

5.6. Mikroprocesorski program za skalarno upravljanje AM

Pri skalarnom upravljanju asinhroni motor se napaja trofaznim naizmeničnim naponom čija su frekvencija i amplituda određeni u skladu sa zadatim vrednostima brzine rotora i fluksa. Osnovni zadatak, generisanje trofaznog PWM signala na osnovu zadate frekvencije i amplitude, vrši trofazni sinusni PWM modulator opisan u poglavlju 5.2. Uz minimalne izmene SVPWM modulator (poglavlje 5.3.) takođe može da generiše trofazni PWM signal na osnovu istih parametara, treba samo koristiti polarne (amplituda i ugao) umesto Dekartovih koordinata vektora referentnog napona. Da bi se skalarno upravljanje kompletiralo, zadatak programa je i da definiše amplitudu i frekvenciju napona potrebne da se ostvare zadata brzina i fluks.

U ovom poglavlju će biti opisan mikroprocesorski program za skalarno upravljanje AM u otvorenoj i u zatvorenoj regulacionoj petlji po brzini i sa trofaznim sinusnim PWM modulatorom. U programu postoje sledeći programski moduli:

1. Programski modul **trofazni PWM modulator** koji svake PWM periode na osnovu zadate amplitude i frekvencije statorskog napona generiše tri odgovarajuća faktora ispunje za tri ulaza PWM modula. PWM modulator je već opisan za DC/AC pretvarač opšte namene u poglavlju 5.2. i ovde se više neće detaljno razmatrati.
2. Programski modul za račun **u/f zakona** na osnovu zadate statorske frekvencije računa potrebnu amplitudu statorskog napona.
3. Programski modul **PI regulatora** brzine koji se koristi u slučaju zatvorene povratne sprege po brzini rotora, slika 5.19.

Uprošćena šema pogona sa AM koja sadrži potrebne hardverske i programske blokove za realizaciju skalarnog upravljanja data je na slici 5.21.

Skalarno upravljanje pogon sadrži sledeće hardverske blokove,

1. energetski pretvarač sa ulaznim ispravljačkim stepenom i izlaznim trofaznim invertorskim stepenom,
2. impulsne pojačavače za poluprovodničke prekidače invertora,
3. prilagodna kola za obradu potrebnih mernih signala,
4. asinhroni motor sa davačem prigradenim na vratilo,

Program za skalarno upravljanje koristi sledeće periferije mikrokontrolera koje su upravo i namenjene upravljanju pretvaračima i pogonima:

1. A/D periferiju za merenje dve linijske struje i napona jednosmernog međukola,
2. specijalnu QEI periferiju za obradu kvadrturnih impulsa sa enkodera, u slučaju AM nije potrebno poznavati apsolutni položaj rotora pa se INDEX signal često ne koristi,
3. PWM periferiju koja na osnovu referentnog vremena vođenja generiše PWM upravljačke signale za tri grane invertora.

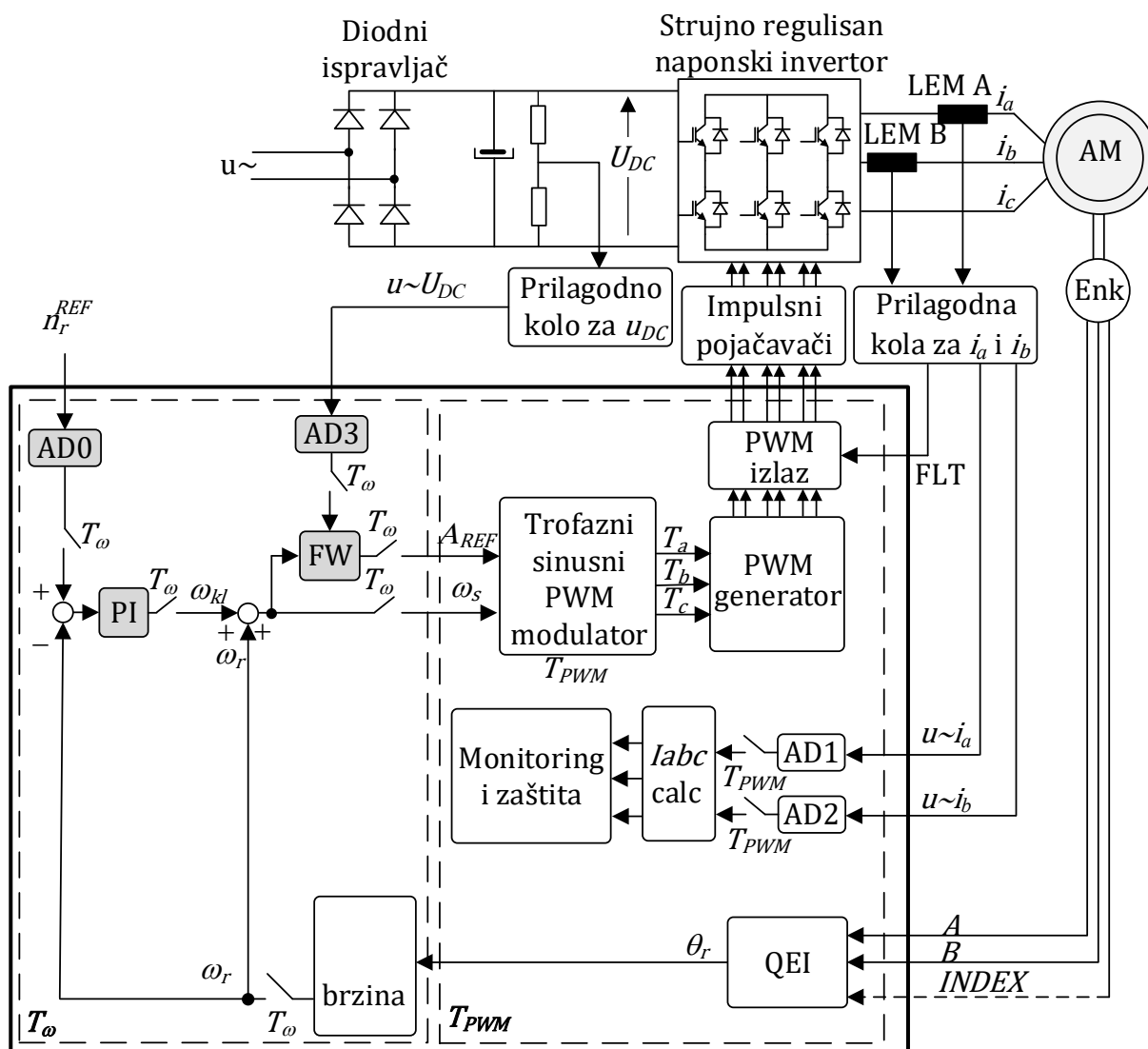
Veći deo skalarnog upravljanja se izvršava sa periodom odabiranja brzinske petlje, T_ω , koja je relativno dugačka (reda veličine 1 ms) i njeno izvršenje ne opterećuje procesor. Akcije koje skalarni kontroler izvršava u periodu T_ω su grupisane na slici 5.21 levo:

