Universitatea Tehnică din Cluj-Napoca
Departamentul Calculatoare

Proiectarea cu Micro-Procesoare

Lector: Mihai Negru

An 3 – Calculatoare și Tehnologia Informației

Seria B

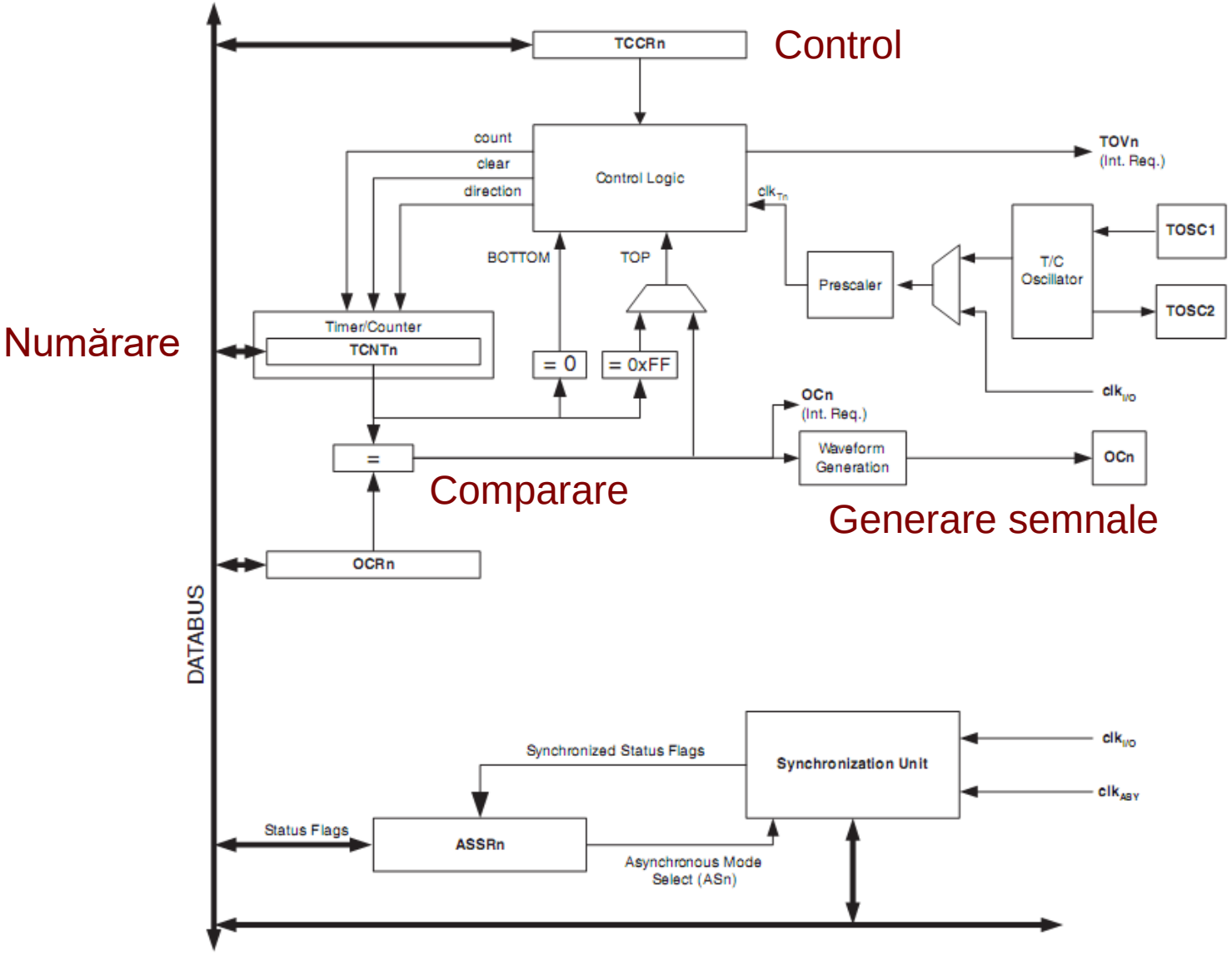
Curs 4: Temporizatoare

http://users.utcluj.ro/~negrum/

Temporizatoare (Timers)

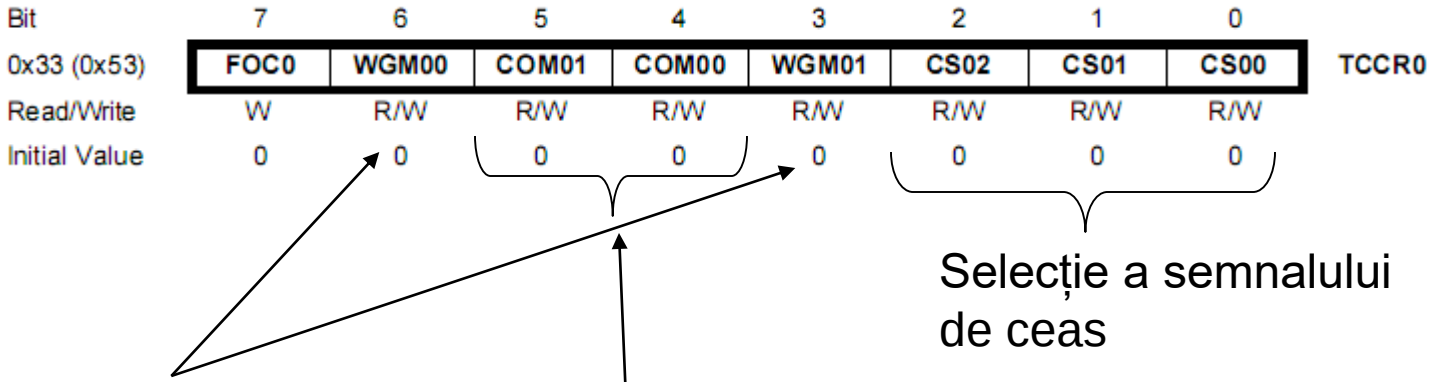
- **Resurse ATMega 2560**
  - Două numărătoare/ temporizatoare pe 8 biți (Timer 0, Timer 2)
  - Patru numărătoare / temporizatoare pe 16 biți (Timer 1, 3, 4, 5)
- **Caracteristici**
  - Alegerea frecvenței ceasului la intrarea temporizatoarelor (prescaler)
  - Citire / scriere stare numărator
  - Generare de forme de undă prin folosirea unui registru de comparare
    - Reglare frecvență și factor de umplere – PWM (pulse width modulation)
  - Generare de cereri de întrerupere la intervale regulate
  - Declanșare la un eveniment extern (capture)
- **Utilizări**
  - Generarea diferitelor forme de undă
  - Sincronizarea programului cu intervale de timp regulate
  - Măsurare intervale de timp

