

LABORATOR 9 – Ceas digital. Utilizarea timerelor

Se cere implementarea unui **ceas digital** care afișează timpului cu o precizie ce depinde numai de stabilitatea frecvenței semnalului de ceas al microcontrolerul ATmega.

Se reamintește că unitatea de comandă a oricărui procesor trece dintr-o stare în altă la apariția frontului activ al semnalului de ceas. În cazul microcontrolerelor din familia ATmega sunt disponibile trei opțiuni privitor la generarea ceasului:

1. semnalul de ceas este generat **extern**
2. semnalul de ceas este generat **intern** de un oscilator RC.
3. semnalul de ceas este generat **intern** de un amplificator, dar sunt necesare **componente externe** adiționale. Amplificatorul intern împreună cu componentele externe formează un oscilator.

Oscilatorul RC intern, generează o frecvență fixă de 8 MHz. Această variantă este foarte ieftină și simplă deoarece nu necesită nici o componentă externă, dar are dezavantajul că frecvența ceasului generat are o precizie mică și în plus variază mult cu temperatura. Acest fapt poate fi acceptat în multe aplicații dar este **intolerabil** în aplicația **ceas digital**. Din acest motiv în acest laborator **ar trebui** să folosim vom folosi varianta 3, adică generare internă cu componente externe suplimentare.

În varianta 3 componenta externă suplimentară este un **cristal de cuarț** deoarece oscilatoarele cu cristal de cuarț oferă performanțe foarte bune la un preț de cost mic. Aproape toate ceasurile digitale folosesc această variantă de oscilator. Pentru a înțelege ce înseamnă performanțe bune, vom trece în revistă doi parametri ai oscilatoarelor cu cristal de cuarț.

Pentru oscilatoarele cu cuarț se definesc 4 parametri, dintre care doi sunt mai importanți:

Precizia frecvenței: $P = \text{frecvența măsurată}^{1)} / \text{frecvența înscrisă pe cuarț}$

¹⁾ măsurarea se face la 25°C, presiune de 756mmHg

Pentru oscilatoarele cu cuarț precizia este de ordinul 20 ppm (părți per milion) sau mai bună. Frecvența generată de cuarț depinde de dimensiunile sale fizice obținute prin tăiere, în procesul de fabricație. Diferența între frecvența reală și cea înscrisă se datorează nerepetitivității procesului de fabricație. De exemplu, dacă un cristal este folosit pentru ceas, 20ppm înseamnă într-o lună o deviație de:

$$60 \frac{s}{min} \times 60 \frac{min}{h} \times 24 \frac{h}{zi} \times 30 \frac{zile}{luna} \times \left(\pm \frac{20}{1000000} \right) = \pm 51.84 \frac{s}{luna}$$

Stabilitatea frecvenței

Orice dispozitiv electronic este afectat de temperatură. Și frecvența generată de oscilatoarele cu cuarț va varia cu temperatura:

$$S = \text{frecvența generată la temperatura } t^{\circ}\text{C} / \text{frecvența generată la } (t-1)^{\circ}\text{C}$$

Tipic, S are valoarea de 1 ppm/ °C. Pentru a corecta deviația frecvenței cu temperatura în aplicațiile de mare precizie, oscilatoarele se închid în incinte termostatare.

Din păcate simulatorul **simulIDE nu are implementată selecția semnalului de ceas** ci doar setarea frecvenței acestuia. Frecvența setată în simulIDE are performanța variantei cu cristal de cuarț deoarece simulatorul se bazează pe ceasul PC-ului iar acesta este generat cu cristal de cuarț.

În concluzie în simulare dispunem de cel mai performant generator de ceas!

Prezentarea Metodei

Varianta soft.

În prelegerea 5, capitolul 1, s-a arăta că **este aproape imposibilă măsurarea timpului prin mijloace pur software. Dacă cineva crede că este totuși posibil, să citească acest subcapitol.** În caz contrar treceți la subcapitolul următor!

