4. Modulul de masurare a timpului (Timer)

Modulul de masurare al timpului, eng. *Timer*, este resursa microcontrolerului ce permite măsurarea cuantelor de timp. Pentru a păstra o terminologie similară cu aceea a literaturii de specialitate, în cadrul acestei lucrări, modulul de timp va fi referit prin termenul *Timer*. Principala caracteristică a unui Timer este aceea că acesta este implementat hardware, adică, întrg mecanismul de cronometrare a timpului este gestionat independent de unitatea aritmetică și logică (UAL), aceasta putând fi utilizată pentru a rula programul principal.

Un Timer este capabil să cronometreze timpul scurs utilizând un aşa numit accumulator (eng. *Counter*). Valoarea acumulată este staocată în registru *TimerCouNTer* (TCNT).Fiecare incrementare a acumulatorului este realizată atunci când osciloatorul emite un puls. Fiecare puls reprezintă simbolic scurgerea unei cuante de timp. Drept exemplu, se consideră un oscilator cu o frecvență de 4.000.000 de Hz. Incrementând acumulatorului la fiecare puls al osciloatorului, când am ajunge la valoarea 4.000.000 în acumulator am ști că a trecut 1 secundă. Pentru a stoca o valoare precum 3.999.999 în memoria microcontrolerului sunt necesari 4 bytes de memorie . Constructiv, modulul Timer nu poate avea un acumulator mai mare de 1 sau 2 bytes (8 sau 16 biți) de memorie, respectiv nu poate acumula o valoare mai mare de 255, respectiv 65535. Soluția este ca Timer-ul să genereze o întrerupere atunci când a ajuns la valoarea maximă posibilă (eng. *TOP*). Dacă s-ar mai fi adăugat o singură cuantă în accumulator s-ar fi produs o așa numită depășire (eng. *overflow*). Pentru flexibilitate, modulul Timer permite definirea unui prag variabil, numit eng. *MAX*, la atingerea căruia se va realiza întreruperea. În esență, max-ul defineşte pragul care dacă este atins se generază o întrerupere de tip timer, iar TOP-ul definește valoarea maximă posibilă a acumulatorului. În întreruperea de timer generată, utilizatorul poate gestiona informația pentru a determina timpul scurs. Se consideră un Timer de 8 biți (acumulatorul poate avea valoarea maxima 255) și un oscilator de 4.000.000 Hz. Pentru a considera scurgerea unei secunde ar trebui să

numărăm 15625 (adica 4.000.000 / 256) de intreruperi de timer. Fiecare intrerupere se va genera atunci cand s-a atins valoarea maximă în acumulator.

În funcție de cum este configurat, când un acumulator ajunge la depășire, modulul Timer-ul poate realiza una din următoarele acțiuni:

- se resetează la 0 (ca și cum MAX ar fi fost egal cu TOP). În acest caz, counterul numără de la 0 până la MAX - 1, după care se întoarce din nou la zero. Modificând valoarea lui MAX (valoarea de TOP nu o putem modifica, fiind stabilită la nivel de hardware), putem face numărătorul să declanșeze evenimente mai des sau mai rar. Stabilirea valorii MAX la zero ar trebui sa aibă ca efect dezactivarea lui (și resetarea se face doar când contorul ajunge la TOP);
- începe sa numere în jos - vom vedea în continuare situații în care acest lucru poate fi util;
- generează un eveniment (la fel ca în situația 1), însă nu se resetează, ci își continuă număratul până la TOP, de unde se resetează și numără din nou, și așa mai departe;
- generează un eveniment și se oprește din numărat (se mai numește și *one-shot timer*).

La fiecare incrementare automată a registrului TCNT are loc o comparație între acest registru și o valoare stocată în registrul OCRn. Această valoare poate fi încarcată de către programator prin scrierea registrului OCRn. Dacă are loc egalitatea se generează o întrerupere, în caz contrar incrementarea continuă(vezi figura 4.1).

