

Proiectare cu Microprocesoare

Curs 4

An 3 cti

An universitar 2018/2019

Semestrul 1

Lector: Radu Dănescu

Temporizatoare

Temporizatoare AVR

- Temporizatoare/numărătoare pe 8 biți
- Temporizatoare/numărătoare pe 16 biți

Caracteristici

- Divizor de frecvență pentru semnalul de ceas de intrare (prescaler)
- Starea numărătorului poate fi citită și scrisă
- Generare de unde folosind un registru de comparație
 - Reglarea frecvenței și a factorului de umplere (PWM)
- Generare de cereri de întrerupere la intervale regulate de timp
- Răspuns la evenimente externe (input capture)

Utilizare

- Generare de semnale
- Sincronizarea programului la intervale de timp regulate
- Măsurarea intervalelor de timp

Temporizatoare

Atmega 328P

1x 8 bit Timer0 cu PWM, 1x 8 bit Timer2 PWM și operare async.

1x 16 bit Timer1 cu PWM

Caracteristici temporizatoare 8 biti

- 2 unități de comparare independente
- 3 surse de intreruperi (TOVx, OCFxA, OCFxB)

Atmega 2560

1x 8 bit Timer0 cu PWM, 1x 8 bit Timer2 cu PWM și operare async.

4x 16 bit Timer(1,3,4,5) cu PWM

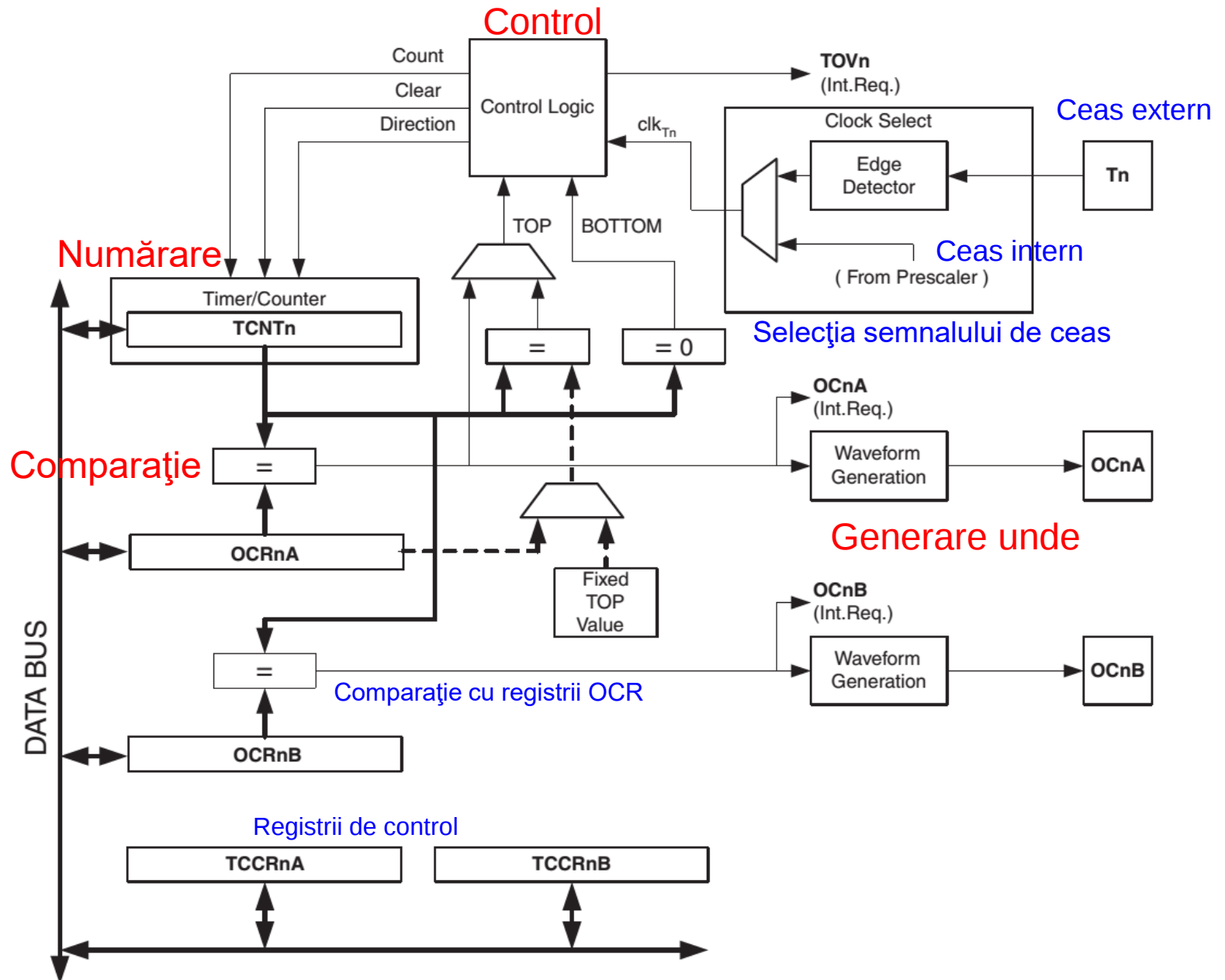
Caracteristici temporizatoare 16 biti

- 3 unitati de comparare independente
- 4 surse de intrerupere (TOVx, OCFxA, OCFxB, OCFxC, ICFx,
- 1 unitate captura intrare
- Numarator de evenimente externe

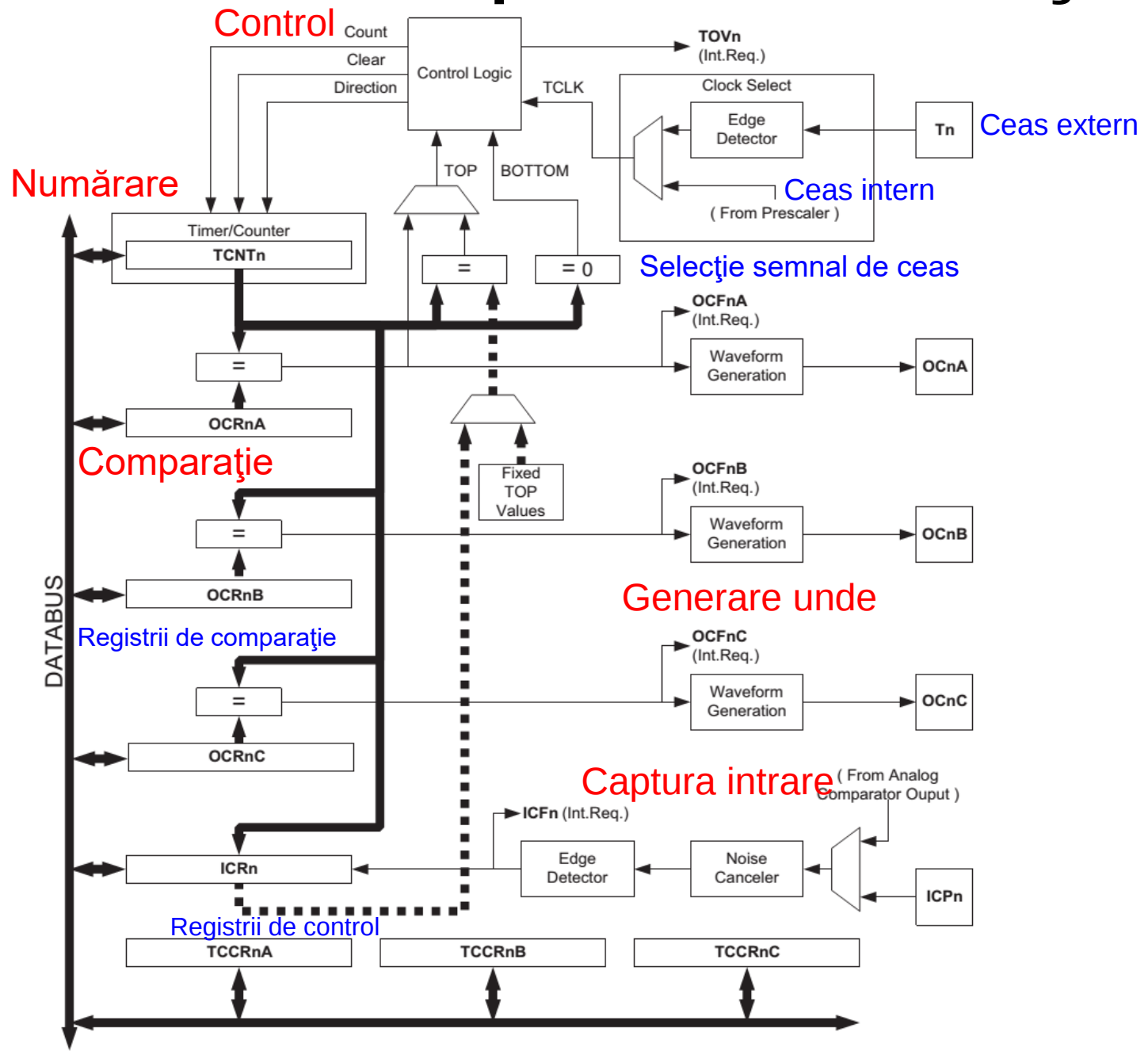
Caracteristici comune

- Registrii de comparare cu dublu buffer
- Ștergere temporizator la egalitatea comparației (Auto Reîncărcare)
- Modulare în lățimea pulsului cu fază corectată
- Perioadă PWM variabilă
- Generator de frecvențe variabile