Teoretic, o primă variantă pentru a determina când a trecut o secundă se bazează pe faptul că orice instrucțiune este executată de procesor într-un număr fix de perioade de ceas. Această soluție este implementată în software. O variantă a acestei metode conține o buclă de întârziere soft, de tipul

```
for (i=0; i<n; i++) { }
```

, întârzierea depinzând de valoarea lui n :

Sec=0;

La nesfârșit

P1. for(i=0; i<n; i++) { }

P2. Sec++;

P3. Dacă sec==60 sec=0;

P4. Afișează sec;

Repetă.

Pentru ca descrierea de mai sus să fie cât mai simplă s-au luat în considerare numai calculul și afișarea secundelor. Afișarea precisă a secundelor presupune ca buclă *La nesfârșitRepetă* să se execute în exact o secundă, indiferent dacă $sec==0$ sau nu. Pentru ca metoda să funcționeze trebuie ca suma timpilor de execuție pentru toate instrucțiunile cod mașină din care este alcătuită buclă să fie exact o secundă.

Un prim aspect de care trebuie de care trebuie ținut seama este ca **timpul de execuție să nu se schimbă de la o iterație la alta a buclei**. De obicei această condiție nu este îndeplinită. De exemplu pasul *P3: Dacă sec==60 sec=0;* implică timpi de execuție diferiți, după cum variabila sec este sau nu egală cu 60. Dacă $sec==60$ se execută $sec=0$ iar dacă nu se trece direct la pasul următor. Astfel de structuri trebuie echilibrate astfel:

P3: Dacă sec==60 sec=0; altfel așteaptă un timp egal cu timpul necesar să execuți sec=0;.

Un alt aspect care îngreunează calculul timpului de execuție al buclei este necesitatea de a ști timpii de execuție pentru toate funcțiile din bibliotecă apelate. De exemplu, dacă în buclă se apelează *sprintf(...)* pentru a genera stringul de afișat trebuie să știm timpul său de execuție. Acest timp este aproape imposibil de aflat.

CONCLUZIE: în general este imposibil sau extrem de greu și cu siguranță inefficient să controlăm riguros timpul prin intermediul buclelor soft.

Varianta hard/soft (CO-DESIGN)

Problema ar putea fi rezolvată mult mai ușor dacă se folosește un **numărător hardware**. În momentul în care numărătorul ciclează (overflows) se setează un bistabil. **Fie numele acestui bistabil TF** (timer flag).

Trebuie subliniat că:

- Numărătorul nu se oprește niciodată. Acesta numără modulo N adică 0, 1, 2,...N-1, 0, 1, 2,... N-1, 0, 1, 2,... N-1.... **Valoarea lui N se va discuta ulterior.**
- Atingerea valorii N-1 setează bistabilul TF (Timer Flag).
- Software-ul poate citi, seta sau reseta TF.

Bazându-ne pe numărător și pe bistabilul TF „Ceas digital” se poate implementa parte în hardware, parte în software astfel:

HARWARE	SOFTWARE
Numărător	TF
0	0 { Așteaptă până când TF devine ,1’
1	0 .
2	0 .
.	.
N-1	0 .
0	1 }
1	1 { Resetează TF.
2	1 .
3	0 }
4	0 { Calculează timpul
5	0 .
6	0 .
.	.
1245	0 }
1246	0 { Dacă este cazul, afișează timpul
1247	0 .
.	.
31236	0 }
31237	0 { Așteaptă până când TF devine ,1’
31238	0 .
31239	0 .
.	.
N-1	0 .
0	1 }
1	1 { Resetează TF.
2	1 .
3	0 }
4	0 { Calculează timpul
5	0 .
6	0 .
.	.

În modelul de mai sus valorile numărătorului au rol de exemplificare.