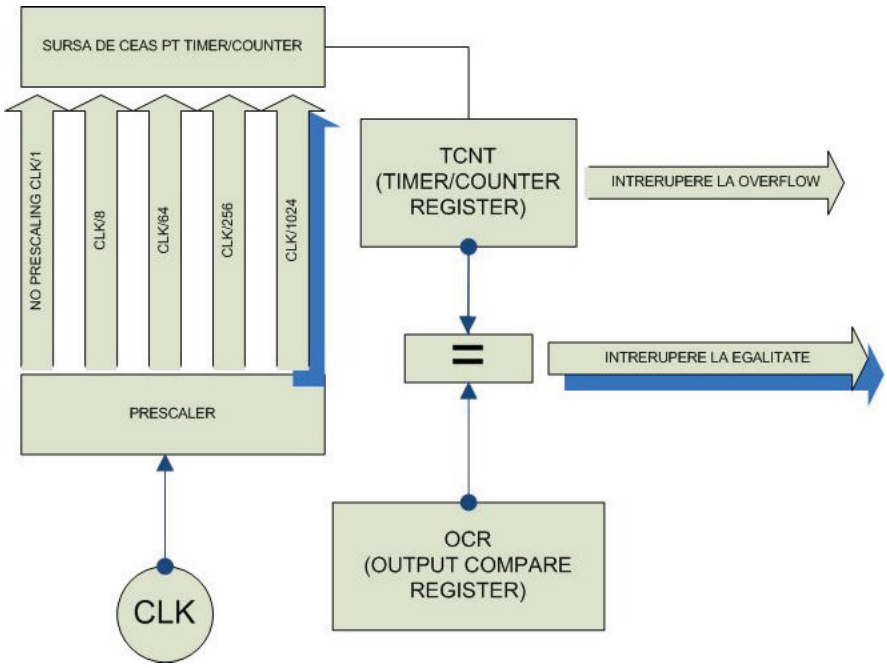

Fig. 4.1 Modul de funcționare al unui timer pentru un microcontroler din familia Atmega.

Modulul Timer este prevăzut cu mai multe registre accumulator. Acest lucru permite numărători paralele. De exemplu, microcontrolerul Amega 328P deține 3 astfel de registre, două de 8-biți și unul de 16-biți.

În funcție de valoarea până la care se face acumularea, când se face resetarea acumulatorului sau felul în care se face accumularea (increment sau decrement) pot exista următoarele

moduri de funcționare:

- *normal*. La resetare, numărătoarea începe de la 0 și depășirea se produce la 255. Acest mod este folosit uzual pentru a măsura timpul;
- *CTC (Clear Timer on Compare)*. Pornește de la valoarea 0. Registrul OCR definește valoarea la care se face depășirea.
- *fast PWM*. Pornește de la valoarea 0. Registrul OCR sau valoarea 255 definesc când se face depășirea. Pinul OC0x generează forme de undă de tip PWM;
- *phase-correct PWM*. Registrul TCNT se incrementează până la depășire după care se decrementează. Valoarea de depășire se poate defini până la OCR sau 255. Pinul OC0x generează forme de undă de tip PWM.

Descrierea detaliată a fiecărui mod de lucru al Timer-ului poate fi urmărită în foaia de catalog a fiecărui microcontroler.

4.1 Regiștrii Timer-ului

Microcontrolerul Atmega 328P deține 2 module de tip timer/counter de 8 biți, cu funcție de comparare și prescalar separat, și un timer de 16 biți cu funcții de comparare, capturare și prescalar separat.

Modul de funcționare al timerului de 8 biți poate fi urmărit în diagrama bloc din figura 4.2, unde n reprezintă numărul unității de timer, respectiv A, B reprezintă unitatea de comparare.