Structura unui temporizator de 8 biți



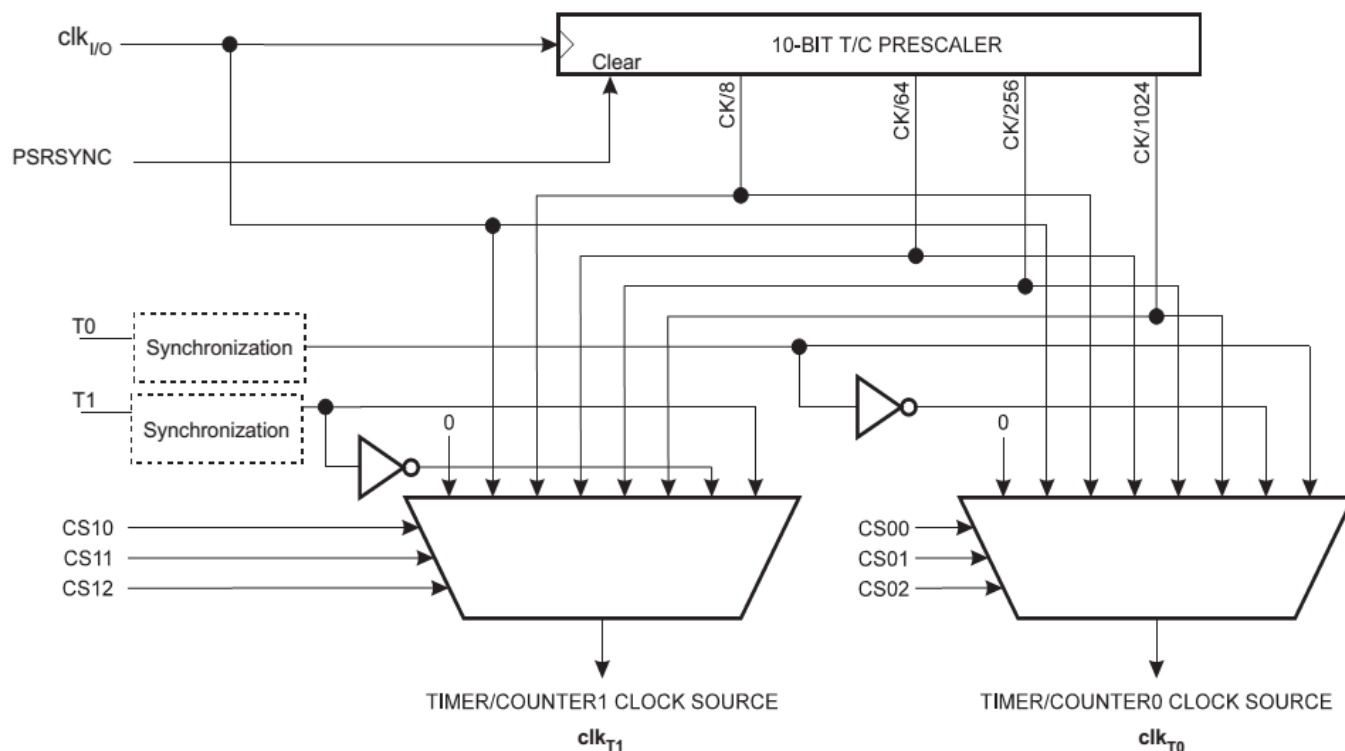
Structura unui temporizator de 16 biți



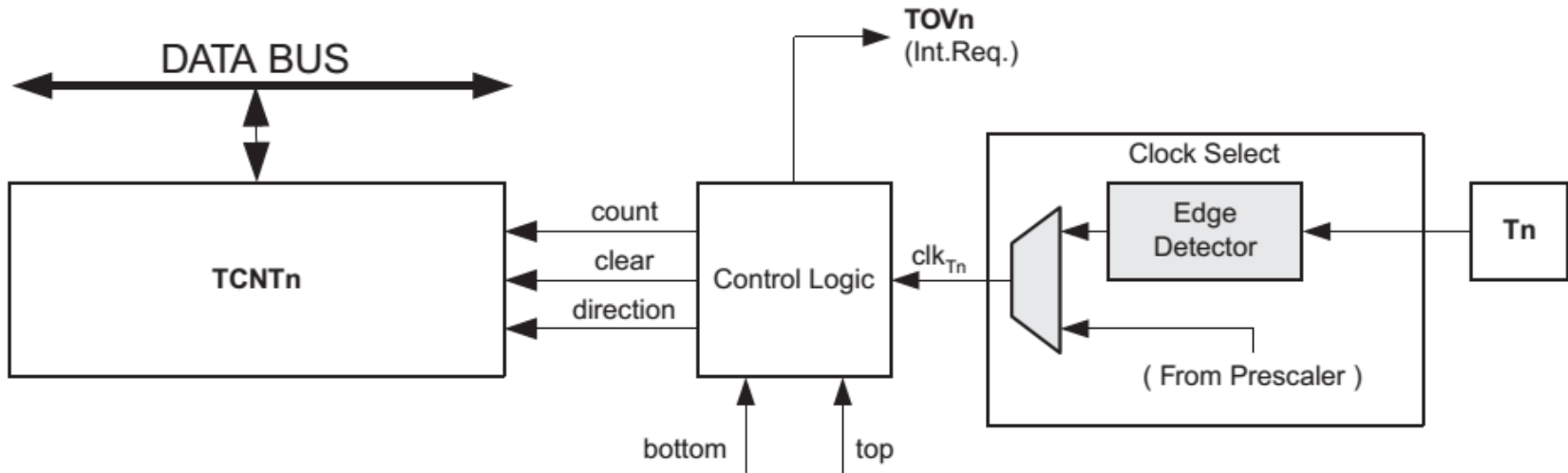
Selecția semnalului de ceas

- Biții CS02 .. CS00 controlează divizarea semnalului de ceas la intrarea numărătorului
- Se reglează viteza (frecvența) de numărare

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$\text{clk}_{I/O}/(\text{No prescaling})$
0	1	0	$\text{clk}_{I/O}/8$ (From prescaler)
0	1	1	$\text{clk}_{I/O}/64$ (From prescaler)
1	0	0	$\text{clk}_{I/O}/256$ (From prescaler)
1	0	1	$\text{clk}_{I/O}/1024$ (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.



