PRÀCTICA 6

Treballant amb comptadors i interrupcions. Exemple: el servomotor.

En aquesta pràctica s'introdueixen els comptadors i el concepte d'interrupció. Els comptadors són perifèrics que permeten crear senyals de sincronisme precises. Les interrupcions són excepcions al microcontrolador que permeten executar rutines de servei en moments determinats.

Per a comprendre millor el seu ús s'aplicaran els coneixements adquirits al control d'un servomotor.

Sobretot convé fixar-se en:

- l'ús de comptadors com a eina de sincronisme precisa basada en el rellotge de sistema

- l'estructura del programa quan es treballa amb interrupcions

· l'origen (posició 0x0000) el configura la rutina d'inici de programa

· segueixen les adreces de les rutines d'interrupció que estan fixades

. el programa principal

· les rutines de maneig d'interrupció

- el control que s'ha de realitzar sobre els registres per treballar amb comptadors i interrupcions

Resum del material que cal per a realitzar aquesta pràctica.

- *AVRstudio i programador PonyProg*
- *Microcontrolador ATMEGA8 i placa de desenvolupament*
- *Servomotor*
- *Components discrets*

NOTA:

- *Recordeu que és indispensable que...*
- *- Llegiu els enunciats previ a l'inicia de la sessió de pràctiques*
- *- ENVIEU PREVI A L'INICIA DE LA PRÀCTICA L'INFORME PRELIMINAR AL PROFESSOR DE PRÀCTIQUES*
- *- Agafeu les dades experimentals en la pràctica i prepareu l'informe de la pràctica pel portfoli just després de realitzar la pràctica*

Sobre comptadors...

RECORDATORI (ràpid) DEL FUNCIONAMENT DELS COMPTADORS

(Cal llegir el datasheet per aprofundir en el funcionament dels comptadors)

Els microcontrolador ATmega8/128 disposen de diferents comptadors/timers de 8 bits i de 16 bits, tots ells amb prescaler i diferents modes de funcionament.

Funcionament

Els modes de funcionament que permeten els comptadors, en termes generals, són:

- Mode *normal*. El comptador compta amunt des del valor 0. Quan arriba al darrer valor torna al valor 0 i reinicia el compte. En el pas del darrer valor a 0 activa un senyal d' overflow.

Aquest senyal només es pot reiniciar escrivint un 1 en el mateix o habilitant la interrupció per overflow del comptador.

- Mode CTC, *Clear Timer on Compare match*. El comptador es posa a 0 quan el comptador arriba al valor carregat en registres determinats. Es genera, si així està programat, interrupció per comparació.
- Mode *PWM ràpid*. El comptador permet generar un pols PWM entre un valor baix i un d'alt, tenint un valor de comparació entremig. La sortida canvia de valor en arribar al valor de comparació i retorna al valor inicial en arribar al valor alt.
- Els modes CTC *amb correcció de fase* i *CTC amb correcció de fase i freqüència* són modes de funcionament amb doble pendent. Actuen amb valors de comparació entremig dels valors baix i alt del comptador que generen polsos PWM precisos. La diferència entre els dos modes està en els registres i valors de comparació.

Els modes de funcionament dels comptadors es basen en una:

- Una unitat de comparació que activa interrupcions (si aquestes estan habilitades) en arribar el comptador als valors emmagatzemats en els registres de comparació.
- La unitat de sortida permet programar el mode de funcionament de la sortida en funció del mode de funcionament del comptador.

Registres

El funcionament dels comptadors es basa la programació d'uns registres determinats, similars per tots els comptadors.

