LABORATORIJA ZA MIKROPROCESORSKO UPRAVLJANJE ENERGETSKIM PRETVARA^IMA I POGONIMA

etf-beograd

PRIMEDBE:

PRIMEDBE@YAHOO.COM

MCK240

TMS320F240 DSP Motion Control Kit

1. POSTUPAK INSTALACIJE:

1.1. Raspakovati *MCWIN* programski paket u istoimeni direktorijum na izabranom disku (C: , D: ...);
1.2. Priklju~iti na serijski port plo~u *MCK240*;
1.3. Uklju~iti napajanje na plo~u *MCK240*;
1.4. Otvoriti direktorijum *MCWIN* na odgovaraju}em disku, prona}i izvr{ni program *Mcwin.exe* i startovati ga (radi br`eg startovanja programa mogu}e je napraviti pre~icu - *shortcut* - za ovaj izvr{ni program i smestiti je na *Desktop* ra~unara i odatle ga startovati).

Napomena: Ukoliko nakon instalacije i startovanja pojedinih aplikacija program javlja "communication error" proveriti da li je sve dobro povezano i ako jeste onda resetovati plo~u preko tastera *RESET* koji se nalazi na plo~i te nastaviti sa radom

2. ZADATAK

Prema prilo`enom uputstvu uraditi slede}e ve`be po zadatom redosledu:

2.1. - Pomo}u korisni~kog interfejsa (u grafi~kom okru`enju) u aplikaciji *I/O ports* uklju~iti LED diode na portovima 5, 6 i 7.
- Zatim, pomo}u programa *Monitor* o~itati vrednosti bajtova za konfiguraciju porta *B* i informaciju o vrednosti signala na portovima.
- Pomo}u programa *Monitor* uticati na rad aplikacije kako je to sugerisano u uputstvu.

2.2. - O~itati vrednost napona na *A/D* konvertoru pomo}u grafi~kog okru`enja za datu aplikaciju *A/D* konverzije.
- Pomo}u programa *Monitor* o~itati parametre *A/D* konverzije koji se mogu videti u grafi~kom okru`enju.

2.3. - Za sva tri tipa *PWM* modulacije generisati razli~ite talasne oblike napona na izlazu invertorskog mosta i konstatovati okretanje osovine motora nakon startovanja aplikacije preko *Run* tastera.
- Pomo}u *Monitor* programa o~itati periodu *PWM* nosioca i vrednosti u *Full Compare Unit* registrima i proveriti ta~nost vrednosti pojedinih napona pomo}u odgovaraju}ih izraza datih u uputstvu.
- Za *Space Vector* modulaciju:
1) Setovati malu vrednost *Space Vector* amplitude
2) Selektovati smer suprotan od smera kazaljke na satu
3) Setovati po~etnu vrednost *Space Vector* ugla $\theta \in (1°;59°)$ i startovati aplikaciju.
4) Davati vrednosti uglu θ: 30, 150, 270, 30, 150, 270, ...
5) Ponoviti proceduru za smer kazaljke na satu.

2.4. Pokretanjem osovine motora pomo}u programa za *QEP* o~itavati aktuelnu vrednost pozicije koja se zadaje ru~nim okretanjem osovine motora i pomo}u programa *Monitor* o~itati aktuelnu izmerenu vrednost pozicije.

2.1 Primer editovanja asemblerskog programa

Mogu}e je menjati sadr`aj asemblerskog koda u svim aplikacijama sa namerom da se promeni deo ili cela funkcija odre|ene aplikacije. Uzmimo, na primer, asemblerski program koji defini{e rad digitalnih *I/O* portova (zadatak 2.1.):

- Otvoriti direktorijum *MCWIN* na odgovaraju}em disku te u}i u poddirektorijum *Demos*;
- U direktorijumu *Demos* prona}i poddirektorijum *Gpio* i u}i u njega;
- Otvoriti, uz pomo} nekog editora (npr. *Notepad),* asemblerski program *Gpio_a.asm*;
- U asemblerskom kodu prona}i slede}i deo koda:

```
...
; output values
        LDP   #_config_io
        LACC  _config_io,8
        OR    _output_value
        LDP   #DP_PF2
        SACL  PBDATDIR
        LACC  PBDATDIR
        LDP   #_input_value
        AND   #0FFh
        SACL  _input_value
...
```

i editovati ga tako da se unese nova linija - XOR #00FFh - te gore napisani deo koda sada izgleda:

```
...
; output values
        LDP   #_config_io
        LACC  _config_io,8
        OR    _output_value
        LDP   #DP_PF2
        XOR   #00FFh
        SACL  PBDATDIR
        LACC  PBDATDIR
        LDP   #_input_value
        AND   #0FFh
        SACL  _input_value
…
```

- Snimiti izvr{enu izmenu koda i iza}i iz editora;
- Aktivirati zatim *Go.bat* fajl (aktivira se kao bilo koji drugi izvr{ni program) koji se nalazi u direktorijumu *Gpio* i sa~ekati da se izvr{i kompilacija i povezivanje novog koda (pojavljuje se prozor koji daje obave{tenje o statusu kompilacije i povezivanja i daje informaciju da li ima gre{aka i upozorenja);
- Startovati ikonu ***I/O Ports*** u ***Processor Evaluation Control Panel***-u i uraditi ponovo zadatak 2.1. [ta se de{ava sa LED diodama?

Na sli~an na~in mogu}e je menjati kodove te vr{iti kompilaciju drugih aplikacija sa ciljem ostvarenja funkcija od interesa.

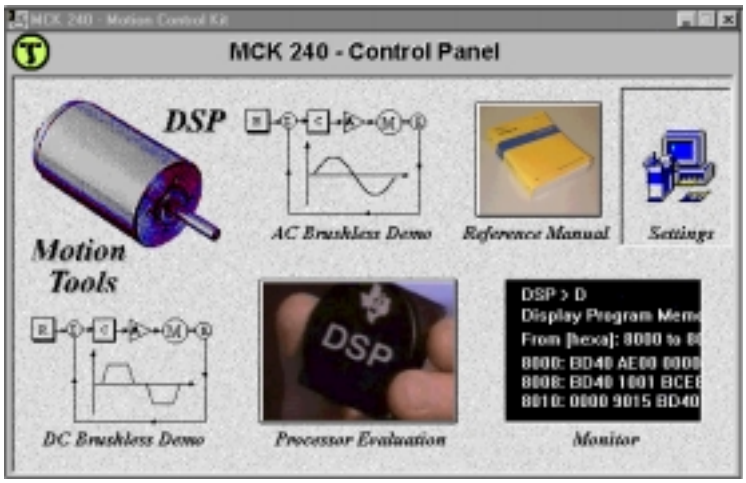
3. PREGLED PROGRAMSKIH APLIKACIJA

MCK240 sadr`i nekoliko softverskih aplikacija grupisanih i nazvanih zajedni~kim imenom **MCWIN**. Tu spadaju:

- MCWIN okru`enje
- Program za komunikaciju (MONITOR)
- Procesorke aplikacije (PROCEV)
- Alat za analizu rada motora (DSPMOT)
- Primeri rada motora (BLAC i BLDC)

3.1. MCWIN okru`enje

Ova okru`enje se startuje dvostrukim klikom levog tastera mi{a na ikonu MCWIN koja se nalazi na *Desktop*-u ili jednostavno pokretanjem programa *Mcwin.exe* iz komandne linije. Startovanjem datog programa na ekranu }e se pojaviti prozor koji izgleda, kao na slici:



Slika 1. MCWIN okru`enje

Zna~enje svake ikone u ovom prozoru bi}e opisano u daljem tekstu.

3.2. Program MONITOR

Ovaj program se mo`e startovati jednim klikom na *Monitor* ikonu u MCWIN prozoru ili startovanjem programa *Monitor.exe* iz komandne linije. Startovanjem ovog programa otvara se novi prozor u DOS modu u kome se, ako je komunikacija PC sa MCK240 plo~om ispravna, ispisuje slede}a poruka o komandama:

```
Help menu. Commands:
H/h/? = Help
D = Display program memory
d = Display data memory
M = Modify program memory
m = Modify data memory
F = Fill program memory
f = Fill data memory
S = Save program memory on file
s = Save data memory on file
i = Inspect data memory location
a = Adjust data memory location
x = Set/reset XON/XOFF handshake
p = Set/reset checksum protocol
b = Set serial baud rate
c = Configure DARAM B0 as PM/DM
L = Download program in Flash
```

Postoje dva razli~ita na~ina rada MONITOR programa:

- Normalan na~in rada, kada se na dnu prozora mo`e videti poruka:

```
PC Monitor for motion Control Kit MCK240. (c) Copyright Technosoft 1997
```

- Terminalni na~in rada, kada se mo`e videti poruka:

```
PC Monitor works as terminal
```

3.2.1. Komande programa MONITOR

Napomena za sve primere:
Boldovan tekst unosi korisnik, pri ~emu su **<CR>** = Enter i **<SP>** = Space tasteri sa tastature.

H/h/?

Prikazuje *help* tekst na ekranu PC.

D/d

Prikazuje sadr`aj programske memorije/memorije za podatke - izme|u startne i krajnje adrese koje zadaje korisnik.

Primer: Prikazivanje sadr`aja memorije za podatke, izme|u adrese 200h i 210h.

```
DSP>d<CR>
  Display data memory
  From [hexa]: 200<SP> to 210<SP>
200: 0000 0000 0000 0000 0000 0000 0000 0000
210: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
DSP>
```

M/m

Modifikuje sadr`aj programske memorije/memorije za podatke - od adrese koja se specificira pokretanjem ove komande. Kada se pritisne **Enter**, **Space** ili, memorijska lokacija se ne menja. Izlaz iz komande se posti`e pritiskanjem **Esc** ili **.**.

Primer: Modifikovanje sadr`aja memorije za podatke na adresi 209h.

```
DSP>m<CR>
  Display data memory
  From [hexa]: 209<CR>
209: 005F New value: 25A7<CR>
20A: 005D New value: <CR>
20B: 035C New value: .
DSP>
```

F/f

Popunjava sadr`aj programske memorije/memorije za podatke - izme|u startne i krajnje adrese.

Primer: Upisivanje vrednosti 5555h u programsku memoriju od adrese 200h do 210h.

```
DSP>F<CR>
  Fill program memory
  From [hexa]: 200<SP> to 210<SP> with value [hexa]: 5555<SP>
DSP>
```

S/s

Sme{ta sadr`aj programske memorije/memorije za podatke - izme|u startne i krajnje adrese u *file* ~ije ime odre|uje korisnik.

Primer: Sme{tanje memorije za podatke od adrese 200h do 210h u *savefile.dat.*

```
DSP>s<CR>
  Save data memory
  From [hexa]: 200<SP> to 210<SP> to file: savefile.dat<CR>
DSP>
```

i

Neprekidno ~ita i prikazuje sadr`aj memorije za podatke na zadatoj adresi. Kraj komande posti`e se pritiskom bilo kog tastera na tastaturi.

Primer: Provera vrednosti memorije za podatke na adresi 200h.

```
DSP>i<CR>
  Inspect data memory
  From [hexa]: 200<SP>
01A4
DSP>
```

a

Neprekidno ~ita i prikazuje sadr`aj memorije za podatke na zadatoj adresi i omogu}uje izmenu sadr`aja na istoj lokaciji. Tasterima ← i → se bira cifra koja se `eli promeniti, a tasterima ↑ i ↓ se izabrana cifra inkrementira i dekrementira za 1. Kraj komande posti`e se pritiskom tastera **Enter**.

Primer: Provera vrednosti memorije za podatke na adresi 200h.

```
DSP>a<CR>
  Adjust data memory
  From [hexa]: 200<SP>
01A4
DSP>
```

x

Omogu}ava i zabranjuje XON-XOFF *handshake* protokol tokom razmene podataka serijskom vezom.

Primer: Omogu}avanje XON-XOFF *handshake* protokola.

```
DSP>x<CR>
  XON-XOFF handshake [1:enabled/0:disabled]
  Actual = 0 New value: 1<CR>
  XON-XOFF handshake is enabled
DSP>
```

p

Omogu}ava i zabranjuje MCK240 *checksum&acknowledgement* protokol. Kada je protokol omogu}en svaka grupa podataka koja se {alje na MCK240 se zavr{ava sa *checksum*. Posle prihvatanja svih podataka DSP plo~a {alje *acknowledge* karakter o korektnosti prijema podataka. Ista procedura se koristi kada se podaci {alju sa MCK240 na PC ra~unar. Ovaj protokol detektuje gre{ke u komunikaciji i poku{ava da ih automatski otkloni, a ukoliko to ne mo`e javlja poruku o gre{ci. Ovaj protokol garantuje celokupnost i originalnost razmenjenih podataka, ako nema poruke o gre{ci. Me|utim to smanjuje brzinu prenosa za oko 30 %.

Primer: Omogu}avanje *checksum* protokola.

```
DSP>p<CR>
  Checksum protocol [1:enabled/0:disabled]
  Actual = 0 New value: 1<CR>
  Checksum protocol is enabled
DSP>
```

b

Omogu}ava promenu brzine razmene podataka serijskom vezom. Brzina se istovremeno menja i na PC i na MCK240 plo~i.

Primer: Postavljanje brzine razmene podataka serijskom vezom na 19200 *baud.*

```
DSP>b<CR>
  Set baud rate: 9600 | 19200 | 38400
  Actual = 9600 New value: 19200<CR>
  XON-XOFF handshake is enabled
DSP>
```

c

Konfiguri{e blok B0 internog DARAM TMS320F240 kao memoriju za podatke ili programsku memoriju.

Primer: Postavljanje bloka B0 kao memorije za podatke

```
DSP>c<CR>
  Configure DARAM block B0 [0=PM/1=DM]: 1<CR>
DSP>
```

L

Sme{ta DSP program u *Flash* EEPROM memoriju 'F240 kontrolera. U *Flash* memoriju DSP mo`e se smestiti standardna COFF datoteka (ekstenzija *.out*). Jednom sme{ten program u ovu memoriju ostaje tu sve dok se ne upi{e novi program. Upis se vr{i u dva koraka: prvo se program sme{ta u eksterni RAM, a zatim u *flash* memoriju.

Primer: Sme{tanje datoteke *myfile.out* u *flash* memoriju.

```
DSP>L<CR>
  Download COFF object file in Flash memory
  File name (and extension): myfile.out<CR>
  Previous data from flash segment 3 (1800h-1fffh) will be lost.
  Continue? [y/n]y
  Download flash routines ...
0ABC
  Send flash segments mask
  Flash clearing ...  done
  Flash erasing ...   done
  Send program code & data
127B
  Flash coding ...    done
  Flash boosting ...  done
  Program downloaded OK in flash memory
  Entry point = 182E
DSP>
```

Primedbe:

1) Program koji se sme{ta u *flash* memoriju mora biti u odgovaraju}em adresnom opsegu (izme|u 1000h i 3fffh)

2) 'F240 DSP kontroler ima *flash* memoriju sa 8 segmenata od 2 kw svaki, u koje se nezavisno mo`e upisivati i brisati. Prva dva segmenta (tj. 4 kw) su rezervisana za MCK240 MONITOR i ne mogu se koristiti za programiranje (adrese 0000h-0fffh).

3) **L** komanda se mo`e koristiti za upis programa u *flash* memoriju, ali ne i za startovanje programa. Za izvr{avanje programa koristi se **r** komanda koja koristi startnu adresu programa dobijenu pomo}u **L** komande.

4) Tokom upisa programa u *flash* memoriju omogu}eni su XON-XOFF *handshake* i MCK240 *checkusm&acknowledge* protokoli. Posle upisa vra}aju se po~etna stanja. Sve ove modifikacije se izvr{avaju automatski i daje se vidljivi signal korisniku kada su zavr{ene.

5) Mo`e se testirati program upisan u *flash* memoriju i bez veze sa PC ra~unarom. Program se mora upisati na startnu adresu 1000h, a na konektoru J1 spojiti pinovi 2-3. U ovom slu~aju, 'F240 monitor MON240 automatski ska~e na adresu 1000h i program }e biti izvr{en.

```
l
```

Sme{ta DSP program u internu ili eksternu RAM memoriju. U RAM memoriju DSP mo`e se smestiti standardna COFF datoteka (ekstenzija *.out*).

Primer: Sme{tanje datoteke *myfile*1*.out* u RAM memoriju.

```
DSP>l<CR>
  Download COFF object file in RAM memory
  File name (and extension): myfile1.out<CR>
0ABC
  Program downloaded OK in RAM memory
  Entry point = 8071
DSP>
```

Primedbe:

1) Program koji se sme{ta u RAM memoriju mora biti u odgovaraju}em adresnom opsegu (izme|u 8000h i 0ffffh za eksternu RAM memoriju i izme|u 300h i 35fh, i 60h i 7fh, za internu RAM memoriju)

2) **l** komanda se mo`e koristiti za upis programa u RAM memoriju, ali ne i za startovanje programa. Za izvr{avanje programa koristi se **r** komanda koja koristi startnu adresu programa dobijenu pomo}u **L** komande.

3) Tokom upisa programa u RAM memoriju omogu}eni su XON-XOFF *handshake* i MCK240 *checkusm&acknowledge* protokoli. Posle upisa vra}aju se po~etna stanja. Sve ove modifikacije se izvr{avaju automatski i daje se vidljivi signal korisniku kada su zavr{ene.

r

Startuje program prethodno upisan u *flash* ili RAM memoriju.
Primer: Izvr{avanje programa koji startuje sa adrese 250h.

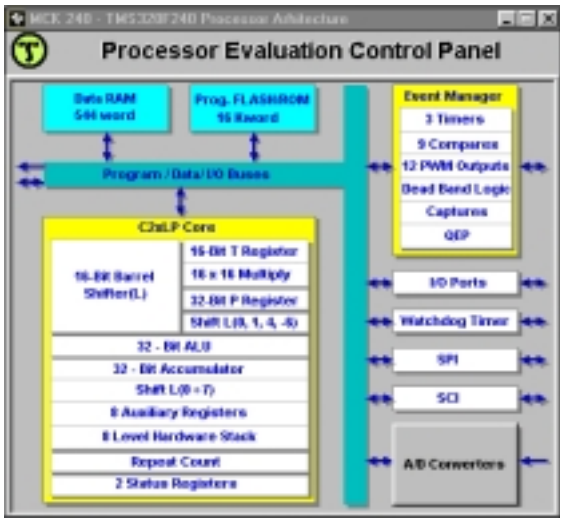
```
DSP>r<CR>
 Start address [hexa]: 250<CR>
DSP>
```

q

Izlazak iz MONPC *monitor* programa. Kada se pozove ova komanda, prvo se postavljaju *default* vrednosti za MCK240-PC serijsku vezu (brzina serijske veze 9600, 8 bita za podatke, 1 stop bit, bez parnosti, bez XON-XOFF *handshake* protokola, bez MCK240 *checksum&acknowledge* protokola) i onda se zatvara prozor *monitor* programa. Ukoliko su se *default* vrednosti za serijsku vezu menjale tokom rada u monitor programu, MCK240 plo~a se mora resetovati prilikom slede}eg uklju~enja.

3.3. PROCESORSKE APLIKACIJE

PROCEV program se startuje klikom na ikonu *Processor evaluation* ili startovanjem sa komandne linije programa *Procev.exe*. Ovaj program omogu}ava izvr{avanje jednostavnih aplikacija, omogu}avaju}i programiranje i testiranje osnovnih *I/O* funkcija na *DSP* ~ipu (tajmeri, *PWM*, prihvatni registri, *QEP*, *GPIO*, *SCI*, *SPI*, *watchdog* i *RTI*, *A/D* konvertori). Startovanjem *PROCEV* programa na monitoru *PC* }e se pojaviti slede}a slika:

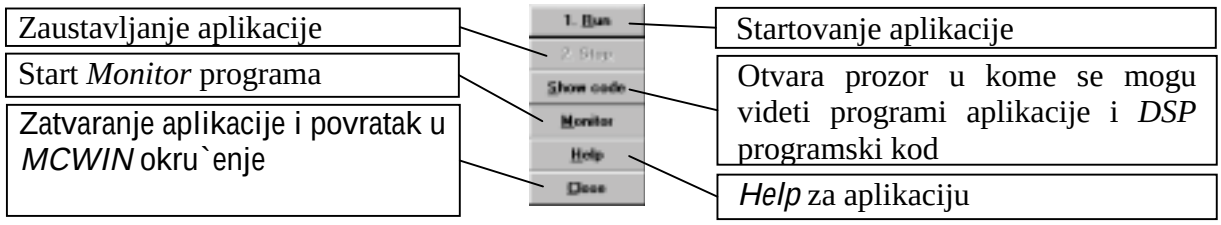


Slika 2. Prozor *PROCEV* programa

Sa datog prozora mogu se startovati aplikacije koje opisuju funkcionisanje slede}ih *I/O* funkcija *DSP*:

1) *A/D* konverzija
2) Prihvatni registri
3) Digitalni *I/O* portovi
4) Generisanje impulsa pomo}u *PWM*
5) *QEP* kola za pravougaone enkoderske impulse
6) *SCI* modul za serijsku komunikaciju
7) *SPI* modul za serijsku vezu sa periferijom
8) *GPT* tajmer op{te namene
9) *WD watchdog* i *RTI real time interrupt* moduli
10) Sme{tanje programa u *flash* memoriju
11) Sme{tanje programa u *RAM* memoriju

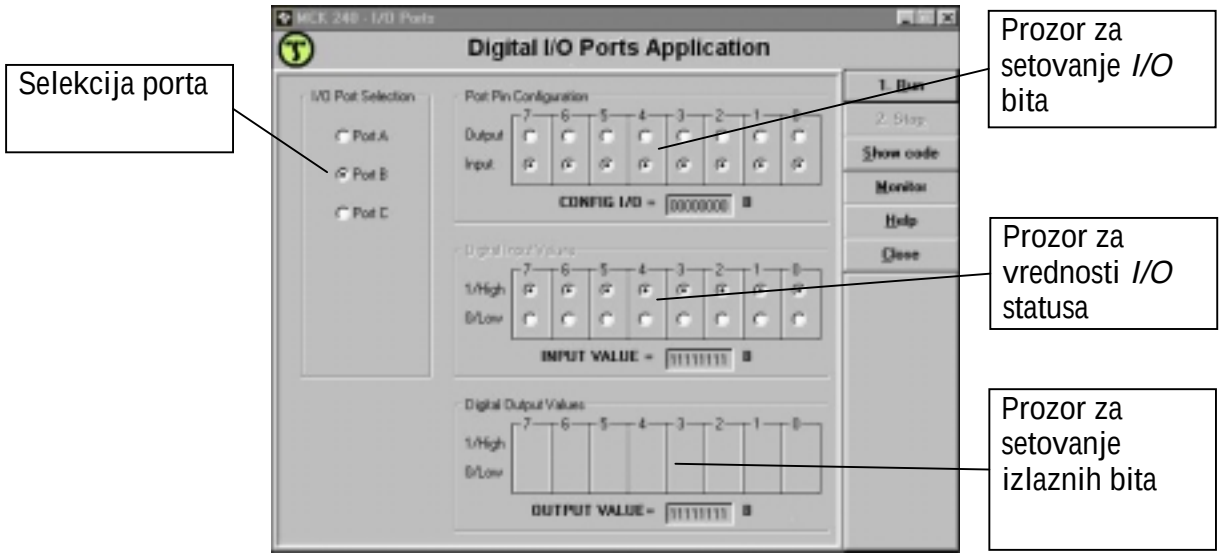
U gornjem desnom uglu svake aplikacije postoje kontrolni tasteri koji su prikazani na slede}oj slici:



Slika 3. Komandni aplikacioni tasteri

3.3.1. Digitalni I/O portovi

Ova aplikacija omogu}ava konfigurisanje 8 nezavisnih digitalnih I/O - porta B, dok je aktiviranje setovanih digitalnih izlaza pra}eno uklju~ivanjem odgovaraju}ih led dioda. Ova aplikacija se startuje pritiskom na ikonu ***I/O Ports*** u ***Processor Evaluation Control Panel*** prozoru. Na slede}oj slici je dat prikaz ove aplikacije:



Slika 4. Prozor aplikacije za *I/O* registre

Asemblerski program koji sadr`i kod ove aplikacije je ***gpio_a.asm*** (...\Mcwin\Demos\Gpio) i mo`e se videti pritiskom na taster ***Show code***. Zaglavlje asemblerskog programa je ***gpio_a.h*** koje sadr`i promenljive i funkcije za ovu aplikaciju. Startna adresa programa je **8000h** u eksternoj programskoj memoriji i zove se ***start***. Programske promenljive su i: _stop, _config_io, _input_io, _input_value,_output_value; One su eksternoj programskoj memoriji i po~inju od adrese **a000h.** U slu~aju ponu|enog asemblerskog programa ***gpio_a.asm*** va`i:

_stop = a000h ... program radi sve dok najni`i bit sadr`aja ove adrese ne postane 1. Zato se na ovu adresu pri startovanju programa upisuje 0, a program se prekida upisivanjem 1. Dakle, radi se o indeksu za stopiranje programa (1 za kraj programa).

_config_io = a001h ... sadr`aj ove adrese odre|uje (treba da zada) koji }e pinovi porta B biti INPUTI (odgovaraju}i bit=0) ili OUTPUTI (odgovaraju}i bit=1). Odnosno, ni`i bajt sadr`aja ove adrese odre|uje funkciju pinova na portu B (0-izlaz ili 1-ulaz).

_input_value=a002h ... ni`i bajt sadr`aja ove adrese je 8-bitni broj pro~itan sa porta B. Prisustvo High napona na pinu se ~ita upisivanjem vrednosti 0 na mesto odgovaraju}eg bita (Suprotno obja{njenju u helpu aplikacije!). S druge strane, bit je 1 – ako je na odgovaraju}em pinu Low nivo napona.

_output_value=a003h ... sadr`aj ove adrese je uvek 0FFh i mo`e biti promenjen kori{}enjem programa Monitor. Ideja je da se preko _output_value setuje High nivo napona na svim pinovima koji su preko _config_io definisani kao digitalni Outputi. Vrednost bita 1 je samo uslov da na odgovaraju}em pinu bude setovan High nivo napona. S druge strane, bit vrednosti 0 (na adresi _output_value) nikako ne mo`e setovati High nivo napona na odgovaraju}em pinu. To se mo`e testirati: jer bitX=0 (na adresi _output_value) mo`e biti upisan putem programa Monitor.

*Sadr`aji _config_io, _input_value, _output_value su obja{njeni na osnovu helpa i rada ponu|enog programa. Zapravo, ideja je da: 1) bitovi odre|eni sa _config_io defini{u da li su pinovi porta ulazi ili izlazi, 2) bitovi odre|eni sa _input_value sadr`e informaciju o stanju na pinovima porta B (da li su izlazi ili ulazi aktivirani ili ne), 3) bitovi odre|eni sa _output_value

setuju specificirane izlaze. Prakti~no, sadr`aj _output_value je uvek 0FFh, i kao takav treba da uti~e na setovanje High nivoa na svim pinovima porta B koji su setovani kao digitalni Outputi.

Ilustracije radi (razmatra se koncept ponu|enog programskog re{enja):
Led dioda je uklju~ena <=> vrednost odgovaraju}eg bita na adresi:

_config_io je 1
_output_value je 1
_input_value je 0

Stavljanjem komentara (;) ispred XOR (ekskluzivno ILI) setuje se 1 na bitu u _input_value; Odnosno, vrednost odgovaraju}eg bita na adresi:

_config_io je 1
_output_value je 1
_input_value je 1

i tada je led dioda isklju~ena. Za{to?

Evo o ~emu se radi:

Prakti~no 3 bita odre|uju funkcionisanje 1 pina na portu i to:

- odgovaraju}i bit u Output Control Register-u setuje da li }e pin imati funkciju digitalnog I/O (recimo, da bi pin bio digitalni I/O – vrednost odgovaraju}eg bita treba da je 0, jer u suprotnom pin }e izvr{avati svoju primarnu funkciju).
- odgovaraju}i bit u vi{em bajtu I/O port Data & Direction Register-a setuje da li }e pin biti digitalni Input (0) ili digitalni Output (1). Setovanje ovog bita nema zna~aja ukoliko pin nije digitalni I/O (odnosno, prethodno pomenuti bit nije ispravno setovan).
- odgovaraju}i bit u ni`em bajtu I/O port Data & Direction Register-a setuje da li }e na pinu biti setovan High/Low nivo napona (ukoliko se radi o digitalnom Outputu), ili se u njega upisuje sadr`aj (0/1) na osnovu “pro~itane vrednosti” High/Low nivoa napona na pinu (ukoliko se radi o digitalnom Inputu). Dakle, setovanje ovog (tre}eg) bita podrazumeva ispravno setovanje gore 2 pomenuta bita.

Zato, nu`ni koraci u na{oj aplikacije su:

- Upisivanje 0 u sve bitove vi{eg bajta Output Control Register-a A (nalazi se na adresi OPCRA=07090h). Na taj na~in, pinovi B0-B7 porta B }e funkcionisati kao digitalni I/O.
- Upisivanje sadr`aja adrese _config_io u vi{i bajt I/O port B Data & Direction Register-a (nalazi se na adresi PBDATDIR=0709Ah). Upisana vrednost 0/1 bita setuje da odgovaraju}i pin bude digitalni Input ili digitalni Output.
- Setovanje sadr`aja ni`eg bajta I/O port B Data & Direction Register-a (nalazi se na adresi PBDATDIR=0709Ah). I ovde treba obratiti pa`nju po{to **NIJE** kako pi{e u helpu ili kako stoji u SPRU161A.PDF. ZAPRAVO: Da bi na pinu koji je setovan kao digitalni Output setovali High nivo napona – odgovaraju}i bit treba da ima vrednost 0 (u suprotnom je 1). U odgovaraju}i bit se upisuje vrednost 0 ako je na pinu High nivo napona i ako se radi o digitalnom Inputu.

Pre nego {to pre|emo na obja{njenje ponu|enog asemblerskog programa, uraditi slede}u ve`bu:

- *Digital I/O Ports Application* niti *Processor Evaluation Control Panel* nisu aktivirani.
- Aktivirati program Monitor.
- U programu Monitor proveriti sadr`aj adresa: 07090h (OPCRA) i 0709Ah (PBDATDIR) – komanda: d
- Na adresu 07090h upisati 0000h (ako to ve} nije tako), a na adresu 0709Ah upisati 0F000h (recimo) – komanda: m
- [ta se de{ava? Pratiti sadr`aj adrese 07090Ah (komanda: i) i istovremeno aktiviranjem tastera uklju~iti neke od led dioda. Eksperimentisati promenom sadr`aja adrese 07090Ah.

Sledi obja{njenje ponu|enog asemblerskog programa *gpio_a.asm*

(Listing programa je u boji, a obja{njenja i primedbe nisu)

```
;*****************************************************************************
; File Name:        gpio_a.asm
; Project:     MCK240
; Originator:  R.Giuclea
; Description:      ASM file for GPIO demo (digital I/O)
; Copyright © 1997 Technosoft
;*****************************************************************************
; Include Files
;-------------------------------------------------- ...komentari ...
                .include    ..\F240_a.h
                .include    ..\demos_a.h
                .include    gpio_a.h      ...program uklju~uje sadr`aj zaglavlja F240_a.h, demos_a.h i gpio_a.h;
;=============================================================================
                .text                                   ... .text, .include, .global, itd. su asemblerske direktive. Tipi~no, direktive su korisne da inicijalizuju ili rezervi{u memoriju, kontroli{u izgled listinga, uslovno objedine blokove koda, defini{u globalne ili eksterne promenljive, itd. Direktive ne koriste memorijski prostor u kona~nom object modulu. To je zato jer se one samo koriste da kontroli{u asembler.
...ova direktiva inicijalizuje sekciju koja sadr`i sve izvr{ne kodove - koji }e uvek biti sme{teni u programsku memoriju. Instrukcije asemblera koje slede ovu direktivu }e biti sme{tene u memorijsku sekciju pod nazivom “text”.
; ----------------------------------------------------------------------------
_start:                                 ... labela
...sledi deo programa koji se odnosi na inicijalizaciju sadr`aja na adresi _stop:
        LDP     #_stop                  ... _stop = a000h (pi{e u  gpio.map).
                                        ...Primedba: pro~itati direktno adresiranje: SPRU160A.PDF, strana 142-146. Prakti~no, setovan je data page pointer DP=320. Naime, starijih 9 bita a000h odre|uju vrednost pointera DP, a mla|ih (najni`ih) 7 bita a000h odre|uju redni broj adrese na strani koju pokazuje pointer DP.
                                        ... Zna~i, u memoriji za podatke odabrali smo stranu 320 (od mogu}ih 512 strana: 0-511) na kojoj mo`e biti sadr`ano 128 re~i; odnosno, adrese od a000h do a07Fh su nam dostupne; i odabrali smo prvu re~ na toj strani.
        SPLK    #0,_stop                ... SPLK instrukcija omogu}ava da svih 16 bita budu neposredno upisani u memorijsku lokaciju za podatke. Ovde je 0 taj 16 bitni podatak koji se upisuje na adresu a000h (jer je _stop=a000h)
                                        ... Zna~i, setuje se 0 kao sadr`aj adrese _stop, i to je sve tako dok se klikom na Stop u Digital I/O Ports Application ne setuje 1 kao sadr`aj adrese _stop ({to je uslov za izlaz iz programa). Mogu}e je da se setovanje 1 na najni`em bitu adrese _stop izvr{i i kori{}enjem programa Monitor (recimo, uraditi za ve`bu!).
; init GPIO                             ... sledi deo programa koji defini{e pinove porta B (B0-B7) kao digitalne I/O. Zatim se setuje koji su pinovi Inputi, a koji Outputi.
;
        LDP     #DP_PF2                 ; Peripheral File 2 Data Page Pointer
                                        ... DP_PF2 je =0E1h i ta vrednost je setovana u F240_a.h; Zna~i, u memoriji za podatke smo sada na strani 225; tj. inicijalizovana je nova vrednost data page pointera (DP=0E1h).
        LACC    OPCRA                   ... OPCRA =07090h i to je setovano u F240_a.h, kao adresa Output Control Register-a A. Sadr`aj te adrese (stanja izvesnih bitova) defini{e da li }e i koji pinovi imati ulogu digitalnih I/O. Zapravo, ovo je I/O Mux Control Register.
                                        ... Primedba: pro~itati 11.4. odeljak SPRU161A.PDF, strana 304, 307. Tu pi{e da ukoliko bitovi 8-15 u I/O Mux Control Register-u (koji se nalazi na adresi OPCRA =07090h) su 0, tada pinovi porta B (B0-B7) postaju digitalni I/O.
                                        ... LACC naredba spu{ta sadr`aj sa adrese 07090h (OPCRA) u akumulator; (shiftovanje nije definisano iako je deo sintakse – tj. nema shiftovanja u konkretnom slu~aju); (7090h=225*128+16, TJ. 7 najni`ih bita u 07090h je zapravo 16dec, a slede}ih 9bita je E1h ili 225dec)
        AND     #0FFh           ; configure port B as digital I/O (reset bits 8-15)
                                        ... Pro~itati neposredno adresiranje: SPRU160A.PDF, strana 139-141.
                                        ... konstanta 0FFh predstavlja 16bitnu re~ i sprovodi se operacija logi~ko I nad svakim bitom te re~i i odgovaraju}im bitom u akumulatoru. Kako je 0FFh= 0000000011111111 (ni`i bajt ~ine jedinice, a vi{i nule), to zna~i da zapravo resetujemo bitove 8-15 u akumulatoru.
                                        ... Ukoliko su resetovani bitovi 8-15 na adresi OPCRA <=> port B je konfigurisan kao digitalni I/O (ali mi smo u ACC, a ne na adresi OPCRA!!!). Mada, kada se resetuje DSP, bitovi 8-15 i 0-3 u I/O Mux Control Register-u A se tako|e resetuju, pa zato program radi. Recimo: Zaustaviti rad programa i u programu monitor setovati sadr`aj 0FF00h na adresu OPCRA (=07090h). Startovati program. Prime}ujete – program ne radi!
Zato u ponu|eni program treba dodati naredbu:
        SACL    OPCRA                   ... - ~iji }e smisao biti kasnije komentarisan u listingu.
;
```

LDP #_config_io ... _config_io =a001h (videti gpio.map) i sadr`aj ove adrese je 8-bitna promenljiva (videti korisni~ki interface za zadavanje ove promenljive u Digital I/O Ports Application) kojom se zadaje konfiguracija porta B – tj. koji pin (B0-B7) je Output, a koji Input (digitalni Input – 0, digitalni Output – 1).

