

Mikroprocesorski softver, jun 2015. REŠENJA ZADATAKA

1. Sistemski takt procesora CCLK je učestanosti 10 MHz (perioda 0,1 μ s). Delitelj za dobijanje takta za periferije (PCLK) je podešen na 1.

Tajmer0 je inicijalizovan tako što su u sledeće registre redom upisane naznačene vrednosti:

T0TCR \leftarrow 0x02 (0000 0010 binarno)

T0PR \leftarrow 99

T0EMR \leftarrow 0x0080 (0000 0000 1000 0000 binarno)

T0MR1 \leftarrow 200

T0MCR \leftarrow 0x0028 (0000 0000 0010 1000 binarno)

Ostali registri nisu inicijalizovani (nalaze se u stanju posle reseta). Šta će se dešavati u procesoru (da li će biti zahteva za prekid, ako da, koji i kada, da li će se promeniti stanje nekog od izlaznih pinova i ako hoće, kako...) posle upisa vrednosti 0x01 u registar **T0TCR**? (8)

Koja vrednost bi se očitala iz registra **T0TC** jednu sekundu posle upisa vrednosti **0x01** u registar **T0TCR** i šta će se tačno desiti sa izlaznim signalom ako se tada u registar **T0TCR** upiše vrednost 0x02 a odmah u sledećoj instrukciji 0x01? (6)

Šta sve treba uraditi da bi se signal pojavio na nožici procesora i na kojoj nožici se može pojaviti?..... (5)

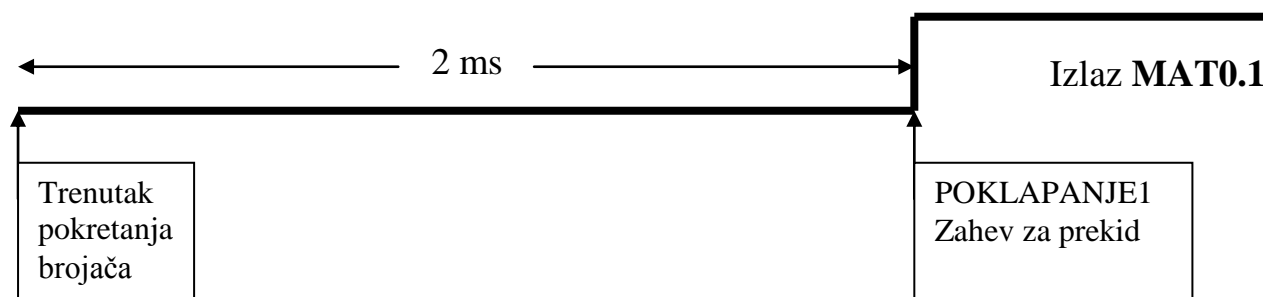
Prva instrukcija (upis u T0TCR) resetuje (vraća na nulu i zaustavlja) tajmer T0. Druga instrukcija podešava predelitelj na 100. Dakle, učestanost takta koji se dovodi glavnom brojaču (T0TC) je 100kHz (10MHz/100), odnosno glavni brojač se uvećava svakih 10 μ s (0,1 μ s*100).

Upisom u T0EMR se za registar poklapanja 1 definiše spoljni događaj prilikom poklapanja T0TC sa T0MR1. Biti T0EMR.6 i T0EMR.7 definišu da se u slučaju poklapanja1 (poklapanje T0TC sa T0MR1) stanje izlaznog pina MAT0.1 (P0.5 ili P0.27) postavlja na 1 ako je bilo 0 (ako je već bilo 1, ostaje 1).

Upisom u T0MCR se definiše šta još treba da se desi i slučaju poklapanja1 (poklapanje T0TC sa T0MR1) biti T0MCR.3 = 1, T0MCR.4 = 0, i T0MCR.5 = 1 definišu da se prilikom poklapanja1 brojač T0TC zaustavlja (ne broji dalje), ne briše se i periferija postavlja zahtev za prekid.