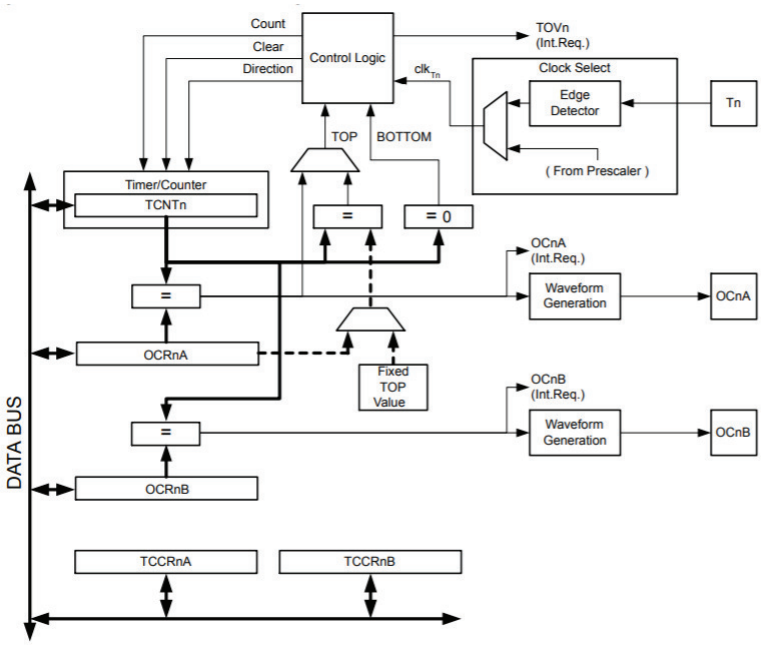

Fig. 4.2 Diagrama bloc a modulului timer de 8 biți.

Pentru exemplificare, în continuare vor fi descriși regiștrii timer-ul 0. Pentru o descriere completă a regiștrilor celorlalte module timer se recomandă consultarea documentației producătorului [2].

În figurile 4.3, respectiv 4.4 sunt prezentați regiștrii TCCRA (T*imer* C*ounter* C*ontrol* R*egister A*), respectiv TCCR0B (T*imer* C*ounter* C*ontrol* R*egister B*). Aceștia permit con-

figurarea modului de generare a formei de undă, respectiv a valorii prescalarului semnalului de ceas.

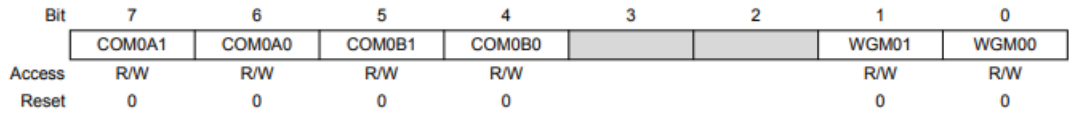

Fig. 4.3 Registrul *timer/counter control register 0A* (TCCR0A).

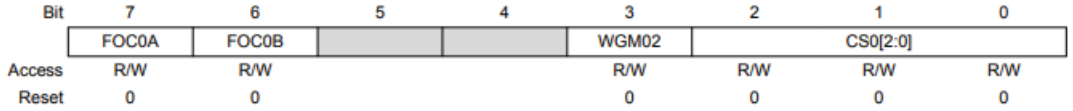

Fig. 4.4 Registrul *timer/counter control register 0B* (TCCR0B).

Principalii biți ai registrului TCCR0A sunt:

– Biții 7:6 - COM0A1 și COM0A0, se adresează doar modului de funcționare cu generare de formă de undă tip PWM. Ei prezintă următoarea funcționalitate:
  - 00: OC0A dezactivat;
  - 01: și WGM02 = 0: PORT cu funcție normală; OC0A deconectat;
  - 10: și WGM02 = 1: basculează OC0A atunci când comparatorul semnalează o potrivire cu cea a valorii de comparare;
  - 11: logică inversată (LOW la bază și HIGH la atingerea valorii de comparare);
– Biții 5:4 - COM0B1 și COM0B0, se adresează doar modului de funcționare cu generare de formă de undă tip PWM. Ei prezintă următoarea funcționalitate:
  - 00: OC0B dezactivat;
  - 01: mod de funcționare tip înapoi (eng. Reversed);
  - 10: mod de funcționare ne-inversat (eng. Non-Inverted) (HIGH la bază și LOW la atingerea valorii de comparare);
  - 11: logică inversată (LOW la bază și HIGH la atingerea valorii de comparare);
– Biții 1:0 - WGM01 și WGM00, în combinație cu WGM02 permite configurarea formei de undă generate de către timer. Funcționalitatea acestor biți este prezentată în tabelul 4.1;

