Osnovi digitalne elektronike IR

vežba 4

Odsek za elektroniku

Elektrotehnički fakultet,
Univerzitet u Beogradu

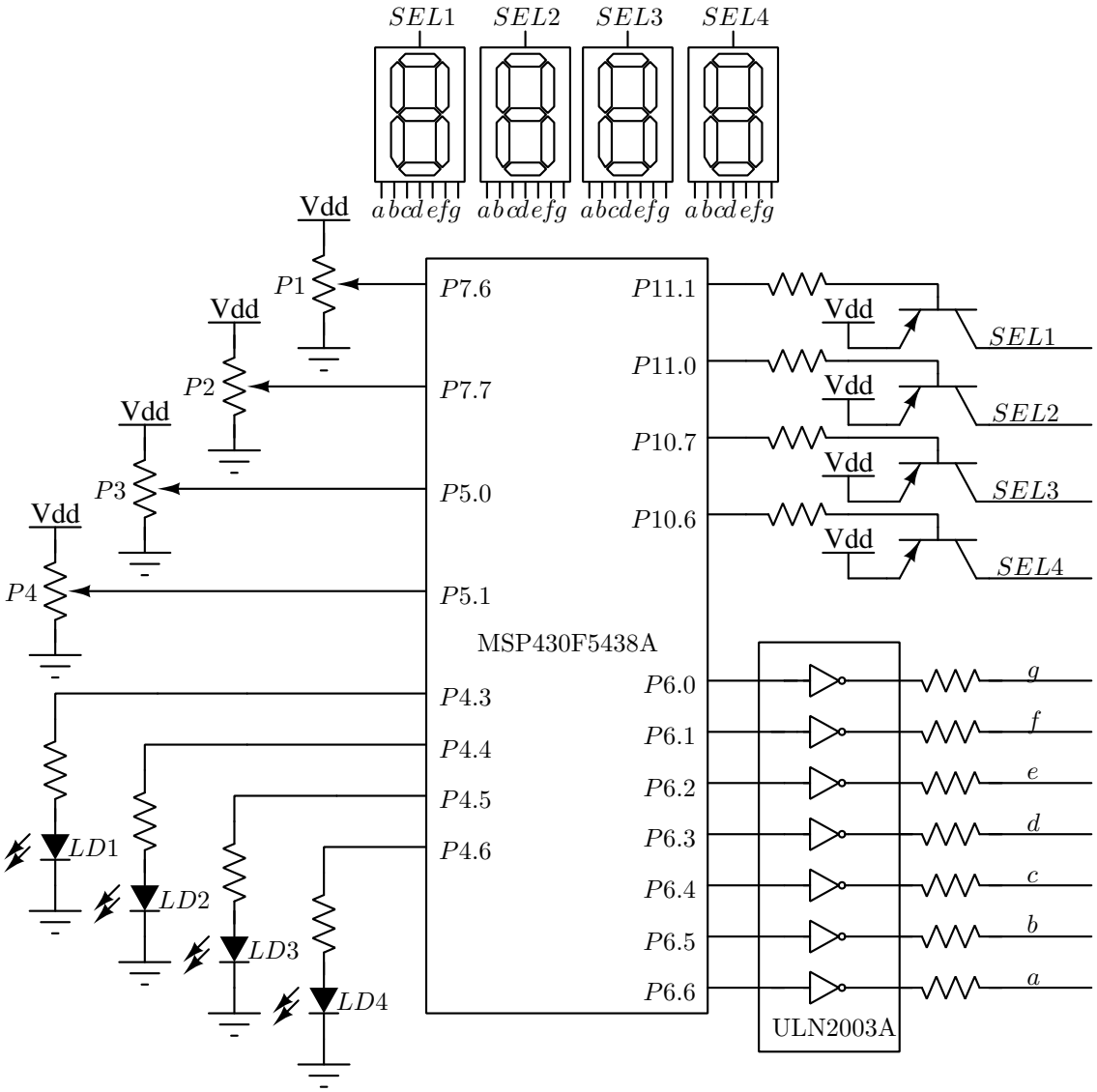
2018/2019

Pregled

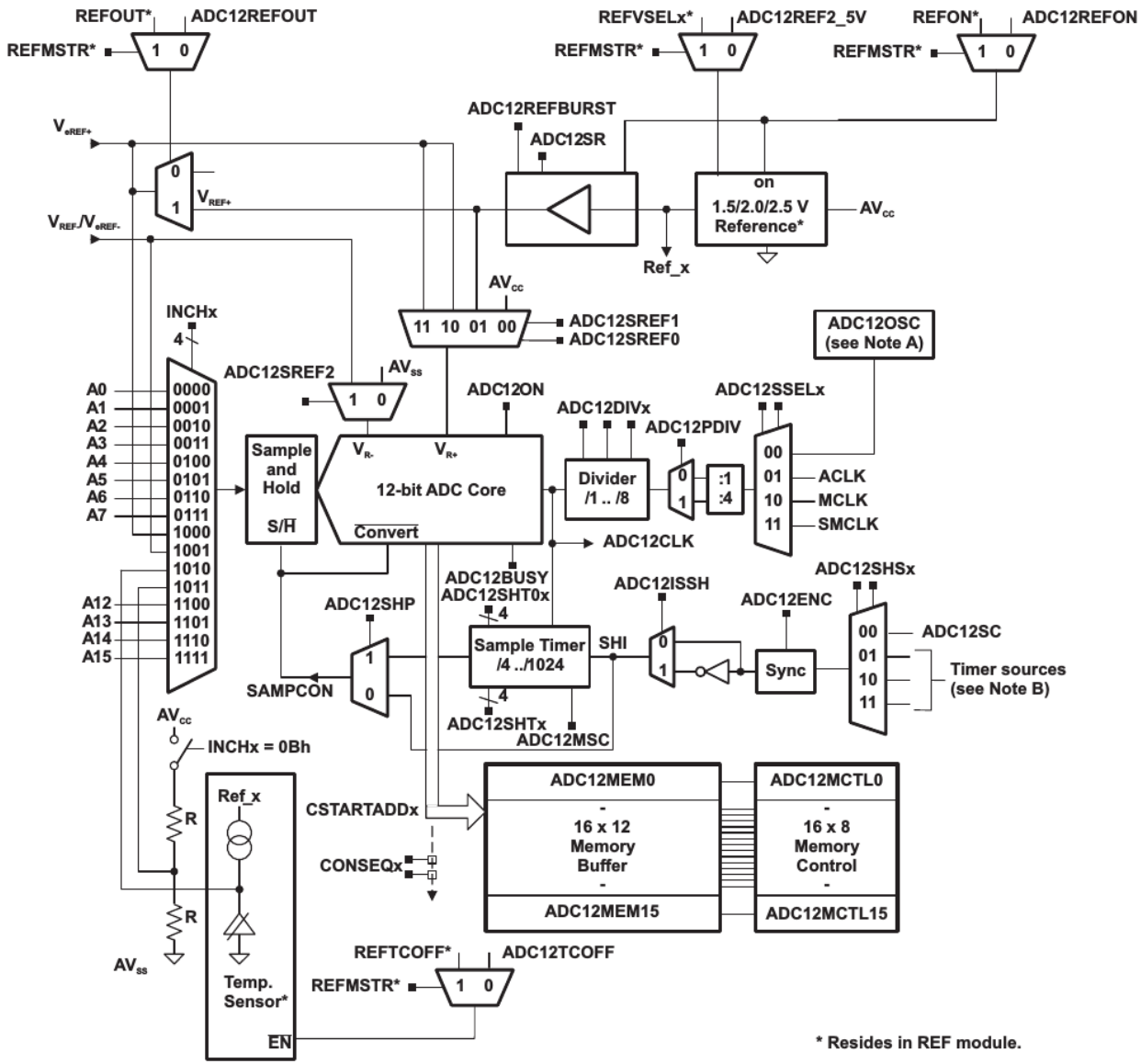
1. AD Konvertor i tajmeri
   - 12bit AD Konvertor
   - Tajmeri
2. Primeri
   - AD Konvertori
   - Tajmeri
   - Tajmeri i AD konvertori

Pregled

Opis hardvera



ADC12 1/3


REFOUT*
ADC12REFOUT
REFMSTR*
REFVSELx*
ADC12REF2_5V
REFON*
ADC12REFON
ADC12REFBURST
ADC12SR
on
1.5/2.0/2.5 V
Reference*
Ref_x
INCHx
ADC12SREF2
ADC12SREF1
ADC12SREF0
ADC12OSC
(see Note A)
ADC12ON
ADC12DIVx
ADC12PDIV
ADC12SSELx
Sample
and
Hold
S/H
12-bit ADC Core
Convert
Divider
/1 .. /8
:1
:4
ACLK
MCLK
SMCLK
ADC12CLK
ADC12BUSY
ADC12SHT0x
ADC12SHP
ADC12ISSH
ADC12ENC
ADC12SHSx
Sample Timer
/4 ../1024
SHI
Sync
ADC12SC
Timer sources
(see Note B)
SAMPCON
ADC12SHTx
ADC12MSC
INCHx = 0Bh
CSTARTADDx
CONSEQx
ADC12MEM0
16 x 12
Memory
Buffer
ADC12MEM15
ADC12MCTL0
16 x 8
Memory
Control
ADC12MCTL15
REFTCOFF*
ADC12TCOFF
Temp.
Sensor*
EN
* Resides in REF module.

ADC12 2/3

12-bitni AD konvertor sa sukcesivnim aproksimacijama

Do 12 nezavisnih eksternih kanala

Posebni kanali za interni temperaturni senzor i eksternu naponsku referencu

Maksimalna brzina konverzija do 200ksps

ADC12 3/3

- Softverski kontrolisano SH kolo
- Tri interne ili eksterna naponska referenca
- 16 nezavisnih baferskih registara
- Prekid sa brzim dekodovanjem izvora prekida

ADC12 funkcionisanje 1/2

Konvertuje ulazni napon u opsegu V_{ref-} do V_{ref+} po sledećoj formuli:

$$N_{ADC} = 4095 \times \frac{V_{in} - V_{ref-}}{V_{ref+} - V_{ref-}}$$

Referentni naponi su konfigurabilni

Za konfiguraciju 12bit AD konvertora zadužena su kontrolni registri (Videti *User Guide*)

Pre korišćenja AD konvertora neophodno je uključiti AD konvertor setovanjem `ADC12ON` bita. Zbog čega se isključuje AD konvertor kada se ne koristi?

ADC12 funkcionisanje 2/2

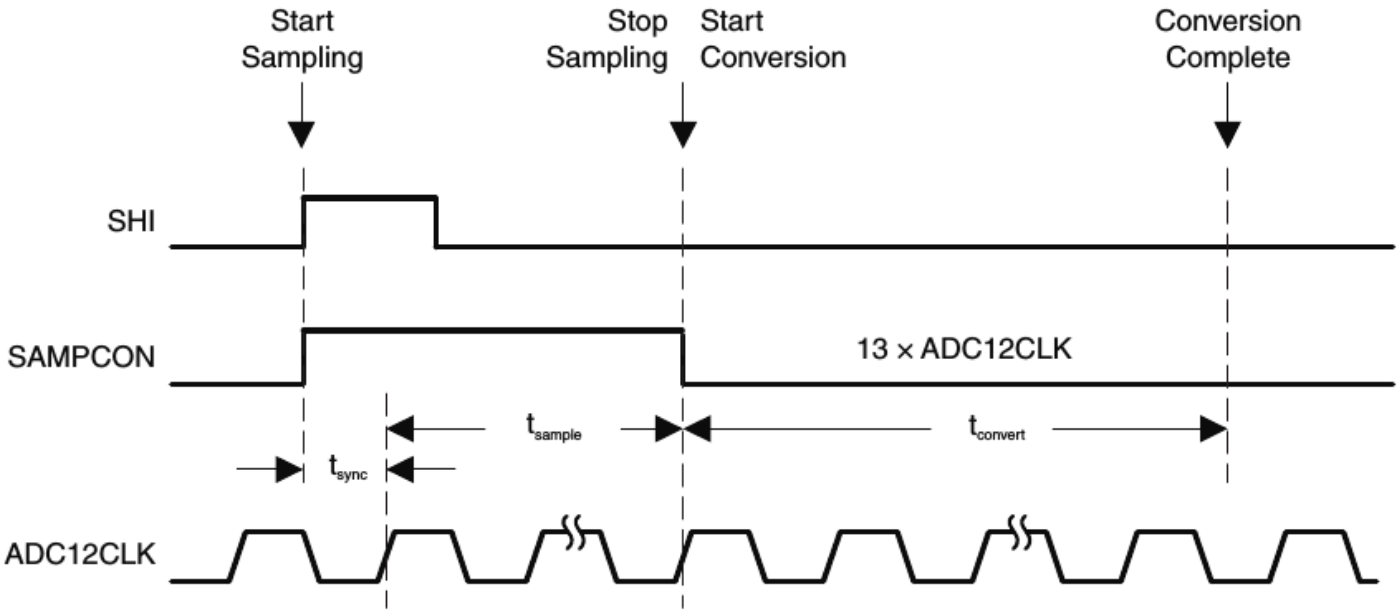
Da bi konfigurisanje AD konvertora bilo moguće neophodno je isključiti konverziju sto se postiže resetovanjem `ADC12ENC` bita. Ponovno uključivanje AD konverzije vrši se setovanjem ovog bita nakon što je konvertor konfigurisan

Konverzija signala započinje nakon detektovanja uzlazne ivice SHI signala. Ovaj signal može biti generisan softverski (setovanjem `ADC12SC` bita) ili od strane nekog od tajmerskih modula `Timer A` i `Timer B`

