LABORATOR 9 – Ceas digital. Utilizarea timerelor

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

Se reaminteşte că unitatea de comandă a oricărui procesor trece dintr-o stare în altă la apariţia frontului activ al semnalului de ceas. În cazul microcontrolerelor din familia ATmega sunt disponibile trei opţiuni 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** adiţionale. Amplificatorul intern împreună cu componentele externe formează un oscilator.

Oscilatorul RC intern, generează o frecvență fixă de 8 MHz. Această variantă este foarte ieftină şi simplă deoarece nu necesită nici o componentă externă, dar are dezavantajul că frecvenţă ceasului generat are o precizie mică şi în plus variază mult cu temperatura. Acest fapt poate fi acceptat în multe aplicaţii dar este **intolerabil** în aplicaţia **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 cuarţ** deoarece oscilatoarele cu cristal de cuarţ oferă performanţe foarte bune la un preţ de cost mic. Aproape toate ceasurile digitale folosesc această variantă de oscilator. Pentru a înțelege ce înseamnă performanțe bune, vom trece in revistă doi parametrii ai oscilatoarelor cu cristal de cuarţ.

Pentru oscilatoarele cu cuarţ se definesc 4 parametrii, dintre care doi sunt mai importanţi:

Precizia frecvenţei: P = frecvenţa măsurată[1)] / frecvenţa înscrisă pe cuarţ

[1)] măsurarea se face la 25^0C, presiune de 756mmHg

Pentru oscilatoarele cu cuarţ precizia este de ordinul 20 ppm (părţi per milion) sau mai bună. Frecvenţa generată de cuarţ depinde de dimensiunile sale fizice obţinute prin tăiere, în procesul de fabricaţie. Diferenţa între frecvenţa reală şi cea înscrisă se datorează nerepetitivităţii procesului de fabricaţie. De exemplu, dacă un cristal este folosit pentru ceas, 20ppm înseamnă într-o lună o deviaţie 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 frecvenţei

Orice dispozitiv electronic este afectat de temperatură. Şi frecvenţa generată de oscilatoarele cu cuarţ va varia cu temperatura:

S = frecvenţa generată la temperatura t^0C / frecvenţa generată la $(t-1)^0$C

Tipic, S are valoarea de 1 ppm/ ^{0}C. Pentru a corecta deviaţia frecvenţei cu temperatura în aplicaţiile de mare precizie, oscilatoarele se închid în incinte termostatate.

Din păcate simulatorul **simulIDE nu are implementată selecţia 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 timpul prim 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 instrucţiune este executată de procesor într-un număr fix de perioade de ceas. Această soluţie este implementată in software. O variantă a acestei metode conţine o bucla de întârziere soft, de tipul

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

, întârzierea depinzând de valoarea lui *n*:

Sec=0;

La nesfârşit

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

P2. Sec++;

P3. Dacă sec==60 sec=0;

P4. Afişează sec;

Repetă.

Pentru ca descrierea de mai sus să fie cât mai simplă s-au luat în considerare numai calculul şi afişarea secundelor. Afişarea precisă a secundelor presupune ca bucla *La nesfârşit .....Repetă* să se execute în exact o secundă, indiferent dacă sec==0 sau nu. Pentru ca metoda să funcţioneze trebuie ca suma timpilor de execuţie pentru toate instrucţiunile cod maşină din care este alcătuită bucla să fie exact o secundă.

Un prim aspect de care trebuie de care trebuie ţinut seama este ca timpul de execuţie să nu se schimbă de la o iteraţie la alta a buclei. De obicei această condiţie nu este îndeplinită. De exemplu pasul P3: *Dacă sec==60 sec=0;* implică timpi de execuţie diferiţi, 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 aşteptă un timp egal cu timpul necesar să execuţi sec=0;*.

Un alt aspect care îngreunează calculul timpului de execuţie al buclei este necesitatea de a şti timpii de execuţie pentru toate funcţiile din biblioteci apelate. De exemplu, dacă în buclă se apelează *sprintf*(…) pentru a o genera stringul de afişat trebuie să ştim timpul său de execuţie. Acest timp este aproape imposibil de aflat.

CONCLUZIE: în general este imposibil sau extrem de greu şi cu siguranţă ineficient să controlăm riguros timpul prin intermediul buclelor soft.

Varianta hard/soft (CO-DESIGN)

Problema ar putea fi rezolvată mult mai uşor dacă se foloseşte 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 opreşte 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 (**T**imer **F**lag).
- Software-ul poate citi, seta sau reseta TF.