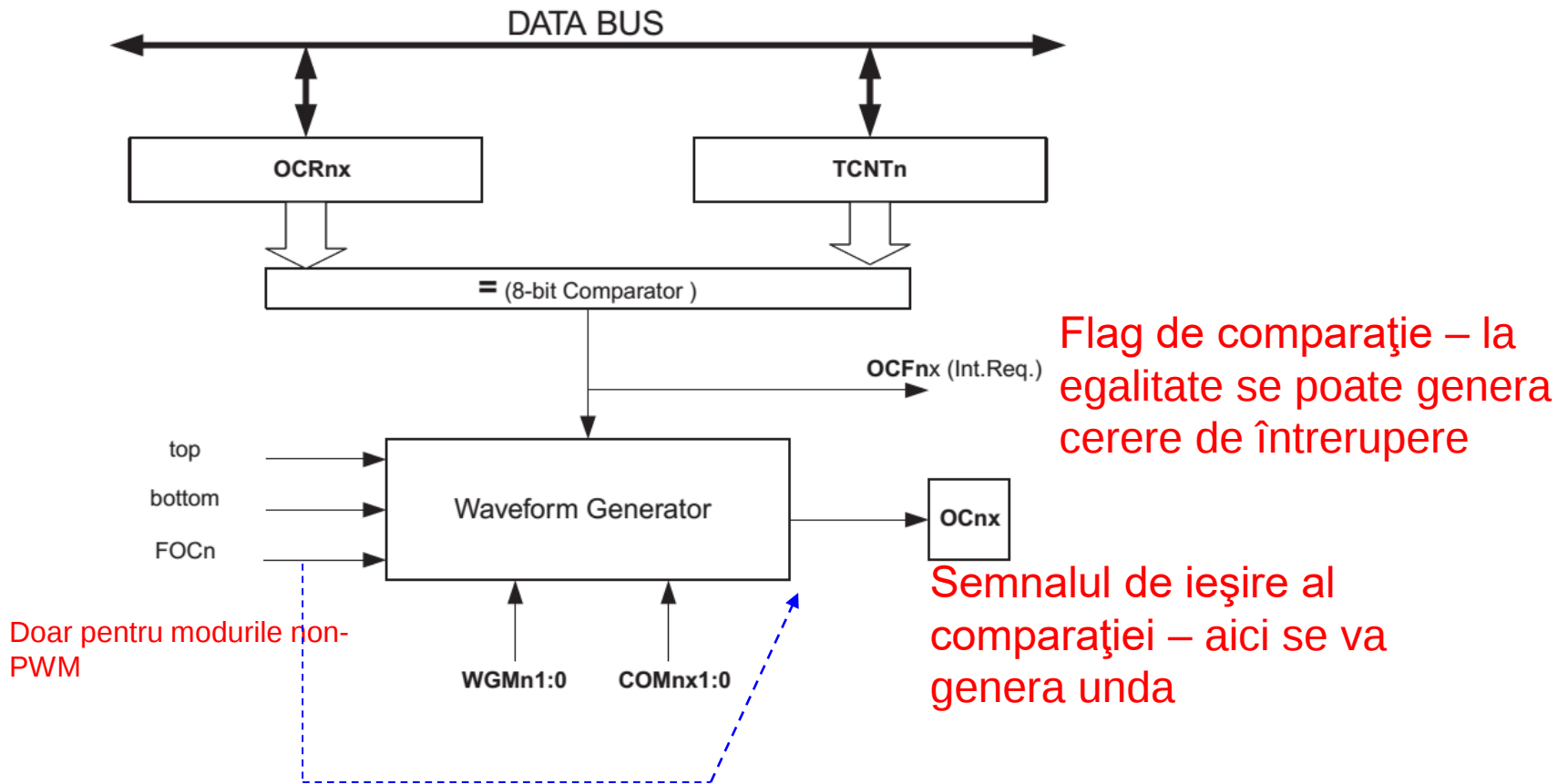
Unitatea de numărare



- **count** incrementează sau decrementează TCNT0 cu 1.
- **direction** selectează între incrementare și decrementare.
- **clear** șterge TCNT0 (pune toți biții pe zero).
- **clk_{T0}** semnalul de ceas pentru numărare.
- **top** semnalează că TCNT0 a atins valoarea maximă (0xFF).
- **bottom** semnalează că TCNT0 a atins valoarea minimă (zero).
- CPU poate citi valoarea TCNT0 (cu prioritate)
- Flag-ul Timer/Counter Overflow (TOV0) este setat conform operației selectate de biții WGM02:0 bits. TOV0 poate fi folosit pentru generarea unei întreruperi.

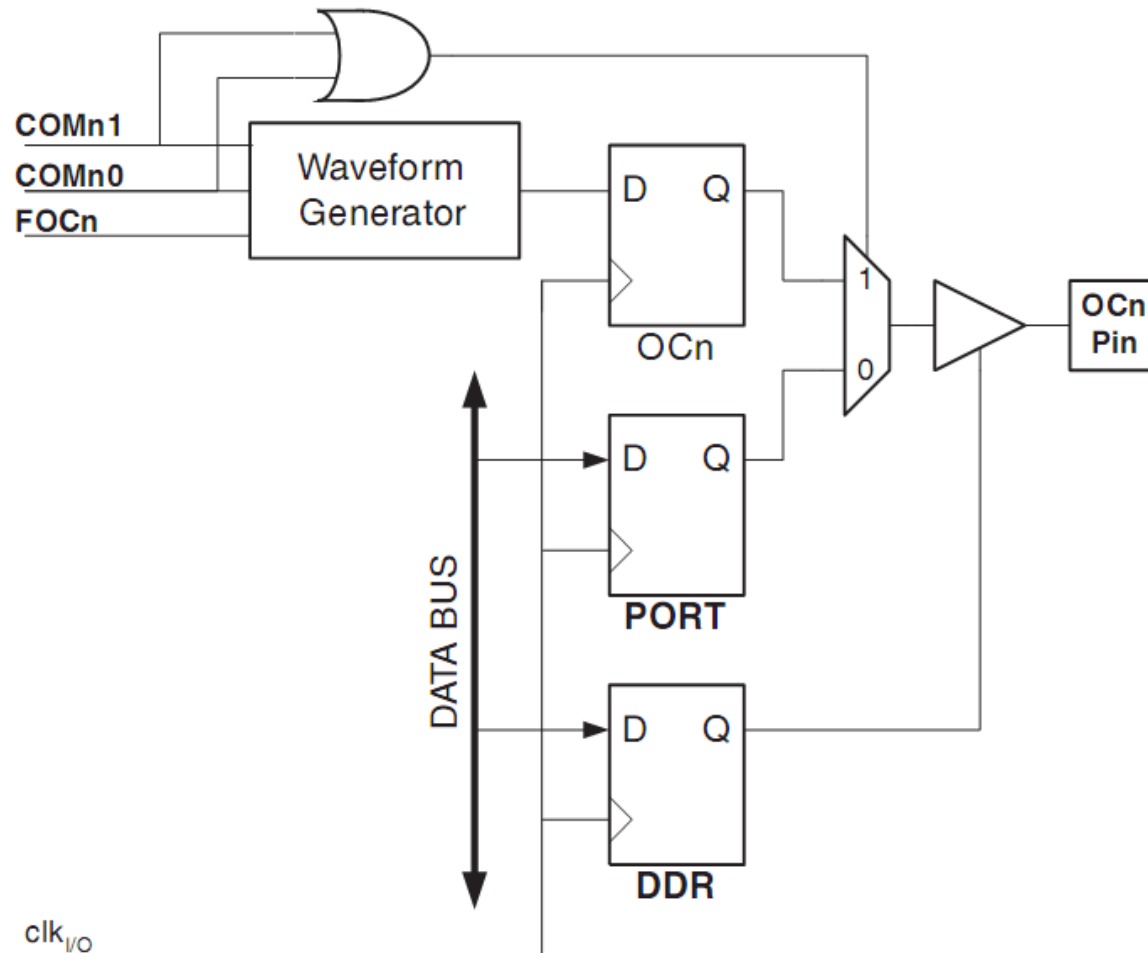
Unitatea de comparație

- Comparație între conținutul registrului de numărare (**TCNT0**) și un registru de comparare (**OCR0x**) $\Rightarrow$ folosită pentru generarea diferitelor semnale



Unitatea de comparație

- Undele generate sunt vizibile prin pinii unor porturi I/O
- Direcția acestor pini trebuie configurată prin program ca ieșire



Tipuri de undă (moduri de funcționare)

Table 14-8. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on ⁽¹⁾⁽²⁾
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

Biții WGM02:0 împreună cu biții COM1:0 definesc comportamentul temporizatorului.

Table 14-2. Compare Output Mode, non-PWM Mode

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on Compare Match
1	0	Clear OC0A on Compare Match
1	1	Set OC0A on Compare Match

Table 14-3. Compare Output Mode, Fast PWM Mode⁽¹⁾

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected. WGM02 = 1: Toggle OC0A on Compare Match.
1	0	Clear OC0A on Compare Match, set OC0A at BOTTOM, (non-inverting mode).
1	1	Set OC0A on Compare Match, clear OC0A at BOTTOM, (inverting mode).

Table 14-4. Compare Output Mode, Phase Correct PWM Mode⁽¹⁾

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected. WGM02 = 1: Toggle OC0A on Compare Match.
1	0	Clear OC0A on Compare Match when up-counting. Set OC0A on Compare Match when down-counting.
1	1	Set OC0A on Compare Match when up-counting. Clear OC0A on Compare Match when down-counting.