Na raspolaganju je 16 registara za čuvanje rezultata konverzije. Uz svaki registar može da se asocira bilo koji kanal

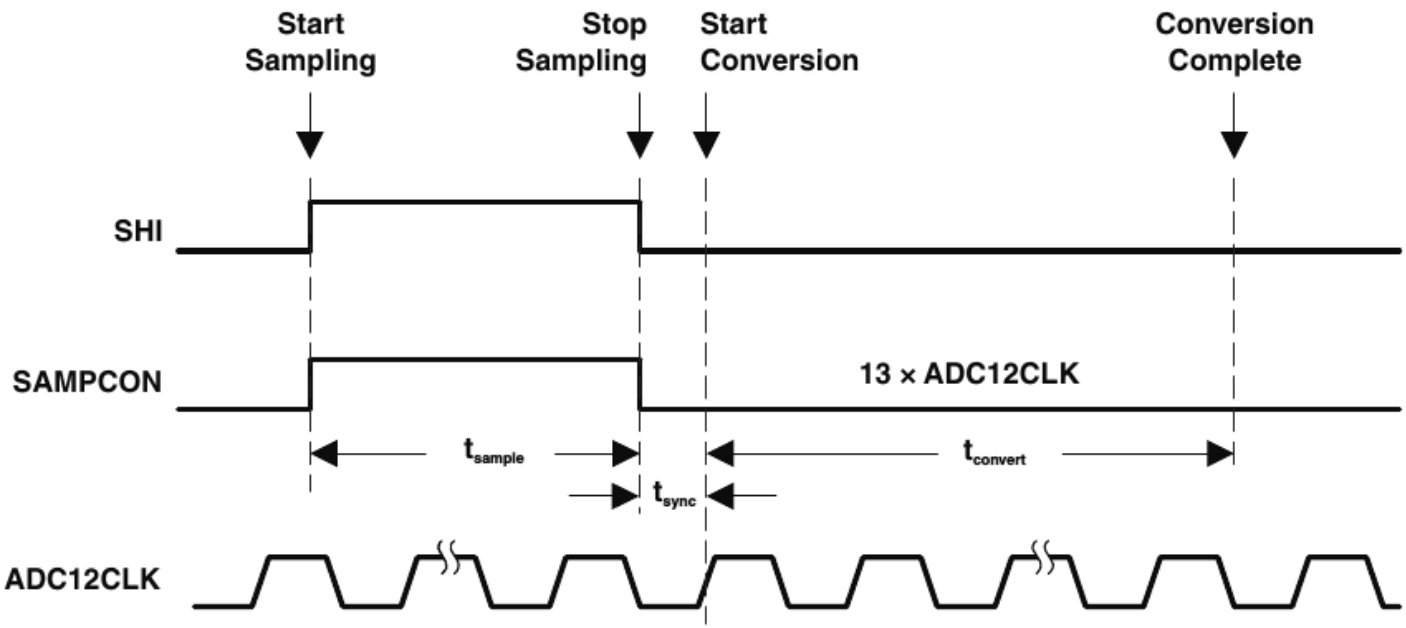
Zadavanje konverzije 1/2

Jedan način kontrole je softversko podešavanje trajanja *sampling* perioda



Zadavanje konverzije 2/2

Drugi način je direktna kontrola `SHI` signalom



Modovi konverzije

ADC12CONSEQx	Mode	Operation
00	Single-channel single-conversion	A single channel is converted once.
01	Sequence-of-channels (autoscan)	A sequence of channels is converted once.
10	Repeat-single-channel	A single channel is converted repeatedly.
11	Repeat-sequence-of-channels (repeated autoscan)	A sequence of channels is converted repeatedly.

ADC12 prekidi 1/2

16 prekida asociranih sa `ADC12IFGx` flegom koji se setuje kada se u odgovarajući registar upiše rezultat konverzije

Prekid asociran sa `ADC12OV` flegom koji se događa kada se u neki od baferskih registara `ADC12MEMx` upisuje novi rezultat pre nego što je stari pročitan

`ADC12TOV` prekid koji se događa kada se inicira nova konverzija pre nego što je tekuća završena

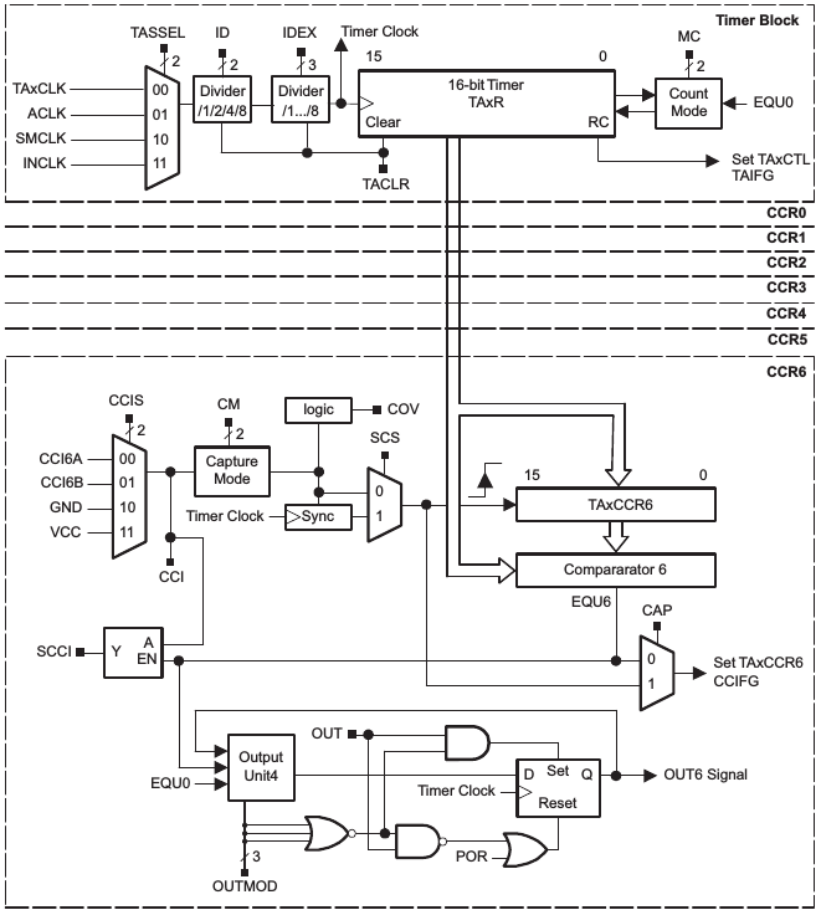
ADC12 prekidi

Svaki od 18 izvora prekida je moguće individualno maskirati

Iako postoji samo jedan prekidni vektor za sve ADC12 prekide, korišćenjem generatora prekidnog vektora ADC12IV u kome je kodiran jedan od 18 flegova koji izazivaju prekid lako se realizuje grananje u prekidnoj rutini

Tajmer A

Tajmer A je 16-bitni tajmer sa nekoliko *capture/compare* blokova, čiji broj varira kod različitih predstavnika familije od 3 do 7

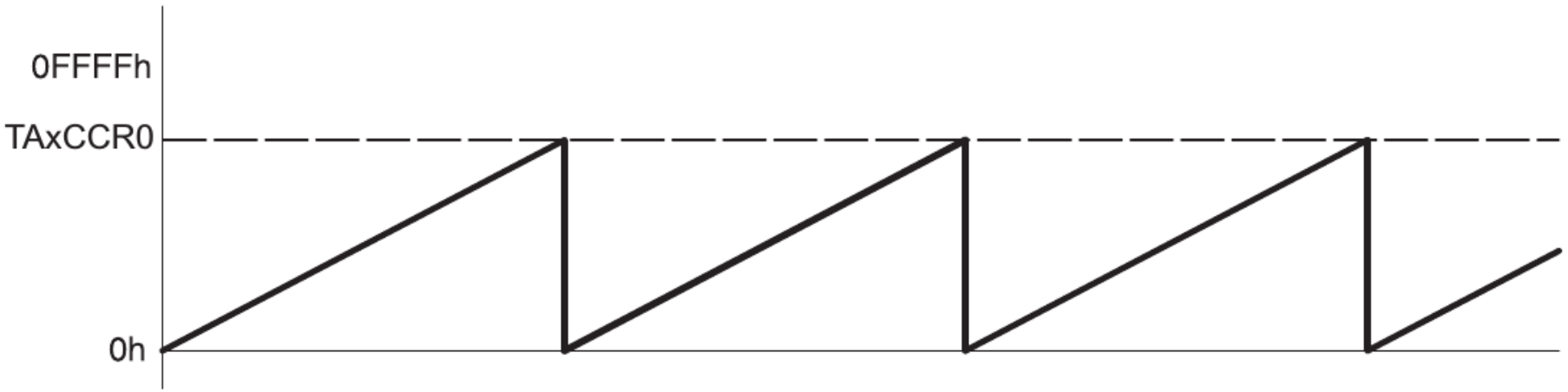


Modovi rada brojača TAxR

MC	Mode	Description
00	Stop	The timer is halted.
01	Up	The timer repeatedly counts from zero to the value of TAxCCR0
10	Continuous	The timer repeatedly counts from zero to 0FFFFh.
11	Up/down	The timer repeatedly counts from zero up to the value of TAxCCR0 and back down to zero.

Brojač na gore (UP mod)

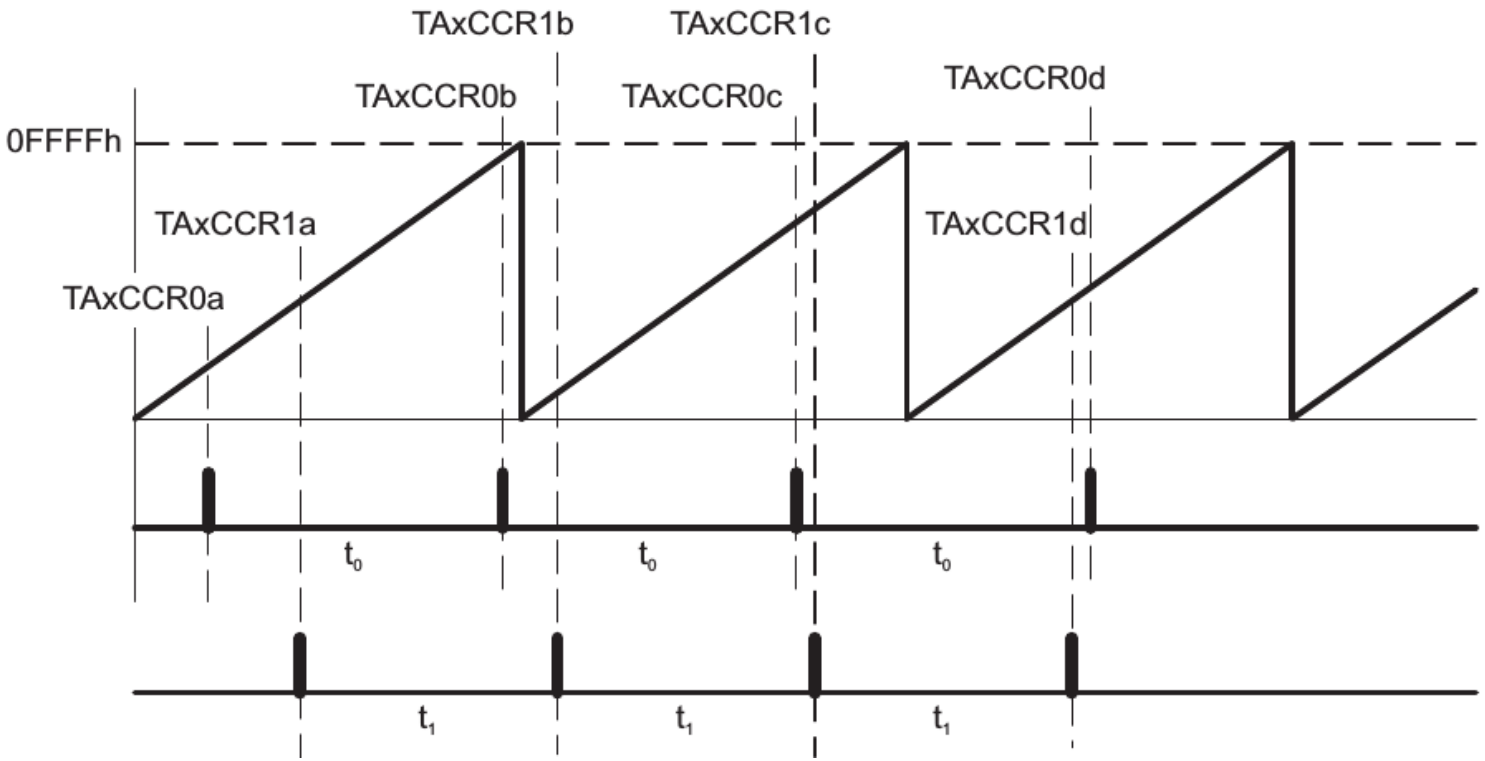
Po dostizanju vrednosti `TAxCCR0` setuje se `CCIFG` fleg a pri resetovanju brojača na `0x0000` setuje se `TAxIFG` fleg