- Registres comptadors i de comparació: emmagatzemen dades i són simples quan el comptador és de 8 bits, i dobles si són de 16 bits.
  - *Timer counter*, TCNTi[1]. És el registre comptador.
  - *Output capture register*, OCRi. Registres de comparació.
  - *Input capture register*, ICRi. El registre emmagatzema el valor del comptador davant un event en l'entrada ICPi.
- Registres de control
  - *Timer Counter Control Register*, TCCRi (pot ser doble segons el comptador: A i B). Entre els senyals importants hi ha:
    - *CSi2:0*. Selecció del prescaler. El rellotge del comptador és el rellotge de sistema dividit pel factor seleccionat d'acord amb aquest senyal.
    - *COMxi1:0*[2]. Selecció del mode de sortida per comparació.
    - *WGMi3:0*. Selecció del mode de funcionament del comptador.
  - *Timer Interrupt Mask*, TIMSK i *Extended Timer Interrupt Mask*, ETIMSK. Registres de màscara d'interrupció. Reuneix els senyals d'habilitació d'interrupció per comparació i per overflow de tots els comptadors. ETIMSK existeix quan el comptador té més de 8 bits d'interrupció.
  - *Timer Interrupt Flag Register,* TIFR. És el registre que conté els senyals d'overflow o que indiquen que s'ha arribat al valor de comparació. Els flags s'activen en complir-se la condició. S'activen automàticament quan la corresponent interrupció està habilitada, si no s'ha de fer manualment.

[1] El subíndex i fa referència al número de comptador. L'ATmega8 té dos comptadors, l'ATmega128 en té 4.

[2] El subíndex x fa referència al registre comparador. Segons el comptador n'hi pot haver fins a 3: A, B i C.

6.1 Led on/led off

Primer programa simple de funcionament d'un comptador. El programa empra un comptador per encendre i apagar un led a una freqüència de 1 seg. L'exemple és un model per comprendre els passos a seguir quan es treballa amb comptadors i interrupcions.

Funcionalitat del programa

- Es tracta d'encendre un led a la freqüència d'1 Hz.

Ús del comptador

- Es farà servir el Timer1 amb interrupció per overflow.
- En produir-se overflow es genera interrupció
- El prescaler es posarà a 64, de manera que la freqüència base del rellotge del comptador sigui de f_{timer} = 1MHz/64 = 15625 Hz.

Tasques de la interrupció

- Reinicialitzar el comptador al valor tal que doni la freqüència desitjada
- Canviar el valor del led

Funcionament del comptador

- S'inicialitza el comptador amb el prescaler a 64 esmentat i funcionament en mode normal amb interrupció per overflow
- Com que el led canvia de valor cada cop que es provoca overflow, per aconseguir una freqüència d'1 Hz, es necessiten 2 overflow per segon, el que duu a un període de 1/2 segon.
- El valor a carregar al comptador quan es reinicia és de 2^16 – f_{timer}/2 = 57723 = 0xE17B. Per aconseguir aquesta freqüència va bé el Timer1 de 16 bits.

Programa que encén els leds a la freqüència d'1 Hz

```
; Programa 6.1: Ús de timers: control simple
; Gener 2010

.INCLUDE "m8def.inc"
.DEF srri       = r1            ; Registre temporal
.DEF temp       = r16           ; Registre temporal
.DEF rh         = r20           ; Registre comptador en subrutina d'espera
.DEF rl         = r21           ; Registre comptador en subrutina d'espera

        .ORG 0x0    RJMP inici
        RETI
        .ORG 0x8    RJMP isr_tmr1ovf
        RETI

inici:  LDI         temp, low(RAMEND); Byte baix de la posició de memòria final
        OUT         SPL, temp       ; SPL guarda la posició baixa de l'apuntador al stack
        LDI         temp, high(RAMEND); Byte alt de la posició de memòria
        OUT         SPH, temp       ; SPH guarda la posició alta de l'apuntador al stack
        RCALL       setupPorts      ; Setup dels ports
        RCALL       setupTimer1     ; Setup dels ports

        SEI
main:   NOP                         ; La rutina principal esdevé un bucle d'espera
        RJMP main

; Setup timer1
setupTimer1:LDI     temp,(1<<CS11)|(1<<CS10) ;Timer amb prescaler a 64
```

```
        OUT     TCCR1B,temp
        LDI     temp,(1<<TOIE1)
        OUT     TIMSK,temp
        RET

; Setup ports
setupPorts: SBI     DDRB, 2
        RET

;Rutina servei d'interrupció Timer1_OVF
isr_tmr1ovf: IN srri, SREG          ; Es guarda stack
        LDI     rh, 0xE1            ; Es reinicialitza el Timer1 a 57723 = 0xE17B
        LDI     rl, 0x7B
        OUT     TCNT1H, rh
        OUT     TCNT1L,rl
        IN      temp, PINB          ; Es complementa PORTB
        COM     temp
        OUT     PORTB, temp
        OUT     SREG, srri          ; Es retorna stack
        RETI
```