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

HARWARE		**SOFTWARE**
Numărător	**TF**	
0	0	**{** Aşteaptă 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, afişează timpul
1247	0	.
.	.	
31236	0	**}**
31237	0	**{** Aşteaptă 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 aplicaţiilor, în care o parte din funcţiuni sunt implementate software, iar cealaltă parte hardware, se numeşte CO-DESIGN. Relaţia dintre e partea de hardware şi software a aplicaţiei este prezentată în figura următoare:

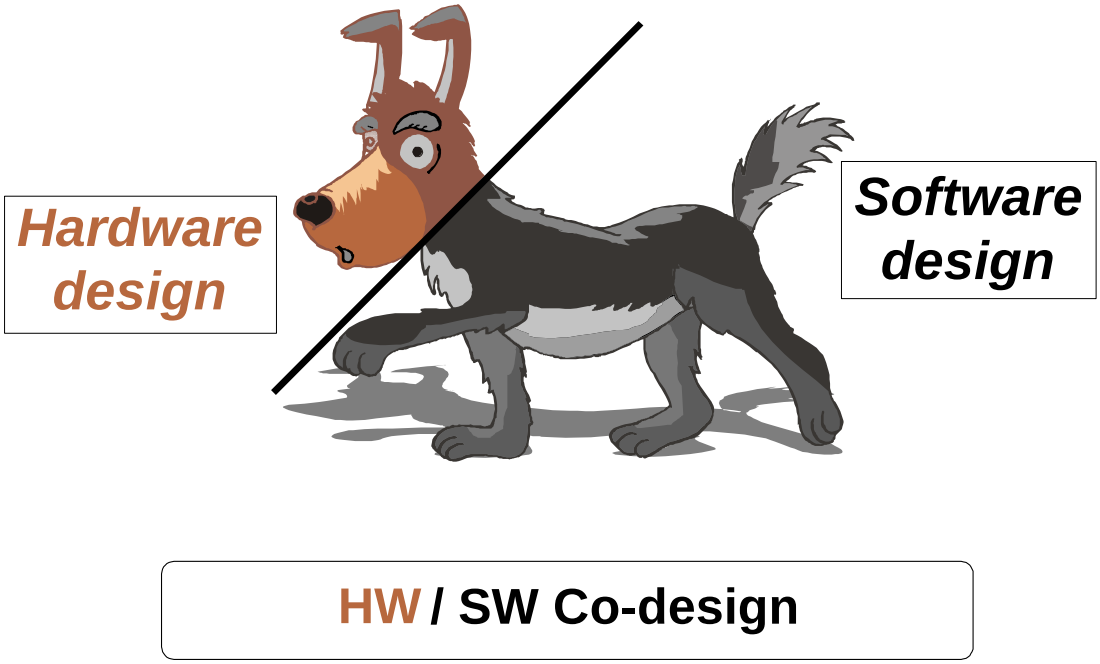

figura 1

Atenţie: 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.

Particularităţi de implementare

Pentru a trece la implementare este necesar să parcurgeţi documentaţia necesară pentru înţelegerea modului de funcţionare a timerelor 0, 1 şi 2. Informaţiile necesare se află în prelegerea 5 şi în fişierul **ATmega16A.pdf** disponibil la

https://sites.google.com/edu.ucv.ro/lemeni/documentatiedatasheet?authuser=0

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

Atenţie: în documentaţie termenul folosit pentru numărător este timer/counter.

SCOPUL LUCRĂRII

Se cere afişarea pe LCD a timpului în formatul hh:mm:ss. Orele se vor afişa în format 24 fără zerourile nesemnificative. Exemplu: se va afişa 5:03:40, nu 05:03:40..

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.

Se dă $\mathbf{T_r=1s}$ şi $\boldsymbol{f_{CLK\_CPU}}$ **= 8 MHz.**

După modelul din curs **faceți calculele pentru determinarea lui *p*, N şi k** pentru a obţine un $\mathbf{T_{cycle}}$ cât mai mare.

După ce aţi determinat *p*, N , *k* şi T_{cycle} chemați profesorul pentru validare.

Desfăşurarea lucrării

Pasul 1: Crearea montajului

Se va folosi același circuit simulIDE ca in laboratorul 7 –„Tastatura”

Pasul 2: Crearea proiectului

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

a) **Se va crea** proiectul cu numele **ceas**.

b) **Copiaţi kbd.c**, **IOfn.c** şi **defs.h** din folderul **kbd** în folderul proiectului **ceas**.

c) **Adăugați** la proiect fișierele copiate la pasul b.

d) **Eliminați fișierul main.c** din proiect.

e) **Redenumiți kbd.c** ca **ceas.c**.

Pasul 3: Setarea modului și a valorii *p*

În secțiunea de iniţializare 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 şi *p* au fost calculate anterior.

Modul de funcţionare al numărătorului 2 şi factorul de scalare *p* se setează prin intermediul registrului TCCR2. Detalii despre acest registru cât şi despre celelalte registre care controlează numărătorul 0 se găsesc în fişierul **ATMega16A Data Sheet** de la pagina 130 şi în prelegerea 5.

Mai întâi vom seta biţii WGM21 şi 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: biţii WGM21 şi WGM20 apar în TCCR2 în ordine inversă faţa de tabelul 17-2.

In 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	clk_{T2S}/(No prescaling)
0	1	0	clk_{T2S}/8 (From prescaler)
0	1	1	clk_{T2S}/32 (From prescaler)
1	0	0	clk_{T2S}/64 (From prescaler)
1	0	1	clk_{T2S}/128 (From prescaler)
1	1	0	clk_{T2S}/256 (From prescaler)
1	1	1	clk_{T2S}/1024 (From prescaler)

Biții FOC2, COM21 şi COM20 deocamdată nu se programează şi vor rămâne la valorile default (zero).