Zbog ovakve inicijalizacije, od trenutka kada se brojač T0TC pokrene (upisom 0x01 u T0TCR), brojač će početi da se uvećava za 1 svakih 10 μ s. Kada dosigne 200 (koliko je upisano u T0MR1) desiće se poklapanje1 i tada će se brojač zaustaviti (neće brojati dalje), a periferija će jezgru (tačnije, kontroleru prekida odnosno, VIC-sistenskoj periferiji) proslediti zahtev za prekid (MR1 prekid tajmera 0).

Dakle izlaz MAT0.1 koji je posle reseta bio 0, će posle tačno 2ms (200*10 μ s) skočiti na nivo ločičke jedinice. Ako je MAT0.1 izlaz povezan za nožicu P0.5 (upisom PINSEL0.11:10 = 10₂) ili P0.27 (upisom PINSEL1.23:22 = 11₂) na nožici za koju je povezan će se 2ms po pokretanju brojača pojaviti rastuća ivica.



Pošto je brojač zaustavljen, svako očitavanje po isteku 2ms od njegovog pokretanja (pa i posle 1s) će dati vrednost pri kojoj je zaustavljen odnosno 200.

Ako se brojač resetuje i odmah ponovo pokrene, ceo ciklus će se ponoviti, ponovo će se izlaz posle 2ms od startovanja postaviti na 1. Ako izlaz nije vraćen na 0, odnosno ako je iz prvog ciklusa ostao na nivou logičke jedinice, u drugom ciklusu se neće desiti nikakva promena.

Odgovor na treći deo zadatka je dat ranije, nožice su P0.5 i P0.27.

2. Na ulaz AD0.0 dovodi se promenljivi signal čija amplituda ne prelazi 1,5V. Izraditi deo C-kôda koji bi obezbedio da se svake milisekunde odmeri signal, i na A_{OUT} izlaz DA konvertora postaviti (takođe svake milisekunde) dvostruka vrednost signala dovedenog na AD0.0. Na primer, ako napon doveden na AD0.0 varira između 0,2V i 1V, napon na na A_{OUT} izlazu DA konvertora treba da varira između 0,4V i 2V? Referentni napon DA konvertora je 3,00V. Pretpostaviti da funkcija `void pauza_1ms (void)` već postoji (ne treba je pisati).(10)

Šta se dobija, a šta gubi upisivanjem jedinice u bit 16 **DACR** registra i kada se to koristi?..... (5)

```
unsigned int rez, izlaz;
while (1)
{
    pauza_1ms ();
    AD0CR = 0x01200201;          SEL=01, CLKDIV=2 (takt se deli, na primer, sa 3, zadatkom nije
                                precizirano), BURST=0, PDN=1, START=001
    while (!AD0DR&(1<<31));      Čekanje kraja konverzije
    rez = (AD0DR>>6) & 0x03FF;   Preuzimanje rezultata AD konverzije
    izlaz = rez << 1;             Množenje rezultata AD konverzije sa 2
    DACR = izlaz << 6;           Punjenje DA konvertora
}
```

Umesto poslednje tri instrukcije bilo je moguće čitanje AD, množenje i punjenje DA obaviti jednom instrukcijom. U tom slučaju nije potrebno deklarirati promenljive `rez` i `izlaz`.

```
DACR = (AD0DR&0x0FFC0)<<1;
```

3. Čemu služi, zašto je nužan i kako se koristi registar sa oznakom **PWMLER** (*PWM Latch Enable Register*)(4)