A RECORDAR (fixeu-vos en el programa)

Treballant amb interrupcions

- S'ha de definir el stack.
- En entrar en la rutina de servei d'interrupció cal guardar el stack, ja que el status pot canviar en les següents instruccions.
- En sortir de la rutina s'ha de recuperar el stack.
- Cal recordar que per habilitar de forma general les interrupcions cal emprar la instrucció SEI.

Respecte al comptador

- Una bona rutina d'inicialització (*setup*) del comptador estalvia molta feina posterior.

TREBALL PREVI:

- Fer el diagrames de flux del programa
- Realitzar la simulació per a comprovar-ne el funcionament
- Donar la taula que mostri tot els períodes que es poden tenir segons els Timers disponibles en l'ATmega8 i els diferents prescalers.

TREBALL DE LABORATORI:

- Realitzar el muntatge i comprovar el funcionament.
- Donar la freqüència exacta d'oscil·lació (comprovar amb l'oscil·loscopi).

6.2 Actuant sobre un servomotor

El servomotor

El servomotor és un motor de contínua amb realimentació molt emprat en tasques de posicionament per la seva fortalesa en mantenir la seva posició. Treballa amb un tren de polsos que el posiciona amb força a un angle determinat respecte al centre de l'eix. Degut a aquesta fortalesa és molt emprat en aeromodelisme.

El funcionament és molt simple. Consta de tres fils: alimentació (fil vermell), terra (fil negre) i

control (fil blanc/groc). El tren de polsos s'envia a través del fil blanc i posiciona el motor. La freqüència de funcionament és de 50 Hz (20 ms de període). L'ampla de pols va des d'1 ms a 2 ms. Quan es troba a 1.5 ms el motor queda centrat respecte a l'eix central. Conforme el tren de polsos es mou cap a 1 ms el motor es posiciona a una posició entre l'eix central i el màxim esquerra, arribant al valor màxim quan el pols és d'1 ms. Alternativament, si el pols avança cap a 2 ms el motor es centre cap a la dreta, quedant del tot a la dreta quan el pols és de 2 ms. Així, el control del motor es porta a terme per modulació de l'ampla de pols (PWM).

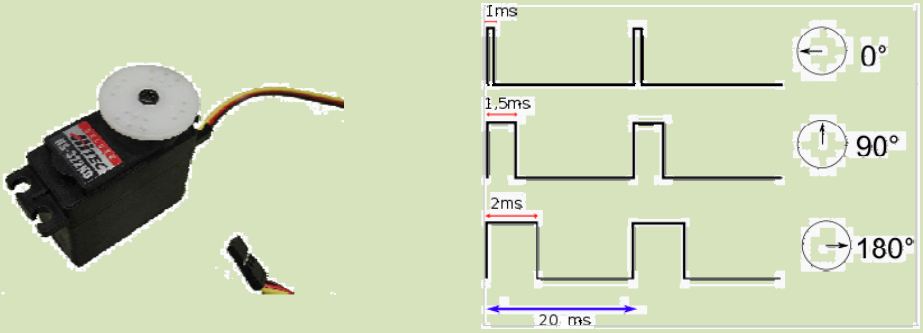

Figura 6.1

Quan a aquest motor se li treu la realimentació el seu funcionament és de gir constant. Amb un pols d'1.5 ms el motor queda aturat; conforme el pols va cap a 1 ms, el motor va girant més ràpid cap a l'esquerra; i si el pols va augmentant cap a 2 ms el motor va girant més ràpid cap a 2 ms.