Tipuri de undă (moduri de funcționare)

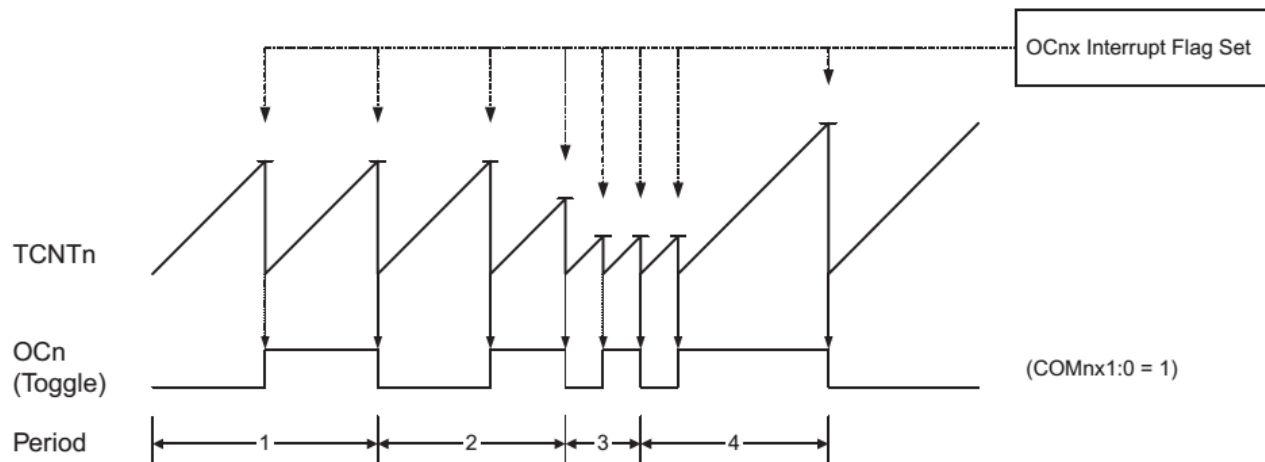
Normal

- Numărare simplă (incrementare): 0 ... 255
- Când numărătorul se saturează (0xFF), o întrerupere de overflow este generată prin setarea flag-ului TOV0, și numărătorul repornește de la 0.

Temă: calculați frecvența de generare a TOV0 pentru diferite valori ale prescaler-ului.

CTC – Clear Timer on Compare Match (generator de frecvențe variabile)

- Când valoarea numărătorului (TCNT0) ajunge la valoarea OCR0 value, numărătorul repornește de la 0
- **Frecvența undelor generate se reglează prin scrierea registrului OCR0**
 - ex: COM01:COM00 = 01 – semnalul OC0 comută la egalitate



$$f_{OCn} = \frac{f_{clk_I/O}}{2 \cdot N \cdot (1 + OCRn)}$$

N = prescale factor

(1, 8, 64, 256, 1024)

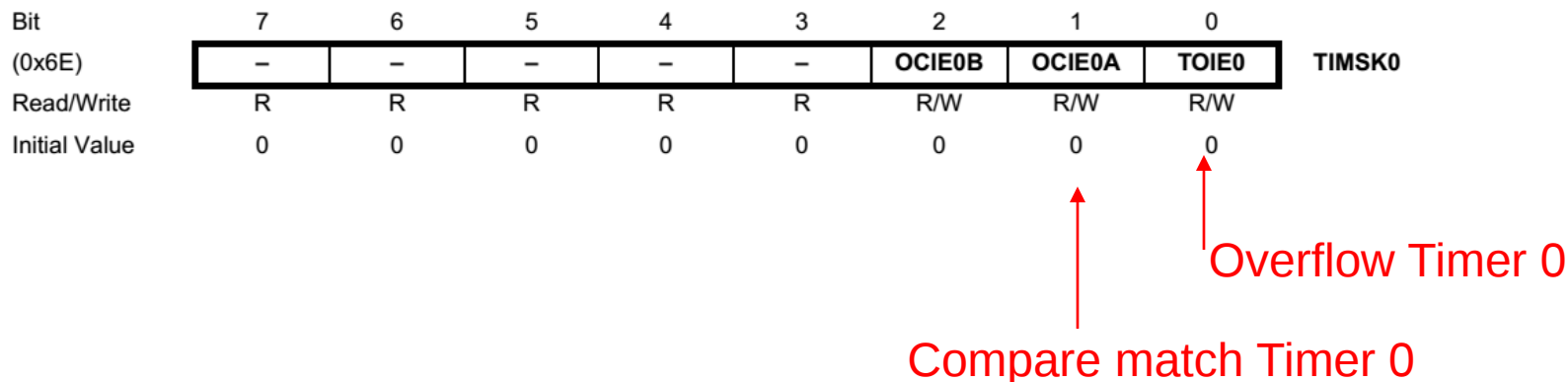
Întreruperi generate de temporizator

- Evenimente care generează întreruperi:
 - *Overflow* (Saturare)
 - *Compare match* (Atingerea valorii din registrul de comparație)
 - Eveniment extern (*capture*) – doar pentru temporizatoarele pe 16 biți

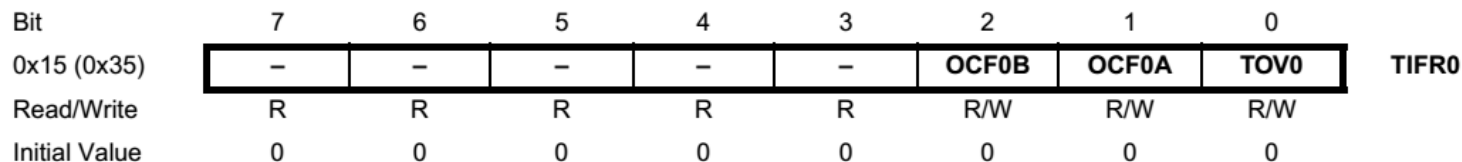
	Adresă		Descriere
14	\$001A	TIMER2 COMPA	Timer/Counter2 Compare Match A
15	\$001C	TIMER2 COMPB	Timer/Counter2 Compare Match B
16	\$001E	TIMER2 OVF	Timer/Counter2 Overflow
17	\$0020	TIMER1 CAPT	Timer/Counter1 Capture Event
18	\$0022	TIMER1 COMPA	Timer/Counter1 Compare Match A
19	\$0024	TIMER1 COMPB	Timer/Counter1 Compare Match B
20	\$0026	TIMER1 COMPC	Timer/Counter1 Compare Match C
21	\$0028	TIMER1 OVF	Timer/Counter1 Overflow
22	\$002A	TIMER0 COMPA	Timer/Counter0 Compare Match A
23	\$002C	TIMER0 COMPB	Timer/Counter0 Compare match B
24	\$002E	TIMER0 OVF	Timer/Counter0 Overflow

Întreruperi generate de temporizator

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



- Accesare stare condiții de întrerupere – registrul TIFR



- Utilizarea posibilă a întreruperilor
 - Generarea de forme de undă prin program
 - Temporizare regulată pentru diferite evenimente ex: baleiere SSD, baleiere matrice de LED-uri, etc.
 - Schimbarea parametrilor pentru unde hardware

Example

- **Exemplu 1** – generare undă de **frecvență constantă specificată**
- Definirea problemei: generați un semnal de 50 Hz
- Mod de lucru: CTC (Clear on Compare Match) – permite reglarea perioadei semnalului prin scrierea registrului OCR
- Calculul frecvenței:

$$f_{OCn} = \frac{f_{clk_I/O}}{2 \cdot N \cdot (1 + OCRn)}$$

- $f_{OCn} = 50$
- $N = 1024$ = divizarea maximă permisă de prescaler
- $F_{clk_io} = 16,000,000 = 16$ MHz, frecvența microcontrollerului
- $OCR0 = 16,000,000 / (2 \cdot 1024 \cdot 50) - 1 = 154$

Exemple

- **Exemplu 1** – generare undă de **frecvență constantă specificată**

```
.org 0x0000
```

```
jmp reset
```

```
reset:
```

```
ldi r16, 0b01000010
```

```
out TCCR0A, r16 ; configurare Timer0
```

```
ldi r16, 0b00000101
```

```
out TCCR0B, r16
```

```
ldi r16, 154 ; OCR calculat
```

```
out OCR0A, r16
```

```
ldi r16, 0xff
```

```
out DDRB, r16 ; Portul B conține OC0A - ieșire
```

```
donothing:
```

```
rjmp donothing ; Se poate monitoriza cu osciloscop pe OC0A
```

(PB7 pentru MEGA, PD6 pentru UNO) !!!

