Laboratorul 2 P.M

Lucrul cu Intreruperile. Lucrul cu Timere .PWM. Lucrul cu Memoria

I. ASPECTE TEORETICE

In acest laborator sunt tratate urmatoarele aspecte:

I.1 Lucrul cu intreruperile

I.2 Lucrul cu Timere(PWM)

I.3 Lucrul cu Memoria

I.1 Lucrul cu intreruperile

***O* intrerupere** reprezinta un semnal sincron sau asincron de la un periferic ce semnalizeaza aparitia unui eveniment care trebuie tratat de catre procesor. Tratarea intreruperii are ca efect suspendarea firului normal de executie al unui program si lansarea in executie a unei rutine de tratare a intreruperii (RTI).

Intreruperile hardware au fost introduse pentru a se elimina buclele pe care un procesor ar trebui sa le faca in asteptarea unui eveniment de la un periferic. Folosind un sistem de intreruperi, perifericele pot atentiona procesorul in momentul producerii unei intreruperi (IRQ), acesta din urma fiind liber sa-si ruleze programul normal in restul timpului si sa isi intrerupa executia doar atunci cand este necesar.

Inainte de a lansa in executie o RTI, procesorul trebuie sa aiba la dispozitie un mecanism prin care sa salveze starea in care se afla in momentul aparitiei intreruperii. Aceasta se face prin salvarea intr-o memorie, de cele mai multe ori organizata sub forma unei stive, a registrului contor de program (Program Counter), a registrelor de stare precum si a tuturor variabilelor din program care sunt afectate de executia RTI. La sfarsitul executiei RTI starea anterioara a registrelor este refacuta si programul principal este reluat din punctul de unde a fost intrerupt.

Pentru a asocia o intrerupere cu o anumita rutina din program, procesorul foloseste tabela vectorilor de intrerupere (TVI). Fiecarei intreruperi ii este asociata o adresa la care programul va face salt in cazul aparitiei acesteia. Aceste adrese sunt predefinite si sunt mapate in memoria de program intr-un spatiu contiguu care alcatuieste TVI. Adresele intreruperilor in TVI sunt setate in functie de prioritatea lor, cu cat adresa este mai mica cu atat prioritatea este mai mare.

Pentru ATMega16, TVI este data in tabelul de mai jos:

Vector No.	Program Address[2]	Source	Interrupt Definition
1	$000[1]	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	$008	TIMER2 OVF	Timer/Counter2 Overflow
6	$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	$00E	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	$010	TIMER1 OVF	Timer/Counter1 Overflow
10	$012	TIMER0 OVF	Timer/Counter0 Overflow
11	$014	SPI, STC	Serial Transfer Complete
12	$016	USART, RXC	USART, Rx Complete
13	$018	USART, UDRE	USART Data Register Empty
14	$01A	USART, TXC	USART, Tx Complete
15	$01C	ADC	ADC Conversion Complete
16	$01E	EE_RDY	EEPROM Ready
17	$020	ANA_COMP	Analog Comparator
18	$022	TWI	Two-wire Serial Interface
19	$024	INT2	External Interrupt Request 2
20	$026	TIMER0 COMP	Timer/Counter0 Compare Match
21	$028	SPM_RDY	Store Program Memory Ready

Se observa ca TVI este plasata de la prima adresa a memoriei de program si ca intreruperile sunt puse din doua in doua adrese consecutive. Prioritatea cea mai mare o are intreruperea de RESET, de la adresa 0, apoi intreruperea externa 0 (INT0).

Perifericele care pot genera intreruperi la ATMega16 sunt timerele, interfata seriala (USART), convertorul analog-digital (ADC), controllerul de memorie EPROM, comparatorul analog si interfata seriala I2C. Deasemenea, procesorul poate sa primeasca cereri de intreruperi externe din trei surse (INT0, 1 si 2) ce corespund unui numar egal de pini exteriori.

Activarea/Dezactivarea intreruperilor:

Intreruperile pot fi activate sau dezactivate de utilizator in program prin setarea individuala a bitilor de interrupt enable pentru fiecare periferic folosit si prin setarea flagului de "Global Interrupt Enable" (I) din Status Register (SREG).

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
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	

Tratarea unei intreruperi pentru ATMega16 se face ca in figura alaturata.

Sa presupunem ca programul nostru primeste intreruperea externa INT0 in timp ce executa instructiunea *ldi R16,0xFF*. Dupa efectuarea instructiunii, registrul contor program (PC) este automat salvat in stiva si apoi initializat la valoarea corespunzatoare adresei lui INT0 ($002). Acest lucru se traduce prin saltul programului de la adresa curenta ($123) la adresa $002. Aici, programul gaseste instructiunea *jmp $2FF* care-i permite sa sara la rutina efectiva de tratare a intreruperii. La sfarsitul oricarei RTI trebuie sa existe instructiunea *reti* care reface din stiva registrul PC (programul sare inapoi la adresa de unde ramasese inainte de intrerupere).

In timpul executiei unei intreruperi, bitul I din SREG este setat la 0 si resetat la iesire, adica orice alta intrerupere care poate aparea in timpul intreruperii curente nu va fi luata in seama. Cu toate acestea, utilizatorul poate sa reseteze bitul I din software, permitand astfel executia de intrerupere in intrerupere.