Principalii biți ai registrului TCCR0B:

– Biții 7:6 - FOC0A și FOC0B (eng. F*orce* O*ut* C*ompare*). Sunt activi doar când biții WGM sunt setați pentru a configura o formă de undă de tip PWM. Acest bit nu va genera o întrerupere și nici o resetare a comparatorului atunci când se va ajunge la valoarea de top. Biții FOC0A și FOC0B sunt citiți mereu ca valoare 0;
– Bitul 3 - WGM02, în combinație cu alți regiștrii similari, permite configurarea formei de undă generate de către timer. Funcționalitatea acestui bit este prezentată în tabelul 4.1;
– Biții 2:0 - CS02, CS01 și CS00 permit configurarea prescalarului (vezi tabelul 4.2).

În figura 4.5 sunt prezentați biții registrului TIMSK0 (T*imer* C*ounter* I*nterrupt* M*ask*

Tab. 4.1 Forme de unda ce pot fi generate cu un timer.

Mod	WGM02	WGM01	WGM00	Descriere	TOP
0	0	0	0	Normal	0xFF
1	0	0	1	PWM, Phase corrected	0xFF
2	0	1	0	CTC	OCRA
3	0	1	1	Fast PWM	0xFF
4	1	0	0	Rezervat	-
5	1	0	1	Fast PWM, phase corrected	OCR0A
6	1	1	0	Rezervat	-
7	1	1	1	Fast PWM	OCR0A

Tab. 4.2 Lista prescalarilor disponibili pentru divizarea semnalului de ceas.

CS02	CS01	CS00	Descriere
0	0	0	Timer / counter dezactivat
0	0	1	Fără prescalar
0	1	0	Semnal de ceas / 8
0	1	1	Semnal de ceas / 64
1	0	0	Semnal de ceas / 256
1	0	1	Semnal de ceas / 1024
1	1	0	Sursă externă de ceas conectată la pinul T0 (front crescător)
1	1	1	Sursă externă de ceas conectată la pinul T0 (front descrescător)

R*egister*). Acest registru este utilizat pentru a activa întreruperile generate de către evenimente provenite de la circuitul comparator al timerului, după cum urmează:

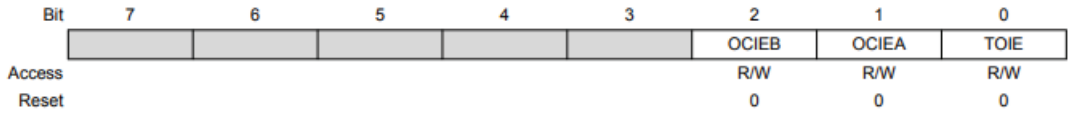

Fig. 4.5 Registrul *timer/counter interrupt mask register* (TIMSK0).

- Bitul 2 - OCIEB (eng. O*utput* C*ompare* I*nterrupt* E*nable* B). Activează (setat pe valoarea 1) sau dezactivează (setat pe valoarea 0) întreruperea generată atunci când valoarea din registrul TCNT0 este egală cu valoarea din OCR0B. Pre-condiție: bitul OCF0B trebuie să fie setat în TIFR0;
- Bitul 1 - OCIEA (eng. O*utput* C*ompare* I*nterrupt* E*nable* A). Activează (setat pe valoarea 1) sau dezactivează (setat pe valoarea 0) întreruperea generată atunci când valoarea din registrul TCNT0 este egală cu valoarea din OCR0A. Pre-condiție: bitul OCF0A trebuie să fie setat în TIFR0;
- Bitul 0 - TOIE (eng. T*imer* O*utput* I*nterupt* E*nable*). Activează (setat pe valoarea 1) sau dezactivează (setat pe valoarea 0) întreruperea generată de depășirea valori maxime acumulate (0xFF) în registrul TCNT0 (eng. overflow interrupt). Pre-condiție: bitul TOV trebuie să fie setat în TIFR0.

