

Generación de PWM para la familia HC08JL

Introducción

Todos los microcontroladores HC08 incluyen al menos un módulo de Timer que es muy útil para generar o capturar señales, ya sean estas periódicas en el tiempo o no.

En particular, el HC08JL3 posee un módulo de timer con dos canales, mientras que el HC08JL8 y el HC08JL16 poseen dos módulos de timers, ambos con dos canales, pero en cualquiera de ellos podemos configurar las funciones de “input capture” y “output compare”, y además generar PWM “unbuffered” y “buffered”.

Con la función de input capture (captura de entrada) podemos medir el tiempo transcurrido entre dos eventos, ya sean dos flancos ascendentes (rising-edge), dos descendentes (falling-edge) o entre dos flancos consecutivos independientemente del sentido (any-edge), lo que nos permite, entre otras cosas, conocer la frecuencia de una señal periódica, el ancho de un pulso, la distancia entre estos, etc.

Output compare (salida por comparación) permite generar en el pin de salida un “1” lógico (set), un “0” lógico (clear) o una conmutación entre “0” y “1” (toggle) cada vez que se produce una igualdad entre un contador interno del microcontrolador y un valor establecido en otro registro ya sea por el programador o por el programa en sí; de esta manera se pueden producir pulsos de duración y separación variables según se requiera.

PWM o pulse width modulation (modulación por ancho de pulso) es una aplicación práctica del output compare que genera una señal cuyo ancho de pulso dependerá de un valor (que se puede modificar) y que nos permite, desde codificar información (para transmitir, por ejemplo, con un transceptor infrarrojo) hasta utilizarlo como dimmer para variar la intensidad de una lámpara, la temperatura de un calentador resistivo o la velocidad de un motor.

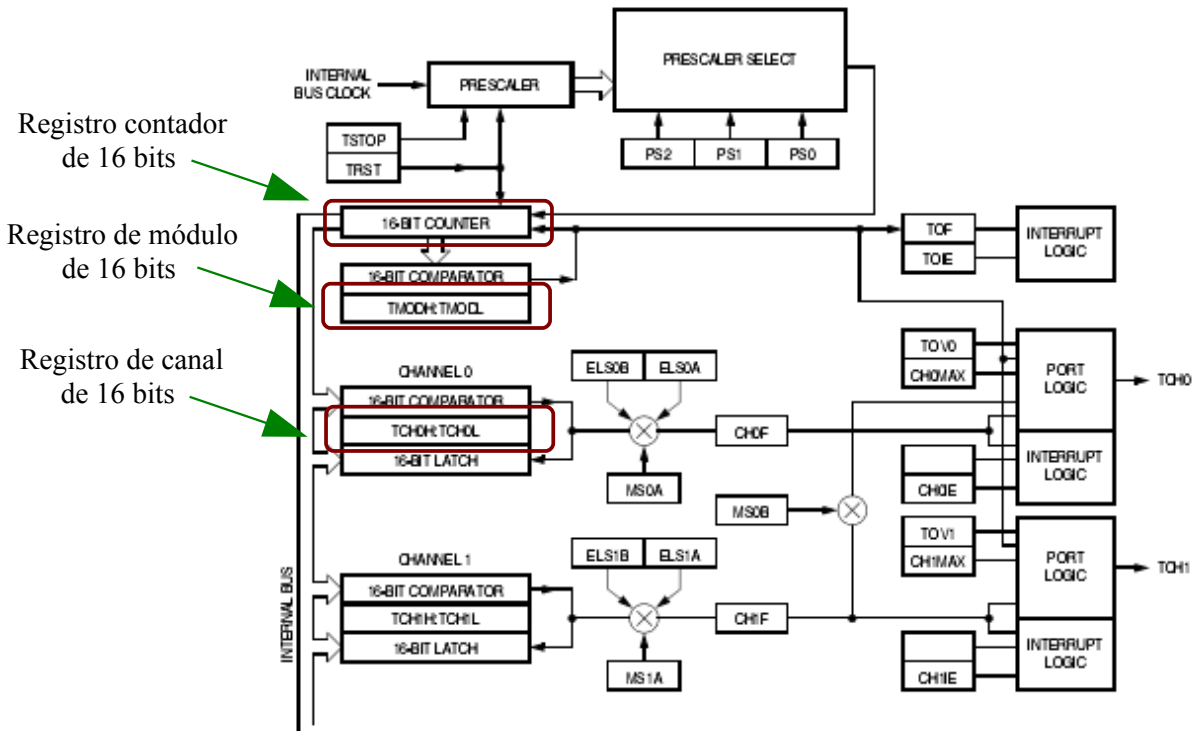


Figura 1: Interfaz del módulo Timer de la familia HC08JL

Principio de funcionamiento del TIM (Timer Interface Module)

