LABORATOR 11 – Tratarea tastaturii prin întreruperi

Prezentare

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

Soluţia 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 parcurgeţi obligatoriu prelegerile 6, 4 şi 5 (în această ordine).

Desfăşurarea lucrării

Pasul 1: Program de test

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

Când se apasă ‚#', se afişează mesajul "**Calculez…**" şi se intră într-o buclă de aşteptate cu durata de 4-5 secunde. Această comportare este întâlnită frecvent în practică: se preiau de la tastatură mai multe valori numerice şi 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 aşteptare se va implementa cu un *for* cu corp vid. For-ul incrementează o variabila până la valoarea corespunzătoare aşteptării de 4-5 secunde.

Programul are următoarea structură:

La nesfârşit
    Citeşte un caracter de la tastatură.
        Dacă tasta apăsată **este o cifră**:
            Afişează caracterul corespunzător tastei pe linia 1, coloana 1.
        Dacă tasta apăsată **este #**:
            Şterge linia 1 a afişajului.
            Afişează 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 funcţionalitatea descrisă mai sus fără întreruperi.

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

- kbdint.c ce conţine programul principal. Acest fişier este o copie a programului principal din laboratorul 7 – „Tastatura”.
- IOfn.c ce conţine funcţiile de afişare pe LCD şi scanare a tastaturii. Acest fişier este o copie a fişierului iofn.c din laboratorul 7 – „Tastatura”.

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

Când funcţionează Chemaţi profesorul!

Pasul 2: Iniţializare sistemului de întreruperi

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

Primii paşi aparţin fazei de iniţializare. Pentru un IRQ oarecare paşii care trebuie urmaţi în faza de iniţializare sunt:

i. Se configurează blocul care generează cererea de întrerupere - IRQ*i*.

ii. Se demaschează cererea de întrerupere IRQ*i*.

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 aceşti paşi 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 secţiunea de iniţializări din *main* **copiaţi** 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

Bit	7	6	5	4	3	2	1	0
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

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 secţiunea de iniţializări din *main* **configuraţi TOIE0** pentru a demasca cererea de întrerupere TOV0. Documentaţi (scrieți un comentariu) această setare.

iii. **Se demaschează întreruperea INT**. În subcapitolul „Întreruperi mascabile şi nemascabile” din curs s-a precizat că mascarea întreruperii se face prin intermediul bistabilului IF. La Atmega16 acest bistabil se află în registrul SREG şi se numeşte I:

SREG – AVR Status Register

Bit	7	6	5	4	3	2	1	0
	I	T	H	S	V	N	Z	C
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bistabilul IF din curs este bitul I din SREG.

►În secțiunea de inițializări din *main* **configuraţi I** pentru a demasca întreruperea procesorului. Documentaţi 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 instabilităţii. Secţiunea de program care sesizează apăsarea unei taste şi generează codului acesteia a fost prezentat în prelegerea 4 şi este reluat parţial 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** şi **kbcode** îşi păstrează semnificaţia numai că ele vor fi scrise în rutina de întrerupere şi citite în restul programului. Pentru ca acest lucru să fie posibil variabilele **kbhit** şi **kbcode** vor deveni globale şi vor fi utilizate după cum urmează:

- Variabila **kbhit** este setată în rutina de întrerupere dacă s-a detectat o tastă apăsată şi este resetată de secțiunea de program care foloseşte codul tastei.
- Variabila **kbcode** conţine codul tastei apăsate fiind scrisă în rutina de întrerupere şi citită secțiunea de program care foloseşte codul tastei

►**Mutaţi aceste variabile** din *main* şi declaraţi-le volatile în fişierul 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 paşi. Urmăriţi textul scris cu roşu:

a) Rutina de întrerupere asociată lui TOV0 se va scrie în fişierul kbdint.c ce conţine *main*(). Mai întâi includeţi headerul interrupt.h în secţiunea 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(). Declaraţia 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. **Mutaţi declararea acestor variabile din *main* în ISR.**

Atenţie: variabilele locale nu îşi păstrează valoare între apeluri. Detecţia apăsării prin compararea a două valorii succesive întoarse de *kbscan* presupune memorarea valorii întoarse la apelul curent. **Modificaţi** declaraţia variabilei locale `code_now` astfel încât aceasta să-şi păstreze valoarea între apeluri.

d) În codul de la pasul 3 scanarea tastaturii se face din 3 în 3 execuţii ale buclei principale. Valoarea 3 se stabileşte 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` şi tot ceea ce ţine de aceasta se şterge**.

e) **Codul de la pasul 3 care are legătură cu detecţia 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) Terminaţi rutina de întrerupere:

```
}//end ISR
```

Testaţi! Dacă tasta apăsată pe durata afişării mesajului **Calculez…** se afişează pe LCD după trecerea perioadei de aşteptare, chemaţi profesorul pentru validare. Dacă aţi ajuns aici aveţi nota 5.

Partea opţională

Pasul 5: LED cu intensitate variabilă

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

a) Conectaţi un LED pe pinul OC0.

b) Creaţi un nou proiect. Acest proiect este o copie a proiectului **kbdint** din acest laborator.

c) Setaţi biţii COM01 şi COM00 şi programaţi OC0 la fel ca în laboratorul 10.

Variaţi intensitatea LED-ului prin intermediul tastelor C şi D la fel ca în laboratorul 10. **Nu afişaţi valoarea intensităţii. Luminozitatea trebui să asculte de tastele C şi D şi pe durata afişării mesajului „Calculez…”**

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

Dacă funcţionează, chemaţi profesorul pentru validare.

Dacă aţi ajuns aici aveţi între 1 şi 3 puncte

Pasul 6: Afişarea intensităţii

Afişaţi intensitatea LED-ului la fel ca în laboratorul 10. Atenţie la reentranţa funcţiilor folosite în rutina de întrerupere.

Dacă funcţionează, chemaţi profesorul pentru validare.

Dacă aţi ajuns aici aveţi între 1 şi 2 puncte.