0	1	0	CTC	
---	---	---	-----	--

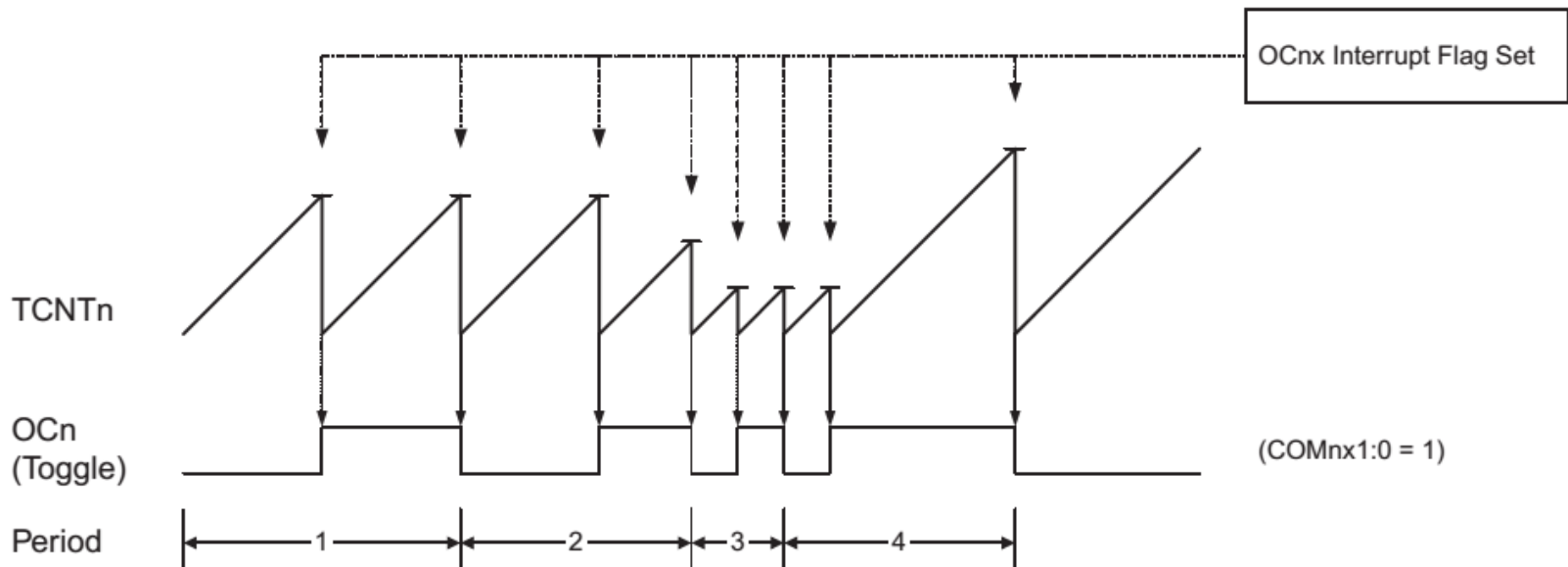
Table 14-2. Compare Output Mode, non-PWM Mode

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on Compare Match
1	0	Clear OC0A on Compare Match
1	1	Set OC0A on Compare Match

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stops)
0	0	1	$\text{clk}_{\text{I/O}}$ /(No prescaling)
0	1	0	$\text{clk}_{\text{I/O}}/8$ (From prescaler)
0	1	1	$\text{clk}_{\text{I/O}}/64$ (From prescaler)
1	0	0	$\text{clk}_{\text{I/O}}/256$ (From prescaler)
1	0	1	$\text{clk}_{\text{I/O}}/1024$ (From prescaler)
1	1	0	External clock source on T0 pin. Clock divider = 1
1	1	1	External clock source on T0 pin. Clock divider = 2

Example

- **Exemplu 2** – folosirea întreruperilor la comparare
- Definirea problemei: generați un semnal cu perioada de 1 sec / **1Hz (imposibil doar prin OCR0, cea mai mică frecvență este $f_{\text{MIN}} = 30 \text{ Hz}$ la divizare de 1024)**
- Re-utilizăm configurația din exemplul 1
- Frecvența anterioară – 50 Hz, cu Toggle la CTC $\Rightarrow$ 2 egalități în 1/50 sec. $\Rightarrow$ 1 egalitate la 1/100 sec. $\Rightarrow$ **o întrerupere Comp Match la fiecare 10 ms**
- Vom comuta un semnal la fiecare 50 întreruperi, pentru a genera un semnal '0' de 500 ms, și un semnal '1' pentru alte 500 ms.
- Semnalul rezultat va avea o perioadă de 1s



Example

- **Exemplu 2** – folosirea întreruperilor la comparare

.org 0x0000

jmp reset

.org 0x002A

; adresa (vectorul) pentru for Timer0CompA

jmp timercpm

reset:

ldi r16, low(RAMEND)

; inițializare stivă, necesară pentru întreruperi

out SPL, r16

ldi r16, high(RAMEND)

out SPH, r16

ldi r16, 0b**01**0000**10**

out TCCR0A, r16

; aceeași configurație ca pentru exemplul 1

ldi r16, 0b0000**0101**

out TCCR0B, r16

ldi r16, 154

out OCR0A, r16

ldi r16, 0xff

; Vom folosi pentru ieșire LED-uri conectate la PORTA

out DDRA, r16

Examples

- **Exemplu 2** – folosirea întreruperilor la comparare

ldi r16, 0b00000010 ; activarea întreruperii Timer0 CompA
out TIMSK0, r16

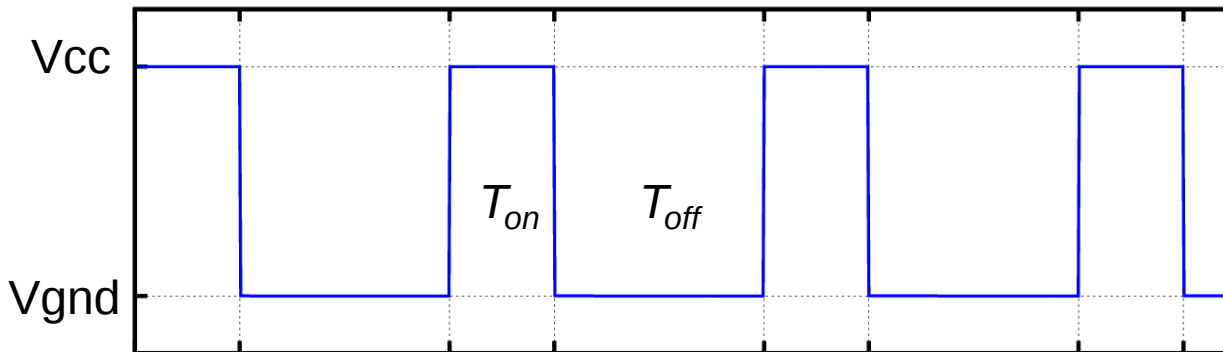
ldi r18, 0 ; numărător de evenimente
ldi r19, 0 ; starea semnalului care va fi comutat
sei ; activarea globală a întreruperilor

loop: ; programul principal va fi blocat aici
rjmp loop

timercpm: ; ISR apelată la fiecare 10 ms (Timer0 CompA Match)
 inc r18 ; se incrementează numărul de evenimente
 cpi r18, 50
 brne exit ; nu am ajuns la 50 ⇒ ieșire
; Dacă s-a ajuns la 50 de evenimente
 com r19 ; comutare r19
 out PORTA, r19 ; afișare conținut pe LED-uri
 ldi r18, 0 ; resetare număr evenimente
exit:
reti

Pulse Width Modulation (PWM)

- Unele sisteme trebuie controlate prin variația tensiunii
 - Ex: viteza unui motor, strălucirea unui LED
- Sistemele digitale pot produce doar două valori : 0 (GND) și 1 (Vcc)
- Tensiunea (medie) variabilă poate fi obținută prin reglarea factorului de umplere D (*duty cycle*)