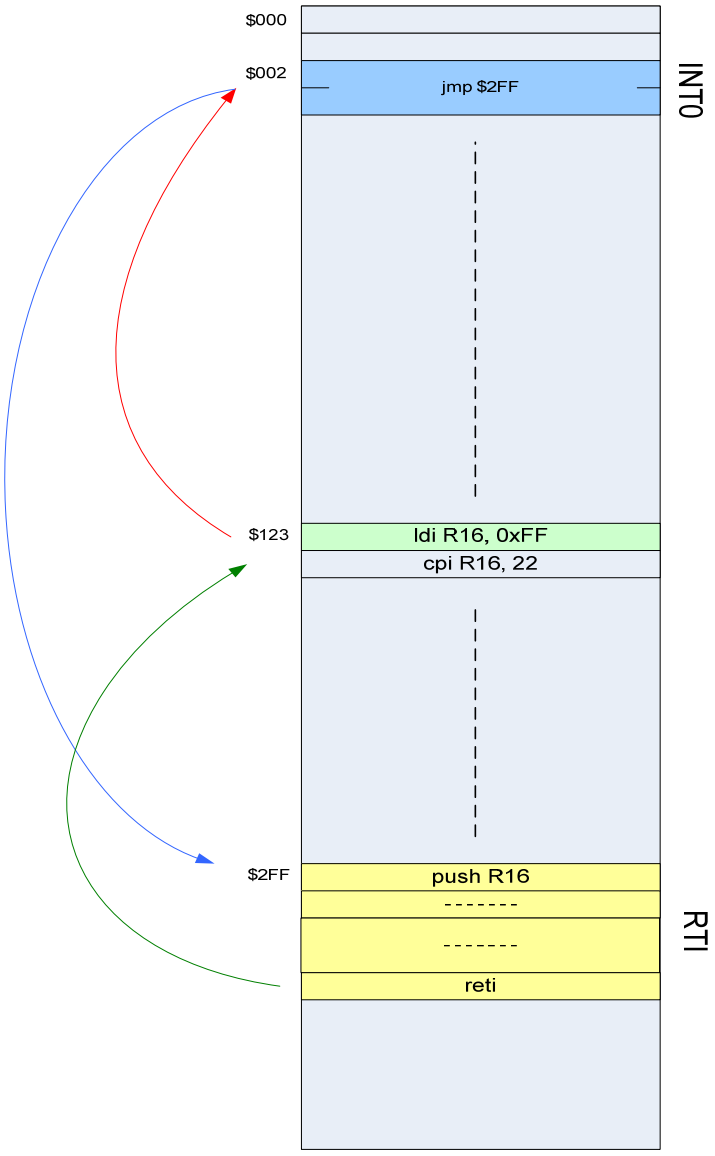


Figura 1. Tratarea Intreruperilor

Concret, o secventa de cod care ar trata o intrerupere este:

```
.cseg; aici incepe segmental de cod
;tabela de intreruperi
.org 0
      jmp main                              ; intreruperea de reset
      jmp int0_handler            ;     external int 0
      jmp int1_handler;      external int 1
main:
      ldi r16, HIGH(RAMEND)                        ; initializeaza stiva
      out SPH, r16
      ldi r16, LOW(RAMEND)
      out SPL, r16
[…call la  rutine]
sei  ; set interrupt enable dam drumul ingreruperilor  setand bitul "I"  din registrul SREG
;/////RUTINA DE TRATARE INTRERUPERI/////
      int0_handler:
      push  r16 ;salvam un registru in stiva
      in r16, SREG ;citim valoarea din  SREG in registrul general r16
      push r16; punem pe stiva valoarea
      push r17; salvam pe stiva r17
      in r16, OCR2 ; in registrul  r16 incarcam  valoarea din registrul OCR2
      ldi r17, ALPHA_STEP;
      add r16, r17; adunare fara carry;
      brcs int0_exit  ; daca avem carry =1 in sreg salt la int0_exit
      out OCR2, r16
```

```
int0_exit:
    pop r17; scoate din stiva r17
    pop r16; scoate din stiva r16
    out SREG, r16 ;reface SREG
    pop r16; reia valoarea lui r16 din stiva
    reti; se reintoarce din intrerupere
```

Registre pentru tratarea intreruperilor externe

Moving Interrupts Between Application and Boot Space

The General Interrupt Control Register controls the placement of the Interrupt Vector table.

General Interrupt Control Register – GICR

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Acest registru este raspunzator pentru plasarea tabelului vectorului de intreruperi.

Bitul 0 din acest registru se numeste **IVCE** (Interrupt Vector Change Enable) .Acest bit trebuie initilizat cu valoarea logica "1" pentru a permite schimbarea urmatorului bit din registru si anume IVSEL.Setand acest bit nu se vor mai putea genera intreruperi.

Bitul 1 din acest registru se numeste **IVSEL** (Interrupt Vector Select).Cand acest bit are valoarea logica "0" vectorii de intreruperi sunt plasati la inceputul memoriei de program(Flash). Cand acest bit este setat cu valoarea logica "1" vectorii de intreruperi sunt mutati la inceputul zonei de boot .

Bitul 7 –INT1:External Interrupt Request 1 Enable:

Cand INT1 e setat cu valoarea logica "1" si bitul "I" din SREG este setat cu aceeasi valoare, pinul destinat intreruperii externe INT1 este activat. Bitii ISC11 si ISC10 definesc logica de generare a intreruperii, respective daca aceasta este generata pe frontul crescator sau descrescator. Intreruperea va fi setata conform rutinei asociate vectorului de intrerupere.

Bitul 6–INT0:External Interrupt Request 0 Enable:

Cand INT0 e setat cu valoarea logica "1" si bitul "I" din SREG este setat cu aceeasi valoare pinul destinat intreruperii externe 0 este activat. Bitii ISC01 si ISC00 definesc logica de generare a intreruperii, respective daca aceasta este generata pe tranzitia de crestere sau pe cea de scadere.