Control del servomotor

Com a segon exemple, en aquest apartat s'emprarà un comptador per a realitzar el control del servomotor. Donat que es vol aprofitar la potència del comptador per a realitzar de forma autònoma (sense intervenció de la CPU) el control de pols PWM, s'emprarà el mode *Fast PWM* de funcionament del comptador.

El control del pols es realitza per divisió de la freqüència base de rellotge, el que proporciona la freqüència de rellotge del timer. Aleshores, el compte dels períodes del pols de la freqüència de treball del servomotor són múltiples d'aquesta freqüència. En concret, les dades de treball són les següents (es posa una taula amb els dos microcontroladors):

	ATmega8	**ATmega128**
Freqüència microcontrolador	1 MHz	16 MHz
Timer emprat	Timer1 (16 bits)	Timer1 (16 bits)
Prescaler considerat	8	64
f_{Timer}	1MHz/8 = 8 μs	16MHz/64 = 4 μs
Número de comptes per 1 ms → És el número de comptes per cada ms de compte del Timer	$10^{-3}/f_{Timer}$ = 125 = 0x007D	$10^{-3}/f_{Timer}$ = 250 = 0x00FA
Resolució	125 passos/ms	250 passos/ms
Número de comptes fins a 20 ms → És el número màxim de comptes per donar overflow en el Timer	$0.02/f_{Timer}$ = 2500 = 0x09C4	$0.02/f_{Timer}$ = 5000 = 0x1388
Pin de sortida per comparació amb register B	PORTB.2	PORTB.6

La figura 6.2 mostra la temporització que es segueix per controlar el servomotor emprant el mode *Fast PWM*. Per tant, el funcionament proposat és el següent:

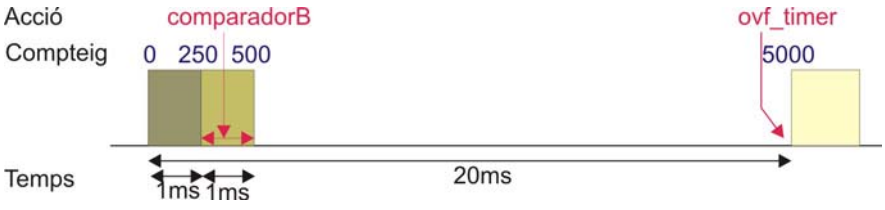

Figura 6.2

- El comptador inicia en 0.
- En el temps marcat pel registre de comparació B (OCR1B) es genera una interrupció per comparació que posa la sortida a 0 (s'ha de programar així, d'acord amb les especificacions del *datasheet*). El comptador continua comptant fins a arribar a 5000, valor que s'ha programat en el registre de comparació A (OCR1A) i que el mode de funcionament agafa com a valor *TOP* de compte. En aquest valor es genera una interrupció per overflow que torna el comptador a 0 i la sortida a 0.
- El valor a carregar en OCR1B ha d'estar entre 250 i 500, valors que corresponen a una duració d' 1ms i 2ms, respectivament. Aquest valor indica la posició on es deté el servomotor realimentat, o la velocitat de gir en motors sense realimentació.
- El mode funciona per interrupció per overflow (quan s'arriba a 20m) i per comparació amb registre B (que dóna la velocitat de gir).
- El mode de comparació de la sortida en posa en *Fast PWM*, valor 2 (senyal COM1B1:0). Això és, la sortida es posa a 0-lògic quan s'activa la rutina de servei d'interrupció per comparació del registre B i es torna a 1-lògic en arribar a l'overflow.
- Finalment cal recordar que, si es fa sortir el mode de sortida controlada per timer (COM1B1:0 ≠ 0), s'ha d'activar el driver del corresponent pin com a sortida.

