Univerzitet u Sarajevu
Elektrotehnički fakultet
Ugradbeni sistemi 2023 / 24

Izvještaj za laboratorijsku vježbu br. 6

Prekidi i tajmeri

Ime i prezime: **Ivona Jozić**
Broj indeks-a: **19357**

23. april 2024.

Sadržaj

1 Pseudokod i / ili dijagram toka

U pseudokodovima za laboratorijsku vježbu 6, data je ideja kako pristupiti rješavanju zadataka, a kompletnna implementacija, kako zbog svoje kompleksnosti, tako i duzine, data je kao prilog u odjeljku 5 ovog izvještaja.

1.1 Zadatak 1

```
BusOut prikaz1
BusOut prikaz2

brojac1 -> 0
brojac2 -> 0

while(1)
      brojac1 -> (brojac1 + 1) % 16
      prikaz1 -> brojac1
      sacekaj(zadani_period)

      if(taster1 pritisnut)
            brojac2 -> (brojac2 + 1) % 16
      end_if

      prikaz2 -> brojac2

end_while
```

1.2 Zadatak 2

```
BusOut prva_cifra
BusOut druga_cifra

DigitalOut generisanje
AnalogIn signal

brojac -> 0
while(1)

      generisanje -> 1
      sacekaj(1 ms)
      generisanje -> 0
      sacekaj(1 ms)

      if(signal)
            brojac -> brojac + 1
            prva_cifra -> brojac / 10
            druga_cifra -> brojac % 10
            sacekaj()
      end_if

end_while
```

1.3 Zadatak 3

```
brojac -> 0
while(1)
      if(rotacija u smjeru kazaljke)
            brojac -> brojac + 1
      else if(rotacija u_smjeru kontra kazaljke)
            brojac -> brojac – 1
      end_if

      if(taster pritisnut)
            brojac -> 0
      end_if

end_while
```

2 Analiza programskog rješenja

2.1 Zadatak 1

U prvom zadatku je bilo potrebno analizirati zadani kod koji se izvršava na razvojnom sistemu LPC1114ETF, a zatim implementirati istu funkcionalnost korištenjem sistema prekida. U zadatku su dana dva objekta BusOut koji upravljaju stanjem LED dioda. Svaki od BusOut objekata prikazuju vrijednost po jednog brojača, s tim da se brojac1 koji se prikazuje na objektu prikaz1 inkrementira za jedan na svaki period T (1s ili 2s), dok se brojac2 inkrementira za jedan samo ukoliko je pritisnut taster, uz to da se stanje tastera očitava na svakih T sekundi. Kada se za T postavi 1s sistem radi sasvim zadovoljavajuće, međutim kada se postavi T = 2s, što utiče na period očitavanja stanja tastera, pojavljuje se problem sa očitavanjem. Što se više povećava T, to je problem sa trenutkom očitavanja tastera još izraženiji.

U zadatku se tražila i implementacija ovog programa korištenjem sistema prekida, što je postignuto dodavanjem jednog objekta Ticker, Timer i InterruptIn. Objekat Ticker služi da bi se na svakih T (proizvoljno odabrani period) pozivala procedura koja vrši inkrementiranje brojac1 varijable. Objekat InteruptIn je korišten da bi se pokrenula prekidna rutina prilikom pritiska na taster, dok objekat Timer služi za otitravanje kako se zadržan pritisak na taster ne bi očitavao kao više pritisaka i kako se prekidna rutina ne bi više puta pokretala.

Oba rješenja su data u odjeljku 5 ovog izvještaja, s tim da je rješenje bez korištenja sistema prekida numerisano kao Zadatak 1a, a rješenje koje koristi sistem prekida numerisana kao Zadatak 1b.

2.2 Zadatak 2

Cilj drugog zadatka je bio istovremeno implementirati dvije funkcionalnosti na razvojnom sistemu LPC1114ETF. Jedna funkcionalnost je bila generator četvrtki na pinu dp18. Period četvrti je T=2ms. Kako period T_{ON} = ½ T, to znači da se promjena treba dešavati na svakih 1s. Druga funkcionalnost je brojač impulsa na pinu dp9, koji izbrojane ivice prikazuje u vidu dvije BCD cifre (0 – 9 prikazano na 4 bita).