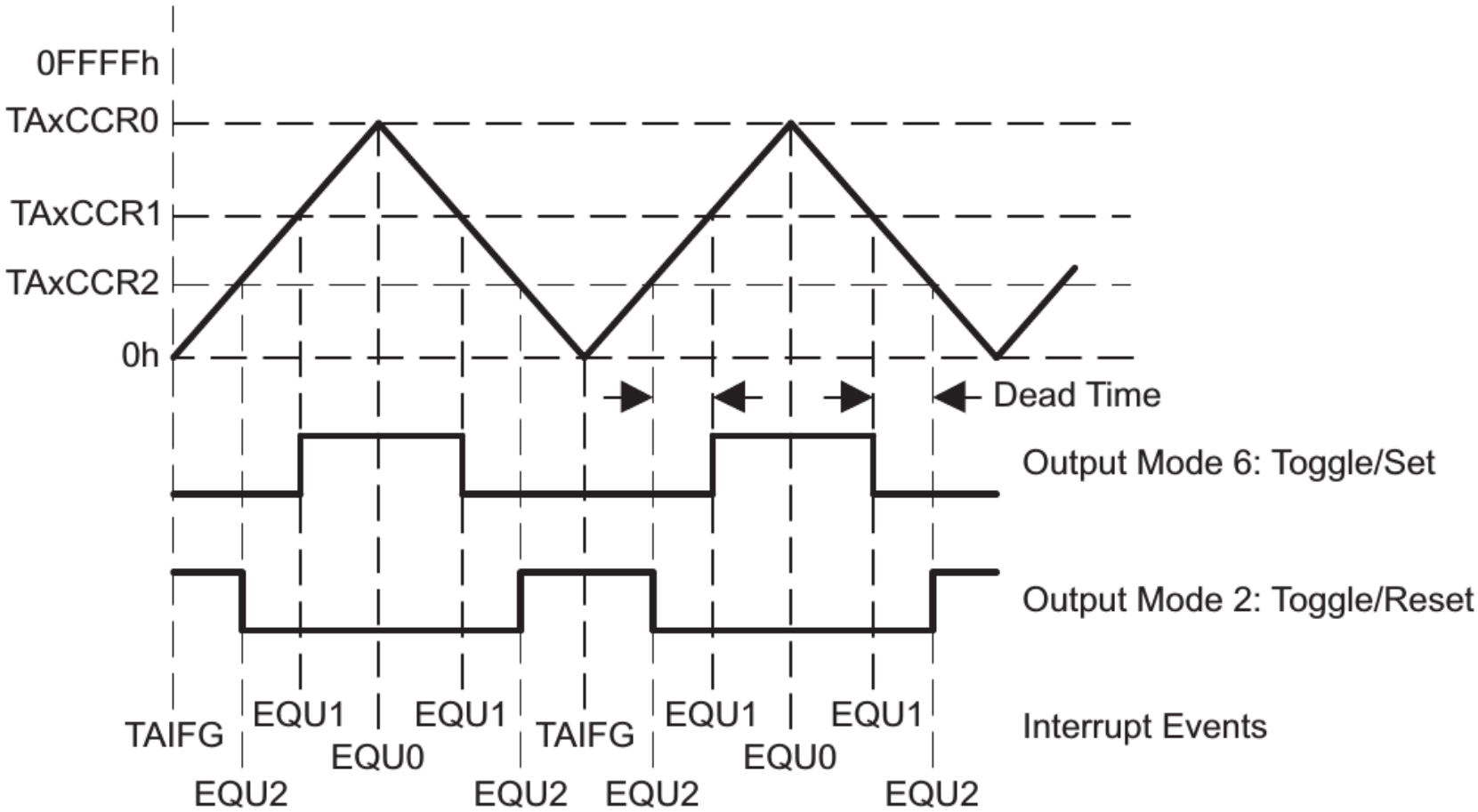
Kontinualni brojač (`Continuous` mod)

Tajmer uvek broji od `0x0000` do `0xffff`.

Ovaj način brojanja u `COMPARE` modu tajmera je pogodan za generisanje signala različitih učestanosti



UP/DOWN brojač



CAPTURE/COMPARE blokovi

Svaki CC blok može da radi u CAPTURE ili COMPARE modu. To je određeno bitom CAP u kontrolnom registru TAxCCTLn

CAPTURE mod se koristi za merenje učestanosti, tj. intervala između uzastopnih događaja na nekom pinu. Svaka promena na određenom pinu kopira trenutnu vrednost brojača TAxR u odgovarajući TAxCCRn registar is setuje odgovarajući CCIFG fleg.

COMPARE mod se koristi za generisanje PWM-a i signala različitih učestanosti. Svako izjednačavanje vrednosti brojača TAxR sa vrednošću u TAxCCRn registru setuje odgovarajući TAxCCRn CCIFG fleg

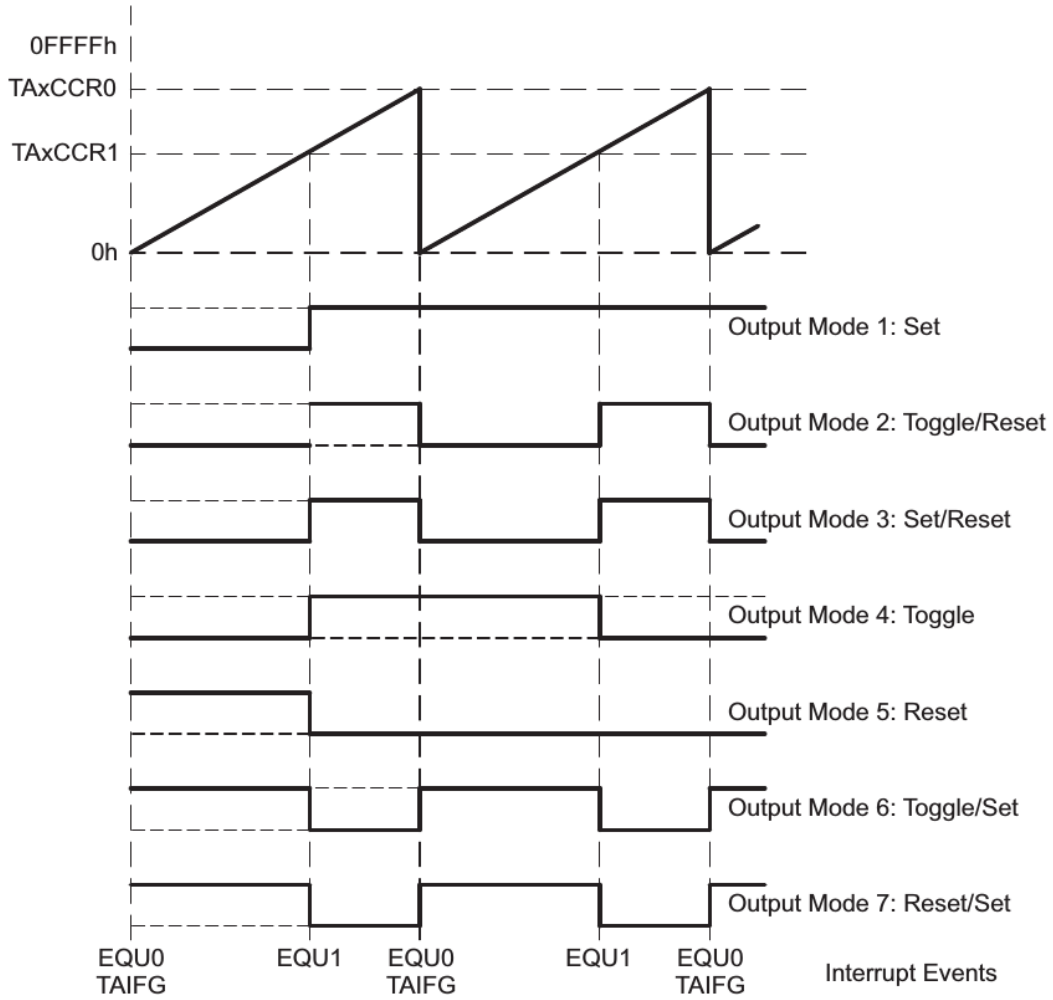
Izlazna jedinica

Svaki `CC` blok sadrži izlaznu jedinicu koja se koristi za generisanje signala kao što je PWM

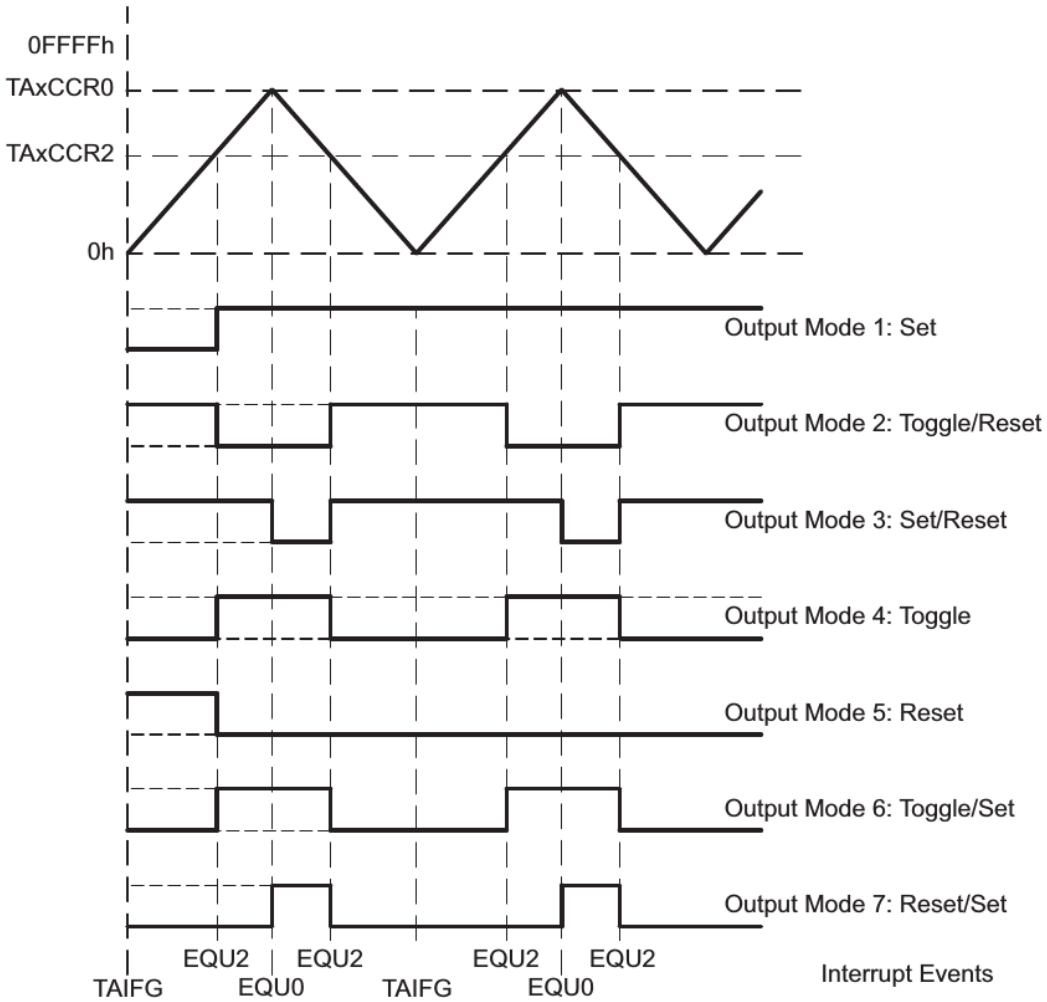
Izlazna jednica može da radi u osam različitih modova

OUTMODx	Mode	Description
000	Output	The output signal OUTn is defined by the OUT bit. The OUTn signal updates immediately when OUT is updated.
001	Set	The output is set when the timer *counts* to the TAxCCRn value. It remains set until a reset of the timer, or until another output mode is selected and affects the output.
010	Toggle/Reset	The output is toggled when the timer *counts* to the TAxCCRn value. It is reset when the timer *counts* to the TAxCCR0 value.
011	Set/Reset	The output is set when the timer *counts* to the TAxCCRn value. It is reset when the timer *counts* to the TAxCCR0 value.
100	Toggle	The output is toggled when the timer *counts* to the TAxCCRn value. The output period is double the timer period.
101	Reset	The output is reset when the timer *counts* to the TAxCCRn value. It remains reset until another output mode is selected and affects the output.
110	Toggle/Set	The output is toggled when the timer *counts* to the TAxCCRn value. It is set when the timer *counts* to the TAxCCR0 value.
111	Reset/Set	The output is reset when the timer *counts* to the TAxCCRn value. It is set when the timer *counts* to the TAxCCR0 value.

