PRIMENA MATLAB C KÔD GENERATORA U ELEKTROMOTORNIM POGONIMA BAZIRANIM NA DSP SA NEPOKRETNIM ZAREZOM

Darko Marčetić, Evgenije Adžić, *Fakultet tehničkih nauka, Novi Sad*
Vladan Jevremović , *Parker SSD Drives, Littlehampton, United Kingdom*

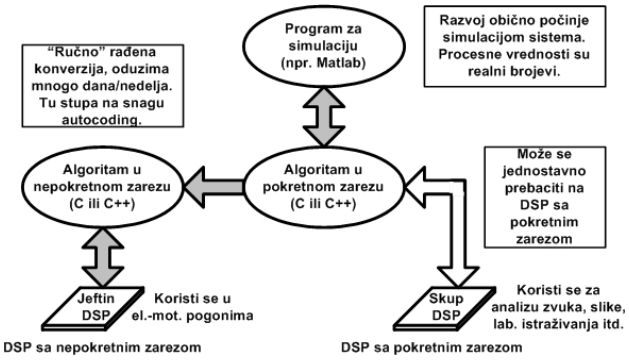
Sadržaj – *U radu je opisan jedan način ubrzanog razvoja programa za digitalni signal procesor (DSP) namenjen radu u okviru elektromotornog pogona. Uvećana konkurencija na tržištu elektromotornih pogona opšte namene, kao i brze promene sigurnosnih normativa, kao i onih vezanih za zaštitu okoline u kojoj uređaj radi, dovode do drastičnog smanjenja dozvoljenog vremena od nastanka ideje do plasmana na tržište (time to market). Sa druge strane, algoritmi upravljanja za elektromotorne pogone obuhvataju sve više zahteva, odgovarajući programski kôd postaje numerički veoma intenzivan i neophodno ga je dugotrajno razvijati i proveravati. Dosadašnji način razvoja kôda, u asembleru ili C jeziku, postaje nedovoljno brz i usled toga neprihvatljiv. Jedno od mogućih rešenja je primena kôd generatora optimizovanog za rad na procesorima skromnih karakteristika (low cost), koji generiše optimalan kôd kako po pitanju procesorskog vremena, tako i po pitanju memorije. Za konkretan primer je korišćen DSP TI F2812 koji operiše sa brojevima u nepokretnom zarezu (fixed-point). Na izabranom DSP-u je eksperimentalno testirana mogućnost izvršenja kôda generisanog na osnovu Matlab modela, koji se izvršava u realnom vremenu (20 kHz). Radi predstavljanja rezultata integracije Matlab kôda u C kôd, implementiran je model SVPWM (Space Vector PWM) modulatora. Matlab model je takođe realizovan u aritmetici sa nepokretnim zarezom, radi prilagođenja korišćenom DSP procesoru.*

1. UVOD

Osnovni kvalitet pogona namenjenog aparatima za domaćinstvo je uvek bila konkurentna proizvodna cena, koja bi uz zadovoljavajuću robustnost održavala pogon na tržištu. Uz očekivanje da će pogon najmanje pet ili duže godina biti prisutan na tržištu sama cena razvoja pogona uglavnom nije uzimana u obzir. Danas se ovo tržište intenzivno menja, konkurencija postaje sve oštrija, učestalo se menjaju regulative i specifikacije koje se postavljaju pred ove pogone. Ovim se radni vek pogona, tačnije vreme prisutnosti na tržištu, uglavnom smanjuje na tri ili manje godina. Sada više nije moguće zanemariti cenu razvoja pogona koja lako dostiže i nekoliko miliona dolara.

