Univerzitet u Sarajevu
Elektrotehnički fakultet
Ugradbeni sistemi 2023/24

Izvještaj za laboratorijsku vježbu br. 6

Prekidi i tajmeri

Ime i prezime: **Mirza Mahmutović**
Broj index-a: **19320**

23. april 2024. godine

Sadržaj

1 Pseudokod i/ili dijagram toka

1.1 Zadatak 1

```
BusOut prikaz1()
BusOut prikaz2()
InterruptIn taster()
Ticker t
Timer debounce

float T = 0.2
int brojac1 = 0, brojac2 = 0

void auto_brojac() {
      brojac1 = (brojac1 + 1) % 16
}

void taster_brojac() {
      if(debounce.read_ms() > 200) {
            brojac2 = (brojac2 + 1) % 16
            debounce.reset()
      }
}
int main() {
      tick.attach(&auto_brojac,T);
      taster.rise(&taster_brojac);
      debounce.start();
      while (true) do
            sleep()
            prikaz1 = brojac1
            prikaz2 = brojac2
      end_while
}
```

1.2 Zadatak 2

```
DigitalOut signal()
InterruptIn input()
BusOut d1()
BusOut d2()
Ticker t
int digit1 = 0, digit2 = 0
void increment() {
      digit1++
      if(digit1 == 10) then
            digit2++;
            digit1 = 0;
            if(digit2 == 10) then
                  digit2 = 0
            end_if
      end_if
      d1 = digit1
      d2 = digit2
}
void toggle() {
      signal = !signal;
}
int main() {
    input.rise(&increment);
    t.attach(&toggle,0.001);
    while (1) {}
}
```

1.3 Zadatak 3

```
leds = [Pin.OUT(),...,Pin.OUT()]
clk = Pin.IN()
dt = Pin.IN()
sw = Pin.IN()
initState = clk.value()
counter = 0
debouncing = 1
```

```
buttonPressed(pin)
    counter = 0
    updateLeds(counter)
updateLeds(x)
      for(i = 0; i < 8; i++) do
            if(x[i] == "1") then
                  leds[i].on()
            else
                  leds[i].off()
            end_if
      end_for
encoderValue(pin)
      currentState = clk.value()
      if (currentState != initState and currentState == 1) then
            if(dt.value() != currentState) then
                  counter += 1
                  if(counter > 255) then
                        counter = 0
                  end_if
            else
                  counter -= 1
                  if(counter < 0) then
                        counter = 255
                  end_if
            end_if
      end_if
      initState = currentState
      updateLeds(counter)

sw.irq(trigger=Pin.IRQ_RISING,handler=buttonPressed)
clk.irq(trigger=Pin.IRQ_RISING,handler=encoderValue)
dt.irq(trigger=Pin.IRQ_RISING,handler=encoderValue)
while(true) do
      sleep()
end_while
```

2 Analiza programskog rješenja

2.1 Zadatak 1

Postavljeni kod u prvom zadatku pokušava istovremeno obavljati dvije funkcionalnosti. Povećavanje dva brojača i prikazivanje njihovih vrijednosti na LED diodama. Pri čemu se prvi brojač automatski povećava svake 0.2s (vrijednost T*1e6), dok se drugi brojač povećava pritiskom na taster. Međutim, ovaj kod pokušava ovu funkcionalnost implementirati bez korištenja sistema prekida. U kodu se, u beskonačnoj while petlji, zbog korištenja *wait()* metode, u suštini taster na pritisak provjerava svake 0.2s. To znači da pritisak na taster neće povećavati vrijednost brojača dok je program u wait periodu. Ovaj nedostatak je posebno očigledan povećavanjem konstante T. Elegantnije i ispravnije rješenje je moguće dobiti korištenjem sistema prekida. Za rješenje ovog problema potreban je prvo InterruptIn koji će na svaku uzlaznu ivicu signala pozvati funkciju koja će povećati vrijednost drugog brojača. Međutim, radi mogućih oscilacija signala prilikom pritiska na taster, ovdje je potrebno koristiti i Timer koji će uzeti u obzir i debouncing period. Što se tiče prvog brojača, tu je dovoljno koristiti Ticker koji će na zadani vremenski period pozivati funkciju za povećavanje vrijednosti prvog brojača. Sada je moguće osloboditi beskonačnu while petlju i u njoj je dovoljno samo periodično ažurirati stanje LED dioda.