Rad izlaznog bloka za brojač u modu brojanja na gore



Rad izlaznog bloka za brojač u UP/DOWN modu brojanja



Prekidi Tajmera A

Dva prekidna vektora su povezana sa tajmerom A:

- `TAxCCR0` vektor za `TAxCCR0 CCIFG` fleg
- `TAxIV` vektor za ostale `CCIFG` flegove i `TAxIFG`

U `CAPTURE` modu `CCIFG` fleg se setuje kada se na spoljni događaj registar `TAxCCRn` napuni vrednošću brojača `TAxR`

U `COMPARE` modu `CCIFG` fleg se setuje kada vrednost brojača `TAxR` dostigne vrednost u `TAxCCRn` registru

`TAIFG` fleg se setuje kada vrednost brojača `TAxR` dostigne `0x0000`

Unutar `TAxIV` prekida se poliranjem određuje koji je događaj izazvao prekid

Registar prekidnih vektora

15	14	13	12	11	10	9	8
TAIV							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
TAIV							
r0	r0	r0	r0	r-(0)	r-(0)	r-(0)	r0

Table 17-8. TAxIV Register Description

Bit	Field	Type	Reset	Description
15-0	TAIV	R	0h	Timer_A interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: Capture/compare 1; Interrupt Flag: TAxCCR1 CCIFG; Interrupt Priority: Highest 04h = Interrupt Source: Capture/compare 2; Interrupt Flag: TAxCCR2 CCIFG 06h = Interrupt Source: Capture/compare 3; Interrupt Flag: TAxCCR3 CCIFG 08h = Interrupt Source: Capture/compare 4; Interrupt Flag: TAxCCR4 CCIFG 0Ah = Interrupt Source: Capture/compare 5; Interrupt Flag: TAxCCR5 CCIFG 0Ch = Interrupt Source: Capture/compare 6; Interrupt Flag: TAxCCR6 CCIFG 0Eh = Interrupt Source: Timer overflow; Interrupt Flag: TAxCTL TAIFG; Interrupt Priority: Lowest

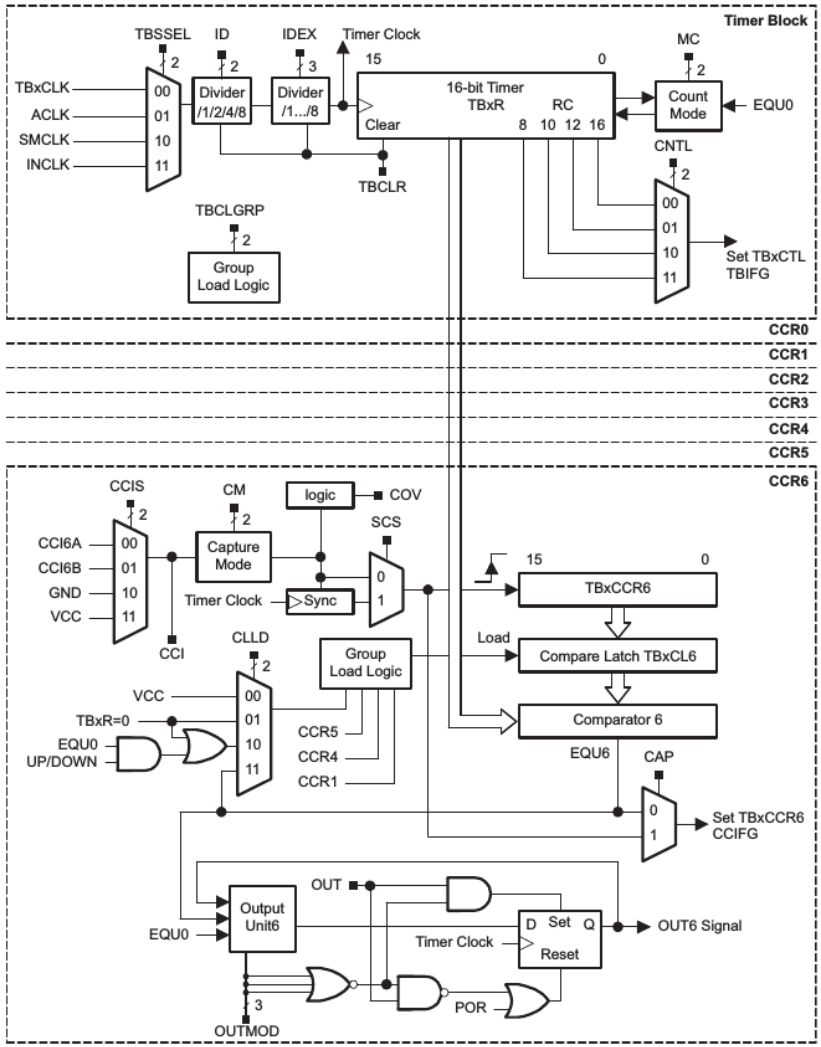
Primer prekidnih rutina tajmera A

```
; Interrupt handler for TA0CCR0 CCIFG.                          Cycles
CCIFG_0_HND
;         ...           ; Start of handler Interrupt latency     6
          RETI                                                   5

; Interrupt handler for TA0IFG, TA0CCR1 through TA0CCR6 CCIFG.

TA0_HND       ...                 ; Interrupt latency            6
          ADD       &TA0IV,PC     ; Add offset to Jump table     3
          RETI                    ; Vector  0: No interrupt      5
          JMP       CCIFG_1_HND   ; Vector  2: TA0CCR1           2
          JMP       CCIFG_2_HND   ; Vector  4: TA0CCR2           2
          JMP       CCIFG_3_HND   ; Vector  6: TA0CCR3           2
          JMP       CCIFG_4_HND   ; Vector  8: TA0CCR4           2
          JMP       CCIFG_5_HND   ; Vector 10: TA0CCR5           2
          JMP       CCIFG_6_HND   ; Vector 12: TA0CCR6           2

TA0IFG_HND                        ; Vector 14: TA0IFG Flag
          ...                     ; Task starts here
          RETI                                                   5

CCIFG_6_HND                       ; Vector 12: TA0CCR6
          ...                     ; Task starts here
          RETI                    ; Back to main program         5

CCIFG_5_HND                       ; Vector 10: TA0CCR5
          ...                     ; Task starts here
          RETI                    ; Back to main program         5

CCIFG_4_HND                       ; Vector 8: TA0CCR4
          ...                     ; Task starts here
          RETI                    ; Back to main program         5

CCIFG_3_HND                       ; Vector 6: TA0CCR3
          ...                     ; Task starts here
          RETI                    ; Back to main program         5

CCIFG_2_HND                       ; Vector 4: TA0CCR2
          ...                     ; Task starts here
          RETI                    ; Back to main program         5

CCIFG_1_HND                       ; Vector 2: TA0CCR1
          ...                     ; Task starts here
          RETI                    ; Back to main program         5
```

Tajmer B

Vrlo sličnih karakteristika kao i tajmer A

Ima više `CC` jedinica i mogućnost sinhronog upisivanja



Pregled

Zadatak 1 - Startovanje AD konverzije i čitanje rezultata

Zadatak

Napisati program koji pritiskom na taster S4 vrši startovanje konverzije napona na potenciometru P1. Na sedmosegmentnom displeju se ispisuje konvertovana vrednost.

Napomena

Na displeju je potrebno ispisivati heksadecimalnu vrednost koja predstavlja 4 najviša bita digitalnog zapisa konvertovane veličine.

Zadatak 1 - Startovanje AD konverzije i čitanje rezultata

Rešenje

Da bi se ispunili uslovi zadatka potrebno je konfigurisati sledeće:

- SAMPCON signal se generiše softverski u *Pulse* modu
- Kao izvor takta koristi se interni oscilator AD konvertora
- Konvertuje se kanal 14 AD konvertora jer je na njega povezan potenciometar P1
- AD Konvertor je u *Single channel single conversion* modu

Kôd

Videti primer *v4-z1-adc-tast* u materijalima

Zadatak 2 - PWM signal

Zadatak

Napisati program koji realizuje kontrolu osvetljaja diode LD1.
Pritiskom na taster S4 vrši se povećanje intenziteta osvetljaja diode.

Zadatak 2 - PWM signal

PWM

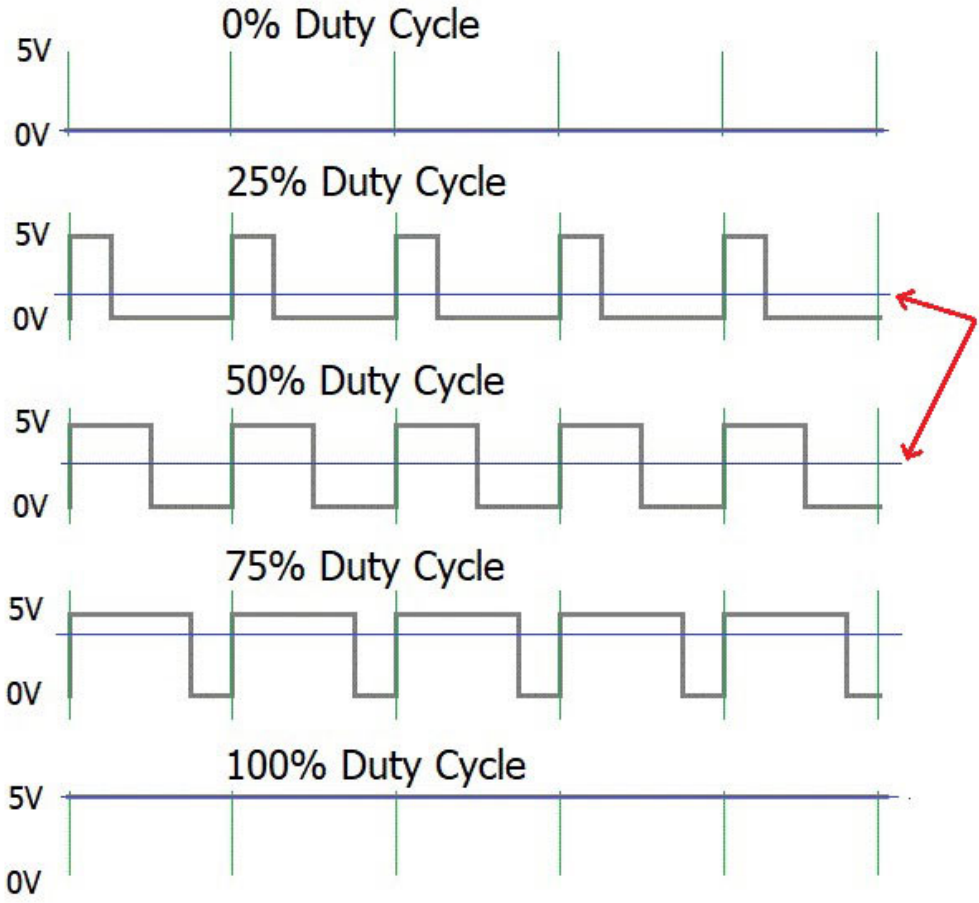
Šta je PWM signal? Koja je veza između intenziteta osvetljaja diode i PWM signala?

Zadatak 2 - PWM signal

PWM - *Pulse Width Modulation*

Način generisanja signala kod koga je moguće menjati trajanje impulsa/pauze.

PWM signal karakteriše perioda (D) i Duty Cycle (D)



Zadatak 2 - PWM signal

Kod signala generisanog na ovaj način lako se menja srednja vrednost signala

Intenzitet osvetljaja diode, brzina okretanja motora, ... zavise od srednje vrednosti signala

Kôd

Videti primer *v4-z2-tmp-pwm* u materijalima

Zadatak 3 - Startovanje AD konverzije koristeći tajmer

Zadatak

Napisati program koji sa periodom od 0.5s vrši startovanje konverzije napona na potenciometru P1. Na sedmosegmentnom displeju se ispisuje konvertovana vrednost.

Napomena

Na displeju je potrebno ispisivati heksadecimalnu vrednost koja predstavlja 4 najviša bita digitalnog zapisa konvertovane veličine.

Zadatak 3 -Startovanje AD konverzije koristeći tajmer

Rešenje

AD konvertor u ovom zadatku se inicijalizuje na sličan način kao i u prethodnom. Razlika je u sledećem:

- Kao početak konverzije uzima se uzlazna ivica signala generisanog od strane tajmerskog modula `Timer A`
- AD Konvertor je u *Repeat-single-channel* modu

Kôd

Videti primer *v4-z3-adc-simple-tmr* u materijalima

Zadatak 4 - Generisanje PWM signala

Zadatak

Napisati program koji generiše PWM signal kod koga je trajanje impulsa srazmerno konvertovanoj vrednosti dobijenoj sa potenciometra P1. PWM signal kontroliše intenzitet osvetljaja diode LD1. Konverzija analogne vrednosti vrši se sa periodom od 0.5s.

Napomena

Korstiti alternativnu funkciju na pinu koji je povezan sa diodom LD1. Zadatak realizovati bez upotrebe prekida!