... U memoriji za podatke biramo stranu odre|enu vredno{}u DP=320. (a001h ~ini vi{ih 9 bita (140h=320dec) i ni`ih 7 bita (1))

LACC _config_io,8 ... Sadr`aj na adresi _config_io =a001h se pomera ulevo za 8 mesta i spu{ta u akumulator (ACC).

OR #0FFh ... Nad bitovima 16-bitne konstante 0FFh i odgovaraju}ih bitova sadr`anih u ACC, izvr{ava se operacija logi~ko ILI i rezultat ostaje u ACC. Rezultat: vi{i bajt je promenljiva setovana u _config_io, dok su svih 8 bitova ni`eg bajta su jedinice.

LDP #DP_PF2 ...DP_PF2 je =0E1h i ta vrednost je setovana u F240_a.h; Zna~i, u memoriji za podatke smo sada na strani 225; tj. inicijalizovana je nova vrednost data page pointera (DP=0E1h=225).

SACL PBDATDIR ... PBDATDIR =0709Ah i to je setovano u F240_a.h, kao adresa za - I/O port B Data & Direction Register. Predvi|eno je da vi{i bajt (ovog registra) sadr`i sadr`aj adrese _config_io, ~ijih 8 bitova odlu~uju da li su pinovi porta B konfigurisani kao digitalni ulazi (bitX=0) ili izlazi (bitX=1). Ni`i bajt, tj. bitovi 7-0 odgovaraju bitovima iz vi{eg bajta (bit7 je u vezi sa bit15 (+8)), jer odre|uju da li }e: Ako je odre|eni pin izlaz (bit(X+8)=1) biti setovan Low/High (bitX=0 ili =1) nivo napona – isklju~en/uklju~en izlaz; odnosno, Ako je odre|eni pin ulaz (bit(X+8)=0) biti pro~itan Low/High nivo napona i na osnovu toga setovan bitX=0 ili =1.

Me|utim, tako pi{e u dokumentaciji, ali tako nije. ZAPRAVO, bi}e: Ako je odre|eni pin izlaz (bit(X+8)=1) mo`e biti setovan High (bitX=0) ili Low (bitX =1) nivo napona – uklju~en/isklju~en izlaz; odnosno: Ako je odre|eni pin ulaz (bit(X+8)=0) bi}e pro~itan Low/High nivo napona i na osnovu toga setovan bitX=1 ili =0.

... SACL instrukcija u principu kopira celokupni sadr`aj akumulatora (ACC) u izlazni shifter gde je mogu}e izvr{iti ovom naredbom pomeranje ulevo za 0-7 bita (ako je pomeranje za broj bita definisano, {to ovde nije slu~aj – zna~i nema pomeranja). Onda se kopira ni`ih 16 bita u memoriju za podatke. Vrednost u ACC ostaje nedirnuta.

Dakle, u konkretnom slu~aju 16 ni`ih bita ACC kopiramo u programsku memoriju na adresu PBDATDIR, i tako dobijamo sadr`aj za I/O port B Data & Direction Register. Kako su u ni`em bajtu setovane sve jedinice – to zna~i da smo sve Outpute pogasili (tj. setovali Low nivo napona). Ve} u slede}em trenutku, sadr`aj ovog bajta se mo`e promeniti, jer }e bitovi koji odgovaraju Inputima sa High nivoom napona dobiti vrednost 0 umesto 1.

;

loop: ... labela...ulazimo u petlju jer imamo potrebu da stalno menjamo sadr`aj sa adrese PBDATDIR i ~itamo novi (da bi videli stanje Inputa i da bi eventualno a`urirali stanje na Outputima)

; output values

LDP #_config_io

LACC _config_io,8 ... sli~no prethodnom – sadr`aj adrese _config_io (koji nosi informaciju koji pinovi porta B }e biti digitalni Inputi ili Outputi) se shiftuje (pomera) ulevo za 8 mesta i spu{ta u akumulator (ACC).

OR _output_value ... _output_value=a003h (videti gpio.map) i sadr`aj ove adrese je 0FFh (videti Digital I/O Ports Application) i mo`e biti promenjen u programu Monitor. (Gde je setovan sadr`aj 0FFh?)

... kao posledica izvr{enja logi~kog ILI nad bitovima ACC i bitovima sadr`aja adrese _output_value, dobija se rezultat u ACC ~iji vi{i bajt je prakti~no sadr`aj _config_io, a ni`i bajt sadr`aj adrese _output_value.

LDP #DP_PF2 ... DP_PF2 je =0E1h i ta vrednost je setovana u F240_a.h; Zna~i, u memoriji za podatke smo sada na strani 225; tj. inicijalizovana je nova vrednost data page pointera (DP=0E1h=225).

XOR #0FFh ... nad bitovima ACC i 0FFh=0000000011111111 se sprovodi operacija Ekskluzivno ILI (0 i 1 daju 1, dok 1 i 1, i 0 i 0 daju 0) i rezultat se sme{ta u ACC. Prakti~no, vi{i bajt ACC je ostao nedirnut (sadr`aj _config_io), a u svim bitovima ni`eg bajta su upisane 0.

SACL PBDATDIR ... kopira se sadr`aj ni`ih 16 bita ACC u memoriju za podatke na adresu PBDATDIR =0709Ah. Vrednost u ACC ostaje nedirnuta. Upisivanjem sadr`aja u PBDATDIR na svim pinovima koji su konfigurisani kao Outputi se postavlja High nivo napona po{to su svi bitovi u ni`em bajtu 0.

LACC PBDATDIR ... spu{ta se sadr`aj sa adrese PBDATDIR =0709Ah u akumulator. I to ne mora biti ista vrednost onoj koja je ve} bila u ACC (videti prethodnu liniju programa), jer se sadr`aj u PBDATDIR menja saglasno High/Low stanju Inputa. High nivo napona na Inputu upisuje 0 kao vrednost bita, a u suprotnom ostaje 1.

LDP #_input_value ... setuje se DP=_input_value

... _input_value=a002h (videti gpio.map) i u sadr`aj ove adrese trebalo bi da se upi{u bitovi koji bi trebalo da prate Low/High (1/0) stanje napona na odgovaraju}em pinu porta B. Odnosno, na osnovu Low/High stanju napona na pinu upisuje se vrednost bita 1/0 u _input_value. Ova promenljiva je sadr`ana u ni`em bajtu I/O port B Data & Direction Register-a (na adresi PBDATDIR =0709Ah).

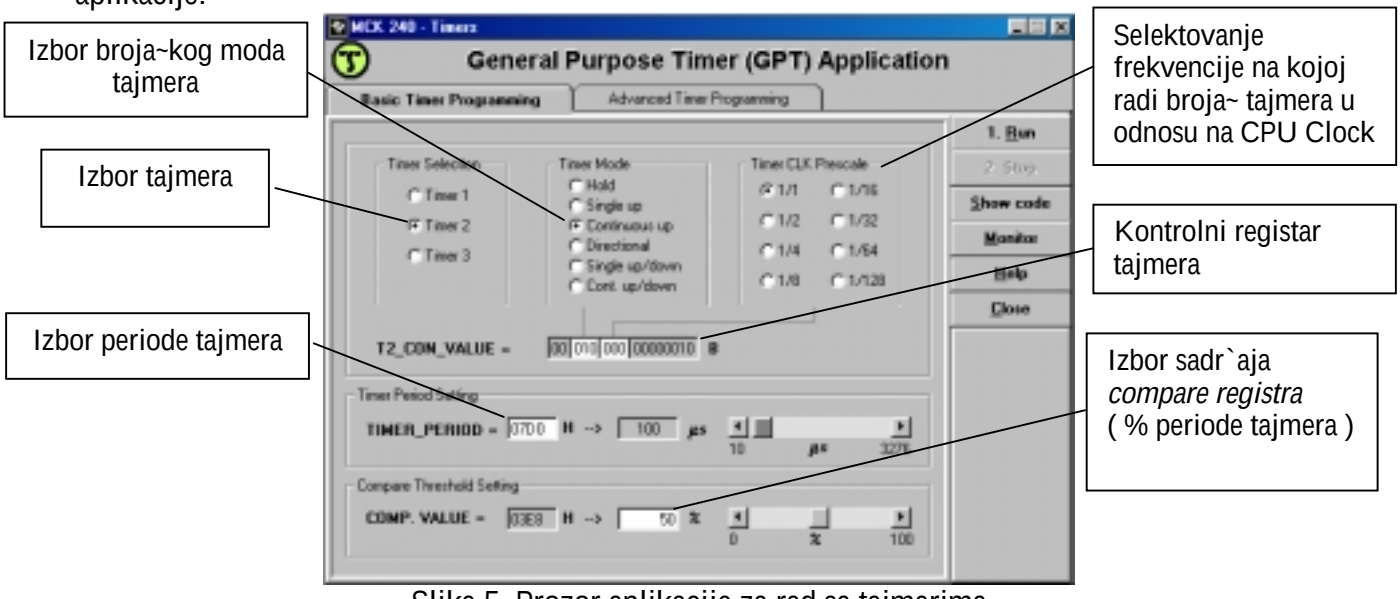
AND #0FFh ... konstanta 0FFh predstavlja 16bitnu re~ i sprovodi se operacija logi~ko I nad svakim bitom te re~i i odgovaraju}im bitom u akumulatoru. Kako je 0FFh= 0000000011111111 (ni`i bajt ~ine jedinice, a vi{i nule), to zna~i da zapravo resetujemo bitove 8-15 u akumulatoru.

```
        SACL    _input_value            ... kopira se sadr`aj ni`ih 16 bita ACC u memoriju za podatke na adresu
_input_value. Vrednost u ACC ostaje nedirnuta. Prakti~no, ni`i bajt sa adrese PBDATDIR se upisuje na adresu
_input_value.
;
; call monitor
        CALL    MON240                  ... Programski broja~ (PC-program counter) se inkrementira i stavlja na
vrh steka. Sadr`aj adrese u programskoj memoriji MON240=0109h (videti gpio_a.h). postaje sadr`aj programskog
broja~a. Poziva se program monitor.
; test if demo ends (_stop =1)
        LDP     #_stop                  ... setuje se DP=_stop = a000h
        BIT     _stop,15                ... posmatra se sadr`aj na adresi _stop. Specificirani bit code=15, odnosi
se na najni`i bit (LSB) na posmatranoj adresi. Instrukcija BIT kopira ovaj bit u TC bit status registra ST1. TC
(test/control flag bit) – je bit 11 status registra ST1 - ~uva rezultat operacije testiranja.
        BCND    loop,NTC                ... Ako je uslov NTC ispunjen (tj. ako je TC=0) program ide na loop
(labela – gore); Odnosno, ako je klikom na Stop u prozoru aplikacije setovano: _stop[0]=1 => izlazak iz petlje.
;
        END_DEMO                        ... END_DEMO je makro, definisan u Demos_a.h; Kada bude setovano
_stop[0]=1 (najni`i bit na adresi _stop da je 1), tj. da je TC=1, program prestaje da se izvr{ava.
```

3.3.2. Tajmeri

3.3.2.1. Osnove programiranja rada tajmera

Aplikacija ilustruje rad (mogu}nosti, pode{avanje parametara, programiranje) sa tajmerima op{te namene. Izabran je General Purpose Timer 2, i korisniku je ostavljeno da izabere parametre rada tajmera – na~in brojanja, interni DSP clock, period tajmera, poredbenu vrednost, i ilustrovano je konfigurisanje kontrolnog registra tajmera – T2CON (vrednost T2_CON_VALUE u komandnom prozoru aplikacije postaje sadr`aj registra T2CON). Omogu}eno je da compare izlaz tajmera aktivira nizak signal kada se desi poklapanje vrednosti broja~a i selektovane poredbene vrednosti (ovde izra`ene kroz procenat periode tajmera). Ova aplikacija se startuje pritiskom na ikonu ***Timers*** u ***Processor Evaluation Control Panel*** prozoru. Na slede}oj slici je dat grafi~ki prikaz ove aplikacije:



Slika 5. Prozor aplikacije za rad sa tajmerima

Klikom na *Run* eksperiment se startuje, ali da bi bio ispra}en koristi se osciloskop za vizuelizaciju promena koje nastaju na *compare output pinu* – T2CMP (pin 13 na J1_konektoru). Naime, rezultat rada aplikacije se sagledava kroz: generisani PWM signal zasnovan na *compare outputu* GP timera 2. Frekvencija signala mo`e da varira izme|u 10µs (clock CPUa/1 (prescale 1/1) i vrednost perioda tajmera c8h) i 0.4s (clock CPUa/128 (prescale 1/128) i vrednost perioda tajmera fff0h). Ciklus aktivnosti signala mo`e biti konfigurisan od 0 do 100%.

Asemblerski program koji sadr`i kod ove aplikacije je ***gpt_1_a.asm*** i mo`e se videti pritiskom na taster ***Show code***. Zaglavlje asemblerskog programa je ***gpt_1_a.h*** koje sadr`i promenljive i funkcije za ovu aplikaciju. Startna adresa programa je **8000h** u eksternoj programskoj memoriji i zove se **_start**. Programske promenljive su u eksternoj memoriji za podatke i po~inju od adrese **a000h**, po slede}em rasporedu: **_stop (=a000h), _t2_con_value (a001h), _timer_period (a002h), _compare_value (a003h).**

Va`ne adrese (kori{}ene u aplikaciji):

_stop = a000h – sadr`i indeks za stopiranje programa (1 za kraj programa)
_t2_con_value = a001h – sadr`i vrednost za konfiguraciju controlnog registra GP Timera 2
_timer_period = a002h – sadr`i vrednost koja }e biti upisana u GP timer 2 period_register
_compare_value = a003h – sadr`i vrednost koja }e biti upisana u GP timer 2 compare_register

Za pra}enje ove aplikacije, korisno je pro~itati:

1. SPRU160A.PDF, str. 7-1 do 7-16. – Na~ini adresiranja (neposredno, direktno, indirektno)
2. SPRU161A.PDF, str. 2-11 do 2-41. – Tajmeri

Tajmeri – neke op{te napomene

Na raspolaganju su 3 tajmera op{te namene: General Purpose Timer 1, 2 i 3 (GPT1, 2, i 3) – kao sastavni deo Event Manager (EV) modula kod TMS320C24x DSP kontrolera. Postoji i mogu}nost da tajmeri GPT2 i GPT3 u kaskadi zajedno obrazuju jedan 32-bitni tajmer.
Svaki od GP tajmera ima istu strukturu za koju mogu biti vezani odgovaraju}i ulazi i izlazi. Strukturu tajmera GP Timer x (x=1, 2, 3) odre|uju:

- 16-bitni *counter* registar TxCNT u kome je sadr`an broja~ tajmera.
- 16-bitni *compare* registar TxCMPR (GP Timer Compare Register) u kome je sadr`ana poredbena vrednost tajmera. Registar TxCMPR ~uva vrednost koja }e biti neprekidno upore|ivana sa broja~em tajmera (sadr`ajem registra TxCNT). Kada se poklapanje dogodi, de{ava se promena na pridru`enom *comapare* izlazu, saglasno odgovaraju}em bitu u GPTCON registru. Operacija pore|enja GP tajmera mo`e biti dozvoljena ili ne – setovanjem odgovaraju}eg bita u TxCON.
- 16-bitni *period* registar TxPR (GP Timer Period Register) u kome je sme{tena perioda tajmera. Rad GP tajmera (misli se na promenu broja~a tajmera) se zaustavlja i zadr`ava na teku}oj vrednosti, resetuje na 0, ili po~inje brojanje u suprtnom smeru (nadole, recimo) kada se desi poklapanje izme|u periode i broja~a tajmera (tj. sadr`aja registara TxPR i TxCNT), zavisno od setovanog moda brojanja u kome se tajmer nalazi.
- 16-bitni *control* registar TxCON, ~ijim sadr`ajem je definisan rad tajmera x
- programabilni predelitelj koji odre|uje vremensku bazu tajmera u odnosu na CPU clock (recimo, CPU generi{e clock od 20MHz, ali }e tajmer koristiti clock od (20/128)MHz)
- logi~ka jedinica za kontrolu i interrupte
- izlazna logi~ka jedinica
- 1 izlazni pin – TxPWM/TxCMP – *GP compare output pin*

Ulazi GP tajmera su:

- Interni CPU clock (ima istu frekvanciju kao i CPU clock – 20MHz u na{em slu~aju).
- Eksterni clock – TMRCLK (njegova maksimalna frekvencija je 1/4 od frekv. CPU clocka). Recimo, ovakav ulaz u tajmer mo`e proizvesti opti~ki enkoder sa QEP kolom. U slu~aju i internog i eksternog clocka, ulazni clock tajmera mo`e biti izveden na osnovu jednog od njih usvajanjem predelitelja, odnosno, frekvencija rada tajmera }e biti ista ili usvojeni broj puta (2, 4, ... 128 puta) manja od frekvencije internog ili eksternog clocka.
- Ulaz pravca (smera) brojanja – TMRDIR (broja~ tajmera mo`e brojati u rastu}em (TMRDIR=1) i opadaju}em poretku (TMRDIR=0)). Zapravo, radi se o ulaznom pinu

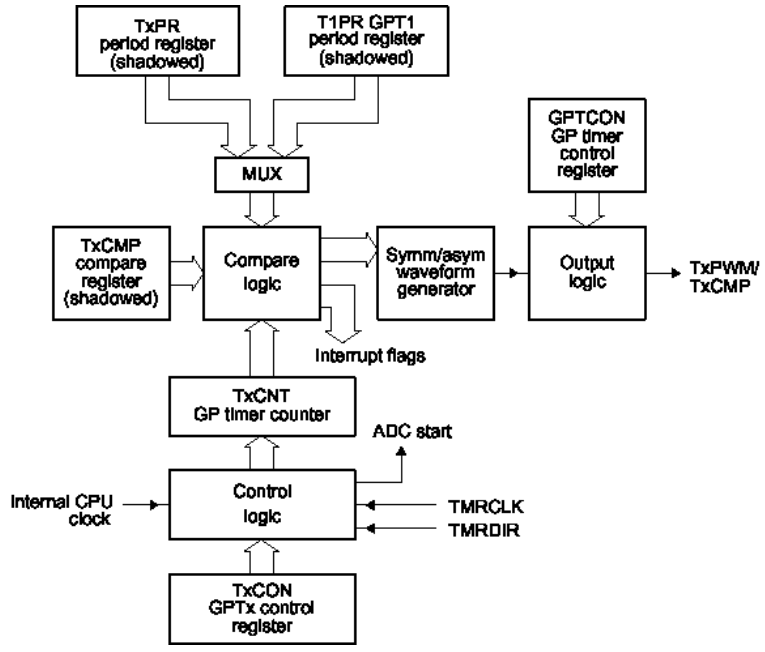
TMRDIR koji visokim ili niskim nivoom (ulaznog) signala odre|uje smer brojanja broja~a.

- Reset signal – RESET.

Izlazi GP tajmera su:

- Signal na pinu TxPWM/TxCMP – *GP timer* ***compare output*** (nastaje kao posledica pore|enja sadr`aja bitnih registara tajmera i na osnovu programirane logike tajmera). Zavisno od toga kako su konfigurisani bitovi registra GPTCON, *compare output* GP tajmera mo`e biti setovan kao *active high* (kada se broja~ tajmera **prvi put** poklopi sa vredno{}u u compare registru - signal **niskog** nivoa postaje signal **viskog** nivoa; a zatim kada (/ako) **drugi put** do|e do poklapanja pomenutih registara signal **visokog** nivoa postaje signal **niskog** nivoa), *active low* (suprotno od active high), *forced high* (odmah dobijamo signal visokog nivoa na izlazu), ili *forced low* (odmah dobijamo signal niskog nivoa na izlazu).
- Signal ka ADC modulu koji startuje AD konverziju (ADC start signal). Bitovi GPTCON registra mogu da specificiraju da ADC start signal bude generisan u trenutku kada do|e do poklapanja vrednosti broja~a tajmera sa vredno{}u FFFFh (overflow), 0000h (underflow) ili periodom tajmera (period match), ili pak poredbenom vredno{}u (compare match). Ova osobina obezbe|uje sinhronizaciju izme|u GP Timer "doga|aja" i starta AD konverzije, bez intervencije CPUa.
- Interrupt flag signal – odnosno, kada broja~ tajmera (sadr`aj TxCNT) se poklopi sa poredbenom vredno{}u tajmera (sadr`aj TxCMP) – (compare match); ili sa periodom tajmera (sadr`aj TxPR) – (period match), ili se desi da broja~ tajmera dostigne vrednost FFFFh (overflow), ili suprotno do|e do vrednosti 0000h (underflow) – programirana logika tajmera registruje odgovaraju}i "doga|aj" i upu}uje signal o tome. Postoji 12 interrupt flagova u EVIFRA i EVIFRB za tri GP tajmera. Svaki GP tajmer mo`e da generi{e 4 interrupta nad slede}im doga|ajima: *overflow* - TxOFINT (x=1, 2, ili 3), *underflow* - TxUFINT (x=1, 2, ili 3), *compare match* - TxCINT (x=1, 2, ili 3), i *period match* - TxPINT (x=1, 2, ili 3).
- Bit koji pokazuje pravac brojanja tajmera.

Kontrola rada tajmera se posti`e setovanjem bitova registara TxCON i GPTCON (GP Timer Control Register).

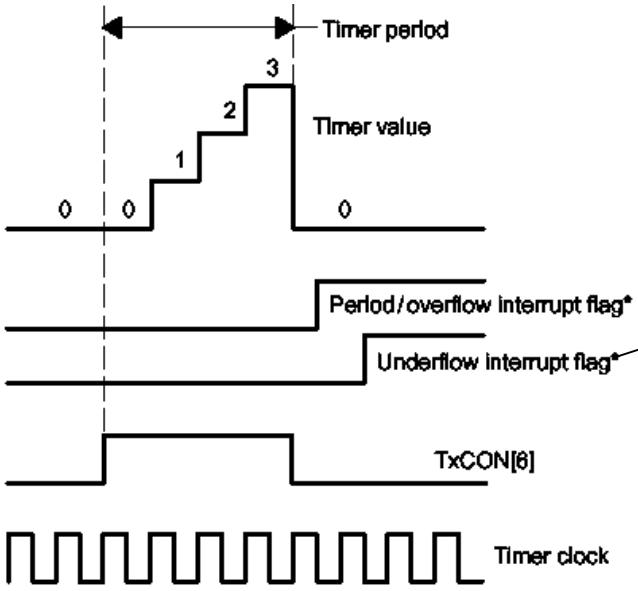


Sl. 6. *Struktura GP tajmera* (oznaka za registar "shadowed" zna~i: Nova vrednost u ove registre mo`e da bude upisana svakog trenutka u toku izvr{enja periode tajmera, **ALI U STVARI:** nova vrednost se upisuje u pridru`eni registar u senci (shadow registar), a tek kasnije se upisuje u

izvr{ne registre (u TxPR po isteku zapo~etog perioda – tj. kada sadr`aj TxCNT registra postane 0, a u TxCMPR u trenutku koji je specificiran sadr`ajem registra TxCON, a to mo`e biti: 1) odmah 2) kada se sadr`aj TxCNT izjedna~i sa sadr`ajem TxPR ili sa 3) nulom). – Ovo su va`ne osobine, posebno kada se imaju u vidu potrebe kod generisanja PWM signala.

Svaki od GP tajmera mo`e biti pode{en (izborom bitova u TxCON) da radi u jednom od slede}ih 6 (broja~kih) modova (rad tajmera po~inje setovanjem TxCON[6]=1 u zadatom broja~kom modu, dok u suprotnom, kada je TxCON[6]=0 - rad tajmera nije dozvoljen, broja~ se zaustavlja, a predelitelj resetuje na x/1):

1. Stop/Hold mod. – Rad GP tajmera se zaustavlja i dr`i na teku}em stanju (parametri tajmera ostaju nepromenjeni – broja~, compare output, predelitelj).
2. Single-Up counting mod. – GP tajmer broji u rastu}em poretku (navi{e), saglasno skaliranom ulaznom clocku, sve dok broja~ ne dostigne vrednost perioda tajmera. Onda se na slede}oj uzlaznoj ivici ulaznog clocka (odmah po izjedna~avanju sadr`aja TxCNT i TxPR) de{ava resetovanje broja~a tajmera na 0 i on prestaje sa radom (setuje se TxCON[6]=0, {to resetuje i predelitelj na x/1).



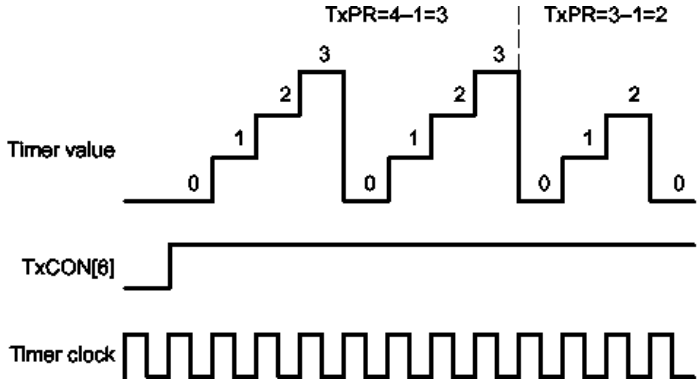
*Timing of interrupt flags is the same among all counting modes.

Sl. 7. *Ilustracija generisanja flaga u Single-Up Counting modu*

- promena vrednosti broja~a (Timer value – sadr`i se u TxCNT) se de{ava posle ispunjenja odgovaraju}eg uslova, a na uzlaznoj ivici clocka tajmera.

- 2 clock ciklusa tajmera po{to je broja~ na 0, setuje se Underflow int. flag.

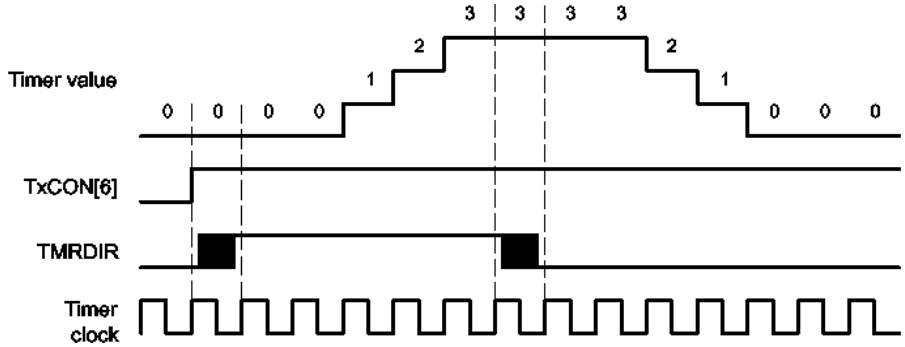
3. Continuous-Up counting mod. – Rad GP tajmera u ovom modu je isti kao i u single-up counting modu s tom razlikom da se proces ponavlja svaki put kada se broja~ tajmera resetuje na 0. GP tajmer u ovom modu broji saglasno skaliranom ulaznom clocku sve dok se vrednost broja~a tajmera ne izjedna~i sa periodom tajmera – tj. vredno{}u u period registru TxPR. Onda se resetuje broja~ tajmera na 0 i startuje slede}i period brojanja...



Sl. 8. *Continuous-Up mod*

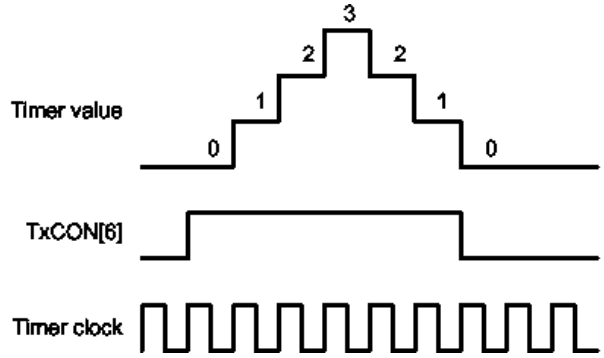
4. Directional-Up/Down counting mod. – U ovom modu tajmer }e brojati nagore ili nadole saglasno skaliranom clocku i ulazu TMRDIR. GP tajmer }e brojati nagore sve dok

njegova vrednost (tj. vrednost njegovog broja~a) ne dostigne vrednost perioda tajmera ili vrednost FFFFh (kada TMRDIR pin zadr`ava visok nivo signala). Kada se vrednost tajmera izjedna~i sa periodom tajmera ili sa FFFFh ukoliko TMRDIR zadr`ava visok nivo, tajmer }e se zadr`ati na toj vrednosti. S druge strane, GP tajmer }e brojati nadole sve dok njegova vrednost ne postane 0 kada TMRDIR zadr`ava nizak nivo signala. Kada je vrednost tajemra 0 i TMRDIR zadr`ava nizak nivo signala, tajmer }e se zadr`ati na nuli.



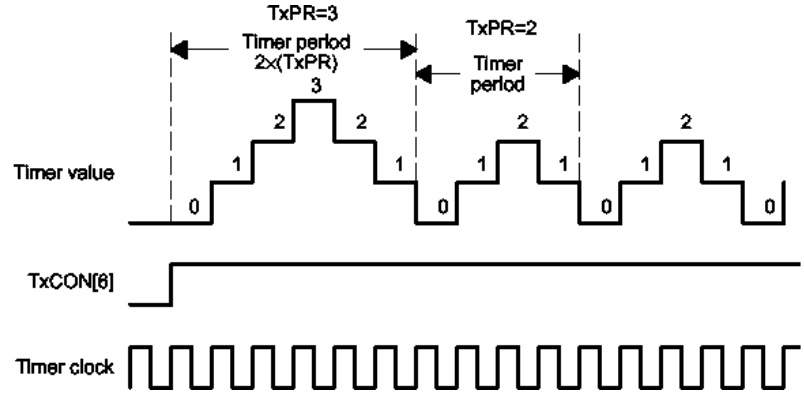
Sl. 9. *Directional-Up/Down counting mod*

5. Single-Up/Down counting mod. – GP tajmer u ovom modu }e brojati u rastu}em poretku (nagore) saglasno skaliranom ulaznom clocku do vrednosti perioda tajmera. Tada se menja smer brojanja tajmera i on broji nadole (u opadaju}em poretku) sve do nule. Kada tajmer do|e do nule, resetuje se - resetuje se TxCON[6]=0 (tajmer prestaje sa radom), resetuje se predelitelj, zaustavlja se njegov broja~ i zadr`ava teku}e stanje.



Sl. 10. *Single-Up/Down counting mod*

6. Continuous-Up/Down counting mod. – Ovaj mod rada tajmera je isti kao i single-up/down counting, samo {to se ponavlja periodi~no, svaki put kada broja~ tajmera do|e do nule. Kada se ovaj mod rada startuje, ne zahteva se softverska ili hardverska intervencija da se ponovi period brojanja. Period brojanja tajmera u ovom modu iznosi 2*(TxPR) ciklusa skaliranog ulaznog clocka, ne ra~unaju}i prvi period. Du`ina prvog perioda brojanja je ista samo ukoliko je broja~ na nuli kada brojanje po~inje (dvostruki sadr`aj registra TxPR, pa se to mno`i sa vremenskom bazom skaliranog ulaznog clocka u tajmer).



Sl. 11. *Continuous-Up/Down counting mod*

O svemu ovome detaljnije pogledati SPRU161A.PDF, str. 2-11 do 2-41.

Analiza ponu|enog programskog re{enja aplikacije

Ponu|eni asemblerski program *gpt_1_a.asm* je zasnovan na slede}em algoritmu:

1. Setuju se bitovi slede}ih registara:
   a. 0 se upisuje na adresu _stop=a000h. Kada najmla|i bit na ovoj adresi postane 1 (_stop[0]=1) tada program prestaje sa radom.
   b. 1 se upisuje u OPCRA[12] => pin 106 DSP-ija se vezuje za GP Timer 2 (tj. dobija funkciju: T2PWM/T2CMP). Videti na str. 11-14 u Spru161a.pdf kako se konfiguri{e Output Control Register A (OPCRA=7090h). (da je OPCRA[12]=0, isti pin bi imao I/O funkciju).
   c. 00 se upisuje u GPTCON[10-9] => GP Timer 2 ne}e startovati AD konverziju; GPTCON=7400h - General Purpose Timer Control Register (v. str. 2-38 u Spru161a.pdf).
   d. 1 se upisuje GPTCON[6] => dozvoljena su sva 3 GP Timer Compare Outputa. Zapravo, sada GPT2 mo`e da koristi *compare output*, odnosno sada mo`e da se pin T2PWM/T2CMP stavi u funkciju.
   e. 1 se upisuje GPTCON[2] => videti Spru161a.pdf str. 2-39) => GPT2 Compare Output (tj. signal na pinu T2PWM/T2CMP) je setovan kao Active low (kada se broja~ tajmera prvi put poklopi sa vredno{}u u compare registru T2CMP - signal viskog nivoa postaje signal niskog nivoa; a zatim kada (tj. ako – {to zavisi od selektovanog moda brojanja tajmera) drugi put do|e do poklapanja pomenutih registara signal niskog nivoa postaje signal visokog nivoa).
   f. (_timer_period=a002h)->(T2PER=7407h). Setuje se period GP Timera 2 tako {to se na adresu T2PER=7407h upisuje sadr`aj koji je korisnik izabrao u komandnom prozoru aplikacije (privremeno sme{ten na adresu _timer_period=a002h).
   g. 0 se upisuje na adresu T2CNT=7405h – broja~ GP Timera 2 se postavlja na 0. (bitno nam je da po~ne da broji od nule)
   h. (_compare_value=a003h)->(T2CMP=7406h). Setuje se sadr`aj Compare registra GPT2. Ovu vrednost bira korisnik u komandnom prozoru aplikacije (kao procenat periode tajmera) i tada se ona upisuje na adresu _compare_value=a003h. Sa ovom vredno{}u (ukoliko je T2CON[1]=1), neprekidno se upore|uje broja~ tajmera, i kada se desi poklapanje => menja se stanje signala na T2PWM/T2CMP pinu.
   i. (_t2_con_value=a001h)->(T2CON=7408h) - konfiguri{e se GP Timer 2. Pogledati Spru161a.pdf str. 2-37.
   j. 1 se upisuje u bit T2CON[6] => start GP Timera 2
2. petlja:
   a. Program Monitor se aktivira (u svakom ciklusu).
   b. Proverava se u svakom ciklusu da li je u najni`i bit adrese _stop=a000h upisana jedinica ({to se de{ava pritiskom na Stop u komandnom prozoru aplikacije). Ako jeste to je uslov za izlazak iz petlje. Ako ne, sledi povratak na po~etak petlje.
3. KRAJ

Detaljniji uvid se mo`e ste}i na osnovu ponu|enog asemblerskog programa *gpt_1_a.asm*, koji sledi u nastavku (listing programa je u boji, a obja{njenja i primedbe nisu).