Structura temporizator 8 biți



Configurare temporizator 8 biți

- **Registrul TCCRn – controlează comportamentul temporizatorului**



Selecție a semnalului de ceas

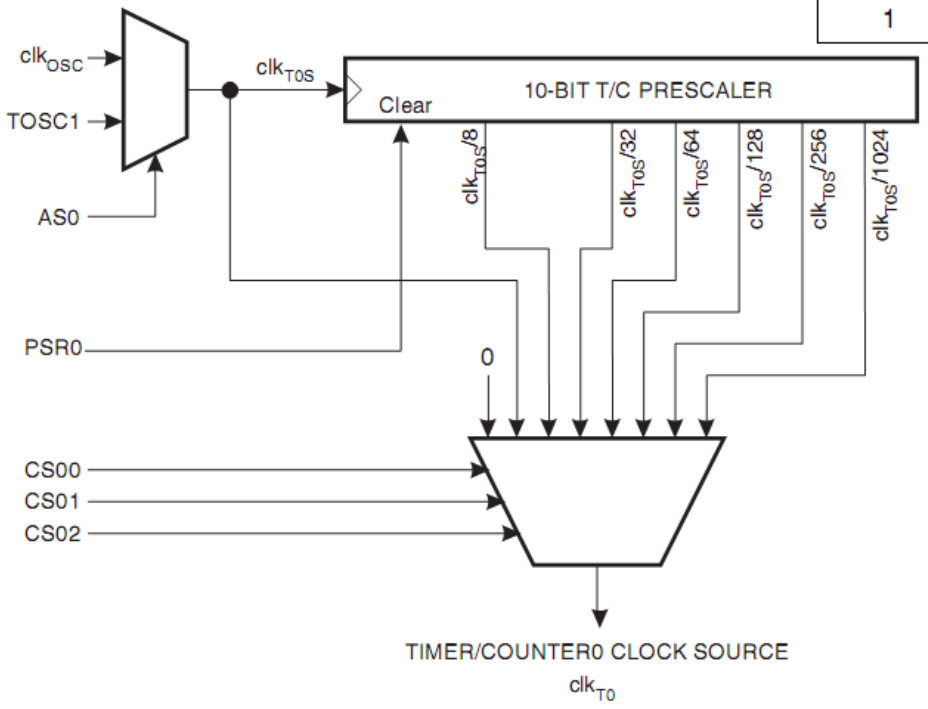
Controlul modului de generare a undelor

Modul de utilizare a rezultatului unității de comparare – depinde de modul de generare a undelor

Selecția semnalului de ceas

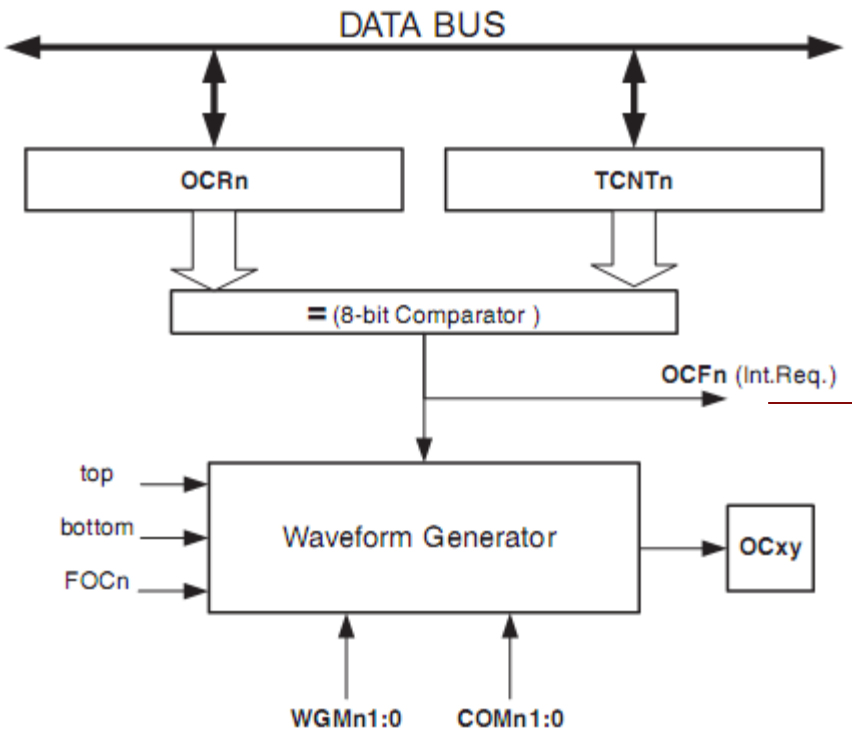
- Biții CS02:CS00 selectează divizarea frecvenței ceasului la intrarea in numărator
- Reglarea vitezei de numarare

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/counter stopped)
0	0	1	clk_{T0S}/(No prescaling)
0	1	0	clk_{T0S}/8 (From prescaler)
0	1	1	clk_{T0S}/32 (From prescaler)
1	0	0	clk_{T0S}/64 (From prescaler)
1	0	1	clk_{T0S}/128 (From prescaler)
1	1	0	clk_{T0S}/256 (From prescaler)
1	1	1	clk_{T0S}/1024 (From prescaler)



Unitatea de comparare

- Comparatia dintre registrul care conține valoarea numărătorului, **TCNT0**, si registrul valorii de comparat, **OCR0**, este folosită pentru a genera diferite forme de undă