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

- Tensiunea medie

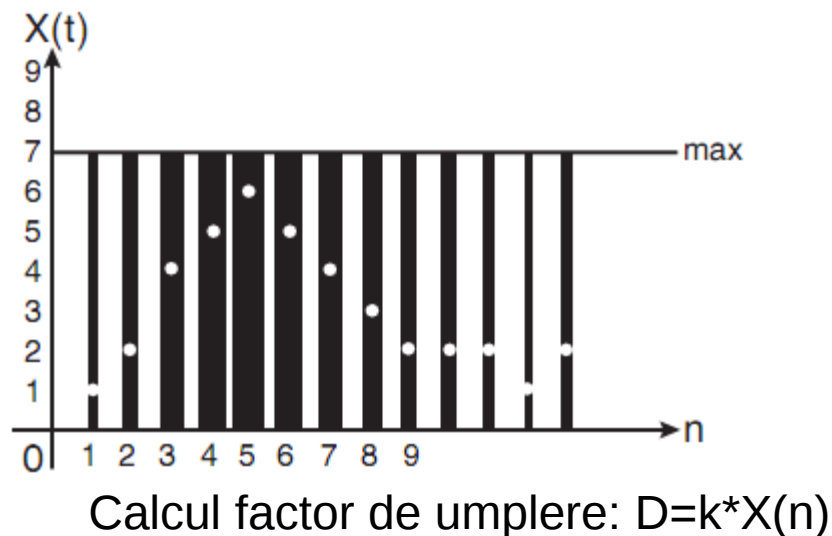
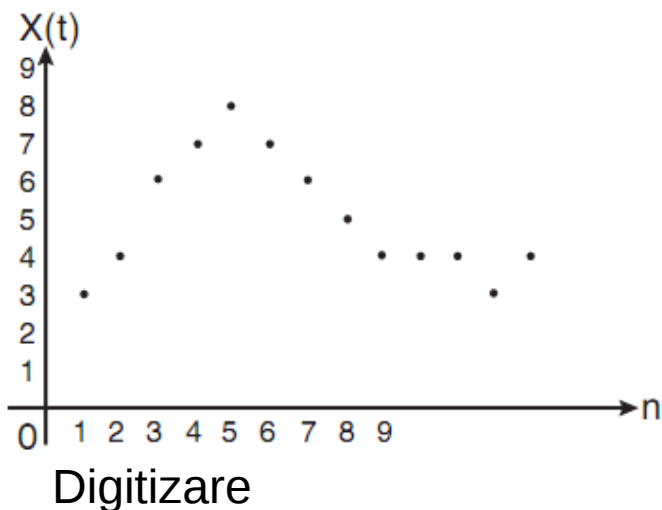
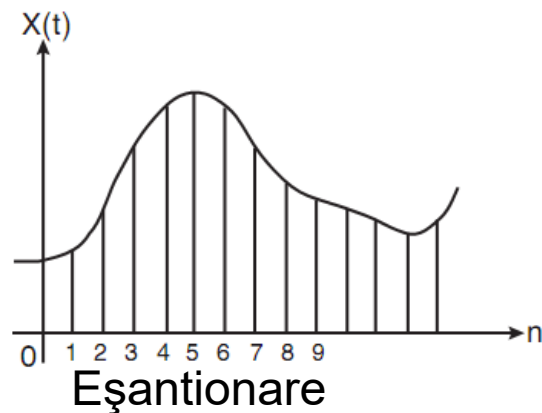
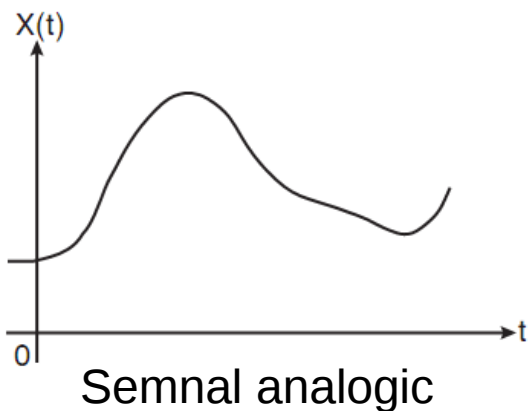
$$V_{AVG} = DV_{CC} + (1 - D)V_{GND}$$

- Dacă $V_{GND} = 0$:

$$V_{AVG} = DV_{CC}$$

Pulse Width Modulation (PWM)

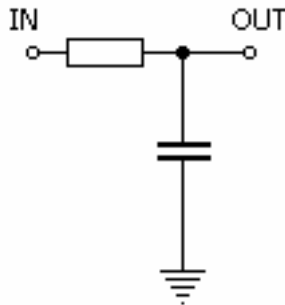
- Semnale analogice de frecvență joasă (ex. sunet) pot fi redade ca PWM
- Pași: eșantionare, digitizare, calcul D pe baza valorii digitale



Pulse Width Modulation (PWM)

Folosirea semnalului PWM

- Ca atare, dacă aplicația permite: putere variabilă LED, reglare viteză motor, etc (inerția dispozitivului produce efectul de mediere)
- Poate fi filtrat printr-un filtru Trece-Jos (low pass) pentru a reconstrui semnalul analogic:
 - Limita superioară a semnalului analogic (*cutoff frequency*) este reglată mult sub frecvența semnalului purtător
 - Filtru simplu trece jos: RC

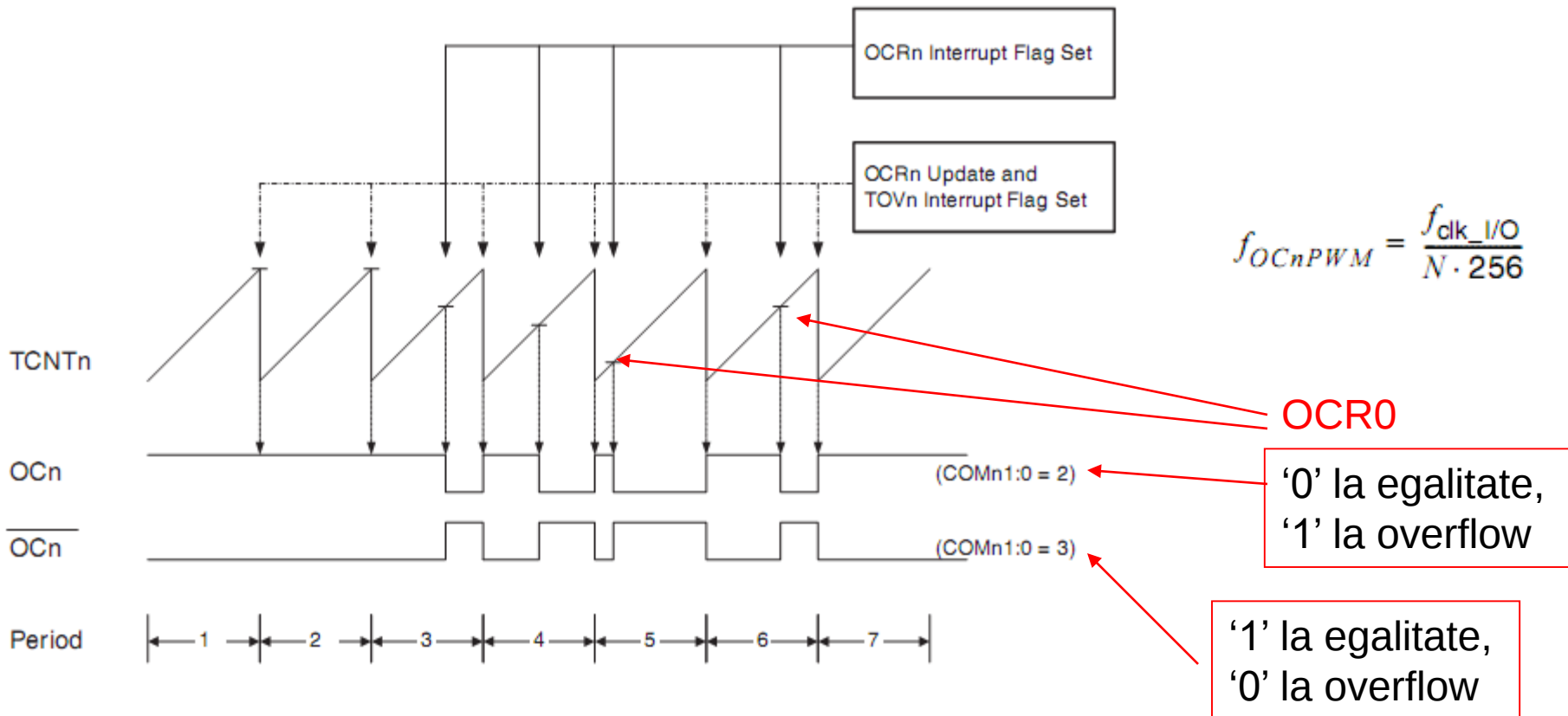


$$f_{cutoff} = \frac{1}{2\pi RC}$$

Tipuri de undă (moduri de funcționare)

