LABORATOR 11 – Tratarea tastaturii prin întreruperi

Prezentare

Programul care detectează apăsarea unei taste scris în laboratorul 7 prezintă un mare dezavantaj: dacă durata de execuție a buclei while(1) este prea mare datorită unor prelucrări laborioase, apăsările de tastele pe durata acestor prelucrări nu vor fi sesizate.

Soluția constă în tratarea tastaturii prin întreruperi. În loc să facem scanarea tastaturii în bucla while(1) o vom face atunci când timerul 0 ciclează. Ciclarea timerului 0 va genera un apel către rutina de tratare corespunzătoare iar scanarea tastaturi se va face în această rutină. Pentru a elimina impulsurile parazite generate de apăsarea sau la eliberarea unei taste timerul va cicla la aproximativ 10 ms.

Pentru acest laborator parcurgeți obligatoriu prelegerile 6, 4 și 5 (în această ordine).

Desfășurarea lucrării

Pasul 1: Program de test

Se va scrie un program care simulează o prelucrare de lungă durată. Programul afișează pe linia 1, coloana 1 doar caracterele numerice preluate de la tastatură.

Când se apasă ,#', se afișează mesajul "Calculez..." și se intră într-o buclă de așteptate cu durata de 4-5 secunde. Această comportare este întâlnită frecvent în practică: se preiau de la tastatură mai multe valori numerice și apoi cu valorile preluate se fac calcule. Calculele încep după ce se apasă Enter. Aceste calcule pot dura mult timp.

În acest laborator bucla de așteptare se va implementa cu un *for* cu corp vid. For-ul incrementează o variabila până la valoarea corespunzătoare așteptării de 4-5 secunde.

Programul are următoarea structură:

La nesfârșit

Citeste un caracter de la tastatură.

Dacă tasta apăsată **este o cifră**:

Afișează caracterul corespunzător tastei pe linia 1, coloana 1.

Dacă tasta apăsată **este** #:

Șterge linia 1 a afișajului.

Afișează mesajul "Calculez..." pe linia 1 începând cu coloana 1.

Aşteaptă în buclă 4-5 secunde.

Şterge afişajul.

Dacă tasta apăsată nu este cifra sau #:

Consuma tasta fără a face altceva.

Repetă

Mai întâi vom implementa funcționalitatea descrisă mai sus fără întreruperi.

Creați un nou proiect numit **kbdint**. Proiectul va fi compus din două module:

- kbdint.c ce conține programul principal. Acest fișier este o copie a programului principal din laboratorul 7 "Tastatura".
- IOfn.c ce conține funcțiile de afișare pe LCD și scanare a tastaturii. Acest fișier este o copie a fișierului iofn.c din laboratorul 7 "Tastatura".

Ce se întâmplă dacă pe durata afișării mesajului Calculez... se apasă o tastă numerică? Dar două? Dar trei?

Când funcționează Chemați profesorul!

Pasul 2: Iniţializare sistemului de întreruperi

În general lucru pe întreruperi presupune o succesiune de pași. Această succesiune trebuie parcursă pentru fiecare cerere de întrerupere (IRQ).

Primii pași aparțin fazei de inițializare. Pentru un IRQ oarecare pașii care trebuie urmați în faza de inițializare sunt:

- i. Se configurează blocul care generează cererea de întrerupere IRQi.
- ii. Se demaschează cererea de întrerupere IRQi.
- iii. Se demaschează întreruperea INT. (IF=1)

Deoarece scanarea tastaturii se va face în rutina de întrerupere asociată timerul 0, acest timer se va configura pentru a genera întrerupere la aproximativ 10 ms.

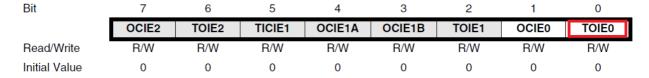
Paşii i-iii se execută o singură data, înainte de while(1). În cazul timerului 0 acești pași sunt:

i. Se configurează blocul care generează întreruperea, adică timerul 0.

Vom folosi setările timerul 0 din laboratorul 9. În laboratorul 8 timerul o a fost configurat în modul fast PWM. În acest mod numărătorul ciclează din aproximativ 8 ms în 8 ms. Ciclarea timerului provoacă setarea indicatorului TOV0 din registrul TIFR. Registrele timerului 0 sunt descrise în capitolul 14.9 din datele de catalog.

- ▶ În secțiunea de inițializări din *main* copiați setările timerului 0 din laboratorul 8.
- ii. **Se demaschează cererea de întrerupere.** În capitolul "Mascarea cererilor de întrerupere" din curs s-a precizat că orice cerere de întrerupere poate fi mascată. Bitul care maschează cererea TOV0 se află în registrul TIMSK:

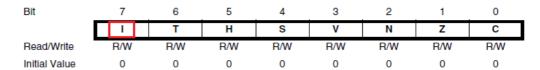
Timer/Counter Interrupt Mask Register



Daca bitul 0 – **TOIE0: Timer/Counter0 Overflow Interrupt Enable** – este 1, cererea TOV0 este activă, adică poate genera întrerupere. În caz contrar TOV0 este ignorată.

- ▶În secțiunea de inițializări din *main* **configurați TOIE0** pentru a demasca cererea de întrerupere TOV0. Documentați (scrieți un comentariu) această setare.
- iii. **Se demaschează întreruperea INT**. În subcapitolul "Întreruperi mascabile și nemascabile" din curs s-a precizat că mascarea întreruperii se face prin intermediul bistabilului IF. La Atmega16 acest bistabil se află în registrul SREG și se numește I:

SREG - AVR Status Register



Bistabilul IF din curs este bitul I din SREG.