În figura 4.6 sunt prezentați biții registrului TIFR0 (T*imer* I*nterrupt* F*lag* R*egister*). În acest registru sunt observabile evenimentele provocate de către unitatea timer. Semnificația biților acestui registru este următoarea:

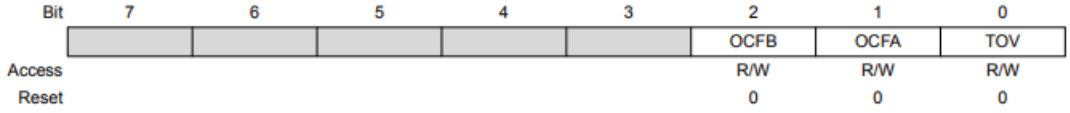

Fig. 4.6 Registrul *timer/counter interrupt flag register* (TIFR0).

- Bitul 2 - OCFB (eng. O*utput* C*ompare* F*lag* B). Acest bit este setat automat de către timer atunci când valoarea din registrul TCNT0 este egală cu valoarea din OCR0B;
- Bitul 1 - OCFA (eng. O*utput* C*ompare* F*lag* A). Acest bit este setat automat de către timer atunci când valoarea din registrul TCNT0 este egală cu valoarea din OCR0A;
- Bitul 0 - TOV (eng.T*imer* OV*erflow*). Acest bit este setat automat de către timer atunci când valoarea din registrul TCNT0 depășește valoarea de TOP (0xFF).

În figura 4.7 este prezentat registrul TCNT0 (T*imer* C*ounter* 0). Acest registru conține valoarea accumulată incremental la excitațiile oscilatorului. Biții lui permit accesul direct, pentru scriere și/sau citire, la contorul (acumulatorul) de 8 biți al timerului 0.

În figura 4.8, respectiv 4.9 sunt prezentate registrele OCR0A (O*utput* C*ompare* R*egister* 0 A), respectiv OCR0B. Acești regștrii conțin valoarile de prag ce sunt folosite de către timer pentru a genera diferite forme de undă.

Tratarea unei întreruperi generate de către un modul timer se face în interiorul funcției ISR(). După cum a fost prezentat în tabelului 3.1 din capitolul anterior, sintaxa funcției

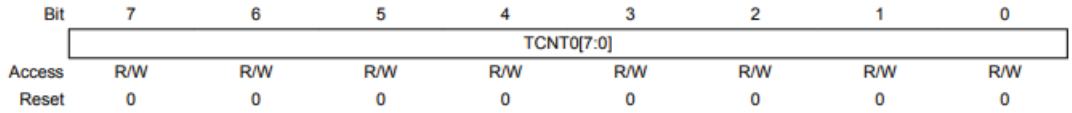

Fig. 4.7 Registrul *timer/counter register* (TCNT0) (folosit pentru a stoca valoarea acumulată).

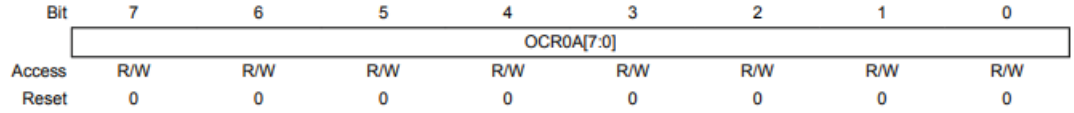

Fig. 4.8 Registrul *output compare register A* (OCR0A).

ISR, pentru diferitele surse de întrerupere provocate de timer (overflow, respectiv compare A și B), este următoarea:

Algoritmul 4.1 Rutina de deservire a întreruperii generate de depășirea valorii de TOP a registrului TCNT0

```
1
2 ISR (TIMER0_OVF_vect)
3 {
4     //ISR_code
5 }
```

Algoritmul 4.2 Rutina de deservire a întreruperii generate atunci când valoarea din registrul TCNT0 este egală cu valoarea din OCR0A

```
1
2 ISR (TIMER0_COMPA_vect)
3 {
4     //ISR_code
5 }
```

Algoritmul 4.3 Rutina de deservire a întreruperii generate atunci când valoarea din registrul TCNT0 este egală cu valoarea din OCR0B

```
1
2 ISR (TIMER0_COMPB_vect)
3 {
4     //ISR_code
5 }
```