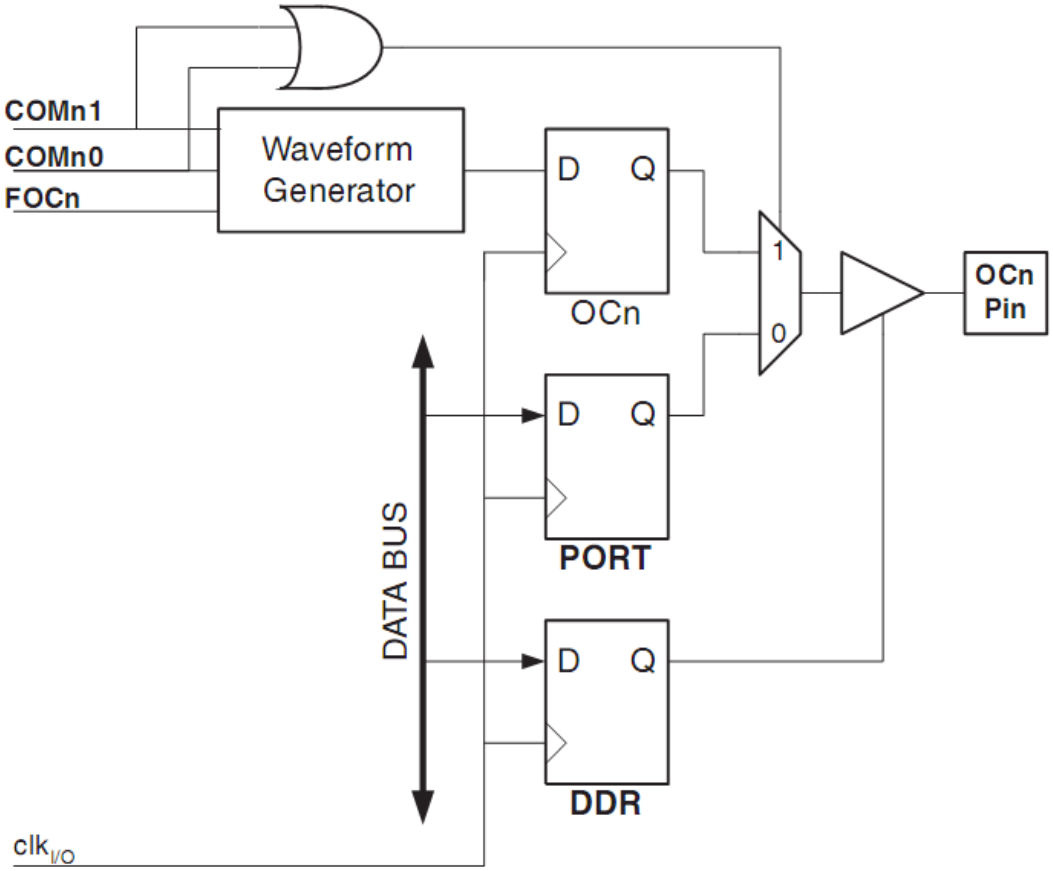


Output compare flag – la producerea egalității acest semnal cere întrerupere

Bit de output compare – aici va fi generată unda

Unitatea de comparare

- Undele generate sunt vizibile la exterior prin pinii porturilor I/O
- Bitul din portul I/O corespunzator bitului OCn trebuie configurat ca ieșire



Semnalul OC0 este comun cu bitul 4 din portul B

Tipuri de unde (moduri de functionare)

Mode	WGM00:01	Mode
0	00	Normal
1	10	PWM, Phase Correct
2	01	CTC
3	11	Fast PWM

Biții WGM00:01 în combinație cu biții COM1:0 definesc comportamentul temporizatorului!

Normal, CTC

COM0[1:0]	Description
00	Normal, OC0 disconnected
01	Toggle OC0 on compare match
10	Clear OC0 on compare match
11	Set OC0 on compare match

PWM, Phase Correct

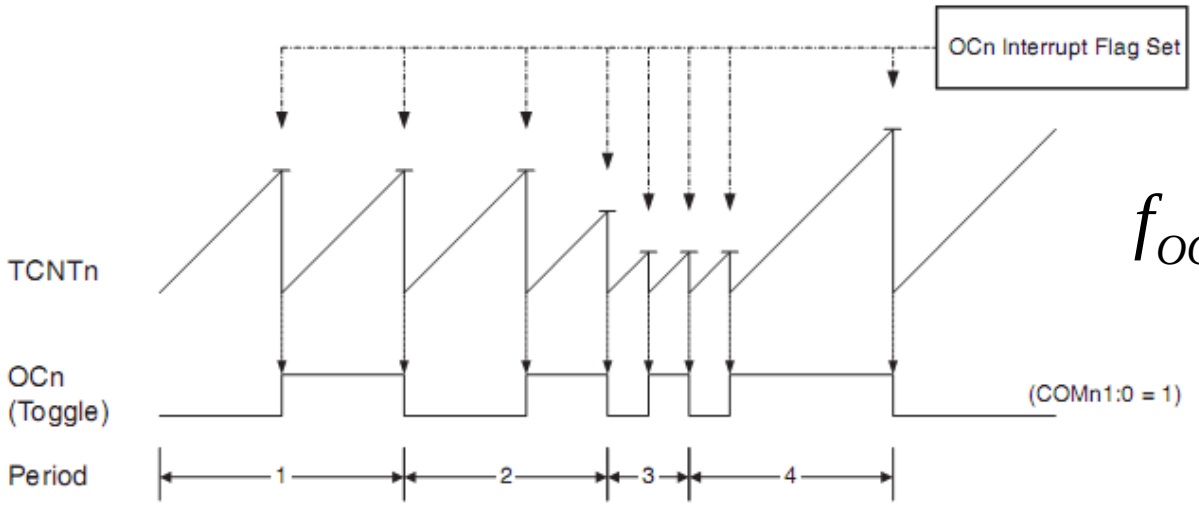
COM0[1:0]	Description
00	Normal, OC0 disconnected
01	Reserved
10	Clear OC0 on compare match when up-counting. Set OC0 on compare match when down counting
11	Set OC0 on compare match when up-counting. Clear OC0 on compare match when down counting.