2.2 Zadatak 2

U drugom zadatku je potrebno opet istovremeno realizirati dvije funkcionalnosti. Prva jeste generator četvrtki, pri čemu period četvrtki treba biti 2ms. Druga funkcionalnost je brojač implusa dovedenih na drugi pin, pri čemu izbrojane uzlazne ivice treba prikazati kao dvije BCD cifre na LED diodama. Ovaj zadatak je također urađen na dva načina, bez i sa korištenjem sistema prekida. U pseudokodu je dato samo rješenje koje koristi sistem prekida, jer je logika u oba rješenja ista. U izvještaju je postavljen izvorni kod oba načina. U rješenju koje ne koristi sistem prekida se u while petlji mijenja vrijednost izlaznog signala (generator četvrtki), te zatim provjerava vrijednost na ulaznom pinu (brojač impulsa). Ukoliko se ustanovi da je trenutna vrijednost signala 1, a prethodna je bila 0, tada se povećava brojač i ažurira prethodna vrijednost signala. U slučaju da je vrijednost signala 0, a prethodna je bila 1, tada se samo ažurira vrijednost varijable koja čuva prethodni signal. Na kraju while petlje se brojač prikazuje na LED diodama i postavlja se wait od 1ms. Neispravno brojanje impulsa se golim okom moglo primijetiti na frekvenciji od otprilike 1kHz. Drugo, bolje rješenje je korištenjem sistema prekida. Logika mijenjanja signala i brojanja impulsa je ista kao u prvom rješenju. Ovdje su samo te dvije funkcionalnosti podijeljene u dvije funkcije, što while petlju ostavlja praznom.

Registrovanjem uzlazne ivice signala na InterruptIn se poziva funkcija koja povećava brojač impulsa, dok se na Ticker stavlja funkcija za mijenjanje vrijednosti signala.

2.3 Zadatak 3

U trećem zadatku se prvi put susrećemo sa rotacijskim enkoderom. Potrebno je koristeći rotacijski enkoder i odgovarajući sistem prekida realizirati binarni brojač naprijed i nazad. Stanje brojača je potrebno prikazati na 8 LED dioda. Klikom na enkoder, brojač se treba postaviti na 0. Na samom početku potrebno je deklarisati tri pina s kojima je enkoder određen (*sw*, *dt*, *clk*) i odgovarajuće LED diode za prikaz brojača. Pošto je zadatak potrebno riješiti koristeći sistem prekida, funkcionalnosti se dijele u funkcije, pri čemu se registrovanom uzlaznom ivicom na bilo kojem od pinova enkodera poziva odgovarajuća funkcija. Pa tako registrovanom uzlaznom ivicom na pinu *sw* se poziva funkcija *buttonPressed*, koja služi za postavljanje brojača na 0. Registrovanom uzlaznom ivicom, bilo na pinu *clk,* bilo na pinu *dt,* poziva se funkcija *encoderValue* koja služi za registrovanje promjene enkodera, te mijenjanje vrijednosti brojača na osnovu te promjene. Ukoliko je *dt* pin prvi registrovao uzlaznu ivicu onda se enkoder rotira u smjeru kazaljke na satu, dok za rotaciju suprotnu u odnosu na smjer kazaljke na satu prvo pin *clk* registruje uzlaznu ivicu. Funkcija *encoderValue*, upravo na osnovu ove logike određuje u kojem smjeru se rotira enkoder i shodno tome mijenja vrijednost brojača.

