekidaèima Q_1 do Q_6 potrebno je omoguæiti upravljanje statorskim strujama tako da se minimizira uticaj nelinearnosti i zavisnost struja od stanja i parametara motora.

Strujni regulator realizuje upravljaèki algoritam naponskog invertora. Njegove karakteristike bitno utièu na performanse vektorski kontrolisanog pogona, jer pored dinamike treba da obezbede i što manju grešku stacionarnog stanja. U konkretnom sluèaju primenjen je nelinearni pristup zasnovan na diskretnom karakteru izlaznih napona. Stanje prekidaèa u invertorskom mostu se određuje dovoðenjem greške regulisane brzine na ulaz komparatora sa histerezisom. Regulacija je izvršena u stacionarnon $\alpha-\beta$ koordinatnom sistemu gde se upravljanje invertorom vrši na osnovu vektora strujne greške: $\Delta\vec{i}_{\alpha\beta} = \vec{i}_{\alpha\beta}{}^{*} - \vec{i}_{\alpha\beta}$. Na ovaj naèin, zatvaranjem povratne sprege po vektoru struje $\vec{i}_{\alpha\beta}$ umesto po faznim strujama, izbegava se primena tri nezavisna regulatora međusobno zavisnih struja: $i_a + i_b + i_c = 0$. Zadate referentne vrednosti struja su i_α^* i i_β^*, dok je veza između faznih struja motora i komponenti vektora struje u stacionarnom $\alpha-\beta$ koordinatnom sistemu data jednaèinama:

$$\begin{aligned} i_\alpha &= i_a - \frac{i_b}{2} - \frac{i_c}{2} \\ i_\beta &= \frac{\sqrt{3}}{2}(i_b - i_c). \end{aligned} \qquad (3.2)$$

Naèin realizacije strujnog regulatora prikazan je na kraju ovog poglavlja. Informacije o faznim strujama se dobijaju na osnovu poznatih vrednosti tri otpornika na jednom kraju vezanih u zvezdu. Na ovaj naèin je dobijen šant za merenje struja i ostvarena veza namota statora u zvezdu.

Skup izlaznih vektora, koji se mogu postiæi upravljanjem prekidaèima invertora u vertikalama A, B i C, dat je na slici 3.3. Slova A, B i C oznaèavaju gornje prekidaèe u granama invertora, a podvuèeno $\overline{A}$, $\overline{B}$ i $\overline{C}$ donje prekidaèe u istim granama.



Sl. 3.3. Određivanje vektora statorskog napona na osnovu strujne greške u $\alpha-\beta$ koordinatnom sistemu

Pomoæne promenljive K_1, K_2 i K_3 se formiraju iz komponenata greške $\Delta i_\alpha = i_\alpha^* - i_\alpha$ i $\Delta i_\beta = i_\beta^* - i_\beta$:

$$K_1 = \Delta i_\alpha$$
$$K_2 = \frac{1}{2}\Delta i_\beta - \frac{1}{2\sqrt{3}}\Delta i_\alpha \qquad (3.3)$$
$$K_3 = -\frac{1}{2}\Delta i_\beta - \frac{1}{2\sqrt{3}}\Delta i_\alpha .$$

Signali za upravljanje prekidaèima invertora (B_1 do B_6) određuju se na osnovu znaka pomoænih promenljivih K_1, K_2 i K_3. Na slici 3.3. se može uoèiti da nema preklapanja oblasti $K_1 > 0$, $K_2 > 0$ i $K_3 > 0$, niti preklapanja oblasti $K_1 < 0$, $K_2 < 0$ i $K_3 < 0$. U sluèaju da komparatori promenljivih K_1, K_2 i K_3 nemaju ofset, pri radu invertora ne bi dolazilo do uspostavljanja "nultih" naponskih vektora $V_7 = \overline{A}\,\overline{B}\,\overline{C}$ i V_8 = ABC. Potreba za "nultim" vektorima se javlja kod primene komparatora sa histerezisom zbog rešavanja problema koji se ogleda u poveæanju komutacione uèestanosti u oblasti malih brzina. Ofset jednakog polariteta i amplitude prouzrokuje periodièni pojavu "nultih" vektora napona. Ofset negativnog polariteta za posledicu ima produženo vreme voðenja prekidaèa Q_2, Q_4 i Q_6 i skraæenje vremena voðenja prekidaèa Q_1, Q_3 i Q_5, jer prouzrokuje periodièni primenu "nultog" naponskog vektora V_7. Obrnuto važi za ofset pozitivnog polariteta i "nulti" naponski vektor V_8. Na prikazani naèin se vektor statorske struje $\vec{i}_{\alpha\beta}$ održava u histerezisom određenoj oblasti oko referentne vrednosti, što èini amplitudnu i faznu grešku zanemarljivim. Taènost u regulaciji statorskih struja se održava sve dok invertor ima dovoljnu naponsku marginu, odnosno dok elektromotorne sile indukovane u namotajima ne dostignu vrednost maksimalnog raspoloživog napona. Nelinearni strujni regulatori su praktièno neosetljivi na nesavršenosti invertora, a uticaj parametara i stanja motora na regulaciju struje je gotovo eliminisan.

Šema kartice prenaponske i prekostrujne zaštite data je na kraju poglavlja. Potreba za prenaponskom zaštitom se javlja usled napajanja invertorskog mosta preko

diodnog ispravljaèa, koji nije u moguænosti da energiju vraæa u mrežu. Zahtev za disipacijom energije u tom sluèaju zadovoljava takozvani blok za koèenje, koji se sastoji od tranzistora, diode i njoj paralelno vezanog otpornika, koji se mogu videti na šemi energetskog kola. Upravljanje tranzistorom u bloku za koèenje vrši se na osnovu informacije o naponu koja dolazi sa pozitivne sabirnice invertorskog mosta na "minus" ulaz operacionog pojaèavaèa. U sluèaju prekoraèenja naponske margine, dolazi do uključèenja prekidaèa Q_7 i energija se disipira na otporniku vezanom redno sa prekidaèem. Informacija o struji stiže do upravljaèke logike prekostrujne zaštite preko šanta vezanog u negativnu sabirnicu invertorskog mosta, takođe prikazanog na šemi energetskog kola. U sluèaju dostizanja gornje strujne granice dolazi do uključèivanja prekostrujne zaštite, što dovodi do paljenja donjih tranzistora Q_2, Q_4 i Q_6 u granama invertora èime se blokira njegov rad. Nakon otklanjanja uzroènika kvara, potrebno je deblokirati invertor i stanje na izlazu vratiti u normalno radno stanje tako što æemo dovesti signale ispod donjeg praga. To je moguæe uraditi samo pritiskom na taster za reset.

Pojaèavaèi impulsa, ili kako se još nazivaju upaljaèi tranzistora, predstavljaju vezu između strujnog regulatora i tranzistora u energetskom kolu invertora. Izlazi sa kartice strujne regulacije, kojih ima dvanaest, dolaze po dva na ulaz svakog od pojaèavaèa impulsa, gde se prvo vrši galvansko odvajanje kola regulacije od energetskog kola preko optokaplera. Upravljaèki signal sa izlaza optokaplera, preko pojaèavaèkog stepena, dolazi na bazu i emitor energetskog tranzistora. To znaèi da imamo šest upaljaèa tranzistora, plus sedmi upaljaè za tranzistor u bloku za koèenje kome upravljaèki signal stiže sa kartice prenaponske zaštite. Na karticu upaljaèa tranzistora, koja je prikazana na kraju ovoga poglavlja, smeštena su po dva upaljaèa zajedno, što znaèi da imamo èetiri kartice upaljaèa tranzistora, pri èemu je jedan upaljaè neiskorišæen.

Šema bloka za napajanje elektronike je data na kraju poglavlja. U blok za napajanje dolazi jednosmerni napon od 300V sa posebnog ispravljaèa prikazanog na šemi energetskog kola invertora, èime je razdvojeno napajanje energetskog dela od upravljaèkog. Elektronika upravljaèkih kola se napaja sa +/- 15V koji se dobijaju na izlazu bloka za napajanje. Ovo spuštanje napona je izvedeno preko samooscilujuæeg DC-DC pretvaraèa, kod koga glavni prekidaè u oscilatornom kolu T_2, pravi èetvrtke na primaru feritnog transformatora TR_2. Na sekundaru ovog transformatora se nalazi jedanaest izvoda, na èijim krajevima su vezani diodni mostovi koji preko stabilizatora napona daju željeni napon na izlazu. U invertoru postoje dve kartice blokova za napajanje, sa kojih se napajaju èetiri kartice pojaèavaèa impulsa i po jedna kartica strujnog regulatora i jedna prenaponske i prekostrujne zaštite.

Ovim smo završili opis hardvera upotrebljenog u ovom radu, dokumentovanog šemama koje slede, a na koje smo se pozivali u toku izlaganja i prelazimo na softversku osnovu rada u narednom poglavlju.