2.2.1 Zadatak 2a

U dijelu pod a) je traženo da se rješenje implementira bez upotrebe sistema prekida, što je izvedeno korištenjem dva objekta BusOut koji služe za prikazivanje BCD cifara. Što se generisanja četvrtki tiče, to je realizirano u while petlji u kojoj se naizmjenično na objekat DigitalOut postavljaju vrijednosti 1 i 0 u trajanju od po 1s. Kada je riječ o signalu, nakon očitane uzlazne ivice se inkrementira brojač, a zatim ispisuje. Dok god traje uzlazna ivica, program je „zaglavljen“ u while petlji kako se brojač ne bi inkrementirao više puta.

U zavisnosti od toga kojoj funkcionalnosti je data prednost, ona druga pri frekvenciji od 500Hz počinje da kasni. Kako prioritet ima generisanje četvrtki, brojač prestaje da radi u skladu sa očekivanjima pri navedenoj frekvenciji.

Interesantno je primijetiti da prilikom korištenja objekta PWMOut umjesto DigitalOut za generisanje četvrtki i postavljenjem duty cycle na 0.5 program radi u skladu sa opisom bez obzira na visinu frekvencije, što se u rješenju sa DigitalOut objektom postiže tek prilikom

korištenja sistema prekida. Razlog za to je fizička neovisnost PWM modula od procesora. Korištenjem PWM se postiže da ove dvije funkcionalnosti budu i fizički odvojene zbog čega se postiže da mogu neovisno raditi u istom programu.

2.2.2 Zadatak 2b

U dijelu zadatka pod b) je tražena implementacija istih funkcionalnosti ali korištenjem sistema prekida. U rješenju je korištena ista logika kao i u dijelu pod a), s tim da se koriste objekat Ticker i InterruptIn. Ticker služi za vršenje izmjene na objeku DigitalOut na svakih 1s, dok je objekat InterruptIn korišten za inkrementiranje brojača pri uzlaznoj ivici signala.

Prilikom implementacije rješenja uz korištenje sistema prekida visina frekvencije ne utječe na način izvršenja programa, odnosno program radi ispravno i za jako visoke frekvencije.

2.3 Zadatak 3

U trećem zadatku je bilo potrebno spojiti rotacijski enkoder na razvojni sistem picoETF, a zatim implementirati brojač koji se inkrementira pri rotaciji enkodera u smjeru kazaljke na satu, dekrementira ukoliko se enkoder okreće u smjeru suprotno od kazaljke na satu, te se resetira u slučaju pritiska na dugme. Stanje brojača je potrebno prikazati na LED diodama (GP4 – GP11). Rješenje je implementirano korištenjem sistema prekida.

Izlazni pinovi enkodera su povezani na pinove GP0 (clk), GP1(dt), GP2(sw – taster), te na napajanje i uzemljenje. Implementirane su dvije procedure, *click()* – za resetiranje brojača prilikom pritiska na taster, i *interrupt_routine()* koja služi za inkrementiranje ili dekrementiranje brojača u zavisnosti od smjera rotacije enkodera. Ukoliko se pri silaznoj ivici signala clk desi da je signal dt na 0, to znači da se enkoder okreće u smjeru suprotnom u odnosu na kazaljku na satu i da je potrebno dekrementirati brojač. Međutim, ukoliko je vrijednost signala dt jednaka 1 na silaznoj ivici signala clk, znači da se enkoder rotira u smjeru kazaljke na satu i da je potrebno inkrementirati brojač. U rješenju je korišten sistem za obradu hardverskih prekida u MicroPhython programskom jeziku. Konkretno, korištena je metoda *irq* za povezivanje odgovarajućih procedura koje se izvršavaju u zavisnosti od pojavljivanja odgovarajuće ivice. Stanje LED dioda se ažurira u while petlji.

S obzirom da je enkoder vrlo osjetljiv na male pomake dijela koji se rotira, bilo je neophodno dodati *sleep(0.1)* u while petlji u kojoj se ažurira stanje LED dioda.

3 Korišteni hardverski resursi

Za potrebe laboratorijske vježbe 6 korišteni su razvojni sistemi picoETF, LPC1114ETF. Pored njih korištene su LED diode integrisane na razvojnim sistemima i rotacijski enkoder, te osciloskop.