1. čitanje referentne brzine, koja se u primeru zadaje kao naponski signal i pretvara u digitalnu komandu brzine preko ulaza A/D konvertora,

2. merenje brzine, tj. u slučaju enkodera njen proračun na osnovu promene položaja u periodu T_ω . Mikrokontrolerska QEI periferija prima impulse sa enkodera i meri promenu položaja rotora u realnom vremenu, bez dodatnog opterećenja procesora, koji treba samo jednom u periodu T_ω da pročita i skalira rezultat,
3. proračun linearnog diskretnog PI regulatora brzine koji na osnovu greške brzine upravlja referentnom učestanošću klizanja,
4. proračun referentne statorske učestanosti primenom jednačine slaganja brzina,
5. proračun referentne amplitude napona statora primenom u/f zakona upravljanja.

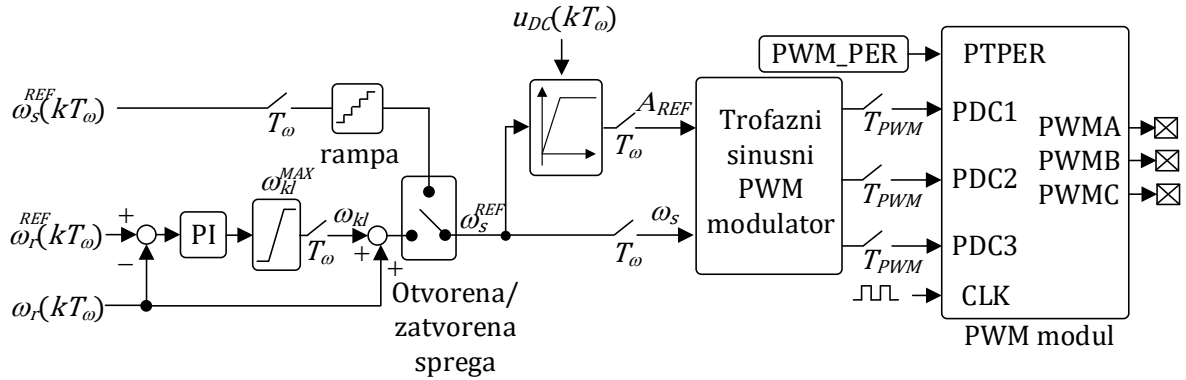
Ovim jednačinama skalarni program, jednom u periodu T_ω , računa nove vrednosti A_{REF} i ω_s .

Manji deo programa za skalarno upravljanje AM, praktično samo diskretne jednačine trofaznog sinusnog PWM modulatora, mora da se izvršava u svakoj PWM periodi. Pošto je PWM perioda relativno kratka (reda veličine 50 μ s), jedino ovaj deo programa može da optereti procesor. Ostale akcije koje su prikazane na slici 5.21 desno da se izvršavaju u svakoj PWM periodi se uglavnom automatski izvršavaju u nekoj od periferija, bez opterećenja procesora.



Slika 5.21. Blok šema skalarnog upravljanja sa korišćenim programskim blokovima

Mikrokontrolerski program za skalarno upravljanje AM menja parametre izlaznog napona invertora, njegovu učestanost i amplitudu, tako da se ostvari referentna brzina rotora, uz nominalni nivo fluksa u I zoni, i uz konstantan napon statora u II zoni. Potrebni blokovi za mikrokontrolersku realizaciju skalarnog upravljanja dati su na slici 5.22.



Slika 5.22. Blokovi mikrokontrolerskog programa za skalarno upravljanje AM

Mikroprocesorski program za skalarno upravljanje AM se bazira na programskom bloku za trofaznu sinusnu PWM modulaciju. Sinusni PWM blok na osnovu zadate statorske učestanosti ω_s i relativne amplitude statorskog napona A_{REF} ($A_{REF} = 1 \rightarrow |u_{AN}| = u_{DC}/2$), u svakom PWM intervalu računa nova vremena uključanja tranzistora za tri grane invertora, u odnosu na periodu PWM signala (PWM_PER), čime su ona spremna za direktni upis u registre PWM modula za faktore ispune PDC1, PDC2 i PDC3. Program za skalarno upravljanje je moguće bazirati i na SVPWM programskom bloku, uz zadavanje amplitude i učestanosti referentnog polifazora napona.