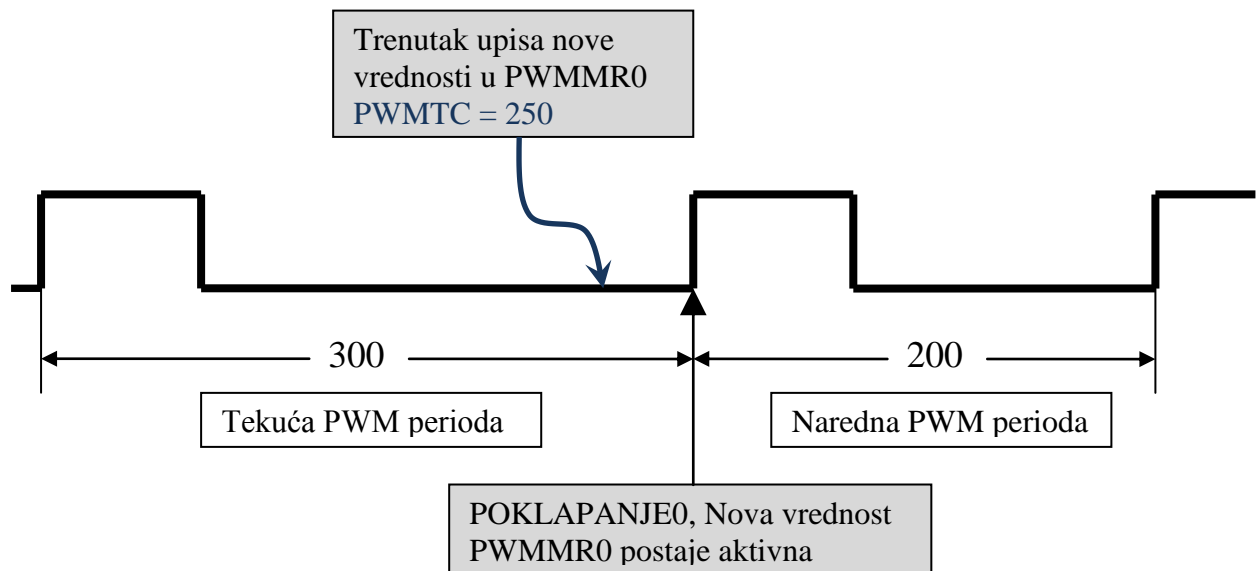
Navesti koje nepravilnosti bi se mogle desiti u generisanju PWM signala kada ovaj mehanizam ne bi postojao, kao što ne postoji kada PWM radi u režimu klasičnog tajmera-brojača. (2)

PWMLER je detaljno opisan u priručniku (materijal pod nazivom Širinski modulator).

Kod upisa novih vrednosti u registre poređenja bitno je sledeće:

- Upisivanjem u registre PWMMR0 do PWMMR6, podatak se ne upisuje direktno u radni registar (onaj koji se poredi sa tajmerom) već u poseban registar u senci. Podaci se čuvaju u registru u senci sve do kraja tekuće PWM periode i tek po njenom završetku, sadržaj iz registara u senci se prebacuje u radni registar i u sledećoj PWM periodi poređenje se vrši sa novom vrednošću. Ovo važi za sve PWMMRx registre kada PWM brojač radi u PWM režimu (PWM režim se uključuje upisom jedinice u PWMTCR.3). Na taj način, nove vrednosti možemo upisati u registre poređenja u bilo kom trenutku, a one će postati „važeće“ tek u novoj PWM periodi. PWM će završiti celu tekuću periodu sa starim vrednostima.
- Kraj tekuće PWM periode je u trenutku poklapanja0 (poklapanje PWMTCR sa PWMMR0).
- „Aktiviranje“ novoupisanih vrednosti tek na kraju PWM periode sprečava grešku do koje može doći kada se, na primer, stanje PWMMR0 promeni sa veće vrednosti na manju u trenutku kada je