Za sistem picoETF:

- 8 LED dioda

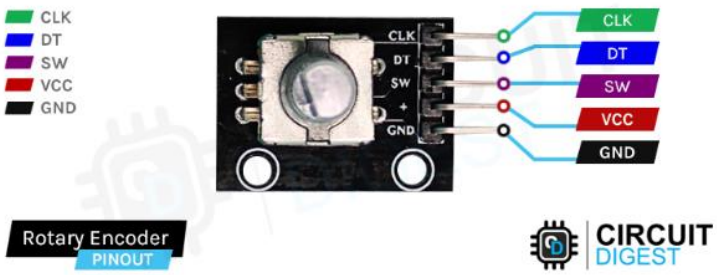
Za sistem LPC1114ETF:

- 8 LED dioda

- rotacijski enkoder

- osciloskop (korišten za ispitivanje analognog signala u zadatku 2, rađenom na LPC1114ETF)

ULAZI	IZLAZI
Rotacijski enkoder (digitalni; picoETF, GP0 (clk), GP1(dt), GP2(sw))	LED0 – LED7 (digitalni; LCP1114ETF P0_0 – P0_7)
	LED0 – LED7 (digitalni; picoETF GP4 – GP11)



Slika 1. – Izlazni pinovi rotacijskog enkodera korištenog u 3. zadatku

4 Zaključak

Prilikom izvođenja Laboratorijske vježbe 6 nije bilo poteškoća s obzirom da su oba razvojna sistema picoETF i LPC1114ETF od ranije poznata. Također nije bilo problema sa povezivanjem osciloskopa niti rotacijskog enkodera. Jedina stvar na koju je bilo potrebno dodatno obratiti pažnju to da se 2. zadatak nije mogao implementirati korištenjem objekta PWMOut. Također, zbog osjetljivosti rotacijskog enkodera u 3. zadatku bilo je neophodno dodati sleep pri ažuriranju stanja LED dioda bez obzira na to što je rješenje implementirano korištenjem sistema prekida. Cilj vježbe je bio upoznavanje sa prekidima i tajmerima, što je postignuto.

5 Prilog

U prilogu su dati kodovi koje se izvršavaju na razvojnim okruženjima LPC1114ETF (1. i 2. zadatak) i picoETF (3. zadatak). Date kodove je moguće pokrenuti i u Mbed, odnosno Wokwi simulatoru u zavisnosti od korištenog razvojnog sistema, s tim da će za testiranje u Mbed simulatoru biti potrebno izmijeniti inicijalizaciju pinova.

5.1.1 Zadatak 1a / izvorni kod

Za ispravno izvršenje sljedećeg koda neophodno je upload-ati zaglavlje *lpc1114etf.h*.

```
#include "mbed.h"
#include "lpc1114etf.h"

BusOut prikaz1(LED3, LED2, LED1, LED0);
BusOut prikaz2(LED7, LED6, LED5, LED4);
DigitalOut enable(LED_ACT);
DigitalIn taster(Taster_1);

const float T(0.2);
// const float T(2.0);
/*Za T=2.0 se promjena na led diodama desava svako 2s, ali je nedostatak to sto se stanje tastera takodjer ocitava na svako 2s i nije moguce da do izmjene dodje u kracem vremenskom periodu*/

int brojac1(0);
int brojac2(0);

int main() {
    enable=0;

    prikaz1=brojac1;
    prikaz2=brojac2;

    while(1) {
        wait_us(T*1e6);
        brojac1=(brojac1+1)%16;
        if (taster) brojac2=(brojac2+1)%16;
        prikaz1=brojac1;
        prikaz2=brojac2;
    }
}
```