Orice numărător ATMega provoacă setarea indicatorul TF. **Indicatorul TF nu există ca atare**, el este un nume generic. Numărătorul 2 are asociaţi doi indicatori: TOV2 şi 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 conţinutul 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 indică ciclarea Timer/counterului 2 configurat ca numărător modulo N?

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 ştiți care din indicatorii TOV2 sau OCF2 este indicatorul TF.

La ceas.c (o copie a lui kbd.c) **se va adăuga** codul pentru ceas.

Programul principal va avea structura prezentată în curs şi detaliată în continuare:

```
//include şi prototipuri
int main(){
  //declaraţii variabile pentru tastatură – deja există

  unsigned int cycles=0;
  // alte declaraţii variabile pentru ceas
  //iniţializări pentru ceas

  sysinit();
  //Foarte important: după sysinit() și înainte de while(1):
```

```
    // configurează Timer 2 (modul, CTC, p, N) prin intermediul lui TCCR2
    // şi OCR2;

  while(1){
    Codul pentru tastatură. NU modifica!

    Dacă TF == ,1'{
      P1. Resetează TF;
      P2. cycles++;
      P3. Dacă cycles == k{
      P4.    cycles=0;
      P5.    Calculează timpul în secunde, minute, ore;
      P6.    Afişează timpul pe linia 2, începând cu coloana 1;
      P7.
            //Verifica încadrarea in timp:
      P8.    Dacă TF == 1{
      P9.       Afişează mesaj de eroare şi intră în buclă infinită.
         }
       }
    }
  }//end while
}//end main
```

Atenţie:

1. Modificaţi funcţionarea tastaturii astfel încât caracterul preluat de la tastatură să se afişeze întotdeauna pe linia 1, coloana 1. Folosiți gotoLC(1, 1);
2. **TF nu poate fi testat cu construcţii de genul TF==1.** Testarea trebuie să se facă aşa cum este explicat în laboratorul 3, în paragraful „Testarea unui bit într-un cuvânt".
3. Ştergerea flagului TF din registrul TIFR se face **scriind un ‚1'** in poziţia corespunzătoare flagului ce se vrea şters.
4. Aşa cum s-a precizat anterior, paşii P1-P6 trebuie să se execute într-un timp mai mic decât timpul de ciclare al numărătorului. Operația care necesită cel mai mult timp este afişarea timpului de la pasul P7. Afişarea unui caracter necesită 43µs iar ştergerea LCD 1,53 ms. Cum scrierea timpului înseamnă scrierea a 8 caractere, în total afişarea timpului necesită 1,53ms+8*43µs ≈ 2 ms.

   Pentru a scăpa de ştergerea LCD-ului folosiţi suprascrierea; în loc de clrLCD+putsLCD/putchLCD **folosiţi gotoLC+putsLCD/putchLCD**.
5. Pentru afişarea de la P6 vom folosi, **doar pentru testare**, sprintf: astfel:

   ```
   #include <stdio.h>
   ...
   char buf[17];
   ...
   sprintf(buf, "%2d:%02d:%02d", nume_ore, nume_minute, nume_secunde);
   putsLCD(buf);
   ....
   ```

   `nume_ore`, `nume_minute`, `nume_secunde` sunt numele variabilelor pentru ore, minute și secunde ce diferă de la echipă la echipă.
6. Dacă afișarea plus toate celelalte prelucrări din bucla `while(1)` se execută în mai mult de T_{cycle}, programul nu va funcţiona corect. Pentru a fi siguri că această condiţie este îndeplinită, după

afişarea timpului se va testa TF: dacă valoarea acestuia este ‚1' înseamnă ca paşii P1-P6 au consumat prea mult timp, numărătorul a ciclat şi programul nu va funcţiona (ceasul va rămâne în urmă).

Pasul 5: Testare

Testaţi conform următoarei proceduri:

- Tastatura trebuie să funcţioneze.
- Setaţi timpul la 23:59:50 şi rulaţi. Verificaţi că tranziţia 23:59:59 → 0:00:00 se face corect.
- Faceţi ca timpul afişat pe LCD să fie egal cu ceasul din Widows. Mai întâi setaţi timpul iniţial în avans cu aproximativ un min faţă de timpul calculatorului. Timpul iniţial trebuie să aibă secundele egale cu zero. Daţi comanda Run când timpul setat în program şi ceasul Windows devin egali. Se admite o diferenţă de o secundă. De exemplu dacă ceasul Widows este 16:43:13 setaţi timpul iniţial la 16: 44:00. **Imediat** după ce ați pornit simularea, chemați profesorul.
- Lăsaţi să ruleze cel puțin 5 minute. In acest interval nu executați nici un alt program, nu atingeți tastatura sau mouse-ul.
- Dacă după cele 5 minute nu există diferențe între timpul afişat pe LCD şi timpul PC, apelați profesorul pentru validare.