Zadatak 4 - Generisanje PWM signala

Rešenje

U ovom primeru Tajmer A koristimo kao triger za početak AD konverzija dok tajmer B koristimo za generisanje PWM signala

Ako se detaljnije pogleda *DataSheet* ovog mikrokontrolera videćemo da je alternativna funkcija na pinu

PIN NAME (P4.x)	x	FUNCTION	CONTROL BITS OR SIGNALS	
			P4DIR.x	P4SEL.x
P4.0/TB0.0	0	4.0 (I/O)	I: 0; O: 1	0
		TB0.CCI0A and TB0.CCI0B	0	1
		TB0.0(1)	1	1
P4.1/TB0.1	1	4.1 (I/O)	I: 0; O: 1	0
		TB0.CCI1A and TB0.CCI1B	0	1
		TB0.1(1)	1	1
P4.2/TB0.2	2	4.2 (I/O)	I: 0; O: 1	0
		TB0.CCI2A and TB0.CCI2B	0	1
		TB0.2(1)	1	1
P4.3/TB0.3	3	4.3 (I/O)	I: 0; O: 1	0
		TB0.CCI3A and TB0.CCI3B	0	1
		TB0.3(1)	1	1

Zadatak 4 - Generisanje PWM signala

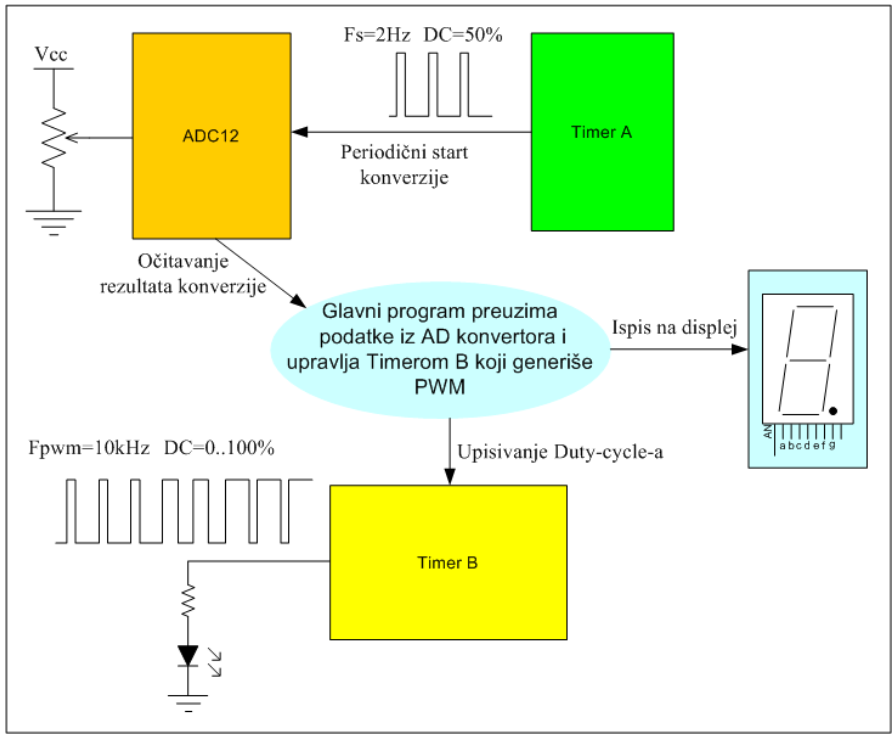
U ovom primeru funcionalnost programa implementirana je jedino u glavnoj programskoj petlji

Niz slajdova koji slede u nastavku treba da ilustruju implementirani algoritam

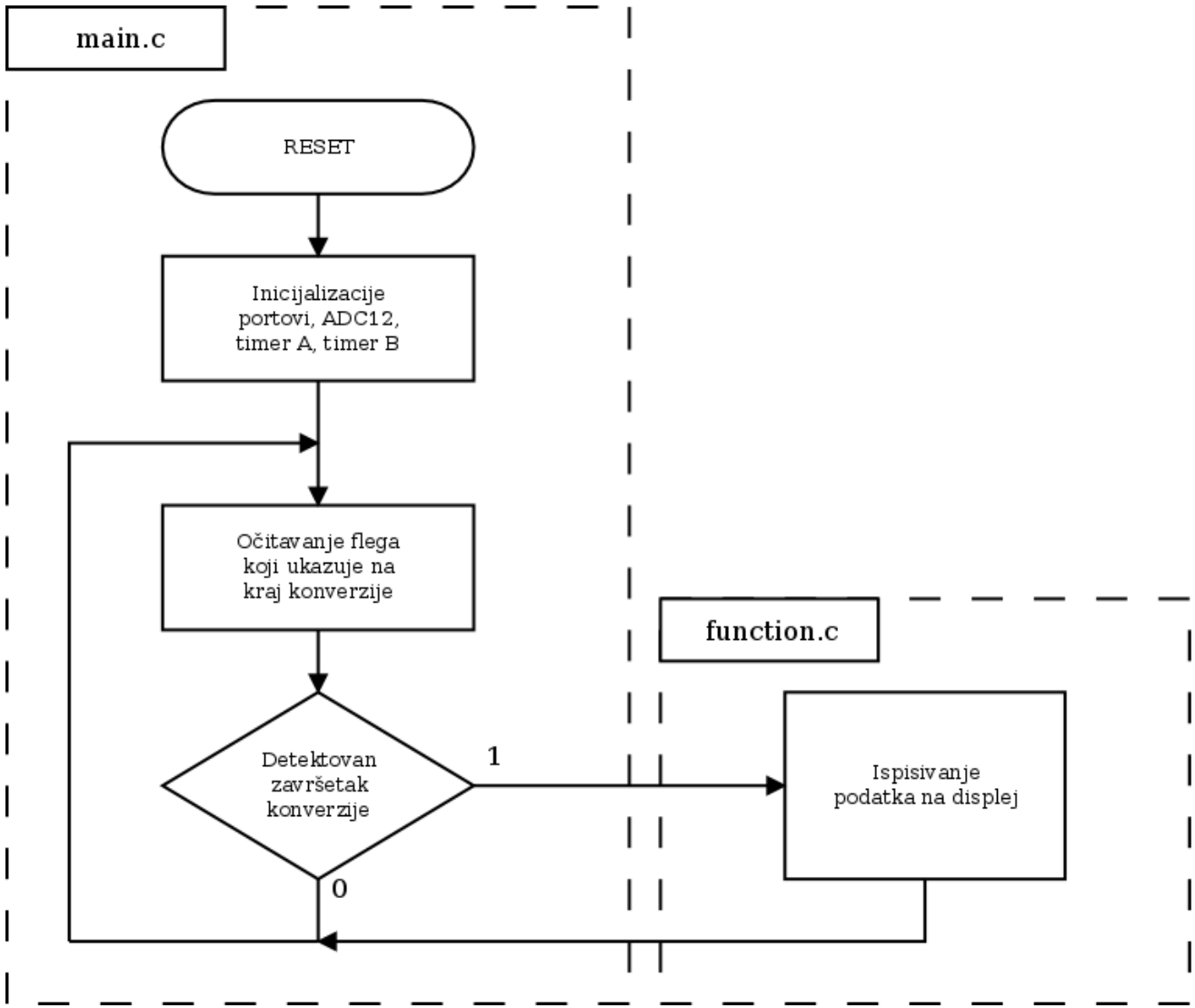
Kôd

Videti primer *v4-z4-adc-tmr* u materijalima

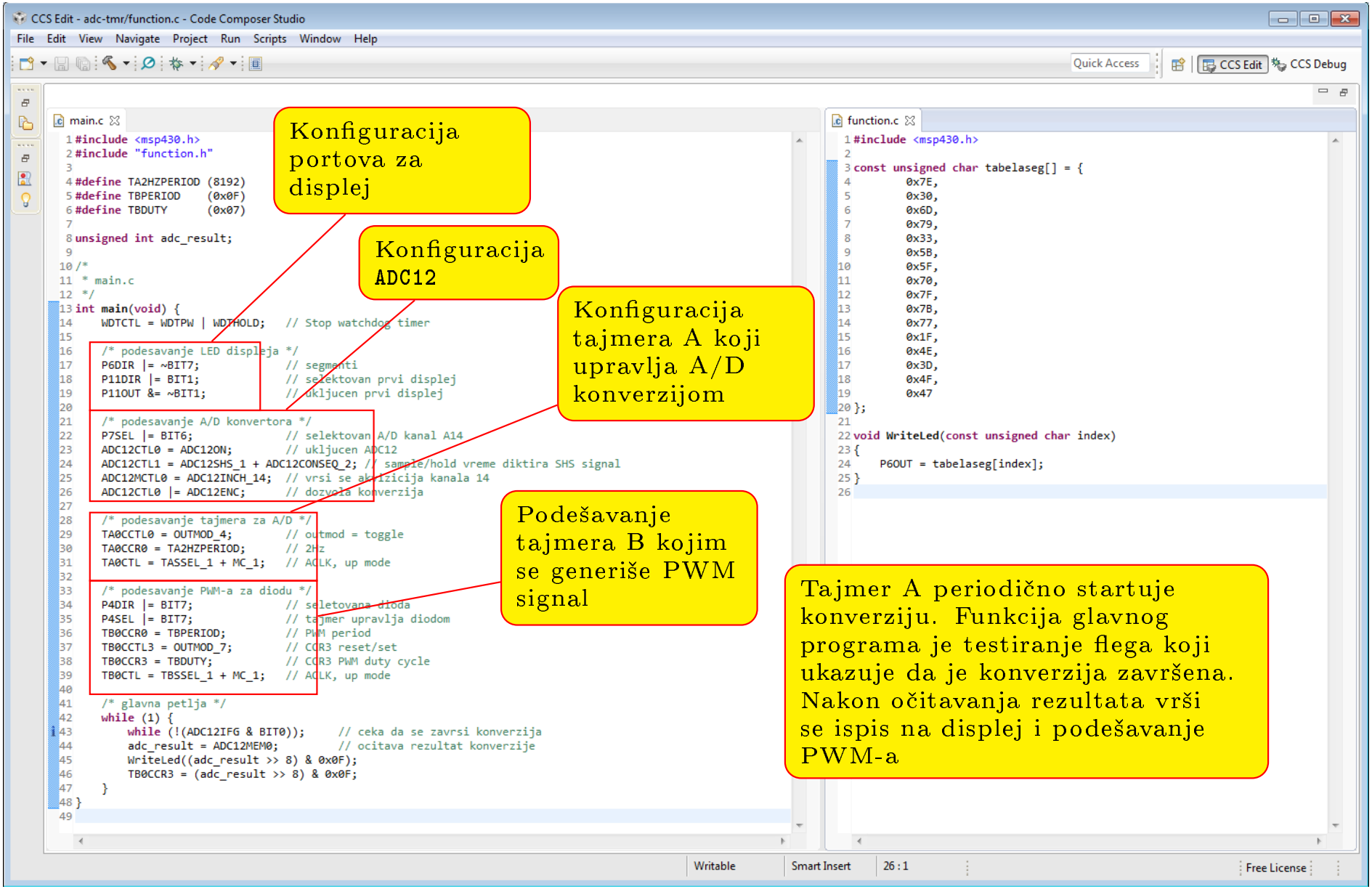
Zadatak 4 - Generisanje PWM signala



Zadatak 4 - Generisanje PWM signala



Zadatak 4 - Generisanje PWM signala




```c
#include <msp430.h>
#include "function.h"

#define TA2HZPERIOD (8192)
#define TBPERIOD    (0x0F)
#define TBDUTY      (0x07)

unsigned int adc_result;

/*
 * main.c
 */
int main(void) {
    WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

    /* podesavanje LED displeja */
    P6DIR |= ~BIT7;             // segmenti
    P11DIR |= BIT1;             // selektovan prvi displej
    P11OUT &= ~BIT1;            // ukljucen prvi displej

    /* podesavanje A/D konvertora */
    P7SEL |= BIT6;              // selektovan A/D kanal A14
    ADC12CTL0 = ADC12ON;        // ukljucen ADC12
    ADC12CTL1 = ADC12SHS_1 + ADC12CONSEQ_2; // sample/hold vreme diktira SHS signal
    ADC12MCTL0 = ADC12INCH_14;  // vrsi se akvizicija kanala 14
    ADC12CTL0 |= ADC12ENC;      // dozvola konverzija

    /* podesavanje tajmera za A/D */
    TA0CCTL0 = OUTMOD_4;        // outmod = toggle
    TA0CCR0 = TA2HZPERIOD;      // 2Hz
    TA0CTL = TASSEL_1 + MC_1;   // ACLK, up mode

    /* podesavanje PWM-a za diodu */
    P4DIR |= BIT7;              // selektovana dioda
    P4SEL |= BIT7;              // tajmer upravlja diodom
    TB0CCR0 = TBPERIOD;         // PWM period
    TB0CCTL3 = OUTMOD_7;        // CCR3 reset/set
    TB0CCR3 = TBDUTY;           // CCR3 PWM duty cycle
    TB0CTL = TBSSEL_1 + MC_1;   // ACLK, up mode

    /* glavna petlja */
    while (1) {
        while (!(ADC12IFG & BIT0));     // ceka da se zavrsi konverzija
        adc_result = ADC12MEM0;         // ocitava rezultat konverzije
        WriteLed((adc_result >> 8) & 0x0F);
        TB0CCR3 = (adc_result >> 8) & 0x0F;
    }
}
```

```c
#include <msp430.h>

const unsigned char tabelaseg[] = {
        0x7E,
        0x30,
        0x6D,
        0x79,
        0x33,
        0x5B,
        0x5F,
        0x70,
        0x7F,
        0x7B,
        0x77,
        0x1F,
        0x4E,
        0x3D,
        0x4F,
        0x47
};

void WriteLed(const unsigned char index)
{
    P6OUT = tabelaseg[index];
}
```

Zadatak 4 - Generisanje PWM signala

CCS Edit - adc-tmr/function.c - Code Composer Studio

File Edit View Navigate Project Run Scripts Window Help

main.c

```
#include <msp430.h>
#include "function.h"

#define TA2HZPERIOD (8192)
#define TBPERIOD    (0x0F)
#define TBDUTY      (0x07)

unsigned int adc_result;

/*
 * main.c
 */
int main(void) {
    WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

    /* podesavanje LED displeja */
    P6DIR |= ~BIT7;             // segmenti
    P11DIR |= BIT1;             // selektovan prvi displej
    P11OUT &= ~BIT1;            // ukljucen prvi displej

    /* podesavanje A/D konvertora */
    P7SEL |= BIT6;              // selektovan A/D kanal A14
    ADC12CTL0 = ADC12ON;        // ukljucen ADC12
    ADC12CTL1 = ADC12SHS_1 + ADC12CONSEQ_2; // sample/hold vreme diktira SHS signal
    ADC12MCTL0 = ADC12INCH_14;  // vrsi se akvizicija kanala 14
    ADC12CTL0 |= ADC12ENC;      // dozvola konverzija

    /* podesavanje tajmera za A/D */
    TA0CCTL0 = OUTMOD_4;        // outmod = toggle
    TA0CCR0 = TA2HZPERIOD;      // 2Hz
    TA0CTL = TASSEL_1 + MC_1;   // ACLK, up mode

    /* podesavanje PWM-a za diodu */
    P4DIR |= BIT7;              // seletovana dioda
    P4SEL |= BIT7;              // tajmer upravlja diodom
    TB0CCR0 = TBPERIOD;         // PWM period
    TB0CCTL3 = OUTMOD_7;        // CCR3 reset/set
    TB0CCR3 = TBDUTY;           // CCR3 PWM duty cycle
    TB0CTL = TBSSEL_1 + MC_1;   // ACLK, up mode

    /* glavna petlja */
    while (1) {
        while (!(ADC12IFG & BIT0));     // ceka da se zavrsi konverzija
        adc_result = ADC12MEM0;         // ocitava rezultat konverzije
        WriteLed((adc_result >> 8) & 0x0F);
        TB0CCR3 = (adc_result >> 8) & 0x0F;
    }
}
```

