Prekidi i tajmeri

Programska podrška sistemu prekida

Kompajler AVR-GCC na pruža mogućnost upravljanja prekidima AVR familije mikrokontrolera na jednostavan način. Ovo će biti ilustrovano primerima koji koriste prekide Tajmera 0, kao i prekide izazvane promenom stanja ulaznih pinova (Pin Change Interrupt).

Za početak, potrebno je uključiti biblioteku koja podržava rad sa prekidima:

```
#include <avr/interrupt.h>
```

Da bi prekid bio omogućen, potrebno je obezbediti sledeće:

- Mora biti setovan bit za dozvolu prekida u odgovarajućem konfiguracionom registru.
- Mora biti setovan bit I u statusnom registru. U slučaju kada je I=0 podrazumeva globalnu zabranu prekida. Bit I se setuje pomoći makroa **sei()**, a resetuje pomoću makroa **cli().**
- Mora postojati prekidna rutina. Prekidne rutine su funkcije koje se automatski pozivaju prilikom pojave zahteva za prekidom, pod uslovom da je prekid dozvoljen. Prekidne rutine deklaršu se na sledeći način:

```
ISR(vektor_prekida)
{
    //telo prekidne rutine...
}
```

Kao parametar prekidne rutine navodi se vektor prekida, u zavisnosti od toga šta je izvor prekida. Mikrokontroler Atmega328P ima 25 vektora prekida + reset vektor (na adresi 0 u programskoj memoriji). Definicije vektora prekida navedene su u nastavku i dostupne su korisniku ukoliko je u okviru programa uključena biblioteka interrupt.h:

```
/* Interrupt Vectors */
/* Interrupt Vector 0 is the reset vector. */
#define INT0_vect          _VECTOR(1)   /* External Interrupt Request 0 */
#define INT1_vect          _VECTOR(2)   /* External Interrupt Request 1 */
#define PCINT0_vect        _VECTOR(3)   /* Pin Change Interrupt Request 0 */
#define PCINT1_vect        _VECTOR(4)   /* Pin Change Interrupt Request 1 */
#define PCINT2_vect        _VECTOR(5)   /* Pin Change Interrupt Request 2 */
#define WDT_vect           _VECTOR(6)   /* Watchdog Time-out Interrupt */
#define TIMER2_COMPA_vect  _VECTOR(7)   /* Timer/Counter2 Compare Match A */
#define TIMER2_COMPB_vect  _VECTOR(8)   /* Timer/Counter2 Compare Match A */
#define TIMER2_OVF_vect    _VECTOR(9)   /* Timer/Counter2 Overflow */
#define TIMER1_CAPT_vect   _VECTOR(10)  /* Timer/Counter1 Capture Event */
#define TIMER1_COMPA_vect  _VECTOR(11)  /* Timer/Counter1 Compare Match A */
#define TIMER1_COMPB_vect  _VECTOR(12)  /* Timer/Counter1 Compare Match B */
#define TIMER1_OVF_vect    _VECTOR(13)  /* Timer/Counter1 Overflow */
#define TIMER0_COMPA_vect  _VECTOR(14)  /* TimerCounter0 Compare Match A */
#define TIMER0_COMPB_vect  _VECTOR(15)  /* TimerCounter0 Compare Match B */
#define TIMER0_OVF_vect    _VECTOR(16)  /* Timer/Couner0 Overflow */
#define SPI_STC_vect       _VECTOR(17)  /* SPI Serial Transfer Complete */
#define USART_RX_vect      _VECTOR(18)  /* USART Rx Complete */
#define USART_UDRE_vect    _VECTOR(19)  /* USART, Data Register Empty */
#define USART_TX_vect      _VECTOR(20)  /* USART Tx Complete */
#define ADC_vect           _VECTOR(21)  /* ADC Conversion Complete */
#define EE_READY_vect      _VECTOR(22)  /* EEPROM Ready */
#define ANALOG_COMP_vect   _VECTOR(23)  /* Analog Comparator */
#define TWI_vect           _VECTOR(24)  /* Two-wire Serial Interface */
#define SPM_READY_vect     _VECTOR(25)  /* Store Program Memory Read */
```

Tajmer 0

Zadatak:

Podesiti tajmer 0 tako da izaziva prekid 1000 puta u sekundi (tj. jednom svake milisekunde), a zatim iskoristiti prekidnu rutinu za automatski ispis karaktera na 7SEG displeju.