Bitul 5–INT2:External Interrupt Request 2 Enable:

Cand INT0 e setat cu valoarea logica "1" si bitul "I "din SREG este setat cu aceeasi valoare, pinul destinat intreruperii externe 2 este activat. Bitul ISC2 defineste logica de generare a intreruperii, respective daca aceasta este generata pe tranzitia de crestere sau pe cea de scadere.

Intreruperile externe

Aceste intreruperi sunt generate prin intermediul pinilor INT0, INT1 si INT2. Ele sunt activate chiar daca acesti pini sunt setati ca fiind de output. Modul in care se pot genera intreruperile externe poate fi setat prin configurarea registrelor MCUCR si MCUCSR.

MCU Control Register – MCUCR

The MCU Control Register contains control bits for interrupt sense control and general MCU functions.

Bit	7	6	5	4	3	2	1	0	
	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
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	

- **Bit 3, 2 – ISC11, ISC10: Interrupt Sense Control 1 Bit 1 and Bit 0**

Table 34. Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

MCU Control and Status Register – MCUCSR

Bit	7	6	5	4	3	2	1	0	
	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	See Bit Description					

- **Bit 6 – ISC2: Interrupt Sense Control 2**

De exemplu, pentru a folosi intreruperea externa INT2 sunt necesare urmatoarele configurari:

1. Bitul I din SREG trebuie sa fie setat (global interrupt enable)
2. Bitul INT2 din GICR trebuie setat (INT2 enable)
3. Daca ISC2 este initializat cu valoarea zero INT2 va fi activata pe front descrescator (tranzitie din 1 in 0 a pinului INT2), in caz contrar ea va fi activata pe front crescator.

Perifericele care pot genera intreruperi sunt: timerele, convertorul analog-digital, interfetele seriale (RS232, I2C, SPI) etc.

I.2 Lucrul cu Timerele(PWM)

Principiul de functionare al unui Timer

Timerul/Counterul, dupa cum ii spune si numele ofera facilitatea de a masura intervale fixe de timp si de a genera intreruperi la expirarea intervalului masurat. Un timer, odata initializat va functiona independent de unitatea centrala (core μP). Acest lucru permite eliminarea buclelor de delay din programul principal.

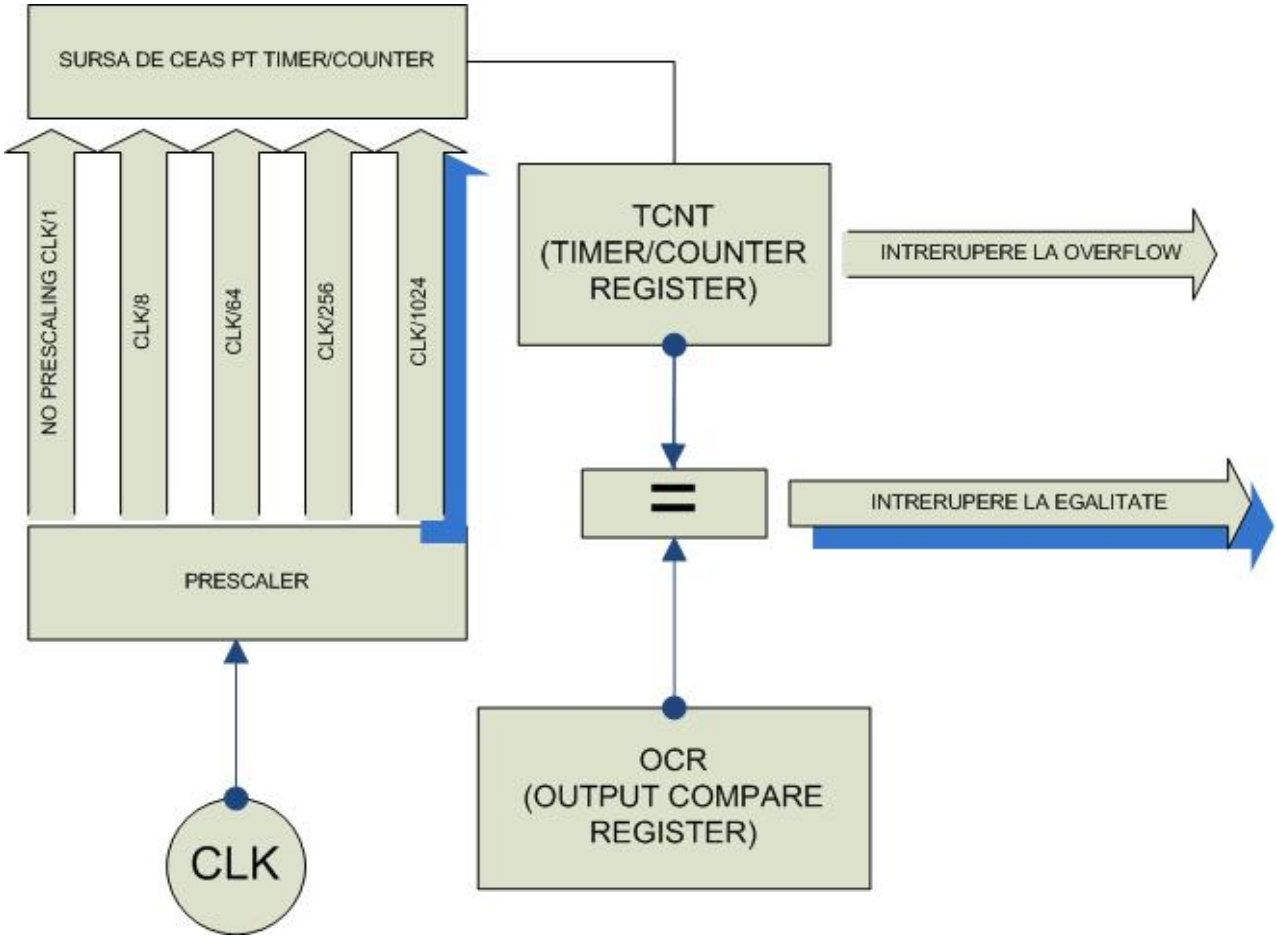


Figura 2. Schita de functionare a unui timer

Principiul de functionare a unui Timer poate fi descris in linii mari de cele trei unitati:

1. Registrul numarator (Timer Counter, TCNT) care masoara efectiv intervalele de timp si care este incrementat automat cu o frecventa data.
2. Avand ceasul intern sau alt ceas conectat pentru a obtine diferite intervale folosim un **"Prescaler"**. Acesta are menirea de a diviza in functie de necesitatile aplicatiei frecventa de ceas si odata cu divizarea sa incrementeze registrul **TCNT**.
3. La fiecare incrementare a TCNT are loc o comparatie intre acest registru si o valoare stocata in registrul **OCRn**. Aceasta valoare poate fi incarcata prin software de utilizator. Daca are loc egalitatea se genereaza o intrerupere, in caz contrar intrementarea continua.

Timerele sunt prevazute cu mai multe canale astfel in paralel se pot desfasura diferite numaratori. ATmega16 este prevazut cu 3 unitati de timer: doua de opt biti si una de numarare pe saisprezece biti.

Principalele caracteristici sunt:

-Design true 16bit(permite 16bit PWM)
-Doua unitati de independente de comparare output(doua canale)
-Unitate de captura input
-Un anulator de zgomot pe intrare
-Stergerea cronometrului la potrivirea compararii (auto reincarcare)
-Puls de corectare a fazelor cu modulator (PWM)
-Perioada PWM variabila
-Generator de frecventa
-Counter extern de evenimente
-Patru surse de intrerupere independente

Majoritatea registrelor si referintelor de biti din aceasta sectiune sunt prezentate intr-o forma generala.Un "n" mic inlocuieste numarul timerului si un "x" mic inlocuieste unitatea de comparare de output. Atunci cand se foloseste registrul sau bitul definit intr-un program , forma precisa trebuie folosita. O diagrama simplificata a timerului este prezentata in schema de mai jos.Plasarea exacta a pinilor I/O se regaseste in figura de mai jos. Registrei I/O accesibili de pe CPU, inclusiv bitii si pinii I/O sunt prezentati cu font bold.

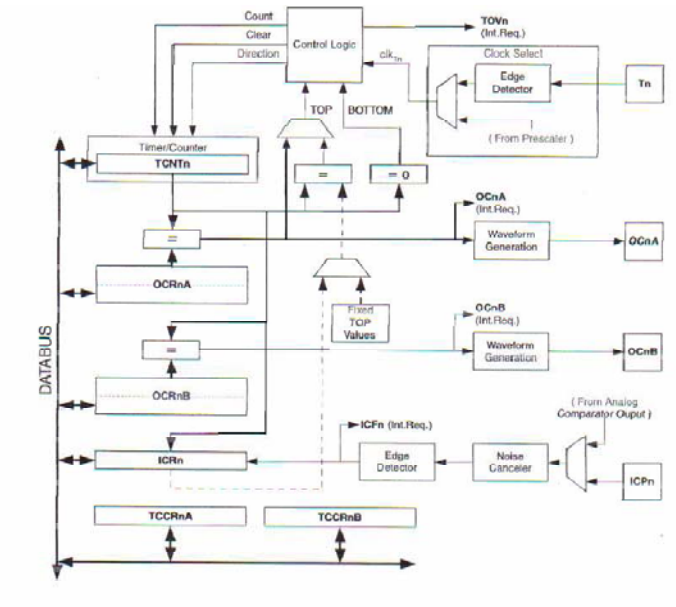


Figure 3.Diagrama bloc a unui timer

Registre:

TCNT1, OCR1A/B si ICR1 sunt toate registre de 16 biti. Proceduri speciale trebuie urmate cand se acceseaza registre de 16 biti.

Registrele de control T/C(TCCR1A/B) sunt registrei de 8 biti si nu au restrictii de acces CPU.Semnalele de cerere de intrerupere sunt vizibile la TIFR (Registrul semnal de intrerupere timer). Toate intreruperile sunt mascate individual in TMISK, TIFR si TMISK(nu sunt prezentati in figura deoarece aceste registre sunt folosite si de alte unitati timer). T/C poate fi masurat intern , prin intermediul prescalei, sau de catre o sursa externa de ceas la pinul T1. Block-ul logic de selectare a ceasului controleaza ce sursa de ceas sau limita de T/C se utilizeaza pentru a creste(sau a descreste) valoarea sa. T/C este inactiv cand nu este selectata nici o sursa de ceas. Output-ul de la selectarea logica a ceasului este numita cronometrarea ceasului(clkT1).