La figura anterior ilustra el módulo de timer de un JL3, idéntico (a los fines de este apunte) a los encontrados en el JL8 y el JL16; estos últimos tienen dos módulos de timer en lugar de uno solo y la posibilidad de poner un reloj externo para el segundo timer.

El módulo cuenta, como muestra la figura, con dos canales de timers que pueden ser configurados independientemente como input capture u output compare; posee un contador de 16 bits (16-bit counter) que puede ser configurado como contador libre (free-running counter) o contador hasta módulo (modulo up-counter) y puede ser leído en cualquier momento sin afectar la cuenta; este registro provee la referencia de tiempo para ambas funciones mencionadas y tiene como referencia de entrada el resultado de dividir el reloj de bus interno del MCU por uno de los siete prescalers definidos. Cada ciclo de esa referencia producirá el incremento en uno del contador.

También existen tres comparadores de 16 bits que comparan el contador con los registros de 16 bits de módulo (TMODH:TMODL) y de canal, (uno por cada canal TCHxH:TCHxL). Cada vez que esa comparación se iguala se produce la acción programada o se dispara una interrupción según se haya configurado.

Generación de PWM

La circuitería utilizada es la asociada con la función de output compare, función que debe poseer el microcontrolador. Utilizando esta función y configurando la conmutación cuando desborda (toggle on overflow), se obtiene el PWM.

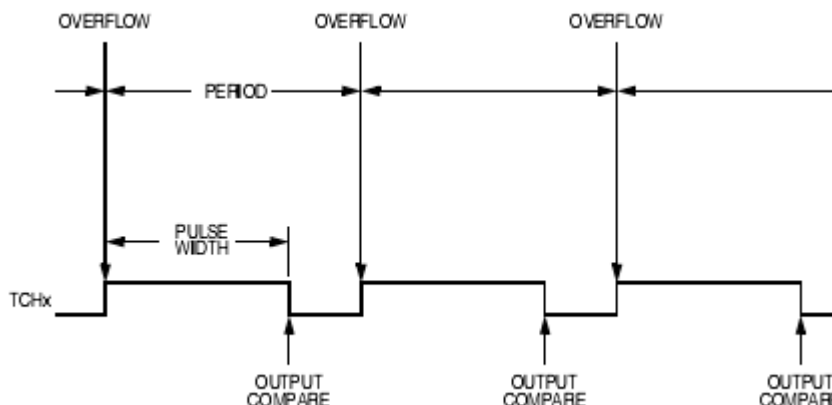


Figura 2: Período y ancho de pulso del PWM

Como muestra la figura, el período del PWM se obtiene configurando la característica de conmutación cuando desborda el contador (toggle on-overflow), mientras que el ancho de pulso lo dará la acción configurada como salida (set o clear en output compare).

El período (frecuencia) de la señal lo podemos fijar a través del registro de 16 bits TMOD (TMODH y TMODL), que indica la cantidad de tics que se deben producir en el contador libre para generar la conmutación por desborde (“overflow” en la figura) que se produce cuando ambos registros (contador y TMOD) se igualan. Teniendo en cuenta que el tiempo con que se produce cada tic (incremento de contador) depende del reloj de bus interno y el prescaler, podemos obtener varias frecuencias según se requiera.

Para generar el ancho de pulso se debe configurar ya sea set o clear (el caso de la figura) como salida del output compare, acción que se produce cuando se igualan el contador con el registro de canal, de esta manera, cambiando el valor del registro de canal se varía el ancho del pulso.

Observación: como el registro de módulo es único, si se utilizan los dos canales como PWM, en un instante podrán tener anchos de pulsos diferentes pero la frecuencia será la misma para ambos.

Existen dos variantes de PWM, la denominada unbuffered y la buffered. Para entender la diferencia entre ambas analizaremos un poco el funcionamiento de estas.

PWM unbuffered

En esta modalidad se puede configurar un PWM por cada canal del timer y se denomina así ya que para cada cambio de ancho de pulso sólo se requiere escribir el nuevo valor en el registro de canal correspondiente.