Această modalitate de implementarea aplicațiilor, în care o parte din funcțiuni sunt implementate software, iar cealaltă parte hardware, se numește CO-DESIGN. Relația dintre e partea de hardware și software a aplicației este prezentată în figura următoare:

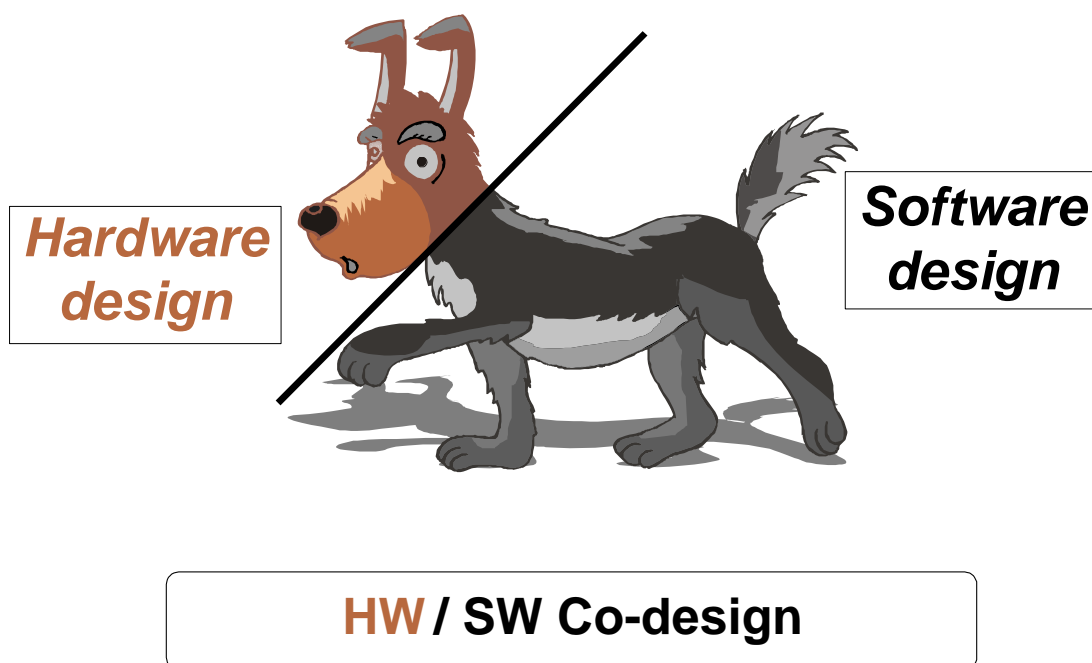


figura 1

Atenție: Metoda hard/soft, expusă mai sus, este de tip polling. Pentru ca polling-ul să funcționeze este necesar ca operațiile software:

1. Resetează TF.
2. Calculează timpul.
3. Afișează timpul.

să se execute într-un timp mai scurt decât un ciclu de numărare. Dacă aceste acțiuni durează mai mult decât un ciclu de numărare, setarea TF nu va fi sesizată și ceasul va rămâne în urmă. În acest caz trebuie folosită metoda întreruperii.

Particularități de implementare

Pentru a trece la implementare este necesar să **parcurgeți documentația** necesară pentru înțelegerea modului de funcționare a timerelor 0 și 2. Informațiile necesare se află în prelegerea 5 și în fișierul **ATmega16A.pdf** disponibil la

https://drive.google.com/file/d/1loup5AoqPNgRn_eNs9udbx0o9Kb8AqD/view

începând de la pagina 71 până la pagina 135. Din prelegerea 5 parcurgeți obligatoriu capitolele 2 și 3, **în special subcapitolul 3.4.**

Atenție: în documentație termenul folosit pentru numărător este timer/counter.

SCOPUL LUCRĂRII

Se cere:

1. **Afișarea pe LCD a timpului** în formatul **hh:mm:ss**. Orele se vor afișa în format 24 fără zerourile nesemnificative. Exemplu: se va afișa 5:03:40, nu 05:03:40..
2. La apăsarea oricărei taste **se va emite un sunet scurt** de confirmare.

Pentru implementarea ceasului digital vom folosi timerul 2. Acest timer va fi folosit pentru implementarea ceasului digital conform modelului din prelegere 5, capitolul 3.4. Față de implementarea din curs diferă timerul folosit și frecvența ceasului microcontrolerului.

Se dă $T_r=1s$ și $f_{CLK_CPU} = 8\text{ MHz}$.