Registrele de comparare ale output-ului cu buffere duble (OCR1A/B) sunt comparate cu valoarea T/C in permanenta. Rezultatul comparatiei poate fi folosit de catre Generatorul de Unde pentru a genera un PWM sau o frecventa variabila de output la pinul de Comparare a Output-ului(OC1A/B). Operatia de comparare va seta si semnalul de potrivire-comparare (OCF1A/B) care poate fi folosit pentru a genera o cerere de intrerupere de comparare output.
Registrul de captura input poate captura valoarea T/C la o comanda data extern fie pe pinul de captura input sau pe pinii analog de comparatie. Unitatea de captura imput include o unitate de filtru digitala pentru a micsora posibilitatile de interferenta a zgomotului.
Valoarea TOP, sau valoarea maxima T/C poate fi definita in cateva moduri de operare fie de catre registrul OCR1A, de catre registrul ICR1 sau de catre un set de valori fixate. Cand se foloseste OCR1A ca valoare TOP intr-un mod PWM, registrul OCR1A nu poate fi utilizat pentru a genera PWM output. In acest caz valoarea TOP va dubla bufferele permitand valorii sa fie schimabata intr-un ciclu de operare. Daca se cere o valoare TOP stabilita, registrul ICR1 poate fi utilizat ca o alternativa, permitand astfel ca OCR1A sa fie folosit ca output PWM.

Definitii

BOTTOM = counterul ajunge la capat cand devine0x0000.
MAX = Counterul ajunge la maxim cand devine 0xFFFF
TOP = Counterul ajunge la valoarea TOP cand devine egal cu cea mai mare valoare din secventa de numarare. Valoarea TOP poate fi desemnata astfel incat sa fie una din aceste valori fixe: 0x00FF, 0X01FF, 0X003FF sau valoarea memorata la Registrei OCR1A sau ICR .Aceasta desemnare depinde de modul de operare.

Accesarea registrelor pe 16 biti:

TCNT1, OCR1A/B si ICR1 sunt registre de 16 biti care pot fi accesate de catre AVR CPU cu ajutorul bus-ului de date de 8 biti. Registrul de 16 biti trebuie sa sa fie accesat folosind doua operatii fie de scriere, fie de citire. **Pentru a executa o scriere de 16 biti, byte-ul HIGH trebuie sa fie scris inaintea byte-ului LOW. Pentru o operatie de citire 16 biti , byte-ul LOW trebuie sa fie citit inainte de byte-ul HIGH.**

I.3 Lucrul cu memoria –SRAM

SRAM-ul este in esenta memoria cu care este inzestrat cipul(ATmega16) . In timp ce registrele generale sunt folosite pentru operatii, SRAM-ul este folosit pentru a depozita date in timpul executiei. Ca sa putem utiliza SRAM-ul trebuie sa stim cateva aspect legate de spatial de adresare, de directivele asm si instructiunile care opereaza pe SRAM.

Toate locatiile din memorie pot fi accesate fie direct fie indirect.

I.3.1Adresarea directa

In cazul incarcarii in memorie sau din memorie de la o adresa directa stim exact de unde sau unde sunt datele .

Exemplu:

Folosind adresa 0x60 ca sa adresam valoarea dintr-un byte(sa-i spunem "digit"), putem folosi instructiunile sts si lds.

lds r16, digit(incarcam in registrul general r16 continutul la adresa 0x60 din memoria SRAM)

sts digit,r16(punem valoarea din registrul general r16 in spatiul de adresa SRAM digit)

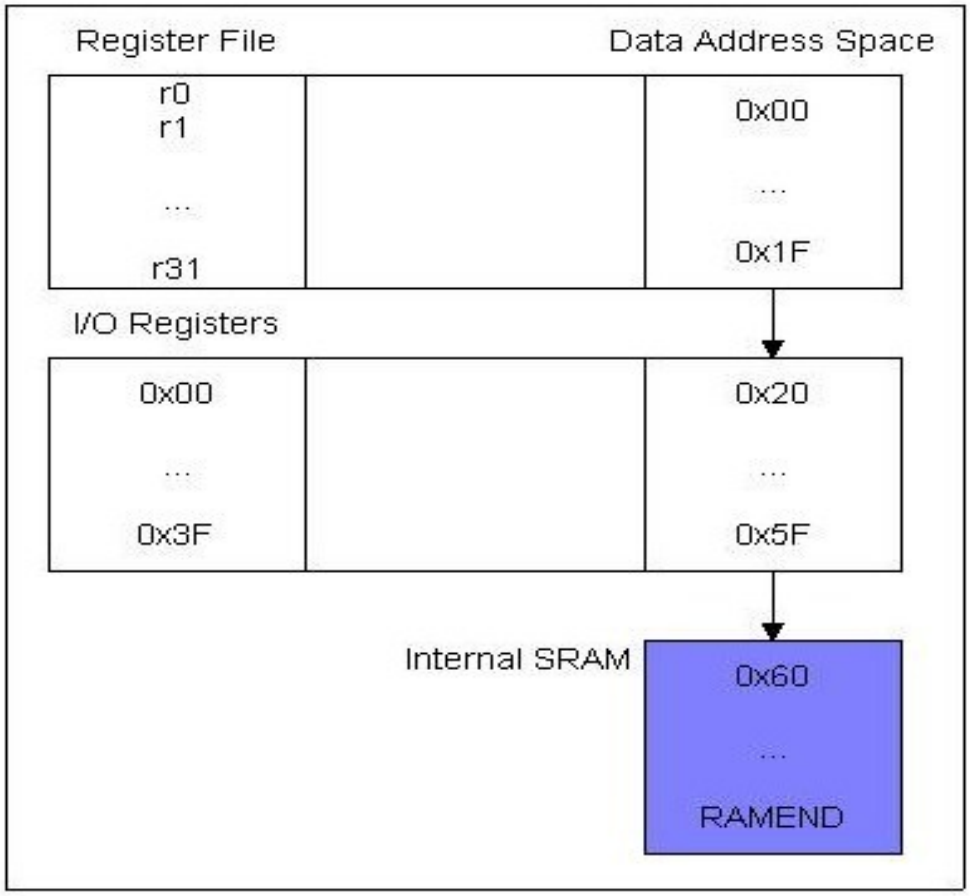
I.3.2 Adresarea indirecta

Adresarea indirecta se face similar cu cea cu ajutorul pointerilor in C sau Pascal: perechile de registre pointeri (r26:r27 sunt numiti X, r28:r29 Y, r30:r31 Z) pot fi folositi sa faca referire la spatial de adresare

AVR. Daca de exemplu X(r28:r29) ia ca valoare 0x60 el va pointa la "digit" si poate fi folosit pentru a face diferite operatii pe valoarea lui "digit".