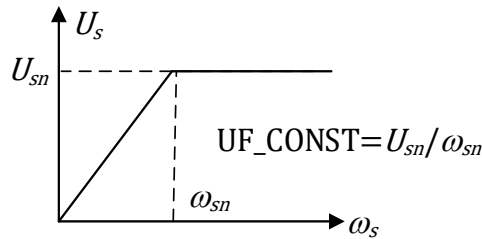
Zadatak ostalog dela skalarnog programa je da trofaznom sinusnom PWM modulatoru jednom u periodu T_ω , koja je uobičajeno mnogo duža od PWM periode (npr. $T_{PWM}=50 \mu s$, $T_\omega=1 ms$), obezbedi nove referentne vrednosti statorskog napona, učestanost ω_s^{REF} i relativnu amplitudu A_{REF} , u skladu sa referentnom vrednošću brzine i potrebnim fluksom. U slučaju otvorene povratne sprege, referentna statorska frekvencija ω_s^{REF} je ujedno ulaz za trofazni PWM modulator i blok za račun u/f zakona upravljanja. U praksi se uobičajeno koristi neka vrsta rampe, čime se izbegavaju nagli skokovi frekvencije. U slučaju zatvorene povratne sprege, koristi se digitalni PI regulator brzine, koji uzima najnoviji odabirak referentne brzine ω_r^{REF} (i ovde se često koristi rampa), i merene brzine ω_r , izmeren u toku prethodne periode T_ω , i na osnovu njihove razlike računa novu vrednost učestanosti klizanja ω_{kl} koja se dalje po potrebi limitira na ω_{kl}^{MAX} . Nova vrednost referentne učestanosti se dobija iz jednačine slaganja brzina,

$$\omega_s(kT_\omega) = \omega_r(kT_\omega) + \omega_{kl}(kT_\omega). \quad (5.70)$$

Programski blok $u/f = const$ održava konstantan fluks tako što uvećava napon linearno sa učestanošću u I zoni, sve dok oni ne dostignu svoje nominalne vrednosti,

$$U_s(kT_\omega) = \frac{U_{sn}}{\omega_{sn}} \omega_s, \quad (5.71)$$

nakon čega se napon zadržava konstantnim i prelazi se u slabljenje polja, II zona, slika 5.23.



Slika 5.23. Uprošćen $u/f = \text{const}$ zakon primenjen u mikrokontroleru

Da bi se referentna vrednost napona prenela sinusnom PWM modulatoru neophodno ju je realizovati u odnosu na trenutnu vrednost ulaznog napona, koji se isto meri u periodu T_ω ,

$$A_{REF}(kT_\omega) = 2 \frac{U_s(kT_\omega)}{U_{DC}(kT_\omega)}, \quad 0 < A_{REF}(kT_\omega) < 1. \quad (5.72)$$

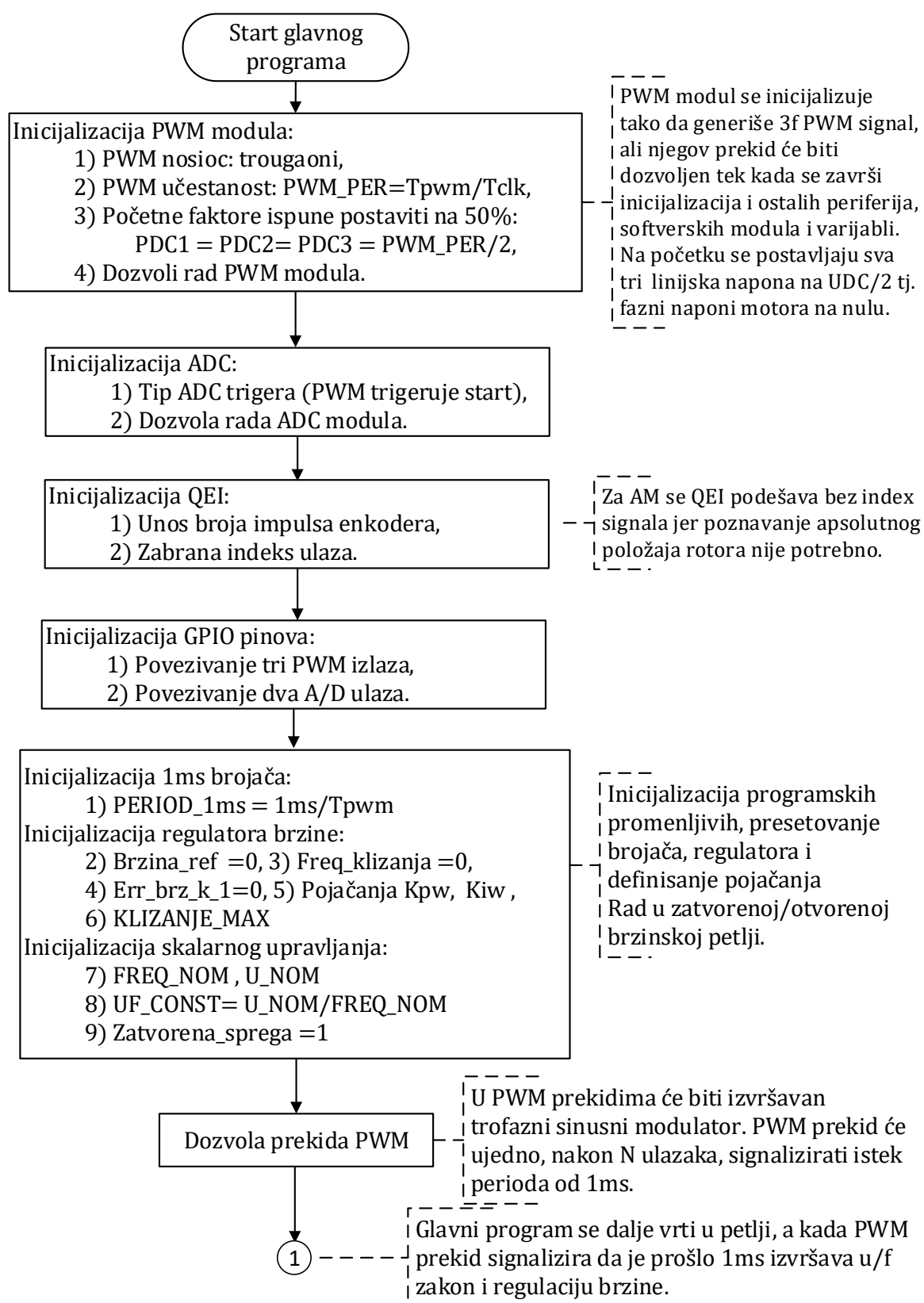
Nove vrednosti $\omega_s(kT_\omega)$ i $A_{REF}(kT_\omega)$ ostaju nepromenjene tokom tekuće periode T_ω , što znači da ih sinusni PWM modulator, koji se poziva svake PWM periode, koristi kao referentne vrednosti sledećih T_ω/T_{PWM} PWM perioda, na primer za $T_\omega = 1$ ms, $T_{PWM} = 50$ μ s dobija se $T_\omega/T_{PWM} = 20$.