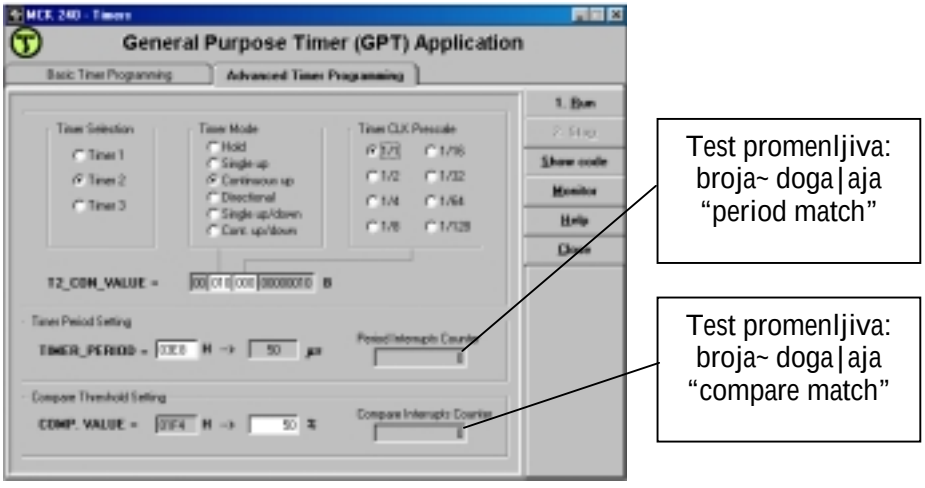
```
;===============================================================================
; File Name:            gpt_1_a.asm
; Project:      MCK240
; Originator:   R.Giuclea
; Description:          ASM file for GPT demo 1
; Copyright © 1997 Technosoft
;===============================================================================
;
; Include Files
;-------------------------------------------------------------------------------
```

```
                .include    ..\F240_a.h
                .include    ..\demos_a.h
                .include    gpt_1_a.h
;=====================================================================
                .text
; ---------------------------------------------------------------------
_start:                                 ... labela
        LDP     #_stop                  ... _stop = a000h (pi{e u  gpt_1.map). Starijih 9 bita a000h odre|uju vrednost data page pointera DP=320.
        SPLK    #0,_stop                ...setuje se 0 kao sadr`aj adrese _stop, i to je sve tako dok se klikom na Stop u komandnom prozoru aplikacije ne setuje 1 kao sadr`aj adrese _stop ({to je uslov za izlaz iz programa).
; configure timer shared pin
        LDP     #DP_PF2                 ; Peripheral File 2 Data Page Pointer
... DP_PF2 je = 0E1h (videti F240_a.h); Setovana je nova vrednost data page pointera (DP=0E1h=225dec). (Ovde imamo 8 - bitni broj (0E1h), a da je 16-bitni – vrednost DP bi bila odre|ena sa starijih 9 bita.)
        SETBIT OPCRA,SETB12             ... SETBIT je makro koji je definisan u Demos_a.h
... OPCRA = 7090h    (videti F240_a.h) – adresa Output Control Registera A; Sadr`aj te adrese defini{e ulogu odgovaraju}ih pinova – I/O ili drugu (vezanu za GP tajmere) (videti Spru161a.pdf, strana 307.)
... SETB12 = 1000h    ; Bit Mask for 12 (videti F240_a.h)
... Su{tina je da se nad bitovima sadr`aja adrese OPCRA sprovede operacija logi~ko ILI sa vredno{}u konstante SETB12 i taj rezultat se sme{ta opet na adresu OPCRA => OPCRA[12]=1 => T2PWM/T2CMP funkcija pina 106 DSP-ija (videti str. 11-4 u Spru161a.pdf).
        SACL    OPCRA           ; Shared T2PWM/T2CMP or IOPB4 pin configured for GPT2
... Sadr`aj akumulatora se upisuje na adresu OPCRA. ME\UTIM, ovo je gre{ka i ovu instrukciju bi trebalo izostaviti (sadr`aj ACC nije prethodno definisan, a {ta smo hteli, postigli smo makroom SETBIT).
; timer will not start ADC automatically
        LDP     #DP_EV                  ; Event Manager Data Page Pointer
... DP_EV je = 0E8h (videti F240_a.h); Setovana je nova vrednost data page pointera (DP=0E8h=232dec).
        LACC    GPTCON
... GPTCON (General purpose timer control register)=7400h (adresa je definisana u F240_a.h)
... Primedba: progledati Event Manager Module: SPRU161A.PDF
... Upisuje se sadr`aj adrese 07400h (GPTCON) u akumulator (ACC);
        AND     #AND_T2TOADC_           ; AND mask for DISABLING ADC start on GPT2
... u Demos_a.h definisana je konstanta AND_T2TOADC_      = 0F9FFh
... operacija logi~ko I se sprovodi nad bitovima sadr`aja adrese 07400h (GPTCON) i 0F9FFh – resetuje se bit 9 i 10 akumulatora.
        SACL    GPTCON                  ; configure GPTCON not to start ADC on GPT2 Event
... Sadr`aj akumulatora se upisuje na adresu 07400h (GPTCON) => GPTCON[10-9]=00 => GP Timer 2 ne}e startovati AD konverziju (videti Spru161a.pdf str. 2-38)
; enable GPT compare outputs and specify active state of GPT2 compare output
        SETBIT GPTCON,SETB6; active low         ... SETBIT je makro koji je definisan u Demos_a.h
... => GPTCON[6]=1 (videti Spru161a.pdf str. 2-39) => dozvoljena su sva 3 GP Timer Compare Outputa
        SETBIT GPTCON,SETB2; active low
... => GPTCON[2]=1 (videti Spru161a.pdf str. 2-39) => GPT2 Compare Output je low
; load and init timer
        LDP     #_timer_period          ... _timer_period=a002h (videti gpt_1.map), setuje se DP=320
        LACC    _timer_period           ... upisuje se sadr`aj _timer_period=a002h u akumulator; Sadr`aj _timer_period=a002h setuje korisnik u komandnom prozoru aplikacije, i to je perioda tajmera.
        LDP     #DP_EV
... DP_EV je = 0E8h (videti F240_a.h); Setovana je nova vrednost data page pointera (DP=0E8h=232dec).
        SACL    T2PER           ; load GPT2 timer period
... sadr`aj akumulatora se upisuje na adresu T2PER=7407h (videti F240_a.h), i tako defini{e period tajmera GPT2. Prakti~no: (_timer_period=a002h)->( T2PER=7407h).
        LACC    #0                      ... konstanta 0h se upisuje u akumulator
        SACL    T2CNT           ; reset GPT2 counter register
... sadr`aj akumulatora se upisuje na adresu T2CNT=7405h (videti F240_a.h) – i radi se o adresi T2 Counter Registera (sadr`aj je broja~ GP Timera 2). Prakti~no: 0->( T2CNT=7405h)
        LDP     #_compare_value         ... _compare_value=a003h (videti gpt_1.map), setuje se DP=320
        LACC    _compare_value          ... upisuje se sadr`aj _compare_value=a003h u akumulator; Sadr`aj _compare_value=a003h setuje korisnik u komandnom prozoru aplikacije, i to je compare vrednost tajmera.
        LDP     #DP_EV                  ... Setovana je nova vrednost data page pointera (DP=0E8h=232dec).
        SACL    T2CMP           ; load GPT2 timer compare register
... sadr`aj akumulatora se upisuje na adresu T2CMT=7406h (videti F240_a.h) – i radi se o adresi T2 Compare Registera. Prakti~no: (_compare_value=a003h)->( T2CMT=7406h)
```

```
        LDP     #_t2_con_value          ... _t2_con_value=a001h (videti gpt_1.map), setuje se DP=320
        LACC    _t2_con_value           ... upisuje se sadr`aj _t2_con_value=a001h u akumulator; Sadr`aj
_t2_con_value=a001h setuje korisnik u komandnom prozoru aplikacije – biraju}i mod brojanja tajmera i frekvenciju
rada tajmera.
        LDP     #DP_EV                  ... Setovana je nova vrednost data page pointera (DP=0E8h=232dec).
        SACL    T2CON           ; load GPT2 configuration register
... sadr`aj akumulatora se upisuje na adresu T2CON=7408h (videti F240_a.h) – i radi se o adresi T2 Control
Registera (videti Spru161a.pdf str. 2-36). Prakti~no: (_t2_con_value=a001h)->( T2CON=7408h)
; start timer
        SETBIT T2CON,SETB6   ;start GPT2        ... SETBIT je makro koji je definisan u Demos_a.h
... => T2CON[6]=1 (videti Spru161a.pdf str. 2-36) => startuje se GP Timer 2
;
loop:                                   ...labela (defini{e po~etak petlje)
; call monitor
        CALL    MON240                  ... Programski broja~ (PC-program counter) se inkrementira i stavlja na
vrh steka. Sadr`aj adrese u programskoj memoriji MON240=0109h (videti cap_a.h). postaje sadr`aj programskog
broja~a. Poziva se program monitor.
; test if demo ends (_stop =1)
        LDP     #_stop                  ... setuje se DP=320
        BIT     _stop,15                ... posmatra se sadr`aj na adresi _stop. Specificirani bit code=15, odnosi
se na najni`i bit (LSB) na posmatranoj adresi. Instrukcija BIT kopira ovaj bit u TC bit status registra ST1. TC
(test/control flag bit) – je bit 11 status registra ST1 - ~uva rezultat operacije testiranja.
        BCND    loop,NTC                ... Ako je uslov NTC ispunjen (tj. ako je TC=0) program ide na loop
(labela – gore); Odnosno, ako je klikom na Stop u prozoru aplikacije setovano: _stop[0]=1 => izlazak iz petlje.
;
        END_DEMO                        ... END_DEMO je makro, definisan u Demos_a.h; Kada bude setovano
_stop[0]=1, tj. da je TC=1, program prestaje da se izvr{ava.
```

3.3.2.2. Programiranje rada tajemra – korak napred

Ova aplikacija sadr`i u potpunosti prethodnu aplikaciju rada sa tajmerima i predstavlja njenu nadgradnju u smislu da pokazuje kako tajmeri op{te namene mogu generisati interrupte. Izabran je General Purpose Timer 2, i korisniku je ostavljeno da izabere parametre rada tajmera – na~in brojanja, interni DSP clock, period tajmera, poredbenu vrednost, i ilustrovano je konfigurisanje kontrolnog registra tajmera – T2CON (vrednost T2_CON_VALUE u komandnom prozoru aplikacije postaje sadr`aj registra T2CON). Omogu}eno je da compare izlaz tajmera aktivira nizak signal kada se desi poklapanje vrednosti broja~a i selektovane poredbene vrednosti (ovde izra`ene kroz procenat periode tajmera). U ovoj aplikaciji dozvoljeni su: *timer period* i *compare interrupt-i* (prekidi generisani u trenucima poklapanja vrednosti broja~a tajmera sa vredno{}u periode tajmera i poredbenom vredno{}u tajmera). U odgovaraju}im prekidnim rutinama inkrementiraju se vrednosti dve test promenljive. Ova aplikacija se startuje pritiskom na ikonu ***Timers*** u ***Processor Evaluation Control Panel*** prozoru. Na slede}oj slici je dat grafi~ki prikaz ove aplikacije:



Slika 12. Prozor aplikacije za rad sa tajmerima

Klikom na *Run* eksperiment se startuje, a vizuelizacija promena se prati u komandnom prozoru aplikacije sagledavaju}i promenu dve promenljive koje se inkrementiraju u svakom interaptu (videti text box: Period Interrupts Counter i Compare Interrupts Conter). Klikom na *Stop* aplikacija prestaje sa radom, ali ostaje zabele`ena vrednost test promenljivih (Period Interrupts Counter i Compare Interrupts Conter) – ove dve sa~uvane vrednosti mogu pomo}i korisniku da razume kako GP Timer 2 radi.

Asemblerski program koji sadr`i kod ove aplikacije je ***gpt_2_a.asm*** i mo`e se videti pritiskom na taster ***Show code***. Zaglavlje asemblerskog programa je ***gpt_2_a.h*** koje sadr`i promenljive i funkcije za ovu aplikaciju. Startna adresa programa je **8000h** u eksternoj programskoj memoriji i zove se **_start**. Postoje dve prekidne rutine koje po~inju sa **_t2per_ISR=804fh** i **_t2cmp_ISR=805ch**. Programske promenljive su u eksternoj memoriji za podatke i po~inju od adrese **a000h**, po slede}em rasporedu: **_stop (=a000h), _t2_con_value (a001h), _timer_period (a002h), _compare_value (a003h), _count_periodes (a004h), _count_compares (a005h).**

Va`ne adrese (kori{}ene u aplikaciji):

_stop = a000h – sadr`i indeks za stopiranje programa (1 za kraj programa)
_t2_con_value = a001h – sadr`i vrednost za konfiguraciju controlnog registra GP Timera 2
_timer_period = a002h – sadr`i vrednost koja }e biti upisana u GP timer 2 period_register
_compare_value = a003h – sadr`i vrednost koja }e biti upisana u GP timer 2 compare_register
_count_periodes = a004h – sadr`i test promenljivu koja se inkrementira u **_t2per_ISR**
_count_compares = a005h – sadr`i test promenljivu koja se inkrementira u **_t2cmp_ISR**

Za pra}enje ove aplikacije, korisno je pro~itati:

3. SPRU160A.PDF, str. 7-1 do 7-16. – Na~ini adresiranja (neposredno, direktno, indirektno)
4. SPRU161A.PDF, str. 2-11 do 2-41. – Tajmeri
5. SPRU160A.PDF, str. 6-9 do 6-22 i 6-35 do 6-36 – Mehanizam prekida
6. SPRU161A.PDF, str. 2-87 do 2-96 – EV Interrupti

Ne{to o **Interruptima**

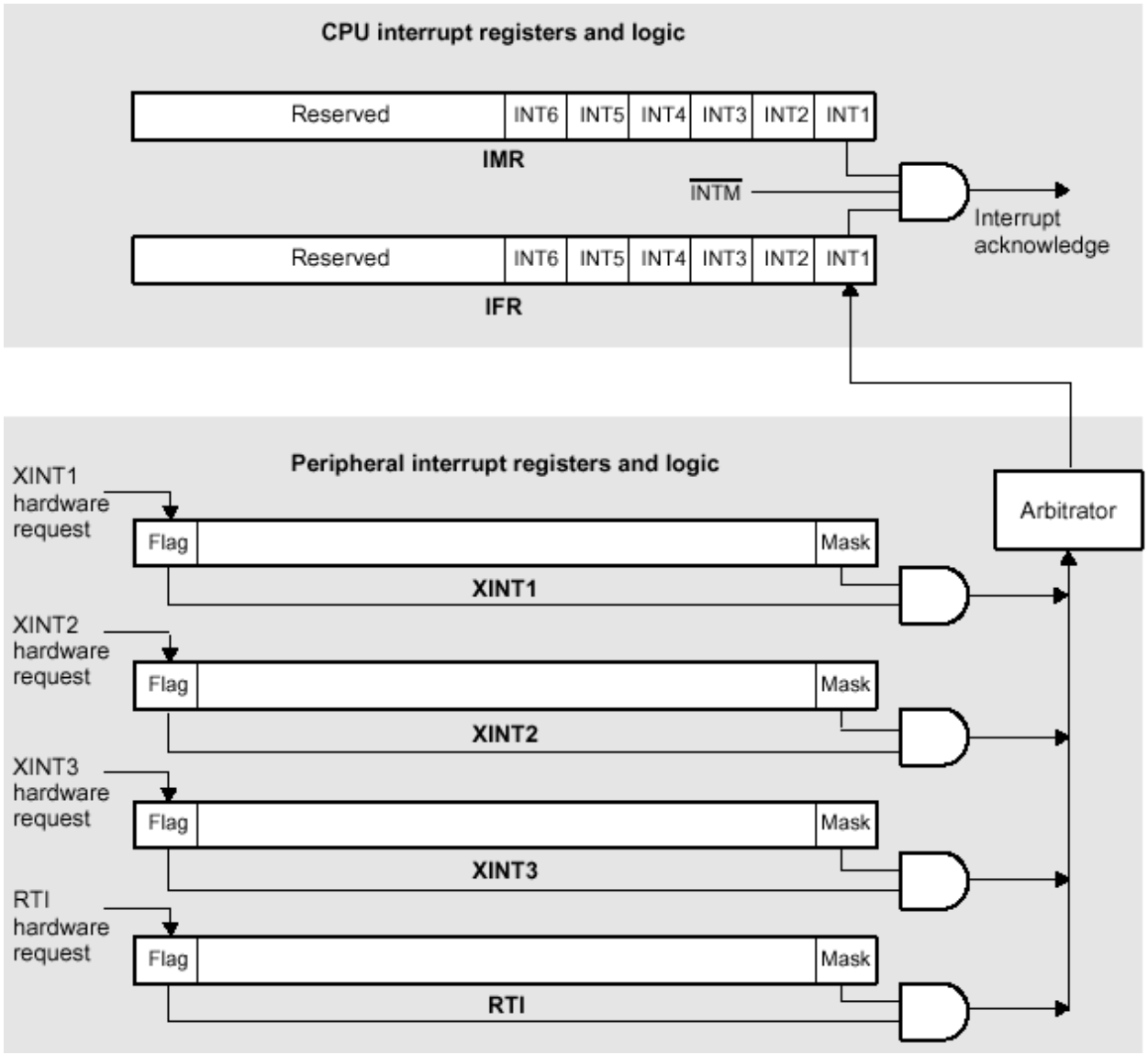
Hardverski ili softverski generisani signali koji prouzrokuju prekid izvr{enja glavnog programa da bi bio izvr{en podprogram se zovu **interrupti (prekidi),** a pomenuti podprogram – **prekidna rutina (ili Interrupt Service Routine – ISR).** Tipi~no, interrupti se generi{u hardverski (recimo to rade A/D i D/A konvertori) kada je potrebno predati podatak ili uzeti podatak iz CPUa. Interrupti su kori{}eni da signaliziraju de{avanje "doga|aja" u op{tem slu~aju (recimo, kada tajmer zavr{ava sa brojanjem). Softverski interrupti se zahtevaju putem instrukcija, a hardverski signalom sa fizi~kog ure|aja (i mogu biti interni i eksterni). Dalje, svaki interrupt mo`e biti maskiraju}i (mogu biti maskirani ili ne; ako su maskirani, to zna~i da su blokirani – tj. ne prikada se izvr{enje glavnog programa da bi se izvr{ila prekidna rutina; a u suprotnom su dozvoljeni, tj. nisu maskirani) ili nemaskiraju}i (oni ne mogu biti blokirani).

Realizacija interrupta, u op{tem slu~aju, te~e kroz tri faze:

1. Dobija se zahtev za interruptom. Prekid glavnog programa mora biti zahtevan softverski ili hardverski.
2. Prihvatanje interrupta. CPU mora prihvatiti interrupt zahtev. Ako je interrupt maskiraju}i, odre|eni uslovi moraju biti zadovoljeni da bi interrupt bio prihva}en. U slu~aju nemaskiraju}ih interrupta, interrupt se prihvata odmah.
3. Izvr{enje prekidne rutine (ISR). Kada je interrupt prihva}en, dolazi do granjanja ka **odgovaraju}em** podprogramu, koga zovemo interrupt service routine (ISR) – ili prekidna rutina. Adresa prekidne rutine (glavni program sledi instrukciju granjanja ka ovoj adresi – ta~nije programski broja~ - PC) je sme{tena na prethodno odre|enoj adresi (vektorskoj lokaciji). Instrukcija grananja se izvr{ava, i izvr{ava se napisana prekidna rutina (ISR).

U konkretnom slu~aju, na raspolaganju je 6 maskiraju}ih interrupt nivoa. Po{to TMS320C24xDSP kontroler mo`e imati vi{e od 6 interrupt izvora, svaki od 6 interrupt nivoa

mo`e biti raspodeljen na vi{e interrupt izvora. Slika dole (Sl. 13) ilustruje kori{}enu strukturu za primanje i prihvatanje maskiraju}ih interrupta. Slika pokazuje 4 interrupt izvora (XINT1, XINT2, XINT3 i RTI) koji su raspodeljeni na interrupt nivou INT1. Sli~na situacija postoji i na drugim interrupt nivoima (INT2-INT6).
Svaki od interrupt izvora ima sopstveni kontrolni registar sa flag bitom i maskiraju}im bitom. Kada je interrupt signal dobijen, flag bit u odgovaraju}em kontrolnom registru je setovan, pokazuju}i da je stigao zahtev za interruptom. Ako je maskiraju}i bit tako|e setovan, signal se {alje u arbitra`nu logi~ku jedinicu, koja mo`e da istovremeno primi sli~ne signale iz jednog ili vi{e drugih kontrolnih registara. Arbitra`na logi~ka jedinica upore|uje nivoe prioriteta konkurentskih interrupt zahteva, i dozvoljava prolaz ka CPU interruptu najvi{eg prioriteta. Posledica toga je setovanje odgovaraju}eg interrupt flaga u CPUovom IFR (Interrupt Flag Register) registru (flag odgovara nivou prioriteta INT1, INT2, ... ili INT6). Ovo pokazuje da je interrupt u o~ekivanju. Ako je odgovaraju}i maskiraju}i bit u IMR (Interrupt Mask Register) registru jedinica, i ako je INTM bit nula, CPU prihvata interrupt i izvr{ava interrupt servisnu rutinu (ISR). Kada je re~ o maskireju}im interruptima – INT1 nosi najvi{i, a INT6 najni`i nivo prioriteta. Opet, svaki od 6 maskiraju}ih interrupt nivoa (INT1-INT6) je povezan na vi{e **maskiraju}ih** interrupt izvora, koji su tako|e rangirani prema prioritetima. Izvor najvi{eg prioriteta prvi {alje interrupt zahtev odgovaraju}em interrupt nivou.



Sl. 13. *Ilustracija primanja i prihvatanja interrupta*

Postoje 2 CPU registra za kontrolu interrupta:

1. Interrupt Flag Register (IFR) je 16bitni registar na adresi 0006h u memoriji za podatke, i sadr`i flag bitove koji signaliziraju ~im maskiraju}i interrupt zahtev primi CPU na jednom od nivoa INT1 do INT6. – Kada se maskiraju}i interrupt zahteva, flag bit u odgovaraju}em kontrolnom registru se setuje na 1 (pri~amo o nivou ispod!). Ako je maskiraju}i bit u istom kontrolnom registru tako|e 1, interrupt zahtev se {alje u CPU, i pri tom setuje odgovaraju}i flag u IFR. Ovo pokazuje da se interrupt o~ekuje ili ~eka da bude prihva}en. Prihvatanjem interrupta od strane CPUa, IFR flag se bri{e (automatski).
2. Interrupt Mask Register (IMR) je 16tobitni registar na adresi 0004h u memoriji za podatke, i sadr`i maskiraju}e bitove koji dozvoljavaju ili blokiraju (maskiraju) da se desi prekid na svakom od interrupt nivoa od INT1 do INT6. Reset stanje ovog registra je da su svi njegovi bitovi nule, tj. da su maskirani svi interrupti (INT1-INT6). Kada interrupt nije maskiran, odgovaraju}i IMR bit je 1, i interrupt je prihva}en ako je i odgovaraju}i IFR bit 1 i ako je i INTM bit 0.

Ako CPU prihvati interrupt, onda izvr{ava njegovu prekidnu rutinu (ISR). Prihvatanje maskiraju}eg hardverskog interrupta podrazumeva da su odgovaraju}i bitovi IFR i IMR registra jedinice i da je INTM bit 0. Interrupt mode (INTM) bit je bit 9 status registra ST0 i kada je INTM=1, svi maskiraju}i interrupti su blokirani (a kada je INTM=0, svi nemaskirani interrupti su dozvoljeni).
Dakle, posle procesa prihvatanja interrupt zahteva, sledi proces servisiranja interrupt zahteva. – Za svaki od 6 interrupt nivoa CPU se grana ka odgovaraju}oj general interrupt service routine (GISR) – tj. izvr{ava se GISR1, ... ili GISR6. U GISRu se obavlja identifikacija interrupt zahteva i grananje ka specific interrupt service routine (SISR). GISR mora da pro~ita vrednost (adresu) koja se ~uva u *interrupt vector registru* da bi mogao da ostvari grananje prema pravoj SISR. Po izvr{enju SISR (koja je napisana prema potrebama aplikacije) sledi povratak na mesto prekinute programske sekvence (u glavnom programu).
– detaljnije o ovome u Spru1601.pdf, na strani 6-22, 6-35 i 6-36.

EV Interrupt izvori (videti Spru161a.pdf, od strane 2-87)

Organizovani su u 3 grupe: A, B i C. Grupa A generi{e zahtev za interruptom ka nivou INT2, a B i C ka INT3 i INT4, respektivno. Postoji interrupt flag registar i interrupt mask registar za svaku od EV grupa: EVIFRA, EVIFRB i EVIFRC; i EVIMRA, EVIMRB i EVIMRC.
Kada se desi "doga|aj" u EV modulu (recimo, broja~ tajmera se izjedna~io sa periodom tajmera), odgovaraju}i interrupt flag u odgovaraju}em EVIFRx (x=A, B ili C) (prim. videti tabelu na strani 2-88) se setuje na vrednost 1. Ukoliko je i odgovaraju}i bit u odgovaraju}em EVIMRx tako|e 1, upu}uje se interrupt zahtev ka odgovaraju}em INTxx (xx=2, 3, ili 4), ali ako nema i drugih kandidata koji bi poslali interrupt zahtev. U suprotnom, za svaku grupu – definisan je prioritet interrupt izvora, {to je definisano vredno{}u pripadaju}eg interrupt vektora (ID).
Naime, interrupt vektor (ID) koji odgovara setovanom intarrupt flagu najve}eg prioriteta me|u setovanim flagovima (u EVIFRx) - se upisuje u akumulator kada je (prethodni) interrupt vektor (EV interrupt grupe) pro~itan posle generisanog interrupt zahteva ka CPU (ka INT2, 3, ili 4). Flag se bri{e kada je njegov interrupt vektor pro~itan. Interrupt flag mo`e biti izbrisan i softverski – direktnim upisivanjem 1 u bit.
Po{to je EV interrupt zahtev usvojen, sadr`aj registra EVIVRx (u njemu je vrednost vektora (ID (videti tabelu 2-12 u Spru161a.pdf)) koji se odnosi na interrupt flag najvi{eg prioriteta me|u setovanim interrupt flagovima) mora biti pro~itan u akumulatoru i pomeren u levo za jedan ili vi{e bitova. Zatim se offset adresa (adresa odstupanja – prema po~etku interrupt ulazne tabele) dodaje u akumulator. Instrukcija BACC se koristi za grananje programa pravo unutar tabele, a drugo grananje iz tabele grana program ka prekidnoj rutini (ISR). (BACC instrukcija sadr`aj ACC(15:0) preme{ta u programski broja~ PC – i nastavlja se izvr{enje programa sa startne adrese u programskoj memoriji koja je sada sadr`aj programskog broja~a). (Primedba vezana za poslednji pasus: u programima koji slede ovo ne}e biti metod za startovanje ISR (bar ne eksplicitno). Kod MCK240, posebnu ulogu ima fajl Vector_a.h – i tu treba obratiti pa`nju.)

Za vi{e detalja – pro~itati sve o interruptima u Spru160a.pdf.

Analiza ponu|enog programskog re{enja aplikacije

Ponu|eni asemblerski program *gpt_2_a.asm* predstavlja nadgradnju programa *gpt_1_a.asm* i zasnovan je na slede}em algoritmu:

1. Setuju se bitovi slede}ih registara:
    a. 0 se upisuje na adresu _stop=a000h. Kada najmla|i bit na ovoj adresi postane 1 (_stop[0]=1) tada program prestaje sa radom.
    b. 1 se upisuje u OPCRA[12] => funkcija pina 106 DSP-ija se vezuje za GP Timer 2 (T2PWM/T2CMP). Videti na str. 11-14 u Spru161a.pdf kako se konfiguri{e Output Control Register A (OPCRA=7090h). (da je OPCRA[12]=0, isti pin bi imao I/O funkciju).
    c. 00 se upisuje u GPTCON[10-9] => GP Timer 2 ne}e startovati AD konverziju; GPTCON=7400h - General Purpose Timer Control Register (v. str. 2-38 u Spru161a.pdf).
    d. 1 se upisuje GPTCON[6] => dozvoljena su sva 3 GP Timer Compare Outputa. Zapravo, sada GPT2 mo`e da koristi *compare output*, odnosno sada mo`e da se pin T2PWM/T2CMP stavi u funkciju.
    e. 1 se upisuje GPTCON[2] => videti Spru161a.pdf str. 2-39) => GPT2 Compare Output (tj. signal na pinu T2PWM/T2CMP) je setovan kao Active low (po prvom poklapanju vrednosti u T2CMP i T2PER signal na pinu 106 pada na low nivo...).
    f. (_timer_period=a002h)->(T2PER=7407h). Setuje se period GP Timera 2 tako {to se na adresu T2PER=7407h upisuje sadr`aj koji je korisnik izabrao u komandnom prozoru aplikacije (privremeno sme{ten na adresu _timer_period=a002h.
    g. 0 se upisuje na adresu T2CNT=7405h – broja~ GP Timera 2 se postavlja na 0.
    h. (_compare_value=a003h)->(T2CMP=7406h). Setuje se sadr`aj Compare registra GPT2. Ovu vrednost bira korisnik u komandnom prozoru aplikacije (kao procenat periode tajmera) i tada se ona upisuje na adresu _compare_value=a003h. Sa ovom vredno{}u (ukoliko je T2CON[1]=1), neprekidno se upore|uje broja~ tajmera, i kada se desi poklapanje => menja se stanje signala na T2PWM/T2CMP pinu.
    i. (_t2_con_value=a001h)->(T2CON=7408h) - konfiguri{e se GP Timer 2. Pogledati Spru161a.pdf str. 2-37.
    j. _t2per_ISR=0804fh -> (tpint2vec=73h); _t2per_ISR adresa se upisuje u odgovaraju}i interrupt vektor.
    k. _t2cmp_ISR=0805ch -> (tcint2vec=74h); _t2cmp_ISR adresa se upisuje u odgovaraju}i interrupt vektor.
    l. 0 se upisuje na adresu _count_periodes=a004h (inicijalizacija test promenljive)
    m. 0 se upisuje na adresu _count_compares=a005h (inicijalizacija test promenljive)
    n. Setuje se bit Interrupt Mask Registera: 1 -> IMR[2] => INT3 vi{e nije maskiran.
    o. 1 -> IMRB[0]; (IMRB – Group B Interrupt Mask Register) => omogu}eno je aktiviranje T2PINT interrupta (tj. GP Timer 2 period interrupt) – tj. vi{e nije maskiran.
    p. 1 -> IMRB[1]; (IMRB – Group B Interrupt Mask Register) => omogu}eno je aktiviranje T2CINT interrupta (tj. GP Timer 2 period interrupt) – tj. vi{e nije maskiran.
    q. 1 se upisuje u bit T2CON[6] => start GP Timera 2
2. petlja:
    a. Program Monitor se aktivira (u svakom ciklusu).
    b. Proverava se u svakom ciklusu da li je u najni`i bit adrese _stop=a000h upisana jedinica ({to se de{ava pritiskom na Stop u komandnom prozoru aplikacije). Ako jeste to je uslov za izlazak iz petlje. Ako ne, sledi povratak na po~etak petlje.
3. KRAJ

Prekidna rutina I – definisana adresom _t2per_ISR=804fh (u glavnom programu upisana u interrupt vektor tpint2vec=73h)

1. (_count_periodes=a004h)+1->(_count_periodes=a004h)
2. Kraj prekidne rutine i povratak u glavni program

Prekidna rutina II – definisana adresom _t2cmp_ISR=805ch (u glavnom programu upisana u interrupt vektor tcint2vec=74h)

1. (_count_compares=a005h)+1->(_count_compares=a005h)
2. Kraj prekidne rutine i povratak u glavni program

Detaljniji uvid se mo`e ste}i na osnovu ponu|enog asemblerskog programa *gpt_2_a.asm*, koji sledi u nastavku (listing programa je u boji, a obja{njenja i primedbe nisu).