Jedna od najvećih stavki pri razvoju novog pogona opšte namene je razvoj programa (software). Razvoj software-a se u startu usporava u iščekivanju novog, jeftinijeg i stabilnijeg hardware-a, zatim raznim *in-house* testovima kojima se podvrgavaju uređaji, kao i sporom komunikacijom između proizvođača i kupca pogona. Vreme do plasmana na tržište se smanjuje i više nije prihvatljivo sporo kodiranje u asembleru, potreban je fleksibilniji pristup. Sa razvojem odličnih C kompajlera za razne DSP i mikroprocesore (TI, Freescale, Analog Device), C jezik potpuno preuzima ovo tržište. Ovim je zadovoljena potreba za brzom izmenom kôda u završnim fazama razvoja, kao i nakon pristizanja rezultata raznih testova na čije se rezultate pogon mora prilagoditi. Ono što i dalje donekle predstavlja problem je razvoj kôda za razne vrste prezentacija (demo) pogona, koje se često ne završavaju pozitivno, jer se žuri sa njihovom pripremom.

Trenutno najbrži način razvoja software-a za elektromotorni pogon je primena raznih kôd generator paketa koji provereni Matlab model [1] prevode u odgovarajući kôd za dSPACE kartice ili razne ciljne (*target*) procesore. Sam dSPACE pristup nije prihvatljiv u ovom slučaju, jer zahteva specifični hardware koji se u potpunosti razlikuje od korišćenog u pogonima opšte namene. Sa druge strane, običan ANSI C generisan kôd može na lak način i uz minimalne izmene da se prevede na razne procesore i da se na taj način ubrza razvoj programa, a ujedno testira i novi hardware. Ovaj pristup nije dosad često korišćen jer je generisani C kôd uglavnom podrazumevao matematiku sa pokretnim zarezom (*float* i *double* brojevi) i nije ga bilo moguće izvršavati u realnom vremenu na većini tzv. *low cost* procesora. Međutim, od skorijeg vremena, moguće je automatski generisati ANSI C kôd u nepokretnom zarezu na osnovu modela razvijenog na Matlab/Simulink platformi. Ovaj proces ilustrovan je na slici 1.



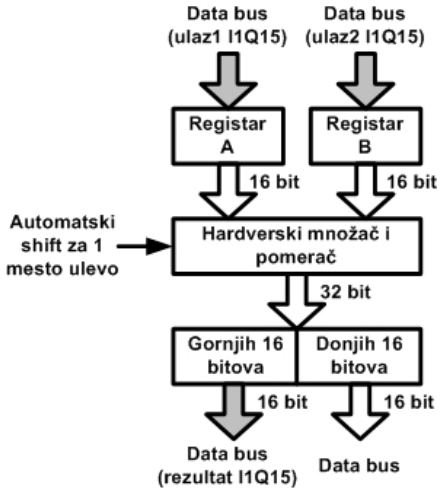
Sl. 1. Savremeni scenario razvoja softvera.

U ovom radu je glavni akcenat dat upravo na generisanje kôda za tzv. *low cost* DSP, koji podržava samo matematiku sa nepokretnim zarezom. Rešenje predviđa da se svi numerički intenzivni delovi programa (modulator, transformacije koordinata, regulatori, estimatori stanja, estimatori parametara, itd.) generišu direktno iz Matlaba, bez intervencije programera, i da se isti izvršavaju jednako brzo i memorijski efikasno kao i optimalno napisan C kôd. Razvoj kôda dat je na primeru SVPWM modulatora [2] za TI DSP F2812 [3].

2. OSNOVE MATEMATIKE SA NEPOKRETNIM ZAREZOM U DSP-u I MATLAB MODELIMA

Model upravljačke strukture objekta kao što je elektromotorni pogon, koju je neophodno izvršavati u realnom vremenu moguće je realizovati primenom sistema normalizovanih vrednosti. Na ovaj način se svi signali koje je neophodno procesirati postavljaju u odgovarajući fiksni opseg koji je moguće predstaviti brojevima sa nepokretnim zarezom. Brojevi sa nepokretnim zarezom zapravo predstavljaju obične celobrojne vrednosti (*integer* tip podataka) o kojima korisnik brine kako će ih interpretirati. Broj u nepokretnom zarezu čini celobrojni i decimalni deo i u zavisnosti od mesta decimalne tačke moguće je predstaviti različite opsege