Fast PWM

COM0[1:0]	Description
00	Normal, OC0 disconnected
01	Reserved
10	Clear OC0 on compare match, set OC0 at TOP
11	Set OC0 on compare match, clear OC0 at TOP

Tipuri de unde (moduri de functionare)

- **Normal**
  - Numărare simplă: 0 → 255
  - Când se produce saturarea → întrerupere de overflow, numărarea se reia de la 0
- **CTC – Clear Timer on Compare Match:** 0 → OCR0
  - Cand valoarea număratorului (TCNT0) ajunge la valoarea din OCR0, se produce resetarea
  - Frecvența undelor generate se poate regla prin scrierea registrului OCR0
    - COM01:COM00 – setați pe 01 → basculare OC0 la egalitate

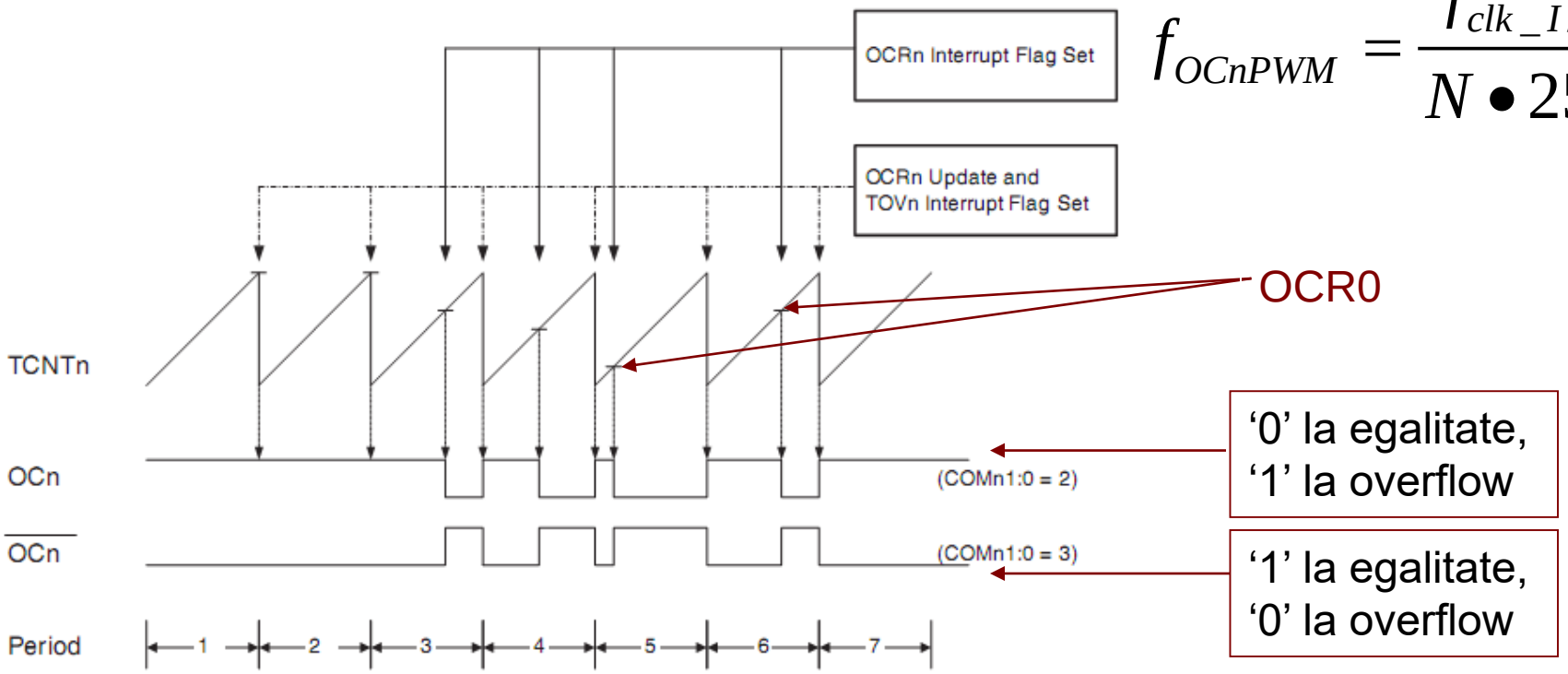


$$f_{OCn} = \frac{f_{clk\_I/O}}{2N(1+OCR_n)}$$

Tipuri de unde (moduri de functionare)

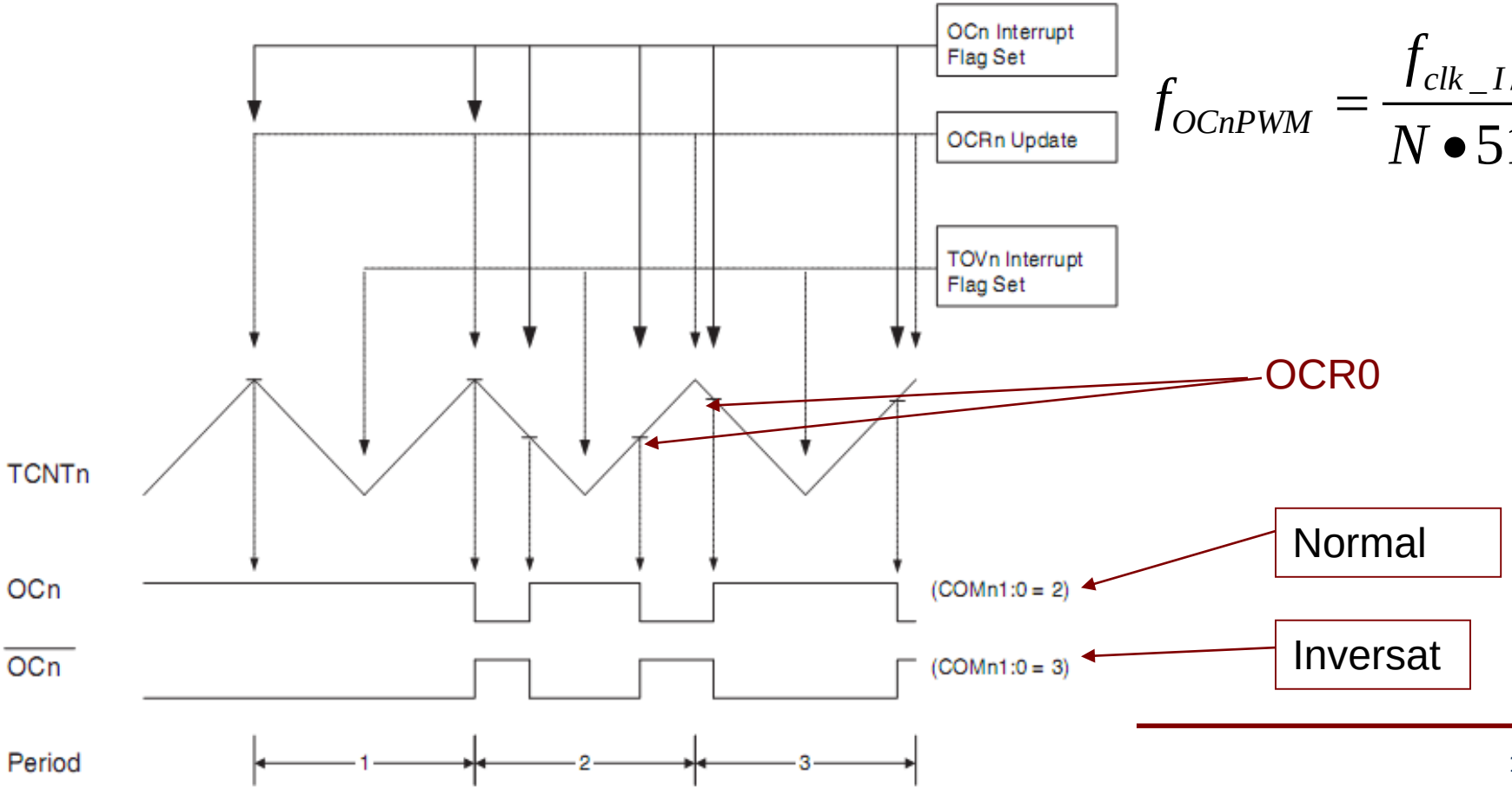
- **Fast Pulse Width Modulation (PWM)**
  - Generare semnale PWM (modulare in latimea pulsului) pe OC0
  - Factorul de umplere se regleaza prin scrierea registrului de comparare OCR0
  - Frecventa este fixa, data de bitii de clock select
  - Factorul de umplere = OCR0 / 255