La desventaja de este método es que, si se cambia el registro de canal a un valor menor y el contador ya lo pasó sin haber alcanzado aún a producir el output compare de ese período, se produce una salida errónea por dos períodos como se muestra en la siguiente figura:

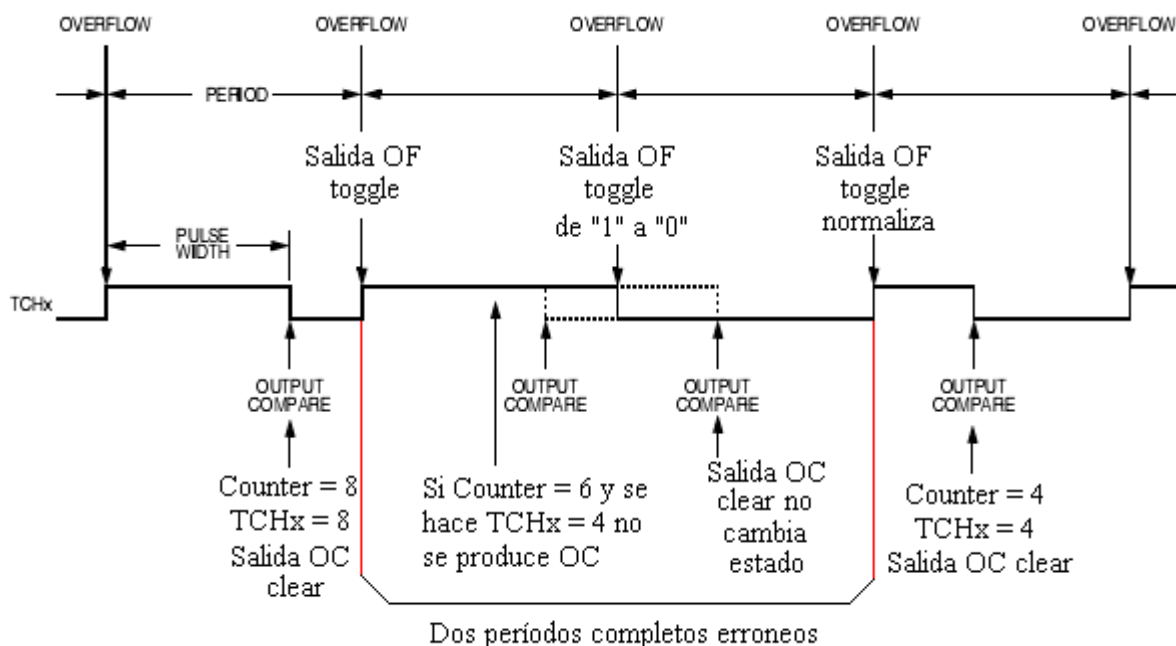


Figura 3: Escritura asincrónica del registro de canal

La manera de evitar esto es la siguiente:

- Si se desea cambiar el registro de canal a un valor menor, habilitar la interrupción de output compare de canal y en el código de atención de interrupción, configurar el nuevo valor del registro de canal. No configurar aquí el registro de canal si se desea poner un valor mayor porque se puede producir dos output compare en el mismo período.
- Si se desea cambiar a un valor mayor, habilitar la interrupción por overflow y cambiar ahí el valor del registro de canal.

De esta manera, a pesar de la complejidad sumada al configurar dos interrupciones, se puede tener un PWM de funcionamiento estable.

PWM buffered

En esta modalidad, los canales 0 y 1 se encadenan, trabajando en conjunto y presentando la salida por el pin correspondiente al canal 0 (el pin del canal 1 estará disponible como puerto de entrada-salida). Aquí, los nuevos valores del registro de canal se deben escribir en aquel que no esté activo

(el MCU comienza utilizando el registro del canal 0) y estará activo a partir del siguiente overflow, cambio que realiza el MCU; será responsabilidad del programador llevar el control de cual es el registro activo y cual es aquel que se deba modificar, pues si se modifica el registro activo, se estaría trabajando en la modalidad unbuffered.

De esta manera, logramos estabilidad de funcionamiento evitando la complejidad de codificar interrupciones, aunque se pierda la funcionalidad de uno de los canales de timer.