5.6.1. Blok dijagram algoritma programa za skalarno upravljanje AM

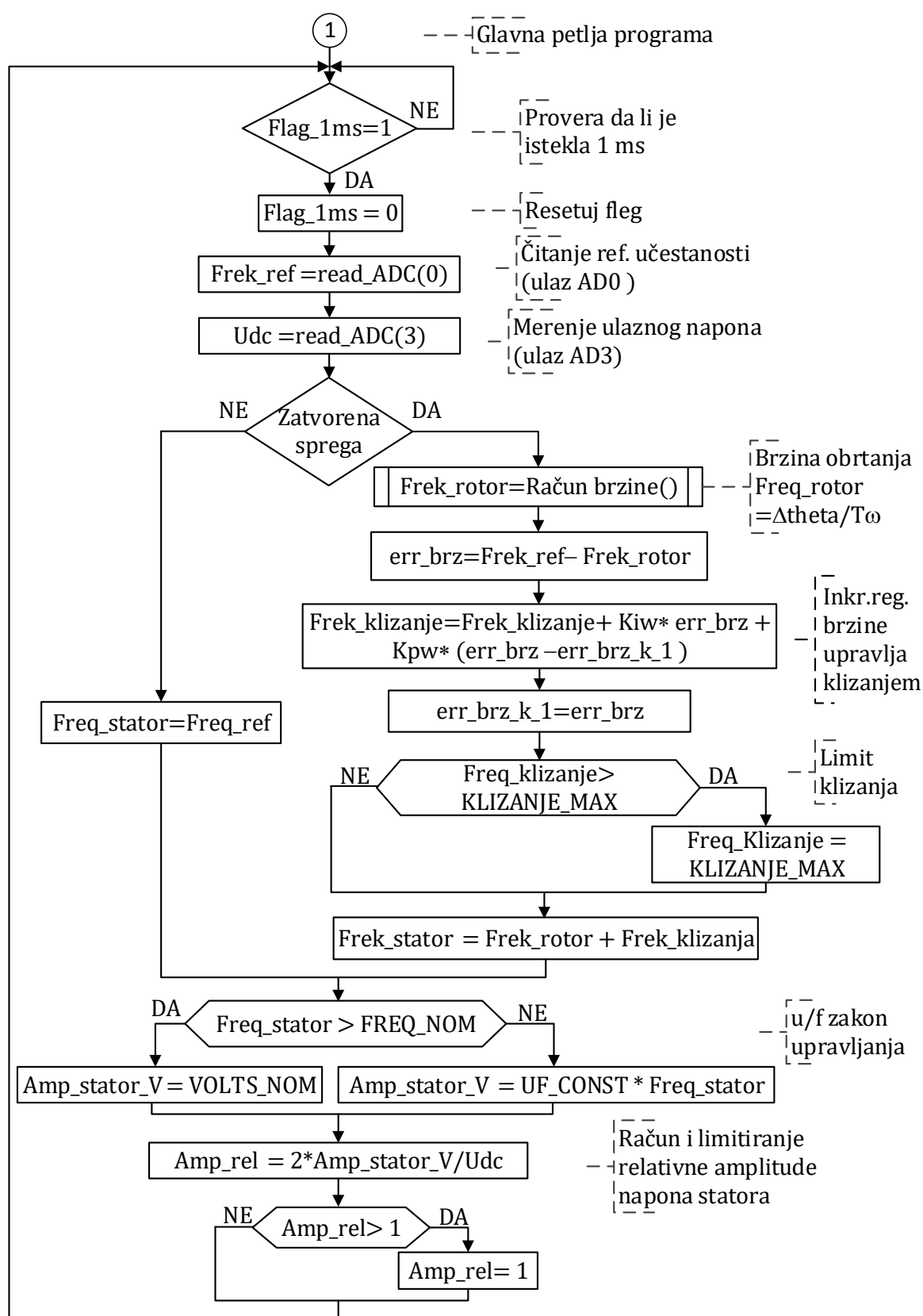
Blok dijagram algoritma programa za skalarno upravljanje trofaznim asinhronim motorom dat je u tri dela, slike 5.24 – 5.26.

Na slici 5.24 je dat početni deo programa (inicijalizacija) u koji mikrokontroler ulazi nakon reseta, tj. dolaska napajanja. U ovom delu se prvo vrši inicijalizacija korišćenih perifernih modula, u ovom slučaju to su PWM modul za generisanje PWM signala, QEI jedinica za merenje brzine obrtanja i A/D modul za merenje potrebnih analognih veličina. U drugom delu inicijalizacije program podešava programske module, globalne promenljive kao i programske konstante, u ovom slučaju postavlja referentnu brzinu na početnu vrednost nula, postavlja pojačanja regulatora brzine (za rad u zatvorenoj povratnoj sprezi), itd.