Writable

15	14	13	12	11	10	9	8
ADC12SHT1x				ADC12SHT0x			
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC12MSC	ADC12REF2_5V	ADC12REFON	ADC12ON	ADC12OVIE	ADC12TOVIE	ADC12ENC	ADC12SC
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Can be modified only when ADC12ENC = 0

Table 28-4. ADC12CTL0 Register Description

Bit	Field	Type	Reset	Description
15-12	ADC12SHT1x	RW	0h	ADC12_A sample-and-hold time. These bits define the number of ADC12CLK cycles in the sampling period for registers ADC12MEM8 to ADC12MEM15. 0000b = 4 ADC12CLK cycles 0001b = 8 ADC12CLK cycles 0010b = 16 ADC12CLK cycles 0011b = 32 ADC12CLK cycles 0100b = 64 ADC12CLK cycles 0101b = 96 ADC12CLK cycles 0110b = 128 ADC12CLK cycles 0111b = 192 ADC12CLK cycles 1000b = 256 ADC12CLK cycles 1001b = 384 ADC12CLK cycles 1010b = 512 ADC12CLK cycles 1011b = 768 ADC12CLK cycles 1100b = 1024 ADC12CLK cycles 1101b = 1024 ADC12CLK cycles 1110b = 1024 ADC12CLK cycles 1111b = 1024 ADC12CLK cycles
11-8	ADC12SHT0x	RW	0h	ADC12_A sample-and-hold time. These bits define the number of ADC12CLK cycles in the sampling period for registers ADC12MEM0 to ADC12MEM7. 0000b = 4 ADC12CLK cycles 0001b = 8 ADC12CLK cycles 0010b = 16 ADC12CLK cycles 0011b = 32 ADC12CLK cycles 0100b = 64 ADC12CLK cycles 0101b = 96 ADC12CLK cycles 0110b = 128 ADC12CLK cycles 0111b = 192 ADC12CLK cycles 1000b = 256 ADC12CLK cycles 1001b = 384 ADC12CLK cycles 1010b = 512 ADC12CLK cycles 1011b = 768 ADC12CLK cycles 1100b = 1024 ADC12CLK cycles 1101b = 1024 ADC12CLK cycles 1110b = 1024 ADC12CLK cycles 1111b = 1024 ADC12CLK cycles
7	ADC12MSC	RW	0h	ADC12_A multiple sample and conversion. Valid only for sequence or repeated modes. 0b = The sampling timer requires a rising edge of the SHI signal to trigger each sample-and-convert. 1b = The first rising edge of the SHI signal triggers the sampling timer, but further sample-and-conversions are performed automatically as soon as the prior conversion is completed.
6	ADC12REF2_5V	RW	0h	ADC12_A reference generator voltage. ADC12REFON must also be set. In devices with the REF module, this bit is only valid if the REFMSTR bit of the REF module is set to 0. In the F54xx devices (non-A), the REF module is not available. 0b = 1.5 V 1b = 2.5 V
5	ADC12REFON	RW	0h	ADC12_A reference generator on. In devices with the REF module, this bit is only valid if the REFMSTR bit of the REF module is set to 0. In the F54xx devices (non-A), the REF module is not available. 0b = Reference off 1b = Reference on
4	ADC12ON	RW	0h	ADC12_A on 0b = ADC12_A off 1b = ADC12_A on
3	ADC12OVIE	RW	0h	ADC12MEMx overflow-interrupt enable. The GIE bit must also be set to enable the interrupt. 0b = Overflow interrupt disabled 1b = Overflow interrupt enabled
2	ADC12TOVIE	RW	0h	ADC12_A conversion-time-overflow interrupt enable. The GIE bit must also be set to enable the interrupt. 0b = Conversion time overflow interrupt disabled 1b = Conversion time overflow interrupt enabled
1	ADC12ENC	RW	0h	ADC12_A enable conversion 0b = ADC12_A disabled 1b = ADC12_A enabled
0	ADC12SC	RW	0h	ADC12_A start conversion. Software-controlled sample-and-conversion start. ADC12SC and ADC12ENC may be set together with one instruction. ADC12SC is reset automatically. 0b = No sample-and-conversion-start 1b = Start sample-and-conversion

Zadatak 4 - Generisanje PWM signala

CCS Edit - adc-tmr/function.c - Code Composer Studio

File Edit View Navigate Project Run Scripts Window Help

main.c

```
1 #include <msp430.h>
2 #include "function.h"
3
4 #define TA2HZPERIOD (8192)
5 #define TBPERIOD    (0x0F)
6 #define TBDUTY      (0x07)
7
8 unsigned int adc_result;
9
10 /*
11  * main.c
12  */
13 int main(void) {
14     WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer
15
16     /* podesavanje LED displeja */
17     P6DIR |= ~BIT7;             // segmenti
18     P11DIR |= BIT1;             // selektovan prvi displej
19     P11OUT &= ~BIT1;            // ukljucen prvi displej
20
21     /* podesavanje A/D konvertora */
22     P7SEL |= BIT6;              // selektovan A/D kanal A14
23     ADC12CTL0 = ADC12ON;        // ukljucen ADC12
24     ADC12CTL1 = ADC12SHS_1 + ADC12CONSEQ_2; // sample/hold vreme diktira SHS signal
25     ADC12MCTL0 = ADC12INCH_14;  // vrsi se akvizicija kanala 14
26     ADC12CTL0 |= ADC12ENC;      // dozvola konverzija
27
28     /* podesavanje tajmera za A/D */
29     TA0CCTL0 = OUTMOD_4;        // outmod = toggle
30     TA0CCR0 = TA2HZPERIOD;      // 2Hz
31     TA0CTL = TASSEL_1 + MC_1;   // ACLK, up mode
32
33     /* podesavanje PWM-a za diodu */
34     P4DIR |= BIT7;              // seletovana dioda
35     P4SEL |= BIT7;              // tajmer upravlja diodom
36     TB0CCR0 = TBPERIOD;         // PWM period
37     TB0CCTL3 = OUTMOD_7;        // CCR3 reset/set
38     TB0CCR3 = TBDUTY;           // CCR3 PWM duty cycle
39     TB0CTL = TBSSEL_1 + MC_1;   // ACLK, up mode
40
41     /* glavna petlja */
42     while (1) {
43         while (!(ADC12IFG & BIT0));     // ceka da se zavrsi konverzija
44         adc_result = ADC12MEM0;         // ocitava rezultat konverzije
45         WriteLed((adc_result >> 8) & 0x0F);
46         TB0CCR3 = (adc_result >> 8) & 0x0F;
47     }
48 }
49
```

15	14	13	12	11	10	9	8
ADC12CSTARTADDx				ADC12SHSx		ADC12SHP	ADC12ISSH
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC12DIVx			ADC12SSELx		ADC12CONSEQx		ADC12BUSY
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)

Can be modified only when ADC12ENC = 0

Table 28-5. ADC12CTL1 Register Description

Bit	Field	Type	Reset	Description
15-12	ADC12CSTARTADDx	RW	0h	ADC12_A conversion start address. These bits select which ADC12_A conversion-memory register is used for a single conversion or for the first conversion in a sequence. The value of CSTARTADDx is 0 to 0Fh, corresponding to ADC12MEM0 to ADC12MEM15.
11-10	ADC12SHSx	RW	0h	ADC12_A sample-and-hold source select 00b = ADC12SC bit 01b = Timer source (see device-specific data sheet for exact timer and locations) 10b = Timer source (see device-specific data sheet for exact timer and locations) 11b = Timer source (see device-specific data sheet for exact timer and locations)
9	ADC12SHP	RW	0h	ADC12_A sample-and-hold pulse-mode select. This bit selects the source of the sampling signal (SAMPCON) to be either the output of the sampling timer or the sample-input signal directly. 0b = SAMPCON signal is sourced from the sample-input signal. 1b = SAMPCON signal is sourced from the sampling timer.
8	ADC12ISSH	RW	0h	ADC12_A invert signal sample-and-hold 0b = The sample-input signal is not inverted. 1b = The sample-input signal is inverted.
7-5	ADC12DIVx	RW	0h	ADC12_A clock divider 000b = Divide by 1 001b = Divide by 2 010b = Divide by 3 011b = Divide by 4 100b = Divide by 5 101b = Divide by 6 110b = Divide by 7 111b = Divide by 8
4-3	ADC12SSELx	RW	0h	ADC12_A clock source select 00b = ADC12OSC (MODCLK) 01b = ACLK 10b = MCLK 11b = SMCLK
2-1	ADC12CONSEQx	RW	0h	ADC12_A conversion sequence mode select 00b = Single-channel, single-conversion 01b = Sequence-of-channels 10b = Repeat-single-channel 11b = Repeat-sequence-of-channels
0	ADC12BUSY	R	0h	ADC12_A busy. This bit indicates an active sample or conversion operation. 0b = No operation is active. 1b = A sequence, sample, or conversion is active.

Writable | Smart Insert | 26 : 1 | Free License

Zadatak 4 - Generisanje PWM signala

CCS Edit - adc-tmr/function.c - Code Composer Studio

File Edit View Navigate Project Run Scripts Window Help

Quick Access | CCS Edit | CCS Debug

main.c

```
#include <msp430.h>
#include "function.h"

#define TA2HZPERIOD (8192)
#define TBPERIOD    (0x0F)
#define TBDUTY      (0x07)

unsigned int adc_result;

/*
 * main.c
 */
int main(void) {
    WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

    /* podesavanje LED displeja */
    P6DIR |= ~BIT7;             // segmenti
    P11DIR |= BIT1;             // selektovan prvi displej
    P11OUT &= ~BIT1;            // ukljucen prvi displej

    /* podesavanje A/D konvertora */
    P7SEL |= BIT6;              // selektovan A/D kanal A14
    ADC12CTL0 = ADC12ON;        // ukljucen ADC12
    ADC12CTL1 = ADC12SHS_1 + ADC12CONSEQ_2; // sample/hold vreme diktira SHS signal
    ADC12MCTL0 = ADC12INCH_14;  // vrsi se akvizicija kanala 14
    ADC12CTL0 |= ADC12ENC;      // dozvola konverzija

    /* podesavanje tajmera za A/D */
    TA0CCTL0 = OUTMOD_4;        // outmod = toggle
    TA0CCR0 = TA2HZPERIOD;      // 2Hz
    TA0CTL = TASSEL_1 + MC_1;   // ACLK, up mode

    /* podesavanje PWM-a za diodu */
    P4DIR |= BIT7;              // seletovana dioda
    P4SEL |= BIT7;              // tajmer upravlja diodom
    TB0CCR0 = TBPERIOD;         // PWM period
    TB0CCTL3 = OUTMOD_7;        // CCR3 reset/set
    TB0CCR3 = TBDUTY;           // CCR3 PWM duty cycle
    TB0CTL = TBSSEL_1 + MC_1;   // ACLK, up mode

    /* glavna petlja */
    while (1) {
        while (!(ADC12IFG & BIT0));     // ceka da se zavrsi konverzija
        adc_result = ADC12MEM0;         // ocitava rezultat konverzije
        WriteLed((adc_result >> 8) & 0x0F);
        TB0CCR3 = (adc_result >> 8) & 0x0F;
    }
}
```

function.c

Writable | Smart Insert | 26 : 1 | Free License

7	6	5	4	3	2	1	0
ADC12EOS	ADC12SREFx			ADC12INCHx			
rw	rw	rw	rw	rw	rw	rw	rw

Can be modified only when ADC12ENC = 0

Table 28-8. ADC12MCTLx Register Description

Bit	Field	Type	Reset	Description
7	ADC12EOS	RW	0h	End of sequence. Indicates the last conversion in a sequence. 0b = Not end of sequence 1b = End of sequence
6-4	ADC12SREFx	RW	0h	Select reference 000b = V(R+) = AVCC and V(R-) = AVSS 001b = V(R+) = VREF+ and V(R-) = AVSS 010b = V(R+) = VeREF+ and V(R-) = AVSS 011b = V(R+) = VeREF+ and V(R-) = AVSS 100b = V(R+) = AVCC and V(R-) = VREF-/VeREF- 101b = V(R+) = VREF+ and V(R-) = VREF-/VeREF- 110b = V(R+) = VeREF+ and V(R-) = VREF-/VeREF- 111b = V(R+) = VeREF+ and V(R-) = VREF-/VeREF-
3-0	ADC12INCHx	RW	0h	Input channel select 0000b = A0 0001b = A1 0010b = A2 0011b = A3 0100b = A4 0101b = A5 0110b = A6 0111b = A7 1000b = VeREF+ 1001b = VREF-/VeREF- 1010b = Temperature diode 1011b = (AVCC – AVSS) / 2 1100b = A12. On devices with the Battery Backup System, VBAT can be measured internally by the ADC. 1101b = A13 1110b = A14 1111b = A15