realnih brojeva. Kako je binarna prezentacija broja zasnovana na stepenu dvojke, sledi da će bitovi nakon zamišljene decimalne tačke označavati "polovine" prethodne pozicije (1/2, 1/4, 1/8 itd.). Na taj način ako su podaci 16-bitni, a decimalni zarez "postavimo" odmah nakon MSB bita (bit najveće važnosti), celobrojna vrednost 2^{16-1}-1 = 32767 označavaće realan broj ~1,0, dok će ceo broj -2^{16-1} = -32768 označavati broj -1,0. Ovakva prezentacija broja označava se obično kao I1Q15 format. Rezolucija realnog broja u ovom formatu $2^{-(16-1)}$ daje dovoljno preciznosti u ovoj primeni. Odmah se uočava da je mana ovakvog načina predstavljanja realnih brojeva opseg brojnih vrednosti, koji se pak može proširiti na račun smanjenja rezolucije. Međutim, dobra osobina brojeva sa nepokretnim zarezom je što se množenjem dva takva broja dobija opet broj u istom formatu, uz odgovarajuću interpretaciju rezultata. Kada DSP množi dva operanda, bilo da su oni celobrojne vrednosti ili brojevi u nepokretnom zarezu, on obavlja identičan posao. Množenjem dva 16-bitna broja u I1Q15 formatu dobija se 32-bitni međurezultat u I2Q30 formatu, tj. rezultat čiji se broj pozicija za celobrojni i razlomljeni deo dobija kao zbir istih kod operanada. Pomeranje za jedno mesto ulevo i izdvajanje viših 16 bitova, dobijamo rezultat u istom I1Q15 formatu. Ovako definisana operacija množenja je potpuno podržana u većini DSP procesora, i izvršava se u jednom ciklusu uz automatsko pomeranje 32-bitnog rezultata za jedan bit ulevo (slika 2).



Sl. 2. Množenje dva broja u nepokretnom zarezu (I1Q15).

Ovim se efektivno izvršava ekvivalentna C naredba za množenje dva broja u I1Q15 formatu:

$$C = (\text{int})(((\text{long})A*(\text{long})B)>>15) \quad (1)$$

Sa druge strane, u Matlabu se modeli uglavnom razvijaju sa aritmetikom nad brojevima u pokretnom zarezu. Uz odgovarajuća podešavanja, moguće je razviti model koji koristi samo signale i blokove za procesiranje brojnih vrednosti u formatu sa nepokretnim zarezom. Ovaj korak je neophodan kako bi generisani C kôd bio optimizovan za rad na DSP procesoru koji po svojoj prirodi procesira podatke u nepokretnom zarezu.

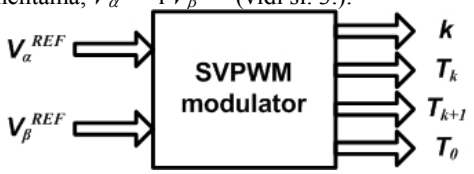
Za signale koje Matlab koristi iz C kôda potrebno je posebno naznačiti tip podataka, jednako ga definisati kao i u C kôdu. Ukoliko se signal dobija na izlazu nekog bloka neophodno je u opcijama koje taj blok nudi naglasiti tip izlaznog podatka (npr. sfix(16) za 16-bitne brojeve u nepokretnom zarezu) i njen format tj. faktor skaliranja (npr. 2^{-15} za brojeve u formatu I1Q15). Ipak, neki Matlab blokovi ne podržavaju signale različite od *double* i ne mogu se koristiti za ovu svrhu.

Na taj način je, uz uložen minimalan napor za definisanje tipova signala u modelu, moguće iskoristiti komfor koji pruža grafičko okruženje Matlab/Simulink za razvoj kompleksnih algoritama upravljanja.

3. MATLAB MODEL SVPWM MODULATORA U NEPOKRETNOM ZAREZU

Kao jedan od numerički najintezivnijih delova kôda za vektorsko upravljanje naizmeničnim pogonima, ističe se SVPWM modulator. Kako se za rad pogona opšte namene zahteva visoka prekidačka frekvencija (iznad čujnog opsega, 16 kHz), na raspolaganju je vrlo kratak vremenski period u okviru kojeg treba izvršiti čitavu kontrolnu petlju. Pogodnost korišćenja automatskog generisanja C kôda koji bi implementirao SVPWM modulator krajnje je očigledna, s obzirom na veliki broj računskih operacija koje algoritam obuhvata. Ipak, kôd generisan na osnovu Matlab modelu mora biti jednako efikasan kao i C kôd, i DSP ga mora izvršiti u okviru dozvoljenog vremena (PWM periode).