Na slici 5.25 je data osnovna petlja programa u koju on ulazi nakon inicijalizacije. Program nikad ne izlazi iz ove petlje (beskonačna petlja) s tim da povremeno stopira njeno izvršenje da bi po dolasku prekida izvršio tom prekidu dodeljenu prekidnu rutinu, u ovom slučaju prekid PWM modula. U ovom delu program jednom u periodu T_ω (1ms), čiji istek signalizira PWM prekidna rutina setovanjem $\text{Flag_1ms}=1$, čita se zadata frekvencija sa potencijometra (Frek_ref) i napon jednosmernog međukola. Zatim, program se grana u zavisnosti da li je zatvorena povratna sprega po brzini rotora. Ako program radi u otvorenoj povratnoj sprezi, zadata frekvencija se ujedno smatra i statorskom frekvencijom ($\text{Frek_stator} = \text{Frek_ref}$) i kao takva direktno prosleđuje bloku za račun u/f zakona, a tu istu vrednost koristi i trofazni sinusni PWM modulator. Ukoliko je zatvorena povratna sprega po brzini, zadata frekvencija se smatra referentnom frekvencijom rotora, i kao takva predstavlja referentnu vrednost za PI regulator brzine koji na svom izlazu računa novu vrednost frekvencije klizanja (Frek_klizanja). Izračunato klizanje se po potrebi limituje, a zatim dodaje na merenu frekvenciju rotora čime se konačno dobija referentna statorska frekvencija ($\text{Frek_stator} = \text{Frek_rotor} + \text{Frek_klizanja}$) koja se prosleđuje bloku za račun u/f zakona a koju će koristiti i trofazni PWM modulator.

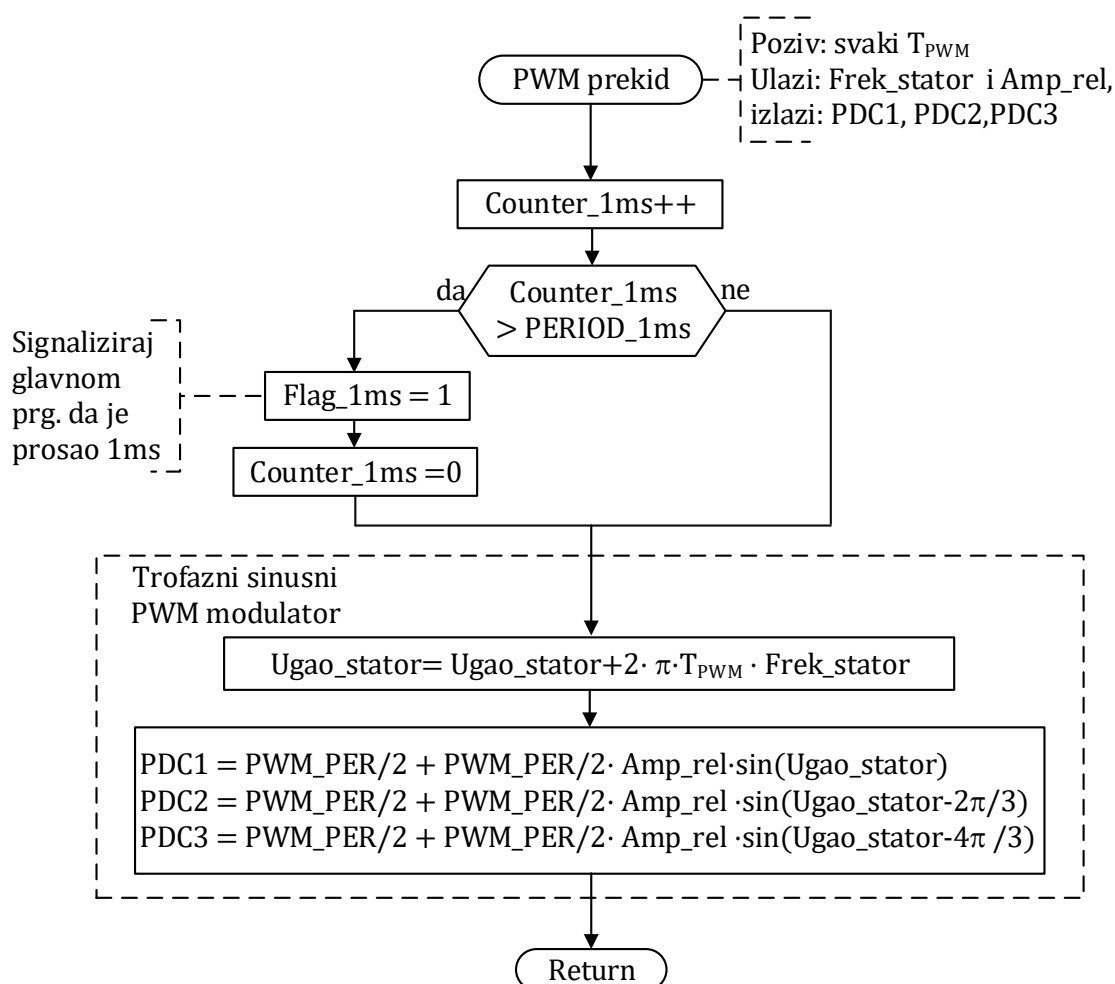


Slika 5.24. Blok dijagram algoritma inicijalizacije programa za skalarno upravljanje AM



Slika 5.25. Blok dijagram algoritma glavnog programa za skalarno upravljanje AM u otvorenoj i zatvorenoj povratnoj sprezi po brzini rotora [4_4]

