

# Generarea de semnale PWM

## a) Scopul lucrării

Familiarizarea cu cele doua module generatoare de semnale PWM din cadrul dsPIC-ului si generarea de semnale PWM independente sau complementare.

## b) Introducere

PWM (**P**ulse **W**idth **M**odulation) este o tehnică folosită pentru a varia în mod controlat tensiunea dată unui dispozitiv electronic.

Microcontrolerele dsPIC au două module generatoare de PWM: PWM<sub>1</sub> cu 6 ieșiri și PWM<sub>2</sub> cu 2 ieșiri.

### 28-Pin SDIP, SOIC

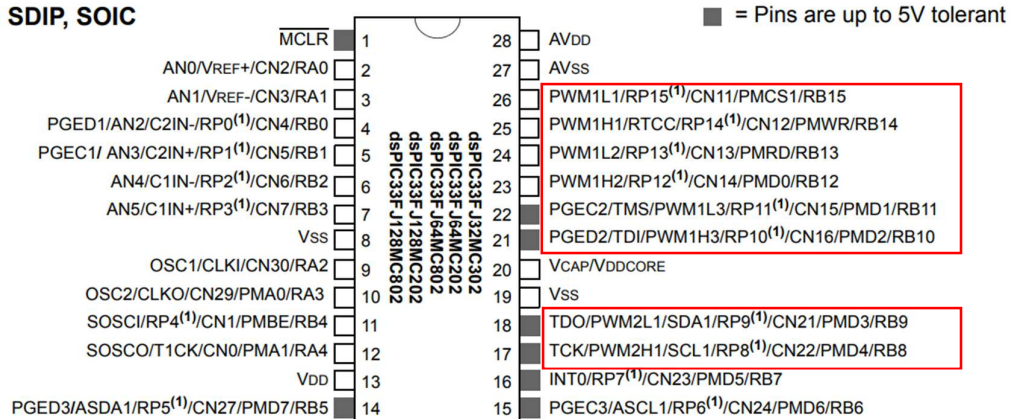


Figura 1. Capsula microcontrolerului dsPIC

Inițiala L sau H din PWM<sub>n</sub>L<sub>m</sub> sau PWM<sub>n</sub>H<sub>m</sub> ne sugerează tipul pinului – Low sau High, iar cifra *m* semnifică numărul perechii de pini Low și High. Microcontrolerul dsPIC oferă posibilitatea de a genera semnale PWM complementare pe aceste perechi de pini.

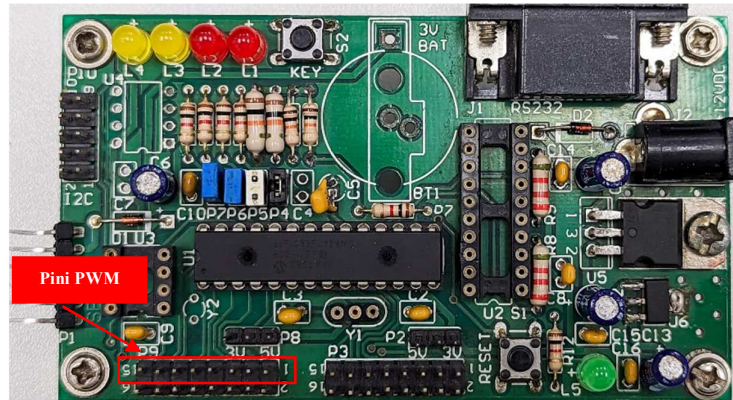


Figura 2. Pini PWM

## Servomotorul

Semnalul de comandă va fi aplicat pe pinul de *Control* și trebuie să aibă următoarele caracteristici:

- **Perioada semnalului PWM** (notată cu  $T$ ) să fie între 10 și 20ms (standard 20ms)
- **Lățimea pulsului** (notată cu  $L$ ) să fie între 1 și 2ms. Pentru  $L = 1\text{ms}$  cama servomotorului va lua o poziție extremă minimă ( $0^\circ$ ) și poziția extremă maximă ( $180^\circ$ ) pentru  $L = 2\text{ms}$ .

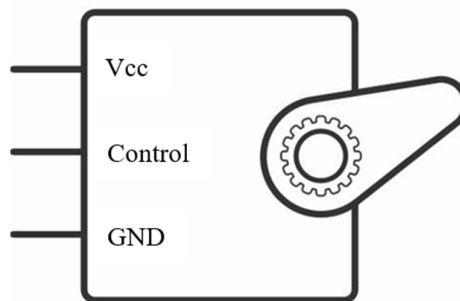


Figura 1. Pini servo motor

Servomotoarele din laborator pot fi alimentate la o tensiune de  $5V$ , aplicată pe pinul de  $V_{cc}$ .