Modul SVPWM modulatora generiše prekidačke PWM signale za trofazni invertorski most direktno koristeći alfa i beta komponente željenog izlaznog napona, V_α^{REF} i V_β^{REF}, kao ulazne reference. Kao najčešće korišćena tehnika modulacije za digitalnu strujnu regulaciju, daje punu kontrolu nad d i q komponentama faznog napona na izlazu invertora. Ove d i q komponente napona, definisani na izlazu strujnih regulatora, transformišu se u stacionarni koordinatni sistem, tj. αβ komponente, i dovode na ulaz *space vector* modulatora. Glavni zadatak SVPWM modulatora je da izračuna potrebna vremena "zadržavanja" odabranih vektora u okviru prekidačke periode kako bi se na izlazu dobio "usrednjen" naponski vektor definisan svojim komponentama, V_α^{REF} i V_β^{REF} (vidi sl. 3.).



Sl. 3. SVPWM modulator sa ulazima V_α^{REF} i V_β^{REF}.

Veza između referentnih vrednosti V_α^{REF} i V_β^{REF} i potrebnih vremena zadržavanja aktivnih i nultih vektora V_0-V_7, definisanih različitim stanjima invertorskog mosta, lako se nalazi posmatrajući sliku 4. Pretpostavljajući da referentni vektor V_{REF} leži u sektoru k, i da su njegovi najbliži susedni vektori V_k i V_{k+1}, važi sledeća jednačina:

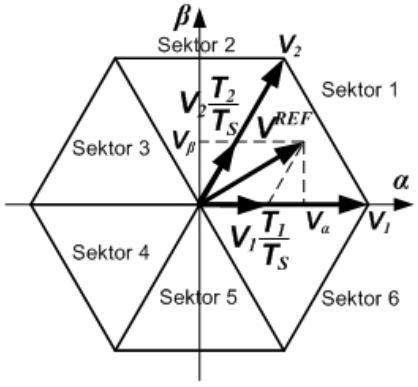
$$\vec{V}_{REF}\frac{T_s}{2} = \vec{V}_k T_k + \vec{V}_{k+1} T_{k+1} \quad (2)$$

gde T_k označava polovinu vremena zadržavanja vektora V_k u okviru prekidačke periode T_s.

Rešavanjem prethodne jednačine i njenim sređivanjem s ciljem da se izbegne jedno dodatno množenje dobijaju se potrebna vremena za vektore:

$$\begin{bmatrix} T_k \\ T_{k+1} \end{bmatrix} = \frac{(T_s/2)}{V_{DC}} \begin{bmatrix} \sqrt{3}\sin(k\frac{\pi}{3}) & -\sqrt{3}\cos(k\frac{\pi}{3}) \\ -\sqrt{3}\sin((k-1)\frac{\pi}{3}) & \sqrt{3}\cos((k-1)\frac{\pi}{3}) \end{bmatrix} \begin{bmatrix} V_\alpha \\ V_\beta \end{bmatrix}$$

$$T_0 = \frac{T_s}{2} - T_k - T_{k+1} \quad (3)$$



Sl. 4. V_{REF} u prvom sektoru generiše se koristeći V_1, V_2 i nulte vektore.

U tom slučaju dobijaju se brojevi u opsegu ±2, zbog čega se mora koristiti I2Q14 format. Da bi se dodatno smanjilo vreme izvršavanja kôda, napravljena je look-up tabela (tabela 1) sa mogućim vrednostima trigonometrijskih funkcija u zavisnosti od sektora u kom se nalazi referentni vektor.