Control del servomotor

El següent programa mostra el control del servomotor d'acord amb els paràmetres anteriorment posats. Algunes consideracions prèvies són:

- La metodologia de treball és com abans: declarar funcions d'inicialització dels perifèrics i després es posa el programa principal. Quan més modular sigui el programa més fàcilment es segueix.
- S'ha d'entendre bé la forma d'operar de cada mode del timer. Llegiu bé tots els paràmetres que hi intervenen, ja que un detall n'altera significativament el funcionament. Per exemple, en el nostre cas, no recordar que el TOP es defineix amb el registre de comparació A, pot portar a un ús inadequat d'aquest registre.
- Donat que s'empra un comptador de 16 bits, tots els registres són dobles, i per tant tots tenen el byte alt i el baix. Cal recordar inicialitzar els dos. A més, cal seguir l'ordre d'inicialització que hi ha en el programa: **per escriure primer s'escriu el byte alt i, per llegir, primer es llegeix el byte baix**.
- Per facilitar veure l'ampla de pols, s'ha posat una directiva constant, EQU, en el programa que facilita canviar l'ampla de pols i ser observat en l'oscil·loscopi.
- Finalment, per a guardar la dada de l'ampla de pols es fa servir el registre X (que agrupa als registres XH i XL). Tot i que aquí no té massa rellevància sí que en l'apartat 6.3, com es veurà, permet simplificar la suma.

```
; Programa 6.1: Ús de timers: control Fast PWM
; Joan Oliver. Gener 2010
; Compta: PREPARAT PER ATmega128 ➔ L'HEU DE PASSAR A ATMEGA8
; Vigileu adreces d'interrupció, paràmetres d'inicialització de registres i pins de sortida.
.INCLUDE "m128def.inc"
.DEF srri     = r1                ; Registre temporal
.DEF temp   = r16                 ; Registre temporal
.EQU constantH = 0x01;            ; Es defineix l'ampla de pols: byte alt i baix ' així a 2 ms
.EQU constantL = 0xF4;

        .ORG 0x0    RJMP inici          ; Adreces rutines de servei d'interrupció
```

```
        .ORG 0x1A  RJMP i_tmr1B
        .ORG 0x1C  RJMP i_tmr1ovf
        RETI

inici: LDI      temp, low(RAMEND); Byte baix de la posició de memòria final
        OUT     SPL, temp       ; SPL guarda la posició baixa de l'apuntador al stack
        LDI     temp, high(RAMEND); Byte alt de la posició de memòriaDDRB 7 23
        OUT     SPH, temp       ; SPH guarda la posició alta de l'apuntador al stack
        RCALL   setupPorts      ; Setup dels ports
        RCALL   setupTimer1     ; Setup timer1
        SEI
main: NOP
        RJMP main

; Setup timer1
; En mode FAST PWM i TOP per registre de comparació A, sortida per comparació invertida i prescaler a 64
setupTimer1: LDI temp, (1<<COM1B1)|(0<<COM1B0)|(1<<WGM11)|(1<<WGM10)
        OUT  TCCR1A,temp
        LDI  temp,(0<<CS12)|(1<<CS11)|(1<<CS10)|(1<<WGM13)|(1<<WGM12)
        OUT  TCCR1B,temp
        LDI  xh, 0x13           ; Càrrega valor d'overflow en OCR1A a 5000 (50 Hz)
        LDI  xl, 0x88
        OUT  OCR1AH, xh
        OUT  OCR1AL,xl
        LDI  xh, constantH      ; Càrrega ampla de pols en OCR1B
        LDI  xl, constantL
        OUT  OCR1BH, xh
        OUT  OCR1BL,xl
        LDI  temp,(1<<OCIE1B)|(1<<TOIE1); Habilitació interrupcions per overflow i comparació en reg B
        OUT  TIMSK,temp
        RET

; Setup ports
setupPorts: SBI DDRB, 6         ; Port de sortida OC1B
        RET

;ISR's timer1                   ; ISR's del timer: quan s'executa només deshabiliten flags
i_tmr1B: RETI
i_tmr1ovf: RETI
```

TREBALL PREVI:

- Fer el diagrama de flux de cada part del programa
- Donar el programa preparat per a la sessió de laboratori que es realitzarà amb el ATmega8
- Realitzar la simulació i comprovar el funcionament[3]

TREBALL DE LABORATORI:

- Realitzar el muntatge i comprovar el funcionament.
- Anotar la freqüència real de funcionament del timer i l'ampla de pols per diferents valors posats en el registre OCR1B.