4. SOFTVER

Softver je u ovom poglavlju izložen postupno, poèev od osnovnih pojmova o asemblerskom jeziku, *assembler*-u, *linker*-u i *debugger*-u, do primera jednostavnog programa. Nakon toga je data i lista svih potprograma glavnog programa sa opisom njihovog rada, da bi na kraju bio izložen i listing programa INDVEK.ASM dokumentovan opširnim komentarima.

4.1. Softverska osnova

Programski jezik koji je korišæen u ovom radu je asembler, baziran na *Microsoft Macro Assembler*, verzija 1.25. Asemblerski jezik ne spada u jezike višeg nivoa, što programiranje u ovom jeziku èini sporijim i težim. Vreme potrebno za definisanje problema, odstranjivanje i ispravljanje grešaka u programu, testiranje i izradu dokumentacije je mnogostruko duže nego kod viših programskih jezika. Pored toga, potrebno je voditi raèuna i o konfiguraciji raèunara na kome se program izvršava.

Sa druge strane, programski jezici višeg nivoa zahtevaju obimniji hardver i softver, teško se optimiziraju i nemaju moguænosti da koriste specifiène karakteristike raèunara. Prednosti asemblerskog jezika ogledaju se i u proizvoðenju efikasnijih programa na mašinskom jeziku. Primena asemblera pokazala se efikasnijom kod pisanja ulazno/izlaznih rutina, vremenskog ubrzavanja i skraæivanja kritièih sekcija, rada sa *ROM BIOS* i *DOS* servisima, kao i kod korišæenja ili modificiranja operativnih sistemskih funkcija.

Program napisan u asemblerskom jeziku predstavlja izvorni modul koji je potrebno prevesti na mašinski jezik, razumljiv mikroprocesoru, što radi program nazvan *assembler*. On konvertuje instrukcije asemblerskog jezika u binarni oblik, tj. u oblik poznat kao object code. Ovako formiran fajl ima ekstenziju *.OBJ i pored izvršnog mašinskog koda sadrži i dodatne informacije o strukturi izvršnog programa. Sada je potrebno izvršiti povezivanje odvojenih delova objekt fajlova u jedan i njegovo konvertovanje u izvršni program sa ekstenzijom *.EXE. To se radi pomoæu programa koji se zove *linker*. Program *debugger* omoguæava da program unesemo u sistemsku memoriju i izvršavamo ga naredbu po naredbu. Istovremeno pratimo šta se dešava sa sadržajima registara i memorijskih lokacija i na taj naèin vršimo proveru predloženog rešenja i otklanjanje eventualno napravljenih grešaka.

Na sledeæem jednostavnom primeru biæe objašnjena struktura programa pisanih u asemblerskom jeziku, njihovo prevoðenje, povezivanje i provera.

```
;*****************************************************************
        name primer           ;Ime programa-procedure koji ucitava dva
                              ;broja od 0 do 9 sa tastature,vrsi njiho-
                              ;ovo mnozenje i ispisuje rezultat na ekranu
        stek  segment stack   ;Ovako se rezervise stek,potreban za sme-
        db     64    dup(?)   ;stanje trenutnih vrednosti registara pri-
        stek  ends            ;likom prekida i skokova.
;*****************************************************************
        data  segment         ;U ovom segmentu se deklarisu promenljive.
        x      db   1         ;Potrebne su 3 varijable u RAM-u: dve du-
        y      db   1         ;zine 1 byte (db) za unesene cifre i jedna
        z      dw   1         ;duzine 2 byte =word (dw) za dobijeni re-
        data  ends            ;zultat.
```

```
;*************************************************************************
        prog  segment        ;Segment u kome se nalazi izvrsni deo.
        assume cs:prog,ss:stek,ds:data ;Pridruzivanje segmenata deklari-
                                       ;sanih navedenim imenima segment-
                                       ;nim registrima.
        radi  proc                     ;Pocetak procedure 'radi'.

        mov ax,data          ;Asembler zahteva eksplicitno punjenje
        mov ds,ax            ;data segmenta na navedeni nacin.
;*************************************************************************
petlja: mov ah,01H    ;Interrupt 21H (General DOS Service) je jedan od 5
        int 21H       ;glavnih DOS-ovih interrupt-a koji omogucuje poziv
                      ;svih DOS-ovih funkcija specifikacijom njenog bro-
                      ;ja u registru AH. Funkcija 01H (Character Input
                      ;with Echo) ceka korisnika da posalje karakter,ci-
                      ;ta karakter sa tastature, smesta ga u registar AL
                      ;i radi "eho", tj. karakter ispisuje na ekranu.
        cmp al,30H    ;Uneseni karakter je u ASCII kodu,pa ako smo uneli
        je  exit      ;broj 0,ciji je ASCII kod 30H,izlazimo iz programa
        sub ax,30H    ;Konvertujemo karaktere od 30H do 39H koji predsta
        mov x,al      ;vljaju ASCII kod brojeva 0 do 9 u njihove stvarne
                      ;vrednosti.
        call novired  ;Pozivamo potprogram 'novired' koji vrsi pozicio-
                      ;niranje cursor-a u novi red.
        mov ah,01H    ;Sada ucitavamo drugi broj, vrsimo njegovu konver-
        int 21H       ;ziju iz ASCII koda i upisujemo ga u promenljivu y
        sub ax,30H    ;na identican nacin kao za prvi broj.
        mov y,al

        call novired  ;Ponovo pozivamo potprogram 'novired'.

        mov al,x      ;Vrsimo mnozenje ova dva broja (duzine byte) nare-
        mul y         ;dbom MUL, koja zahteva da se jedan od operanada
        aam           ;nalazi u akumulatoru AL.Naredbom AAM(ASCII Adjust
        mov z,ax      ;After Multiply) prilagodjavamo dobijeni rezulta
                      ;ispisu na ekran, tako da prvo bude ispisan broj
                      ;desetica pa broj jedinica. AAM instrukcija uzima
                      ;vrednost broja u AX, gde je smesten rezultat po
                      ;izvrsenju naredbe MUL, i u AH upisuje broj dobi-
                      ;jen deljenjem sadrzaja registra AX sa 10. U AL se
                      ;upisuje vrednost dobijena po oduzimanju broja iz
                      ;AH pomnozenog sa 10 od rezultata mnozenja.Ovako
                      ;obradjen broj se smesta u promenljivu z.
        mov dl,ah     ;Dobijeni broj pripremamo za ispis na ekranu tako
        add dl,30H    ;sto prvo formirmo ASCII kod cifre desetica u regi
        mov ah,02H    ;stru DL dodajuci mu broj 30H. Funkcija 02H (Chara
        int 21H       ;cter Output) ispisuje na ekranu karakter ciji je
                      ;ASCII kod smesten u registru DL. Nakon izvrsenja
                      ;instrukcije INT 21H, broj desetica ce biti ispi-
                      ;san na ekranu.
        mov ax,z      ;Opisani postupak ponavljamo za broj jedinica koji
        mov dl,al     ;je bio smesten u donjem byte-u promenljive z,koju
        add dl,30H    ;prebacujemo u registar AX,a odatle iz registra AL
        mov ah,02H    ;u registar DL,gde formiramo ASCII kod cifre jedi-
        int 21H       ;nica.

        call novired  ;Pozivamo potprogram 'novired'.

        jmp petlja    ;Povratak na pocetak petlje.
;*************************************************************************
novired:mov dl,0AH    ;Potprogram kojim se obezbecuje prelazak u novi
        mov ah,02H    ;red. Postavljajuci 0AH u registar DL, funkcija
        int 21H       ;02H ce popuniti tekuci red (Line Feed), a sa
        mov dl,0DH    ;0DH u DL izvrsiti povratak cursor-a na pocetak,
        mov ah,02H    ;sada sledeceg reda (Carriage Return).
```

```
        int 21H
        ret             ;Povratak na mesto sa koga je potprogram pozvan.
;*************************************************************************
exit:   mov ah,4CH      ;Izlazak iz programa i povratak u DOS pomocu funk-
        int 21H         ;cije 4CH (Terminate with Return Code) interrupt-a
                        ;21H.
;*************************************************************************
        prim    endp    ;Kraj procedure 'prim'.
        prog    ends    ;Kraj programskog segmenta.
        end     prim    ;Kraj programa.
;*************************************************************************
```