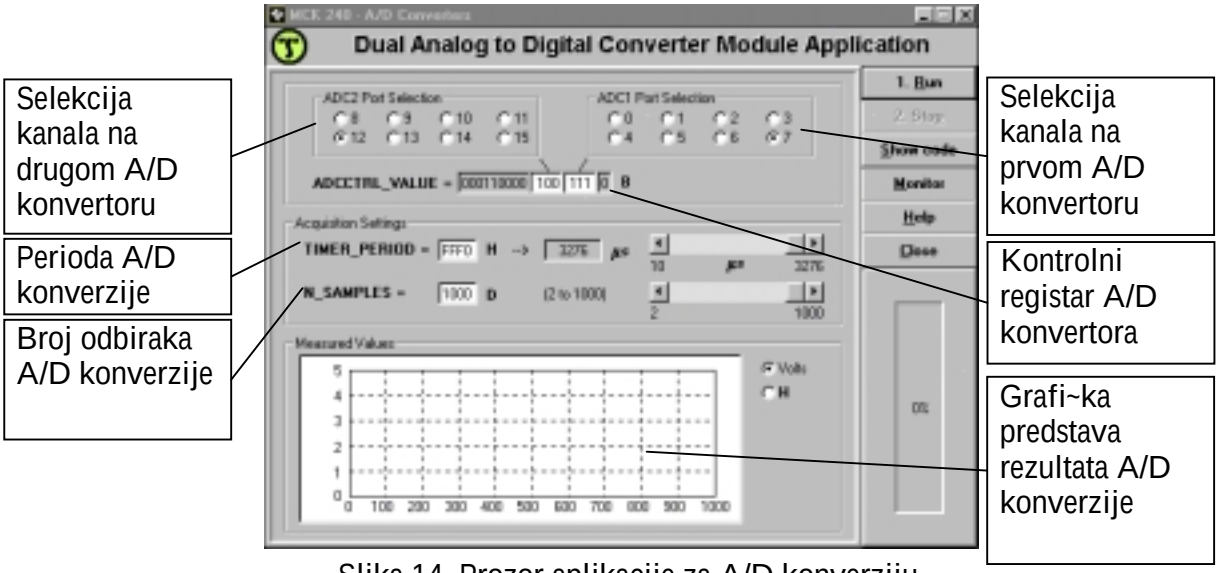
```
;*******************************************************************************
; File Name: gpt_2_a.asm
; Project:     MCK240
; Originator:  R.Giuclea
; Description:          ASM file for GPT demo 2
; Copyright © 1997 Technosoft
;*******************************************************************************
; Include Files
;-------------------------------------------------------------------------------
                .include    ..\F240_a.h
                .include    ..\demos_a.h
                .include    ..\vects_a.h
                .include    gpt_2_a.h
;=============================================
                .text
;-------------------------------------------------------------------------------
_start:                                     ... labela
        LDP     #_stop                      ... _stop = a000h; => DP=320
        SPLK    #0,_stop                    ... setuje se 0 kao sadr`aj adrese _stop= a000h
; configure timer shared pin
        LDP     #DP_PF2                     ; Peripheral File 2 Data Page Pointer
... DP_PF2 je = 0E1h (videti F240_a.h); => nova vrednost data page pointera (DP=0E1h=225dec).
        SETBIT OPCRA,SETB12                 ... SETBIT je makro koji je definisan u Demos_a.h;
=> OPCRA[12]=1 => T2PWM/T2CMP funkcija pina 106 DSP-ija (videti str. 11-4 u Spru161a.pdf).
        SACL    OPCRA           ; Shared T2PWM/T2CMP or IOPB4 pin configured for GPT2
... Sadr`aj akumulatora se upisuje na adresu OPCRA. ME\UTIM, ovo je gre{ka i ovu instrukciju bi trebalo izostaviti
(sadr`aj ACC nije prethodno definisan, a {ta smo hteli, postigli smo makroom SETBIT).
; timer will not start ADC automatically
        LDP     #DP_EV                      ; Event Manager Data Page Pointer   ...DP=0E8h=232dec
        LACC    GPTCON                      ... Upisuje se sadr`aj adrese 07400h (GPTCON- General purpose timer
control register) u akumulator (ACC); (adresa GPTCON je definisana u F240_a.h)
        AND     #AND_T2TOADC_       ; AND mask for DISABLING ADC start on GPT2
... u Demos_a.h definisana je konstanta AND_T2TOADC_      = 0F9FFh
... operacija logi~ko I se sprovodi nad bitovima sadr`aja adrese 07400h (GPTCON) i 0F9FFh – resetuje se bit 9 i 10
akumulatora.
        SACL    GPTCON                      ; configure GPTCON not to start ADC on GPT2 Event
... Sadr`aj akumulatora se upisuje na adresu 07400h (GPTCON) => GPTCON[10-9]=00 => GP Timer 2 ne}e
startovati AD konverziju (videti Spru161a.pdf str. 2-38)
; enable GPT compare outputs and specify active state of GPT2 compare output
        SETBIT GPTCON,SETB6; active low... SETBIT je makro koji je definisan u Demos_a.h
... => GPTCON[6]=1 (videti Spru161a.pdf str. 2-39) => dozvoljena su sva 3 GP Timer Compare Outputa
        SETBIT GPTCON,SETB2; active low
... => GPTCON[2]=1 (videti Spru161a.pdf str. 2-39) => GPT2 Compare Output je low
; load and init timer
        LDP     #_timer_period              ... _timer_period=a002h (videti gpt_2.map), setuje se DP=320
        LACC    _timer_period               ... upisuje se sadr`aj _timer_period=a002h u akumulator; Sadr`aj
_timer_period=a002h setuje korisnik u komandnom prozoru aplikacije, i to je perioda tajmera.
        LDP     #DP_EV                      ... DP=0E8h=232dec.
        SACL    T2PER           ; load GPT2 timer period
... sadr`aj akumulatora se upisuje na adresu T2PER=7407h (videti F240_a.h), i tako defini{e period tajmera GPT2.
Prakti~no: (_timer_period=a002h)->( T2PER=7407h).
        LACC    #0                          ... konstanta 0h se upisuje u akumulator
```

```
        SACL    T2CNT           ; reset GPT2 counter register
```
... sadr`aj akumulatora se upisuje na adresu T2CNT=7405h (videti F240_a.h) – i radi se o adresi T2 Counter Registera (sadr`aj je broja~ GP Timera 2). Prakti~no: 0->(T2CNT=7405h)
```
        LDP     #_compare_value         ... _compare_value=a003h (videti gpt_2.map), setuje se DP=320
        LACC    _compare_value          ... upisuje se sadr`aj _compare_value=a003h u akumulator; Sadr`aj
```
_compare_value=a003h setuje korisnik u komandnom prozoru aplikacije, i to je *compare* vrednost tajmera.
```
        LDP     #DP_EV                  ... DP=0E8h=232dec
        SACL    T2CMP           ; load GPT2 timer compare register
```
... sadr`aj akumulatora se upisuje na adresu T2CMT=7406h (videti F240_a.h) – i radi se o adresi T2 Compare Registera. Prakti~no: (_compare_value=a003h)->(T2CMT=7406h)
```
        LDP     #_t2_con_value          ... _t2_con_value=a001h (videti gpt_1.map), setuje se DP=320
        LACC    _t2_con_value           ... upisuje se sadr`aj _t2_con_value=a001h u akumulator; Sadr`aj
```
_t2_con_value=a001h setuje korisnik u komandnom prozoru aplikacije – biraju}i mod brojanja tajmera i frekvenciju rada tajmera.
```
        LDP     #DP_EV                  ... DP=0E8h=232dec
        SACL    T2CON           ; load GPT2 configuration register
```
... sadr`aj akumulatora se upisuje na adresu T2CON=7408h (videti F240_a.h) – i radi se o adresi T2 Control Registera (videti Spru161a.pdf str. 2-36). Prakti~no: (_t2_con_value=a001h)->(T2CON=7408h)
```
; load ISR addresses to Interrupt Vector in on-chip block B2
```
... sledi deo programa radi koga je **po`eljno pro~itati** Interrupte (str. 6-9 u SPRU160A.PDF) i EV Interrupte (str. 2-87 u SPRU161A.PDF). I **obratiti pa`nju** na sadr`aj Vects_a.h.
```
        LACC    #_t2per_ISR             ... _t2per_ISR = 0804fh (videti gpt_2.map); sadr`aj ove konstante
```
postaje sadr`aj akumulatora.
```
        LDP     #0                      ... Setuje se nova vrednost data page pointera DP=0
        SACL    tpint2vec       ; load _t2per_ISR address into corresponding
                                ; interrupt vector
```
... tpint2vec = 060h (B2_SADDR)+13h =73h (=115) - TPINT2 vector address – def. u F240_a.h i Vects_a.h;
... na ovu adresu se sme{ta sadr`aj akumulatora
```
        LACC    #_t2cmp_ISR             ... _t2cmp_ISR = 0805ch (videti gpt_2.map); ova konstanta postaje
```
sadr`aj akumulatora.
```
        SACL    tcint2vec       ; load _t2per_ISR address into corresponding
                                ; interrupt vector
```
... tcint2vec = 060h (B2_SADDR)+14h =74h (=116) - TCINT2 vector address – def. u F240_a.h i Vects_a.h;
... na ovu adresu se sme{ta sadr`aj akumulatora; tj. _t2cmp_ISR = 0805ch->(tcint2vec = 074h)
```
; init count variables
        LDP     #_count_periodes        ... _count_periodes=a004h (v. gpt_2.map) ; DP=320.
        LACC    #0                      ... 0h postaje sadr`aj akumulatora (ACC)
        SACL    _count_periodes         ... sadr`aj ACC se upisuje na adresu _count_periodes=a004h
        SACL    _count_compares         ... sadr`aj ACC se upisuje na adresu _count_compares=a005h
; unmask interrupts
        LDP     #0                      ... DP=0
        SETBIT IMR,SETB2        ; unmask INT3  !!!! EV IRQAs => EVIVRB==INT3 Internal
                ; INT3 flag is bit 2 of IMR mask register
```
... SETBIT je makro koji je definisan u Demos_a.h
... IMR   =   0004h   ; Interrupt Mask Register   (videti F240_a.h)
... SETB2   =   0004h   ; Bit Mask for 2   (videti F240_a.h)
... Su{tina je da se nad bitovima sadr`aja adrese IMR sprovede operacija logi~ko ILI sa vredno{}u konstante SETB2 i taj rezultat se sme{ta opet na adresu IMR => IMR[2]=1 => INT3 nije vi{e maskiran.
```
        LDP     #DP_EV                  ... Setuje se nova vrednost DP (DP=0E8h=232dec).
        SETBIT IMRB,SETB0       ; enable T2PINT (activate GPT2 period interrupt generation)
```
... SETBIT je makro koji je definisan u Demos_a.h
... IMRB   =   742dh   ; Group B Interrupt Mask Register(videti F240_a.h)
... SETB0   =   0001h   ; Bit Mask for 0   (videti F240_a.h)
... Su{tina je da se nad bitovima sadr`aja adrese IMRB sprovede operacija logi~ko ILI sa vredno{}u konstante SETB0 i taj rezultat se sme{ta opet na adresu IMRB => IMRB[0]=1 => omogu}eno je aktiviranje T2PINT interrupta (tj. GP Timer 2 period interrupt) – tj. vi{e nije maskiran.
```
        SETBIT IMRB,SETB1       ; enable T2CINT (activate GPT2 compare interrupt generation)
                ; EVIMRB == IMRB
```
... => IMRB[1]=1 => omogu}eno je aktiviranje T2CINT interrupta (tj. GP Timer 2 compare interrupt) – tj. vi{e nije maskiran.
```
; start timer
        SETBIT T2CON,SETB6   ;start GPT2
```
... SETBIT je makro koji je definisan u Demos_a.h
... => T2CON[6]=1 (videti Spru161a.pdf str. 2-36) => startuje se GP Timer 2
```
;
```

```
loop:                                   ...labela (defini{e po~etak petlje)
; call monitor
        CALL    MON240                  ... Programski broja~ (PC-program counter) se inkrementira i stavlja na vrh steka. Sadr`aj adrese u programskoj memoriji MON240=0109h (videti gpt_2_a.h). postaje sadr`aj programskog broja~a. Poziva se program monitor.
; test if demo ends (_stop =1)
        LDP     #_stop                  ... setuje se DP=320
        BIT     _stop,15                ... posmatra se sadr`aj na adresi _stop. Specificirani bit code=15, odnosi se na najni`i bit (LSB) na posmatranoj adresi. Instrukcija BIT kopira ovaj bit u TC bit status registra ST1. TC (test/control flag bit) – je bit 11 status registra ST1 - ~uva rezultat operacije testiranja.
        BCND    loop,NTC                ... Ako je uslov NTC ispunjen (tj. ako je TC=0) program ide na loop (labela – gore); Odnosno, ako je klikom na Stop u prozoru aplikacije setovano: _stop[0]=1 => izlazak iz petlje.
;
        END_DEMO                        ... END_DEMO je makro, definisan u Demos_a.h; Kada bude setovano _stop[0]=1, tj. da je TC=1, program prestaje da se izvr{ava.
;-------------------------------------------------------------------------
; timer 2 period interrupt service routine:     ... Sledi listing prekidne rutine I
;
_t2per_ISR:                             ... _t2per_ISR = 0804fh (videti adc.map) – adresa prekidne rutine, koja je bila upisana u odgovaraju}i interrupt vektor (tpint2vec=73h)
        LDP     #_count_periodes        ... _count_periodes=a004h (v. gpt_2.map) ; DP=320
        LACC    #1                      ...  u akumulator se upisuje konstanta 1
        ADD     _count_periodes         ... sadr`aj akumulatora se sabira sa sadr`ajem adrese _count_periodes=a004h i rezultat opet sme{ta u akumulator.
        SACL    _count_periodes         ... sadr`aj akumulatora se upisuje na adresu _count_periodes=a004h
;
        END_ISR                         ... makro definisan u Demos_a.h; Kraj prekidne rutine i povratak u glavni program.
;-------------------------------------------------------------------------
; timer 2 period (!?compare?) interrupt service routine:    ... Sledi listing prekidne rutine II
;
_t2cmp_ISR:                             ... _t2cmp_ISR = 0805ch (videti adc.map) – adresa prekidne rutine, koja je bila upisana u odgovaraju}i interrupt vektor (tcint2vec=74h)
        LDP     #_count_compares        ... _count_compares=a005h (v. gpt_2.map) ; DP=320
        LACC    #1                      ...  u akumulator se upisuje konstanta 1
        ADD     _count_compares         ... sadr`aj akumulatora se sabira sa sadr`ajem adrese _count_compares=a005h i rezultat opet sme{ta u akumulator.
        SACL    _count_compares         ... sadr`aj akumulatora se upisuje na adresu _count_compares=a005h
;
        END_ISR                         ... makro definisan u Demos_a.h; Kraj prekidne rutine i povratak u glavni program.
;-------------------------------------------------------------------------
```

3.3.3. A/D konverzija

Ova aplikacija omogu}uje A/D konverziju sa dva kanala po izboru korisnika sa isprogramiranom periodom konverzije i brojem odbiraka i startuje se klikom na ikonu **A/D *Converters*** u ***Processor Evaluation Control Panel*** prozoru. Jedan kanal je iz ni`ih 8, a drugi iz vi{ih 8 kanala i naponi koji se na njima mogu meriti su u opsegu 0 do 5 V. Sa svakog kanala se mo`e dobiti do 1000 odbiraka. Rezultati A/D konverzije se sme{taju u *buffer* maksimalne du`ine 2×1000 re~i. Prakti~no, mo`emo pratiti samo promenu napona na kanalu 7, koja se posti`e usled zakretanja trimera. Generisanje (ili dovo|enje sa nekog generatora) drugih analognih signala na MCK240 trenutno nije izvedeno, ali mogu}e je. Na slede}oj slici je dat grafi~ki prikaz ove aplikacije:



Slika 14. Prozor aplikacije za A/D konverziju

Asemblerski program koji sadr`i kod ove aplikacije je ***adc_a.asm*** i mo`e se videti pritiskom na taster ***Show code***. Zaglavlje asemblerskog programa je ***adc_a.h*** koje sadr`i promenljive i funkcije za ovu aplikaciju. Startna adresa programa je **8000h** u eksternoj programskoj memoriji i zove se **_start**, a prekidna rutina po~inje sa **_t2per_ISR**. Programske promenljive su u eksternoj memoriji za podatke i po~inju od adrese **a000h**, po slede}em rasporedu: **_timer_period** (=a002h), **_m_samples** (=a003h), **_count_adc** (=a004h), **_count_saved** (=a005h), itd.

Za dobro razumevanje ove aplikacije preporu~uje se prethodno ~itanje slede}ih sadr`aja:

1. SPRU160A.PDF, str. 7-1 do 7-16. – Na~ini adresiranja (neposredno, direktno, indirektno)
2. SPRU161A.PDF, str. 2-11 do 2-41. – Tajmeri
3. SPRU161A.PDF, str. 3-1 do 3-10. – AD konverzija
4. SPRU160A.PDF, str. 6-9 do 6-22 i 6-35 do 6-36 – Mehanizam prekida
5. SPRU161A.PDF, str. 2-87 do 2-96 – EV Interrupti

Va`ne adrese (kori{}ene u aplikaciji):
_timer_period = **a002h** - sadr`i periodu A/D konverzije
_n_samples = **a003h** - sadr`i zadati broj odbiraka A/D konverzije
_count_adc = **a004h** - sadr`i aktuelni broj odbiraka A/D konverzije
_count_saved = **a005h** - sadr`i broj odbiraka sme{tenih u *buffer*
_res_buf = **c000h** – (po~etna) adresa *buffera* od 2000 re~i - za sme{tanje rezultata AD konverzije sa oba izabrana kanala

Programsko re{enje ove aplikacije je zasnovano na slede}em algoritmu:

1. Na adresu _stop=a000h upisuje se 0. Svrha je inicijalizacija programske promenljive – jer kada u najni`em bitu na adresi _stop bude upsana 1, to }e zna~iti kraj izvr{enja programa.

2. Na adesu GPTCON=7400h na mesto bitova 9-10 upisuje se 00. Naime, na raspolaganju su 3 tajmera (GP Timer 1, 2, i 3) i svaki od njih mo`e da startuje AD konverziju. Da li }e i na koji na~in to biti u~injeno – sadr`ano je u registru GPTCON. Upisivanjem 00 na mesto bitova 9-10 u GPTCON, spre~ava se startovanje AD konverzije od strane GP Timera 2 (GP Timer 2 }e u ovoj aplikaciji kontrolisati periodu odabiranja AD konverzije). Bitovi GP Timer Control Registera (GPTCON) specificiraju smer brojanja GP Timera (1, 2, ili 3), i "doga|aj" kojim GP Timer (1, 2, 3) uti~e na startovanje AD konverzije. Re~ je o tome da se obezbedi sinhronizacija izme|u "doga|aja" (napr. poklapanje broja~a tajmera sa setovanom periodom odabiranja) GP Timera[*2] i starta AD konverzije bez intervencije CPUa. Svaki od GP Timera (1, 2, ili 3) je neposredno kontrolisan bitovima registra TxCON (x je 1, 2, ili 3), i svakom Timeru su pridru`eni i registri TxCNT (sadr`aj registra je broja~ tajmera), TxCMPR i TxPER (sadr`aj TxCMPR i TxPER se setuje – shodno izabranim konstantama). Period GP Timera x se setuje upisivanjem odgovaraju}e vrednosti u TxPER. Vrednost koja se ~uva u compare registru (TxCMPR) se konstantno upore|uje sa broja~em GP Timera x – kada se desi poklapanje, de{ava se promena compare outputa, saglasno odgovaraju}em bitu u GPTCON. Ovo je jedna od mogu}nosti kada se odgovaraju}i interrupt flag setuje, i generi{e zahtev za interruptom.
   Mogu}i "doga|aji":
   - compare event (match) – de{ava se kada sadr`aj GP Timer x Counter Registera (TxCNT) je isti kao sadr`aj GP Timer x Compare Registera (TxCMP).
   - overflow event – de{ava se kada sadr`aj GP Timer x Counter Registera (TxCNT) dostigne FFFFh. (broja~ do|e do FFFFh)
   - underflow event – de{ava se kada sadr`aj GP Timer x Counter Registera (TxCNT) dostigne 0000h. (broja~ do|e do 0000h)
   - period event (match) – de{ava se kada sadr`aj GP Timer x Counter Registera (TxCNT) je isti kao sadr`aj GP Timer x Period Registera (TxPER).
3. Na adresu T2PER=7407h upisuje se perioda odabiranja. Perioda odabiranja se bira jednostavno, posredstvom korisni~kog interfejsa aplikacije, i tada se sme{ta na adresu _timer_period=a002h; i s tom vredno{}u se setuje period GP Timera 2 (sadr`aj adrese T2PER=7407h). Sadr`aj T2PER odre|uje period GP Timera 2. GP Timer 2 se u op{tem slu~aju zaustavlja i zadr`ava na vrednosti setovanog perioda, resetuje se na 0 ili startuje brojanje unazad, kada se dogodi podudaranje izme|u perioda (sadr`aj T2PER) i broja~a tajmera (sadr`aja T2CNT=7405h), i zavisno od moda brojanja u kome jeste tajmer.
4. Upisuje se 0 na adresu T2CNT=7405h (Timer 2 Counter Register) – tj. GP Timer 2 broja~ je setovan na nulu. Period GP Timera treba da uvek bude inicijalizovan pre njegovog broja~a.
5. Konstanta T2CNFREG=1002h se upisuje na adresu T2CON=7408h (GP Timer 2 Control Register – odre|uje, izme|u ostalog, mod brojanja GP Timera 2). Konfiguri{e se GP Timer 2. Setovan je Continuos-Up Count Mode – {to zna~i: Tajmer }e brojati (sa CPU clockom od 0.05 μs – po{to je grupa bitova 10-8 u T2CON 000) od vrednosti u T2CNT do vrednosti u T2PER, a onda se resetuje na 0 i ciklus po~inje opet (ali od 0 do sadr`aja T2PER) i sve tako. Kako je bit 1 u T2CON setovan da je 1, omogu}eno je upore|ivanje sadr`aja T2CNT=7405h sa GP Timer 2 Compare Registerom (T2CMPR). Ako su sadr`aji isti setuje se compare interrupt flag (sadr`aj T2CMPR je 0 – proveriti kori{}enjem programa Monitor). Bit 6 u T2CON je 0, {to zna~i da GP Timer 2 nije aktiviran.
6. Konstanta ADC_CONF_2 = 0403h se upisuje u registar ADCTRL2, tj. na adresu ADCTRL2 = 07034h. Na taj na~in se konfiguri{e ADC Control Register 2. ADCTRL2 je jedan od dva registra za kontrolu AD konverzije, odnosno jedan od 4 registra koji se direktno odnose na AD konverziju. Ovde se setuje 1 u bit 10, ~ime se omogu}ava da start AD konverzije bude iniciran "event managerom" – zavisno od "comapare match (event)". Setovanjem 011 u bitove 2-0 – obezbe|uje se *Pescale Value =* 10 – odnosno, garantujemo vreme AD konverzije koje }e biti ve}e od 6μs i na taj na~in ta~nost AD konverzije. Ostali

bitovi u registru su nule – neki od njih pokazuju stanje ADC1 FIFO i ADC2 FIFO registara (prazni su sada!), u koje se sme{taju neposredni rezultati AD konverzije.

7. Sadr`aj adrese _adcctrl_value = a001h (tu je sadr`ana i informacija o izabranim kanalima AD konverzije – videti korisni~ki interfejs aplikacije) se upisuje u registar ADCTRL1, tj. na adresu ADCTRL1 = 07032h. ADC Control Register 1 (ADCTRL1) kontroli{e start konverzije, omogu}ava/onemogu}ava funkciju ADC modula, omogu}ava interrupt, i kraj konverzije. Bit 0 je 0, i bitovi 15-7 su 000110000, dok bitovi 6-4 odre|uju jedan od 8 kanala ADC2, a bitovi 3-1 jedan od 8 kanala ADC1. Bitovi 15-14 se odnose na proces emulacije i oni su 00. Ako je bit 13 (ADCIMSTART) 1 odmah se startuju konverzije (na oba izabrana kanala), dok u suprotom ({to je ovde slu~aj) nema akcije. Bitovi 12-11 omogu}avaju (ako su 11, {to je ovde slu~aj), ili onemogu}avaju (ako su 00) ADC1 i ADC2, respektivno. Bit 10 – AD konverzija se kontinualno izvr{ava (=1) ili nema akcije (=0) – {to je ovde slu~aj. Bit 9 omogu}ava iterrupte – Ako je setovan, interrupt se zahteva kada je bit 8 setovan. Bit 8 je interrupt flag bit – i indicira da li se interrupt doga|aj desio (0 - nije se desio). Bit 7 pokazuje status AD konverzije –> 0 – kraj konverzije, 1 – konverzija je u toku. Bit 0 se odnosi na start konverzije (1 – konverzija startuje, suprotno – nema akcije; ali da bi konverzija mogla startovati bilo kanal 1 bilo kanal 2 mora biti dostupan – tj. jedan od bitova 12-11 mora biti 1).
8. _t2per_ISR=0803bh se upisuje na adresu tpint2vec =73h. Zapravo, upisuje se Interrupt Servise Routine (_t2per_ISR) u Interrupt Vector (tpint2vec). Kada budu ispunjeni uslovi za izvr{nje prekidne rutine, program ide na startnu adresu prekidne rutine 0803bh (ka adresi koja je upisana u interrupt vektor).
9. Inicijalizacija broja~kih promenljivih. 0 se upisuje na adresu _count_adc=a004h (na ovu adresu se kasnije sme{ta broj realizovanih odbiraka AD konverzije – izvr{ena AD konverzija sa oba kanala inkrementira vrednost ovog broja~a za 1), i 0 se upisuje na adresu _count_saved=a005h (na ovu adresu se sme{ta broj snimljenih odbiraka u buffer sa oba kanala).
10. Unmask INT3. IMR (Interrupt Mask Register) je 16-bitni registar na adresi 0004h i sadr`i maskiraju}e bite za sve nivoe maskiraju}ih interrupta (INT1-INT6). Nad bitovima ovog registra se sprovodi operacija logi~ko ILI sa konstantom 0004h i rezultat opet sme{ta u IMR=0004h. Bit 2 IMR registra postaje 1, {to zna~i da interrupt nivoa INT3 nije vi{e maskiran. NAIME, svaki interrupt izvor ima sopstveni kontrolni registar sa flag bitom i maskiraju}im bitom. Kada se interrupt signal dobije, pomenuti flag bit (u odgovaraju}em kontrolnom registru) je setovan (=1) (~ime je indiciran zahtev za interruptom). Ako je maskiraju}i bit tako|e setuje (=1), signal se {alje u arbitra`nu logi~ku jedinicu, koja mo`e istovremeno da primi sli~ne signale od jednog ili vi{e drugih kontrolnih registara. Arbitra`na logi~ka jedinica upore|uje nivoe prioriteta konkurentnih interapt zahteva, i prolazi interrupt najvi{eg nivoa (INT1) prioriteta u CPU. Tada se interrupt flag u CPUovom Interrupt Flag Registeru (IFR) (koji odgovara nivou prioriteta interrupta na ~iji zahtev se prihvata) setuje (=1). To sada pokazuje da se interrupt o~ekuje, odnosno da je u toku ili se ~eka odobrenje za izvr{enje. Ako je odgovaraju}i IMR bit 1 (to smo sada setovali za INT3) i INTM bit je 0, CPU prihvata interrupt i izvr{ava Interrupt servisnu rutinu (ISR). INTM bit je bit 9 status registra STO. Kada je INTM=0, svi nemaskirani interrupti su dozvoljeni, a kada je INTM=1, svi maskiraju}i interrupti su onemogu}eni. Podvucimo da postoje 2 registra u CPU za kontrolu interrupta: 1) IFR na adresi 0006h – sadr`i falg bitove koji pokazuju kada su maskiraju}i interrupt zahtevi ispunjeni u CPU od nivoa INT1 do IsNT6. 2) IMR na adresi 0004h – sadr`i maskiraju}e bitove koji omogu}avaju svaki od interrupt nivoa (INT1-INT6). Recimo kada je bit 2 u IMR 1, bit 2 u IFR 1, i bit INTM=0, dolazi do relalizacije INT3, odnosno, poziva se ISR (Interrupt Service Routine).
11. IMRB (EV Interrupt Mask Register B) je 16-bitni registar na adresi 742dh i sadr`i maskiraju}e bite za sve interrupte iz grupe B (koji su upu}eni ka INT3) koje generi{e Event Manager – videti Tab. 2-10 u SPRU161A.PDF. Nad bitovima ovog registra se sprovodi operacija logi~ko ILI sa konstantom 0001h i rezultat opet sme{ta u

IMRB=742dh. Bit 0 IMRB registra postaje 1, {to zna~i da je omogu}eno aktiviranje T2PINT interrupta (tj. GP Timer 2 period interrupt) – tj. vi{e nije maskiran. – Adresa vektora interrupta T2PINT je definisana u Vects_a.h (programska promenljiva tpint2vec), i sadr`aj adrese tpint2vec je startna adresa za interrupt T2PINT. (Preporu~uje se ~itanje poglavlja 2.10 u SPRU161A.PDF).

12. Startovanje GP Timer 2 (pro~itati pod 5). GP Timer Control Register - T2CON je 16-bitni registar na adresi 7408h i sadr`i bite: 15-14 – odnose se na kontrolu emulacije; 13-11 – odnose se na na~in (mod) brojanja; 10-8 – odre|uju clock timera u odnosu na CPU clock; 7 – odre|uje da li se tajmer startuje sa GP Timer 1; 6 – odre|uje da li }e tajmer startovati ili ne; 5-4 – odre|uju izvor clocka; itd... (pro~itati pod 5). Nad bitovima ovog registra se sprovodi operacija logi~ko ILI sa konstantom 0040h i rezultat opet sme{ta u T2CON=7408dh. Bit 6 T2CON registra postaje 1, {to zna~i: Timer 2 po~inje sa radom. Prakti~no, imamo da je 1042h sadr`aj registra T2CON.
13. Petlja: Program Monitor se aktivira u svakom ciklusu. Proverava se u svakom ciklusu da li je u najni`i bit adrese _stop=a000h upisana jedinica. Ako jeste to je uslov za izlazak iz petlje.
14. KRAJ

Pre startovanja tajmera 2 i izvr{enja prve interrupt rutine (a samim tim i pre prve AD konverzije) – konfigurisani su sadr`aji (za ovu aplikaciju) bitnih registara, kako je to gore obja{njeno. I sa po~etkom rada Timera 2 de{ava se slede}e:

Broja~ GP Timera 2 (sadr`aj adrese T2CNT=7405h) broji od po~etne vrednosti (korakom 0.05μs) do vrednosti sadr`ane u registru T2PER=7407h (period GP Timera 2, i u na{em slu~aju setovana perioda odabiranja). Tada dolazi do resetovanja broja~a (sadr`aj T2CNT postaje 0) i postupak se ponavlja (boja~ sada broji od 0 do vrednosti periode odabiranja, resetuje se, i sve tako...). U trenucima odabiranja, sadr`aji registara T2PER (7407h) i TCNT (7405h) su izjedna-~eni – zbog ~ega se generi{e GP Timer 2 period interrut (T2PINT) **flag**. Odnosno, u EV Interrupt Flag Registeru B (EVIFRB=7430h) se setuje bit 0 (bit 0 = 1, tj. EVIFRB[0]=1). U EV Interrupt Mask Registeru B (EVIMBR=742dh) odgovaraju}i bit je ve} setovan (bit 0 = 1, tj. EVIMBR[0]=1) – odnosno T2PINT nije maskiran. T2PINT ima ve}i nivo prioriteta u odnosu na druge interrupt zahteve ka INT3 (videti Tab. 2-10 u SPRU161A.PDF, drugo to DSP vidi iz vrednosti ID – u interrupt vektoru), te pre drugih konkurentnih interrupt izvora setuje bit 2 u CPUovom Interrupt Flag Registeru (IFR=0006h). Zapazimo da: po{to je EVIFRB[0]=1 i EVIMBR[0]=1, onda je IFR[2]=1. Kako je ve} IMR[2]=1 (CPUov Interrupt Mask Register; IMR=0004h), dolazi do realizacije T2PINT ako je INTM=0. Naime, tada se ~ita sadr`aj *tpint2vec* (KAKO zna da treba da pro~ita adresu tpint2vec=73h na kojoj je _t2per_ISR=803bh? – precizno objasniti! – u tom smislu pogeldati u SPRU160A.PDF str. 6-21, ali i Vects_a.h) i on se koristi kao startna adresa sa koje po~inje izvr{enje Interrupt Service Routine (ISR); - u SPRU160A.PDF pogledati sl. 6-9 na str. 6-36, zatim pogledati i str. 6-35. Definisana ISR se izvr{ava na kraju svake periode odabiranja – sve dok traje izvr{enje programa. ^im CPU prihvati zahtev za T2PINT (IFR[2]=1, IMR[2]=1 i INTM=0), odnosno ~im se u ACC pro~ita interrupt vektor koji odgovara flag bitu, resetuje se flag bit (IFR[2]=0) – to se de{ava hardverski (mogu}e je i softverski – upisivanjem 1 u odgovaraju}i flag bit).

A {ta se de{ava u prekidnoj rutini?

Evo algoritamskog prikaza de{avanja u prekidnoj rutini:

1. Snima se postoje}i sadr`aj AR0 i AR2 (to radi makro definisan u Demos_a.h).
2. Proverava se da li je sadr`aj adrese _count_adc=0004h (broj ostvarenih ciklusa AD konverzije) **manji od** vrednosti sadr`aja adrese _n_samples=a003h (broj zadatih odbiraka po jednom kanalu – zadaje se direktno u prozoru aplikacije i mo`e biti izme|u 2 i 1000). **Ako nije** (tj. ostvaren je zadati broj odbiraka AD konverzije) – na adresu _stop=a000h se upisuje 1, {to je uslov da program prekine sa radom, i izlazi se iz ISR. A **ako jeste** (tj. nije ostvaren zadati broj odbiraka AD konverzije), sledi granjanje ka novoj akviziciji; odnosno, ka new_aq:

new_aq:

3. Start AD konverzije – setuje se da je ADCTRL1[0]=1. Nad bitovima registra ADCTRL1=07032h sprovodi se operacija logi~ko ILI sa konstantom 0001h i taj rezultat se opet sme{ta na adresu ADCTRL1.
4. ^eka se kraj AD konverzije – dok je ADCTRL1[7]=1 vrtimo se u internoj petlji – "~ekamo". Bit 7 registra ADCTRL1 indicira status AD konverzije (da li je u toku (bit7=1) ili je zavr{ena (bit7=0)). AD konverzija }e trajati oko 6µs i tek onda se izlazi iz petlje i prakti~no nastavlja sa izvr{enjem programa.
5. ^ita se i ~uva rezultat AD konverzije sa ADC1 (sa izabranog I kanala). Sadr`aj registra ADCFIFO1=07036h (u koji je neposredno sme{ten rezultat teku}e AD konverzije sa ADC1) se sme{ta na adresu _res_buf + (_count_saved); odnosno na adresu: 0c000h (_res_buf) + sadr`aj adrese _count_saved=a005h (na ovu adresu se sme{ta broj snimljenih odbiraka u buffer sa **oba** kanala).
6. ^ita se i ~uva rezultat AD konverzije sa ADC2 (sa izabranog II kanala). Sadr`aj registra ADCFIFO2=07038h (u koji je neposredno sme{ten rezultat teku}e AD konverzije sa ADC2) se sme{ta na adresu _res_buf + (_count_saved) +1; odnosno na adresu: 0c000h (_res_buf) + 1 + sadr`aj adrese _count_saved=a005h (na ovu adresu se sme{ta broj snimljenih odbiraka u buffer sa **oba** kanala).
7. Sadr`aj adrese _count_saved=a005h se pove}ava za 2 (jer broji snimljene odbirke sa oba kanala); a sadr`aj adrese _count_adc=0004h se pove}ava za 1.
8. Sadr`aj AR0 i AR2 se obnavlja na onaj pre ulaska u prekidnu rutinu (to radi makro definisan u Demos_a.h).
9. KRAJ prekidne rutine.