3 Korišteni hardverski resursi

Na ovoj vježbi su korišteni sljedeći razvojni sistemi:

- *LPC1114ETF,* baziran na mikrokontroleru *NXP LPC1114FN28*
- *picoETF*, baziran na mikrokontroleru *RP2040*
- rotacijski enkoder
- osciloskop
- generator signala

Na sistemu *LPC1114ETF* su za ovu vježbu korišteni sljedeći elementi:

- 8x LED dioda
- 1x taster

Na sistemu *picoETF* su za ovu vježbu korišteni sljedeći elementi:

- 8x LED dioda

4 Zaključak

Ova vježba je za cilj imala da se studenti upoznaju sa korištenjem sistema prekida. Kroz ponuđene zadatke studenti su imali priliku da bolje shvate prednosti korištenja prekida u kodu. Korištenje sistema prekida u kodovima koji trebaju pružiti više funkcionalnosti istovremeno pruža elegantiji i jednostavniji kod. Pored toga, korištenje sistema prekida u općem slučaju vodi i ka kvalitetnijem rješenju postavljenog problema. Uspješno urađenim postavljenim zadacima, cilj vježbe je i postignut.

5 Prilog

5.1 Zadatak 1/izvorni kod

```
#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);

const float T(0.2);
int brojac1(0);
int brojac2(0);
Ticker tick;
Timer debounce;

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

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

```
int main(){
    enable=0;
    tick.attach(&auto_brojac,T);
    taster.rise(&taster_brojac);
    debounce.start();
    while(1) {
        thread_sleep_for(1);
        prikaz1=brojac1;
        prikaz2=brojac2;
    }
}
```

5.2 Zadatak 2a/izvorni kod

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

DigitalOut signal(dp18);
DigitalIn input(dp9);
BusOut d1(LED0,LED1,LED2,LED3);
BusOut d2(LED4,LED5,LED6,LED7);
DigitalOut E(LED_ACT);
int main() {
    E = 0;
    int digit1 = 0, digit2 = 0, oldInput = input;
    while (1) {
        signal = !signal;

        if(input == 1 && oldInput == 0) {
            digit1++;
            if(digit1 == 10) {
                digit2++;
                digit1 = 0;
                if(digit2 == 10)
                    digit2 = 0;
            }
            oldInput = 1;
        }
```

```
        if(input == 0 && oldInput == 1)
            oldInput = 0;
        d1 = digit1;
        d2 = digit2;

        thread_sleep_for(1);
    }
}
```

5.3 Zadatak 2b/izvorni kod

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

DigitalOut signal(dp18);
Ticker t;
InterruptIn input(dp9);
BusOut d1(LED0,LED1,LED2,LED3);
BusOut d2(LED4,LED5,LED6,LED7);
DigitalOut E(LED_ACT);
int digit1 = 0, digit2 = 0;
void increment() {
    digit1++;
    if(digit1 == 10) {
        digit2++;
        digit1 = 0;
        if(digit2 == 10)
            digit2 = 0;
    }
    d1 = digit1;
    d2 = digit2;
}
void toggle() {
    signal = !signal;
}
int main() {
    E=0;
```

```
    input.rise(&increment);
    t.attach(&toggle,0.001);
    while (1) {}
}
```

5.4 Zadatak 3/izvorni kod

```
from machine import Pin, Timer
import time
time.sleep(0.1) # Wait for USB to become ready

leds = [Pin(i,Pin.OUT) for i in range(4, 12)]

clk = Pin(0,Pin.IN)
dt = Pin(1,Pin.IN)
sw = Pin(2,Pin.IN)

initState = clk.value()
counter = 0
debouncing = 1

def buttonPressed(pin):
    global counter
    counter = 0
    print("pritisnuto!")
    updateLeds('{0:08b}'.format(counter))

def updateLeds(x):
    for i in range(0,8):
        if x[i] == "1":
            leds[i].on()
        else:
            leds[i].off()

def encoderValue(pin):
    global initState,counter
    currentState = clk.value()
    if (currentState != initState and currentState == 1):
```

```
        if dt.value() != currentState:
            counter += 1
            if counter > 255:
                counter = 0
        else:
            counter -= 1
            if counter < 0:
                counter = 255
        print("Counter: ",counter)
    initState = currentState
    updateLeds('{0:08b}'.format(counter))

sw.irq(trigger=Pin.IRQ_RISING,handler=buttonPressed)
clk.irq(trigger=Pin.IRQ_RISING,handler=encoderValue)
dt.irq(trigger=Pin.IRQ_RISING,handler=encoderValue)

while True:
    time.sleep(0.02)
```