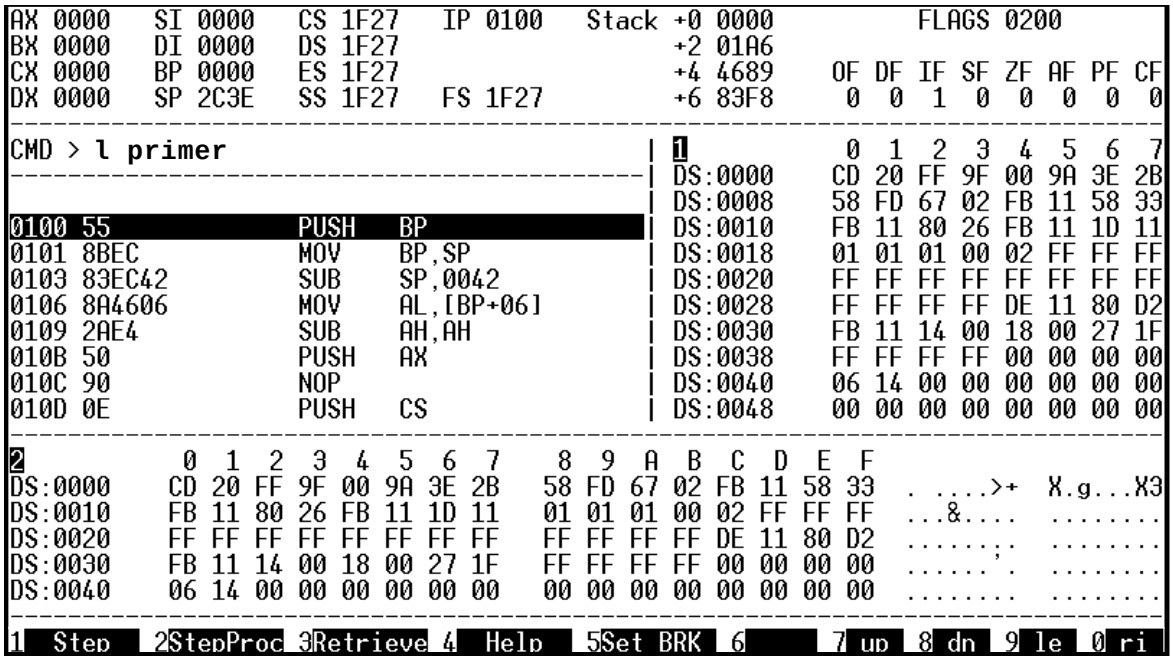
Program PRIMER.ASM vrši množenje dva jednocifrena broja i ispisuje rezultat na monitoru. Da bi ovako napisani program mogao da se izvršava, potrebno je prvo izvršiti njegovo prevođenje pomoæu *Microsoft MACRO Assembler*, verzija 1.25 jednostavnom komandom:

```
MASM PRIMER     <CR>
```

Rezultat ove naredbe biæe kreiranje objekt fajla pod imenom PRIMER.OBJ uz obaveštenje o eventualno napravljenim greškama. Ovaj file æemo sada uèiniti izvršnim pomoæu *Microsoft Linker*, verzija 2.40:

```
LINK PRIMER     <CR>
```

Nakon ovoga dobijamo izvršni program PRIMER.EXE, koji je spreman za upotrebu. Otkrivanje moguæih grešaka u programu vršimo pomoæu *IBM Fulscreen Debugger* (FSD), verzija 1.4. Po startovanju debagera pojavljuje se maska programa koja u gornjem levom delu ekrana prikazuje sadržaje svih registara, dok se desno od njih nalazi sadržaj Procesorske Statusne Reèi (PSW). U levom delu ekrana ispod registara nalazi se komandna linija, ispod koje se vidi deo listinga programa èiji se sadržaj proverava. Sadržaj memorijskih lokacija na navedenim adresama prikazan je desno od listinga, kao i ispod njega. U poslednjem redu ekrana nalazi se meni sa znaèenjem tastera za èesto korišæene komande. Upisivanje programa u FSD vrši se naredbom: `l ime_programa`, ispisanom u komandnoj liniji. Po završenom testiranju izlazak se obezbeđuje naredbom `quit.`



sl. 4.1. Radni ekran *IBM Fulscreen Debugger*

Podrška navedenom softveru, kao i programu INDVEK.ASM je *IBM PC/AT* raèunar opšte namene, koga pokreæe operativni sistem *MS DOS*, verzija 5.0.

4.2. Struktura programa

Osnovu ovoga rada predstavlja program INDVEK.ASM, koji možemo podeliti na dve meðusobno povezane celine. Jednu celinu predstavlja deo koji se odnosi na inicijalizaciju perifernih jedinica, promenu uèestanosti interapta i komunikaciju sa korisnikom. U ovu celinu spada i deo kojim se ispisuje rezultat na ekranu, kao i deo koji obezbeðuje nedestruktivni izlazak iz programa. Skup ovih delova predstavlja proceduru nazvanu `prim`.

Na samom poèetku programa rutinom UZIMA saèuvani su segmentna adresa i ofset originalne prekidne rutine interapta 1CH neophodni za izlazak iz programa. Rutinom INICI su postavljeni modovi u kojima rade PPI-e, dok se u delu programa nazvanom PROM vrši promena uèestanosti generisanja hardverskog interapta 08H od strane sistemskog tajmera. Ovom rutinom dodeljuje se i adresa nove procedure na èije izvršavanje se prelazi sa nastankom interapta 1CH, koga generiše interapt 08H. Deo nazvan PETLJA obezbeðuje ispis izraèunte brzine na ekranu. Ova rutina izvršava se asinhrono, prekidana od strane interapta, nastavlja sa radom po završetku prekidne rutine od mesta na kome je bila prekinuta. Takoðe obezbeðuje i izlazak u rutinu DIALOG kojom se komunicira sa korisnikom. Još jedna rutina je predviðena za komunikaciju sa korisnikom.To je deo programa nazvan UNOS u kome se od korisnika zahteva da unese željenu brzinu motora u predviðenom opsegu, da bi se potom uneta brzina prilagodila obliku neophodnom za dalji rad programa. Rutina EXIT vraæa sistemskom tajmeru njegovu standardnu uèestanost i interaptu 1CH dodeljuje njegovu originalnu prekidnu rutinu. Nakon toga vrši se izlazak iz programa i povratak u *DOS*.

Drugu celinu programa INDVEK.ASM predstavlja procedura `prenos`. Ona se izvršava svake milisekunde generisanjm interapta 1CH i predstavlja deo programa u kome se softverski realizuje indirektna vektorska kontrola.

Rutina BROJAC uzima informaciju o trenutnom položaju rotora, koja se zatim obraðuje u delu programa pod imenom BRZROT. Softverska realizacija brzinskog PI regulatora u inkrementalnom obliku ostvarena je rutinom PIREG. Nakon toga je izraèunat položaj vektora rotorskog fluksa u rutini UGAO, da bi transformaciju koordinata iz rotacionog d-q sistema u stacionarni $\alpha-\beta$ koordinatni sistem realizovali u delu programa nazvanom STRUJE. U ovoj rutini su korišæene odgovarajuæe vrednosti sinusa i kosinusa izraèunatog ugla rotorskog fluksa dobijene iz tabele SINBRZ.INC, koja je formirana programom SINBRZ.BAS napisanim u *QBASIC*. Ovde je prikazan listing ovoga programa:

```
;*********************************************************************
DIM m(256)

pi = 3.1415926#

OPEN "sinbrz.inc" FOR OUTPUT AS #1

FOR i = 0 TO 255
        teta = 2 * pi * i / 256
        m(i) = SIN(teta) * 4000
NEXT i

PRINT #1, ";sinusna tabela za vrednosti ugla od 0 do 255:"
FOR i = 0 TO 255
PRINT #1, "  DW",
x = m(i)
a$ = HEX$(x)
PRINT #1, "0"; a$; "H;"
NEXT i
CLOSE #1
END
;*********************************************************************
```

Rutinom EKRAN je omoguæeno praæenje prelaznih procesa izbacivanjem odgovarajuæih vrednosti brzine na dodatni D/A konvertor. Lista potprograma se završava rutinom izlaz kojom se vraæaju sadržaji registara na vrednosti koje su imali pre trenutka nastajanja interapta.