Zadatak 4 - Generisanje PWM signala

CCS Edit - adc-tmr/function.c - Code Composer Studio

File Edit View Navigate Project Run Scripts Window Help

Quick Access | CCS Edit | CCS Debug

main.c

```
#include <msp430.h>
#include "function.h"

#define TA2HZPERIOD (8192)
#define TBPERIOD    (0x0F)
#define TBDUTY      (0x07)

unsigned int adc_result;

/*
 * main.c
 */
int main(void) {
    WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

    /* podesavanje LED displeja */
    P6DIR |= ~BIT7;             // segmenti
    P11DIR |= BIT1;             // selektovan prvi displej
    P11OUT &= ~BIT1;            // ukljucen prvi displej

    /* podesavanje A/D konvertora */
    P7SEL |= BIT6;              // selektovan A/D kanal A14
    ADC12CTL0 = ADC12ON;        // ukljucen ADC12
    ADC12CTL1 = ADC12SHS_1 + ADC12CONSEQ_2; // sample/hold vreme diktira SHS signal
    ADC12MCTL0 = ADC12INCH_14;  // vrsi se akvizicija kanala 14
    ADC12CTL0 |= ADC12ENC;      // dozvola konverzija

    /* podesavanje tajmera za A/D */
    TA0CCTL0 = OUTMOD_4;        // outmod = toggle
    TA0CCR0 = TA2HZPERIOD;      // 2Hz
    TA0CTL = TASSEL_1 + MC_1;   // ACLK, up mode

    /* podesavanje PWM-a za diodu */
    P4DIR |= BIT7;              // seletovana dioda
    P4SEL |= BIT7;              // tajmer upravlja diodom
    TB0CCR0 = TBPERIOD;         // PWM period
    TB0CCTL3 = OUTMOD_7;        // CCR3 reset/set
    TB0CCR3 = TBDUTY;           // CCR3 PWM duty cycle
    TB0CTL = TBSSEL_1 + MC_1;   // ACLK, up mode

    /* glavna petlja */
    while (1) {
        while (!(ADC12IFG & BIT0));     // ceka da se zavrsi konverzija
        adc_result = ADC12MEM0;         // ocitava rezultat konverzije
        WriteLed((adc_result >> 8) & 0x0F);
        TB0CCR3 = (adc_result >> 8) & 0x0F;
    }
}
```

function.c

```
#include <msp430.h>
```

Writable | Smart Insert | 26 : 1 | Free License

15	14	13	12	11	10	9	8
Reserved						TASSEL	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ID		MC		Reserved	TACLR	TAIE	TAIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	w-(0)	rw-(0)	rw-(0)

Table 17-4. TAxCTL Register Description

Bit	Field	Type	Reset	Description
15-10	Reserved	RW	0h	Reserved
9-8	TASSEL	RW	0h	Timer_A clock source select 00b = TAxCLK 01b = ACLK 10b = SMCLK 11b = INCLK
7-6	ID	RW	0h	Input divider. These bits along with the TAIDEX bits select the divider for the input clock. 00b = /1 01b = /2 10b = /4 11b = /8
5-4	MC	RW	0h	Mode control. Setting MCx = 00h when Timer_A is not in use conserves power. 00b = Stop mode: Timer is halted 01b = Up mode: Timer counts up to TAxCCR0 10b = Continuous mode: Timer counts up to 0FFFFh 11b = Up/down mode: Timer counts up to TAxCCR0 then down to 0000h
3	Reserved	RW	0h	Reserved
2	TACLR	RW	0h	Timer_A clear. Setting this bit resets TAxR, the timer clock divider logic, and the count direction. The TACLR bit is automatically reset and is always read as zero.
1	TAIE	RW	0h	Timer_A interrupt enable. This bit enables the TAIFG interrupt request. 0b = Interrupt disabled 1b = Interrupt enabled
0	TAIFG	RW	0h	Timer_A interrupt flag 0b = No interrupt pending 1b = Interrupt pending

Zadatak 4 - Generisanje PWM signala

CCS Edit - adc-tmr/function.c - Code Composer Studio

File Edit View Navigate Project Run Scripts Window Help

Quick Access | CCS Edit | CCS Debug

main.c

```
1 #include <msp430.h>
2 #include "function.h"
3
4 #define TA2HZPERIOD (8192)
5 #define TBPERIOD    (0x0F)
6 #define TBDUTY      (0x07)
7
8 unsigned int adc_result;
9
10 /*
11  * main.c
12  */
13 int main(void) {
14     WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer
15
16     /* podesavanje LED displeja */
17     P6DIR |= ~BIT7;             // segmenti
18     P11DIR |= BIT1;             // selektovan prvi displej
19     P11OUT &= ~BIT1;            // ukljucen prvi displej
20
21     /* podesavanje A/D konvertora */
22     P7SEL |= BIT6;              // selektovan A/D kanal A14
23     ADC12CTL0 = ADC12ON;        // ukljucen ADC12
24     ADC12CTL1 = ADC12SHS_1 + ADC12CONSEQ_2; // sample/hold vreme diktira SHS signal
25     ADC12MCTL0 = ADC12INCH_14;  // vrsi se akvizicija kanala 14
26     ADC12CTL0 |= ADC12ENC;      // dozvola konverzija
27
28     /* podesavanje tajmera za A/D */
29     TA0CCTL0 = OUTMOD_4;        // outmod = toggle
30     TA0CCR0 = TA2HZPERIOD;      // 2Hz
31     TA0CTL = TASSEL_1 + MC_1;   // ACLK, up mode
32
33     /* podesavanje PWM-a za diodu */
34     P4DIR |= BIT7;              // seletovana dioda
35     P4SEL |= BIT7;              // tajmer upravlja diodom
36     TB0CCR0 = TBPERIOD;         // PWM period
37     TB0CCTL3 = OUTMOD_7;        // CCR3 reset/set
38     TB0CCR3 = TBDUTY;           // CCR3 PWM duty cycle
39     TB0CTL = TBSSEL_1 + MC_1;   // ACLK, up mode
40
41     /* glavna petlja */
42     while (1) {
43         while (!(ADC12IFG & BIT0));     // ceka da se zavrsi konverzija
44         adc_result = ADC12MEM0;         // ocitava rezultat konverzije
45         WriteLed((adc_result >> 8) & 0x0F);
46         TB0CCR3 = (adc_result >> 8) & 0x0F;
47     }
48 }
49
```

function.c

```
1 #include <msp430.h>
2
3 const unsigned char tabelaseg[] = {
4         0x7E,
5         0x30,
6         0x6D,
7         0x79,
8         0x33,
9         0x5B,
10        0x5F,
11        0x70,
12        0x7F,
13        0x7B,
14        0x77,
15        0x1F,
```

15	14	13	12	11	10	9	8
TAxR							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
TAxR							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 17-5. TAxR Register Description

Bit	Field	Type	Reset	Description
15-0	TAxR	RW	0h	Timer_A register. The TAxR register is the count of Timer_A.

Writable | Smart Insert | 26 : 1 | Free License

Zadatak 4 - Generisanje PWM signala

CCS Edit - adc-tmr/function.c - Code Composer Studio

File Edit View Navigate Project Run Scripts Window Help

Quick Access | CCS Edit | CCS Debug

main.c

```
1 #include <msp430.h>
2 #include "function.h"
3
4 #define TA2HZPERIOD (8192)
5 #define TBPERIOD    (0x0F)
6 #define TBDUTY      (0x07)
7
8 unsigned int adc_result;
9
10 /*
11  * main.c
12  */
13 int main(void) {
14     WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer
15
16     /* podesavanje LED displeja */
17     P6DIR |= ~BIT7;             // segmenti
18     P11DIR |= BIT1;             // selektovan prvi displej
19     P11OUT &= ~BIT1;            // ukljucen prvi displej
20
21     /* podesavanje A/D konvertora */
22     P7SEL |= BIT6;              // selektovan A/D kanal A14
23     ADC12CTL0 = ADC12ON;        // ukljucen ADC12
24     ADC12CTL1 = ADC12SHS_1 + ADC12CONSEQ_2; // sample/hold vreme diktira SHS signal
25     ADC12MCTL0 = ADC12INCH_14;  // vrsi se akvizicija kanala 14
26     ADC12CTL0 |= ADC12ENC;      // dozvola konverzija
27
28     /* podesavanje tajmera za A/D */
29     TA0CCTL0 = OUTMOD_4;        // outmod = toggle
30     TA0CCR0 = TA2HZPERIOD;      // 2Hz
31     TA0CTL = TASSEL_1 + MC_1;   // ACLK, up mode
32
33     /* podesavanje PWM-a za diodu */
34     P4DIR |= BIT7;              // seletovana dioda
35     P4SEL |= BIT7;              // tajmer upravlja diodom
36     TB0CCR0 = TBPERIOD;         // PWM period
37     TB0CCTL3 = OUTMOD_7;        // CCR3 reset/set
38     TB0CCR3 = TBDUTY;           // CCR3 PWM duty cycle
39     TB0CTL = TBSSEL_1 + MC_1;   // ACLK, up mode
40
41     /* glavna petlja */
42     while (1) {
43         while (!(ADC12IFG & BIT0));     // ceka da se zavrsi konverzija
44         adc_result = ADC12MEM0;         // ocitava rezultat konverzije
45         WriteLed((adc_result >> 8) & 0x0F);
46         TB0CCR3 = (adc_result >> 8) & 0x0F;
47     }
48 }
49
```

function.c

```
1 #include <msp430.h>
2
3 const unsigned char tabelaseg[] = {
4         0x7E,
5         0x30,
6         0x6D,
7         0x79,
8         0x33,
9         0x5B,
10        0x5F,
11        0x70,
12        0x7F,
13        0x7B,
14        0x77
```

15	14	13	12	11	10	9	8
TAxCCRn							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
TAxCCRn							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Table 17-7. TAxCCRn Register Description

Bit	Field	Type	Reset	Description
15-0	TAxCCR0	RW	0h	Compare mode: TAxCCRn holds the data for the comparison to the timer value in the Timer_A Register, TAR. Capture mode: The Timer_A Register, TAR, is copied into the TAxCCRn register when a capture is performed.

Writable | Smart Insert | 26 : 1 | Free License

Zadatak 4 - Generisanje PWM signala

CCS Edit - adc-tmr/function.c - Code Composer Studio

File Edit View Navigate Project Run Scripts Window Help

main.c

```
1 #include <msp430.h>
2 #include "function.h"
3
4 #define TA2HZPERIOD (8192)
5 #define TBPERIOD    (0x0F)
6 #define TBDUTY      (0x07)
7
8 unsigned int adc_result;
9
10 /*
11  * main.c
12  */
13 int main(void) {
14     WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer
15
16     /* podesavanje LED displeja */
17     P6DIR |= ~BIT7;             // segmenti
18     P11DIR |= BIT1;             // selektovan prvi displej
19     P11OUT &= ~BIT1;            // ukljucen prvi displej
20
21     /* podesavanje A/D konvertora */
22     P7SEL |= BIT6;              // selektovan A/D kanal A14
23     ADC12CTL0 = ADC12ON;        // ukljucen ADC12
24     ADC12CTL1 = ADC12SHS_1 + ADC12CONSEQ_2; // sample/hold vreme diktira SHS signal
25     ADC12MCTL0 = ADC12INCH_14;  // vrsi se akvizicija kanala 14
26     ADC12CTL0 |= ADC12ENC;      // dozvola konverzija
27
28     /* podesavanje tajmera za A/D */
29     TA0CCTL0 = OUTMOD_4;        // outmod = toggle
30     TA0CCR0 = TA2HZPERIOD;      // 2Hz
31     TA0CTL = TASSEL_1 + MC_1;   // ACLK, up mode
32
33     /* podesavanje PWM-a za diodu */
34     P4DIR |= BIT7;              // seletovana dioda
35     P4SEL |= BIT7;              // tajmer upravlja diodom
36     TB0CCR0 = TBPERIOD;         // PWM period
37     TB0CCTL3 = OUTMOD_7;        // CCR3 reset/set
38     TB0CCR3 = TBDUTY;           // CCR3 PWM duty cycle
39     TB0CTL = TBSSEL_1 + MC_1;   // ACLK, up mode
40
41     /* glavna petlja */
42     while (1) {
43         while (!(ADC12IFG & BIT0));     // ceka da se zavrsi konverzija
44         adc_result = ADC12MEM0;         // ocitava rezultat konverzije
45         WriteLed((adc_result >> 8) & 0x0F);
46         TB0CCR3 = (adc_result >> 8) & 0x0F;
47     }
48 }
49
```

Writable | Smart Insert | 26 : 1 | Free License

Bit	Field	Type	Reset	Description
15-14	CM	RW	0h	Capture mode 00b = No capture 01b = Capture on rising edge 10b = Capture on falling edge 11b = Capture on both rising and falling edges
13-12	CCIS	RW	0h	Capture/compare input select. These bits select the TAxCCR0 input signal. See the device-specific data sheet for specific signal connections. 00b = CCIxA 01b = CCIxB 10b = GND 11b = VCC
11	SCS	RW	0h	Synchronize capture source. This bit is used to synchronize the capture input signal with the timer clock. 0b = Asynchronous capture 1b = Synchronous capture
10	SCCI	RW	0h	Synchronized capture/compare input. The selected CCI input signal is latched with the EQUx signal and can be read via this bit.
9	Reserved	R	0h	Reserved. Reads as 0.
8	CAP	RW	0h	Capture mode 0b = Compare mode 1b = Capture mode
7-5	OUTMOD	RW	0h	Output mode. Modes 2, 3, 6, and 7 are not useful for TAxCCR0 because EQUx = EQU0. 000b = OUT bit value 001b = Set 010b = Toggle/reset 011b = Set/reset 100b = Toggle 101b = Reset 110b = Toggle/set 111b = Reset/set
4	CCIE	RW	0h	Capture/compare interrupt enable. This bit enables the interrupt request of the corresponding CCIFG flag. 0b = Interrupt disabled 1b = Interrupt enabled
3	CCI	R	0h	Capture/compare input. The selected input signal can be read by this bit.
2	OUT	RW	0h	Output. For output mode 0, this bit directly controls the state of the output. 0b = Output low 1b = Output high
1	COV	RW	0h	Capture overflow. This bit indicates a capture overflow occurred. COV must be reset with software. 0b = No capture overflow occurred 1b = Capture overflow occurred
0	CCIFG	RW	0h	Capture/compare interrupt flag 0b = No interrupt pending 1b = Interrupt pending