Exemplu:

```
ldi XL, 0X60 ;incarcam  in r26(XL) cu valoarea low(0x60)

ldi XH, 0X00 ; incarcam  in r27(XH) cu valoarea high(0x60)

ld r16,X;    incarca in r16 valoarea lui X, care pointeaza la 0x60
```



In diagrama de mai sus observam ca prima adresa de SRAM este 0x60.

Exemplu (diferenta intre adresarea indirect si adresarea directa):

Stim ca registrul de I/O PORTA se gaseste in tabelul de registre :

$1B ($3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	66

-Adresa la care gasim continutul registrului PORTA este $3B.

-Spatiul de adresare in registrele de I/O este 0x00.

Luand in considerare aceste doua aspecte putem face diferentierea intre modul indirect de adresare si cel direct:

Indirect:

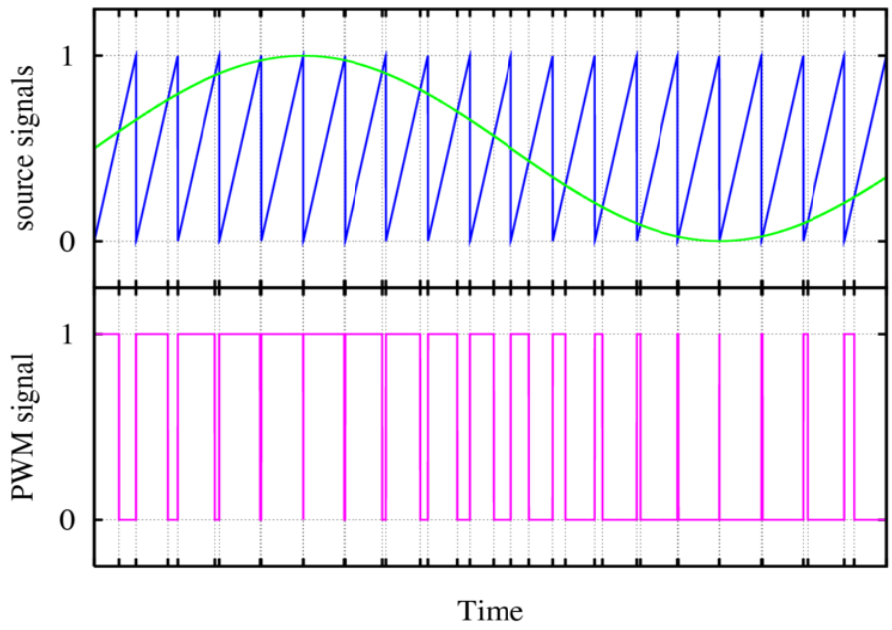
```
ldi XL, 0x3B

ldi XH,0x00

lds r16,X

out 0x1B, r16
```

Instructiunile menite pentru SRAM sunt foarte folositoare in cazul utilizarii de stringuri sau in alte cazuri in care datele sunt depozitate succesiv element dupa element (acesta este cazul mastilor de cifre din laboratorul doi). In acest caz putem opera atat asupra unitatii de date cat si asupra indexului aferent.

`st X+,r16;` continutul registrului general r16 este memorat la adresa la care pointeaza X iar X este incrementat cu "1"

PWM(aspecte teoretice)

Pulse Width Modulation(in traducere libera Modularea in latimea pulsului) este folosita pentru a manipula circuite cu caracter analogic dintr-un domeniu digital. Aceasta modulatie vine in sprijinul multor aplicatii deoarece inseamna consum mic, eliminarea zgomotului si aplicatii cu cost redus. Aplicatiile care includ PWM inseamna : controlul, conversia, masurare.