4.3. Listing programa

U ovom poglavlju je dat listing programa INDVEK.ASM sa adekvatnim komentarima. Programom je omoguæeno zadavanje referentnih vrednosti kontrolisane velièine i oèitavanje njene trenutne vrednosti na ekranu. Celokupna komunikacija sa korisnikom obavlja se na srpskom jeziku. Program je preveden, povezan i isproban na laboratorijskom modelu pogona. Pokazao je zadovoljavajuæe rezultate, što je dokumentovano i osciloskopskim snimcima u narednom poglavlju.

```
;*********************************************************************
***
;*     Program   za   indirektnu   vektorsku   kontrolu   asinhronog   motora
*
;*                  diplomski         rad         Gorana         Medenice
*
;*                                      septembar                    1995.
*
;*********************************************************************
***
      name indvek             ;Ime programa.

      stek   segment   stack      ;Na ovaj nacin rezervisemo prostor na
steku
      db    128  dup(?)       ;gde ce se smestati tekuce vrednosti regi-
```

```
      stek  ends              ;stara prilikom prekidnih rutina.
;*************************************************************************
***
      data  segment           ;Deo programa u kome se deklarisu prome-
                              ;nljive.
      include sinbrz.inc        ;Uvrstavamo tabelu sinusnih vrednosti u
data
                              ;segment, nezavisno formiranu u qbasic-u.
      zeljena   db   'Vasa   zeljena   brzina   je   (-01390   *   001390):$'
;Tekstualni
      upozori db 'Zeljena brzina nije korektna.$'             ;podaci za
      izlazak db 'Zelite li da napustite program (y/n)?$'     ;komunika
      promena db 'Zelite li da promenite brzinu (y/n)?$'      ;ciju sa

;korisnikom.
      smer         db   ?          ;Promenljive koje koristimo za ispis
trenu-
      x          db  ?        ;tne brzine na monitoru.
      y          db  ?
      z          db  ?
      w          db  ?
      hil             dw   3E8H        ;Definisemo konstantne vrednosti
promenlji-
      sto        dw  64H      ;vih.
      des        dw  0AH
      tmp        db  ?        ;Promenljiva iskoriscena kao brojac.
      zapis        dw   ?           ;Promenljive koje koristimo prilikom
obrade
      suma          dw   ?            ;signala sa encoder-a i racunanja
trenutnog
      terot      dw  ?        ;polozaja uglova rotora i klizanja u cilju
      teslow      dw  ?          ;pronalazenja ugla koji zaklapa d-osa
rota-
      teshigh    dw  ?        ;cionog koordinatnog sistema sa alfa-osom
      inklow      dw  ?         ;nepomicnog koordinatnog sistema vezanog
za
      inkhigh    dw  ?        ;stator.
      teta_e     db  ?
      trenlow    dw  ?        ;Pomocne promenljive.
      trenhigh   dw  ?
      znak       db  ?
      provera    dw  ?
      stara      dw  ?
      brzina     dw  ?        ;Trenutna vrednost brzine.
      zadbrz     dw  ?        ;Zadata vrednost brzine.
      idzad      dw  126      ;Struja statora paralelna sa d-osom kojom
                              ;se    upravlja    fluksom,    konstantne
vrednosti.
      iqzad      dw  ?        ;Struja statora normalna na d-osu kojom se
                              ;kontrolise moment.
      Kint       dw  10       ;Integralna konstanta u PI brzinskom reg.
      Kprop      dw  7        ;Proporcionalna konstanta u PI brz. reg.
      sinus      dw  ?        ;Promenljive u koje se smestaju vrednosti
      cosin      dw  ?        ;sinusa i cosinusa ugla obrtnog polja.
      stari_seg   dw   1             ;Ovde  se  cuvaju  segment  i  offset
originalne
      stari_off  dw  1        ;prekidne rutine koja se menja u programu.
      zadata     db  7        ;Rezervisano za unos zadate brzine.
      data  ends
;*************************************************************************
***
      prog  segment                       ;Izvrsni deo programa.
      assume cs:prog,ss:stek,ds:data      ;Dodeljivanje segmentnim regi-
                                          ;strima adresa segmenata.
      prim  proc                          ;Pocetak procedure 'prim'.
```

```
      mov ax,data              ;Asembler  zahteva  eksplicitno  punjenje
data
      mov ds,ax                ;segmenta.
;*************************************************************************
***
UZIMA:mov ah,35H              ;Na ovaj nacin ce preko interrupt-a 21H i
      mov al,1CH              ;njegove funkcije 35H biti sacuvana origi-
      int 21H                    ;nalna prekidna rutina interrupt-a 1CH
koji
      mov dx,es                  ;se koristi u izmenjenom obliku u ovom
pro-
      mov stari_seg,dx        ;gramu.
      mov stari_off,bx
;*************************************************************************
***
INICI:mov dx,303H             ;Postavljanje CR prve PPI-e koji se nalazi
      mov al,9AH                ;na adresi 303H cime se definisu modovi
ra-
      out dx,al               ;da portova A,B,C. Modovi se odreduju na
                              ;osnovu  brojne  vrednosti  upisane  u  CR
(9AH).
      mov dx,307H                ;Postavljanje CR druge PPI-e koji se
nalazi
      mov al,80H              ;na adresi 307H
      out dx,al

      mov dx,30BH                ;Postavljanje CR trece PPI-e koji se
nalazi
      mov al,92H              ;na adresi 30BH
      out dx,al
;*************************************************************************
***
      mov  tmp,23                   ;Postavljanje  pocetnih  vrednosti
promenljivih.
      mov suma,0
      mov stara,0
                              ;Pozivanje potprograma za komunikaciju sa
      call unos               ;korisnikom i unos podataka sa tastature.
;*************************************************************************
***
PROM: cli                     ;Onemogucavanje dogadjanja interrupt-a.
      mov dx,43H              ;Promena ucestanosti generisanja interrupt
      mov al,36H              ;08H od strane sistemskog timer-a sa 55ms
      out dx,al                ;na 1ms unosenjem nove vrednosti u brojac
0
      mov dx,40H              ;koji se nalazi na adresi 40H.
      mov al,54H
      out dx,al
      mov al,02H
      out dx,al
      sti                     ;Omogucavanje desavanja interrupt-a.

      mov  dx,seg  prekid           ;Promena  sadrzaja  prekidne  rutine
interrupt
      mov ds,dx               ;1CH pomocu funkcije 25H interrupt-a 21H.
      mov  dx,offset  prekid       ;Interrupt-u  1CH  se  dodeljuje  nova
prekidna
      mov ah,25H              ;rutina 'prekid' napisana nize u programu.
      mov al,1CH
      int 21H
;*************************************************************************
***
      mov ax,data
      mov ds,ax               ;Spora ili glavna petlja koja se odradjuje
```

```
                              ;van interrupt-a, asinhrono.
PETLJA:mov ah,06H             ;Omogucavanje izlazka iz petlje pritiskom
       mov dl,0FFH            ;na bilo koji taster pomocu funkcije 06H
       int 21H                ;interrupt-a 21H, cime se prelazi na pot-
       jnz dialog             ;program 'dialog'.

       mov dl,0Dh             ;Pozicioniranje cursor-a na pocetak reda
       mov  ah,02h                 ;pomocu funkcije 02H interrupt-a 21H
posta-
       int  21h                    ;vljanjem 0DH u registar dl(carriage
return).

       mov  ax,brzina               ;Priprema za ispisivanje vrednosti
trenutne
       cmp ax,0                  ;brzine na monitor,pri cemu brzina moze
ima-
       jge  skok                    ;ti pozitivnu i negativnu vrednost,
zavisno
       neg ax                 ;od smera obrtanja,sto je potrebno prvo
       mov smer,148           ;ispitati.

skok: sub dx,dx               ;Postavljanje vrednosti 0 u registar dx.
       div hil                ;Deljenje brzine koja se nalazi u registru
       mov x,al               ;ax sa 1000 da bi dobili cifru koju cemo
                              ;ispisati    na    monitoru    na    mestu
hiljada,sme-
       mov ax,dx               ;stenu u x. Prebacivanje ostatka deljenja
u
       sub dx,dx              ;registar ax i deljenje tog ostatka sa 100
       div sto                 ;da bi dobili cifru stotina,koju smestamo
u
       mov y,al               ;promenljivu y.

       mov ax,dx              ;Dobijanje cifre desetica na identican na-
       sub dx,dx              ;cin i njeno smestanje u promenljivu z.
       div des
       mov z,al
       mov w,dl               ;I ostatak deljenja kao broj jedinica sme-
                              ;sten u promenljivu w.
       mov dl,smer               ;Krecemo sa ispisom brzine na monitor.
Znak
       add dl,30H              ;broja, tj. njegov ASCII kod, smestamo u
dl
       mov ah,02H             ;i pomocu funkcije 02H (Character output)
       int 21H                ;interrupt-a 21H ispisujemo.

       mov dl,x                  ;Cifru hiljada ubacujemo u registar dl,
do-
       add dl,30H             ;dajemo toj vrednosti broj 30H i dobijamo
       mov ah,02H             ;ASCII vrednost te cifre.Karakteri od 0 do
       int 21H                   ;9 imaju ASCII vrednosti od 30H do 39H,
ko-
                              ;je se sada nalaze u registru dl i na taj
       mov dl,y               ;nacin se pomocu funkcije 02H interrupt-a
       add dl,30H             ;21H ispisuju na monitor. Istu logiku
       mov ah,02H             ;primenjujemo i za ispis cifara stotina,
       int 21H                ;desetica i jedinica.

       mov dl,z
       add dl,30H
       mov ah,02H
       int 21H

       mov dl,w
       add dl,30H
```

```
      mov ah,02H
      int 21H
                              ;Postavljamo 0 u promenljivu smer radi po-
      mov  smer,0                   ;novnog odredjivanja znaka brzine i
vracamo
      jmp petlja              ;se na pocetak petlje.
;*****************************************************************
***
DIALOG:call novired           ;Pozivanje potprograma 'novired'.
      lea dx,izlazak           ;Smestanje offset-a promenljive 'izlazak'
u
      mov ah,09H                ;registar dx koji predstavlja pokazivac
ni-
      int 21H                      ;ske karaktera za funkciju 09H koja
ispisu-
                              ;je tu nisku na monitor.
      mov ah,01H              ;Funkcija 01H cita karakter sa tastature i
      int 21H                  ;prikazuje ga na ekranu, smestajuci ga u
al.
      cmp al,79H                ;Uporedjujemo uneti znak sa ASCII kodom
slo-
      je exit                   ;va Y i ukoliko je identican skacemo na
pot-
      cmp al,59H              ;program 'exit' koji nam omogucuje izlazak
      je exit                     ;iz programa.Proveru vrsimo i za malo
slovo
      call novired            ;y (59H). Sa ovim obezbecujemo prelazak u
      lea dx,promena          ;novi red. Ponavljamo operaciju za prome-
      mov ah,09H              ;nljivu 'promena' kojom se obezbecuje pro-
      int 21H                 ;mena zadate brzine na taj nacin sto ce se
      mov ah,01H                 ;potvrdnim odgovorom, tj. pritiskom na
tas-
      int 21H                 ;ter y, preci na potprogram 'unos' koji
      cmp  al,79H                    ;obezbecuje unosenje podataka sa
tastature.
      jne  ispit                      ;Postupak je identican prelasku
napotprogram
      call unos               ;'exit'. Pozivanje potprograma 'unos'.
ispit:cmp al,59H
      jne kraj
      call unos
kraj: call novired
      jmp petlja              ;Povratak u petlju.
;*****************************************************************
***
EXIT: cli                     ;Izlazak iz programa na taj nacin sto se
      mov al,36H              ;sistemski timer vraca na generisanje
      out 43H,al              ;interrupt-a 08H na svakih 55 ms punjenjem
      mov al,0FFH             ;njegovog brojaca 0 na adresi 40H jedini-
      out 40H,al              ;cama.
      out 40H,al
      sti

      mov dx,stari_off        ;Pored toga potrebno je i vratiti origina-