$$f_{OCnPWM} = \frac{f_{clk\_I/O}}{N \bullet 256}$$



Tipuri de unde (moduri de functionare)

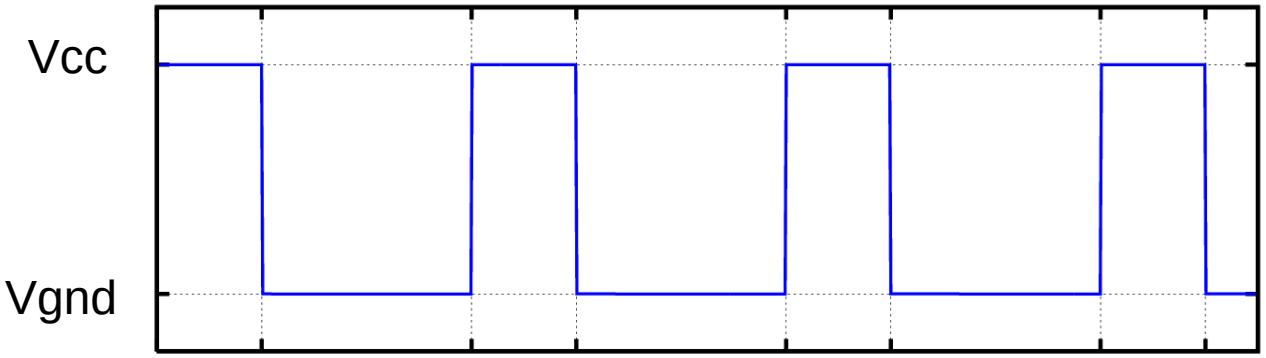
- **Phase Correct PWM mode –** Generare semnale PWM cu corecția fazei
  - Pulsul este simetric fața de mijlocul perioadei
  - Factorul de umplere se reglează prin scrierea registrului de comparare OCR0
  - Numărare crescătoare / descrescătoare, ieșirea se modifică la egalități succesive
  - Factorul de umplere = OCR0 / 255



$$f_{OCnPWM} = \frac{f_{clk\_I/O}}{N \bullet 510}$$

Pulse Width Modulation (PWM)

- Unele sisteme necesită control prin variația tensiunii de intrare
- Exemplu: turația unui motor, intensitatea unui LED
- Sistemele digitale pot produce la ieșire doar valori de 0 (GND) și 1 (Vcc)
- Factorul de umplere D (duty cycle)



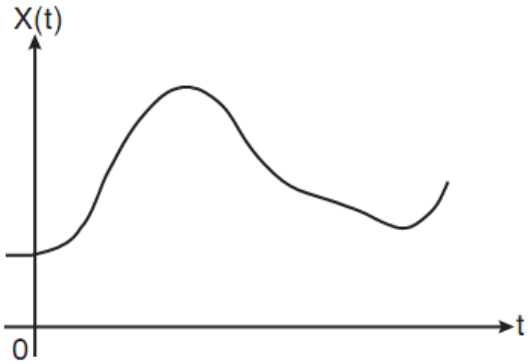
$$D = \frac{T_{on}}{T_{on} + T_{off}}$$

- Tensiunea medie – poate fi oricât intre cele doua extreme, depinde de D

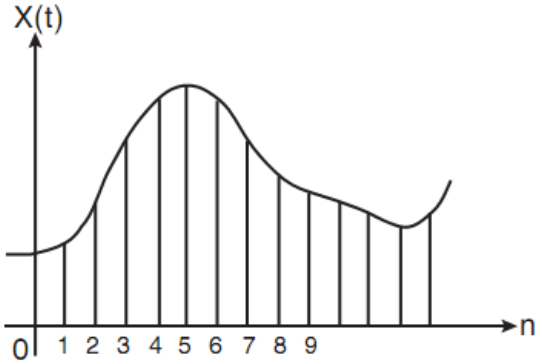
$$V_{MED} = DV_{CC} + (1-D)V_{GND}$$

Pulse Width Modulation (PWM)