4.2 Exemplu de aplicaţie

Să se elaboreze și simuleze, utilizându-se Tinkercad, un ceas digital folosindu-se un timer de 8 biți. Pentru reglarea orei, sistemul va conține două butoane care, tratate utilizând

Bit	7	6	5	4	3	2	1	0
	OCR0B[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Fig. 4.9 Registrul *output compare register B* (OCR0B).

sistemul de întreruperi externe, va oferi posibilitatea reglării orelor, respectiv a minutelor (incremental la fiecare apăsare de buton). Afișarea orei se va realiza pe un afișaj LCD cu 2 lini și 16 coloane. Contorizarea timpului se va face într-o rutină specială pentru întrerupere de tip timer. Se impune ca întreruperea să fie generată odată la fiecare 4 milisecunde. Schema electrică propusă este prezentată în figura 4.10. Componentele utilizate în schemă sunt:

- 1x Arduino Uno R3;
- 1x LCD 16x2;
- 2x 1 kohm Resistor (pull-down pentru butoane);
- 1x 70 ohm Resistor (pentru a stabili intensitatea scrisului);
- 1x 200 ohm Resistor(pentru a stabili iluminarea fundalului LCD-ului);
- 2x Butoane;

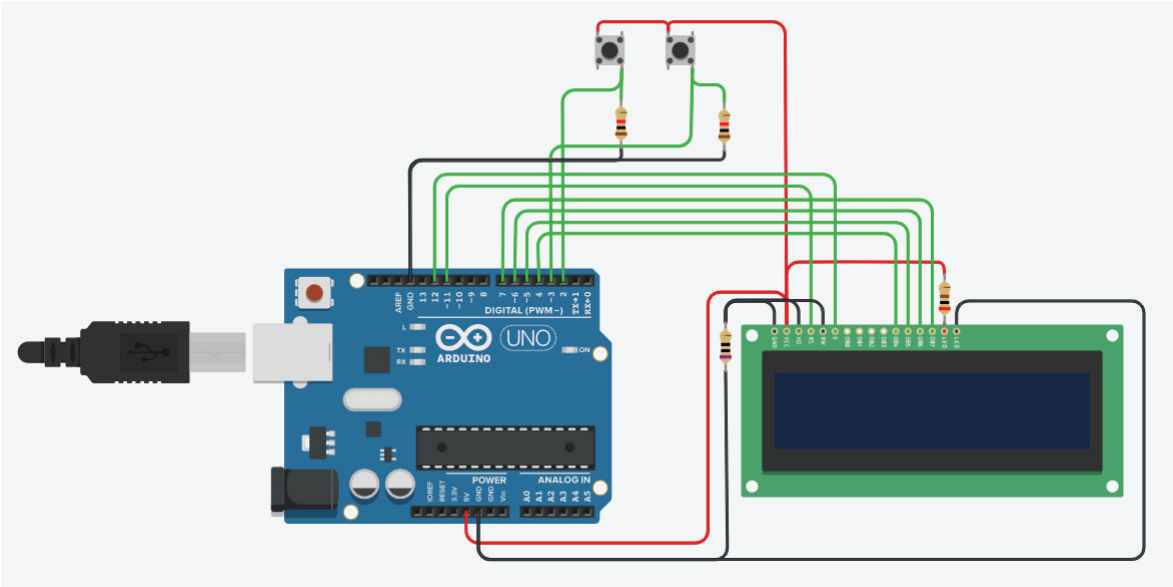

Fig. 4.10 Schema electrică de funcționare a unui ceas digital.