[3] En el moment de realitzar aquesta memòria la simulació fallava en mostrar el pols de sortida

6.3. Control de motor segons distància de xoc

En aquest apartat es demana que implementeu, fent servir les dues pràctiques vistes fins ara, un detector de distància de xoc d'un hipotètic robot. Actuarà de la següent forma:

- Si el robot es troba lluny de l'obstacle, aquest anirà endavant.
- Si es troba massa prop (que xoqui) anirà enrere.
- I si es troba en una distància intermèdia, romandrà aturat.

Fixeu-vos, que amb el que s'ha vist fins ara, el que cal és unir els dos programes (detector de distància per infraroig i control de motor) tenint en compta aquestes consideracions:

- El registre OCR1B conté l'ampla de pols, que va des d'un mínim d'1 ms fins a 2 ms.
- El primer milisegon és fix, i a aquest cal sumar-li la distància que hi ha fins a l'obstacle.
- La distància a l'obstacle ve donada pel valor que ens retorna el detector d'obstacles per infraroig (fet a la pràctica anterior). Aquest retorna un valor que està entre 0 i 256.
- Aleshores només cal sumar[4] ambdós valors per a que el robot actuï davant el xoc imminent:
  - Quan el robot es trobi lluny el motor rebrà un valor 0 de l'infrarroig, fet que farà girar fort el motor en un sentit determinat.
  - Quan el robot es trobi prop el motor rebrà el valor 253 de l'ADC i, per tant, girarà fort cap a l'altre sentit.
  - Quan el robot es trobi just entremig el valor rebut de l'ADC provocarà un pols intermig en el motor que, a la distància precisa, l'aturarà.

Per tant, Hi ha dues possibles solucions a l'exercici:

- Unir els dos programes, treballant independentment els dos, i sumant la sortida del conversor al valor base que proporciona 1 ms al servomotor.
- Acoblant el funcionament del conversor en el funcionament del motor, d'acord a la figura 6.3.

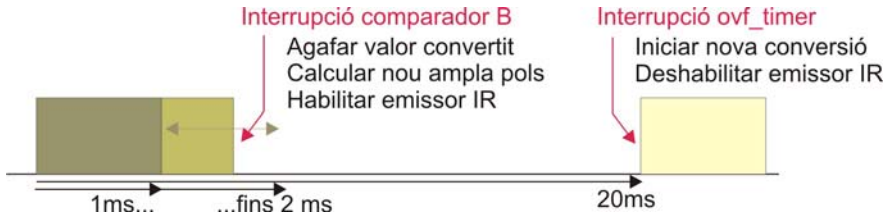

Figura 6.3

TREBALL PREVI:

- Donar un diagrama de flux (per blocs – no cal especificar totes les instruccions) del funcionament del programa, on quedi clar quan s'inicia cada tasca i les inicialitzacions que es fan en els registres.

- Donar el programa que s'ha simulat comprovant el funcionament

TREBALL DE LABORATORI:

- Realitzar el muntatge i comprovar el funcionament.

- Donar una taula de velocitat de gir segons distància on es trobi el robot (ajudeu-vos de l'oscil·loscopi).

[4] Segons el microcontrolador s'ha d'escalar: i) Si es treballa amb el ATmega128, sumant ambdós valors ja es té el pols que va des d'1ms a 2ms. Si es treballa amb el ATmega8 és millor escalar la sortida del ADCa un valor entre 0 i 256/2: d'aquesta forma el pols anirà des del valor 125 (corresponent a 1 ms) al valor 253 (corresponent als 2 ms).