Rešenje:

Za početak, potrebno je podesiti frekvenciju takta. Frekvencija oscilatora iznosi $f_{OSC} = 16MHz$, a frekvencija prekida tajmera je u ovom slučaju $f_T = 1000Hz = 1kHz$. Pošto je $f_{OSC}/f_T \neq 2^n$, potrebno je koristiti **CTC** mod tajmera, gde je moguće precizno podesiti periodu brojanja. U **CTC** modu, vrednost komparatorskog registra (**OCR0A**) računa se na sledeći način:

$$f_T = \frac{f_{osc}}{N \cdot (OCR0A + 1)}$$

$$OCR0A = \frac{f_{osc}}{N \cdot f_T} - 1 \Rightarrow N = 64, OCR0A = 249$$

```c
#include <avr/io.h>
#include <avr/interrupt.h>

const unsigned char simboli[] = {
        0x0c, 0xa4, 0x27, 0xc4
};  //look-up tabela sa simbolima
unsigned char DISP_BAFER[4] = {
        0xfe, 0xfe, 0xfe, 0xfe
};  //bafer displeja

unsigned long millis = 0;
unsigned char disp = 3;

ISR(TIMER0_COMPA_vect)
{
    //prekid tajmera 0 usled dostizanja vrednosti registra OCR0A
    if (++disp > 3)
        disp = 0;

    PORTB = ~(1 << (3-disp));   //ukljucenje tranzistora
    PORTD = DISP_BAFER[disp];   //ispis na trenutno aktivan displej
    millis++;                   //sistemsko vreme
}

int main()
{
    unsigned long t0;
    unsigned char i;

    //inicijalizacija portova:
    DDRB = 0x0f;            //PB3-PB0 -> izlazi
    DDRD = 0xff;            //port D -> izlaz

    //inicijalizacija tajmera 0:
    TCCR0A = 0x02;          //tajmer 0: CTC mod
    TCCR0B = 0x03;          //tajmer 0: fclk = fosc/64
    OCR0A = 249;            //perioda tajmera 0: 250 Tclk (OCR0A + 1 = 250)
    TIMSK0 = 0x02;          //dozvola prekida tajmera 0
                            //usled dostizanja vrednosti registra OCR0A

    sei();                  //I = 1 (dozvola prekida)

    while(1)
    {
        t0 = millis;
        while ((millis - t0) < 500);    //pauza 500ms
        for (i = 0; i < 4; i++)
            DISP_BAFER[i] = simboli[i];
        t0 = millis;
        while ((millis - t0) < 500);    //pauza 500ms
        for (i = 0; i < 4; i++)
            DISP_BAFER[i] = 0xfe;
    }
    return 0;
}
```

NAPOMENA: Da bi se program korektno kompajlirao, potrebno je isključiti optimizaciju. Opcija kojom se ovo postiže je Project -> Properties -> C/C++ Build -> Settings -> Tool Settings -> AVR Compiler -> Optimization -> Optimization Level -> No Optimizations (-O0)

Zadatak za vežbu:

Korišćenjem prekida tajmera 0, napisati program koji svake sekunde inkrementira stanje brojača i ispisuje ga na 7SEG displeju. Početno stanje brojača je 0.

Prekid usled promene na ulaznim pinovima

Zadatak:

Realizovati elektronsku kockicu na sledeći način: na 7SEG displeju D4 (krajnji displej sa desne strane) ispisuju se redom decimalne cifre 1-6, frekvencijom 25Hz. Pritiskom na taster DESNO, "obrtanje kockice" se zaustavlja i na displeju se zadržava cifra koja je bila prikazana u trenutku pritiska tastera. Proces "obrtanja" se nastavlja nakon pritiska na taster LEVO.