Realizovani modulator obuhvata i slučaj nadmodulacije, kada referentni vektor izlazi van šestougla definisanog osnovnim vektorima invertorskog pretvarača (videti sliku 4). Najbolji način da se detektuje nadmodulacija je da se prati izračunato vreme nultog vektora. Ukoliko nedostaje vremena za oba aktivna vektora za predstavljanje reference V^{REF}, izračunato vreme nultog vektora biće negativno:

$$IF\,(T_0 = \frac{T_s}{2} - T_k - T_{k+1} < 0) \Rightarrow nad\ mod\ ulacija$$

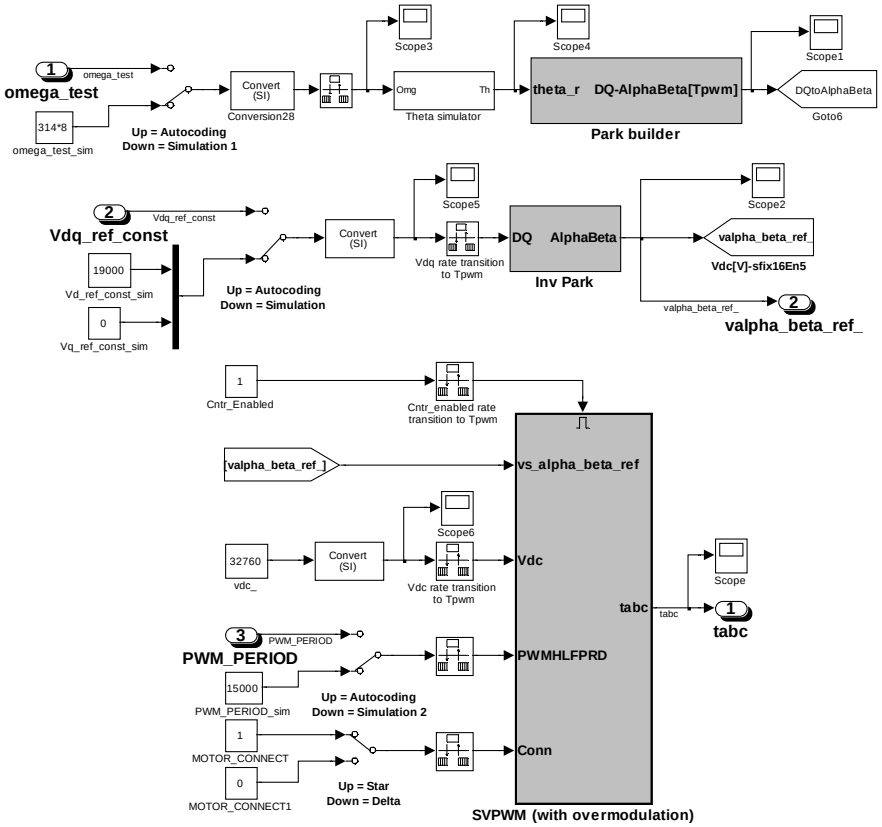
k	$\sin(k\frac{\pi}{3})$	$-\cos(k\frac{\pi}{3})$	$-\sin(\frac{(k-1)\pi}{3})$	$\cos(\frac{(k-1)\pi}{3})$
1	*1.5=24575*	*-0.433=-14188*	*0 =0*	*0.866=28377*
2	*1.5=24575*	*0.433=14188*	*-1.5=-24575*	*0.433=14188*
3	*0*	*0.866=28377*	*-1.5=-24575*	*-0.433=-14188*
4	*-1.5=-24575*	*0.43=14188*	*0*	*-0.866=-28377*
5	*-1.5=-24575*	*-0.43=-14188*	*1.5=24575*	*-0.433=-14188*
6	*0*	*-0.86=-28377*	*1.5=24575*	*0.433=14188*

Tabela 1. Look-up tabela sadrži sve potrebne trigonometrijske rezultate.