După modelul din curs **faceți calculele pentru determinarea lui p , N și k** pentru a obține un T_{cycle} cât mai mare.

După ce ați determinat p , N , k și T_{cycle} chemați profesorul pentru prima validare intermediară.

Desfășurarea lucrării

Pasul 1: Crearea montajului

Se va folosi același circuit simulIDE ca în laboratorul 7 – „Tastatura”. Creați o copie și numiți-o **ceas.simu**.

Pasul 2: Crearea proiectului

Se va crea un proiect nou cu numele **ceas** care va conține aceleași surse ca proiectul „kbd” din laboratorul 7. Pentru crearea acestui proiect procedați după cum urmează:

- Se va crea** proiectul cu numele **ceas**.
- Copiați** **kbd.c**, **LCD.c** și **LCD.h** din folderul **kbd** în folderul proiectului **ceas**.
- Adăugați** la proiect fișierele copiate la pasul b)
- Eliminați fișierul** **main.c** din proiect.
- Redenumiți** **kbd.c** ca **ceas.c**.

Pasul 3: Setarea modului și a valorii p

În secțiunea de inițializare a programului trebuie să setăm timerul 2 astfel încât să numere modulo N . Modul în care orice numărător ATmega numără modulo N este modul CTC. Valoarea N se înscrie în registrul OCR. De asemenea trebuie programat factorul de scalare p . N și p au fost calculate anterior.

Modul de funcționare al numărătorului 2 și factorul de scalare p se setează prin intermediul registrului TCCR2. Detalii despre acest registru cât și despre celelalte registre care controlează numărătorul 0 se găsesc în fișierul **ATmega16A Data Sheet** de la pagina 130 și în prelegerea 5.

Mai întâi vom seta biții WGM21 și WGM20 pentru ca numărătorul 2 să fie în mod CTC. Setările se fac conform tabelului 17-2, reprodus în continuare:

Bit	7	6	5	4	3	2	1	0	
	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	TCCR2
Read/Write	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	

Table 17-2. Waveform Generation Mode Bit Description⁽¹⁾

Mode	WGM21 (CTC2)	WGM20 (PWM2)	Timer/Counter Mode of Operation	TOP	Update of OCR2	TOV2 Flag Set on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR2	Immediate	MAX
3	1	1	Fast PWM	0xFF	BOTTOM	MAX

Atenție: biții WGM21 și WGM20 apar în TCCR2 în ordine inversă față de tabelul 17-2.

În continuare se programează valoarea lui p calculată anterior. Setările se fac conform tabelului 17-6, reprodus în continuare:

7	6	5	4	3	2	1	0	
FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	TCCR2
W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

Table 17-6. Clock Select Bit Description

CS22	CS21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$\text{clk}_{T2S}/(\text{No prescaling})$
0	1	0	$\text{clk}_{T2S}/8$ (From prescaler)
0	1	1	$\text{clk}_{T2S}/32$ (From prescaler)
1	0	0	$\text{clk}_{T2S}/64$ (From prescaler)
1	0	1	$\text{clk}_{T2S}/128$ (From prescaler)
1	1	0	$\text{clk}_{T2S}/256$ (From prescaler)
1	1	1	$\text{clk}_{T2S}/1024$ (From prescaler)

Biții COM21 și COM20 se programează astfel încât OC2 să își schimbe valoarea la fiecare ciclu al număratorului 2 (Toggle Mode). În acest fel pe pinul OC2 generează un semnal periodic cu perioada $2T_{\text{cycle}}$. Acest semnal poate fi folosit dacă pe OC2 se adaugă un difuzor cu condiția ca $T_{\text{OC2}}=2T_{\text{cycle}}$ să fie în plaja de frecvențe pe care urechea umană le poate auzi (20Hz – 20 KHz).

7	6	5	4	3	2	1	0	
FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	TCCR2

Table 17-3. Compare Output Mode, non-PWM Mode

COM21	COM20	Description
0	0	Normal port operation, OC2 disconnected.
0	1	Toggle OC2 on compare match
1	0	Clear OC2 on compare match
1	1	Set OC2 on compare match

Bitul FOC2 nu se programează și va rămâne la valoarea default (zero).