      mov  bx,stari_seg             ;lnu  rutinu  interrupt-u  1CH  pomocu
funkcije
      mov ds,bx                ;25H. Segment i offset originalne rutine
su
      mov ah,25H              ;bili sacuvani u promenljivim 'stari_seg'
i
      mov al,1CH              ;'stari_off'.
      int 21H

      mov ah,4CH                  ;Izlazak iz programa i povratak u DOS
pomo-
```

```
      int 21H                 ;cu funkcije 4CH interrupt-a 21H.
;*************************************************************************
***
UNOS: call novired                ;Potprogram koji omogucuje unosenje
zeljene
      lea dx,zeljena            ;brzine sa tastature. Ispis promenljive
'ze-
      mov ah,09H              ;ljena' na vec opisani nacin.
      int 21H
      lea dx,zadata           ;Unosenje karaktera sa tastature i smesta-
      mov ah,0AH              ;nje u promenljivu 'zadata'ciji je offset
      int 21H                 ;prebacen u registar dx koji sluzi kao po-
      add dx,2                    ;kazivac funkciji 0AH interrupt-a 21H.
Broj
      mov bx,dx                   ;unesen sa tastature predstavlja nisku
zna-
      mov ax,[bx]             ;kova,a ne stvarnu vrednost pa je potrebno
      cmp al,2DH              ;izvrsiti obradu niske. To radimo tako sto
      jne znaci               ;uzimamo po dva karaktera smestena u ah i
      mov znak,1                ;al,od njih oduzimamo po 30H i dobijamo
ci-
znaci:add bx,2                ;fre, prvo hiljada i stotina, a potom i
      mov ax,[bx]             ;desetica i jedinica.
      sub ah,30H
      sub al,30H
      xchg ah,al                    ;Mesta cifara hiljada i stotina su
zamenjena
      aad                     ;a onda je izvrseno njihovo  prilagodjenje
      mul sto                   ;binarnom brojnom sistemu i mnozenje sa
100.
      mov provera,ax          ;Dobijeni rezultat se cuva u promenljivoj.
      add bx,2                  ;Ponavljamo postupak za dobijanje cifara
de-
      mov ax,[bx]             ;setica i jedinica.
      sub ah,30H
      sub al,30H
      xchg ah,al
      aad                      ;Dobijamo vrednost za desetice i jedinice
i
      add provera,ax            ;sabiramo je sa promenljivom 'provera' u
ko-
      call novired              ;joj se sada nalazi brojna vrednost koja
od-
      cmp provera,0             ;odgovara zadatoj brzini. Vrsimo proveru
da
      jl greska                    ;li je zadata brzina u predvidjenom
opsegu.
      cmp provera,1390        ;Ukoliko nije skacemo na label-u 'greska',
      jg greska                     ;a ukoliko jeste  smestamo je u
promenljivu
      cmp znak,0              ;'zadbrz' sa kojom dalje radimo.
      je pozit
      neg provera
      mov znak,0
pozit:mov ax,provera
      mov zadbrz,ax
      jmp skoci
greska:lea dx,upozori         ;Izbacujemo upozorenje na ekran o pogresno
      mov ah,09H              ;unetoj brzini i vracamo se na pocetak
      int 21H                 ;'unos'-a.
      jmp unos
skoci:ret                          ;Povratak na mesto sa koga je potpr.
pozvan
;*************************************************************************
***
```

```
NOVIRED:mov dl,0AH              ;Potprogram koji obezbecuje prelazak u
novi
      mov ah,02H                    ;red.Postavljajuci 0AH u registar
dl,funkci-
      int 21H                     ;ja 02H ce popuniti tekuci red (Line
Feed),
      mov dl,0DH              ;a sa 0DH u dl izvrsiti povratak cursor-a
      mov ah,02H              ;na pocetak, sada sledeceg reda (Carriage
      int 21H                 ;Return).
      ret                     ;Povratak.

      prim endp               ;Zavrsetak procedure 'prim'.
;*************************************************************************
***
      prenos proc                   ;Prekidna rutina koja se dodeljuje
interrupt
                              ;1CH umesto originalne.
prekid:push ax                    ;Smestanje sadrzaja registara koji se
kori-
      push dx                     ;ste u ovoj prekidnoj rutini na stek.
Vred-
      push ds                       ;nosti u registrima koje su bile u
trenutku
      push bx                 ;nastajanja interrupt-a bice sacuvane na
      push cx                 ;steku po LIFO principu.

      mov ax,data
      mov ds,ax
;*************************************************************************
***
BROJAC:mov dx,302H            ;Izbacivanje jedinica na port A prve PPI-e
      mov al,0FFH                  ;koji se nalazi na adresi 302H,radi
lecova-
      out dx,al               ;nja stanja brojaca.
      mov dx,308H             ;Punjenje porta A (adresa 308H) i porta B
      in al,dx                ;(adresa 309H) trece PPI-e sa trenutnim
      mov ah,al               ;stanjem brojaca.Signal koji stize sa
      mov dx,309H             ;encoder-a.
      in al,dx
;*************************************************************************
***
BRZROT:mov bx,ax                   ;Trenutno stanje brojaca se nalazi u
regist-
      sub ax,zapis            ;ru ax, a prethodno stanje je zapamceno u
      cmp ax,1250             ;promenljivoj 'zapis'. Potrebno je dobiti
      jl napre                  ;ugao za koji se promeni polozaj rotora
iz-
      sub ax,2500             ;mecu dva interrupt-a kao razliku ova dva
      jmp pravo                   ;broja.Potrebno je izvrsiti korekciju
dobi-
napre:cmp ax,-1250              ;jenog rezultata u slucaju da je brojac
bio
      jg pravo                      ;setovan ili resetovan izmecu dva
ocitavanja
      add ax,2500             ;Dobijena razlika se smesta u promenljivu
pravo:mov zapis,bx            ;'terot'.
      mov terot,ax
      add suma,ax                ;Kako se brzina ne racuna tokom svakog
prola-
      cmp tmp,0                   ;ska kroz prekidnu rutinu zbog velike
ucesta-
      ja presko                  ;nosti interrupt-a,potrebno je pomeraje
racu-
      mov tmp,24                  ;nate u svakom interrupt-u sabirati i
obradi-
```

```
      mov ax,suma               ;ti svake 12-e ms.Rezultat se upisuje u
pro-
      mov bx,2                  ;menljivu 'brzina', a promenljiva 'suma'
se
      imul bx                 ;postavi na 0.
      mov brzina,ax
      mov suma,0
;*************************************************************************
***
PIREG:mov ax,zadbrz             ;Realizacija brzinskog PI regulatora.Od
ula-
      sub ax,brzina           ;zne reference 'zadbrz' oduzimamo izmerenu
      cmp ax,0                  ;vrednost 'brzina'. Ubacivanjem jedinica
ili
      jge razlic                  ;nula u dx registar zavisno od znaka
razlike,
      mov dx,0FFFFH               ;obezbecujemo korektnost deljenja sa
znakom.
      jmp pocni                ;Deljenje sa Kint =10 je ustvari mnozenje
sa
razlic:mov dx,0000H            ;stvarnim integralnim koeficijentom koji
iz-
pocni:idiv Kint               ;nosi 0.1 .Dobijeni kolicnik se oduzima od
      sub iqzad,ax            ;struje Iq koja odrecuje vrednost momenta.
      mov ax,stara               ;Proporcionalni deo regulatora koji je
ovde
      sub ax,brzina              ;realizovan u inkrementalnom obliku.Od
pret-
      cmp ax,0                     ;hodno sracunate brzine, smestene u
promen-
      jl ceta                       ;ljivoj 'stara',oduzimamo trenutnu
vrednost
      mov dx,0000H              ;brzine i mnozimo je sa proporcionalnim
koe-
      jmp sigma               ;ficijentom koji iznosi 0.125 . Dobijenu
ceta: mov dx,0FFFFH              ;vrednost oduzimamo od Iq, a trenutnu
brzinu
sigma:idiv Kprop              ;smestamo u promenljivu 'stara'.
      sub iqzad,ax
      mov ax,brzina
      mov stara,ax
      cmp iqzad,-992           ;Proveravamo da li je sracunata struja Iq
u
      jg gama                         ;predvidjenim granicama nametnutim
ogranice-
      mov iqzad,-992          ;nim mogucnostima D/A konvertora. Ukoliko
gama:  cmp  iqzad,992                ;njena vrednost prelazi dozvoljeni
maksimum,
      jl presko               ;dodeljujemo joj limitiranu vrednost.
      mov iqzad,992
presko:dec tmp
      mov al,00H              ;Vrsimo odmrzavanje stanja brojaca.
      mov dx,302h
      out dx,al
;*************************************************************************
***
UGAO: mov ax,iqzad            ;Trazimo trenutni polozaj rotorskog fluksa
      mov bx,19               ;u odnosu na alfa osu stacionarnog koordi-
      imul bx                 ;natnog sistema. Prvo racunamo ugao kliza-
      mov bx,2467             ;nja na osnovu trenutne vrednosti struje
      imul bx                 ;Iq i konstantnih vrednosti Tr i Id prila-
      mov teslow,ax           ;godjenih u konstanti 19x2467 i 32-o bitni
      mov teshigh,dx          ;rezultat smestamo u promenljive.
      mov ax,terot            ;Racunamo ugao rotora mnozeci broj impulsa
      mov bx,199              ;koji su stigli sa encoder-a izmedju doga-
```

```
      imul bx                 ;djanja dva interrupt-a sa konstantom 199x
      mov bx,8633             ;8633. Dobijenom broju dodajemo vrednost
      imul bx                 ;sracunatu za ugao klizanja. Primenjeno je
      sub ax,teslow              ;oduzimanje i oduzimanje sa pozajmicom
koje
      sbb dx,teshigh          ;podrzava logiku pozitivnih i negativnih
      sub inklow,ax           ;brojeva. Inkrementirajuci prethodnu vred-
      sbb inkhigh,dx          ;nost za ugao fluksa sa novodobijenim bro-
      mov ax,inkhigh          ;jem, dobijamo ugao 'teta_e' kao 8 gornjih
      mov teta_e,ah           ;cifara krajnje izracunate vrednosti.
;*************************************************************************
***
STRUJE:mov ax,2               ;Racunanje izlaznih struja Ialfa i Ibeta.
      mul teta_e                ;Iz tabele 'sinbrz' uvrstene na pocetku
da-
      mov bx,ax               ;ta segment-a uzimamo vrednosti za sinus i
      mov ax,[bx]             ;kosinus ugla rotorskog fluksa 'teta_e',
      mov sinus,ax               ;potrebnih za transformaciju iz d-q u
alfa-
      imul iqzad               ;beta koordinatni sistem. Transformacija
se
      mov trenlow,ax          ;vrsi po jednacinama:
      mov  trenhigh,dx                  ;Ialfa  =idzad*sin(teta_e)  -
iqzad*cos(teta_e)
      add  teta_e,64                       ;Ibeta  =idzad*cos(teta_e)
+iqzad*sin(teta_e)
      cmp teta_e,255          ;i ovde su one realizivane.
      jbe delta
      sub teta_e,255
delta:mov ax,2
      mul teta_e
      mov bx,ax
      mov ax,[bx]
      mov cosin,ax
      imul idzad
      sub ax,trenlow          ;Izracunata vrednost za Ialfa kao 32-bitna
      sbb dx,trenhigh          ;informacija se sada nalazi u registrima
ax
      mov bx,31493            ;i dx. Potrebno je tu vrednost prilagoditi
      idiv bx                    ;mogucnostima ispisa na DAC-u, na taj
nacin
      cmp ax,0                    ;sto je delimo sa brojem 31493. Radi
dobija-
      jl alfneg                      ;nja tacnije informacije, vrsimo
zaokruziva-
      cmp dx,15746               ;nje dobijenog broja. Ostatak koji se
nalazi
      jb ispalf               ;u registru dx uporedjujemo sa polovinom
      inc ax                      ;vrednosti delioca i ukoliko je veci
inkre-
      jmp ispalf              ;mentiramo izlazni rezultat. Potrebno je
alfneg:neg dx                        ;predvideti i dobijanje negativnog
ostatka,
      cmp dx,15746            ;njegovo invertovanje i u slucaju potrebe
      jb ispalf               ;sada dekrementiranje izlazne vrednosti.
      dec ax
ispalf:add ax,128                ;Dobijenoj vrednosti dodajemo 128 radi
pri-
      mov dx,304H             ;lagodjavanja ispisa informacije na DAC-u
      out dx,al               ;(od 0 do 255) i vrednost za struju Ialfa
      mov ax,idzad            ;izbacujemo na port A druge PPI-e, koji se
      imul sinus              ;nalazi na adresi 304H. Celokupni postupak
      mov trenlow,ax          ;ponavljamo za izracunavanje struje Ibeta.
      mov trenhigh,dx
      mov ax,iqzad
```