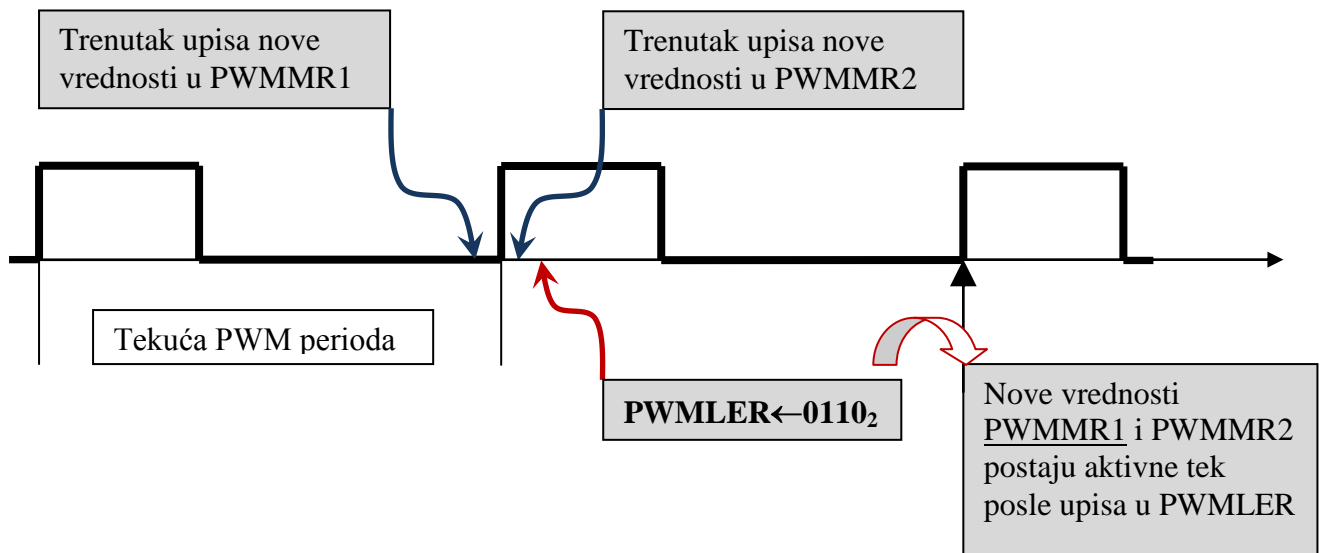
ta manja vrednost već prošla. Na primer, ako je PWMMR0 bio 300 pa želimo da ga smanjimo na 200, i to uradimo u trenutku kada je PWMTTC dostigao 250, poklapanje PWTC (≥ 250) sa PWMMR0 (200) se neće desiti nikad, tačnije, desiće se tek kada PWMTTC odbroji do pune vrednosti (0xFFFFFFFF), premoti se, i stigne ponovo do 200. Zahvaljujući registrima u senci, nova vrednost (200) će postati aktivna tek kada se tekuća perioda završi, odnosno kada PWMTTC odbroji do 300, koliko stoji u radnom (aktivnom) PWMMR0 registru. Tekuća perioda (ona u toku koje je promenjena vrednost u PWMMR0) će trajati 300, a tek sledeća 250. Ovaj mehanizam postoji uvek kada tajmer radi u PWM režimu i za ovoaj tip zaštite od opisane greške, PWMLER registar nije nužan i opis je dat kao da **PWMLER mehanizam ne postoji**.



Slika Zaštita pomoću registara u senci od upisa u PWMMR0 vrednosti koja je već prošla

- Zaštita od ove greške pomoću registara u senci, međutim, nije dovoljna ako se koristi više PWM signala, a potrebno je da se trajanje impulsa (definisano stanjem PWMMR1 do PWMMR6) menja istovremeno na svim signalima. Problem nastaje zbog toga što se upis u registre PWMMR1 do PWMMR6 ne može obaviti jednom instrukcijom, već je potrebno onoliko instrukcija koliko se stanja registara menja, a instrukcije se izvršavaju jedna za drugom. Pretpostavimo da PWMMR1 određuje trajanje impulsa na PWM1 izlazu, a PWMMR2 trajanje impulsa na PWM2 izlazu i da je potrebno promeniti ove dve širine impulsa. Ako se instrukcija kojom se upisuje novo stanje PWMMR1 izvrši neposredno pre kraja tekuće PWM periode, a instrukcija kojom se upisuje novo stanje PWMMR2, po završetku ove periode, PWM1 će promeniti širinu u prvoj sledećoj periodi, a PWM2 tek u narednoj, što je u mnogim primenama nedopustivo. Mnogo gora greška može da nastane kada se kontrolišu dve ivice PWM, pa jedna promena se desi u prvoj periodi po upisu, a druga promena u drugoj. Zahvaljujući mehanizmu zasnovanom na PWMLER registru omogućena je istovremena promena svih PWMMRx registara.