Detaljniji uvid se mo`e ste}i na osnovu ponu|enog asemblerskog programa *adc_a.asm*, koji sledi u nastavku (listing programa je u boji, a obja{njenja i primedbe nisu).

```
;*******************************************************************************
; File Name: adc_a.asm
; Project:     MCK240
; Originator:  R.Giuclea
; Description:              ASM file for ADC demo
; Copyright © 1997 Technosoft
;*******************************************************************************
; Include Files
;-------------------------------------------------------------------------------          ...komentari ...
        .include    ..\F240_a.h
        .include    ..\demos_a.h
        .include    ..\vects_a.h
        .include    adc_a.h         ...program uklju~uje sadr`aj zaglavlja F240_a.h, demos_a.h, vects_a.h, i adc_a.h;
;===============================================================================
                .text               .text, .include, .global, itd. su asemblerske direktive. Tipi~no, direktive su
korisne da inicijalizuju ili rezervi{u memoriju, kontroli{u izgled listinga, uslovno objedine blokove koda, defini{u
globalne ili eksterne promenljive, itd. Direktive ne koriste memorijski prostor u kona~nom object modulu. To je zato
jer se one samo koriste da kontroli{u asembler.
                                    ...ova direktiva inicijalizuje sekciju koja sadr`i sve izvr{ne kodove - koji }e uvek
biti sme{teni u programsku memoriju. Naime, instrukcije asemblera koje slede ovu direktivu }e biti sme{tene u
memorijsku sekciju pod nazivom "text".
;-------------------------------------------------------------------------------------------------------------------
_start:                             ... labela
                                    ...sledi deo programa koji se odnosi na inicijalizaciju sadr`aja na adresi _stop,
odnosno: 0 -> (_stop). Kada postane _stop[0]=1 => kraj programa.
        LDP     #_stop              ... _stop = a000h (pi{e u  adc.map).
... Primedba: pro~itati direktno adresiranje: SPRU160A.PDF, strana 142-146.
... Setovan je data page pointer DP=320. Naime, starijih 9 bita a000h odre|uju vrednost pointera DP, a mla|ih
(najni`ih) 7 bita a000h odre|uju redni broj adrese na strani koju pokazuje pointer DP. Zna~i, u memoriji za podatke
odabrali smo stranu 320 (od mogu}ih 512 strana: 0-511) na kojoj mo`e biti sadr`ano 128 re~i; odnosno, adrese od
a000h do a07Fh su nam dostupne; i odabrali smo prvu re~ na toj strani.
        SPLK    #0,_stop            ... SPLK instrukcija omogu}ava da svih 16 bita budu neposredno upisani u
memorijsku lokaciju za podatke. Ovde je 0 taj 16 bitni podatak koji se spu{ta na adresu a000h (jer je _stop=a000h).
Zna~i, setuje se 0 kao sadr`aj adrese _stop, i to je sve tako dok se ne sempluje zadati broj odbiraka ili se klikom na
```

Stop u aplikaciji ne setuje 1 kao sadr`aj adrese _stop ({to je uslov za izlaz iz programa). Mogu}e je da se setovanje 1 na najni`em bitu adrese _stop izvr{i i kori{}enjem programa Monitor (recimo, uraditi za ve`bu!).

; timer will not start ADC automatically ...sledi deo programa: 1) (GPTCON=7400h) -> ACC 2) AND F9FFh 3) ACC -> (GPTCON), {to ima za rezultat GPTCON[9-10]=00, tj. onemogu}ava se startovanje AD konverzije doga|ajem na GP Timeru 2.

```
        LDP     #DP_EV                  ; Event Manager Data Page Pointer
```

...DP_EV je = 0E8h (videti F240_a.h); Setovana je nova vrednost data page pointera (DP=0E8h=232dec). – Ovde imamo 8 - bitni broj (0E8h), a da je 16-bitni – vrednost DP bi bila odre|ena sa starijih 9 bita.

```
        LACC    GPTCON
```

... GPTCON (General purpose timer control register)=7400h (adresa je definisana u F240_a.h)
... Primedba: progledati Event Manager Module: SPRU161A.PDF
... LACC naredba spu{ta sadr`aj sa adrese 07400h (GPTCON) u akumulator (ACC); (shiftovanje nije definisano iako je deo sintaksеLACC – tj. nema shiftovanja u konkretnom slu~aju); (7400h=232*128+0, TJ. 7 najni`ih bita u 07400h je zapravo 0dec, a slede}ih 9bita je E1h ili 232dec)

```
        AND     #AND_T2TOADC_           ; AND mask for disabling automaticaly ADC start on GPT2
```

... u Demos_a.h definisana je konstanta AND_T2TOADC_ = 0F9FFh
... operacija logi~ko I se sprovodi nad bitovima sadr`aja adrese 07400h (GPTCON) i 0F9FFh – resetuje se bit 9 i 10 akumulatora.

```
        SACL    GPTCON                  ... rezultat se sme{ta na adresu 07400h (GPTCON).
```

; load and init timer ... sledi deo programa: 1) (_timer_period=a002h) -> ACC (perioda oda-biranja je sada u akumulatoru); 2) ACC -> (T2PER=7407h) (setuje se period GP Timera 2); 3) 0 -> (T2CNT=7405h) (inicijalizuje se sadr`aj T2 Counter Registera); 4) 1002h -> (T2CON=7408h) (konfiguri{e se GP Timer 2);

```
        LDP     #_timer_period          ... _timer_period=0a002h (pi{e u adc.map).
```

... Sadr`aj adrese _timer_period = a002h odre|uje periodu odabiranja.
... Setuje se nova vrednost data page pointera (DP), tj. u memoriji za podatke biramo stranu odre|enu sa DP=320 (a002h ~ini vi{ih 9 bita (140h=320dec) i ni`ih 7 bita (2dec)).

```
        LACC    _timer_period           ... sadr`aj adrese _timer_period = a002h se spu{ta u akumulator -
```

(_timer_period) -> ACC.

```
        LDP     #DP_EV                  ...setuje se nova vrednost data page pointera (DP= DP_EV = 0E8h = 232dec).
        SACL    T2PER           ; load GPT2 timer period
```

... u F240_a.h je definisano da je T2PER = 7407h.
...na ovu adresu se sme{ta sadr`aj akumulatora (perioda odabiranja), i defini{e se period GP Timera 2.

```
        SPLK    #0h, T2CNT      ; reset GPT2 counter register                   ... 0 -> (T2CNT)
```

... T2CNT = 7405h (videti F240_a.h) – i radi se o adresi T2 Counter Registera (sadr`aj je broja~ GP Timera 2)
... SPLK instrukcija omogu}ava da svih 16 bita budu neposredno upisani u memorijsku lokaciju za podatke. Ovde je 0 taj 16 bitni podatak koji se spu{ta na adresu 7405h.

```
        SPLK    #T2CNFREG, T2CON; configure timer in countinuous up mode, prescaler X1
```

...T2CON = 7408h (videti F240_a.h) – i radi se o adresi T2 Control Registera.
...T2CNFREG = 1002h (videti adc_a.h) – i radi se o konstanti.
...konstanta 1002h (svih 16 bita) se spu{ta na adresu 7408h (T2CON), tj. 1002h -> (T2CON) – konfiguri{e se GP Timer 2 (videti SPRU161A.PDF, odeljak 2.3.3. i ispred).

; init ADC registers ... sledi deo programa: 1) 0403h ->(ADCTRL2) 2) (_adcctrl_value=a001h) -> (ADCTRL1)

```
        LDP     #DP_PF1                 ; ADC Registers Data Page Pointer
```

... DP_PF1 = 0E0h (videti F240_a.h).
... Setuje se nova vrednost data page pointera DP=224 (0E0h).

```
        LACC    #ADC_CONF_2
```

... ADC_CONF_2 = 403h (videti adc_a.h) – ova konstante se upisuje u ACC.

```
        SACL    ADCTRL2                 ; set ADC Configuration Register
```

... ADCTRL2 = 07034h (videti F240_a.h), i na ovu adresu se sme{ta sadr`aj akumulatora.
... Primedba: pro~itati glavu 3 u SPRU161A.PDF.

```
        LDP     #_adcctrl_value
```

... _adcctrl_value =0a001h (videti adc.map) - sadr`aj ove adrese izme|u ostalog odre|uje izbor (oba) kanala A/D konverzije (pro~itati glavu 3 u SPRU161A.PDF).
... starijih 9 bita a001h odre|uju vrednost pointera DP (=320)

```
        LACC    _adcctrl_value          ... sadr`aj _adcctrl_value = a001h se spu{ta u akumulator.
        LDP     #DP_PF1
```

... DP_PF1 = 0E0h (videti F240_a.h).
... Setuje se nova vrednost data page pointera DP=224 (0E0h).

```
        SACL    ADCTRL1                 ; set ADC Control & Status Register
```

... ADCTRL1 = 07032h (videti F240_a.h), i na ovu adresu se sme{ta sadr`aj akumulatora: ACC -> (ADCTRL1).

; load ISR addresses to Interrupt Vector in on-chip block B2

... sledi deo programa radi koga je po`eljno pro~itati Interrupte (str. 6-9 u SPRU160A.PDF) i EV Interrupte (str. 2-87 u SPRU161A.PDF). I obratiti pa`nju na sadr`aj Vects_a.h.

```
        LACC    #_t2per_ISR
```

... _t2per_ISR = 0803bh (videti adc.map); ova konstanta postaje sadr`aj akumulatora.

```
        LDP     #0              ... Setuje se nova vrednost data page pointera DP=0
        SACL    tpint2vec       ; load _t2per_ISR address into corresponding
                                ; interrupt vector
... tpint2vec = 060h (B2_SADDR)+13h =73h (=115) - TPINT2 vector address – def. u F240_a.h i Vects_a.h;
... na ovu adresu se sme{ta sadr`aj akumulatora
; init count variables         ... sledi deo programa: 1) 0 -> (_count_adc=a004) 2) 0 -> (_count_saved=a004)
– radi se o inicijalizaciji broj~kih promenljivih.
        LDP     #_count_adc
... _count_adc=a004h (videti adc.map) – sadr`aj ove adrese se odnosi na aktuelni broj odbiraka A/D konverzije – sa
svakog kanala pojedina~no.
... starijih 9 bita a004h odre|uju vrednost pointera DP (=320)
        LACC    #0              ... 0 se upisuje u akumulator.
        SACL    _count_adc      ...na adresu _count_adc=a004h (videti adc.map) se sme{ta sadr`aj akumulatora.
        SACL    _count_saved
... na adresu _count_adc=a005h (videti adc.map) se sme{ta sadr`aj akumulatora.
... sadr`aj _count_adc=a005h je promenljiva koja broji broj odbiraka sme{tenih u buffer – odnosno, broji aktuelni
broj odbiraka A/D konverzije – ali sa oba kanala zajedno (ukupan broj snimljenih odbiraka).
; unmask interrupts            ... sledi deo programa: 1) (IMR=0004h) ILI 0004h -> (IMR=0004h) (Primedba:
pro~itati od str. 6-13 do 6-19 u SPRU160A.PDF) 2) (IMRB=742dh) ILI 0001h -> (IMRB=742dh) (Primedba:
pro~itati od str. 2-87 do 2-98 u SPRU161A.PDF).
        LDP     #0              ... setuje se nova vrednost DP=0
        SETBIT IMR,SETB2        ; unmask INT3
... SETBIT je makro koji je definisan u Demos_a.h
... IMR     =   0004h    ; Interrupt Mask Register      (videti F240_a.h)
... SETB2   =   0004h    ; Bit Mask for 2               (videti F240_a.h)
... Su{tina je da se nad bitovima sadr`aja adrese IMR sprovede operacija logi~ko ILI sa vredno{}u konstante SETB2 i
taj rezultat se sme{ta opet na adresu IMR. => INT3 nije vi{e maskiran.
        LDP     #DP_EV
... DP_EV = 0E8h (videti F240_a.h); Setuje se nova vrednost DP (DP=0E8h=232dec).
        SETBIT IMRB,SETB0       ; enable T2PINT (activate GPT2 period interrupt generation)
... SETBIT je makro koji je definisan u Demos_a.h
... IMRB    =   742dh    ; Group B Interrupt Mask Register   (videti F240_a.h)
... SETB0   =   0001h    ; Bit Mask for 0                    (videti F240_a.h)
... Su{tina je da se nad bitovima sadr`aja adrese IMRB sprovede operacija logi~ko ILI sa vredno{}u konstante SETB0
i taj rezultat se sme{ta opet na adresu IMRB.
; start timer                  ... startuje se Timer 2: 1) (T2CON=7408h) ILI 0040h -> (T2CON=7408h)
        SETBIT T2CON,SETB6      ;start GPT2
... SETBIT je makro koji je definisan u Demos_a.h
... T2CON =  7408h    ; T2 Control Register              (videti F240_a.h)
... SETB6 =  0040h    ; Bit Mask for 6                   (videti F240_a.h)
... Su{tina je da se nad bitovima sadr`aja adrese T2CON sprovede operacija logi~ko ILI sa vredno{}u konstante
SETB6 i taj rezultat se sme{ta opet na adresu T2CON => start GP Timer 2.
loop:                           ... labela
; call monitor
        CALL    MON240          ... Programski broja~ (PC-program counter) se inkrementira i stavlja na vrh
steka. Sadr`aj adrese u programskoj memoriji MON240=0109h (videti adc_a.h). postaje sadr`aj programskog
broja~a. Poziva se program monitor.
; test if demo ends (_stop =1)  ... sledi deo programa: Ispituje se da li je _stop[0]=1? Ako jeste => kraj
programa; Ako nije => ostajemo u petlji.
        LDP     #_stop          ... setuje se DP=320
        BIT     _stop,15        ... posmatra se sadr`aj na adresi _stop. Specificirani bit code=15, odnosi se na
najni`i bit (LSB) na posmatranoj adresi. Instrukcija BIT kopira ovaj bit u TC bit status registra ST1. TC (test/control
flag bit) – je bit 11 status registra ST1 - ~uva rezultat operacije testiranja.
        BCND    loop,NTC        ... Ako je uslov NTC ispunjen (tj. ako je TC=0) program ide na loop (labela –
gore). Odnosno, ako je klikom na Stop u prozoru aplikacije setovano: _stop[0]=1 => izlazak iz petlje.
;
        END_DEMO                ... END_DEMO je makro, definisan u Demos_a.h; Kada bude setovano _stop=1,
tj. da je TC=1, program prestaje da se izvr{ava.
;-------------------------------------------------------------------------------------------------------------------
; timer 2 period interrupt service routine:     ... Sledi listing prekidne rutine
;
_t2per_ISR:                     ... labela; _t2per_ISR = 0803bh (videti adc.map)
        SAVE_AR0_AR2            ... makro definisan u Demos_a.h ~ija je svrha da sa~uva (snimi) sadr`aj
registara AR0 i AR2.
;
```

; test if (_count_adc = _n_samples) ... sledi deo programa: 1) (_count_adc=a004h) - (_n_samples =a003h) -> ACC; 2) Ako je ACC<0 => nova konverzija, tj. program se grana ka new_aq; 3) Ako nije ACC<0 => 1 -> (_stop)

```
        LDP     #_count_adc
```

... _count_adc=a004h (videti adc.map) – sadr`aj ove adrese se odnosi na aktuelni broj odbiraka A/D konverzije.
... starijih 9 bita a004h odre|uju vrednost pointera DP (=320).

```
        LACC    _count_adc      ... sadr`aj adrese _count_adc=a004h se spu{ta u akumulator.
        SUB     _n_samples      ... _n_samples =a003h (videti adc.map) – sadr`aj ove adrese sadr`i zadati broj odbiraka koji treba da bude napravljen.
```

... vrednost u akumulatoru se umanjuje za vrednost na adresi _n_samples i rezultat operacije se sme{ta u akumulator (ACC-(_n_samples) -> ACC).

```
        BCND    new_aq, LT      ; _count_adc < _n_samples, new aquisition
```

... Ako je ACC<0 (uslov LT ispunjen) program ide na new_aq (labela)

```
        SPLK    #1, _stop       ; else, _count_adc = _n_samples, set stop index
```

... Upisuje se 1 kao sadr`aj adrese _stop=a000h, {to je uslov da program prekine sa radom.

```
        B       end_routine     ;       last aquisition has been made
```

... end_routine je labela koja vodi ka kraju prekidne rutine i povratku u glavni program. Program se grana ka labeli end_routine.

```
new_aq:                         ... labela
; start A/D conversion          ... (ADCTRL1=07032h) ILI 0001h -> (ADCTRL1=07032h)
        LDP     #DP_PF1         ... DP_PF1 = 0E0h (videti F240_a.h); Setuje se nova vrednost DP=224 (0E0h).
        SETBIT ADCTRL1,SETB0    ;set SOC bit to start A/D conversion
```

... SETBIT je makro koji je definisan u Demos_a.h
... ADCTRL1 = 07032h (videti F240_a.h)
... SETB0 = 0001h ; Bit Mask for 0 (videti F240_a.h)
... Su{tina je da se nad bitovima sadr`aja adrese ADCTRL1 sprovede operacija logi~ko ILI sa vredno{}u konstante SETB0 i taj rezultat se sme{ta opet na adresu ADCTRL1 => ADCTRL1[0]=1

```
; wait end of conversion        ... ~eka se kraj AD konverzije – vrtimo se u petlji sve dok je ADCTRL1[7]=1 (dok je AD konverzija u toku).
pool:   LACL    ADCTRL1         ... ADCTRL1 = 07032h (videti F240_a.h)
```

...sadr`aj adresirane lokacije se sme{ta u Akumulator. Pri tome, gornja polovina bitova (15-8) akumulatora su nule (0)

```
        AND     #EOC_MSK
```

... EOC_MSK = 80h (videti adc_a.h); mask for testing bit 7 (end of conversion bit)
... U akumulatoru su svi bitovi 0, sem eventualno bita 7. Prakti~no u slede}em koraku proveravamo da li je ADCTRL1[7]=0 (da li je AD konverzija zavr{ena).

```
        BCND    pool, NEQ       ; loop untill EOC (= 0)
```

...Ako je ACC≠0 (uslov NEQ ispunjen) program ide na pool (labela).

```
; read and store A/D conversion results
```

... sledi deo programa: 1) (ADCFIFO1= 07036h) -> res_buf=0c000h + (_count_saved), tj. ~itamo i ~uvamo rezultat AD konverzije sa ADC1.

```
        LACL    ADCFIFO1        ; load ch_0_7 result
```

... ADCFIFO1 = 07036h (videti F240_a.h).
... sadr`aj adresirane lokacije se sme{ta u akumulator. (ADCFIFO1)->ACC(15-0)

```
        LDP     #_count_saved
```

... sadr`aj _count_saved=a005h broji broj odbiraka sme{tenih u *buffer*
... Setuje se nova vrednost data page pointera DP=320 (a005h ~ini vi{ih 9 bita (140h=320dec) i ni`ih 7 bita (5dec)).

```
        LAR     AR0,_count_saved; AR0 = _count_saved value
```

... sadr`aj adrese _count_saved=a005h se preme{ta u pomo}ni registar AR0: (_count_saved)->(AR0)

```
        LAR     AR2, #_res_buf  ; AR2 = &_res_buf
```

... _res_buf=0c000h (videti adc.map) – *buffer* od 2000 re~i u koji }e se sme{tati rezultati AD konverzije zauzima memorijsku lokaciju od adrese 0c000h (i narednih 1999 adresa je raspolo`ivo).
... **_res_buf** =0c000h (konstanta) se preme{ta u pomo}ni registar AR2: _res_buf -> (AR2)

```
        LARP    AR2             ; ARP = 2
```

... u ARP (auxiliaru register pointer) se upisuje 2, posle ~ega je teku}i pomo}ni registar AR2.

```
        MAR     *0+             ; AR2 = &(_res_buf + _count_saved)
```

... dodaje se sadr`aj AR0 na sadr`aj teku}eg pomo}nog registra AR2: _res_buf+(_count_saved) -> (AR2)

```
        SACL    *,AR0           ; _res_buf[_count_saved] = ch_0_7 result, new AR0
```

... sadr`aj akumulatora se kopira na adresu koju pokazuje sadr`aj AR2 – zna~i ACC(15-0) -> _res_buf+(_count_saved). I sada AR0 postaje teku}i pomo}ni registar (tj. ARP=0).

```
        MAR     *+,AR2          ;_count_saved++, new AR2
```

... sadr`aj AR0 se pove}ava za 1. U ARP se upisuje 2 (ARP=2), posle ~ega je teku}i pomo}ni registar AR2.

... sledi deo programa: 1) (ADCFIFO2= 07038h) -> res_buf=0c000h +1+(_count_saved), tj. ~itamo i ~uvamo rezultat AD konverzije sa ADC2.

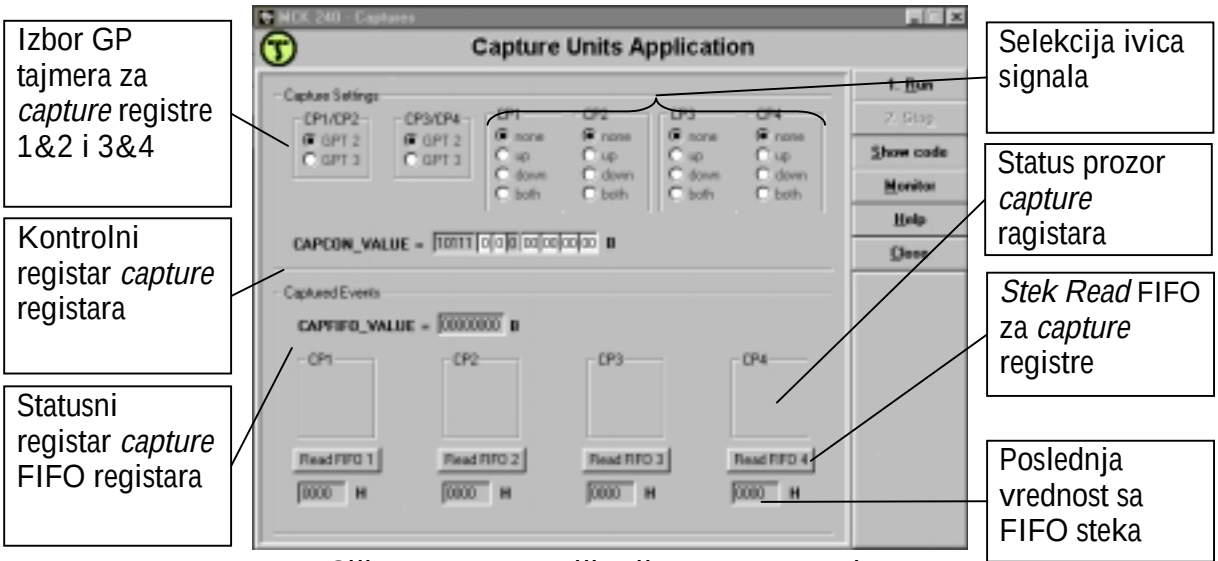
```
        LDP     #DP_PF1         ... DP_PF1 = 0E0h (videti F240_a.h).
```

... Setuje se nova vrednost data page pointera DP=224 (0E0h).

```
        LACL    ADCFIFO2        ; load ch_8_F result
... ADCFIFO2    = 07038h (videti F240h).
... sadr`aj adresirane lokacije se sme{ta u akumulator: (ADCFIFO2= 07038h) -> ACC(15:0).
        LAR     AR2, #_res_buf  ; AR2 = &_res_buf
... _res_buf=0c000h (videti adc.map)
... _res_buf  =0c000h (konstanta) se preme{ta u pomo}ni registar AR2: _res_buf -> (AR2)
        MAR     *0+             ; AR2 = &(_res_buf + _count_saved)
... dodaje se sadr`aj AR0 na sadr`aj teku}eg pomo}nog registra - AR2: (AR0)+(AR2)->(AR2), odnosno:
(_count_saved)+1   +   _res_buf   ->   (AR2)
        SACL    *, AR0          ; _res_buf[_count_saved] = ch_8_F result, new AR0
... ACC(15-0) -> (AR2) – sadr`aj ACC se kopira na adresu koju pokazuje sadr`aj AR2.
tj. ACC(15-0) ->(_count_saved)+1+_res_buf;
... AR0 postaje aktuelni AR (ARP=0).
        MAR     *+              ; _count_saved++
... dodaje se 1 na sadr`aj teku}eg pomo}nog registra – AR0, tj. sadr`aj _count_saved je uve}an za jo{ 1.
; save counter of A/D saved conversions
        LDP     #_count_saved
...sadr`aj _count_saved=a005h broji odbirrke sme{tene u buffer; Setuje se DP=320.
        SAR     AR0,_count_saved; save new _count_saved value
...sadr`aj AR0 se stavlja na adresu _count_saved=a005h (prakti~no, sadr`aj a005h je uve}an za 2)
; increment counter of A/D made conversions                    ... sada se sadr`aj _count_adc pove}ava za 1
        LDP     #_count_adc                                    ... DP=320
        LACC    _count_adc                                     ... (_count_adc) -> ACC
        ADD     #1                                             ... ACC+1-> ACC
        SACL    _count_adc      ; save new _count_adc value    ... ACC -> (_count_adc)
end_routine:                                                   ... labela
        END_ISR_AR0_AR2                                        ... makro definisan u Demos_a.h – sadr`aj
AR0 i AR2 se obnavlja na onaj pre ulaska u prekidnu rutinu. Sledi povratak u glavni program.
;-------------------------------------------------------------------------------------------------------------------------------
```

3.3.4. *Capture* registri

Ova aplikacija pokazuje kako prihvatni registri detektuju promene ulaznih signala (uzlazna ivica, silazna ivica ili obe). Ova aplikacija se startuje pritiskom na ikonu ***Captures*** u ***Processor Evaluation Control Panel*** prozoru. Na slede}oj slici je dat grafi~ki prikaz ove aplikacije:



Slika 15. Prozor aplikacije za *capture* registre

Asemblerski program koji sadr`i kod ove aplikacije je ***cap_a.asm*** i mo`e se videti pritiskom na taster ***Show code***. Zaglavlje asemblerskog programa je ***cap_a.h*** koje sadr`i promenljive i funkcije za ovu aplikaciju. Startna adresa programa je **8000h** u eksternoj programskoj memoriji i zove se **_start**. Programske promenljive su u eksternoj memoriji za podatke i po~inju od adrese **a000h**, po slede}em rasporedu: **_stop (=a000h), _capcon_value (a001h), _capfifo_value (a002h),**

_read_fifo_no (a003h), _fifo1_value (a004h), _fifo2_value (a005h), _fifo3_value (a006h), _fifo4_value (a007h).

Va`ne adrese (kori{}ene u aplikaciji):

_stop = a000h – sadr`i indeks za stopiranje programa (1 za kraj programa)
_capcon_value = a001h – sadr`i vrednost za CAPCON registar. Naime, korisnik aplikacije, izborom "doga|aja" koje }e registrovati capture pinovi, i tajmera koji }e podr`avati rad capture jedinica, zapravo setuje sadr`aj **_capcon_value** koji }e u asemblerskom programu poslu`iti za konfigurisanje CAPCON registra.
_capfifo_value = **a002h** – ~uva vrednost CAPFIFO registra, odnosno sadr`i bitove statusa za svaki od ~etiri FIFO steka.
_read_fifo_no = a003h – pokazuje vrednost *capture* jedinice ~iji se stek mo`e ~itati. Svaki put kada se klikne na **Read FIFO** (u komandnom prozoru aplikacije) u ovu varijablu se upisuje odgovaraju}a vrednost (1, 2, 3 ili 4).
_fifo1_value = a004h – ~uva poslednju vrednost sa FIFO1 stek registra
_fifo2_value = a005h – ~uva poslednju vrednost sa FIFO2 stek registra
_fifo3_value = a006h – ~uva poslednju vrednost sa FIFO3 stek registra
_fifo4_value = a007h – ~uva poslednju vrednost sa FIFO4 stek registra

Prihvatne jedinice (Capture units) TMS320C24x DSP (op{ti pojmovi)

Capture units (capture – sakupljanje, hvatanje) – *prihvatne jedinice* omogu}avaju registrovanje doga|aja, odnosno registrovanje promena na *capture* ulaznim pinovima. Postoje 4 prihvatne jedinice (Capture Units 1, 2, 3, i 4) i svaka od njih je povezana sa odgovaraju}im *capture* ulaznim pinom. Tako|e, svaka od njih mo`e da izabere GP Timer 2 ili 3 kao svoju vremensku bazu. Vrednost GP Timera 2 ili 3 se "uhvati" (registruje) i ~uva u odgovaraju}em FIFO steku (sa 2 nivoa dubine), kada se detektuje specificirana promena na *capture* ulaznom pinu – CAPx (x=1÷4).

Po{to je jedinica za registraciju doga|aja (capture unit) aktivirana (tj. tako setovana da je u funkciji), specificirana promena na pridru`enom ulaznom pinu prouzrokuje da vrednost broja~a (u tom trenutku!) selektovanog GP Timera bude sa~uvana na odgovaraju}em FIFO steku. Istovremeno, odgovaraju}i interapt flag je setovan, i, ako falg nije maskiran i nema drugih nemaskiranih interapta u istoj grupi prioriteta koji mogu da se o~ekuju, generi{e se interapt zahtev ka CPU. Odgovaraju}i bitovi statusa u CAPFIFO su prilago|eni (pode{eni) da poka`u novi status FIFO steka svakog trenutka kada je nova vrednost broja~a "uhva}ena" u FIFO stek. Ka{njenje od trenutka de{avanja promene na *capture input-u* do trenutka kada se registruje vrednost broja~a GP Timera – iznosi CPU clock ciklusa. "Uhva}ena" vrednost broja~a mo`e biti pro~itana iz ISR (Interrupt Service Routine) ako je interapt kori{}en. Ako interapt nije po`eljan ili jednostavno nije odabran - u ovom slu~aju mogu biti kori{}eni ili interapt flag ili bitovi statusa – da signaliziraju da "uhva}ena" vrednost broja~a mo`e biti pro~itana.

Jedinice za registrovanje doga|aja (capture units) su kontrolisane sa dva 16-bitna registra CAPCON i CAPFIFO. Registri T2CON i T3CON (za kontrolu rada tajmera GPT2 i GPT3) se tako|e koriste, po{to vremenska baza za *capture* kola je obezbe|ena sa GP Timer 2 ili 3.

Za ispravno funkcionisanje *capture unit* treba da bude izvr{eno pode{avanje slede}ih registara:

1. Inicijalizacija CAPFIFO registra (na adresi 7422h). Brisanje odgovaraju}ih bitova statusa.
2. Pode{ava se selektovani GP timer u jednom od njegovih operativnih modova.
3. Setuju se vrednosti pripadaju}eg GP timer compare registera ili GP timer period registera, ako je neophodno.
4. Setuju se bitovi CAPCON (Capture Control Registera – na adresi 7420h)

Svaki *capture unit* ima svoj namenski FIFO stek (sa dva nivoa dubine). Registar na vrhu FIFO steka mo`e da pru`i samo ~itanje sadr`aja, i uvek sadr`i stariju vrednost broja~a, “uhva}enu” sa odogovaraju}im *capture unit*. Dakle, ~itanjem FIFO steka (pripadaju}e prihvatne jedinice) dolazi se do starije vrednosti broja~a, stavljene na stek. Kada se starija vrednost - na vrhu steka - pro~ita, novija vrednost broja~a u registru na dnu steka se stavlja na vrh steka (First Input First Output – FIFO).

- Vrednost broja~a selektovanog GP Timera koja je “uhva}ena” sa capture jedinicom kada se specificirana promena dogodila na njegovom ulaznom pinu }e biti upisana na vrh steka ako je stek prazan. Istovremeno, odgovaraju}i status bitovi se setuju na 01 (u CAPFIFO registru - na adresi 7422h). Status bitovi se resetuju na 00 ako je izvr{eno ~itanje FIFO steka pre nego {to je napravljena slede}a registracija doga|aja na ulaznom pinu.
- Ako se druga registracija doga|aja (capture) desi pre nego {to je ve} “uhva}ena” vrednost broja~a pro~itana, “novouhva}ena” vrednost broja~a ide u donji registar steka. U tom slu~aju, odgovaraju}i status bitovi se setuju na 10. Kada se FIFO stek pro~ita pre nego {to se desi slede}e “hvatanje”, starija vrednost broja~a u registru na vrhu steka }e biti pro~itana, a novija vrednost u donjem registru }e biti gurnuta u registar na vrhu steka i odgovaraju}i status bitovi }e biti setovani na 01.
- Ako se “hvatanje” desi kada ve} postoje dve vrednosti broja~a koje su registrovane u FIFO steku, starija vrednost broja~a u registru na vrhu steka }e biti “izgurana” i izgubljena, vrednost boja~a u donjem registru steka }e se “popeti” u registar na vrhu steka, a “novouhva}ena” vrednost broja~a }e biti upisana u donji registar, i status bitovi se setuju na 11 – da poka`u da je jedan ili vi{e starijih vrednosti broja~a izgubljeno.

Analiza ponu|enog programskog re{enja aplikacije

Ponu|eni asemblerski program *cap_a.asm* je zasnovan na slede}em algoritmu:

1. Setuju se bitovi slede}ih registara:
    a. 0 se upisuje na adresu _stop=a000h. Kada najmla|i bit na ovoj adresi postane 1 (_stop[0]=1) tada program prestaje sa radom.
    b. 0 se upisuje na adresu _read_fifo_no=a003h. Sadr`aj ovog registra mo`e biti 0, 1, 2, 3 ili 4, i pokazuje u programu da li se ~ita sadr`aj registra sa steka pripadaju}e prihvatne jedinice 1, 2, 3, ili 4 (capture unit 1, 2, 3, ili 4) ili nema ~itanja steka – 0. Izbor se pravi klikom na dugme **Read FIFO1, 2, 3**, ili **4** – u komandnom prozoru aplikacije.
    c. 0 se upisuje na mesto bita 15 na adresi CAPCON=7420h. Setovan je bit CAPRES CAPCON registra: CAPRES=CAPCON[15]=0 => bri{u se sadr`aji svih registara vezanih capture units.
    d. 1111 se upisuje u OPCRB[4-7] => pinovi 4, 5, 6 i 7 porta B igraju ulogu Capture Pinova 1, 2, 3, i 4. Naime, konfiguri{u se bitovi Output Control Registera B (OPCRB=7092h) koji odre|uju funkciju pinova 4-7 porta B (pogledati str. 11-15 u Spru161a.pdf).
    e. 00 se upisuje u GPTCON[9-10] => spre~ava da GP Timer 2 startuje AD konverziju; GPTCON=7400h - General Purpose Timer Control Register (v. str. 2-38 u Spru161a.pdf)
    f. 00 se upisuje u GPTCON[11-12] => spre~ava se da GP Timer 3 startuje AD konverziju.
    g. FFFFh se upisuje na adresu T2PER=7407h.
    h. FFFFh se upisuje na adresu T3PER=740bh.
    i. 0 se upisuje na adresu T2CNT=7405h – sadr`aj broja~a GP Timera 2 se postavlja na 0.
    j. 0 se upisuje na adresu T3CNT=7409h – sadr`aj broja~a GP Timera 3 se postavlja na 0.

k. 1702h se upisuje na adresu T2CON=7408h - konfiguri{e se GPT2
l. 1702h se upisuje na adresu T3CON=740ch - konfiguri{e se GPT3
m. sadr`aj adrese _capcon_value=a001h se upisuje na adresu CAPCON=7402h. Sadr`aj adrese _capcon_value=a001h se setuje u komandnom prozoru aplikacije shodno izboru tajmera i doga|aja koje treba registrovati. Konfiguri{e se registar CAPCON (videti str. 2-75 u Spru161a.pdf). Ono {to se u svakom slu~aju setuje sadr`ajem ovog registra (za ovu aplikaciju) je da se omogu}ava rad sve 4 prihvatne jedinice i da registrovani "doga|aji" ne}e uticati na start AD konverzije.
n. 1 se upisuje u bit T2CON[6] => start GP Timera 2
o. 1 se upisuje u bit T3CON[6] => start GP Timera 3

4. petlja:
   a. Sadr`aj registra CAPFIFO=7422h se upisuje na adresu _capfifo_value=a002h. Registar CAPFIFO sadr`i bitove statusa za svaki od ~etiri FIFO steka, dok se sadr`aj _capfifo_value=a002h mo`e ~itati iz komandnog prozora aplikacije.
   b. Program Monitor se aktivira (u svakom ciklusu).
   c. Ispituje se sadr`aj adrese _read_fifo_no=a003h – da li je 1, 2, 3, ili 4; Odnosno, da li je korisnik aplikacije kliknuo na *Read FIFO* 1, 2, 3, ili 4 – da bi pro~itao sadr`aj registra na vrhu steka pripadaju}e capture unit (upisuje se u komandnom prozoru aplikacije). Obavlja se procedura "~itanja" steka:
      - ako je (_read_fifo_no)=1 => (FIFO1=7423h)->(_fifo1_value=a004h)
      - ako je (_read_fifo_no)=2 => (FIFO2=7424h)->(_fifo1_value=a005h)
      - ako je (_read_fifo_no)=3 => (FIFO3=7425h)->(_fifo1_value=a006h)
      - ako je (_read_fifo_no)=4 => (FIFO4=7426h)->(_fifo1_value=a007h)
   d. 0 se upisuje na adresu _read_fifo_no=a003h.
   e. Proverava se u svakom ciklusu da li je u najni`i bit adrese _stop=a000h upisana jedinica ({ro se de{ava pritiskom na Stop u komandnom prozoru aplikacije). Ako jeste to je uslov za izlazak iz petlje. Ako ne, sledi povratak na po~etak petlje.

