

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

Koje vrednosti, i u koje redistre, treba upisati da bi se, bez korišćenja prekida i softvera, na jednom od izlaza, po slobodnom izboru, generisala periodična povorka četvrtki učestanosti 1 kHz (500 μ s impuls i 500 μ s pauza).(10)

Analizirati kako bi se na nekom drugom izlazu (ponovo po slobodnom izboru) generisao još jedan isti takav signal pomeren u odnosu na prvi za 250 μ s. Skicirati deo C-kôda. Ukoliko je neophodno, koristiti prekide.(4)

Zadatak je moguće rešiti pomoću periferije tajmer ili pomoću PWM periferije. Rešenje pomoći PWM je najjednostavnije s obzirom na drugi (fazno pomereni) signal koji treba generisati. Prvi signal se može generisati, na primer, na PWM1 pinu, sa kontrolom jedne ivice, a drugi, recimo na PWM3 sa kontrolom obe ivice. S obzirom da se traži perioda 1 ms (1000 μ s), i da bi rezolucija 1:100 bila više nego dovoljna, za periodu takta koji broji PWM brojač može se usvojiti perioda od 10 μ s, odnosno predelitelj PCLK takta može biti 100 jer je (100*0,1 μ s=10 μ s). U tom slučaju perioda PWM bi iznosila 100 takta koji broji PWM brojač. PWM3 treba inicijalizovati da radi u režimu kontrole obe ivice, a PWM1 u režimu kontrole jedne ivice što je podrazumevan režim. MR0 određuje periodu (100), MR1 trenutak opadajuće ivice na PWM1 (50); MR2 trenutak rastuće ivice na PWM3 signalu (25), a MR3 trenutak opadajuće ivice ivice na PWM3 signalu (75).

PWMTCR = 0x02 ; (0000 0010 binarno) Zaustavljanje i reset brojača.

PWMPR = 99 ; Predelitelj je 100 (učestanost PCLK se deli sa 100 da bi se dobio takt glavnog brojača)

PWMMCR = 0x0002 ; (0000 0000 0000 0010 binarno) Poklapanje sa MR0 resetuje brojač

PWMPCR = 0x0A08 ; PWM1 i PWM3 uključeni, PWM3 u režimu kontrole dve ivice

PWMMR0 = 100 ; Perioda PWM je 100 jer je :(100*10 μ s = 1 ms)

PWMMR1 = 50 ; Vidi komentar pre koda

PWMMR2 = 25 ; Vidi komentar pre koda

PWMMR3 = 75 ; Vidi komentar pre koda

PWMTCR = 0x09 ; (0000 1001 binarno) Puštanje brojača u PWM režimu.

PWMLER = 0x0F ; (0000 1111 binarno) Aktiviranje novih vrednosti upisanih u MR0 do MR3

Šta treba promeniti da bi, pored generisanja signala, periferija postavljala zahtev za prekid svake milisekunde u trenucima rastuće ivice prvog izlaznog signala?(4)

Treba promeniti vrednost upisanu u PWMMCR tako da poklapanje sa MR1 (tu je definisana rastuća ivica prvog signala) postavlja zahtev za prekid. Postavljanje zahteva za prekid pri poklapanji TC sa MR0 određuje bit 3 PWMMCR registra.

PWMMCR = 0x000A ; (0000 0000 0000 1010 binarno) Poklapanje TC sa MR0 resetuje brojač a poklapanje sa MR1 postavlja zahtev za prekid.

Opisati kako radi i čemu služi predelitelj u tajmer-periferiji LPC2138.(2)

2. Na ulaze AD0.0 i AD1.0 se dovode dva sporopromenljiva signala amplitude koja je manja od 3,3 V. Svake milisekunde generisati na izlazu DA konvertora signal jednak srednjoj vrednosti signala sa ulaza AD0.0 i AD1.0. Pored toga, na jedan GPIO port, po sopstvenom izboru, generisati logičku jedinicu ako je signal na AD0.0 veći od signala na AD1.0, a nulu ako nije. Učestanost perifernog takta PCLK je 4MHz. Pretpostaviti da funkcija `void pauza_1ms (void)` koja pravi pauzu od 1ms, već postoji. Zanimariti vreme izvršavanja ostalih instrukcija.

Potrebno je u petlji koja se ponavlja svake milisekunde pokrenuti istovremenu konverziju na oba kanala, sačekati kraj konverzije, izračunati vrednost za DAC i upisati je u DACR.

Inicijalizacija AD0:

AD0CR_bit.SEL = 00000001₂ ;Priprema kanala 0 za start ([AD0CR.7..0](#))

AD0CR_bit.CLKDIV = 00000000₂ ; Takt od 4MHz je u redu (< 4,5MHz), ne treba ga deliti ([AD0CR.15..8](#))

AD0CR_bit.BURST = 0₂ ; Isključen režim automatske promene kanala (*burst mode*) ([AD0CR.16](#))

AD0CR_bit.PDN = 1₂ ; Uključeno napajanje AD ([AD0CR.21](#))

AD0CR_bit.START = 000₂ ; Start konverzije odložen za kasnije ([AD0CR.26..24](#))

U ostale bite AD0CR treba upisati nulu (reset stanje).