Orice numărator ATmega provoacă setarea indicatorul TF. **Indicatorul TF nu există ca atare**, el este un nume generic. Număratorului 2 are asociați doi indicatori: TOV2 și OCF2. TOV2 - Timer Overflow Flag – este setat imediat după ce numărătorul a trecut din starea 0xFF în starea 0x00 iar OCF2 – este setat imediat după ce starea numărătorului a fost egală cu conținutul registrului OCR2. Aceștia doi indicatori sunt disponibili prin intermediul registrului TIFR, al cărui conținut este reprodus mai jos.

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
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	

Mai multe informații despre acest registru se găsesc în datele de catalog, la pagina 85.

Care dintre cei doi indicator este TF? Ce indicator indică ciclarea Timer/counter-ului 2 în mod CTC?

Pasul 4: Scrierea programului principal

Timerul 2 va fi folosit în mod **asemănător** cu timerul 0 din implementarea ceasului digital din prelegerea 5, capitolul 3.4 „Ceas Digital cu timerul 0”. Pentru implementare aveți de nevoie de:

- p , N , k determinat anterior.
- Să știți care din indicatorii TOV2 sau OCF2 este indicatorul TF.

La ceas.c **se va adăuga** codul pentru ceas. Codul care trebuie adăugat **este scris cu verde** mai jos.

Deoarece diferențele între timerul 0 și timerul 2 sunt foarte mici modificarea codului din curs este foarte simplă:

```
//define și prototipuri
int main(){
    //declarații variabile pentru tastatură - deja există

    unsigned char cycles=0;
    // declarații alte variabile pentru ceas

    LCDinit();

    DDRA = 0;
    PORTA = 0;

    // setarea TCCR2 se face EXACT ca mai jos. La examen se cere acest stil.
    // înlocuiți ? cu valorile calculate
    //          ┌───CTC mode
    TCCR2=0b???????;
    //          ┌───P=?
    //          └───OC2 ?
    OCR2=?;

    while(1){
        //codul pentru tastatura.

        if( testbit(TIFR, ?) ){
            setbit(TIFR, ?);
            cycles++;

            if(cycles == ?){ k=?
                // a trecut o secundă
                cycles=0;
                // actualizează secunde, minute, ore
                // afișează timpul

                //Verifica încadrarea in timp:
                if( testbit(TIFR, ?) ){
                    Afișează mesaj de eroare și intră în buclă infinită.
                }
            }
        }
        //...
    } //end while
} //end main
```

Atenție:

1. Modificați funcționarea tastaturii astfel încât caracterul preluat de la tastatură să se afișeze întotdeauna pe linia 1, coloana 1. Folosiți gotoLC(1, 1);

2. Ștergerea indicatorului de ciclare din registrul TIFR se face **scriind un ,1'** în poziția corespunzătoare indicatorului ce se vrea șters.
3. Așa cum s-a precizat anterior, bucla principală trebuie să se execute într-un timp mai mic decât timpul de ciclare al numărătorului. Operația care necesită cel mai mult timp este afișarea timpului. Afișarea unui caracter necesită $43\mu s$ iar ștergerea LCD $1,53\text{ ms}$. Cum scrierea timpului înseamnă scrierea a 8 caractere, în total afișarea timpului necesită $1,53\text{ms} + 8 \cdot 43\mu s \approx 2\text{ ms}$.

Pentru a scăpa de ștergerea LCD-ului folosiți suprascrierea; în loc de `clrLCD + putsLCD/putchLCD` **folosiți `gotoLC + putsLCD/putchLCD`**.

4. Pentru afișarea timpului vom folosi, `sprintf`: astfel:

```
#include <stdio.h>
...
char buf[17];
...
sprintf(buf, "%2d:%02d:%02d", ore, minute, secunde);
gotoLC(2, 1);
putsLCD(buf);
....
```

În loc de `ore, minute, secunde` puteți folosi alți identificatori.