3. KRAJ

Detaljniji uvid se mo`e ste}i na osnovu ponu|enog asemblerskog programa *cap_a.asm*, koji sledi u nastavku (listing programa je u boji, a obja{njenja i primedbe nisu).

```
;***************************************************************************
; File Name: cap_a.asm
; Project:     MCK240
; Originator:  R.Giuclea
; Description:          ASM file for Capture Units demo
; Copyright © 1997 Technosoft
;***************************************************************************
; Include Files
;---------------------------------------------------------------------------
                        .include    ..\F240_a.h
                        .include    ..\demos_a.h
                        .include    cap_a.h
;=============================================================
                        .text
; ------------------------------------------------------------------------------------------------
_start:                                 ... labela
        LDP     #_stop                  ... _stop = a000h (v.  cap.map). Starijih 9 bita a000h odre|uju pointer DP=320.
        SPLK    #0,_stop                ... SPLK instrukcija omogu}ava da svih 16 bita budu neposredno upisani u memorijsku lokaciju za podatke. Ovde je 0 taj 16 bitni podatak koji se spu{ta na adresu a000h (jer je _stop=a000h). Zna~i, setuje se 0 kao sadr`aj adrese _stop, i to je sve tako dok se klikom na Stop u aplikaciji ne setuje 1 kao sadr`aj adrese _stop ({to je uslov za izlaz iz programa).
        SPLK    #0,_read_fifo_no ... 0 postaje 16-tobitni sadr`aj adrese _read_fifo_no=a003h (videti cap.map).
... sadr`aj adrese _read_fifo_no odre|uje sa kog steka ~itamo (biramo izme|u ~etiri jedinice za registraciju (capture unit) - i to u prozoru aplikacije)
        LDP     #DP_EV              ... DP_EV je = 0E8h (v. F240_a.h); Setovana je nova vrednost data page pointe-ra (DP=0E8h=232dec). (Imamo 8 - bitni broj (0E8h), a da je 16-bitni – DP bi bio odre|en sa starijih 9 bita.)
        RESBIT CAPCON,RSTB15        ; reset all Capture Unit Registers
... radi se o makrou koji je definisan u Demos_a.h; Radi se o resetovanju 15tog bita registra CAPCON, tj. CAPCON[15]=0; RSTB15=7FFFh (videti F240_a.h).
... CAPCON[15]=0 => bri{u se (resetuju se) svi registri vezani za capture units (videti Spru161a.pdf, str.107 – poglavlje 2.8.3)
```

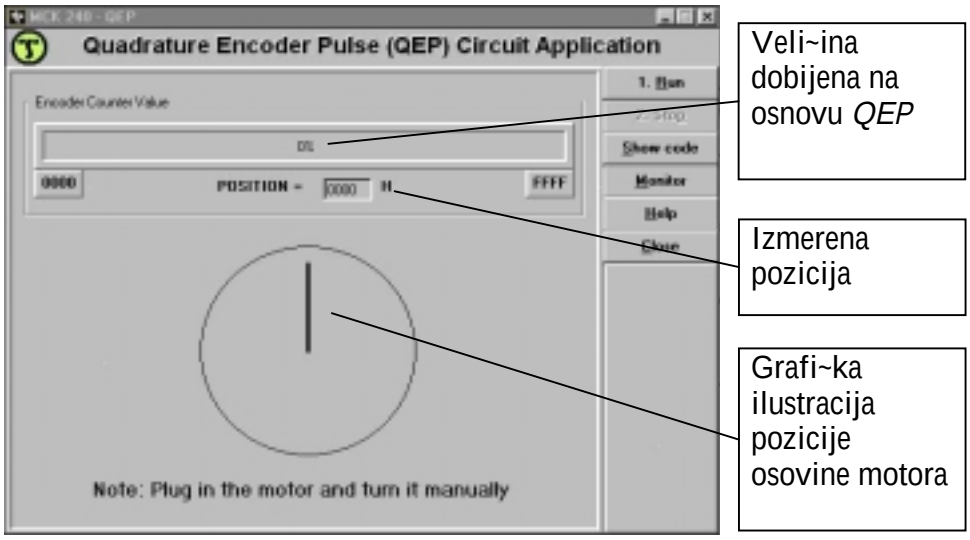
```
; configure shared pins as capture input pins
        LDP     #DP_PF2                 ; Peripheral File 2 Data Page Pointer
... DP_PF2 je = 0E1h (videti F240_a.h); Setovana je nova vrednost data page pointera (DP=0E1h=225dec).
        LACC    OPCRB           ... OPCRB=07092h    ; Output Control Register B
... Sadr`aj registra OPCRB=7092h (videti F240_a.h) se upisuje u akumulator (ACC).
        OR      #CAPENMSK       ... CAPENMSK=00F0h (v. cap_a.h).
        SACL    OPCRB           ; CAPx (x=1-4) pins configured capture input pins
... Sadr`aj ACC se upisuje na adresu OPCRB=7092h, i pri tom je sasvim sigurno OPCRB[4-7]=1111, ~ime je odlu~no da pinovi 4-7 porta B igraju ulogu Capture pinova (videti tabelu 11-6 u Spru161a.pdf); OPCRB[4] zna~i konfiguraciju pina 4 kao CAP1, a OPCRB[7] zna~i konfiguraciju pina 7 kao CAP4.
;
        LDP     #DP_EV                  ; Event Manager Data Page Pointer
...DP_EV je = 0E8h (videti F240_a.h); Setovana je nova vrednost data page pointera (DP=0E8h=232dec).
; init and start timers GPT2 and GPT3
        LACC    GPTCON
... GPTCON (General purpose timer control register)=7400h (adresa je definisana u F240_a.h)
... Primedba: pogledati Event Manager Module: SPRU161A.PDF
... LACC naredba spu{ta sadr`aj sa adrese 07400h (GPTCON) u akumulator (ACC);
        AND     #AND_T2TOADC_           ; AND mask for DISABLING ADC start on GPT2
... u Demos_a.h definisana je konstanta AND_T2TOADC_      = 0F9FFh
... operacija logi~ko I se sprovodi nad bitovima sadr`aja adrese 07400h (GPTCON) i 0F9FFh => resetuje se bit 9 i 10 akumulatora.
        SACL    GPTCON                  ; configure GPTCON not to start ADC on GPT2 Event
... rezultat se sme{ta na adresu 07400h (GPTCON)
... AD konverzija ne}e biti startovana “doga|ajem“ koji registruje GP Timer 2
        AND     #AND_T3TOADC_           ; AND mask for DISABLING ADC start on GPT3
... operacija logi~ko I se sprovodi nad bitovima akumulatora i AND_T3TOADC_=0E7FFh  – resetuje se bit 11 i 12 akumulatora.
        SACL    GPTCON                  ; configure GPTCON not to start ADC on GPT3 Event
... rezultat se sme{ta na adresu 07400h (GPTCON) - AD konverzija ne}e biti startovana “doga|ajem“ koji registruje GP Timer 3, a niti “doga|ajem“ koji registruje GP Timer 2
;
        LACC    #0FFFFh         ... u akumulator se upisuje konstanta 0FFFFh
        SACL    T2PER           ; initialize GPT2 timer period
... u F240_a.h je definisano da je T2PER = 7407h.
... na ovu adresu se sme{ta sadr`aj akumulatora  - 0FFFFh, i defini{e se period GP Timera 2.
        SACL    T3PER           ; initialize GPT3 timer period
... u F240_a.h je definisano da je T3PER = 740bh.
... na ovu adresu se sme{ta sadr`aj akumulatora  - 0FFFFh, i defini{e se period GP Timera 3.
        LACC    #TIM_CNT_INI            ... TIM_CNT_INI=0h      (v. cap_a.h) – se upisuje u akumulator; TIM_CNT_INI - timer counter initial value for Capture Units
        SACL    T2CNT           ; reset GPT2 counter register ... 0h se upisuje na adresu T2CNT = 7405h (videti F240_a.h) – radi se o adresi T2 Counter Registera (sadr`aj je broja~ GP Timera 2)
        SACL    T3CNT           ; reset GPT3 counter register ... 0h se upisuje na adresu T3CNT = 7409h (videti F240_a.h) – radi se o adresi T3 Counter Registera (sadr`aj je broja~ GP Timera 3)
;
        LACC    #TxCON_INI_CAP          ... TxCON_INI_CAP=01702h (videti cap_a.h)
konstanta koja slu`i za konfiguraciju T2CON i T3CON => CONT_UP,X/128, internal DSPCLK, no compare – videti str. 2-37 (Spru161a.pdf)
        SACL    T2CON           ; configure GPT2 to be used for CAP
...T2CON = 7408h (videti F240_a.h) – i radi se o adresi T2 Control Registera.
...Upisuje se 01702h na adresu T2CON = 7408h
        SACL    T3CON           ; configure GPT2 to be used for CAP
...T3CON = 740ch (videti F240_a.h) – i radi se o adresi T3 Control Registera.
...Upisuje se 01702h na adresu T3CON = 740ch
; configure Capture Control Register CAPCON
        LDP     #_capcon_value          ... _capcon_value=a001h (cap.map); setuje se DP=320
        LACC    _capcon_value           ... sadr`aj adrese _capcon_value=a001h se upisuje u akumulator. Sadr`aj adrese se setuje u komandnom prozoru aplikacije shodno izboru tajmera i doga|aja koje treba registrovati.
        LDP     #DP_EV                  ...DP_EV je = 0E8h (videti F240_a.h); Setovana je nova vrednost data page pointera (DP=0E8h=232dec).
        SACL    CAPCON                  ; set CAPCON register as programmed by user
... sadr`aj akumulatora (odnosno, adrese _capcon_value=a001h) se upisuje na adresu CAPCON=7420h (v. F240_a.h i videti Spru161a.pdf na str. 2-75); CAPCON je Capture Control Register.
; start timers
```

```
        SETBIT T2CON,SETB6   ;start GPT2          ... SETBIT je makro koji je definisan u Demos_a.h
... T2CON = 7408h     ; T2 Control Register (videti F240_a.h)
... SETB6 = 0040h     ; Bit Mask for 6 (videti F240_a.h)
... Su{tina je da se nad bitovima sadr`aja adrese T2CON sprovede operacija logi~ko ILI sa vredno{}u konstante
SETB6 i taj rezultat se sme{ta opet na adresu T2CON => T2CON[6]=1 => start GP Timer 2.
        SETBIT T3CON,SETB6   ;start GPT3          ...sli~no, => T3CON[6]=1 => start GPT3
loop:                                             ... labela
; read and store CAPFIFO status and control register
        LDP     #DP_EV                            ... DP_EV je = 0E8h (v. F240_a.h); => DP=0E8h=232dec
        LACC    CAPFIFO                 ; read register value ... sadr`aj registra CAPFIFO=7422h (videti
F240_a.h) se upisuje u akumulator. CAPFIFO sadr`i bitove statusa za svaki od ~etiri FIFO steka – videti
Spru161a.pdf, strana 2-77.
        LDP     #_capfifo_value  ;                ... _ capfifo_value =a002h (cap.map); setuje se DP=320
        SACL    _capfifo_value   ; store for transmission to user ... sadr`aj akumulatora se upisuje na adresu
_capfifo_value =a002h; prakti~no, na ovu adresu se upisuje sadr`aj registra CAPFIFO.
; call monitor
        CALL    MON240                            ... Programski broja~ (PC-program counter) se inkrementira i
stavlja na vrh steka. Sadr`aj adrese u programskoj memoriji MON240=0109h (videti cap_a.h). postaje sadr`aj
programskog broja~a. Poziva se program monitor.
; test FIFO read index
        LDP     #_read_fifo_no                    ... _ read_fifo_no=a003h (cap.map); setuje se DP=320
        LACC    _read_fifo_no                     ... sadr`aj adrese _read_fifo_no=a003h se upisuje u
akumulator. Sadr`aj adrese _read_fifo_no=a003h predstavlja broj prihvatnog registra (capture unit) ~iji stek korisnik
`eli da pro~ita. U komandnom prozoru aplikacije, korisnik klikom na dugme Read FIFO x – setuje sadr`aj registra na
adresi _read_fifo_no=a003h, tj x->(a003h)
        SUB     #1                                ... (ACC)-1->ACC
        BCND    read1,EQ                          ... Ako je uslov EQ ispunjen (tj. ako je sadr`aj akumulatora
jednak nuli) program ide na read1 (labela – dole) – prakti~no ako je (_read_fifo_no)=1.
        SUB     #1                                ... (ACC)-1->ACC
        BCND    read2,EQ                          ... Ako je uslov EQ ispunjen (tj. ako je sadr`aj akumulatora
jednak nuli) program ide na read2 (labela – dole) – prakti~no ako je (_read_fifo_no)=2.
        SUB     #1                                ... (ACC)-1->ACC
        BCND    read3,EQ                          ... Ako je uslov EQ ispunjen (tj. ako je sadr`aj akumulatora
jednak nuli) program ide na read3 (labela – dole) – prakti~no ako je (_read_fifo_no)=3.
        SUB     #1                                ... (ACC)-1->ACC
        BCND    read4,EQ                          ... Ako je uslov EQ ispunjen (tj. ako je sadr`aj akumulatora
jednak nuli) program ide na read3 (labela – dole) – prakti~no ako je (_read_fifo_no)=4.
no_read:                                          ... labela
        B       end_read                          ... Program se grana ka labeli end_routine (dole)
read1:
        LDP     #DP_EV                            ... DP_EV je = 0E8h (v. F240_a.h); => DP=0E8h=232dec
        LACC    FIFO1           ; read CAP1FIFO register value
... sadr`aj registra FIFO1=7423h se upisuje u akumulator; FIFO1 - Capture Channel 1 FIFO Top;
        LDP     #_fifo1_value    ;                ... _fifo1_value =a004h (videti cap.map); setuje se DP=320
        SACL    _fifo1_value     ; store for transmission to user
... sadr`aj akumulatora se upisuje na adresu _fifo1_value =a004h (videti cap.map) – na ovoj adresi se ~uva
posledenje-pro~itana vrednost sa vrha steka pripadaju}eg prihvatnog registra (capture unit).
        B       end_read                          ... Program se grana ka labeli end_routine (dole)
read2:
        LDP     #DP_EV                            ... DP_EV je = 0E8h (v. F240_a.h); => DP=0E8h=232dec
        LACC    FIFO2           ; read CAP2FIFO register value
... sadr`aj registra FIFO2=7424h se upisuje u akumulator; FIFO2 - Capture Channel 2 FIFO Top;
        LDP     #_fifo2_value    ;                ... _fifo2_value=a005h (videti cap.map); setuje se DP=320
        SACL    _fifo2_value     ; store for transmission to user
... sadr`aj akumulatora se upisuje na adresu _fifo2_value =a005h (videti cap.map) – na ovoj adresi se ~uva
posledenje-pro~itana vrednost sa vrha steka pripadaju}eg prihvatnog registra (capture unit).
        B       end_read                          ... Program se grana ka labeli end_routine (dole)
read3:
        LDP     #DP_EV                            ... DP_EV je = 0E8h (v. F240_a.h); => DP=0E8h=232dec
        LACC    FIFO3           ; read CAP3FIFO register value
... sadr`aj registra FIFO3=7425h se upisuje u akumulator; FIFO3 - Capture Channel 3 FIFO Top;
        LDP     #_fifo3_value    ;                ... _fifo3_value=a006h (videti cap.map); setuje se DP=320
        SACL    _fifo3_value     ; store for transmission to user
... sadr`aj akumulatora se upisuje na adresu _fifo3_value =a006h (v. cap.map) – na ovoj adresi se ~uva posledenje-
pro~itana vrednost sa vrha steka pripadaju}eg prihvatnog registra (capture unit).
```

```
        B       end_read                    ... Program se grana ka labeli end_routine (dole)
read4:
        LDP     #DP_EV                      ... DP_EV je = 0E8h (v. F240_a.h); => DP=0E8h=232dec
        LACC    FIFO4           ; read CAP4FIFO register value
... sadr`aj registra FIFO4=7426h se upisuje u akumulator; FIFO4 - Capture Channel 4 FIFO Top;
        LDP     #_fifo4_value   ;           ... _fifo4_value=a007h (videti cap.map); setuje se DP=320
        SACL    _fifo4_value    ; store for transmission to user
... sadr`aj akumulatora se upisuje na adresu _fifo4_value =a007h (videti cap.map) – na ovoj adresi se ~uva
posledenje-pro~itana vrednost sa vrha steka pripadaju}eg prihvatnog registra (capture unit).
        B       end_read                    ... Program se grana ka labeli end_routine (dole)
end_read:
        LDP     #_read_fifo_no              ... _ read_fifo_no=a003h (cap.map); setuje se DP=320
        SPLK    #0,_read_fifo_no ; test completed, reinitialize with 0...0->(_ read_fifo_no=a003h)
; test if demo ends (_stop =1)
        LDP     #_stop                      ... setuje se DP=320
        BIT     _stop,15                    ... posmatra se sadr`aj na adresi _stop. Specificirani bit
code=15, odnosi se na najni`i bit (LSB) na posmatranoj adresi. Instrukcija BIT kopira ovaj bit u TC bit status registra
ST1. TC (test/control flag bit) – je bit 11 status registra ST1 - ~uva rezultat operacije testiranja.
        BCND    loop,NTC                    ... Ako je uslov NTC ispunjen (tj. ako je TC=0) program ide na
loop (labela – gore); Odnosno, ako je klikom na Stop u prozoru aplikacije setovano: _stop[0]=1 => izlazak iz petlje.
;
        END_DEMO                            ... END_DEMO je makro, definisan u Demos_a.h; Kada bude
setovano _stop[0]=1, tj. da je TC=1, program prestaje da se izvr{ava.
```

3.3.5. Kolo za pravougaone impulse enkodera (QEP)

Ova aplikacija omogu}ava merenje i ~uvanje informacije sa inkrementalnog enkodera koriste}i *Event Manager QEP Circuit*. *Quadrature Encoder Pulse Circuit* omogu}ava dekodovanje i brojanje pravougaonih impulsa sa pinova CAP1/QEP1 i CAP2/QEP2. Ova aplikacija se startuje pritiskom na ikonu QEP u ***Processor Evaluation Control Panel*** prozoru. Na slede}oj slici je dat grafi~ki prikaz ove aplikacije:



Slika 16. Prozor aplikacije za *QEP*

Asemblerski program koji sadr`i kod ove aplikacije je ***qep_a.asm*** i mo`e se videti pritiskom na taster ***Show code***. Zaglavlje asemblerskog programa je ***qep_a.h*** koje sadr`i promenljive i funkcije za ovu aplikaciju. Startna adresa programa je **8000h** u eksternoj programskoj memoriji i zove se **_start**. Programske promenljive su u eksternoj memoriji za podatke i po~inju od adrese **a000h**, po slede}em rasporedu: **_stop** (=a000h), **_position** (=a001h).

Va`ne adrese (kori{}ene u aplikaciji):

_stop = a000h – sadr`i indeks za stopiranje programa (1 za kraj programa)
_position = a001h – sadr`i izmerenu poziciju enkodera, dobijenu kao poslednju vrednost sa *GPT3* broja~a.

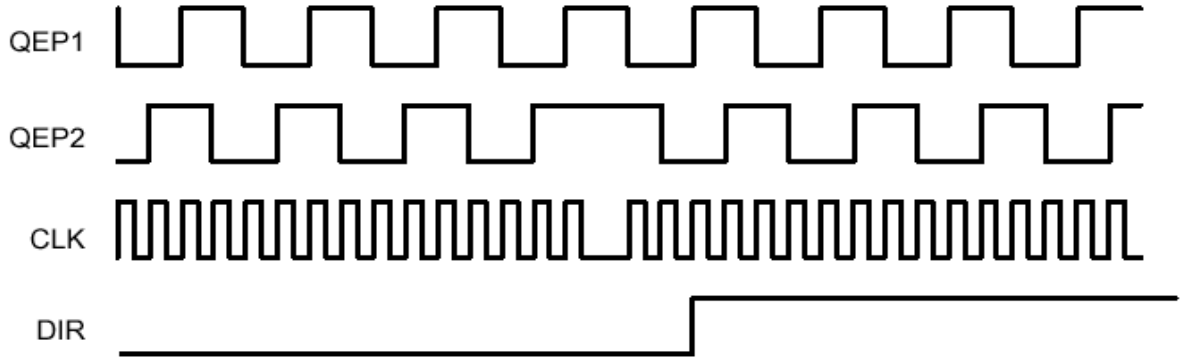
QEP (Quadrature Encoder Pulse) kolo

QEP kolo (ina~e sastavni deo Event Manager modula), kada je njegova funkcija dozvoljena, dekodira i broji kvadratne kodirane impulse koji dolaze sa pinova CAP1/QEP1 i CAP2/QEP2. QEP kolo mo`e biti kori{}eno kao veza sa opti~kim enkoderom da bi se dobila informacija o poziciji i brzini rotacije enkodera, odnosno rotacije odgovaraju}eg vratila.

Postoje dva QEP ulazna pina koja su zajedni~ka za QEP kolo i prihvatne jedinice 1 i 2 (Capture units 1 i 2). Podesnom konfiguracijom bitova CAPCON registra (CAPCON[14-13]=11) potrebno je omogu}iti funkciju QEP kola i onemogu}iti funkciju prihvatnih jedinica 1 i 2 (capture units 1 i 2).

Vremenska baza za QEP kolo mo`e biti obezbe|ena sa GP Timerom 2, 3 ili oba zajedno (ukoliko se `eli 32-bitni tajmer). Izbor se defini{e konfiguracijom bitova T2CON ili T3CON registra. Izabrani GP Timer ili 32-bitni tajmer mora biti setovan u **directional-up/down count mode** sa QEP kolom kao izvorom clocka.

Kvadratni kodirani impulsi su dve sekvence impulsa sa promenljivom frekvencijom i fiksnim faznim pomeranjem od ~etvrtine perioda (90^o). Kada su generisani opti~kim enkoderom na vratilu motora, smer rotacije motora mo`e biti odre|en prepoznavanjem koja od dve sekvence je vode}a sekvenca, i ugaona pozicija i brzina mogu biti odre|eni brojem i frekvencijom impulsa.



Logika odre|ivanja smera QEP kola u EV modulu (posebna logi~ka jedinica za to) odre|uje koja od sekvenci je vode}a. Generi{e se signal smera (DIR) kao ulazni signal (smera) u selektovani tajmer. Izabrani tajmer broji navi{e (up) ako je CAP1/QEP1 ulaz – vode}a sekvenca, ili nadole (down) ako je CAP2/QEP2 ulaz – vode}a sekvenca.

Obe ivice impulsa od strane obe sekvence sa ulaza - QEP kolo broji. Otuda je frekvencija generisanog clocka u GP tajmeru ~etiri puta ve}a od frekvencije svake ulazne sekvence impulsa. (ovaj generisani clock je povezan na ulazni clock selektovanog GP tajmera ili 32-bitnog tajmera).

Selektovani GP tajmer uvek po~inje brojanje od njegove teku}e vrednosi u broja~u (sadr`aj registra TxCNT). @eljena vrednost mo`e biti prethodno upisana u broja~ izabranog GP tajmera da omogu}i rad QEPa. Kada je QEP kolo selektovano kao izvor clocka, selektovani tajmer }e ignorisati TMRDIR i TMRCLK ulazne pinove.

Va`no je zapaziti da *directional-up/down counting mode* selektovanog GP tajmera sa QEP kolom kao clockom je razli~ito od normalnog (uobi~ajenog) *directional-up/down counting mode.* Kada selektovani tajmer za rad QEPa broji navi{e (up) ka vrednosti svog perioda (sadr`aj TxPER registra), GP tajmer se ne}e zaustaviti, ali umesto toga nastavi}e brojanje navi{e sve dok se ne promeni smer brojanja. Ako je po~etna vrednost broja~a GP tajmera ve}a od vrednosti perioda tajmera, tajmer }e brojati (navi{e) do FFFFh (ili FFFF FFFFh, ako je kori{}en 32-bitni tajmer) i okrenuti se u 0 (pre}i u 0) ako je odre|en smer brojanja nagore. Kada tajmer broji nadole ka 0, GP tajmer }e pre}i u FFFFh (ili FFFF FFFFh, ako je kori{}en 32-bitni tajmer) ako je odre|en smer brojanja nadole.

Da bi bio startovan rad QEP kola potrebno je:

1. Setovati sadr`aj compare, counter i period registra selektovanog GP tajmera
2. Konfigurisati bitove T2CON ili T3CON registra tako da GP Timer 2, 3, ili oba, bude u directional-up/down ili 32-bitnom modu sa QEP kolom kao izvorom clocka i da bude omogu}en rad selektovanog tajmera.
3. Konfigurisati bitove CAPCON registra tako da bude omogu}en rad QEP kola.

Analiza ponu|enog programskog re{enja aplikacije

Ponu|eni asemblerski program *cap_a.asm* je zasnovan na slede}em algoritmu:

1. Setuju se bitovi slede}ih registara:
   a. 0 se upisuje na adresu _stop=a000h. Kada najmla|i bit na ovoj adresi postane 1 (_stop[0]=1) tada program prestaje sa radom.
   b. 11 se upisuje u OPCRB[4-5] => pinovi 4, i 5 porta B postaju CAP1/QEP1 i CAP2/QEP2 pinovi. Videti na str. 11-15 u Spru161a.pdf kako se konfiguri{e Output Control Register B (OPCRB=7092h).
   c. 00 se upisuje u GPTCON[11-12] => GP Timer 3 ne}e startovati AD konverziju; GPTCON=7400h - General Purpose Timer Control Register (v. str. 2-38 u Spru161a.pdf).
   d. FFFFh se upisuje na adresu T3PER=740ch.
   e. 0 se upisuje na adresu T3CNT=7409h – sadr`aj broja~a GP Timera 3 se postavlja na 0.
   f. 1830h se upisuje na adresu T3CON=740ch - konfiguri{e se GP Timer 3.
   g. 1 se upisuje u bit T3CON[6] => start GP Timera 3
   h. 111 se upisuje u CAPCON[12-14] (na adresu CAPCON=7402h). Konfiguri{e se registar CAPCON (videti str. 2-75 u Spru161a.pdf). => omogu}en je rad QEP kola i prihvatne jedinice 3.
2. petlja:
   i. Sadr`aj registra T3CNT=7409h (tj. broja~ GPT3) se upisuje na adresu _position=a001h. (u komandnom prozoru aplikacije je obezbe|ena vizuelizacija sadr`aja ovog registra)
   j. Program Monitor se aktivira (u svakom ciklusu).
   k. Proverava se u svakom ciklusu da li je u najni`i bit adrese _stop=a000h upisana jedinica ({to se de{ava pritiskom na Stop u komandnom prozoru aplikacije). Ako jeste to je uslov za izlazak iz petlje. Ako ne, sledi povratak na po~etak petlje.
3. KRAJ

Detaljniji uvid se mo`e ste}i na osnovu ponu|enog asemblerskog programa *qep_a.asm*, koji sledi u nastavku (listing programa je u boji, a obja{njenja i primedbe nisu).

```
;*******************************************************************************
; File Name:   qep_a.asm
; Project:     MCK240
; Originator:  R.Giuclea
; Description: ASM file for QEP demo (encoder)
; Copyright © 1997 Technosoft
;*******************************************************************************
; Include Files
;-------------------------------------------------------------------------------
    .include    .. \F240_a.h
    .include    .. \demos_a.h
    .include    qep_a.h
;===============================================================
    .text
; ------------------------------------------------------------------------------
_start:             ... labela
  LDP  #_stop    ... _stop = a000h (pi{e u  qep.map). Starijih 9 bita a000h odre|uju data page pointer DP=320.
  SPLK  #0,_stop                                          ...setuje se 0 kao sadr`aj adrese _stop
; init encoder
  LDP  #DP_PF2; Peripheral File 2 Data Page Pointer... DP_PF2 je = 0E1h (v. F240_a.h); => DP=0E1h=225dec
  LACC  OPCRB                                  ... OPCRB=07092h    ; Output Control Register B – OPCRB.
... Sadr`aj registra OPCRB=7092h (videti F240_a.h) se upisuje u akumulator (ACC).
  OR #QEPENMSK     ;0030H, no b5. b4. of the IOPC C-port, but CAP/QEP 1&2
...QEPENMSK=0030h (videti qep_a.h)
  SACL  OPCRB      ;CAPx/QEPx (x=1,2) pins configured for QEP
... Sadr`aj ACC se upisuje na adresu OPCRB=7092h, i pri tom je sasvim sigurno OPCRB[4-5]=11, ~ime je odlu~no da pinovi 4 i 5 porta B igraju ulogu CAP1/QEP1 i CAP2/QEP2 pinova (videti tabelu 11-6 u Spru161a.pdf);
  LDP  #DP_EV     ;Event Manager Data Page Pointer... DP_EV je = 0E8h (v. F240_a.h); => DP=0E8h=232dec
```

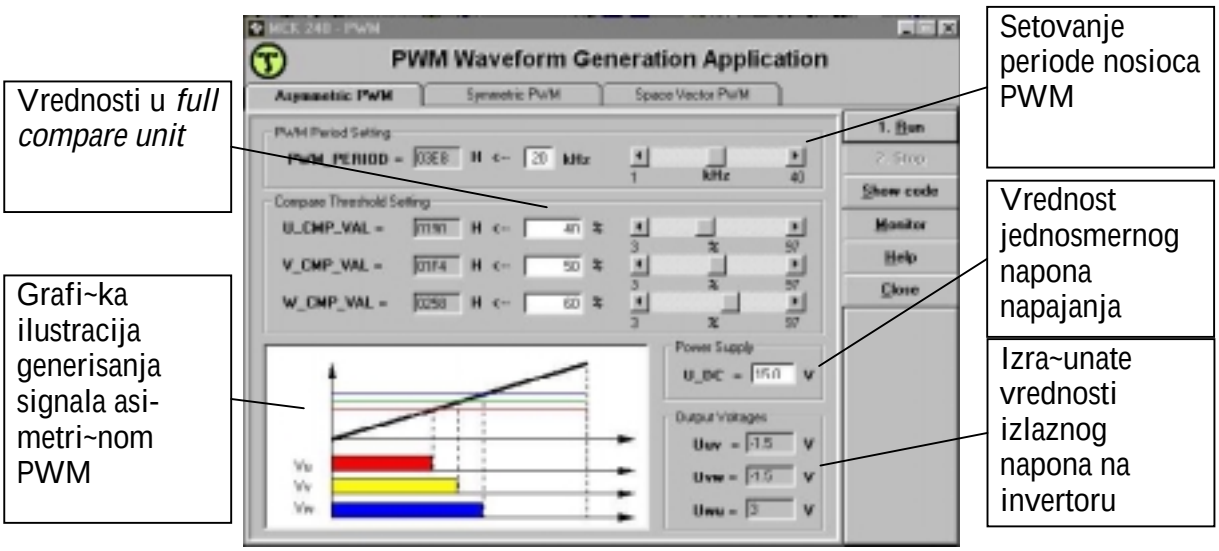
```
  LACC  GPTCON... GPTCON (General Purpose Timer Control Register)=7400h (adresa je definisana u F240_a.h)
... sadr`aj sa adrese 07400h (GPTCON) se upisuje u akumulator (ACC);
  AND   #AND_T3TOADC_  ; AND mask for DISABLING ADC start on GPT3
... AND_T3TOADC=0E7FFh (videti Demos_a.h).
... operacija logi~ko I se sprovodi nad bitovima sadr`aja akumulatora (tj. adrese 07400h (GPTCON)) i 0E7FFh, i rezultat opet upisuje u akumulator => resetuje se bit 11 i 12 akumulatora.
  SACL  GPTCON      ; configure GPTCON not to start ADC on GPT3 Event
... sadr`aj akumulatora se upisuje na adresu 07400h (GPTCON) => GP Timer 3 ne}e startovati AD konverziju
  LACC  #0FFFFh     ; !!! Ovo nema mnogo veze, radi isto i sa 7FFF... u akumulator se upisuje konstanta 0FFFFh
   SACL  T3PER    ; initialize GPT3 timer period                    ... u F240_a.h je definisano da je T3PER = 740bh
... na ovu adresu se sme{ta sadr`aj akumulatora  - 0FFFFh, i defini{e se period GP Timera 3.
  LACC  #T3CNT_INI_QEP   ;this is zero. OVO IMA veze, inicijalno stanje !
...T3CNT_INI_QEP=0h (videti qep_a.h)
  SACL  T3CNT    ; reset GPT3 counter register
... 0h se upisuje na adresu T3CNT = 7409h (videti F240_a.h) – radi se o adresi T3 Counter Registera (sadr`aj je broja~ GP Timera 3)
  LACC  #T3CON_INI_QEP ;1830h !!!!  Emulation suspend is active !!! Ali to
                ;nije bitno, radi isto i sa EMU passive, load #9830H
                ;DIRECTIONAL UP/DOWN COUNTING, Timer disable(b6),
                ;QEP circuit enable (b5b4)
...T3CON_INI_QEP=01830h         (videti qep_a.h) i ova vrednost se upisuje u akumulator.
  SACL  T3CON        ;configure GPT3 to be used for QEP
...T3CON = 740ch (videti F240_a.h) – i radi se o adresi T3 Control Registera.
...Upisuje se 01830h na adresu T3CON = 740ch – videti str. 2-36 (Spru161a.pdf) => GP Timer 3 broja}e u Directional-Up/Down Count Mode, QEP kolo je aktivirano, i GPT3 jo{ nije startovan
; start encoder
  SETBIT   T3CON,SETB6 ;start GPT3 to count QEP circuit pulses
... SETBIT je makro koji je definisan u Demos_a.h
... T3CON = 740ch     ; T3 Control Register (videti F240_a.h)
... SETB6 = 0040h     ; Bit Mask for 6 (videti F240_a.h)
... Su{tina je da se nad bitovima sadr`aja adrese T3CON sprovede operacija logi~ko ILI sa vredno{}u konstante SETB6 i taj rezultat se sme{ta opet na adresu T3CON => T3CON[6]=1 => start GP Timer 3.
  LACC  CAPCON
... CAPCON je Capture Control Register – CAPCON=7420h – pogledati str. 2-75 u Spru161a.pdf;
... Sadr`aj ovog registra se upisuje u akumulator.
  OR #CAPCON_QEP_EN    ;0E000 Hex SET b15, 14, 13, NO reset, QEP Enable
                ;ALL other bits b12.-b0 irrelevant completely
...CAPCON_QEP_EN=0E000h (videti qep_a.h)        ; enable QEP in CAPCON register
  SACL  CAPCON      ; enable QEP in CAPCON register
... Sadr`aj akumulatora se upisuje na adresu CAPCON=7420h (v. F240_a.h i Spru161a.pdf na str. 2-75); Konkretno, 111 se upisuje u CAPCON[12-14], ~ime je omogu}en rad QEP kola i prihvatne jedinice 3 (capture unit 3).
loop:
; read and store encoder
  LDP   #DP_EV                    ... DP_EV je = 0E8h (videti F240_a.h); => (DP=0E8h=232dec).
  LACC  T3CNT    ; read GPT3 current counter value
... sadr`aj adrese T3CNT = 7409h (videti F240_a.h), na kojoj je vrednost broja~a GPT3, se upisuje u akumulator
  LDP   #_position  ; ..._position  = a001h (videti qep.map); Setuje se nova vrednost data page pointera DP=320
  SACL  _position   ; store position
... Sadr`aj akumulatora (odnosno vrednosti broja~a GP Timera 2) se upisuje na adresu _position  = a001h
; call monitor
  CALL  MON240                    ... Programski broja~ (PC-program counter) se inkrementira i stavlja na vrh steka. Sadr`aj adrese u programskoj memoriji MON240=0109h (videti cap_a.h). postaje sadr`aj programskog broja~a. Poziva se program monitor.
; test if demo ends (_stop =1)
  LDP   #_stop                    ... setuje se DP=320
  BIT   _stop,15                  ... posmatra se sadr`aj na adresi _stop. Specificirani bit code=15, odnosi se na najni`i bit (LSB) na posmatranoj adresi. Instrukcija BIT kopira ovaj bit u TC bit status registra ST1. TC (test/control flag bit) – je bit 11 status registra ST1 - ~uva rezultat operacije testiranja.
  BCND  loop,NTC                  ... Ako je uslov NTC ispunjen (tj. ako je TC=0) program ide na loop (labela – gore); Odnosno, ako je klikom na Stop u prozoru aplikacije setovano: _stop[0]=1 => izlazak iz petlje.
  END_DEMO                        ... END_DEMO je makro, definisan u Demos_a.h; Kada bude setovano _stop[0]=1, tj. da je TC=1, program prestaje da se izvr{ava.
```

3.3.6. Generisanje *PWM* signala

PWM (pulsewidth-modulated) signal predstavlja niz (sekvencu) impulsa promenljive {irine. Po jedan impuls razli~ite {irine je sadr`an u fiksnom vremenskom intervalu koji se ponavlja i zove se PWM period ili *period PWM nosioca*. Inverzna vrednost PWM perioda se zove *frekvencija PWM nosioca*. [irine PWM impulsa su odere|ene ili modulisane od impusla do impulsa saglasno drugoj sekvenci `eljenih vrednosti, odnosno, moduli{u}em signalu. Frekvencija moduli{u}eg signala je tipi~no mnogo manja od frekvencije PWM nosioca.
Za generisanje PWM signala, potreban je odgovaraju}i tajmer da ponavlja brojanje unutar perioda koji je isti kao i period PWM nosioca. *Compare* registar se koristi da sadr`i moduli{u}e vrednosti. Sadr`aj *compare* registra se konstantno upore}uje sa broja~em tajmera. Kada se vrednosti poklope, de{ava se promena (od viskog ka niskom, od niskog ka visokom nivou signala) na pridru`enom izlazu. Kada se vrednosti poklope po drugi put, ili do|e do dostizanja vrednosti perioda tajmera de{ava se druga promena (u suprotnom smeru od prethodne) signala na pridru`enom izlaznom pinu. Na ovaj na~in, izlazni impuls je generisan - ~ija du`ina (signala on /ili off) je proporcionalna vrednosti u *compare* registru. Ovaj proces se ponavlja za svaki period tajmera sa razli~itim (moduli{u}im) vrednostima u *compare* registru. Kao rezultat, generisan je PWM signal na pridru`enom izlazu.
^esta je potreba da se PWM signalima pobu|uju dva ure|aja vezana u seriju, i tada se zahteva da ne do|e do preklapanja ta dva PWM izlazna signala (tj. zahteva se da PWM signali budu zaka{njeni jedan u odnosu na drugi). Mrtvo vreme (**dead-band – mrtva zona**) je ~esto uba~eno izme|u isklju~enja jednog i uklju~enja drugog tranzistora. Ovo ka{njenje dozvoljava kompletno isklju~enje jednog tranzistora pre uklju~enja drugog tranzistora. Zahtevano vreme ka{njenja zavisi od karakteristika uklju~ivanja i isklju~ivanja tranzistora i uop{te zahteva same aplikacije.