Zadatak 4 - Generisanje PWM signala

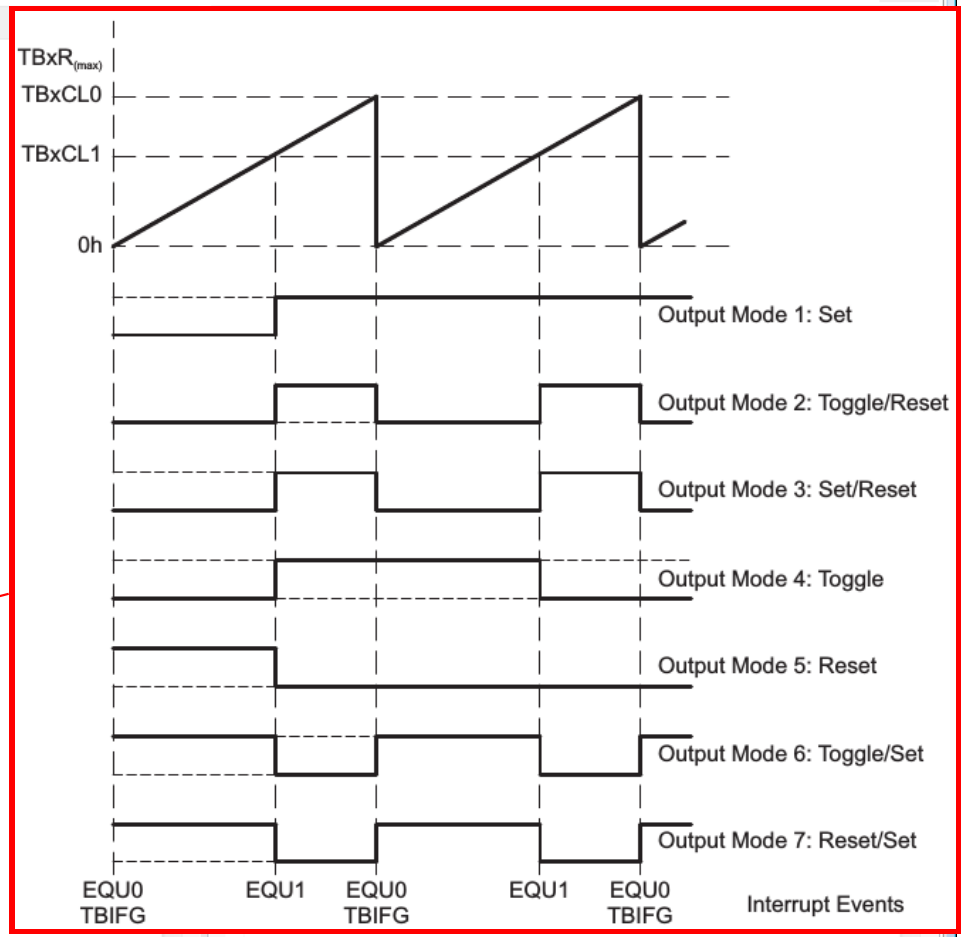
CCS Edit - adc-tmr/function.c - Code Composer Studio

File Edit View Navigate Project Run Scripts Window Help

Quick Access | CCS Edit | CCS Debug

main.c

```
1 #include <msp430.h>
2 #include "function.h"
3
4 #define TA2HZPERIOD (8192)
5 #define TBPERIOD    (0x0F)
6 #define TBDUTY      (0x07)
7
8 unsigned int adc_result;
9
10 /*
11  * main.c
12  */
13 int main(void) {
14     WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer
15
16     /* podesavanje LED displeja */
17     P6DIR |= ~BIT7;             // segmenti
18     P11DIR |= BIT1;             // selektovan prvi displej
19     P11OUT &= ~BIT1;            // ukljucen prvi displej
20
21     /* podesavanje A/D konvertora */
22     P7SEL |= BIT6;              // selektovan A/D kanal A14
23     ADC12CTL0 = ADC12ON;        // ukljucen ADC12
24     ADC12CTL1 = ADC12SHS_1 + ADC12CONSEQ_2; // sample/hold vreme diktira SHS signal
25     ADC12MCTL0 = ADC12INCH_14;  // vrsi se akvizicija kanala 14
26     ADC12CTL0 |= ADC12ENC;      // dozvola konverzija
27
28     /* podesavanje tajmera za A/D */
29     TA0CCTL0 = OUTMOD_4;        // outmod = toggle
30     TA0CCR0 = TA2HZPERIOD;      // 2Hz
31     TA0CTL = TASSEL_1 + MC_1;   // ACLK, up mode
32
33     /* podesavanje PWM-a za diodu */
34     P4DIR |= BIT7;              // seletovana dioda
35     P4SEL |= BIT7;              // tajmer upravlja diodom
36     TB0CCR0 = TBPERIOD;         // PWM period
37     TB0CCTL3 = OUTMOD_7;        // CCR3 reset/set
38     TB0CCR3 = TBDUTY;           // CCR3 PWM duty cycle
39     TB0CTL = TBSSEL_1 + MC_1;   // ACLK, up mode
40
41     /* glavna petlja */
42     while (1) {
43         while (!(ADC12IFG & BIT0));     // ceka da se zavrsi konverzija
44         adc_result = ADC12MEM0;         // ocitava rezultat konverzije
45         WriteLed((adc_result >> 8) & 0x0F);
46         TB0CCR3 = (adc_result >> 8) & 0x0F;
47     }
48 }
49
```



Writable | Smart Insert | 26 : 1 | Free License

Zadatak 5 - Generisanje PWM signala

Zadatak

Napisati program koji generiše PWM signal kod koga je trajanje impulsa srazmerno konvertovanoj vrednosti dobijenoj sa potenciometra P1. PWM signal kontroliše intenzitet osvetljaja diode LD1. Konverzija analogne vrednosti vrši se sa periodom od 0.5s.

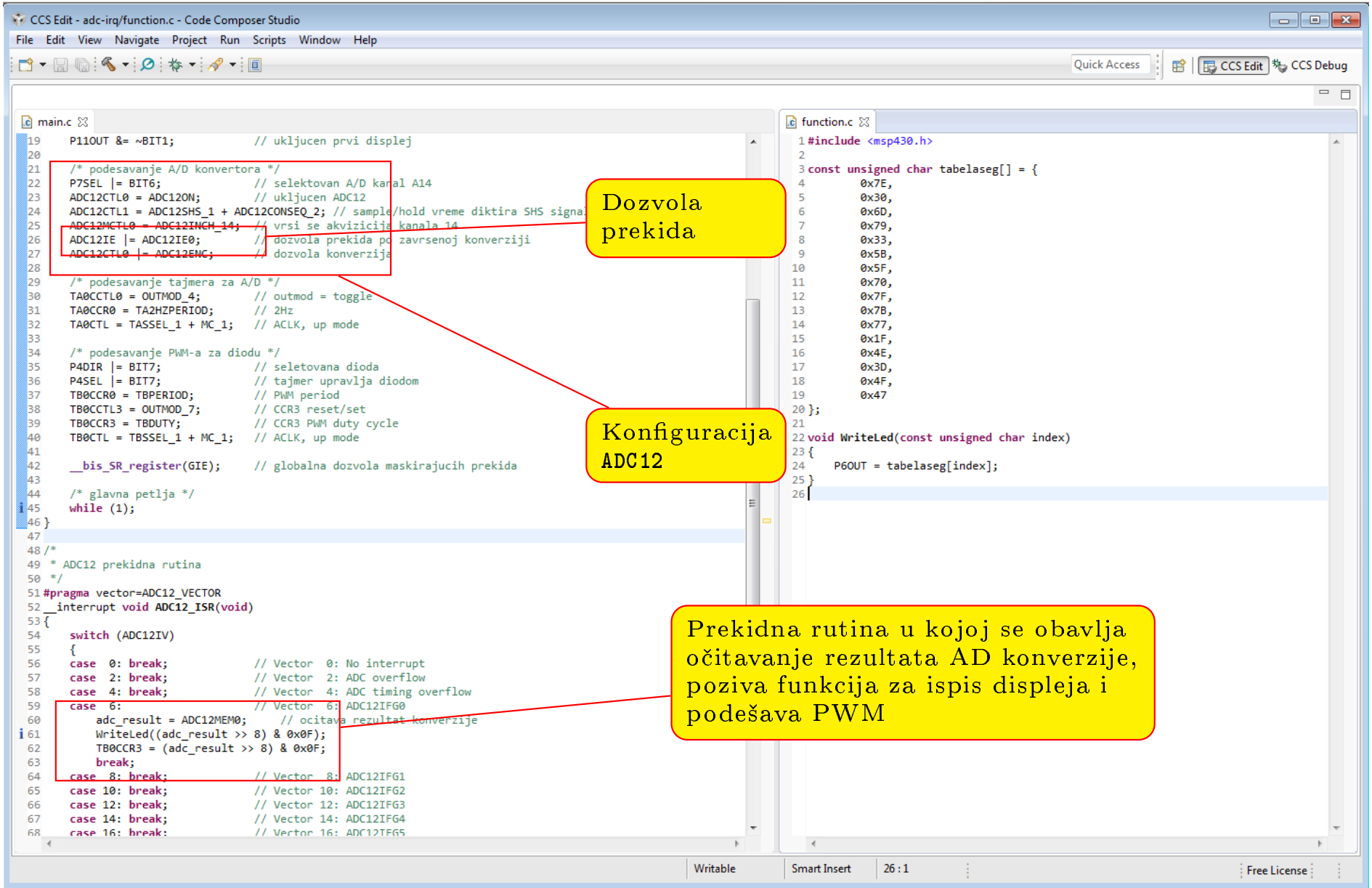
Napomena

Korstiti alternativnu funkciju na pinu koji je povezan sa diodom LD1. Zadatak realizovati korišćenjem prekida AD konvertora!

Rešenje

Videti primer *v4-z5-adc-tmr* u materijalima

Zadatak 5 - Generisanje PWM signala




```
P11OUT &= ~BIT1;            // ukljucen prvi displej

/* podesavanje A/D konvertora */
P7SEL |= BIT6;              // selektovan A/D kanal A14
ADC12CTL0 = ADC12ON;        // ukljucen ADC12
ADC12CTL1 = ADC12SHS_1 + ADC12CONSEQ_2; // sample/hold vreme diktira SHS signal
ADC12MCTL0 = ADC12INCH_14;  // vrsi se akvizicija kanala 14
ADC12IE |= ADC12IE0;        // dozvola prekida po zavrsenoj konverziji
ADC12CTL0 |= ADC12ENC;      // dozvola konverzije

/* podesavanje tajmera za A/D */
TA0CCTL0 = OUTMOD_4;        // outmod = toggle
TA0CCR0 = TA2HZPERIOD;      // 2Hz
TA0CTL = TASSEL_1 + MC_1;   // ACLK, up mode

/* podesavanje PWM-a za diodu */
P4DIR |= BIT7;              // seletovana dioda
P4SEL |= BIT7;              // tajmer upravlja diodom
TB0CCR0 = TBPERIOD;         // PWM period
TB0CCTL3 = OUTMOD_7;        // CCR3 reset/set
TB0CCR3 = TBDUTY;           // CCR3 PWM duty cycle
TB0CTL = TBSSEL_1 + MC_1;   // ACLK, up mode

__bis_SR_register(GIE);     // globalna dozvola maskirajucih prekida

/* glavna petlja */
while (1);
}

/*
 * ADC12 prekidna rutina
 */
#pragma vector=ADC12_VECTOR
__interrupt void ADC12_ISR(void)
{
    switch (ADC12IV)
    {
    case  0: break;             // Vector  0: No interrupt
    case  2: break;             // Vector  2: ADC overflow
    case  4: break;             // Vector  4: ADC timing overflow
    case  6:                    // Vector  6: ADC12IFG0
        adc_result = ADC12MEM0;     // ocitava rezultat konverzije
        WriteLed((adc_result >> 8) & 0x0F);
        TB0CCR3 = (adc_result >> 8) & 0x0F;
        break;
    case  8: break;             // Vector  8: ADC12IFG1
    case 10: break;             // Vector 10: ADC12IFG2
    case 12: break;             // Vector 12: ADC12IFG3
    case 14: break;             // Vector 14: ADC12IFG4
    case 16: break;             // Vector 16: ADC12IFG5
```

```
#include <msp430.h>

const unsigned char tabelaseg[] = {
        0x7E,
        0x30,
        0x6D,
        0x79,
        0x33,
        0x5B,
        0x5F,
        0x70,
        0x7F,
        0x7B,
        0x77,
        0x1F,
        0x4E,
        0x3D,
        0x4F,
        0x47
};

void WriteLed(const unsigned char index)
{
    P6OUT = tabelaseg[index];
}
```

Kraj četvrtog dela...