Programul care realizează cerințele impuse este următorul:

Algoritmul 4.4 Utilizarea modulului timer 0 pentru implementarea unui ceas digital.

```
1 /*
2  * Conectarea pinilor la placa Arduino UNO:
3  *  LCD RS -> 11
4  *  LCD E (Enable) -> 12
5  *  LCD DB4 -> 4
6  *  LCD DB5 -> 5
7  *  LCD DB6 -> 6
8  *  LCD DB7 -> 7
```

```
 *  LCD RW -> GND
 *  LCD VSS -> GND
 *  LCD V0 -> Rezistor 70Ohm -> GND
 *  LCD VCC -> 5V
 */

// include libraria pentru LCD
#include <LiquidCrystal.h>

// asociere pini Arduino <-> LCD
const byte RS = 11, EN = 12;
const byte DB4 = 4, DB5 = 5, DB6 = 6, DB7 = 7;
LiquidCrystal lcd(RS, EN, DB4, DB5, DB6, DB7);


int contor = 0;

int secunde = 0;
int minute = 0;
int ore = 0;

void setup()
{
    // definire numar linii si coloane pentru LCD
    lcd.begin(16 , 2);
    // afisare mesaj pe prima linie a LCD-ului
    lcd.print("Ora exacta: ");

    // setare pini butoane ca pini de intrare
    DDRD |= (1 << 2) | (1 << 3);

    // configurare intreruperi externe
    EICRA |=(1 << ISC11) | (1<< ISC10) | (1 << ISC01) | (1 <<
        ↪ ISC00);
    EIMSK |=(1 << INT1) | (1 << INT0);
    EIFR |=(0 << INTF1) | (0 << INTF0);
    PCICR |=(0 << PCIE2) | (0 << PCIE1) | (0 << PCIE0);

    // configurare timer sa execute o intrerupere
    // la fiecare 4ms
    // seteaza mod de functionare CTC
```

```
49      TCCR0A = (1 << WGM01) | (0 << WGM00);
50
51      // configurare valoare de TOP
52      OCR0A = 0xF9;
53
54      // activeaza intrerupere aferenta modului de
55      // functionare de tip comparare
56      TIMSK0 |= (1 << OCIE0A);
57
58      // configurare prescalar 256
59      TCCR0B = (1 << CS02) | (0 << CS01) | (0 << CS00);
60
61      // activare intreruperi globale
62      SREG |= (1 << SREG_I);
63  }
64
65  void loop()
66  {
67      // afisare ora
68      lcd.setCursor(0, 1);
69      if (ore < 10)
70      {
71          // daca avem mai putin de 10 ore (0-9),
72          // atunci afisam un 0 inainte (ex: 08)
73          lcd.print(0);
74          lcd.setCursor(1, 1);
75      }
76      lcd.print(ore);
77
78      // afisare minute
79      lcd.setCursor(2, 1);
80      lcd.print(":");
81      lcd.setCursor(3, 1);
82      if (minute < 10)
83      {
84          lcd.print(0);
85          lcd.setCursor(4, 1);
86      }
87      lcd.print(minute);
88
89      // afisare secunde
```

```
90      lcd.setCursor(5, 1);
91      lcd.print(":");
92      lcd.setCursor(6, 1);
93      if (secunde < 10)
94      {
95          lcd.print(0);
96          lcd.setCursor(7, 1);
97      }
98      lcd.print(secunde);
99  }
100
101 // functie pentru incrementarea orelor
102 // numarul maxim de ore pe zi este 24 (0-23)
103 inline void incrementOre()
104 {
105     ++ore;
106     if (ore >= 24)
107     {
108         // daca avem mai mult de 24 de ore,
109         // revenim la 0
110         ore %= 24;
111     }
112 }
113
114 // functie pentru incrementaerea minutelor
115 // numarul maxim de minute intr-o ora este 60 (0-59)
116 inline void incrementMinute()
117 {
118     ++minute;
119     if (minute >= 60)
120     {
121         // daca avem mai mult de 60 de minute,
122         // atunci incrementam orele
123         incrementOre();
124         // revenire la 0 minute
125         minute %= 60;
126     }
127 }
128
129 // rutina de intrerupere pentru contorizarea secundelor
130 ISR(TIMER0_COMPA_vect)
```

```
131 {
132     // dezactivare intreruperi globale
133     SREG &= ~(1 << SREG_I);
134 
135     // au mai trecut 4ms, deci incrementam contor
136     ++contor;
137 
138     // daca contorul a ajuns la 250, adica 250x4ms=1s
139     if (contor >= 250)
140     {
141         // incrementam secundele si resetam contor
142         ++secunde;
143         contor = 0;
144 
145         if (secunde >= 60)
146         {
147             // daca avem 60 de secunde, incrementam
148             // minutele si revenim la 0 secunde
149             incrementMinute();
150             secunde %= 60;
151         }
152     }
153 
154     // activare intreruperi globale
155     SREG |= (1 << SREG_I);
156 }
157 
158 // rutina intrerupere pentru INT0 (pinul 2)
159 ISR(INT0_vect)
160 {
161     SREG &= ~(1 << SREG_I);
162     // incrementam minutele
163     incrementMinute();
164     SREG |= (1 << SREG_I);
165 }
166 
167 // rutina intrerupere pentru INT1 (pinul 3)
168 ISR(INT1_vect)
169 {
170     SREG &= ~(1 << SREG_I);
171     // incrementam orele
```

```
172     incrementOre();
173     SREG |= (1 << SREG_I);
174 }
```