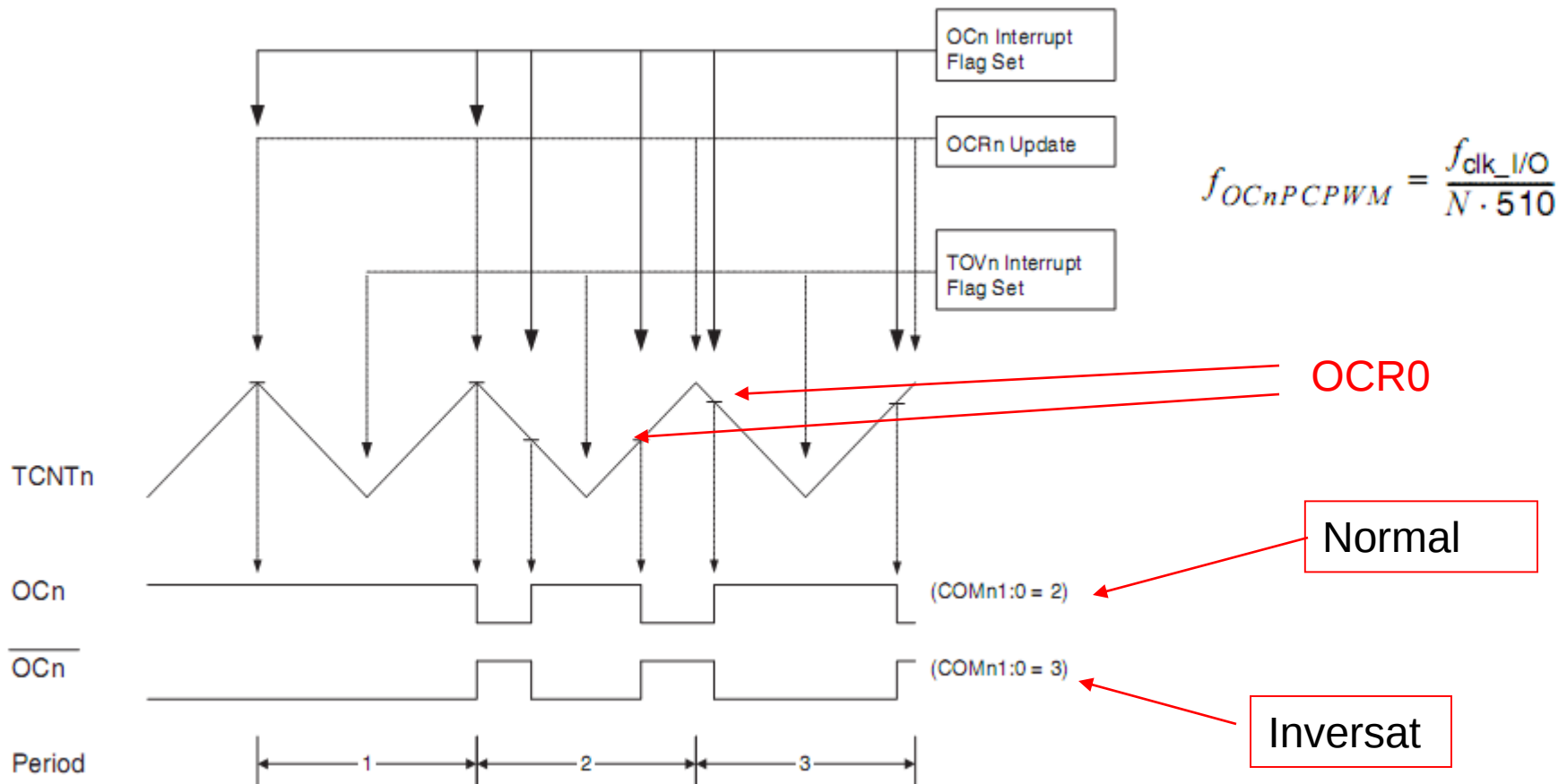
Modul Fast Pulse Width Modulation (PWM) – PWM rapid

- Generare semnale PWM (“modulare in latimea pulsului”) pe OC0x
- Factorul de umplere este setat prin scrierea valorii registrului OCR0x
- **Frecvența este fixă**, controlată prin biții Clock Select (CS) (**prescaler**)
- **Factor de umplere = $OCR0x / 255$** (T_{on} / T , $T = T_{on} + T_{off}$)



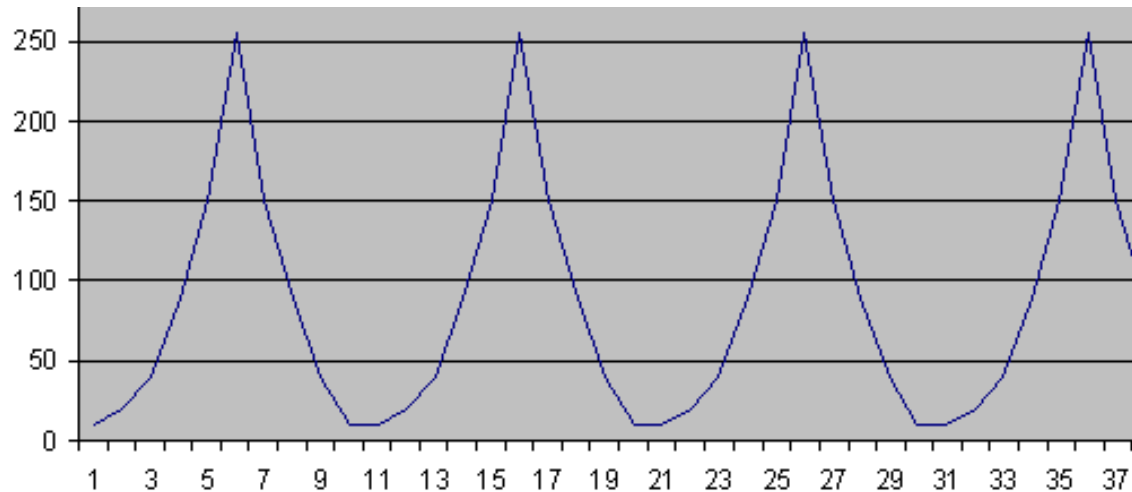
Tipuri de undă (moduri de funcționare)

- **Pulse Width Modulation (PWM) cu fază corectă**
 - Generare semnale PWM cu corecție de fază
 - Pulsul este simetric față de mijlocul perioadei (când TCNT0 = BOTTOM)
 - Factorul de umplere setat prin scrierea registrului OCR0x
 - Numărare crescătoare/descrescătoare, schimbarea ieșirii la egalități succesive
 - Factor de umplere = $OCR0 / 255$



Example

- **Exemplu 3** – utilizare PWM
- Definirea problemei: generați un semnal analogic periodic
- Forma semnalului va fi definită prin valori discrete într-o tabelă LUT
- Utilizare mod PWM phase correct
- OCR0A va defini lățimea pulsului
- Valoarea OCR0A se va schimba la saturarea numărătorului
- Valori pentru o perioadă: 10, 20, 40, 90, 150, 255, 150, 90, 40, 20, 10



Exemple

• Exemplu 3 – utilizare PWM

```
.org 0x0000
jmp reset
.org 0x002E           ; Timer 0 OVF ISR
jmp timerovf
```

reset:

```
ldi r16, low(RAMEND)
out SPL, r16
ldi r16, high(RAMEND)
out SPH, r16
```

```
ldi r16, 0b10000001
out TCCR0A, r16
ldi r16, 0b00000001
out TCCR0B, r16
```

```
ldi r16, 0xff
out DDRB, r16 ; activare ieşire OC0A
```

```
ldi r16, 0b00000001 ; activare intrerupere Timer0 Ovf
out TIMSK0, r16
```

1	0	0	1	PWM, Phase Correct
---	---	---	---	--------------------

Table 14-4. Compare Output Mode, Phase Correct PWM Mode⁽¹⁾

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected. WGM02 = 1: Toggle OC0A on Compare Match.
1	0	Clear OC0A on Compare Match when up-counting. Set OC0A on Compare Match when down-counting.
1	1	Set OC0A on Compare Match when up-counting. Clear OC0A on Compare Match when down-counting.

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stops)
0	0	1	clk _{I/O} /1 (No prescaling)
0	1	0	clk _{I/O} /8 (From prescaler)
0	1	1	clk _{I/O} /64 (From prescaler)
1	0	0	clk _{I/O} /256 (From prescaler)
1	0	1	clk _{I/O} /1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock
1	1	1	External clock source on T0 pin. Clock

Example

- **Exemplu 3** – utilizare PWM

ldi r18, 0 ; index in LUT

sei ; activare intreruperi globale

loop: rjmp loop ; programul se blochează aici

timerovf: ; Timer 0 Ovf ISR – apelată la sfârșitul perioadei de numărare

ldi r17, 0 ; calcul adresă în LUT

ldi zh, high(2*translut)

ldi zl, low(2*translut)

add zl, r18

adc zh, r17

lpm r17, Z

out OCR0A, r17 ; valoarea citită din LUT se scrie în OCR

inc r18

cpi r18, 10 ; adresa maximă din LUT

brne exit ; dacă $r18 < 10 \Rightarrow$ iese din ISR

ldi r18, 0 ; dacă $(r18 = 10) \Rightarrow$ revenim la începutul LUT

exit:

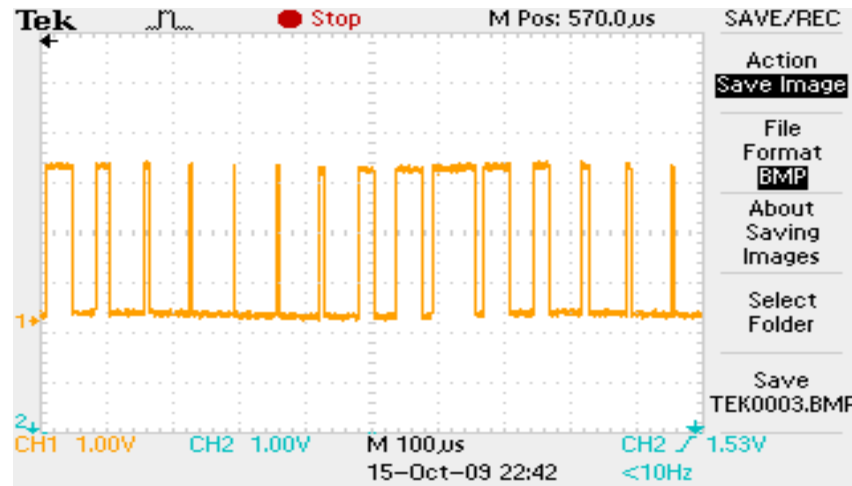
reti

translut: ; LUT – definirea funcției

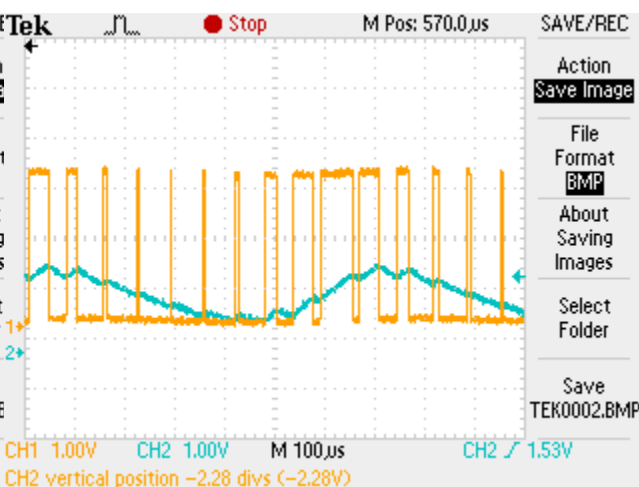
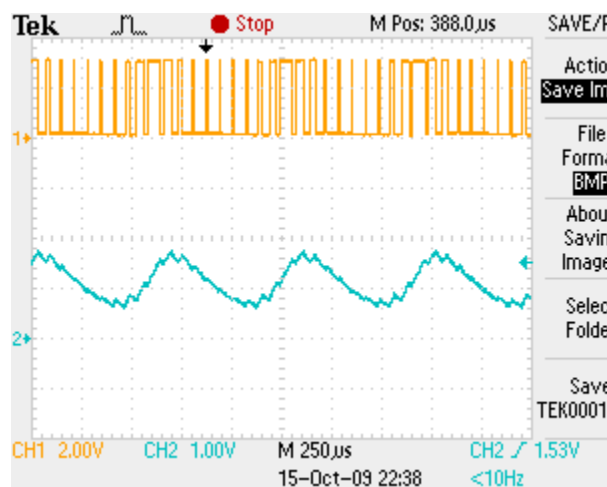
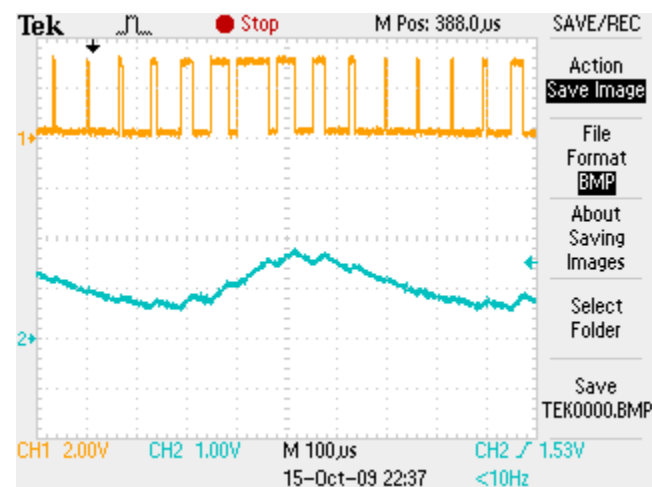
.db 10, 20, 40, 90, 150, 255, 150, 90, 40, 20, 10

Example

- Exemplu 3 – utilizare PWM



- Prin adăugarea unui filtru RC, $R = 1K$, $C = 0.22 \mu F \Rightarrow$ undă analogică:



Example

- Exemplu 4
- Definirea problemei: măsurați intervalul dintre două evenimente externe
- Configurație – Normal, cu frecvența de numărare minimă
- Evenimentele externe vor genera o întrerupere externă (front crescător)

.org 0x0000

rjmp reset

.org 0x0002

rjmp int0ISR

reset:

ldi r16, low(RAMEND)

out SPL, r16

ldi r16, high(RAMEND)

out SPH, r16

ldi r16, 0b00000000

out TCCR0A, r16

ldi r16, 0b00000101

out TCCR0A, r16

ldi r16, 0xff

out DDRA, r16 ; activare ieșire (port A – LEDs)

; ISR pentru intrerupere externă 0

WGM02	WGM01	WGM00	Timer/Counter Mode of Operation
0	0	0	Normal

Table 14-2. Compare Output Mode, non-PWM Mode

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on Compare Match
1	0	Clear OC0A on Compare Match
1	1	Set OC0A on Compare Match

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$\text{clk}_{I/O}$ (No prescaling)
0	1	0	$\text{clk}_{I/O}/8$ (From prescaler)
0	1	1	$\text{clk}_{I/O}/64$ (From prescaler)
1	0	0	$\text{clk}_{I/O}/256$ (From prescaler)
1	0	1	$\text{clk}_{I/O}/1024$ (From prescaler)

Example

- **Exemplu 4** – măsurarea timpului

ldi r16, 0b00000011 ; configurare INT0 – front crescător

sts EICRA, r16

ldi r16, 0b00000001 ; activare INT0

out EIMSK, r16

ldi r17, 0 ; ultima valoare a numărătorului

sei

loop: **rjmp** loop

int0ISR:

in r16, **TCNT0** ; citirea valorii registrului de numărare

mov r18, r16

sub r16, r17 ; scădem valoarea anterioară

mov r17, r18 ; starea nouă devine cea veche

out PORTA, r16 ; afișare diferențe

reti

Exemple

Exemplu 5 - afișați un număr de două cifre folosind PMOD-SSD conectat la portul A

```
.org 0x0000
    jmp main
.org 0x002A                ; adresa pentru for Timer0 Comp ISR
    jmp timercpm
.def temp = r20
.def index0 = r21 ; indexul cifrei unităților din memoria program
.def index1 = r22 ; indexul cifrei zecilor din memoria program
.def digit0 = r23 ; codul SSD - cifra unitatilor
.def digit1 = r24 ; codul SSD - cifra zecilor
.set tab_size = 16 ; numar total de cifre in LUT

// macro folosit pentru citirea valorilor din tabela @0 (din memoria
program), de la index @1, iesirea in @2
.macro rdb                ; adresa inceput LUT, deplasament, registru iesire
    ldi zh, high(2*@0)    //biti superiori ai 2*data_address
    ldi zl, low(2*@0)     //biti inferiori ai 2*data_address
    add zl, @1            //se aduna @1 (offset) la zl
    ldi temp,0            //temp - intotdeauna zero
    adc zh, temp          //aduna cu carry 0 la zh
    lpm @2, Z             //se citește locația de la adresa Z
.endmacro
```

Exemple

main:

```
ldi r16, low(RAMEND)
out SPL, r16
ldi r16, high(RAMEND)
out SPH, r16

//SSD conectat la PORTA
ldi temp, 0xFF
out DDRA, temp

// Configuratie timer 0 : mod CTC
ldi r16, 0b01000010
out TCCR0A, r16 ; configuratia din exemplul 1
ldi r16, 0b00000101
out TCCR0B, r16
ldi r16, 154 ; 100 de comutari pe secunda - calcul ex 1 !
out OCR0, r16

ldi r16, 0b00000010 // activare intrerupere Timer0 Comp
out TIMSK, r16

ldi index0, 4
rdb sevstable2, index0, digit0 //citeste codul SSD pentru cifra 0
ori digit0, 0x80
ldi index1, 6
rdb sevstable2, index1, digit1 //citeste codul SSD pentru cifra 1

ldi r19, 0 ; indica cifra activa
sei ; activare intreruperi globale
```

Example

```
loop:
    rjmp loop

timercpm:        // Timer comp 0 ISR
    cpi r19, 0
    brne display_digit1
display_digit0:
    out PORTA, digit0
    rjmp toggle
display_digit1:
    out PORTA, digit1
toggle:
    com r19        ; comutare r19
    reti

sevstable2:      ; code table in program memory for LED anodes
    .db 0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,
        0x71
```

Temă:

- Calculați perioada de reîmprospătare pentru afișajul SSD
- Cât timp este o cifră aprinsă sau stinsă ?
- Modificați exemplul anterior pentru o frecvență de reîmprospătare de 500 Hz

Studiu suplimentar

Muhammad Ali **Mazidi**, Sarmad Naimi, Sepehr Naimi,

The AVR Microcontroller and Embedded Systems Using Assembly And C,
1st Edition, Prentice Hall, 2009.

http://www.microdigitaled.com/AVR/AVR_books.htm

Cod sursa pentru exemple din carte:

http://www.microdigitaled.com/AVR/Code/AVR_codes.htm

Disponibile in retea: [\\D110-13](#) - Sala 107