Pri detekciji nadmodulacije vremena zadržavanja vektora V_k i V_{k+1} treba iznova izračunati tako da zajedno dele čitavu prekidačku periodu T_s: I ovde treba numerisati jednacine

$$T_k = \frac{T_s}{2}\frac{T_k}{T_k + T_{k+1}}$$
$$T_{k+1} = \frac{T_s}{2}\frac{T_{k+1}}{T_k + T_{k+1}} \qquad (4)$$

Na osnovu prethodnog realizovan je model SVPWM modulatora u Matlab/Simulink-u koji koristi sve signale predstavljene u formatu sa nepokretnim zarezom. Na taj način model je prilagođen konverziji u C kôd za DSP koji podržava samo aritmetiku sa nepokretnim zarezom. Na slici 5 prikazan je model SVPWM-a u Matlab-u. Lako se uočavaju ulazne promenljive, koje glavni program (C-shell) treba da prosledi generisanom Matlab kôdu: *omega_test* (zadavanje frekvencije), *Vdq_ref_const* (niz od dva člana za zadavanje referentnih vrednosti napona po d i q osi) i *PWM_PERIOD* (za zadavanje prekidačke frekvencije PWM signala), kao i

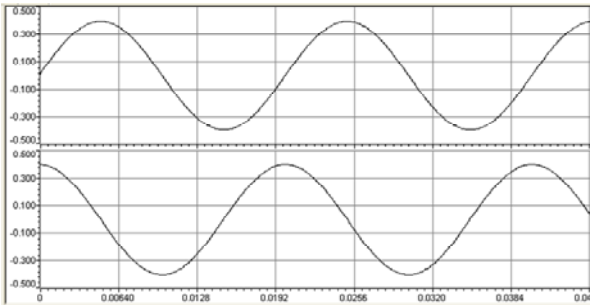


Sl. 5. Model SVPWM-a u Matlabu prilagođen za autocoding i simulaciju.

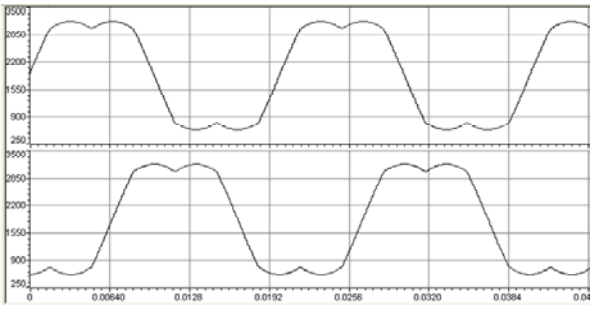
izlazne promenljive koje generisani Matlab kôd nakon računa jednačina modela prosleđuje glavnom programu: *tabc* (vremena punjenja PWM registara, niz od tri 16-bitne vrednosti koje određuju faktor ispune PWM signala) i *valpha_beta_ref_* (samo kao prikaz referentnih vrednosti na ulazu modulatora). Da bi postojala interakcija između Matlab i C kôda potrebno je sve ulazne promenljive u Matlab-u definisati kao *ImportedExtern*, dok klasu čuvanja izlaznih promenljivih treba podesiti na *ExportedGlobal*. Za sve korišćene blokove u Matlabu treba naglasiti tip, odnosno format podataka koje očekuje na ulazu i koji će biti na izlazu (u ovom slučaju to su 16-bitni brojevi sa nepokretnim zarezom u I1Q15 i I2Q14 formatu), s ciljem da se generiše efikasan kôd koji se brzo izvršava. Pre kompajliranja Matlab kôda potrebno je podesiti ciljni hardver za koji se generiše C kôd (u ovom slučaju TI C2000, ert.tlc) kao i ciljni direktorijum za kôd. Matlab automatski generiše C kôd modela, kao i primer glavnog programa koji pokazuje kako treba koristiti generisani kôd.

5. PROVERA GENERISANOG KÔDA - REZULTATI EKSPERIMENATA

Svi eksperimenti su vršeni na razvojnom sistemu ezDSP TI F2812. U razvojnom okruženju Code Composer Studio, u glavnom programu izvršena je inicijalizacija DSP-a, PWM i ADC periferija kao i generisane Matlab strukture, dok je glavna Matlab funkcija koja implementira SVPWM modulator povezana sa prekidom PWM modula, gde se ona periodično poziva (20 kHz).