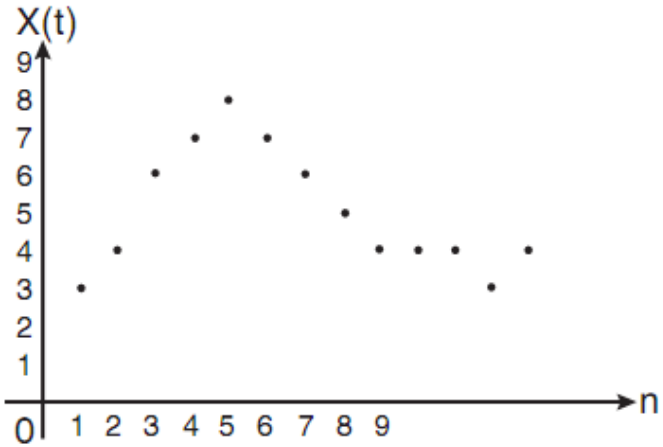
- Semnale analogice de joasă frecventa (ex. Sunet) pot fi codificate PWM
- Pași: eșantionare, digitizare, calculare D pe baza valorii digitale



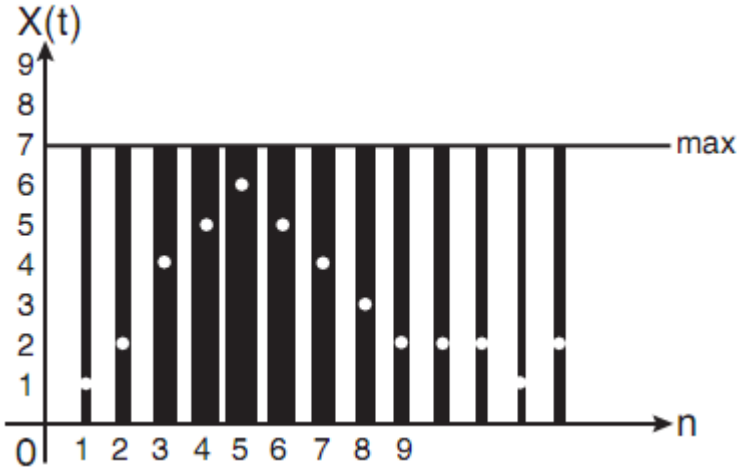
Semnal analogic



Eșantionare



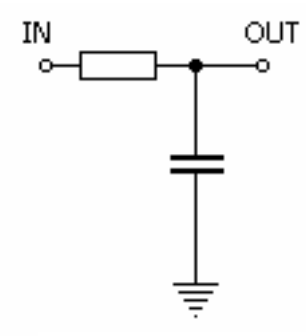
Digitizare a eșantioanelor



Calculare a factorului de umplere

Pulse Width Modulation (PWM)

- Utilizarea semnalului PWM
  - Se poate utiliza ca atare, dacă aplicația permite: variația luminozității unui bec, turația unui motor, etc – inerția dispozitivelor produce efectul de mediere a tensiunii
  - Se poate filtra printr-un filtru trece-jos, pentru a reconstrui semnalul analogic
    - Se reglează frecvența limită superioară (*cutoff frequency*) pentru domeniul de frecvență al semnalului analogic, care este mai mic decât frecvența semnalului purtător
    - Filtru trece jos simplu – filtrul RC



$$f_{cutoff} = \frac{1}{2\pi RC}$$

Întreruperi generate de temporizatoare

- Evenimentele care generează întreruperi:
  - Saturare (*overflow*)
  - Egalitate la comparare (*compare match*)
  - Eveniment extern (captură) – disponibil doar la temporizatoarele de 16 biți

Adresa		Descriere
0x0012	TIMER2 COMP	Timer/Counter2 Compare Match
0x0014	TIMER2 OVF	Timer/Counter2 Overflow
0x0016	TIMER1 CAPT	Timer/Counter1 Capture Event
0x0018	TIMER1 COMPA	Timer/Counter1 Compare Match A
0x001A	TIMER1 COMPB	Timer/Counter1 Compare Match B
0x001C	TIMER1 OVF	Timer/Counter1 Overflow
0x001E	TIMER0 COMP	Timer/Counter0 Compare Match
0x0020	TIMER0 OVF	Timer/Counter0 Overflow
0x0030(3)	TIMER1 COMPC	Timer/Countre1 Compare Match C
0x0032(3)	TIMER3 CAPT	Timer/Counter3 Capture Event
0x0034(3)	TIMER3 COMPA	Timer/Counter3 Compare Match A
0x0036(3)	TIMER3 COMPB	Timer/Counter3 Compare Match B
0x0038(3)	TIMER3 COMPC	Timer/Counter3 Compare Match C
0x003A(3)	TIMER3 OVF	Timer/Counter3 Overflow

Întreruperi generate de temporizatoare

- Activare/ dezactivare întreruperi – registrul TIMSK (accesibil cu instrucțiuni I/O)

Bit	7	6	5	4	3	2	1	0	
0x37 (0x57)	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
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	

Saturare Timer 0

Egalitate comparare pt. Timer 0

- Accesare stare întreruperi – registrul TIFR

Bit	7	6	5	4	3	2	1	0	
0x36 (0x56)	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	

- Utilizare posibilă a întreruperilor
  - Asigurarea unui interval regulat între diferite acțiuni programate – exemplu, comutarea intre celulele unui SSD
  - Generare forme de undă software
  - Modificare parametri pentru generarea undelor hardware

Exemplu 1

- Generare semnal de o anumită frecvență specificată
  - Problema: să se genereze un semnal cu frecvența de 50 Hz
  - Modul de lucru ales: CTC (Clear on Compare Match) – permite reglarea perioadei prin setarea registrului de comparare
  - Calculul frecvenței:

$$f_{OCn} = \frac{f_{clk\_I/O}}{2N(1+OCR_n)}$$

  - f_{OCn} = 50
  - N = 1024 → divizarea maximă de frecventă permisă din prescaler
  - fclk_I/O = 16 MHz, frecvența plăcii
  - OCR0 = 16000000 / (2*1024*50) – 1 = 155

Exemplu 1

- Generare semnal de frecvență specificată – cod sursa

```
.include "m64def.inc"
.org 0x0000
    jmp reset

reset:
  ldi r16, 0b00011111
  out TCCR0, r16; configurare Timer0

  ldi r16, 155      ; OCR calculat
  out OCR0, r16

  ldi r16, 0xff
  out DDRB, r16 ; activeaza iesirea OC0 (PB4)

loop:
rjmp loop
```