5.1.2 Zadatak 1b / izvorni kod

Za ispravno izvršenje sljedećeg koda neophodno je upload-ati zaglavlje *lpc1114etf.h*.

```
#include "mbed.h"
#include "lpc1114etf.h"

BusOut prikaz1(LED3, LED2, LED1, LED0);
BusOut prikaz2(LED7, LED6, LED5, LED4);
DigitalOut enable(LED_ACT);
InterruptIn taster(Taster_1);

Ticker t;
Timer debounce;

const float T(0.2);
// const float T(2.0);

int brojac1(0);
int brojac2(0);

void prikaz_brojac1(){
    brojac1=(brojac1+1)%16;
    prikaz1=brojac1;
}

void prikaz_brojac2(){
    if(debounce.read_ms()>=200){
        brojac2=(brojac2+1)%16;
        prikaz2=brojac2;
        debounce.reset();
    }
}

int main() {
    enable=0;
    prikaz1=0;
    prikaz2=0;

    debounce.start(); //timer koji sluzi za rjesavanje otitravanja
    t.attach(&prikaz_brojac1, T); //ticker koji poziva proceduru prikaz_brojac1 svakih T
    taster.rise(&prikaz_brojac2); //pritiskom na taster se inkrementira i prikazuje brojac2

    while(1) {
        wait_us(1);
    }
}
```

5.2.1 Zadatak 2a / izvorni kod

Za ispravno izvršenje sljedećeg koda neophodno je upload-ati zaglavlje *lpc1114etf.h*.

```
#include "mbed.h"
#include "lpc1114etf.h"

DigitalOut E(LED_ACT);
BusOut prva_cifra(LED3, LED2, LED1, LED0);
BusOut druga_cifra(LED7, LED6, LED5, LED4);

DigitalOut generisani(dp18);
AnalogIn signal(dp9);

void ispisi(int &brojac){ // procedura za ispis brojaca u BCD formatu
    if(brojac>99) brojac=0;
    druga_cifra=brojac%10;
    prva_cifra=brojac/10;
}

int main() {
    E=0;
    prva_cifra=0;
    druga_cifra=0;

    int brojac=0;

    while (true) {

        generisani=1;
        wait_us(1000);
        generisani=0;
        wait_us(1000);

        if(signal.read()==1){
            brojac++;
            ispisi(brojac);
            while(signal.read()==1) continue;
        }
    }
}
//radi OK do 500HZ
/*nakon toga jedna od funkcionalnosti ne funkcionira u zavisnosti od toga koja je primarna
aktivnost*/
```

5.2.2 Zadatak 2b / izvorni kod

Za ispravno izvršenje sljedećeg koda neophodno je upload-ati zaglavlje *lpc1114etf.h*.

```
#include "mbed.h"
#include "lpc1114etf.h"

DigitalOut E(LED_ACT);
BusOut prva_cifra(LED3, LED2, LED1, LED0);
BusOut druga_cifra(LED7, LED6, LED5, LED4);

DigitalOut generisani(dp18);

InterruptIn signal(dp9);

Ticker t;

int brojac=0;

void ispisi(){ // procedura za ispis brojaca u BCD formatu
    brojac++;

    if(brojac>99) brojac=0;
    druga_cifra=brojac%10;
    prva_cifra=brojac/10;
}

void generisanje_toggle(){
    generisani = !generisani;
}

int main() {
    E=0;
    generisani=0;

    t.attach_us(&generisanje_toggle, 1000.0); // ticker koji mijenja stanje
                                              //DigitalOut na svakih 1s

    signal.rise(&ispisi); //pri svakoj uzlaznoj ivici se desava inkrementiranje brojaca

    while (true) {
        wait_ns(1);
    }
}
//bez obzira na visinu frekvencije radi ispravno
```

5.3 Zadatak 3 / izvorni kod

```
from machine import Pin
from time import sleep

sleep(0.1)

leds = [Pin(i, Pin.OUT) for i in range(4, 12)]
encoder_clk = Pin(0, Pin.IN)
encoder_dt = Pin(1, Pin.IN)
encoder_click = Pin(2, Pin.IN)

brojac = 0

def interrupt_routine(pin): #procedura koja se izvrsava pri rotaciji enkodera
    global brojac
    if encoder_dt.value() == 0:
        brojac -= 1
    else:
        brojac += 1

def click(pin): #procedura koja se izvrsava pri pritisku tastera
    global brojac
    brojac = 0

encoder_clk.irq(handler=interrupt_routine, trigger=Pin.IRQ_FALLING) #pokretanje prekidne
                                                                    #rutine pri rotaciji enkodera
encoder_click.irq(handler=click, trigger=Pin.IRQ_RISING) #pokretanje prekidne rutine pri
                                                         #pritisku na taster
while True:
    for i in range(8):
        leds[i].value((brojac>>i)&1)
    sleep(0.1)
```