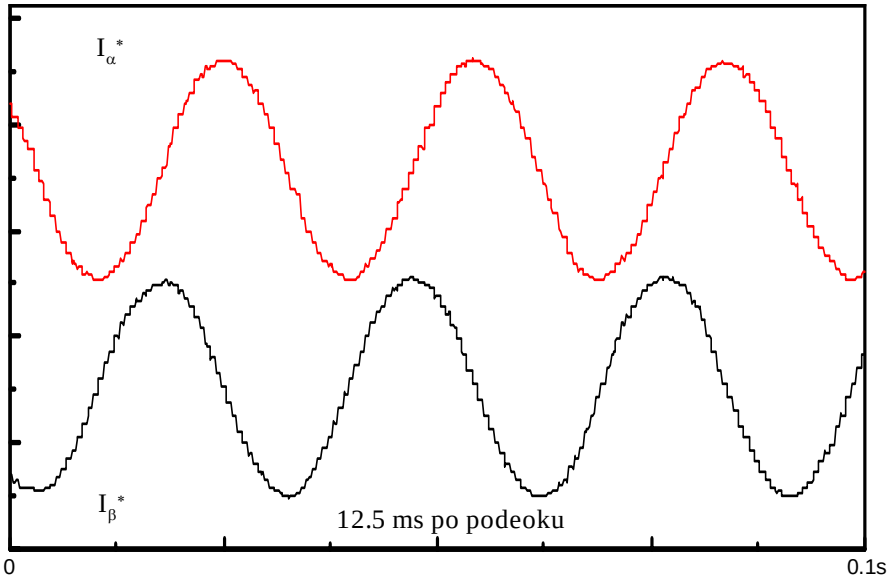
```
      imul cosin
      add ax,trenlow          ;U registrima ax (donjih) i dx (gornjih 16
      adc dx,trenhigh         ;bita) se nalazi 32-bitna informacija o
      mov bx,31493            ;struji Ibeta, koju prilagodjavamo ispisu
      idiv bx                 ;na D/A konvertoru.
      cmp ax,0
      jl betneg
      cmp dx,15746              ;Vrsimo zaokruzivanje dobijenog broja za
po-
      jb ispbet               ;zitivne vrednosti.
      inc ax
      jmp ispbet
betneg:neg dx                      ;Zaokruzivanje dobijene vrednosti za
negati-
      cmp dx,15746            ;van ostatak.
      jb ispbet
      dec ax
ispbet:add ax,128                ;Izracunatu vrednost za struju Ibeta
izbacu-
      mov dx,305H               ;jemo na port B druge PPI-e (na adresi
305H)
      out dx,al               ;koji prosledjuje tu informaciju na DAC.
;*************************************************************************
***
EKRAN:mov ax,brzina               ;Na ovaj nacin omogucujemo pracenje
prelazne
      add ax,1400             ;pojave na osciloskopu preko dodatnih D/A
      mov bl,11                  ;konvertora. Vrsimo prilagodjavanje
vredno-
      div bl                  ;sti brzine zahtevima za upis podataka na
      mov dx,306H                ;DAC-u i njegovom naponskom opsegu.
Dobijenu
      out dx,al                ;vrednost saljemo na port C druge PPI-e
koja
                              ;se nalazi na adresi 306H i direktno je
po-
      mov ax,iqzad            ;vezana sa D/A konvertorom. Na isti nacin
      add ax,992                ;pratimo i moment,koji je srazmeran sa
stru-
      mov bl,8                     ;jom iqzad, samo preko drugog D/A
konvertora
      div bl                  ;koji je povezan sa portom C trece PPI-e i
      mov dx,30AH             ;nalazi se na adresi 30AH.
      out dx,al
;*************************************************************************
***
IZLAZ:pop cx                  ;Po zavrsetku prekidne rutine i povratka u
      pop bx                     ;glavnu petlju, potrebno je vratiti
sadrza-
      pop ds                      ;je registara koji su korisceni u
interrupt
      pop dx                    ;na vrednosti koje su bile u njima u
trenu-
      pop ax                  ;tku nastajanja interrupt-a.
                              ;Povratak u glavnu petlju na mesto gde se
      iret                     ;nalazio u trenutku nastajanja interrupt-
a.
;*************************************************************************
***
      prenos endp             ;Kraj procedure 'prenos'.

      prog ends               ;Kraj programskog segmenta.
      end prim                ;Kraj programa.
;*************************************************************************
***
```