5. Dacă afișarea plus toate celelalte prelucrări din bucla `while(1)` se execută în mai mult de T_{cycle} , programul nu va funcționa corect. Pentru a fi siguri că această condiție este îndeplinită, după afișarea timpului se va testa indicatorul de ciclare: dacă valoarea acestuia este ,1' actualizarea și afișare timpului au consumat prea mult timp, numărătorul a ciclat și ceasul va rămâne în urmă.

Pasul 5: Testare

Testați conform următoarelor proceduri:

- Tastatura trebuie să funcționeze.
- Setati timpul la 23:59:50 și rulați. Verificați că tranziția 23:59:59 → 0:00:00 se face corect.
- Faceți ca timpul afișat pe LCD să fie egal cu ceasul din Windows. Mai întâi setati timpul inițial în avans cu aproximativ un min față de timpul calculatorului. Timpul inițial trebuie să aibă secunde egale cu zero. Dați comanda Run când timpul setat în program și ceasul Windows devin egali. Se admite o diferență de o secundă. De exemplu dacă ceasul Windows este 16:43:13 setati timpul inițial la 16: 44:00. **Imediat** după ce ați pornit simularea, **chemați profesorul**. Profesorul va nota timpul la care ați început testul.
- Lăsați să ruleze cel puțin 5 minute. În acest interval nu executați nici un alt program, nu atingeți tastatura sau mouse-ul.
- Dacă după cele 5 minute nu există diferențe între timpul afișat pe LCD și timpul PC, **chemați profesorul pentru a doua validare intermediară**.

Pasul 6: Adăugare difuzor.

Mai întâi modificați LCD.c. În codul de mai jos


```
void initLCD(){
    PORTD = 0;
    DDRD = 0xff;
    _delay_ms(1000);
    ...
}
```

înlocuiți

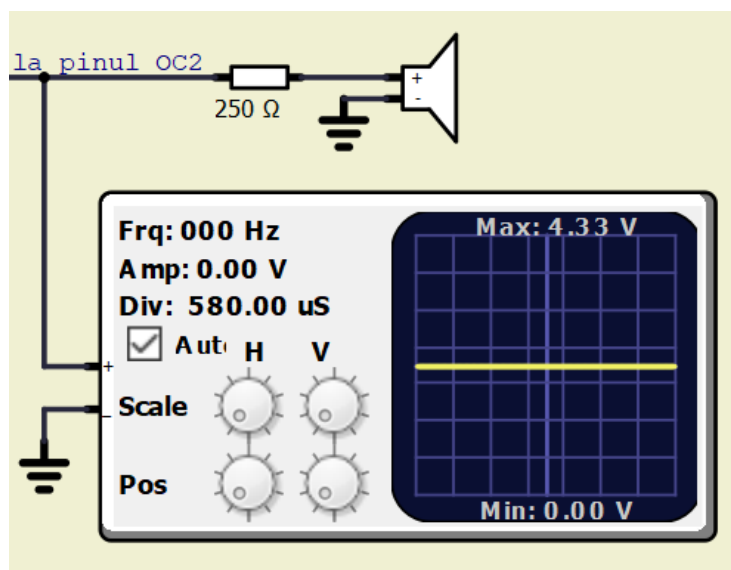
```
DDRD = 0xff;
```

cu

```
DDRD = 0x3f;
```

În continuare vom conecta componenta **Audio Out** la pinul OC2 al microcontrolerului. Componenta Atmega16 din simulIDE nu afișează toate funcțiile unui pin al microcontrolerului ci doar numele de bază, cel corespunzător funcției de intrare/ieșire digitală. Numele complet al pinilor microcontrolerului este prezentat în figura 1-1 de la pagina 2 din documentație. Aflați numele de bază la pinului OC2.

Tot la pinul OC2 conectați un osciloscop, ca în figura următoare:



Pentru ca pinul OC2 să se comporte conform programării din TCCR2 acesta trebuie configurat ca ieșire.

În secțiunea de inițializare a timerului configurați bitul DDR corespunzător pentru ca OC2 să fie ieșire.