Mode	WGM00:01	Mode
0	00	Normal
1	10	PWM, Phase Correct
2	01	CTC
3	11	Fast PWM

Normal, CTC

COM0[1:0]	Description
00	Normal, OC0 disconnected
01	Toggle OC0 on compare match
10	Clear OC0 on compare match
11	Set OC0 on compare match

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/counter stopped)
0	0	1	clk_{T0S}/(No prescaling)
0	1	0	clk_{T0S}/8 (From prescaler)
0	1	1	clk_{T0S}/32 (From prescaler)
1	0	0	clk_{T0S}/64 (From prescaler)
1	0	1	clk_{T0S}/128 (From prescaler)
1	1	0	clk_{T0S}/256 (From prescaler)
1	1	1	clk_{T0S}/1024 (From prescaler)

Exemplu 2

- Utilizarea întreruperii la egalitatea comparației
  - Problema: să se genereze un semnal cu perioada de 1 secundă – imposibil prin reglarea directă a timerelor, frecvența e prea joasa
  - Vom utiliza configurația stabilită in exemplul 1
  - Frecventa rezultată – 50 Hz, cu modul Toggle on CTC, implica două egalități în 1/50 secunde
  - Vom folosi întreruperea declansată la egalitate dintre numarator si OCR
  - Vom bascula (toggle) un semnal la fiecare 50 de astfel de evenimente
  - Acest semnal are perioada de 1 secundă

Exemplu 2

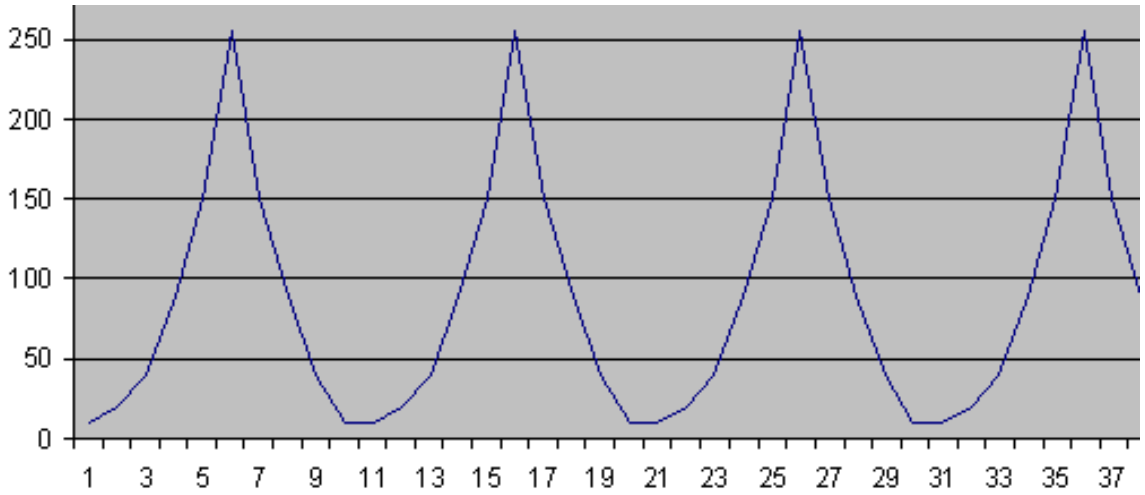
```
.include "m64def.inc"
.org 0x0000
    rjmp reset

.org 0x001E                 ; adresa vectorului pentru intreruperea Timer0 Comp
    rjmp timercpm

reset:
    ldi r16, low(RAMEND)    ; intreruperile necesita stiva
    out SPL, r16
    ldi r16, high(RAMEND)
    out SPH, r16

    ldi r16, 0b00011111     ; configuratia anterioara
    out TCCR0, r16
    ldi r16, 155
    out OCR0, r16

    ldi r16, 0xff           ; folosim LED-urile ca iesire
    out DDRE, r16
```

Exemplu 2

```
    ldi r16, 0b00000010  ; activare punctuala intrerupere Timer0 Comp
    out TIMSK, r16

    ldi r18, 0           ; numaratorul de evenimente
    ldi r19, 0           ; registru folosit pentru bascularea semnalului
    sei                  ; activate globala intreruperi

loop:
    rjmp loop

timercpm:                ; rutina de tratare a intreruperii
    inc r18              ; incrementare contor evenimente
    cpi r18, 50
    brne exit            ; nu a ajuns la 50, iesire
    com r19              ; basculare r19
    out PORTE, r19       ; afisare
    ldi r18, 0           ; reinitializare contor
exit:
    reti
```

Exemplu 3

- Folosirea PWM
  - Problema: să se genereze o funcție analogică oarecare
  - Funcția va fi definită prin nivele discrete, stocate intr-un LUT
  - Poate fi rezultat al unui proces de conversie Analog/Digital
  - Utilizăm modul de lucru PWM phase correct
  - OCR0 va defini lățimea pulsului
  - Vom modifica OCR0 la finalul fiecărei perioade de numărare
  - Valorile unei perioade: 10, 20, 40, 90, 150, 255, 150, 90, 40, 20, 10



Exemplu 3

```
.include "m64def.inc"
.org 0x0000
    rjmp reset
.org 0x0020        ; adresa vector Timer0Ovf
    rjmp timerovf

reset:
    ldi r16, low(RAMEND)
    out SPL, r16
    ldi r16, high(RAMEND)
    out SPH, r16

    ldi r16, 0b01100001 ; configurare Timer0
    out TCCR0, r16
    ldi r16, 0xff
    out DDRB, r16          ; activare iesire OC0

    ldi r16, 0b00000001 ; activare intrerupere
    out TIMSK, r16
```

Mode	WGM00:01	Mode
0	00	Normal
1	10	PWM, Phase Correct
2	01	CTC
3	11	Fast PWM

PWM, Phase Correct

COM0[1:0]	Description
00	Normal, OC0 disconnected
01	Reserved
10	Clear OC0 on compare match when up-counting. Set OC0 on compare match when down counting
11	Set OC0 on compare match when up-counting. Clear OC0 on compare match when down counting.