Principiul de functionare

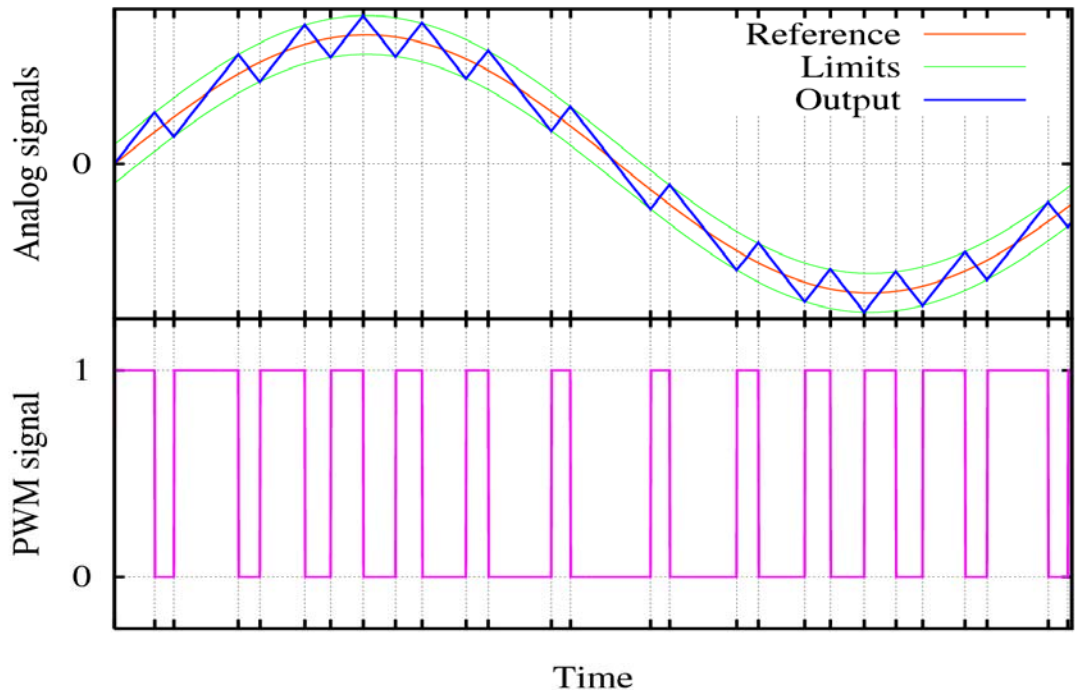
Aceasta modulare foloseste o unda patratica cu ciclu de utilizare rezultand intr-o variatie a valorii medii a formei de unda. Considearand o forma de unda patratica f(t) cu o valoare minima y_{min} si o valoare maxima ymax si un ciclu de utilizare D(ca in figura) valoarea medie a formei de unda e data de relatia:

$$\bar{y} = \frac{1}{T}\int_0^T f(t)\,dt$$

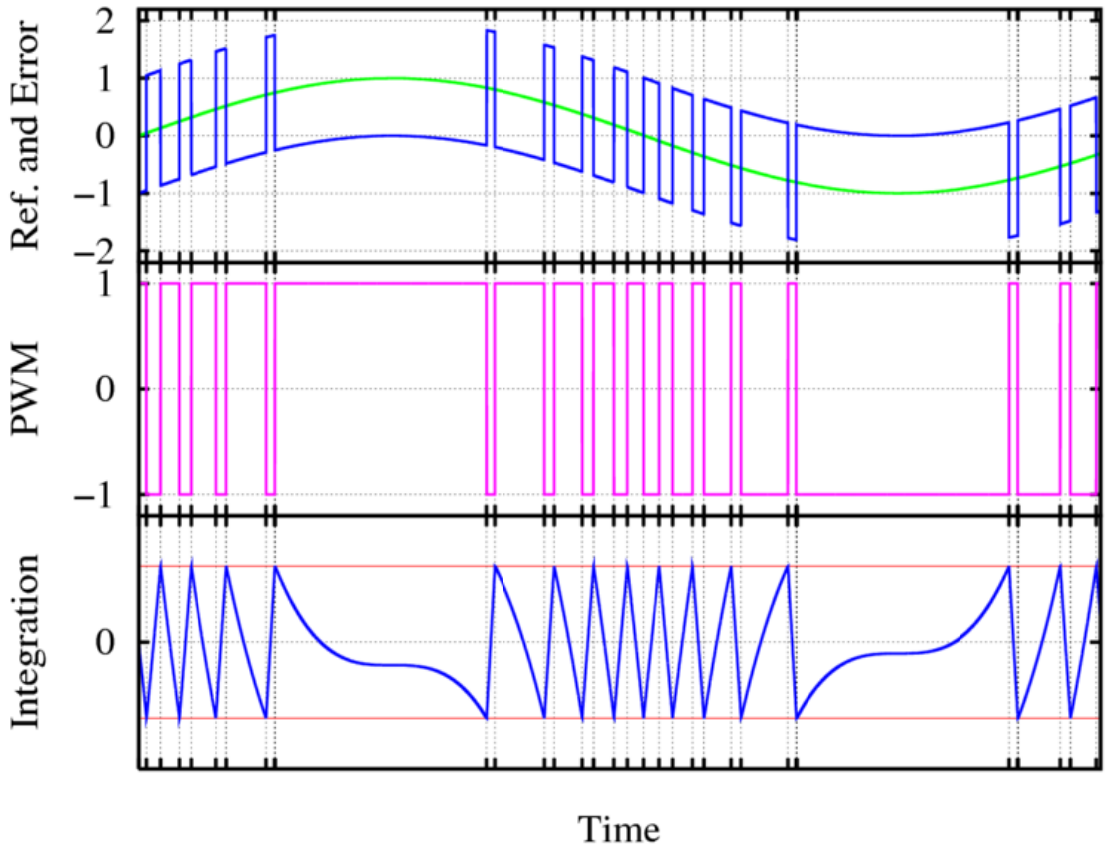
, cum f(t) este o forma de unda patratica valoarea sa maxima se atinge pentru 0<t<D*T .

Tipuri de PWM(modulatie in latimea pulsului)

-Modulatie Delta



Modulatie Delta-Sigma



PWM-Modulatie Digitala

Multe circuite digitale pot genera semnale PWM. Majoritatea microcontrollerelor dispun de aceasta facilitate. Pentru a implementa o asemenea facilitate ele se folosesc un numerator care este incrementat periodic (conectat direct sau indirect la o unitate de ceas) si care este resetat la sfarsitul fiecarei perioade a PWM-ului. Cand valoarea numaratorului este mai mare decat valoarea de referinta, iesirea PWM(output-ul) trece din starea inalta in stare joasa (sau invers).