Pentru exemplul de aplicație mai sus prezentat s-a considerat un oscilator de 16 MHz (disponibil de altfel și placa de dezvoltare Arduino UNO) și un timer de 8 biți. Pentru cronometrarea timpului se are în vedere generarea unei întreruperi provenite de la modulul timer la fiecare 4 ms (0.004s). Se vor avea în vedere urmatoarele considerații.

Frecvența de apariție a întreruperii este:

$$f_{dorita} = \frac{1s}{0.004s} = 250Hz \tag{4.1}$$

Deoarece timer-ul este de 8 biți, valoarea de TOP a registrului accumulator TCNT0 va fi 256. Astfel, frecventa de apariție a întrerupei va fi 62500 de întreruperi pe secundă.

$$f_{realizata} = \frac{16000000}{256} = 62500Hz \tag{4.2}$$

În aceestă configurație, f_{dorita} este mult mai mică decât $f_{realizata}$. Pentru a reduce $f_{realizata}$ putem reduce frecventa oscilatorului prin utilizarea unui prescalar. Spre exemplu, se poate folosi un prescalar de 256 (vezi linia de cod 61). Noua $f_{realizata}$ devine:

$$f_{realizata} = \frac{16000000}{256 * 256} = 244.41 \tag{4.3}$$

De această dată frecvența este mai mică decât valoarea dorită. Deoarece prescalarul are valori pre-definite, pentru a obține exact valoarea dorită putem să micșorăm doar valoarea de TOP (256) cu o valoare preferențială. Această valoare va fi definită în registrul OCR0A. Pentru acest lucru trebuie să configurăm timerul să funcționeze în modul CTC (Clear Timer on Compare). Setarea acestui mod se face din registrul TCCR0A (vezi linia de cod 52). Acest mod generează o întrerupere de tip timer (TIMER0-COMPA) atunci când registrul TCNT0 este egal cu valoarea definită de utilizator în registrul OCR0A (vezi foaia de catalog a microcontrolerului).

Modul de calcul al valorii ce trebuie scrisă în registrul OCR0A (vezi linia de cod 54) este următorul:

$$\frac{16000000}{256 * n} = 250. \tag{4.4}$$

unde :

$$n = \frac{16000000}{256 * 250} = 250 = 0xFA(hexazecimal) \tag{4.5}$$

Așadar în OCR0A trebuie configurată valoarea:

$$OCR0A = 0xFA - 1 = 0xF9(hexazecimal) \tag{4.6}$$

4.3 Aplicație propusă

Să se implementeze un cronometru digital utilizând platforma de dezvoltare Arduino Uno. Rezoluția de măsurare trebuie să fie de 100 microsecunde. Afișarea timpului scurs va fi făcută pe un afișaj LCD 16x2. Pentru măsurarea timpului se impune folosirea timerului de 16 biți. Pornirea cronometrului se va face la apăsarea unui buton fără reținere. Evenimentul se impune a fi tratat într-o întrerupere externă. Oprirea / punerea pe pauză a timpului cronometrat se va realiza prin re-apăsarea aceluiași buton. Resetarea timpului cronometrat se va face prin apăsarea unui al doilea buton, tot fără reținere. La fel, evenimentul pentru resetare va fi tratat într-o întrerupere externă.