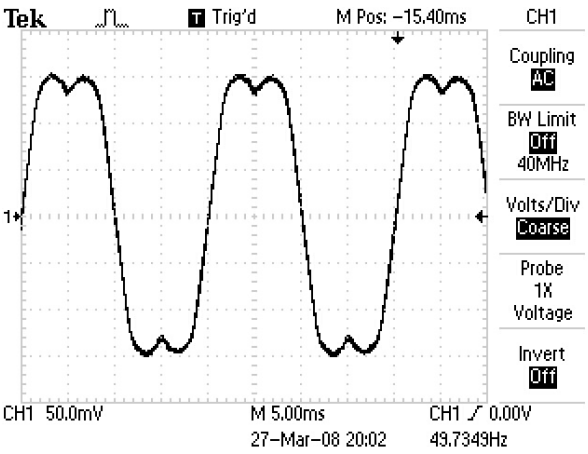


Sl. 6. TI 2812 data log. Ulazi SVPWM modela: reference napona po α β osi.



Sl. 7. TI 2812 data log. Izlazi SVPWM modela: PWM faktori ispune, skalirani u odnosu na PWM_PERIOD.

Na slikama 6 i 7 su prikazani rezultati eksperimenta dobijeni u Code Composer Studio okruženju (koristeći datalog modul), pri čemu su na ulaze SVPWM modulatora dovedene referentne vrednosti V_d^{REF} = 13107 (0,4 r.j.), V_q^{REF} = 0 i zadata je željena frekvencija od 50Hz (*omega_test* = $314*2^3$). Prikazane izlazne promenljive SVPWM modula, t_a i t_b, predstavljaju vrednosti koje direktno treba upisati u DSP PWM registre koji određuju faktor ispune PWM signala. Prekidačka frekvencija je 20 kHz i određena je promenljivom *PWM_PERIOD* = 3750. Može se uočiti da se vrednosti t_a i t_b kreću oko polovine vrednosti promenljive *PWM_PERIOD* (=1875) i da imaju očekivani talasni oblik. Realizovani SVPWM modul testiran je i posmatranjem PWM signala na izlaznim DSP pinovima. Jedan od dobijenih izlaznih PWM signala propušten je kroz niskopropusni filtar prvog reda (R = 1,2 kΩ, C = 220 nF, f_c = 603 Hz) i prikazan na slici 7.



Sl. 7. Rad generisanog kôda u realnom vremenu. PWM izlaz propušten kroz niskopropusni filtar.

6. ZAKLJUČAK

U radu je opisan efikasan način za razvoj DSP programa namenjenog za kontrolu elektromotornog pogona. Predloženi način, generisanje DSP kôda na osnovu Matlab modela sa nepokretnim zarezom, dovodi do kôda koji se efikasno izvršava, ali i efikasno razvija i menja. Ovim se spajaju dva dijametralno suprotna zahteva i može se rešiti problem brzog razvoja optimalnog kôda za pogone opšte namene. Efikasnost generisanog softvera je proverena na *space vector* PWM modulatoru, numerički najintenzivnijem bloku u okviru DSP programa.

LITERATURA

[1] The MathWorks Inc, "*Real-Time Workshop Embedded Coder Developing Embedded Targets*", User guide, Ver. 4.5 (2006b), 2002-2006.

[2] J. Holtz, "Pulse width modulation for electronic power converters," in *Power Electronics and Variable Speed Drives*, B. K. Bose, Ed. Piscataway, NJ: IEEE Press, 1997, pp. 138–208.

[3] Texas Instruments DSP TMS320F2812, data manual.

Abstract – This paper proposes a new efficient way for the motor control software rapid prototype development. The fixed-point Matlab model and corresponding autocoding tools are described. Efficiency of the auto-coded software is tested on TI DSP 2812, executing numerically intensive space vector PWM model software, auto-generated from related fixed–point Matlab block.

APPLICATION OF MATLAB C CODE GENERATOR IN MOTOR DRIVES BASED ON FIXED-POINT DSP

Darko Marčetić, Evgenije Adžić,Vladan Jevremović