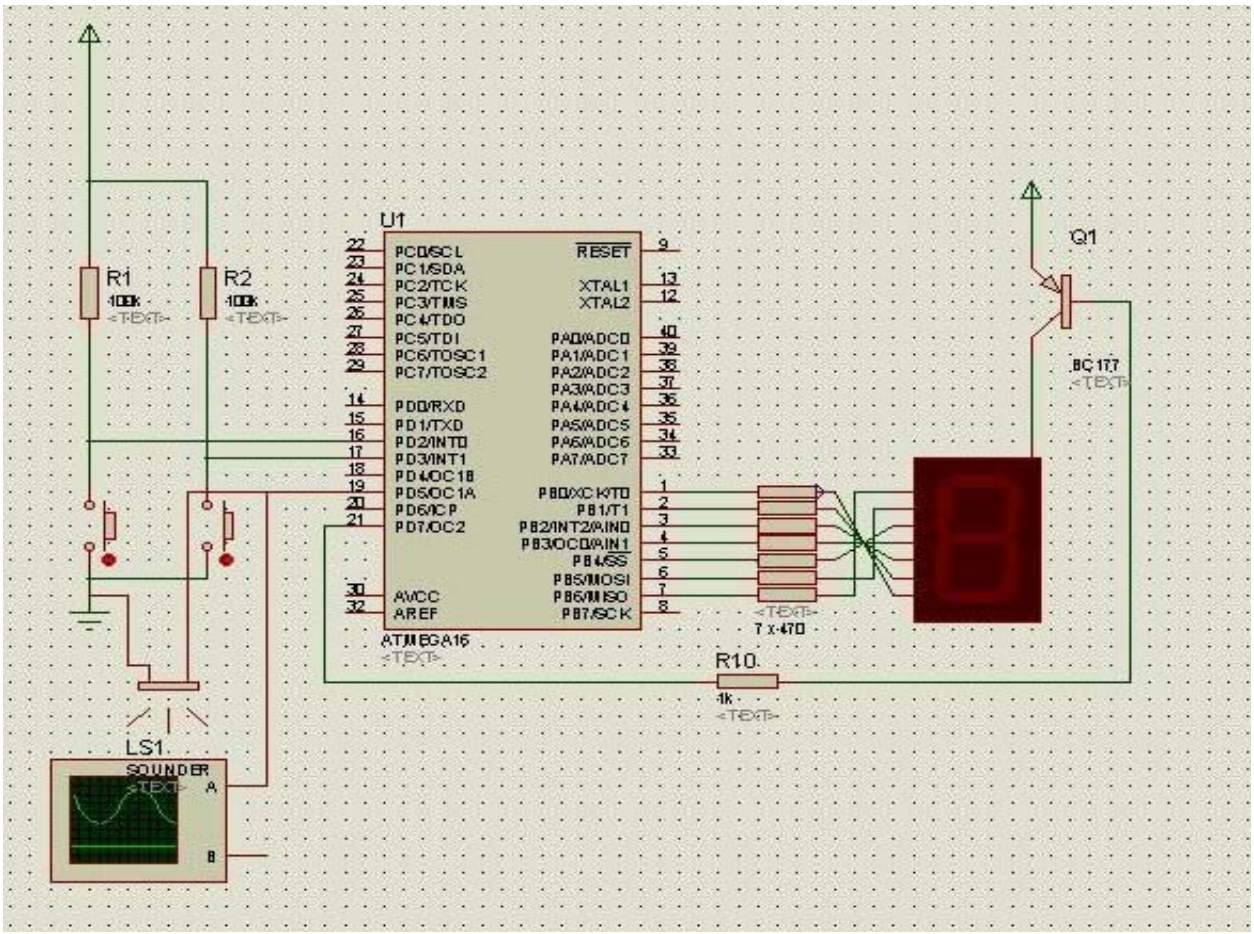
Numaratorul incrementat si periodic resetat este versiunea discreta a metodei de intersectie e semnalului dinte de fierastrau ("intersecting method 's sawtooth"). Comparatorul analog se traduce in acest caz intr-o comparere de intregi intre valoarea numaratorului curent si valoarea digitala curenta de referinta.

Aplicatii :

http://au.truveo.com/tag/pwm

II. LUCRARE PRACTICA

II.1 Pornim prin a deschide schema de simulare in ISIS PROTEUS:



In schema de mai sus este reprezentat schematic suportul hardware pe care trebuie sa-l interfatam in acest laborator.Dupa cum se observa, cu ajutorul microcontrollerului si mai precis folosind facilitatile oferite de: sistemul de intreruperi, timere, memoria microcontrollerului, vom interfata un afisaj cu leduri, un buzzer(sounder in schema) si doua butoane.

II.2 Logica aplicatiei acestei scheme se vrea implementata in acest laborator in felul urmator:

toDo1: scrieti in cod instructiunea de salt la rutina de tratare a intreruperii cmp_handler;

toDo2: initializati portul B avand in vedere ce e cuplat pe acest port;

toDo3: initializati portul D avand in vedere natura sa mixta;(folosirea instructiunii sbi)

toDo4: setati registrele Timerului pe 16 biti din cod corespunzator cazului (comportamentul dorit) ;(pag 111 DataSHEET)

-TCCR1A/TCCR1B ->tabela 44(compare output mode-nonPWM)

-OCR1A (16 biti)(folosirea HIGH(…)/LOW(…))

toDo5: realizati rutina cmp_handler;

->salvarea contextului(SREG eventualele register folosite)

->se executa 1/sec(auto-apelare, lucrul cu variabila digit care e stocata in SRAM)

-> afisare cu print_digit(folosirea instructiunilor lds /sts)

->refacerea contextului