▶În secțiunea de inițializări din *main* **configurați** I pentru a demasca întreruperea procesorului. Documentați această setare.

Pasul 3: Variabile globale

În prelegerea 4 s-a explicat că pentru a sesiza apăsarea unei taste se fac două scanări ala tastaturii cu *kbscan*(). Dacă prima scanare întoarce NOKEY, adică tastă neapăsată, iar a doua un cod diferit de NOKEY înseamnă că s-a apăsat o tastă între cele două scanări. Cele două scanări se fac la un interval mai mare decât durata instabilității. Secțiunea de program care sesizează apăsarea unei taste și generează codului acesteia a fost prezentat în prelegerea 4 și este reluat parțial mai jos:

```
#define NOKEY 0x7f
int main(){
  char code_ante = NOKEY;
  char code_now = NOKEY;
  unsigned char kbhit = 0;
  char kbcode;
  unsigned char loop_cnt=0;
  while(1){ //bucla principala
      //...
      if(loop_cnt==3){ //3 sau altă valoare
         loop cnt=0;
         code ante = code now;
         code now = kbscan();
         if( code_ante == NOKEY && code_now != NOKEY) {
            kbhit=1;
            kbcode=code_now;
         }
      }
```

Variabilele **kbhit** și **kbcode** își păstrează semnificația numai că ele vor fi scrise în rutina de întrerupere și citite în restul programului. Pentru ca acest lucru să fie posibil variabilele **kbhit** și **kbcode** vor deveni globale și vor fi utilizate după cum urmează:

- Variabila **kbhit** este setată în rutina de întrerupere dacă s-a detectat o tastă apăsată și este resetată de secțiunea de program care folosește codul tastei.
- Variabila **kbcode** conține codul tastei apăsate fiind scrisă în rutina de întrerupere și citită secțiunea de program care folosește codul tastei
- Mutați aceste variabile din *main* și declarați-le volatile în fișierul kbdint.c înainte de *main*:

```
volatile unsigned char kbhit = 0;
volatile char kbcode;
```

Pasul 4: Rutina de întrerupere

Codul de la pasul 3, cu mici modificări, va deveni rutina de întrerupere. La scrierea acestei rutine se vor executa următorii pași. Urmăriți textul scris cu roșu:

a) Rutina de întrerupere asociată lui TOV0 se va scrie în fișierul kbdint.c ce conține *main*(). Mai întâi includeți headerul interrupt.h în secțiunea de *include* cu:

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

b) Rutina de întrerupere poate fi scrisă înainte sau după *main*. Din motive de claritate a codului o vom scrie după main(). Declarația rutina de întrerupere cu:

```
ISR(TIMER0_OVF_vect){
```

ISR este definit în interrupt.h iar TIMER0_OVF_vect în io.h.

c) Variabilele code_ante şi code_now declarate până acum în *main* vor deveni variabile locale în ISR. Mutați declararea acestor variabile din *main* în ISR.

Atenție: variabilele locale nu își păstrează valoare între apeluri. Detecția apăsării prin compararea a două valorii succesive întoarse de *kbscan* presupune memorarea valorii întoarse la apelul curent. **Modificați** declarația variabilei locale code_now astfel încât aceasta să-și păstreze valoarea între apeluri.

- d) În codul de la pasul 3 scanarea tastaturii se face din 3 în 3 execuții ale buclei principale. Valoarea 3 se stabilește prin încercări astfel încât scanarea să se facă la interval de aproximativ 10 ms. În varianta cu scanarea în rutina de întreruperi intervalul dintre scanări este dat de ciclarea trimerului 0. Din acest motiv variabila loop_cnt și tot ceea ce ține de aceasta se șterge.
- e) Codul de la pasul 3 care are legătură cu detecția apăsării se mută din *main* în rutina de tratare:

```
code_ante = code_now;
code_now = kbscan();
if( code_ante == NOKEY && code_now != NOKEY){
   kbhit=1;
   kbcode=code_now;
}
```

f) Terminați rutina de întrerupere:

```
}//end ISR
```

Testați! Dacă tasta apăsată pe durata afișării mesajului Calculez... se afișează pe LCD după trecerea perioadei de asteptare, chemați profesorul pentru validare. Dacă ați ajuns aici aveți nota 5.

Partea opțională

Pasul 5: LED cu intensitate variabilă

La fel ca în laboratorul 10 se dorește controlul luminozității unui LED. Timerul 0 a fost programat în modul fast PWM pentru a putea controla luminozitatea unui LED. Pentru aceasta procedați după cum urmează.

- a) Conectați un LED pe pinul OC0.
- b) Creați un nou proiect. Acest proiect este o copie a proiectului **kbdint** din acest laborator.

c) Setați biții COM01 și COM00 și programați OC0 la fel ca în laboratorul 10.

Variați intensitatea LED-ului prin intermediul tastelor C și D la fel ca în laboratorul 10. Nu afișați valoarea intensității. Luminozitatea trebui să asculte de tastele C și D și pe durata afișării mesajului "Calculez..."

Funcționalitatea anterioară, de la pasul 4, trebuie să se păstreze.

Dacă funcționează, chemați profesorul pentru validare.

Dacă ați ajuns aici aveți între 1 și 3 puncte

Pasul 6: Afișarea intensității

Afișați intensitatea LED-ului la fel ca în laboratorul 10. Atenție la reentranța funcțiilor folosite în rutina de întrerupere.

Dacă funcționează, chemați profesorul pentru validare.

Dacă ați ajuns aici aveți între 1 și 2 puncte.