### c) Calculul Factorului de Umplere

Factorul de umplere este fracțiunea dintr-o perioadă în care un semnal sau un sistem este activ. Ciclul de funcționare este de obicei exprimat ca procent sau raport. O perioadă(T) este timpul necesar unui semnal pentru a finaliza un ciclu de pornire și oprire.

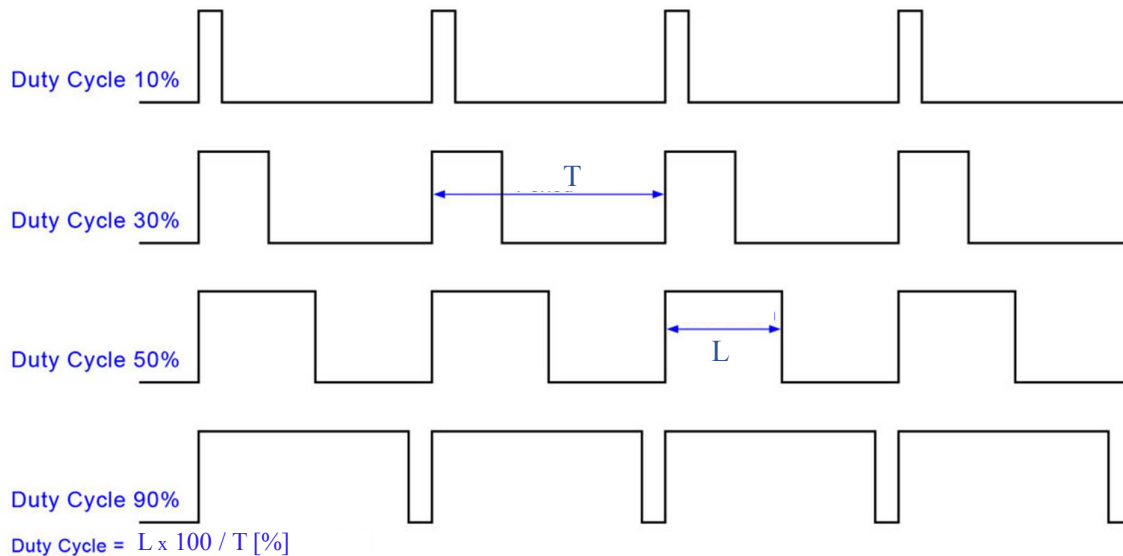


Figura 2. Calculul factorului de umplere

La baza modului generator de PWM se află un numărător, care este comandat de ceasul de magistrală. Ceasul de magistrală are următoarele caracteristici:

- Perioadă – Tcy
- Frecvență – Fcy

În aplicațiile de laborator vom folosi următoarea funcție pentru a seta Tcy = 25ns și Fcy = 40MHz:

```
void initPLL(void)
```

Numărătorul menționat mai sus numără perioadele ceasului primit de la intrare. În permanență valoarea de pe numărător este comparată prin intermediul unui comparator cu valorile din doi regiștri. Un registru ajută pe partea de obținere a lățimii pulsului, celălalt pe partea de obținere a perioadei semnalului PWM generat.