3.3.6.1. Asimetri~na *PWM*

Ova aplikacija omogu}ava generisanje trofaznog napona pomo}u asimetri~ne *PWM* modulacije. *General Purpose Timer* 1 i tri *Full Compare Units* iz *Event manager* su programirani da daju 6 *PWM* signala. Kao rezultat dobijaju se trofazni naponi na izlaznom invertorskom mostu. Ova aplikacija se startuje pritiskom na ikonu ***PWM*** u ***Processor Evaluation Control Panel*** prozoru. Na slede}oj slici je dat grafi~ki prikaz ove aplikacije:



Slika 17. Prozor aplikacije za asimetri~nu *PWM*

Idealna vrednost izlaznog napona invertora se dobija po izrazu:

$$U_{XY} = U_{DC} \cdot \frac{\text{X\_CMP\_VAL} - \text{Y\_CMP\_VAL}}{100}$$

gde X i Y mogu da uzmu vrednosti U, V ili W. Asemblerski program koji sadr`i kod ove aplikacije je ***pwma_a.asm*** i mo`e se videti pritiskom na taster ***Show code***. Zaglavlje asemblerskog programa je ***pwma_a.h*** koje sadr`i promenljive i funkcije za ovu aplikaciju. Startna adresa programa je **8000h** u eksternoj programskoj memoriji i zove se **_start**. Programske promenljive su u eksternoj memoriji za podatke i po~inju od adrese **a000h**, po slede}em rasporedu:

_stop = a000h adresa sadr`i indeks za stopiranje programa (1 za kraj programa)
_pwm_period = a001h adresa sadr`i vrednost periode *PWM* nosioca koji se upisuje u *Timer 1 Period Register*
_u_cmp_value =a002h adresa sadr`i vrednost koja se sme{ta u *Full Compare Unit 1 Register*
_ v_cmp_value=a003h adresa sadr`i vrednost koja se sme{ta u *Full Compare Unit 2 Register*
_w_cmp_value=a004h adresa sadr`i vrednost koja se sme{ta u *Full Compare Unit 3 Register*

Za pra}enje ove aplikacije, korisno je razumeti *Full Compare Units* (Spru161a.pdf, strana 2-41). Evo kra}eg izvoda o tome, kao i o generisanju asimetri~nog PWM signala:

*Jedinice za pore|enje (**Compare Units**)*

U EV modulu postoje:

- 3 jednostavne jedinice za pore|enje (**Simple Compare Units 1, 2, i 3**) – savaka ima po 1 pridru`eni compare/PWM izlaz, dok vremensku bazu obezbe|uje GP Timer 1 ili 2 (videti Spru161a.pdf na strani 2-41). Njihov rad od operacije pore|enja kod GP tajmera se razlikuje samo u slede}em: 1) Vremensku bazu za *simple compare units* mogu dati GPT1 ili GPT2. 2) Bitovi COMCON registra kontroli{u funkciju ovih jedinica (dozvola rada, dozvola aktivnosti izlaza, selekcija vremenske baze, itd.) 3) Bitovi SACTR registra (Simple Action Control Register) odre|uju pona{anje izlaza.
- 3 potpune jedinice za pore|enje (**Full Compare Units 1, 2, i 3**) – svaka ima po 2 pridru-`ena compare/PWM izlaza, dok vremensku bazu obezbe|uje GP Timer 1.

Tri potpune jedinice za pore|enje (Full Compare Units 1, 2, i 3) uklju~uju:

- tri 16-bitna *compare* registra (CMPRx, x=1, 2, 3) sa pridru`nim registrima u senci;
- 6 compare/PWM izlaznih pinova, PWMy/CMPy, y=1, 2, 3, 4, 5, 6.
- 16-bitni COMCON registar (Compare Control Register)
- 16-bitni ACTR registar (Action Control Register) sa pridru`nim registrom u senci
- jedinicu za kontrolnu i interrupt logiku

Ulazi **(Full Compare Units 1, 2, i 3)**:

- signali iz kontrolnih registara (kontrolni signali)
- broja~ GPT1, zatim flag signal za *underflow* i *period match*
- reset

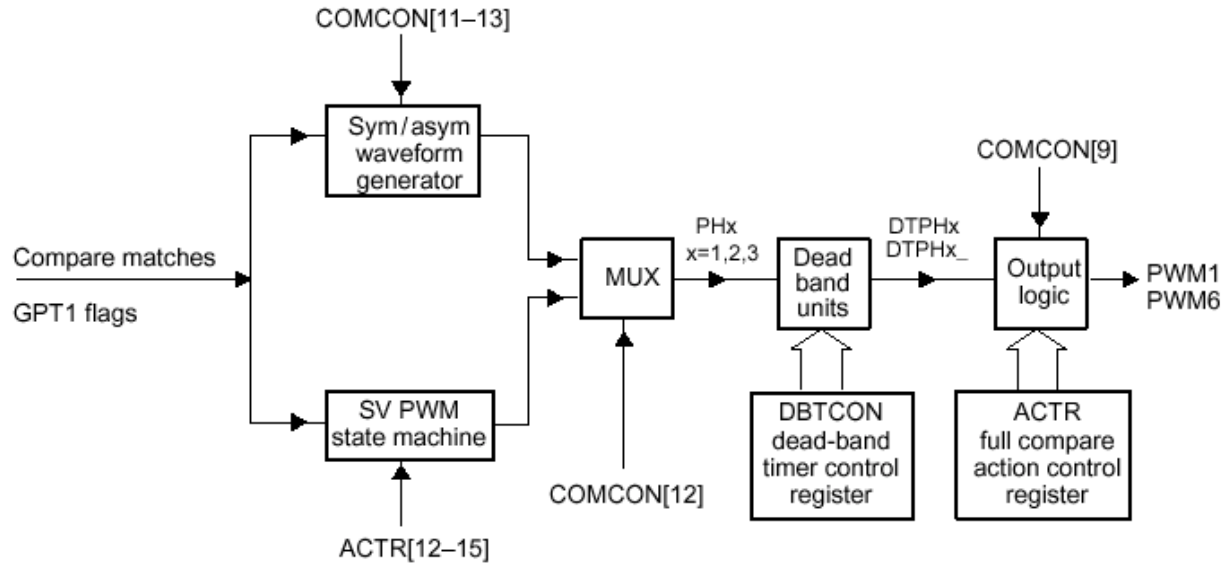
Izlaz **(Full Compare Units 1, 2, i 3)** je *compare match* signal (nastaje kao rezultat pore|enja vrednosti broja~a tajmera GPT1 i sadr`aja odgovaraju}eg *compare* registra CMPRx). Ako je operacija pore|enja dozvoljena, compare match signal setuje interrupt flag i prouzrokuje promenu stanja signala na 2 izlazna pina (svakoj od full compare unit su pridru`ena po dva izlazna pina).

Na~in rada **(Full Compare Units 1, 2, i 3)** je odre|en bitovima registra COMCON, i to:

- da li je operacija pore|enja dozvoljena
- da li su *full compare outputi* dozvoljeni (tj. omogu}eno njihovo kori{}enje)
- uslov koji treba da bude ispunjen da bi sadr`aj compare registra CMPRx bio a`uriran sa sadr`ajem njihovih registara "u senci"
- uslov koji treba da bude ispunjen da bi sadr`aj action control registra ACTR bio a`uriran sa sadr`ajem njegovog registara "u senci"
- da li je space vector PWM moda dozvoljen
- da li je svaka i koja od *full compare unit* u compare ili PWM modu:

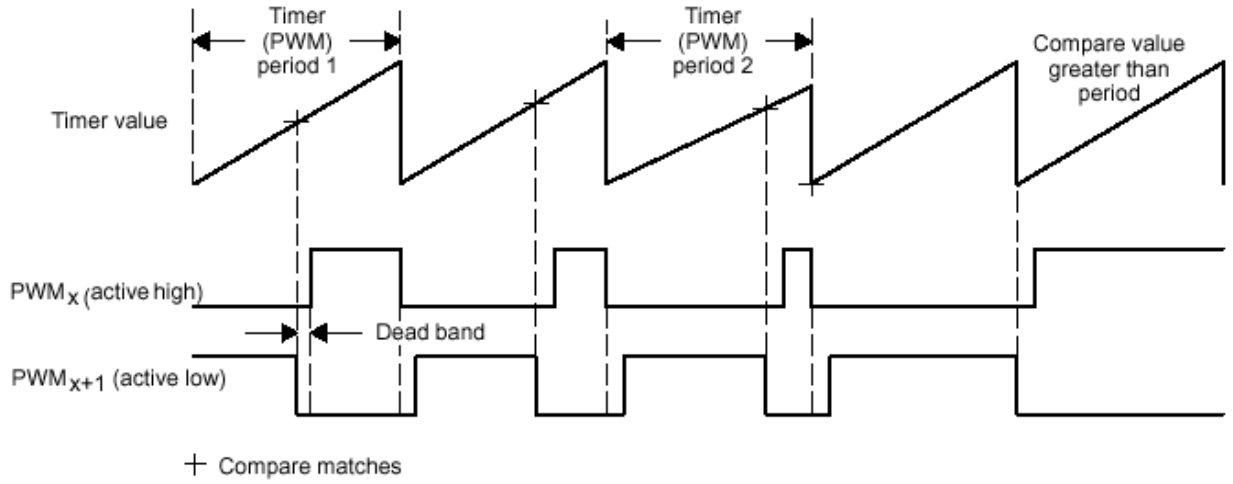
- u *compare* modu – broja~ GPT1 se kontinulano poredi sa sadr`ajem compare registra CMPRx, i kada se poklapanje dogodi promena }e se desiti na oba izlaza, saglasno bitovima ACTR registra (da li }e se dogoditi setovanje signala na low ili high nivo, ili promena teku}eg nivoa izlaznog signala, itd.).
- u PWM modu – rad compare unit je sli~an istom koji se mo`e izvesti isklju~ivo preko GP tajmera (pro~itati od strane 2-30 do 2-35 u Spru161a.pdf). Bitna razlika je u tome {to je njegov rad upravljan kori{}enjem drugih registara. Drugo, PWM kolo je pridru`eno – full compare units – (~iji je funkcionalan blok dijagram prikazan na slici dole) i zajedno omogu}avaju generisanje 6 PWM izlaznih kanala sa programabilnom mrtvom zonom i izlaznim polaritetima.

Uop{te, PWM kola su projektovana da minimiziraju tro{kove i intervencije korisnika oko generisanja {irinski modulisanih impulsa. Rad **full compare units** i pridru`enog **PWM kola** zahteva pode{avanje: **1.** registra T1PR gde se sme{ta period GPT1; **2.** ACTR registra (strana 2-48 u Spru161a.pdf); **3.** DBTCON registra (Dead-Band Timer Control Register – strana 2-54 u Spru161a.pdf); **4.** Inicijalizaciju sadr`aja CMPRx registra; **5.** COMCON registra (strana 2-45 u Spru161a.pdf); i **6.** T1CON registra; Generisanje PWM signala na ovaj na~in ima prednosti u odnosu na generisanje PWM signala kori{}enjem isklju~ivo GP tajmera (obrazlo`eno u Spru161a.pdf na strani 2-53).



Sl. 18. *Blok dijagram PWM kola*

Asimetri~na PWM



Sl. 19. *Asimetri~na PWM*

Da bi generisali asimetri~ni PWM signal, GP Timer 1 se podesi na ***continuous up*** mod brojanja. U registar T1PR - gde se upisuje perioda GP Timera 1, upisuje se zapravo `eljeni period PWM nosioca. Registar COMCON se konfiguri{e da omogu}i operaciju pore|enja i da bude dozvoljen

izlaz PWM signala na pripadaju}im izlaznim pinovima. Ako je mrtva zona (dead-band) selektovana, `eljena vrednost koja odgovara "{irini" mrtve zone treba da bude upisana u vi{em bajtu DBTCON registra, kao period za 8-bitni dead-band tajmer. Jedna vrednost {irine mrtve zone se koristi za sve PWM izlazne kanale.
Podesnom konfiguracijom ACTR registra, uobi~ajeni PWM signal mo`e biti generisan na jednom izlazu koji je pridru`en *full compare unit*, dok je drugi izlaz zadr`an na low ili high nivou signala, ili nivou signala na po~etku, sredini ili kraju PWM perioda. Takva softverska kontrola PWM izlaza u nekim aplikacijama mo`e biti posebno korisna – i savakako doprinosi fleksibilnosti.

Analiza ponu|enog programskog re{enja aplikacije

Ponu|eni asemblerski program *pwma_a.asm* je zasnovan na slede}em algoritmu:

1. Setuju se bitovi slede}ih registara:
    a. 0 se upisuje na adresu _stop=a000h. Kada najmla|i bit na ovoj adresi postane 1 (_stop[0]=1) tada program prestaje sa radom.
    b. 00 se upisuje u GPTCON[8-7] => GP Timer 1 ne}e startovati AD konverziju; GPTCON=7400h - General Purpose Timer Control Register (v. str. 2-38 u Spru161a.pdf).
    c. (_pwm_period=a001h)->(T1PER=7403h). Setuje se period GP Timera 1 tako {to se na adresu T1PER=7403h upisuje sadr`aj koji je korisnik izabrao u komandnom prozoru aplikacije (privremeno sme{ten na adresu _pwm_period=a001h) – kao periodu nosioca PWM.
    d. 0 se upisuje na adresu T1CNT=7401h – broja~ GP Timera 1 se postavlja na 0.
    e. 0 se upisuje na adresu DBTCON=7415h=>mrtva zona nije selektovana, tj. nula je.
    f. const. ACTIVE_HI_LO=0666h se upisuje na adresu ACTR=7413h – sadr`aj registra ACTR defini{e akciju na svakom od 6 *full compare output* pinova. U ovom slu~aju, ako je setovana PWM f-ja pomenutih pinova ima}emo da je PWM1,3,5/CMP1,3,5 – Active high, a PWM2,4,6/CMP2,4,6 - Active low
    g. (_u_cmp_val=a002h)->(CMPR1=7417h) – iz komandnog prozora aplikacije, setuje se sadr`aj **compare registra Full Compare Unit 1,** i to kao % perioda GPT1
    h. (_v_cmp_val=a003h)->(CMPR2=7418h) – iz komandnog prozora aplikacije, setuje se sadr`aj ***compare*** **registra Full Compare Unit 2,** i to kao % perioda GPT1
    i. (_w_cmp_val=a004h)->(CMPR3=7419h) – iz komandnog prozora aplikacije, setuje se sadr`aj ***compare*** **registra Full Compare Unit 3,** i to kao % perioda GPT1
    j. const. **TIM_COUNT_UP=1000h**->(T1CON=7404h) - konfiguri{e se GP Timer 1. (Spru161a.pdf str. 2-37.) – GPT1 }e brojati na CPU clock frekvenciji u Continuous Up modu (**bitno** za asimetri~ni PWM).
    k. FCU_MODE_PWM =7h ->(COMCON=7411h) => selektovan je PWM mod za **Full Compare Unit 1, 2 i 3.**
    l. 1 se upisuje u bit COMCON[15] (=1) => operacija pore|enja je dozvoljena
    m. 1 se upisuje u bit COMCON[9] (=1) => omogu}eno kori{}enje pridru`enih izlaznih pinova (tj. sada mogu da prenesu setovano stanje pridru`enog signala)
    n. 1 se upisuje u bit T1CON[6] (=1)=> start GP Timera 1
2. petlja:
    a. (_u_cmp_val=a002h)->(CMPR1=7417h)
    b. (_v_cmp_val=a003h)->(CMPR2=7418h)
    c. (_w_cmp_val=a004h)->(CMPR3=7419h)
    d. Program Monitor se aktivira (u svakom ciklusu).
    e. Proverava se u svakom ciklusu da li je u najni`i bit adrese _stop=a000h upisana jedinica ({to se de{ava pritiskom na Stop u komandnom prozoru aplikacije). Ako jeste to je uslov za izlazak iz petlje. Ako ne, sledi povratak na po~etak petlje.
3. KRAJ

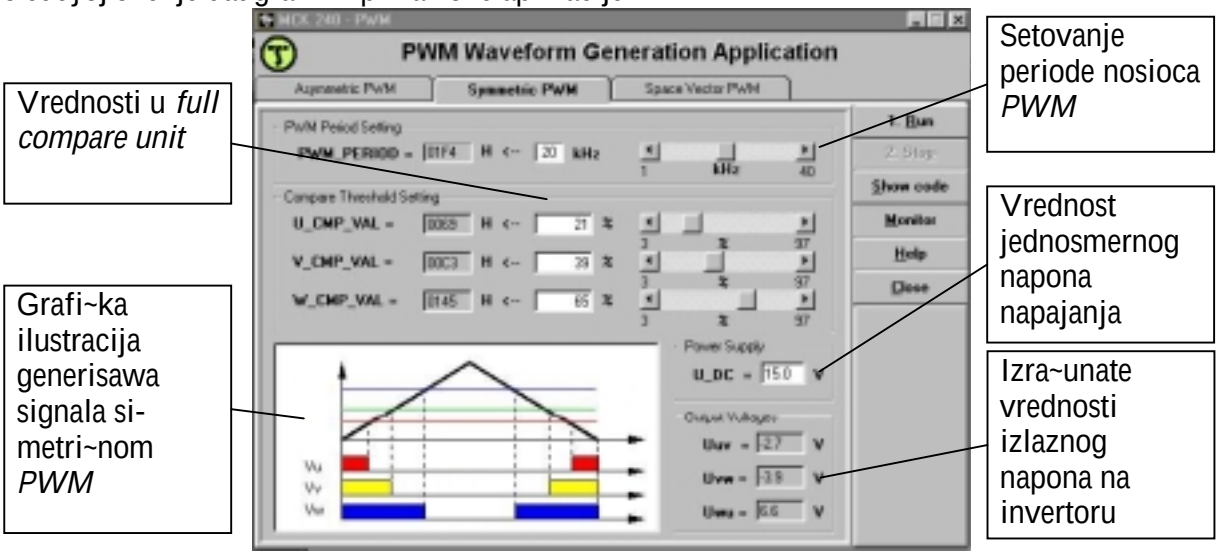
Detaljniji uvid se mo`e ste}i na osnovu ponu|enog asemblerskog programa *pwma_a.asm*, koji sledi u nastavku (listing programa je u boji, a obja{njenja i primedbe nisu).

```
;*******************************************************************************
; File Name:            pwma_a.asm
; Project:      MCK240
; Originator:   R.Giuclea
; Description:          ASM file for asymmetric PWM demo
; Copyright © 1997 Technosoft
;*******************************************************************************
; Include Files
;-------------------------------------------------------------------------
                        .include        ..\F240_a.h
                        .include        ..\demos_a.h
                        .include        pwma_a.h
;=============================================================================
                        .text
; ----------------------------------------------------------------------------
_start:
        LDP     #_stop                  ... _stop = a000h (pi{e u  pwma.map). Starijih 9 bita a000h odre|uju vrednost data page pointera DP=320.
        SPLK    #0,_stop                ... setuje se 0 kao sadr`aj adrese _stop, i to je sve tako dok se klikom na Stop u komandnom prozoru aplikacije ne setuje 1 kao sadr`aj adrese _stop ({to je uslov za izlaz iz programa).
; timer will not start ADC automatically
        LDP     #DP_EV                  ; Event Manager Data Page Pointer
... DP_EV je = 0E8h (videti F240_a.h); Setovana je nova vrednost data page pointera (DP=0E8h=232dec).
        LACC    GPTCON                  ... Upisuje se sadr`aj adrese 07400h (GPTCON) u akumulator (ACC);
        AND     #AND_T1TOADC_           ; AND mask for DISABLING ADC start on GPT1
... u Demos_a.h definisana je konstanta AND_T1TOADC_      = 0FE7Fh
... operacija logi~ko I se sprovodi nad bitovima sadr`aja adrese 07400h (GPTCON) i 0FE7Fh – resetuje se bit 7 i 8 akumulatora.
        SACL    GPTCON                  ; configure GPTCON not to start ADC on GPT1 Event
... Sadr`aj akumulatora se upisuje na adresu 07400h (GPTCON) => GPTCON[8-7]=00 => GP Timer 1 ne}e startovati AD konverziju (videti Spru161a.pdf str. 2-38)
; load and init timer and PWM registers:
;
; load GPT1 timer period with the PWM periode
        LDP     #_pwm_period            ... _pwm_period=a001h (videti pwma.map), setuje se DP=320
        LACC    _pwm_period             ... upisuje se sadr`aj _pwm_period=a001h u akumulator; Sadr`aj _pwm_period=a001h setuje korisnik u komandnom prozoru aplikacije, i to je perioda nosioca PWM .
        LDP     #DP_EV                  ... Nova vrednost data page pointera: DP=0E8h=232dec.
        SACL    T1PER
... sadr`aj akumulatora se upisuje na adresu T1PER=7403h (videti F240_a.h), i tako defini{e period tajmera GPT1. Prakti~no: (_pwm_period=a001h)->( T1PER=7403h).
; set GPT1 counter initial value
        LACC    #0                      ... konstanta 0h se upisuje u akumulator
        SACL    T1CNT
... sadr`aj akumulatora se upisuje na adresu T1CNT=7401h (videti F240_a.h) – i radi se o adresi T1 Counter Registera (sadr`aj je broja~ GP Timera 1). Prakti~no: 0->( T1CNT=7401h)
; set deadbeat parameters
        LACC    #0                      ... konstanta 0h se upisuje u akumulator
        SACL    DBTCON
... sadr`aj akumulatora se upisuje na adresu DBTCON=7415h (videti F240_a.h) – i radi se o adresi Dead-Band Timer Control Registera (strana 2-54, Spru161a.pdf). Prakti~no: 0->( DBTCON=7415h)
; set action control register
        LACC    #ACTIVE_HI_LO           ... konstanta ACTIVE_HI_LO=0666h se upisuje u akumulator
        SACL    ACTR
... sadr`aj akumulatora se upisuje na adresu ACTR=7413h (videti F240_a.h) – i radi se o adresi Full Compare Action Control Registera (strana 2-48, Spru161a.pdf). Prakti~no: 0666h->( ACTR=7413h)
; set value for full compare unit 1 register
        LDP     #_u_cmp_val             ... Setuje se: DP=320. (_u_cmp_val=a002h (pwma.map))
        LACC    _u_cmp_val              ... upisuje se sadr`aj _u_cmp_val=a002h u akumulator; Sadr`aj _u_cmp_val=a002h setuje korisnik u komandnom prozoru aplikacije.
        LDP     #DP_EV                  ... Nova vrednost data page pointera: DP=0E8h=232dec.
        SACL    CMPR1
... sadr`aj akumulatora se upisuje na adresu CMPR1=7417h (videti F240_a.h) – Compare Channel 1 Threshold (CMPR1). Prakti~no: (_u_cmp_val=a002h)->( CMPR1=7417h)
; set value for full compare unit 2 register
```

```
        LDP     #_v_cmp_val             ... Setuje se: DP=320. (_v_cmp_val=a003h (pwma.map))
        LACC    _v_cmp_val              ... upisuje se sadr`aj _v_cmp_val=a003h u akumulator; Sadr`aj
_v_cmp_val=a003h setuje korisnik u komandnom prozoru aplikacije.
        LDP     #DP_EV                  ... Nova vrednost data page pointera: DP=0E8h=232dec.
        SACL    CMPR2
... sadr`aj akumulatora se upisuje na adresu CMPR2=7418h (videti F240_a.h) – Compare Channel 2 Threshold
(CMPR2). Prakti~no: (_v_cmp_val=a003h)->( CMPR2=7418h)
; set value for full compare unit 3 register
        LDP     #_w_cmp_val             ... Setuje se: DP=320. (_w_cmp_val=a004h (pwma.map))
        LACC    _w_cmp_val              ... upisuje se sadr`aj _w_cmp_val=a004h u akumulator; Sadr`aj
_w_cmp_val=a004h setuje korisnik u komandnom prozoru aplikacije.
        LDP     #DP_EV                  ... Nova vrednost data page pointera: DP=0E8h=232dec.
        SACL    CMPR3
... sadr`aj akumulatora se upisuje na adresu CMPR3=7419h (videti F240_a.h) – Compare Channel 3 Threshold
(CMPR3). Prakti~no: (_w_cmp_val=a004h)->( CMPR3=7419h)
; set GPT1 control register
        LACC    #TIM_CONT_UP            ... upisuje se konstanta TIM_COUNT_UP=1000h u akumulator;
        SACL    T1CON
... sadr`aj akumulatora se upisuje na adresu T1CON=7404h (videti F240_a.h) – kontrolni registar za GPT1.
Prakti~no: TIM_COUNT_UP=1000h->( T1CON=7404h)
; set FCU control register - full compare unit works in sym/asym PWM mode
        LACC    #FCU_MODE_PWM           ... upisuje se konstanta FCU_MODE_PWM =7h u akumulator;
        SACL    COMCON
... sadr`aj akumulatora se upisuje na adresu COMCON=7411h (videti F240_a.h) – Compare Control Register (v.
stranu 2-45, Spru161a.pdf). Prakti~no: FCU_MODE_PWM =7h ->( COMCON=7411h)
; start PWM generation
        SETBIT COMCON,SETB15        ;enable FCU compare operation
... SETBIT je makro koji je definisan u Demos_a.h
... => COMCON[15]=1 (videti Spru161a.pdf str. 2-45) => funkcija pore|enja dozvoljena
        SETBIT COMCON,SETB9         ;enable FCU output pins
... => COMCON[9]=1 (videti Spru161a.pdf str. 2-46) => omogu}eno kori{}enje compare output pinova
        SETBIT T1CON,SETB6  ;start GPT1 counter
... => T1CON[6]=1 (videti Spru161a.pdf str. 2-36) => startuje se GP Timer 1
;
loop:                                   ... labela (ozna~ava po~etak petlje)
; set value for full compare unit 1 register    ... Prakti~no: (_u_cmp_val=a002h)->( CMPR1=7417h):
        LDP     #_u_cmp_val             ... Setuje se: DP=320. (_u_cmp_val=a002h (pwma.map))
        LACC    _u_cmp_val              ... upisuje se sadr`aj _u_cmp_val=a002h u akumulator;
        LDP     #DP_EV                  ... Nova vrednost data page pointera: DP=0E8h=232dec.
        SACL    CMPR1                   ... sadr`aj akumulatora se upisuje na adresu CMPR1=7417h
; set value for full compare unit 2 register    ... Prakti~no: (_v_cmp_val=a003h)->( CMPR2=7418h):
        LDP     #_v_cmp_val             ... Setuje se: DP=320. (_v_cmp_val=a003h (pwma.map))
        LACC    _v_cmp_val              ... upisuje se sadr`aj _v_cmp_val=a003h u akumulator.
        LDP     #DP_EV                  ... Nova vrednost data page pointera: DP=0E8h=232dec.
        SACL    CMPR2                   ... sadr`aj akumulatora se upisuje na adresu CMPR2=7418h
; set value for full compare unit 3 register    ... Prakti~no: (_w_cmp_val=a004h)->( CMPR3=7419h)
        LDP     #_w_cmp_val             ... Setuje se: DP=320. (_w_cmp_val=a004h (pwma.map))
        LACC    _w_cmp_val              ... upisuje se sadr`aj _w_cmp_val=a004h u akumulator;
        LDP     #DP_EV                  ... Nova vrednost data page pointera: DP=0E8h=232dec.
        SACL    CMPR3                   ... sadr`aj akumulatora se upisuje na adresu CMPR3=7419h
; call monitor
        CALL    MON240                  ... Programski broja~ (PC-program counter) se inkrementira i stavlja na
vrh steka. Sadr`aj adrese u programskoj memoriji MON240=0109h (videti cap_a.h). postaje sadr`aj programskog
broja~a. Poziva se program monitor.
; test if demo ends (_stop =1)
        LDP     #_stop                  ... setuje se DP=320
        BIT     _stop,15                ... posmatra se sadr`aj na adresi _stop. Specificirani bit code=15, odnosi
se na najni`i bit (LSB) na posmatranoj adresi. Instrukcija BIT kopira ovaj bit u TC bit status registra ST1. TC
(test/control flag bit) – je bit 11 status registra ST1 - ~uva rezultat operacije testiranja.
        BCND    loop,NTC                ... Ako je uslov NTC ispunjen (tj. ako je TC=0) program ide na loop
(labela – gore); Odnosno, ako je klikom na Stop u prozoru aplikacije setovano: _stop[0]=1 => izlazak iz petlje.
;
        END_DEMO                        ... END_DEMO je makro, definisan u Demos_a.h; Kada bude setovano
_stop[0]=1, tj. da je TC=1, program prestaje da se izvr{ava.
```

3.3.6.2. Simetri~na *PWM*

Ova aplikacija omogu}ava generisanje trofaznog napona pomo}u simetri~ne *PWM* modulacije. *General Purpose Timer 1* i tri *Full Compare Units* iz *Event manager* su programirani da daju 6 *PWM* signala. Kao rezultat dobijaju se trofazni naponi na izlaznom invertorskom mostu. Ova aplikacija se startuje pritiskom na ikonu ***PWM*** u ***Processor Evaluation Control Panel*** prozoru. Na slede}oj slici je dat grafi~ki prikaz ove aplikacije:



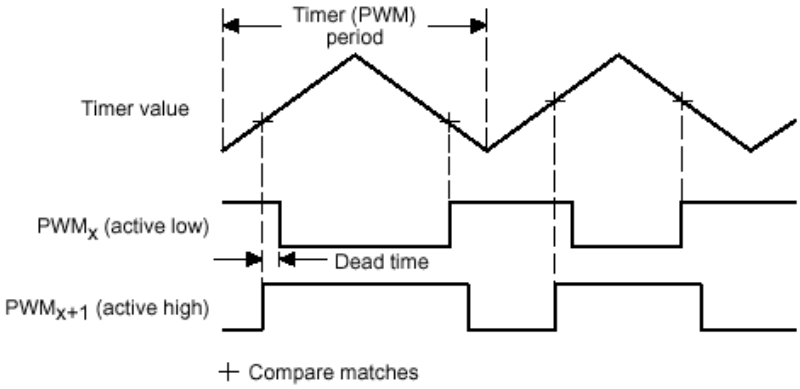
Slika 20. Prozor aplikacije za simetri~nu *PWM*

Idealna vrednost izlaznog napona invertora se dobija po izrazu:

$$U_{XY} = U_{DC} \cdot \frac{\text{X\_CMP\_VAL} - \text{Y\_CMP\_VAL}}{100}$$

gde X i Y mogu da uzmu vrednosti U, V ili W. Asemblerski program koji sadr`i kod ove aplikacije je ***pwms_a.asm*** i mo`e se videti pritiskom na taster ***Show code***. Zaglavlje asemblerskog programa je ***pwms_a.h*** koje sadr`i promenljive i funkcije za ovu aplikaciju. Startna adresa programa je **8000h** u eksternoj programskoj memoriji i zove se **_start**. Programske promenljive su u eksternoj memoriji za podatke i po~inju od adrese **a000h**, po slede}em rasporedu:

a000h: _stop indeks za stopiranje programa (1 za kraj programa)
a001h: _pwm_period sadr`i vrednost periode *PWM* nosioca koji se upisuje u *Timer 1 Period Register*
a002h: _u_cmp_value vrednost koja se sme{ta u *Full Compare Unit 1 Register*
a003h: _ v_cmp_value vrednost koja se sme{ta u *Full Compare Unit 2 Register*
a004h: _ w_cmp_value vrednost koja se sme{ta u *Full Compare Unit 3 Register*



Sl. 21. *Simetri~na PWM*

Procedura generisanja simetri~nog PWM signala je gotovo identi~na proceduri generisanja asimetri~nog PWM signala – jedina razlika se sastoji u tome, {to je sada GP Timer 1 potrebno podesiti da broji u *continuous-up/down* modu.
Prednost simetri~nog PWM signala nad asimetri~nim PWM signalom se sastoji u tome {to sadr`i dve neaktivne zone iste du`ine – na po~etku i kraju svakog PWM perioda.