Naime, promene u ovim registrima ne menjaju ni sadržaj registara u senci već se čuvaju u privremenim registrima, a tek upisom u PWMLER, te nove vrednosti prelaze u registre u senci. Upis u PWMLER registar treba obaviti jednom instrukcijom, a podatak koji se upisuje u ovaj registar se sastoji od logičkih jedinica na odgovarajućim, po jedna jedinica za svaki od PWMMRx registara čije stanje treba promeniti. U prethodnom primeru, ako perioda (PWMMR0) nije menjana, a menjani su sadržaji registara PWMMR1 i PWMMR2, posle upisa novih vrednosti u ove registre, u PWMLER registar treba upisati 0x6, odnosno binarno, 0110₂ i tek u trenutku upisa u PWMLER, nove vrednosti PWMMR1 i PWMMR2 prelaze u registre u senci i čekaju sledeće poklapanje0 (kraj PWM periode) da bi postali aktivni. Dakle, na kraju periode obeležene na slici kao „tekuća PWM perioda“, PWMMR1 neće postati aktivan.



Slika Zaštita pomoću mehanizma upisa u PWMLER registar

4. Koje sve registre treba inicijalizovati i na koje vrednosti da bi se ostvarila šema prekida u kojoj najviši prioritet ima TIMER1, za njim TIMER0 ako oba prekida treba da koriste FIQ mehanizam prekida Objasni šta treba testirati u prekidnoj rutini i skicirati C-kod ulaska u FIQ prekidnu rutinu. Početne adrese programa za opsluživanje prekida tajmera su **poc_TIMER1** i **poc_TIMER0**, respektivno. Simbolička adresa početka FIQ prekida je **poc_FIQ**. Voditi računa o redosledu upisa u registre.(10)

Prekid sa brzim ulazom (FIQ) praktično ne koristi VIC periferiju. Ovu vrstu prekida periferija samo prepoznaje (na osnovu sadržaja registra VICIntSelect) i prosleđuje zahtev jezgru. Ako postoji više periferija koje su kvalifikovane da koriste FIQ, VIC periferija objedinjuje zahteve od ovih periferija u jedan zajednički. Time se uloga VIC periferije, što se tiče FIQ, završava. Obrada zahteva je prepuštena jezgru (kao da VIC periferije nema).

Dakle, asemblersku instrukciju za bezuslovni skok na početnu adresu poc_FIQ treba upisati na 0x1C (FIQ vektor) u okviru vektora izuzetaka (*exception vectors*).

Pošto postoji samo jedna prekidna rutina za FIQ, na njenom početku treba testirati statusni registar (najbolje je testirati VICFIQStatus, mada je moguće i VICRawIntr) i ako zahtev za prekid dolazi od tajmera1, izvršiti rutinu za njegovo opsluživanje. Ako su prekidi zabranjeni u okviru prekidne rutine, time je obezbeđeno da tajmer1 ima viši prioritet od tajmera0.

Upis u VICIntSelect registar :

VICIntSelect = 0x0030; Periferije Tajmer0 i Tajmer1 koriste prekid sa brzim ulazom (FIQ)

Deo koda na ulasku u FIQ prekidnu rutinu:

```
.
.
.
if (VICFIQStatus & (1<<5)) poc_TIMER1()
else if(VICFIQStatus & (1<<4)) poc_TIMER0();
.
.
.
```

Rešenja zadataka su data kao studija i dopuna literaturi. Naravno da na ispitu nije potrebno sa toliko detalja davati odgovore.