Dacă totul funcționează corect ar trebui să auziți un sunet continuu iar frecvența acestui sunet să fie $f_{OC2} = \frac{1}{2T_{cycle}}$. Calculați f_{OC2} și verificați că osciloscopul afișează această frecvență.

În final **eliminați** setarea care programa pe OC2 ca ieșire. Nu dorim ca să avem sunet permanent în difuzor. Acest pas a fost necesar pentru a verifica funcționarea corectă a modulului de generare a formei de undă.

Pasul 7: Beep la apăsare tastă.

Se cere ca la apăsarea unei taste să se emită un sunet cu durata de 100 ms. Metoda de implementare a acestei activități este de tipul „Acțiune cu durată fixă”. Un exemplu de astfel de acțiune este pregătirea unei porții de mititei de către bucătarul unei grădini de vară. Bucătarul așteaptă comanda, pune mititeii pe grătar, memorează timpul când a pus mititeii pe grătar și apoi se uită periodic la ceas făcând diferența între timpul curent și timpul la care a început prăjirea. Când această diferență devine mai mare decât timpul de prăjire din rețeta de mititei, de exemplu 5 minute, oprește prăjirea ridicând mititei de pe grătar.

Pentru a implementa o astfel de acțiune este nevoie de trei elemente:

1. O modalitate de **măsurare a timpului**: Ceasul de la mâna bucătarului în exemplu precedent.

2. **Pornirea acțiunii** la apariția unui eveniment plus **memorarea timpului la care a început acțiunea**.

Amprenta de timp este valoarea indicată de un ceas la un moment dat. În exemplu cu mititeii primirea comenzii este evenimentul care declanșează acțiunea, acțiunea este prăjirea mititeilor iar amprenta de timp este timpul la care mititeii au fost puși pe grătar.

3. **Oprirea acțiunii** după scurgerea unui interval fix de timp. Trecerea intervalului de timp se determină prin diferența între timpul curent și amprenta de timp.

În cazul emiterii unui sunet la apăsarea unei taste cele trei elemente se particularizează astfel:

1. **O modalitate de măsurare a timpului.** Măsurarea timpului se poate face numai hardware cu un numărător așa cum este explicat în prelegere 5.1, 5.2 și 5.3 sau combinat, hardware +software așa cum am procedat la implementarea ceasului în acest laborator.

Deoarece avem deja implementarea pentru ceas vom adăuga o variabilă de tip `unsigned int` numită `ms?` care se va incrementa cu `?` ms la fiecare ciclare a timerului 2. `?` este durata în milisecunde a lui `Tcycle` calculată la începutul acestei lucrări. Restul implementării nu se schimbă:

```
...
if( testbit(TIFR, ?) ){
    setbit(TIFR, ?);
    cycles++;
    ms?+=?;

    if(cycles == ?){ k=?
        // a trecut o secundă
        cycles=0;
    }
}
...
```

2. **Pornirea acțiunii la apariția unui eveniment.** Evenimentul care activează acțiunea este apăsare unei taste iar acțiunea este activarea sunetului. În acest moment trebuie luată o amprentă de timp. O tastă a fost apăsată când `kbhit = 1`;

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

    setbit(DDR?, ?); // setează OC2 ca ieșire pentru a activa sunetul
                    // Această operației a mai fost făcută la pasul 6

    ts = ms?;        // ia o amprentă de timp

    stop_activ = 1;  // activează oprire sunet. Definește această variabilă
                    // și inițializează-o cu zero.
}
} // end if(code_now...
```

3. **Oprirea acțiunii după scurgerea unui interval fix de timp.** Dacă oprirea sunetului este activă și a trecut intervalul fix de timp, 100 ms în acest caz, oprește sunetul:

```
// in while(1) loop, after // end if(code_now...
if(stop_activ și ms? - ts > 100 ){
    clrbit(DDR?, ?); // setează OC2 ca intrare pentru a opri sunetul
    stop_activ = 0;
}
```

Dacă funcționează se va auzi un sunet scurt la apăsarea unei taste, oricare ar fi ea. Dacă tasta este menținută apăsată NU se va auzi nimic.

Dacă funcționează chemați profesorul pentru validarea FINALĂ. 😊