Na slici 5.26 je prikazan blok dijagram algoritma mikroprocesorskog programa PWM prekidne rutine koja se koristi pri skalarnom upravljanju pogonom sa AM. Osnovni zadatak ovog dela programa, račun i upis tri vremena vođenja prekidača u sklopu trofaznog naponskog invertora, je vremenski kritičan i potrebno ga je obaviti u realnom vremenu, tj. pre dolaska sledeće PWM periode. Zato se ove funkcije obavezno i smeštaju u prekidnu rutinu PWM modula, čime je praktično zagarantovano da će se one pozivati i izvršavati sinhrono sa početkom PWM perioda. Samo na taj način je mikroprocesorskim programom moguće generisati izlazni trofazni napon čija se srednja vrednost neprekidno menja u vremenu, u skladu sa zatom statorskom frekvencijom i amplitudom. Naravno, odabrani mikrokontroler mora imati dovoljno procesorske snage da izvrši ovaj račun u toku jedne PWM periode, pre dolaska novog prekida PWM modula. U suprotnom, moglo bi doći do zastoja u generisanju izlaznog naizmeničnog signala, što bi omelo generisanje trofaznog obrtnog polja i dovelo do pada kvaliteta u napajanju AM, mogućih naglih promena struje statora i momenta koji AM generiše, a samim tim i do mogućeg kvara.



Slika 5.26. Blok dijagram algoritma programa PWM prekidne rutine za skalarno upravljanje AM bez obzira na tip povratne sprege po brzini rotora

5.6.2. Primer mikroprocesorskog C programa za dsPIC30F4011 za skalarno upravljanje brzinom AM

U ovom poglavlju dat je primer realizacije mikroprocesorskog C programa za skalarno upravljanje AM u otvorenoj povratnoj sprezi na Microchip DSP dsPIC30F4011, [5_11]. Program je testiran na pretvaraču baziranom na IRAM invertorskom modulu [5_8], [5_12], [5_13]. Korišćeni dsPIC30F4011 je 16-bitni mikrokontroler sa delimičnom Harvard arhitekturom koji radi na *clock* frekvenciji od 30MIPS i ima sve module potrebne za skalarno upravljanje AM. U prvom delu ovog poglavlja priložen je kôd za inicijalizaciju korišćenih perifernih modula, zatim su dati listinzi glavnog programa i PWM prekidne rutine.

5.6.2.1. C program za inicijalizaciju GPIO, ADC i PWM modula

U ovom poglavlju su date funkcije koje treba pozvati jednom nakon reseta i koje inicijalizuju periferije GPIO, ADC i PWM modul. Primeri programa za inicijalizaciju koji slede su dati u standardnom C jeziku ali koriste periferije i registre specifične za Microchip DSP tako da nisu prenosivi na drugi mikrokontroler ili DSP.

Definicija konstanti

U ovom poglavlju je dat listing konstanti koje koristi program. Većina su izvedene, na primer `PERIOD_UF_ZAKON` ili `PERIOD_PWM`, i računaju se na osnovu upisane *clock* frekvencije ($32\text{MHz}=32,000,000$), izabrane `FREQ_UF_ZAKON` (upisano 1kHz tj. 1000), izabrane PWM frekvencije `FREQ_PWM` ($16\text{kHz}=16,000$) i izabrane maksimalne frekvencije izlaznog napona `FREQ_MAX` ($100\text{Hz} = 100$).

```
// programske konstante
#define FREQ_CLOCK      32000000    // upisi DSP clock frekv.
#define FREQ_PWM        16000       // upisi zeljenu PWM frekv.
#define FREQ_UF_ZAKON   1000        // upisi frek izvršenja u/f zakona
#define FREQ_MAX        100         // upisi baznu (max) izlaznu frekv.
#define A_NOM            32767
#define FREQ_NOM         (int) (32767 * (float) 50 / FREQ_MAX)
#define UF_CONST         (long) ((32767*(float)A_NOM/FREQ_NOM))>>2
#define THETA_CONST      (long) 2 * 32767 * FREQ_MAX / FREQ_PWM
#define PERIOD_UF_ZAKON  (int) FREQ_PWM / Freq_Spore_petlje
#define PERIOD_PWM       (int) (FREQ_CLOCK / FREQ_PWM)
#define PERIOD_PWM_2     (int) PERIOD_PWM/2    // zbog troug. nosioca
#define PI               32767    // Relativno PI = 1 u Q1.15
#define DVA_PI_TRECINA  21844    // Relativno 2PI/3 = 2/3 u Q1.15
```

Inicijalizacija GPIO i pinova DSP

U ovom primeru, dva GPIO pina su odabrana za lokalno zadavanje referenci skalarnom pogonu. Pin RB0 je odabran kao analogni AD ulaz i služi za zadavanje frekvencije izlaznog signala preko potencijometra. Pin RB7 je definisan kao digitalni ulaz, i na njega se povezuje prekidač preko koga menjamo smer obrtanja motora, dakle znak zadate frekvencije. Listing funkcije za inicijalizaciju potrebnih pinova, `PIN_init()`, glasi:

```
void PIN_Init(){
ADPCFGbits.PCFG0 = 0;    // pin RB0 odabran kao analogni, on je sada A0 ulaz
ADPCFGbits.PCFG7 = 1;    // pin RB7 odabran kao digitalni,
TRISBbits.TRISB0 = 1;    // smer pina RB0 je odabran da bude ulazni
TRISBbits.TRISB7 = 1;    // smer pina RB7 je odabran da bude ulazni
}
```