Analiza ponu|enog programskog re{enja aplikacije

Ponu|eni asemblerski program *pwms_a.asm* je zasnovan na slede}em algoritmu:

1. Setuju se bitovi slede}ih registara:
   a. 0 se upisuje na adresu _stop=a000h. Kada najmla|i bit na ovoj adresi postane 1 (_stop[0]=1) tada program prestaje sa radom.
   b. 00 se upisuje u GPTCON[8-7] => GP Timer 1 ne}e startovati AD konverziju; GPTCON=7400h - General Purpose Timer Control Register (v. str. 2-38 u Spru161a.pdf).
   c. (_pwm_period=a001h)->(T1PER=7403h). Setuje se period GP Timera 1 tako {to se na adresu T1PER=7403h upisuje sadr`aj koji je korisnik izabrao u komandnom prozoru aplikacije (privremeno sme{ten na adresu _pwm_period=a001h) – kao periodu nosioca PWM.
   d. 0 se upisuje na adresu T1CNT=7401h – broja~ GP Timera 1 se postavlja na 0.
   e. 0 se upisuje na adresu DBTCON=7415h=>mrtva zona nije selektovana, tj. nula je.
   f. const. ACTIVE_HI_LO=0666h se upisuje na adresu ACTR=7413h – sadr`aj registra ACTR defini{e akciju na svakom od 6 *full compare output* pinova. U ovom slu~aju, ako je setovana PWM f-ja pomenutih pinova ima}emo da je PWM1,3,5/CMP1,3,5 – Active high, a PWM2,4,6/CMP2,4,6 - Active low
   g. (_u_cmp_val=a002h)->(CMPR1=7417h) – iz komandnog prozora aplikacije, setuje se sadr`aj ***compare*** **registra Full Compare Unit 1,** i to kao % perioda GPT1
   h. (_v_cmp_val=a003h)->(CMPR2=7418h) – iz komandnog prozora aplikacije, setuje se sadr`aj ***compare*** **registra Full Compare Unit 2,** i to kao % perioda GPT1
   i. (_w_cmp_val=a004h)->(CMPR3=7419h) – iz komandnog prozora aplikacije, setuje se sadr`aj ***compare*** **registra Full Compare Unit 3,** i to kao % perioda GPT1
   j. const. **TIM_COUNT_UP_DN=2800h**->(T1CON=7404h) - konfiguri{e se GP Timer 1. (Spru161a.pdf str. 2-37.) – GPT1 }e brojati na CPU clock frekvenciji u Continuous Up/Down modu (**bitno** za simetri~ni PWM!)
   k. FCU_MODE_PWM =7h ->(COMCON=7411h) => selektovan je PWM mod za **Full Compare Unit 1, 2 i 3.**
   l. 1 se upisuje u bit COMCON[15] (=1) => operacija pore|enja je dozvoljena
   m. 1 se upisuje u bit COMCON[9] (=1) => omogu}eno kori{}enje pridru`enih izlaznih pinova (tj. sada mogu da prenesu setovano stanje pridru`enog signala)
   n. 1 se upisuje u bit T1CON[6] (=1)=> start GP Timera 1
2. petlja:
   a. (_u_cmp_val=a002h)->(CMPR1=7417h)
   b. (_v_cmp_val=a003h)->(CMPR2=7418h)
   c. (_w_cmp_val=a004h)->(CMPR3=7419h)
   d. Program Monitor se aktivira (u svakom ciklusu).
   e. Proverava se u svakom ciklusu da li je u najni`i bit adrese _stop=a000h upisana jedinica ({to se de{ava pritiskom na Stop u komandnom prozoru aplikacije). Ako jeste to je uslov za izlazak iz petlje. Ako ne, sledi povratak na po~etak petlje.
3. KRAJ

Detaljniji uvid se mo`e ste}i na osnovu ponu|enog asemblerskog programa *pwms_a.asm*, koji sledi u nastavku (listing programa je u boji, a obja{njenja i primedbe nisu).

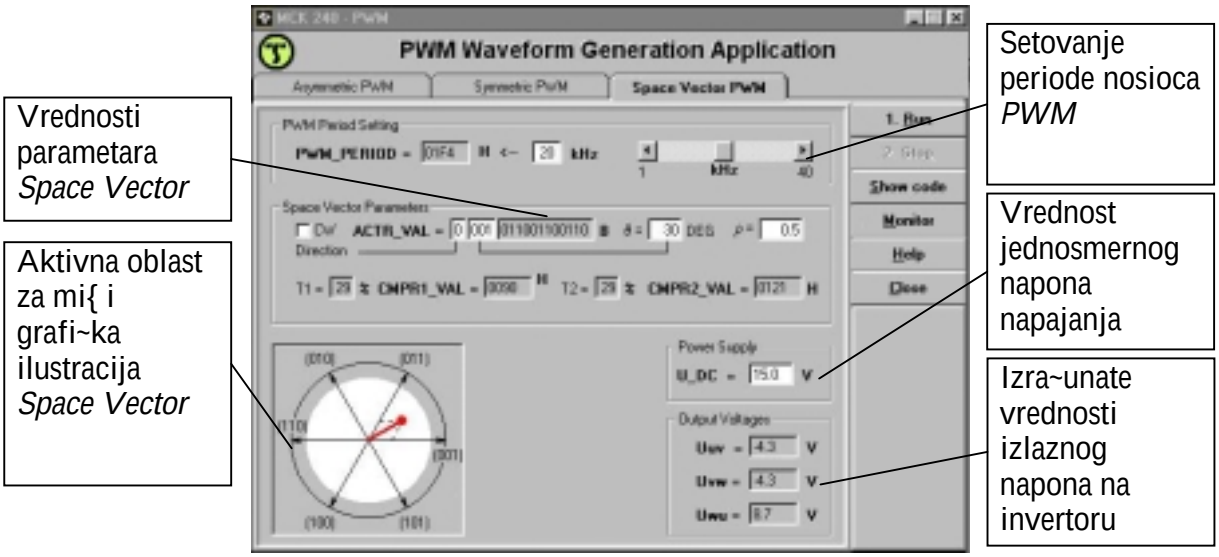
```
;*******************************************************************************
; File Name:            pwms_a.asm
; Project:      MCK240
; Originator:   R.Giuclea
; Description:          ASM file for symmetric PWM demo
; Copyright © 1997 Technosoft
;*******************************************************************************
; Include Files
;-------------------------------------------------------------------------------
                        .include        ..\F240_a.h
                        .include        ..\demos_a.h
                        .include        pwms_a.h
;===============================================================
                        .text
; -------------------------------------------------------------
_start:
        LDP     #_stop                  ... _stop = a000h (pi{e u  pwms.map). => DP=320.
        SPLK    #0,_stop                ... 0->(_stop), i to je sve tako dok se klikom na Stop u komandnom prozoru aplikacije ne setuje 1 kao sadr`aj adrese _stop ({to je uslov za izlaz iz programa).
; timer will not start ADC automatically
        LDP     #DP_EV                  ; Event Manager Data Page Pointer
... DP_EV je = 0E8h (videti F240_a.h); Setovana je nova vrednost data page pointera (DP=0E8h=232dec).
        LACC    GPTCON                  ... Upisuje se sadr`aj adrese 07400h (GPTCON) u akumulator (ACC);
        AND     #AND_T1TOADC_           ; AND mask for DISABLING ADC start on GPT1
... operacija logi~ko I se sprovodi nad bitovima sadr`aja adrese 07400h (GPTCON) i konstante AND_T1TOADC_= 0FE7Fh (v. Demos_a.h) – resetuje se bit 7 i 8 akumulatora.
        SACL    GPTCON                  ; configure GPTCON not to start ADC on GPT1 Event
... Sadr`aj akumulatora se upisuje na adresu 07400h (GPTCON) => GPTCON[8-7]=00 => GP Timer 1 ne}e startovati AD konverziju (videti Spru161a.pdf str. 2-38)
; load and init timer and PWM registers:
;
; load GPT1 timer period with the PWM periode
        LDP     #_pwm_period            ... _pwm_period=a001h (videti pwms.map), setuje se DP=320
        LACC    _pwm_period             ... upisuje se sadr`aj _pwm_period=a001h u akumulator; Sadr`aj _pwm_period=a001h setuje korisnik u komandnom prozoru aplikacije, i to je perioda nosioca PWM .
        LDP     #DP_EV                  ... Nova vrednost data page pointera: DP=0E8h=232dec.
        SACL    T1PER
... sadr`aj akumulatora se upisuje na adresu T1PER=7403h (videti F240_a.h), i tako defini{e period tajmera GPT1. Prakti~no: (_pwm_period=a001h)->( T1PER=7403h).
; set GPT1 counter initial value
        LACC    #0                      ... konstanta 0h se upisuje u akumulator
        SACL    T1CNT
... sadr`aj akumulatora se upisuje na adresu T1CNT=7401h (videti F240_a.h) – i radi se o adresi T1 Counter Registera (sadr`aj je broja~ GP Timera 1). Prakti~no: 0->( T1CNT=7401h)
; set deadbeat parameters
        LACC    #0                      ... konstanta 0h se upisuje u akumulator
        SACL    DBTCON
... sadr`aj akumulatora se upisuje na adresu DBTCON=7415h (videti F240_a.h) – i radi se o adresi Dead-Band Timer Control Registera (strana 2-54, Spru161a.pdf). Prakti~no: 0->( DBTCON=7415h)
; set action control register
        LACC    #ACTIVE_HI_LO           ... konstanta ACTIVE_HI_LO=0666h se upisuje u akumulator
        SACL    ACTR
... sadr`aj akumulatora se upisuje na adresu ACTR=7413h (videti F240_a.h) – i radi se o adresi Full Compare Action Control Registera (strana 2-48, Spru161a.pdf). Prakti~no: 0666h->( ACTR=7413h)
; set value for full compare unit 1 register
        LDP     #_u_cmp_val             ... Setuje se: DP=320. (_u_cmp_val=a002h (pwma.map))
        LACC    _u_cmp_val              ... upisuje se sadr`aj _u_cmp_val=a002h u akumulator; Sadr`aj _u_cmp_val=a002h setuje korisnik u komandnom prozoru aplikacije.
        LDP     #DP_EV                  ... Nova vrednost data page pointera: DP=0E8h=232dec.
        SACL    CMPR1
... sadr`aj akumulatora se upisuje na adresu CMPR1=7417h (videti F240_a.h) – Compare Channel 1 Threshold (CMPR1). Prakti~no: (_u_cmp_val=a002h)->( CMPR1=7417h)
; set value for full compare unit 2 register
        LDP     #_v_cmp_val             ... Setuje se: DP=320. (_v_cmp_val=a003h (pwms.map))
```

```
        LACC    _v_cmp_val              ... upisuje se sadr`aj _v_cmp_val=a003h u akumulator; Sadr`aj
_v_cmp_val=a003h setuje korisnik u komandnom prozoru aplikacije.
        LDP     #DP_EV                  ... Nova vrednost data page pointera: DP=0E8h=232dec.
        SACL    CMPR2
... sadr`aj akumulatora se upisuje na adresu CMPR2=7418h (videti F240_a.h) – Compare Channel 2 Threshold
(CMPR2). Prakti~no: (_v_cmp_val=a003h)->( CMPR2=7418h)
; set value for full compare unit 3 register
        LDP     #_w_cmp_val             ... Setuje se: DP=320. (_w_cmp_val=a004h (pwms.map))
        LACC    _w_cmp_val              ... upisuje se sadr`aj _w_cmp_val=a004h u akumulator; Sadr`aj
_w_cmp_val=a004h setuje korisnik u komandnom prozoru aplikacije.
        LDP     #DP_EV                  ... Nova vrednost data page pointera: DP=0E8h=232dec.
        SACL    CMPR3
... sadr`aj akumulatora se upisuje na adresu CMPR3=7419h (videti F240_a.h) – Compare Channel 3 Threshold
(CMPR3). Prakti~no: (_w_cmp_val=a004h)->( CMPR3=7419h)
; set GPT1 control register
        LACC    #TIM_CONT_UP_DN     ... upisuje se konstanta TIM_COUNT_DN=2800h u akumulator; `eli se
setovati tajmer u continuous up-down modu za simetri~ni PWM.
        SACL    T1CON
... sadr`aj akumulatora se upisuje na adresu T1CON=7404h (videti F240_a.h) – kontrolni registar za GPT1.
Prakti~no: TIM_COUNT_DN=2800h->( T1CON=7404h)
; set FCU control register
        LACC    #FCU_MODE_PWM       ... upisuje se konstanta FCU_MODE_PWM =7h u akumulator;
        SACL    COMCON

... sadr`aj akumulatora se upisuje na adresu COMCON=7411h (videti F240_a.h) – Compare Control Register (v.
stranu 2-45, Spru161a.pdf). Prakti~no: FCU_MODE_PWM =7h ->( COMCON=7411h)
; start PWM generation
        SETBIT COMCON,SETB15        ;enable FCU compare operation
... SETBIT je makro koji je definisan u Demos_a.h
... => COMCON[15]=1 (videti Spru161a.pdf str. 2-45) => funkcija pore|enja dozvoljena
        SETBIT COMCON,SETB9         ;enable FCU output pins
... => COMCON[9]=1 (videti Spru161a.pdf str. 2-46) => omogu}eno kori{}enje compare output pinova
        SETBIT T1CON,SETB6  ;start GPT1 counter
... => T1CON[6]=1 (videti Spru161a.pdf str. 2-36) => startuje se GP Timer 1
;
loop:                                   ... labela (ozna~ava po~etak petlje)
; set value for full compare unit 1 register    ... Prakti~no: (_u_cmp_val=a002h)->( CMPR1=7417h):
        LDP     #_u_cmp_val             ... Setuje se: DP=320. (_u_cmp_val=a002h (pwms.map))
        LACC    _u_cmp_val              ... upisuje se sadr`aj _u_cmp_val=a002h u akumulator;
        LDP     #DP_EV                  ... Nova vrednost data page pointera: DP=0E8h=232dec.
        SACL    CMPR1                   ... sadr`aj akumulatora se upisuje na adresu CMPR1=7417h
; set value for full compare unit 2 register    ... Prakti~no: (_v_cmp_val=a003h)->( CMPR2=7418h):
        LDP     #_v_cmp_val             ... Setuje se: DP=320. (_v_cmp_val=a003h (pwms.map))
        LACC    _v_cmp_val              ... upisuje se sadr`aj _v_cmp_val=a003h u akumulator.
        LDP     #DP_EV                  ... Nova vrednost data page pointera: DP=0E8h=232dec.
        SACL    CMPR2                   ... sadr`aj akumulatora se upisuje na adresu CMPR2=7418h
; set value for full compare unit 3 register    ... Prakti~no: (_w_cmp_val=a004h)->( CMPR3=7419h)
        LDP     #_w_cmp_val             ... Setuje se: DP=320. (_w_cmp_val=a004h (pwms.map))
        LACC    _w_cmp_val              ... upisuje se sadr`aj _w_cmp_val=a004h u akumulator;
        LDP     #DP_EV                  ... Nova vrednost data page pointera: DP=0E8h=232dec.
        SACL    CMPR3                   ... sadr`aj akumulatora se upisuje na adresu CMPR3=7419h
; call monitor
        CALL    MON240                  ... Programski broja~ (PC-program counter) se inkrementira i stavlja na
vrh steka. Sadr`aj adrese u programskoj memoriji MON240=0109h (videti cap_a.h). postaje sadr`aj programskog
broja~a. Poziva se program monitor.
; test if demo ends (_stop =1)
        LDP     #_stop                  ... setuje se DP=320
        BIT     _stop,15                ... posmatra se sadr`aj na adresi _stop. Specificirani bit code=15, odnosi
se na najni`i bit (LSB) na posmatranoj adresi. Instrukcija BIT kopira ovaj bit u TC bit status registra ST1. TC
(test/control flag bit) – je bit 11 status registra ST1 - ~uva rezultat operacije testiranja.
        BCND    loop,NTC                ... Ako je uslov NTC ispunjen (tj. ako je TC=0) program ide na loop
(labela – gore); Odnosno, ako je klikom na Stop u prozoru aplikacije setovano: _stop[0]=1 => izlazak iz petlje.
;
        END_DEMO                        ... END_DEMO je makro, definisan u Demos_a.h; Kada bude setovano
_stop[0]=1, tj. da je TC=1, program prestaje da se izvr{ava.
```

3.3.3.3. *Space Vector PWM*

Ova aplikacija omogu}ava generisanje trofaznog napona pomo}u *Space Vector PWM* modulacije. *General Purpose Timer* 1 i tri *Full Compare Units* iz *Event manager* su programirani da daju 6 *PWM* signala. Kao rezultat dobijaju se trofazni naponi na izlaznom invertorskom mostu. Ova aplikacija se startuje pritiskom na ikonu ***PWM*** u ***Processor Evaluation Control Panel*** prozoru. Na slede}oj slici je dat grafi~ki prikaz ove aplikacije:



Slika 22. Prozor aplikacije za *Space Vector PWM*

Idealna vrednost izlaznog napona invertora se dobija po izrazu:

$$U_{UV} = U_{DC} \cdot \frac{T1 \cdot \left(b_2\left(V_p\right) - b_1\left(V_p\right)\right) + T2 \cdot \left(b_2\left(V_{p+60}\right) - b_1\left(V_{p+60}\right)\right)}{100}$$

$$U_{VW} = U_{DC} \cdot \frac{T1 \cdot \left(b_1\left(V_p\right) - b_0\left(V_p\right)\right) + T2 \cdot \left(b_1\left(V_{p+60}\right) - b_0\left(V_{p+60}\right)\right)}{100}$$

$$U_{WU} = U_{DC} \cdot \frac{T1 \cdot \left(b_0\left(V_p\right) - b_2\left(V_p\right)\right) + T2 \cdot \left(b_0\left(V_{p+60}\right) - b_2\left(V_{p+60}\right)\right)}{100}$$

gde su V_p i V_{p+60} dva susedna vektora okarakterisana tro-bitnim *Gray* kodom. Asemblerski program koji sadr`i kod ove aplikacije je ***pwmv_a.asm*** i mo`e se videti pritiskom na taster ***Show code***. Zaglavlje asemblerskog programa je ***pwmv_a.h*** koje sadr`i promenljive i funkcije za ovu aplikaciju. Startna adresa programa je **8000h** u eksternoj programskoj memoriji i zove se **_start**. Programske promenljive su u eksternoj memoriji za podatke i po~inju od adrese **a000h**, po slede}em rasporedu:

a000h: _stop indeks za stopiranje programa (1 za kraj programa)
a001h: _pwm_period sadr`i vrednost periode PWM nosioca koji se upisuje u *Timer 1 Period Register*
a002h: _actr_val vrednost koja se sme{ta u *Action Control Register*
a003h: _ cmpr1_value vrednost koja se sme{ta u *Full Compare Unit 1 Register*
a004h: _ cmpr2_value vrednost koja se sme{ta u *Full Compare Unit 2 Register*

Za pra}enje ove aplikacije, korisno je pro~itati *Space Vector PWM* (Spru161a.pdf, strana 2-66).

Analiza ponu|enog programskog re{enja aplikacije

Ponu|eni asemblerski program *pwmv_a.asm* je zasnovan na slede}em algoritmu:

1. Setuju se bitovi slede}ih registara:
   a. 0 se upisuje na adresu _stop=a000h. Kada najmla|i bit na ovoj adresi postane 1 (_stop[0]=1) tada program prestaje sa radom.
   a. 00 se upisuje u GPTCON[8-7] => GP Timer 1 ne}e startovati AD konverziju; GPTCON=7400h - General Purpose Timer Control Register (v. str. 2-38 u Spru161a.pdf).
   b. (_pwm_period=a001h)->(T1PER=7403h). Setuje se period GP Timera 1 tako {to se na adresu T1PER=7403h upisuje sadr`aj koji je korisnik izabrao u komandnom prozoru aplikacije (privremeno sme{ten na adresu _pwm_period=a001h) – kao periodu nosioca PWM.
   c. 0 se upisuje na adresu T1CNT=7401h – broja~ GP Timera 1 se postavlja na 0.
   d. 0 se upisuje na adresu DBTCON=7415h=>mrtva zona nije selektovana, tj. nula je.
   e. (_actr_val=a002h)->(ACTR=7413h). Sadr`aj _actr_val=a002h se delimi~no de-fini{e u komandnom prozoru aplikacije, tj. setuju se bitovi 15-12, dok je su drugi bitovi fiksni: _actr_val[11-0]=666h. Bitovi registra ACTR[11-0] defini{u akciju na svakom od 6 *full compare output* pinova. U ovom slu~aju (ACTR[11-0]=666h), ako je setovana PWM f-ja pomenutih pinova ima}emo da je PWM1,3,5/CMP1,3,5 – Active high, a PWM2,4,6/CMP2,4,6 - Active low. Bit ACTR[15] odre|uje smer rotacije space vectora, a bitovi ACTR[14-12] odre|uju smer osnovnog space vectora.
   f. (_cmpr1_val=a003h)->(CMPR1=7417h) – iz komandnog prozora aplikacije, setuje se sadr`aj **compare registra Full Compare Unit 1,** i to na osnovu koordinata space vectora
   g. (_cmpr2_va1=a004h)->(CMPR2=7418h) – iz komandnog prozora aplikacije, setuje se sadr`aj **compare registra Full Compare Unit 1,** i to na osnovu koordinata space vectora
   h. const. **TIM_COUNT_UP_DN=2800h**->(T1CON=7404h) - konfiguri{e se GP Timer 1. (Spru161a.pdf str. 2-37.) – GPT1 }e brojati na CPU clock frekvenciji u Continuous Up/Down modu (**kao** za simetri~ni PWM)
   i. FCU_SVEC_EN =1007h ->(COMCON=7411h) => dozvoljen je space vector PWM mod i selektovan je PWM mod za **Full Compare Unit 1, 2 i 3.**
   j. 1 se upisuje u bit COMCON[15] (=1) => operacija pore|enja je dozvoljena
   k. 1 se upisuje u bit COMCON[9] (=1) => omogu}eno kori{}enje pridru`enih izlaznih pinova (tj. sada mogu da prenesu setovano stanje pridru`enog signala)
   l. 1 se upisuje u bit T1CON[6] (=1)=> start GP Timera 1
4. petlja:
   a. (_actr_val=a002h)->(ACTR=7413h)
   b. (_cmpr1_val=a003h)->(CMPR1=7417h)
   c. (_cmpr2_va1=a004h)->(CMPR2=7418h)
   d. Program Monitor se aktivira (u svakom ciklusu).
   e. Proverava se u svakom ciklusu da li je u najni`i bit adrese _stop=a000h upisana jedinica ({to se de{ava pritiskom na Stop u komandnom prozoru aplikacije). Ako jeste to je uslov za izlazak iz petlje. Ako ne, sledi povratak na po~etak petlje.
3. KRAJ

Detaljniji uvid se mo`e ste}i na osnovu ponu|enog asemblerskog programa *pwmv_a.asm*, koji sledi u nastavku (listing programa je u boji, a obja{njenja i primedbe nisu).

```
;SPACE VECTOR MODULATION:
;This does not work 100% correctly (probably HW/MC240 problem).
;Differences with respect to pure PWM:
;ACTR, SPC vector programmed (b15-14-13-12), 03666h, 0011 0110 0110 0110
```

```
;used only CMPR1 and CMPR2, programming the duration of CW (CMPR1)
;and CCW (CMPR2) active voltage vector of the sector
;COMCON is programmed to 1007 Hex (was 7 for full compare PWM).
;COMCON bit b.12 "Space vector enable" is set (overriding other settings).
;in COMCON, we still need to enable b.15 and b.9 to start the operation
;*******************************************************************************
; File Name:   pwmv_a.asm
; Project:     MCK240
; Originator:  R.Giuclea
; Description: ASM file for space vector PWM demo
; Copyright © 1997 Technosoft
;*******************************************************************************
; Include Files
;-------------------------------------------------------------------------------
    .include      ..\F240_a.h
    .include      ..\demos_a.h
    .include      pwmv_a.h
;===============================================================================
    .text
; ------------------------------------------------------------------------------
_start:
  LDP   #_stop                          ... _stop = a000h (pi{e u  pwmv.map). Starijih 9 bita a000h odre|uju vrednost data page pointera DP=320.
  SPLK  #0,_stop                        ... 0->(_stop), i to je sve tako dok se klikom na Stop u komandnom prozoru aplikacije ne setuje 1 kao sadr`aj adrese _stop ({to je uslov za izlaz iz programa).
; timer will not start ADC automatically
  LDP   #DP_EV      ; Event Manager Data Page Pointer
... DP_EV je = 0E8h (videti F240_a.h); Setovana je nova vrednost data page pointera (DP=0E8h=232dec).
  LACC  GPTCON                          ... Upisuje se sadr`aj adrese 07400h (GPTCON) u akumulator (ACC);
  AND   #AND_T1TOADC_  ; AND mask for DISABLING ADC start on GPT1
... operacija logi~ko I se sprovodi nad bitovima sadr`aja adrese 07400h (GPTCON) i konstante AND_T1TOADC_= 0FE7Fh (v. Demos_a.h) – resetuje se bit 7 i 8 akumulatora.
  SACL  GPTCON      ; configure GPTCON not to start ADC on GPT1 Event
... Sadr`aj akumulatora se upisuje na adresu 07400h (GPTCON) => GPTCON[8-7]=00 => GP Timer 1 ne}e startovati AD konverziju (videti Spru161a.pdf str. 2-38)
; load and init timer and PWM registers:
;
; load GPT1 timer period with the PWM periode
  LDP   #_pwm_period                    ... _pwm_period=a001h (videti pwmv.map), setuje se DP=320
  LACC  _pwm_period                     ... upisuje se sadr`aj _pwm_period=a001h u akumulator; Sadr`aj _pwm_period=a001h setuje korisnik u komandnom prozoru aplikacije, i to je perioda nosioca PWM .
  LDP   #DP_EV                          ... Nova vrednost data page pointera: DP=0E8h=232dec.
  SACL  T1PER
... sadr`aj akumulatora se upisuje na adresu T1PER=7403h (videti F240_a.h), i tako defini{e period tajmera GPT1. Prakti~no: (_pwm_period=a001h)->( T1PER=7403h).
; set GPT1 counter initial value
  LACC  #0                              ... konstanta 0h se upisuje u akumulator
  SACL  T1CNT
 ... sadr`aj akumulatora se upisuje na adresu T1CNT=7401h (videti F240_a.h) – i radi se o adresi T1 Counter Registera (sadr`aj je broja~ GP Timera 1). Prakti~no: 0->( T1CNT=7401h)
; set deadbeat parameters
  LACC  #0                              ... konstanta 0h se upisuje u akumulator
  SACL  DBTCON
... sadr`aj akumulatora se upisuje na adresu DBTCON=7415h (videti F240_a.h) – i radi se o adresi Dead-Band Timer Control Registera (strana 2-54, Spru161a.pdf). Prakti~no: 0->( DBTCON=7415h)
; set initial vector in action control register: 03666 Hex.
  LDP   #_actr_val  ; 0011 -> 0-CCW SPCV direction, 011 -> Segment
... _actr_val=a002h; => DP=320
  LACC  _actr_val   ; 0110 -> 6 -> PWM6-PWM5 Active low-active high
... sadr`aj _actr_val=a002h (videti u komandnom prozoru aplikacije) se upisuje u akumulator
  LDP   #DP_EV                          ... Nova vrednost data page pointera: DP=0E8h=232dec.
  SACL  ACTR                            ... sadr`aj akumulatora se upisuje na adresu ACTR=7413h (videti F240_a.h) – i radi se o adresi Full Compare Action Control Registera (strana 2-48, Spru161a.pdf). Prakti~no: (_actr_val=a002h)->( ACTR=7413h)
; set value for full compare unit 1 register
  LDP   #_cmpr1_val                     ... Setuje se: DP=320. (_cmpr1_va1=a003h (pwmv.map))
  LACC  _cmpr1_val                      ... upisuje se sadr`aj _cmpr1_val=a003h u akumulator; Sadr`aj _cmpr1_val=a003h setuje korisnik u komandnom prozoru aplikacije.
  LDP   #DP_EV                          ... Nova vrednost data page pointera: DP=0E8h=232dec.
```

SACL CMPR1
... sadr`aj akumulatora se upisuje na adresu CMPR1=7417h (videti F240_a.h) – Compare Channel 1 Threshold (CMPR1). Prakti~no: (_cmpr1_val=a003h)->(CMPR1=7417h)
; set value for full compare unit 2 register
LDP #_cmpr2_val ... Setuje se: DP=320. (_cmpr2_va1=a004h (pwmv.map))
LACC _cmpr2_val ... upisuje se sadr`aj _cmpr2_val=a004h u akumulator; Sadr`aj _cmpr2_val=a004h setuje korisnik u komandnom prozoru aplikacije.
LDP #DP_EV ... Nova vrednost data page pointera: DP=0E8h=232dec.
SACL CMPR2
... sadr`aj akumulatora se upisuje na adresu CMPR2=7418h (videti F240_a.h) – Compare Channel 2 Threshold (CMPR2). Prakti~no: (_cmpr2_va1=a004h)->(CMPR2=7418h)
; set GPT1 control register
LACC #TIM_CONT_UP_DN ... upisuje se konstanta TIM_COUNT_DN=2800h u akumulator; `eli se setovati tajmer u continuous up-down modu za simetri~ni PWM.
SACL T1CON
... sadr`aj akumulatora se upisuje na adresu T1CON=7404h (videti F240_a.h) – kontrolni registar za GPT1. Prakti~no: TIM_COUNT_DN=2800h->(T1CON=7404h)
; set FCU control register
LACC #FCU_SVEC_EN ... upisuje se konstanta FCU_SVEC_EN =1007h u akumulator;
SACL COMCON
... sadr`aj akumulatora se upisuje na adresu COMCON=7411h (videti F240_a.h) – Compare Control Register (v.. stranu 2-45, Spru161a.pdf). Prakti~no: FCU_SVEC_EN =1007h ->(COMCON=7411h)
; start PWM generation
SETBIT COMCON,SETB15 ;enable FCU compare operation
... SETBIT je makro koji je definisan u Demos_a.h
... => COMCON[15]=1 (videti Spru161a.pdf str. 2-45) => funkcija pore|enja dozvoljena
SETBIT COMCON,SETB9 ;enable FCU output pins
... => COMCON[9]=1 (videti Spru161a.pdf str. 2-46) => omogu}eno kori{}enje compare output pinova
SETBIT T1CON,SETB6 ;start GPT1 counter
... => T1CON[6]=1 (videti Spru161a.pdf str. 2-36) => startuje se GP Timer 1
;
loop: ... labela (ozna~ava po~etak petlje)
; update PWM values
;
LDP #_actr_val ... _actr_val=a002h; => DP=320
LACC _actr_val ... sadr`aj _actr_val=a002h se upisuje u akumulator
LDP #DP_EV ... Nova vrednost data page pointera: DP=0E8h=232dec.
SACL ACTR ... sadr`aj akumulatora se upisuje na adresu ACTR=7413h (videti F240_a.h) – i radi se o adresi Full Compare Action Control Registera (strana 2-48, Spru161a.pdf). Prakti~no: (_actr_val=a002h)->(ACTR=7413h)
LDP #_cmpr1_val ... Setuje se: DP=320. (_cmpr1_va1=a003h (pwmv.map))
LACC _cmpr1_val ... upisuje se sadr`aj _cmpr1_val=a003h u akumulator;
LDP #DP_EV ... Nova vrednost data page pointera: DP=0E8h=232dec.
SACL CMPR1 ... sadr`aj akumulatora se upisuje na adresu CMPR1=7417h (videti F240_a.h) – Compare Channel 1 Threshold (CMPR1). Prakti~no: (_cmpr1_val=a003h)->(CMPR1=7417h)
LDP #_cmpr2_val ... Setuje se: DP=320. (_cmpr2_va1=a004h (pwmv.map))
LACC _cmpr2_val ... upisuje se sadr`aj _cmpr2_val=a004h u akumulator;
LDP #DP_EV ... Nova vrednost data page pointera: DP=0E8h=232dec.
SACL CMPR2 ... sadr`aj akumulatora se upisuje na adresu CMPR2=7418h (videti F240_a.h) – Compare Channel 2 Threshold (CMPR2). Prakti~no: (_cmpr2_va1=a004h)->(CMPR2=7418h)
; call monitor
CALL MON240 ... Programski broja~ (PC-program counter) se inkrementira i stavlja na vrh steka. Sadr`aj adrese u programskoj memoriji MON240=0109h (videti cap_a.h). postaje sadr`aj programskog broja~a. Poziva se program monitor.
; test if demo ends (_stop =1)
LDP #_stop ... setuje se DP=320
BIT _stop,15 ... posmatra se sadr`aj na adresi _stop. Specificirani bit code=15, odnosi se na najni`i bit (LSB) na posmatranoj adresi. Instrukcija BIT kopira ovaj bit u TC bit status registra ST1. TC (test/control flag bit) – je bit 11 status registra ST1 - ~uva rezultat operacije testiranja.
BCND loop,NTC ... Ako je uslov NTC ispunjen (tj. ako je TC=0) program ide na loop (labela – gore); Odnosno, ako je klikom na Stop u prozoru aplikacije setovano: _stop[0]=1 => izlazak iz petlje.
END_DEMO ... END_DEMO je makro, definisan u Demos_a.h; Kada bude setovano _stop[0]=1, tj. da je TC=1, program prestaje da se izvr{ava.

LITERATURA

1. Texas Instruments, *TMS320C24x DSP Controllers Reference Set, Volume 1: CPU, System and Instruction Set*, SPRU160A, Houston, March 1997.
2. Texas Instruments, *TMS320C24x DSP Controllers Reference Set, Volume 2: Peripheral Library and Specific Devices*, SPRU161A, Houston, March 1997.
3. Technosoft, *Help for MCWIN – Windows Interface for MCK240 (TMS320F240 DSP Motion Control Kit)*, 1997.