Inicijalizacija AD1 je identična.

Pokretanje istovremene konverzije :

ADGSR_bit.BURST = 0₂ ; Isključen režim automatske promene kanala (*burst mode*) ([ADGSR.16](#))

ADGSR_bit.START = 001₂ ; Softverski start konverzije (startuj odmah) ([ADGSR.26..24](#))

U ostale bite ADGSR treba upisati nulu (reset stanje).

unsigned int Rez0, Rez1, Sred_vred

IO0DIR = 0x01 ; Port 0.0 je izlazni, koristiće se za bit o informaciji da je AD0.0 > AD0.1

while (1)

```
{
AD0CR = 0x00200001 ; Inicijalizacija AD0
AD1CR = 0x00200001 ; Inicijalizacija AD1
ADGSR = 0x01000000 ; Startuj konverzije odmah
pauza_1ms () ; Pauza od 1 ms, ujedno i čekanje završetka konverzija
Rez0 = (AD0DR & 0x0000ffff) >> 6;
Rez1 = (AD1DR & 0x0000ffff) >> 6;
Sred_vred = (Rez0+Rez1)/2;
DACR = Sred_vred << 6; DACR_bit.BIAS =0, nije zahtevana smanjena potrošnja
if (Rez0>Rez1) IO0SET = 0x01 else IO0CLR = 0x01;
}
```

Zadavanje vrednosti DA konvertoru je bilo moguće i bez deklaracije promenljivih. Kompletan kôd iza pauze se može zameniti komandom:

```
DACR = ( (AD0DR&0x0000ffff) + (AD1DR&0x0000ffff) ) >>1;
// (deljenje sa 2 zamenjeno pomeranjem udesno za jedno mesto)
```

Opisati kako radi automatska promena kanala (*burst mode*) AD konvertora.(2)

Da li je moguće pokrenuti *burst mode* na oba AD konvertora i ako jeste, kako?(4)

3. Koje sve registre treba inicijalizovati, i na koje vrednosti, da bi se ostvarila šema prekida u kojoj periferije WDT i tajmer0 i koriste brzi prekid (FIQ), a periferije SPI0 i SPI1 takođe koriste mehanizam vektorizovanog prekida.(6)

Potrebno je u VIC periferiji deklarirati prekid od WDT i tajmera0 kao prekide sa brzim ulaskom (FIQ), a prekidima od SPI0 i SPI1 dodeliti dva najviša prioriteta vektorizovanih standardnih prekida (IRQ).

```
VICIntEnable = 0x00000C11 ; Dozvola korišćenim prekidima (bit 0, 4, 10 i 11)
VICIntSelect = 0x00000011 ; Prekidi od WDT i tajmera0 su sa brzim ulaskom (bit 0 i 4)
VICVectCntl0 = 0x0000002A ; Prekidi od SPI0 na mesto najvišeg prioriteta (slot0 aktiviran)
VICVectCntl1 = 0x0000002B ; Prekidi od SPI1 na mesto prioriteta ispod (slot1 aktiviran)
VICVectADDR0 = poc_SPI0 ; Vektor prekida od SPI0 u slot 0
VICVectADDR1 = poc_SPI1 ; Vektor prekida od SPI1 u slot 1
```

Simboličke početne adrese programa za opsluživanje prekida odgovarajućih periferija su **poc_WDT**, **poc_TIMER0**, **poc_SPI0** i **poc_SPI1**. Skicirati deo C-kôda kojim počinje funkcija za opsluživanje prekida sa brzim ulazom sa simboličkom početnom adresom **FIQ_pocetak**. Gde treba upisati ovu adresu?(4)

Asemblersku instrukciju bezuslovnog skoka na FIQ_pocetak treba smestiti na adresu 0x18 (0x18 je FIQ vektor) ili obezbediti da FIQ_pocetak bude 0x18, odnosno, celu prekidnu rutinu prekida sa brzim ulaskom (FIQ) smestiti u ROM počev od adrese 0x18.

Na početku rutine treba proveriti da li je prekid izazvao WDT, ako jeste, odgranati se na rutinu sa početnom adresom poc_WDT, ako nije na rutinu poc_TIMER0. Provera se može obaviti testiranjem statusnog registra VICFIQStatus.

```
if (VICFIQStatus == 0x00000001)
poc_WDT
else poc_TIMER0 ; Nema drugih FIQ prekida, tako da ako nije od WDT onda je od tajmera0
```

Ako bi mehanizam prekida bio isključen, koji registar, i kako bi softver trebalo da testira da bi detektovao da je bilo koji od ovih izora postavio zahtev za prekid?(4)

Ako je mehanizam prekida isključen, treba testirati VICRawIntr statusni registar. Da bi se došlo do informacije da je bilo koja od periferija postavila zahtev za prekid dovoljno je proveriti da li je ovaj statusni registar različit od nule (uz pretpostavku da ni jedna duga periferija, osim ove četiri, ne može da postavi zahtev za prekid).

```
if (VICRawIntr)
{ ... } ; Bilo je zahteva za prekid od neke od četiri periferije.
```

Ili, sigurnije:

```
if (VICRawIntr & 0x00000C11)
```