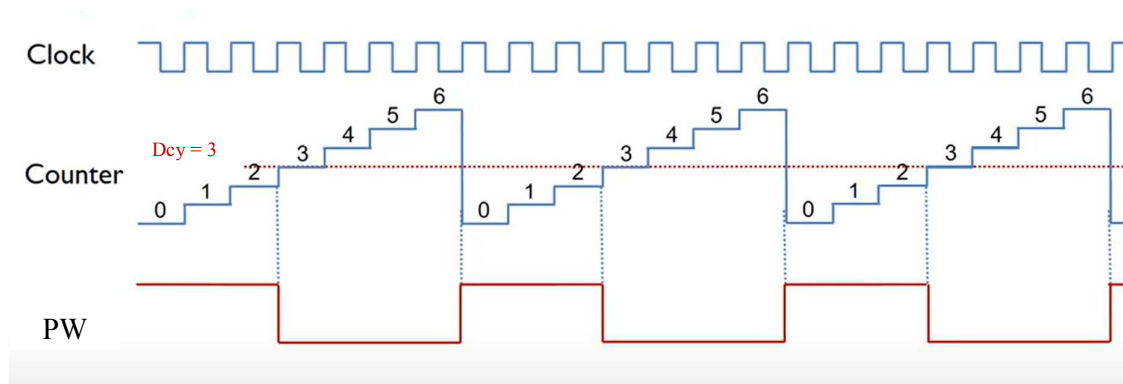


Figura 3. PWM Mode 1. Low-True

**Dcy** este valoare de comparație cu numărătorul și în cazul de mai sus este egală cu 3. Cu ajutorul acesteia putem calcula **lățimea pulsului  $d$**  semnalului PWM, prin formula:

$$Dcy = \frac{d}{T_{cy}} \quad (1)$$

**PER** este valoarea de reset a numărătorului și în cazul de mai sus este egală cu 6. Cu ajutorul acesteia putem calcula **perioada  $T$**  semnalului PWM, prin formula:

$$PER = \frac{T}{T_{cy}} \quad (2)$$

De asemenea putem calcula **factorul de umplere** în modul următor:

$$\text{Duty Cycle} = \frac{Dcy}{PER+1} \quad (3)$$

## Regiștrii de control PWM

1. **PxTCON: PWM TIME BASE CONTROL REGISTER** – este folosit pentru configurarea modului de numărare, a prescalerului și a postscalerului, cât și pentru a porni numărarea;

R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
PTEN	—	PTSIDL	—	—	—	—	—
bit 15		bit 8					

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTOPS<3:0>				PTCKPS<1:0>		PTMOD<1:0>	
bit 7		bit 0					

<b>Legend:</b>							
R = Readable bit		W = Writable bit		U = Unimplemented bit, read as '0'			
-n = Value at POR		'1' = Bit is set		'0' = Bit is cleared		x = Bit is unknown	

Important aici este bitul **PTEN** – PWM Time Base Timer Enable bit

La valoarea 1 - PWM time base este on

La valoarea 0 – PWM time base este off

2. **PxTMR: PWM TIMER COUNT VALUE REGISTER** – din acest registru pot fi citite valoarea curenta a timerului si sensul de numărare (crescător/descrescător);

R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTDIR	PTMR<14:8>						
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTMR<7:0>							
bit 7							bit 0

<b>Legend:</b>							
R = Readable bit	W = Writable bit		U = Unimplemented bit, read as '0'				
-n = Value at POR	'1' = Bit is set		'0' = Bit is cleared		x = Bit is unknown		

Bitul **PTDIR** - *PWM Time Base Count Direction Status bit (read-only)*

1 = numărătorul PWM numără descrescător

0 = numărătorul PWM numără crescător

3. **PxTPER: PWM TIME BASE PERIOD REGISTER** – in acest registru se scrie perioada de timp a modularii, ceea ce da frecventa PWM-ului;

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	PTPER<14:8>						
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTPER<7:0>							
bit 7							bit 0

<b>Legend:</b>							
R = Readable bit	W = Writable bit		U = Unimplemented bit, read as '0'				
-n = Value at POR	'1' = Bit is set		'0' = Bit is cleared		x = Bit is unknown		

Observăm că **PxTPER** este un registru pe 15 biți, ceea ce semnifică că valoarea maximă a constantei de perioadă poate fi **32767**.

4. **PWMxCON1: PWM CONTROL REGISTER 1** – se selectează modul independent sau complementar pentru fiecare pereche de pini de I/O

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	PMOD3	PMOD2	PMOD1
bit 15					bit 8		

U-0	R/W-1	R/W-1	R/W-1	U-0	R/W-1	R/W-1	R/W-1
—	PEN3H <sup>(1)</sup>	PEN2H <sup>(1)</sup>	PEN1H <sup>(1)</sup>	—	PEN3L <sup>(1)</sup>	PEN2L <sup>(1)</sup>	PEN1L <sup>(1)</sup>
bit 7					bit 0		

<b>Legend:</b>							
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'					
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown				