Inicijalizacija ADC modula

Raspoloživi AD konvertor je 10-bitni i u ovom slučaju je podešen da daje broj u *integer* opsegu od 0 do 1023 (po potrebi, moguće je iskoristiti opciju automatskog pomeranja bitova rezultata). Odabiranje frekvencije vrši AD kanal 0, povezan sa RB0 pin. Detaljnije o priloženom kodu za inicijalizaciju AD konvertora unutar dsPIC u praktikumu [2_4].

```
// inicijalizacije ADC periferije
void ADC_Init(void)
{
    ADPCFG = 0xffff; // Omoguci analogni kanal AN0, active low
    ADCSSL = 0x0000; // Preskoci AN0-AN15 kod ulaznog skeniranja
    ADCON1 = 0xfcf8; // Ukljuci AD konvertor(bit15=1), onemoguci IDLE mod
                    // (bit13), format podataka integer (bit<9-8> = 00),
                    // auto convert mode(bit<7-5>=111), simultano odabiranje
                    // kanala (bit3), SAMP bita startuje odabiranje (bit2=0)
                    // Bit 0 ADCON1je AD busy bit i njega ovde ne menjamo
    ADCON2 = 0x08c0; // Reference napona AVss, Avdd (bit<15-13> =000),
                    // ne skenirati ulaze (bit10), konverzija kanala 0
                    // (bit<9-8>), jedan bafer 16-reči, koristi se samo MUXA
    ADCON3 = 0xe272; // Vreme semplovanja 2Tad, koristi se sistemski clock
    ADCHS = 0;       // AD muxA podešen na prvi AD ulaz gde je potencimetar
                    // (bit<7-6> = 00 konverzija u odnosu na GND),
                    // (bit<3-0> = 0000) izabran analogni ulaz 0
}
```

Inicijalizacija PWM modula

U funkciji za inicijalizaciju PWM modula *PWM_init()* osnovno je upisati vrednost u PTPER i time odabrati PWM periodu (frekvenciju). Zatim treba podesiti PWM timer u UP/DOWN mod rada (trougaoni PWM nosilac) i uključiti ga. Takođe za invertorski most obavezno podesiti komplementarni mod prekidača u sve tri grane (često je to *default* opcija). Sva tri faktora ispune se tokom inicijalizacije postavljaju na nulu (često je i to *default* opcija), tako da na motoru nema napona nakon reseta. Uobičajeno je da se dozvoli PWM FLT ulaz koji trenutno prekida PWM signale. Ali, pošto *fault* signal nije podržan od strane hardvera u ovom primeru je zabranjen. Konačno, dozvoljava se PWM prekid i postavlja njegov prioritet.

```
// inicijalizacija PWM periferije
void PWM_Init(void)
{
    PTPER = PERIOD_PWM_2; // Upis perioda (polovina ukupnog jer je up/down mod)
    PTCON = 0x8012;       // PWM vremenska baza uključena (bit 15),
                    // bez podskalera (bit<7-4>) i preskalera (bit<3-2>),
                    // Tip PWM nosioca trougao tj. up/down (bit<1-0> = 10)
    PWMCON1 = 0x08ff;     // Kanali 1,2,3 rade u komplement. modu (bit<10-8>)
                    // PWMxH pinovi se koriste kao PWM izlazi (bit<7-4>)
                    // PWMxL pinovi se koriste kao PWM izlazi (bit<3-0>)
    PWMCON2 = 0x0002;     // PWM Special Event Trigger Out Postscale (bit<11-8>)
                    // Nije sinh sa PWM vremenskom bazom (bit1)
    PDC1 = 0;             // Postavljanje vrednosti PDC registara na nulu
    PDC2 = 0;
    PDC3 = 0;
    IEC2 = 0x0800;        // Dozvoli PWM prekid (bit11=1), isključi FaultA
                    // prekid(bit7=0)
    IPC9 = 0x1000;        // Nivo prioriteta prekida PWM na 1 (bit<14-12>)
}
```

5.6.2.2. Glavni program - main funkcija

U glavnom programu se nakon reseta pozivaju sve funkcije potrebne da se sistem spremi za rad. U ovom primeru se pozivaju funkcije kojim se podešavaju (inicijalizuju) GPIO portovi dsPIC4011, kao i ADC modul i PWM modul koje program dalje koristi. Nakon poziva tih funkcija glavni program ulazi u beskonačnu petlju realizovanu sa *while(1)*, programskim uslovom koji je uvek zadovoljen. U ovoj petlji program svakih 1ms čita zadatu statorsku frekvenciju i računa statorski napon na osnovu $u/f=const$ zakona upravljanja. Glavni program čita frekvenciju kada mu PWM prekid to signalizira preko *flag* zastavice *Proslo_1ms*. Pošto je PWM frekvencija podešena na 16kHz PWM prekidna rutina setuje flag *Proslo_1ms* nakon što izbroji 16 ulazaka. Glavni program non-stop proverava ovaj *flag* i u slučaju da je setovan ulazi u deo programa unutar *if* naredbe. U tom delu programa se čita napon na potencijometru koji se dalje tumači kao zadata izlazna frekvencija. Pošto vreme izvršenja nije kritično, program čita ADC tako što ga prvo startuje, zatim u *while* petlji čeka pad AD BUSY signala što je znak da je rezultat konverzije spreman. Rezultat se uzima iz ADC rezultat registra 0 i upisuje u promenljivu *Freq_Stator*. AD modul je 10-bitni i daje rezultat u opsegu od 0 –1023. Program zato pri upisu ovog rezultata u *Freq_Stator* pomera 5 bitova ulevo čime se dobija Q1.15 format. Sada promenljiva *Freq_Stator* za maksimalnu vrednost napona na potencijometru uzima brojnu vrednost $1023 \cdot 2^5 = 32736$, što je približno jedan u Q1.15 formatu. Dakle, ovim je dobijena normalizovana zadata frekvencija statora u Q1.15 formatu, formatu potrebnom za dalje procesiranje u *fixed-point* sinusnom PWM modulatoru (poglavlje 5.2). Promenljiva *Freq_Stator* menja znak ako je prekidač za promenu smera pritisnut, tj. kada RB7 digitalni ulaz dobije vrednost jedan. Zatim se na osnovu apsolutne vrednosti zadate frekvencije računa prost u/f zakon upravljanja čiji je rezultat zadata amplituda *Amp_Stator*. Obe promenljive, *Freq_Stator* i *Amp_Stator* koristi PWM prekidna rutina koja se izvršava u prekidu koji u pozadini generiše PWM modul.

```
int main(void)
{
    PIN_Init();
    ADC_Init();
    PWM_Init();

    while(1)
    {
        if(Proslo_1ms)                // flag koji kontrolise PWM prekid,
        {                             // njime daje signal da je proslo 1ms
            Proslo_1ms = 0;
            ADCON1bits.SAMP = 1;      // startuje odabiranje AD ulaza
            while(!ADCON1bits.SAMP);  // cekaj da se odabiranje završi
            ConvertADC10();           // pocni konverziju analognog signala
            while(BusyADC10());       // cekaj da se AD konverzija završi
            Freq_Stator = ReadADC10(0); // citaj AD rezultat 0
            Freq_Stator = Freq_Stator << 5; // pomeri 5 bitova udesno (u Q1.15)
            Freq_Stator_abs = Freq_Stator; // zapamti abs f za u/f racun
            if(PORTBbits.RB7 == 1)    // promeni znak ako je RB7=1
                Freq_Stator = -Freq_Stator;
            if(Freq_Stator_abs >= FREQ_NOM) Amp_Stator = A_NOM;
            else Amp_Stator = (int)((long) Freq_Stator_abs * UF_CONST >> 13);
        } // vrši se svakih 1ms
    }
}
```