5. EKSPERIMENTALNI REZULTATI

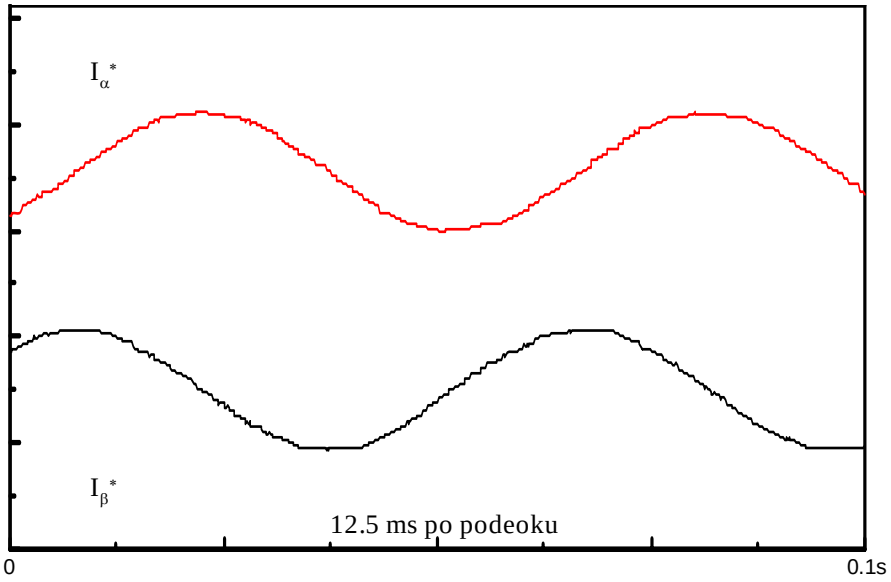
U ovom poglavlju prikazani su eksperimentalni rezultati izloženi u vidu osciloskopskih snimaka talasnih oblika snimljenih u toku prelaznih procesa i stacionarnog rada pogona. Eksperimenti su izvedeni na laboratorijskom u kome je korišæen standardni niskonaponski èetvoropolni asinhroni motor snage 750 W. Promenljive su posmatrane na izlazima operacionih pojaèavaèa D/A konvertora koji se nalaze u okviru opisanog hardvera u treæem poglavlju, kao i na D/A konvertorima ugraðenim upravo za ovu namenu. Talasni oblici su snimani pomoæu digitalnog osciloskopa VC 6020, proizvodnje *HITACHI* i prebaèeni u raèunar preko *GPIB* interfejsa pomoæu programskog paketa *TVOSC*. Obrada podataka izvršena je u programu *MicoCal Origin,* verzija 3.01.

Snimak referentnih vrednosti statorskih struja u stacionarnom stanju je dat na slikama 5.1., 5.2. i 5.3. Snimci su dati u identiènoj razmeri i sa istim vremenskim bazama za razlièite ugaone brzine rotora sa kojih se može videti razlika u promeni uèestanosti i amplitudi struja.

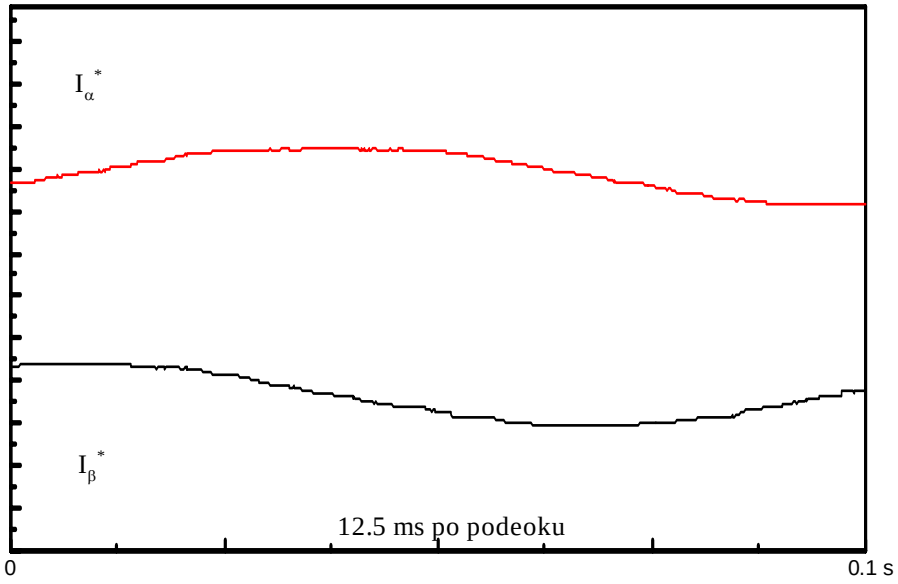
Na slikama 5.4. do 5.7. prikazani su talasni oblici brzina pri razlièitim promenama referentne vrednosti. Uz svaku od slika dato je objašnjenje o prelaznom procesu i primenjenoj razmeri.



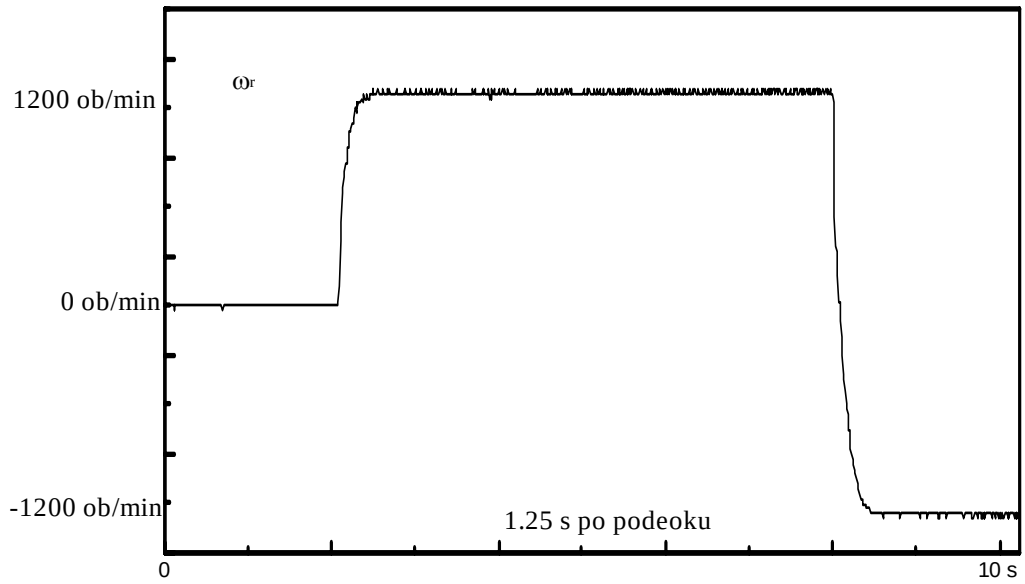
Sl. 5.1. Osciloskopski snimak struja I_α^* (gornji trag) i I_β^* (donji trag) pri ugaonoj brzini rotora od 1350 ob/min. Vertikalna razmera odgovara naponskom opsegu D/A konvertora.