**bit 10-8** *PMOD4:PMOD1 - PWM I/O Pair Mode bits*

1 = perechea de pini PWM I/O este în modul Independent PWM Output

0 = perechea de pini PWM I/O este în modul Complementar PWM Output

**bit 6-4** *PEN3H:PEN1H - PWMxH I/O Enable bits*

1 = Pinul PWMxH este activat pentru ieșirea PWM

0 = Pinul PWMxH dezactivat, pinul I/O devine I/O de uz general

**bit 2-0** *PEN3L:PEN1L - PWMxL I/O Enable bits*

1 = PWMxL este activat pentru ieșirea PWM

0 = PWMxL dezactivat, pinul I/O devine I/O de uz general

5. **PxDCy: PWM DUTY CYCLE REGISTER 1 și 2** – în acești regiștri se introduc valorile pe 16 biți ale factorilor de umplere pentru perechile de ieșiri 1 sau 2;

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PDC2<15:8>							
bit 15					bit 8		

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PDC2<7:0>							
bit 7					bit 0		

<b>Legend:</b>							
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'					
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown				

## Exemplu de calcul pentru valorile regiștrilor PxTPER și PxDCy

Se dă perioada T și lățimea pulsului d:

$$T = 150 \mu s$$

$$d = 25 \mu s$$

Prin intermediul funcției `initPLL()` se setează Tcy la 25 ns.

Aplicăm formula (2) pentru calculul valorii PER pe care o scriem în registrul P1TPER:

$$P1TPER = \frac{T}{T_{cy}} = \frac{150 \mu s}{25 ns} = 6 * 10^3 = 6000.$$

Observăm că valoarea obținută este mai mică de valoarea maximă a constantei de perioadă – **32767**.

În continuare, aplicăm formula (1) pentru calculul valorii Dcy, pe care o vom scrie în registrul P1DCy:

$$P1DCy = \frac{d}{T_{cy}} * 2 = \frac{25 \mu s}{25 ns} * 2 = 2000$$

Înmulțirea cu 2 se datorează faptului că în DCy se dă în semiperioade, nu perioade.

## d) Programul exemplu

```
#include "p33FJ128MC802.h"

// Select Internal FRC at POR
_FOSCSEL(FNOSC_FRC);
// Enable Clock Switching and Configure
_FOSC(FCKSM_CSECMD & OSCIOFNC_OFF); // FRC + PLL
//_FOSC(FCKSM_CSECMD & OSCIOFNC_OFF & POSCMD_XT); // XT + PLL
_FWDT(FWDTEN_OFF); // Watchdog Timer Enabled/disabled by user software

void initPLL(void)
{
    // Configure PLL prescaler, PLL postscaler, PLL divisor
    PLLFBD = 41; // M = 43 FRC
    //PLLFBD = 30; // M = 32 XT
    CLKDIVbits.PLLPOST=0; // N1 = 2
    CLKDIVbits.PLLPRE=0; // N2 = 2

    // Initiate Clock Switch to Internal FRC with PLL (NOSC = 0b001)
    __builtin_write_OSCCONH(0x01); // FRC
    //__builtin_write_OSCCONH(0x03); // XT
    __builtin_write_OSCCONL(0x01);

    // Wait for Clock switch to occur
    while (OSCCONbits.COSC != 0b001); // FRC
    //while (OSCCONbits.COSC != 0b011); // XT

    // Wait for PLL to lock
    while(OSCCONbits.LOCK!=1) {};
}

void init_PWM1(void);
void init_PWM2(void);

int main()
{
    initPLL();
    init_PWM1();
    init_PWM2();
    while(1){} // Infinite Loop
}

void init_PWM1()
{
    P1TCONbits.PTOPS = 0; // Timer base output scale
    P1TCONbits.PTMOD = 0; // Free running

    P1TMRbits.PTDIR = 0; // Numara in sus pana cand timerul = perioada
    P1TMRbits.PTMR = 0; // Baza de timp

    P1DC1 = 0x2710;
    P1DC2 = 0x4E20;
    P1DC3 = 0x7530;
    P1TPER = 0x4E20;

    PWM1CON1bits.PMOD1 = 1; // Canalele PWM1H si PWM1L sunt independente
    PWM1CON1bits.PMOD2 = 1; // Canalele PWM2H si PWM2L sunt independente
    PWM1CON1bits.PMOD3 = 0; // Canalele PWM3H si PWM3L sunt complementare

    PWM1CON1bits.PEN1H = 1; // Pinul PWM1H setat pe iesire PWM
    PWM1CON1bits.PEN1L = 0; // Pinul PWM1L setat pe I/O general purpose

    PWM1CON1bits.PEN2H = 1; // Pinul PWM1H setat pe iesire PWM
    PWM1CON1bits.PEN2L = 0; // Pinul PWM1L setat pe I/O general purpose
```



[illegible]