5.6.2.3. PWM prekidna rutina

PWM prekidna rutina je funkcija *PWMInterrupt()* sa kodnom reči *interrupt* koja je znak C kompajleru da se ta funkcija poziva u prekidu. Pošto C kompajler „zna“ da je ova funkcija pre svog poziva prekinula izvršenje nekog drugog koda (obično glavnu petlju), on automatski dodaje kôd koji sačuva na stek sve važne registre pre izvršenja funkcije i vraća ih sa steka nakon završetka funkcije. Ovim su sadržaji značajnih registara (akumulator, status, radni registri, itd.), koje u opštem slučaju može da promeni prekidna funkcija, sačuvani tako da kôd koji je prekinut nastavlja dalje bez greške nakon povratka iz prekida.

Početak ove funkcije se mora linkovati (smestiti) na mesto u memoriji na koje pokazuje adresni vektor prekida PWM modula, tj. na onu adresu koja se automatski upisuje u programski brojač po dolasku prekida PWM modula.

Program ulazi u PWM prekidnu rutinu po svakom novom prekidu generisanom od strane PWM modula. Ovaj prekid se generiše jednom u PWM periodu i glavni zadatak funkcije koja ga opslužuje je da proračuna nova punjenja tri PDC registra trofaznog PWM modulatora. Potrebni proračuni pri programskoj realizaciji trofaznog sinusnog PWM modulatora su opisani u poglavlju 5.2.

Pored računa PWM modulatora, prekidna rutina u ovom primeru broji svoje ulaske i nakon 16 ulazaka (1ms) setuje *Proslo_1ms* flag, koji predstavlja signal glavnom programu da je došlo vreme za izvršenje radnji planiranih u periodu od 1ms.

Na kraju PWM prekida moramo ukinuti zahtev za prekid, postaviti ga na nulu. Ovo važi za sve vrste prekida, samo je drugi flag u pitanju, jer u protivnom bi kontroler prekida stalno prekidao CPU i program bi konstantno ulazio u istu prekidnu rutinu.

```
// PWM prekidna rutina
void __attribute__((__interrupt__)) _PWMInterrupt(void)
{
    int Temp;
    if(++Counter_1ms_period > PERIOD_UF_ZAKON) // u ovom primeru 16 ulazaka
    {
        Proslo_1ms = 1; // 16 = 16kHz/100Hz // obavestava glavni da je
        Counter_1ms_period = 0; // doslo vreme za racun u/f //
    }
    // pomeraj ugla, racuna se i pamti u 32bitnoj matematici
    dTheta_Stator_32b = ((THETA_CONST*(long)Freq_Stator) >> 14);
    Theta_Stator_32b = (Theta_Stator_32b + dTheta_Stator_32b);
    Theta_Stator = (int)(Theta_Stator_32b); // ali koristi se samo 16-bita
    // Izracunavanje sinusa za dati ugao u Q1.15 formatu
    Sinus_A = SinApprox (Theta_Stator);
    Sinus_B = SinApprox (Theta_Stator + DVA_PI_TRECINA);
    Sinus_C = SinApprox (Theta_Stator - DVA_PI_TRECINA);
    // racun punjenja PDC1, sve u Q1.15
    Temp = (int)(((long) Amp_Stator * (long)Sinus_A) >> 15);
    Temp = (int)((long) PERIOD_PWM_2 * (long)Temp >> 15);
    PDC1 = PERIOD_PWM_2 + Temp; // PD1=period + period*A*sin(ugla)
    // racun punjenja PDC2, sve u Q1.15
    Temp = (int)(((long) Amp_Stator * (long)Sinus_B) >> 15);
    Temp = (int)((long) PERIOD_PWM_2 * (long)Temp >> 15);
    PDC2 = PERIOD_PWM_2 + Temp;
    // racun punjenja PDC3, sve u Q1.15
    Temp = (int)(((long) Amp_Stator * (long)Sinus_C) >> 15);
    Temp = (int)((long) PERIOD_PWM_2 * (long)Temp >> 15);
    PDC3 = PERIOD_PWM_2 + Temp;
    IFS2bits.PWMIF = 0;
}
```