Rešenje:

Slično prethodnom primeru, biće korišćena prekidna rutina tajmera 0 za upravljanje displejom. U ovom slučaju moguć je stacionaran prikaz na displeju, što znači da nema potrebe za osvežavanjem u vremenskom multipleksu, pošto je displej D4 permanentno uključen. Ukoliko se tajmer 0 ponovo podesi na identičan način (prekid na 1ms), a promena stanja kockice se dešava sa periodom 1/25Hz = 40ms, to znači da je stanje kockice potrebno ažurirati jednom na svakih 40 prekida.

Logiku detekcije pritiska tastera moguće je ostvariti korišćenjem prekida izazvanog promenom stanja pinova na koje su povezani tasteri. Kod mikrokontrolera Atmega328P, postoje 3 prekida koja se generišu na ovaj način:

- PCINT0, izazvan promenom na nekom od pinova iz grupe PCINT7..PCINT0
- PCINT1, izazvan promenom na nekom od pinova iz grupe PCINT14..PCINT8
- PCINT0, izazvan promenom na nekom od pinova iz grupe PCINT23..PCINT16

Svi pinovi na koje su povezani tasteri (PC3..PC0) pripadaju grupi koja izaziva prekid PCINT1. Pošto se u aplikaciji koja realizuje elektronsku kocku koriste samo tasteri LEVO (PC0, odnosno PCINT8) i DESNO (PC2, odnosno PCINT10), u okviru registra PCMSK1 potrebno je setovati samo bite na pozicijama 0 i 2. Na taj način, prekid će biti izazvan samo pritiskom na neki od ova dva tastera, dok će ostali biti ignorisani.

Taster	Pin	PCINT ulaz
S0 (LEVO)	PC0	PCINT8
S1 (DOLE)	PC1	PCINT9
S2 (DESNO)	PC2	PCINT10
S3 (GORE)	PC3	PCINT11

```
#include <avr/io.h>
#include <avr/interrupt.h>

const unsigned char cifre[] = {
    0xdd, 0x46, 0x54, 0x9c, 0x34, 0x24
};  //cifre 1..6

unsigned char kockica = 0, brojac = 0, ukljuceno = 1;

ISR(TIMER0_COMPA_vect)
{
    //prekid tajmera 0 usled dostizanja vrednosti registra OCR0A
    if (!ukljuceno)
        brojac = 0;

    if (++brojac == 40)
    {
        brojac = 0;
        //promena stanja kockice desava se na svakih 40ms
        if (++kockica == 7)
            kockica = 1;

        PORTD = cifre[kockica - 1];
    }
}

ISR(PCINT1_vect)
{
    //prekid usled promene stanja pina PCINT10 ili pina PCINT8
    switch ((~PINC) & 0x05)
    {
        case 0x01:  //pritisnut levi taster
            ukljuceno = 1;
            break;
        case 0x04:  //pritisnut desni taster
            ukljuceno = 0;
            break;
    }
}

int main()
{
    //inicijalizacija portova:
    DDRB = 0x01;            //PB0 -> izlaz
    DDRC = 0x00;            //port C -> ulaz
    DDRD = 0xff;            //port D -> izlaz
    PORTB = ~0x01;          //ukljucenje displeja D4
```

```
    PCICR = (1 << PCIE1);    //dozvola prekida usled promene stanja
    PCMSK1 = 0x05;           //pina PCINT10, ili pina PCINT8

    //inicijalizacija tajmera 0:
    TCCR0A = 0x02;           //tajmer 0: CTC mod
    TCCR0B = 0x03;           //tajmer 0: fclk = fosc/64
    OCR0A = 249;             //perioda tajmera 0: 250 Tclk (OCR0A + 1 = 250)
    TIMSK0 = 0x02;           //dozvola prekida tajmera 0
                             //usled dostizanja vrednosti registra OCR0A

    sei();                   //I = 1 (dozvola prekida)

    while(1);                //glavni program se vrti u praznoj petlji
                             //i ceka prekide

    return 0;
}
```

Zadatak za vežbu:

Napisati program koji na 7SEG displeju ispisuje decimalni brojač, čije početno stanje je 0. Brojač se inkrementira pritiskom na taster GORE, a dekrementira pritiskom na taster DOLE. U slučajevima kad je stanje brojača negativno, omogućiti njegov prikaz dodavanjem predznaka „-“ ispred prve značajne cifre.