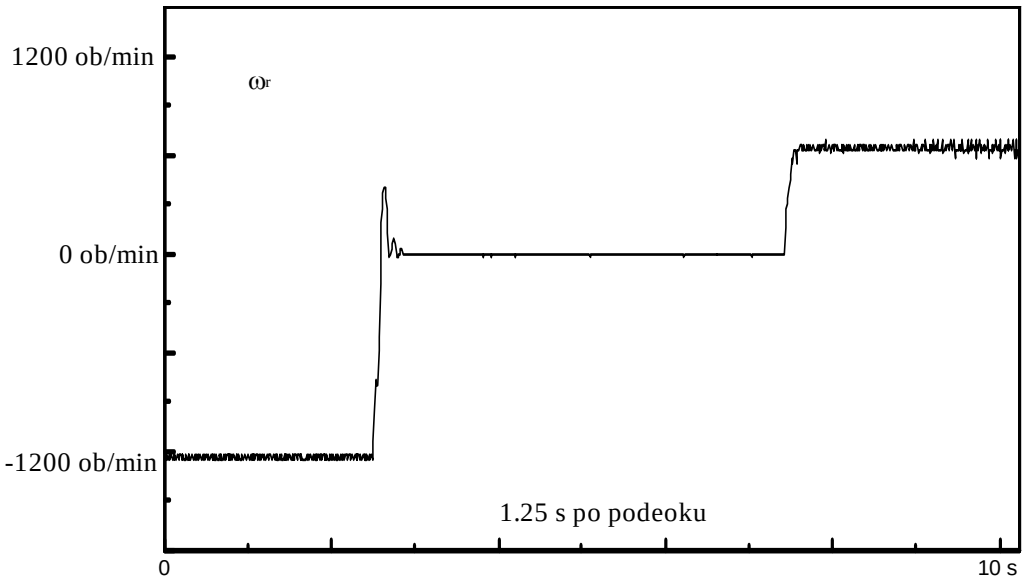
Sl. 5.2. Osciloskopski snimak struja I_α^* (gornji trag) i I_β^* (donji trag) pri ugaonoj brzini rotora od 600 ob/min. Vertikalna razmera odgovara naponskom opsegu D/A konvertora.



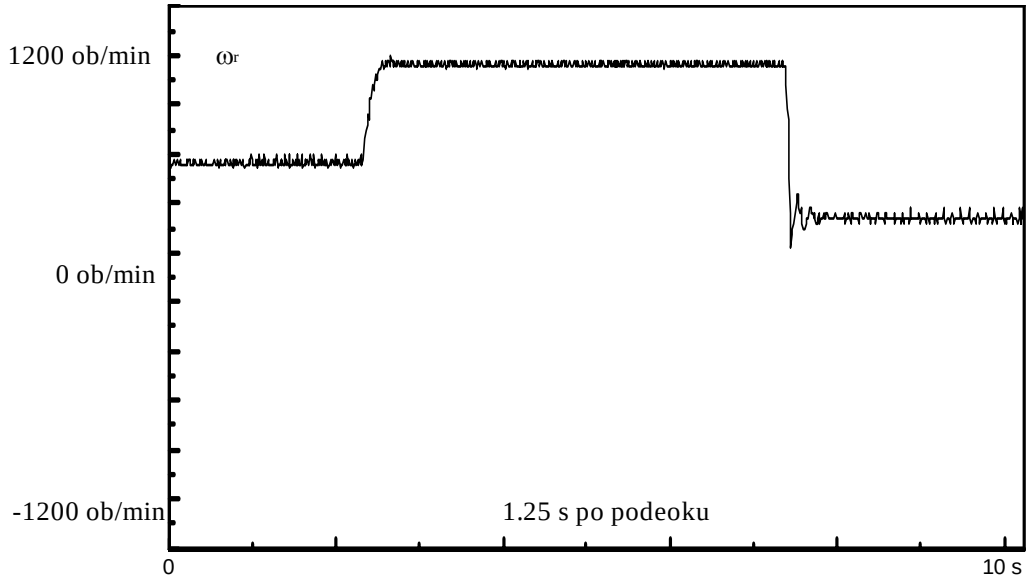
Sl. 5.3. Osciloskopski snimak struja I_α^* (gornji trag) i I_β^* (donji trag) pri ugaonoj brzini rotora od 300 ob/min. Vertikalna razmera odgovara naponskom opsegu D/A konvertora.



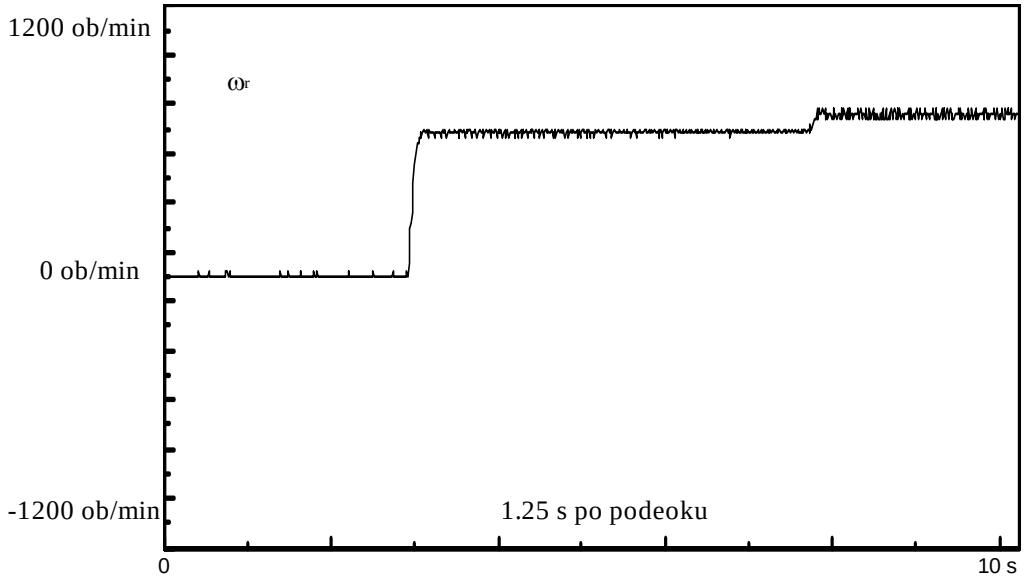
Sl. 5.4. Osciloskopski snimak ugaone brzine rotora pri promeni komandovane brzine sa 0 ob/min na 1350 ob/min, a zatim na -1350 ob/min



Sl. 5.5. Osciloskopski snimak ugaone brzine rotora pri promeni komandovane brzine sa -1350 ob/min na 0 ob/min, a zatim na 600 ob/min



Sl. 5.6. Osciloskopski snimak ugaone brzine rotora pri promeni komandovane brzine sa 600 ob/min na 1200 ob/min, a zatim na 300 ob/min



Sl. 5.7. Osciloskopski snimak ugaone brzine rotora pri promeni komandovane brzine sa 0 ob/min na 800 ob/min, a zatim na 900 ob/min

6. ZAKLJUÈAK

Ovim diplomskim radom je zapoèeto formiranje novog laboratorijskog radnog mesta za ispitivanje regulacije elektromotornih pogona sa asinhronim motorom. Na osnovu postojeæeg hardvera, koji je detaljno proveren i dokumentovan, a zatim povezan sa mikroprocesorom, napisan je softver kojim je izvedena indirektna vektorska regulacija asinhronog motora. Raspregnutim upravljanjem momentom i fluksom ostvarenim na osnovu poznatih vrednosti struje statora i mehaniène brzine rotora, uz priložene rezultate eksperimenata, pokazano je da uz sve navedene prednosti asinhronih motora nad motorima za jednosmerne struje moguæe je ostvariti i adekvatne regulacione karakteristike. Daljim radom na ovom projektu, gde bi bila obuhvaæena zavisnost parametara motora od frekvencije, temperature i zasiæenja magnetnog kola, regulacione karakteristike ovog pogona bi bile poboljšane.

Pisanje softverske osnove u asemblerskom jeziku je doneklo otežalo izradu ovoga rada, ali se pokazalo kao neophodno zbog postojanja ulazno-izlaznih rutina zahtevane efikasnosti, vremenskog ogranièenja trajanja prekidne rutine usled takta generisanja hardverskog interapta i samim tim potrebe za skraæenjem kritiènih sekcija. Komunikacija sa korisnikom i prikazivanje rezultata na ekranu takoðe su uraðeni u asembleru, što je bilo moguæe uraditi i u nekom drugom programskom jeziku (npr. *Microsoft C 5.0*).

Šeme koje su sastavni deo ovog rada su uraðene u programskom paketu *OrCad*, verzija 4.10, predstavljene u hijerarhijski struktuiranom sistemu i nalaze se na disketi koja je priložena uz ovaj rad u direktorijumu SHEME. Na istoj disketi nalaze se i ostali programi izloženi u ovom radu, uključujuæi i tabelu SINBRZ.INC neophodnu za rad glavnog programa.

Potrebno je napomenuti da nastavak na razvoju ove radne stanice nije ogranièen samo na ispitivanja vektorske regulacije asinhronih motora, zahvaljujuæi tako realizovanoj hardverskoj osnovi koja omoguæava i ispitivanja ostalih naèina upravljanja elektromotornim pogonima.

Gotovi seminarski, maturski, maturalni i diplomski radovi iz raznih oblasti, lektire , puškice, tutorijali, referati - specijalizovan tim za usluge visokokvalitetnog pisanja, istraživanja i obradu teksta za kompletan region Balkana.

Posetite nas na sajtovima ispod:

WWW.MATURSKIRADOVI.NET

WWW.SEMINARSKIRAD.ORG

WWW.MATURSKI.NET

WWW.MATURSKI.ORG

WWW.SEMINARSKIRAD.INFO

Dostupni smo Vam 24h 365 dana u godini.

Za gotove verzije rada obratiti se na mail:

maturskiradovi.net@gmail.com

061/ 11-00-105

Seminarski, diplomski, maturski radovi, prevodi na engleski i eseji…

.