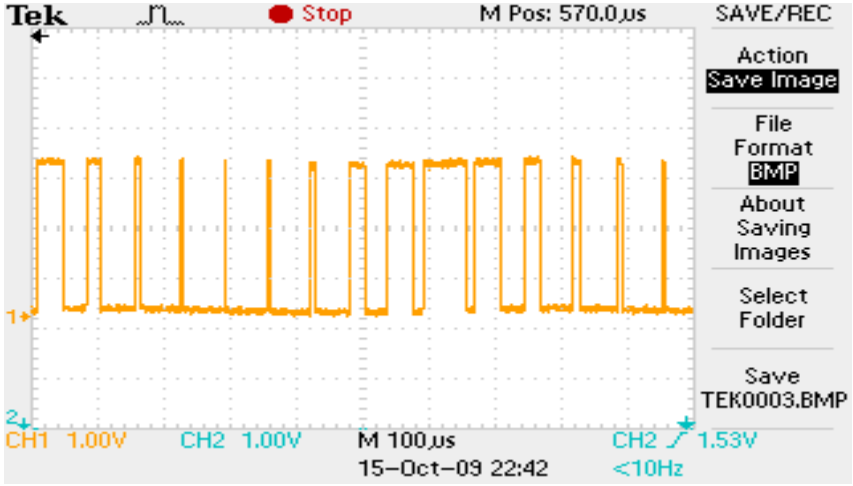
CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/counter stopped)
0	0	1	clk_{T0S}/(No prescaling)
0	1	0	clk_{T0S}/8 (From prescaler)
0	1	1	clk_{T0S}/32 (From prescaler)
1	0	0	clk_{T0S}/64 (From prescaler)
1	0	1	clk_{T0S}/128 (From prescaler)
1	1	0	clk_{T0S}/256 (From prescaler)
1	1	1	clk_{T0S}/1024 (From prescaler)

Exemplu 3

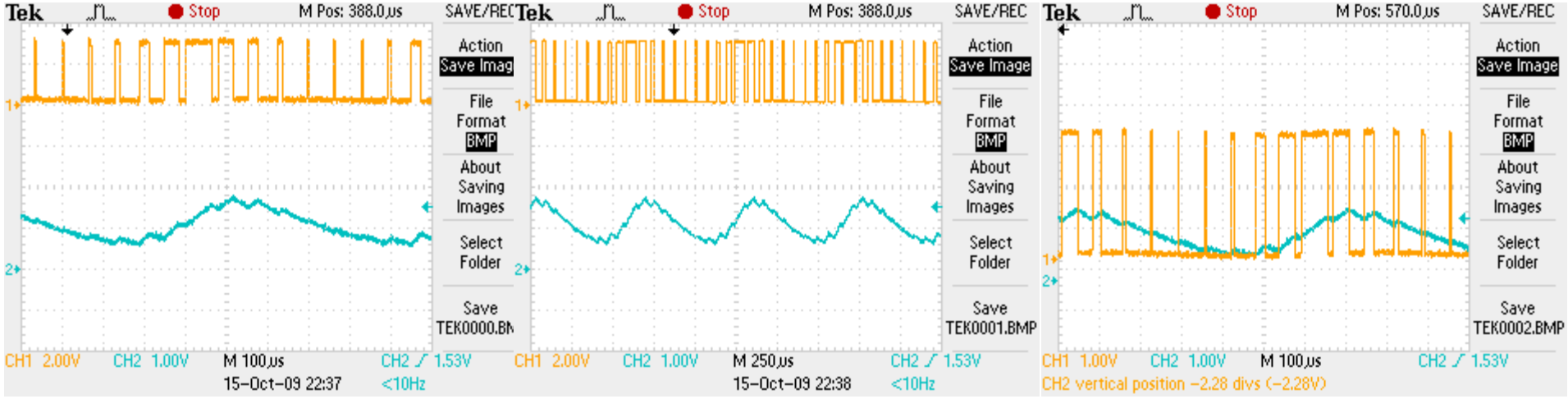
```
    ldi r18, 0          ; iteratorul tabelei de valori
    sei                 ; activare globala intreruperi

loop: rjmp loop         ; programul principal se blocheaza aici

timerovf:               ; Timer0Ovf ISR – se apeleaza la sfarsitul unei perioade
    ldi r17,0           ; calcul adresei in LUT
    ldi zh, high(2*translut)
    ldi zl, low(2*translut)
    add zl, r18
    adc zh, r17
    lpm r17, Z
    out OCR0, r17 ; valoarea din LUT se introduce in OCR
    inc r18
    cpi r18, 10         ; adresa maxima din LUT
    brne exit
    ldi r18,0
exit:
    reti
translut:               ; tabela de valori, aici se defineste functia
.db 10, 20, 40, 90, 150, 255, 150, 90, 40, 20, 10
```

Exemplu 3

- Folosirea PWM – rezultate



- Prin atașarea unui filtru RC, R = 1K, C = 0.22 µF, se obține unda analogică



Exemplu 4

- Citirea stării temporizatorului
  - Problema: măsurarea perioadei dintre două evenimente externe
  - Configurația aleasa – Normal, cu frecvență minimă de numărare
  - Evenimentul extern provoacă o întrerupere externă, pe tranziția 0-1

```
.include "m64def.inc"
.org 0x0000
    rjmp reset
.org 0x0002                 ; adresa vector Intrerupere externa 0
    rjmp int0ISR

reset:
    ldi r16, low(RAMEND)
    out SPL, r16
    ldi r16, high(RAMEND)
    out SPH, r16

    ldi r16, 0b00000111     ; configurare Timer0
    out TCCR0, r16
    ldi r16, 0xff
    out DDRE, r16           ; activare iesire port E - leds
```

Exemplu 4

```
    ldi r16, 0b00000011     ; configurare mod de tratare INT0 – front crescator
    sts EICRA, r16
    ldi r16, 0b00000001     ; activare intrerupere INT0
    out EIMSK, r16

    ldi r17, 0              ; ultima valoare numarator
    sei

loop: rjmp loop

int0ISR:
    in r16, TCNT0           ; citirea starii registrului numarator
    mov r18, r16
    sub r16, r17            ; diferenta fata de valoarea precedenta
    mov r17, r18            ; noua stare devine vechea stare
    out PORTE, r16          ; afisam diferenta
    reti
```

Referințe

1. Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V datasheet
2. Atmel Atmega64 datasheet