Configuración del módulo para PWM

Para un correcto funcionamiento del PWM se recomienda realizar los siguientes pasos para su configuración:

- **Detener el contador y ponerlo a cero**

Esto lo hacemos desde el registro TSC configurando los pines correspondientes.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
Write:	0			TRST				
Reset:	0	0	1	0	0	0	0	0

Figura 4: Registro TSC (TIM Status and Control register)

Los bits en cuestión son el bit 5 (TSTOP) para detener el contador y 4 (TRST) para poner el contador a cero, pero además, al hacer reset, también se resetea los bits correspondientes al prescaler (PS2 a PS0) y hay que volver a configurarlos. También se pueden setear ambos bits al mismo tiempo de modo que se detiene el contador en cero.

Podemos hacer:

`bset TSTOP,TSC ;Pone en 1 TSTOP, detiene contador`

`bset TRST,TSC ;Reset del contador`

o también:

`mov #$30,TSC ;Detiene y resetea contador`

- **Configurar el prescaler**

Para esto, primero tendremos que definir cual es el valor que se necesita de entre los valores que permite la siguiente tabla:

PS2	PS1	PS0	TIM Clock Source
0	0	0	Internal Bus Clock + 1
0	0	1	Internal Bus Clock + 2
0	1	0	Internal Bus Clock + 4
0	1	1	Internal Bus Clock + 8
1	0	0	Internal Bus Clock + 16
1	0	1	Internal Bus Clock + 32
1	1	0	Internal Bus Clock + 64
1	1	1	Not available

Figura 5: Tabla de valores de prescaler

Para definir el valor, junto con el valor del registro de módulo podemos aplicar la siguiente fórmula:

$$TMOD = F_{OSC} / (F_{PWM} * 4 * PS)$$

Donde F_{OSC} es la frecuencia del oscilador externo, F_{PWM} la frecuencia de PWM deseada (en las mismas unidades que F_{OSC}) y PS el valor de prescaler elegido.

Por ejemplo, si se desea obtener una frecuencia de PWM de 5Khz con un cristal de 9,8304Mhz hacemos:

$$TMOD = 9830,4 \text{ Khz} / (5\text{Khz} * 4 * 2)$$

$$TMOD \approx 246$$

donde se ha elegido un prescaler de 2 para que el TMOD sea de 8 bits (TMODH = 0x00).

Ahora podemos configurar el prescaler bit a bit o escribiendo el byte correspondiente en TSC

```
bclr PS2,TSC
```

```
bclr PS1,TSC
```

```
bset PS0,TSC
```

pero más cómodo será:

```
mov #\$21,TSC ;El 2 responde a que TSTOP debe quedar en "1" (detenido)
```

Sólo resta decir de este registro que el bit 7 (TOF) es la bandera de indicación de timer overflow y se pone en uno cada vez que hay un desbordamiento; por otro lado, el bit 6 (TOIE) habilita o no la interrupción por timer overflow y tendremos que ponerlo a uno si deseamos trabajar con esta interrupción.

- **Fijar el período de PWM requerido**

Simplemente escribir en TMOD el valor hallado con la fórmula anterior, para el ejemplo es de 246, entonces:

```
mov #\$00,TMODH
```

```
mov #\$F6,TMODL ;Entonces TMODH:TMODL = 0x00F6 (246)
```

Es importante escribir primero el registro TMODH y luego el TMODL (en ese orden) ya que si se hace al revés, el MCU no tomará la configuración, no tendremos ningún mensaje de error del compilador, simplemente no funcionará

- **Fijar el ancho inicial del pulso**

Si, por ejemplo, se desea inicializar un PWM en el canal 0, con un duty del 50%, para un TMOD de 246, el 50% será de 123, entonces hacemos:

```
mov #\$00,TCH0H
```

```
mov #\$7B,TCH0L ;Duty del 50%, TCH0H:TCH0L = 0x007B (123)
```

El valor en el Power On Reset y Reset de estos registros es indeterminado, así que será conveniente fijar el valor inicial (que bien puede ser cero si no se desean pulsos a la salida al iniciar el MCU).

- **Configurar la función y modalidad deseada**

Hasta ahora no hemos configurado que función cumplirá el timer; el MCU no sabe hasta aquí si será input capture, output compare (que además puede ser buffered o unbuffered) o PWM. Como esas funciones las puede realizar cada canal individualmente, tendrán los registros de status y control asociados que le indiquen a cada uno de ellos la función a cumplir. Estos registros son los

denominados TSC0 y TSC1, representados en la siguiente figura:

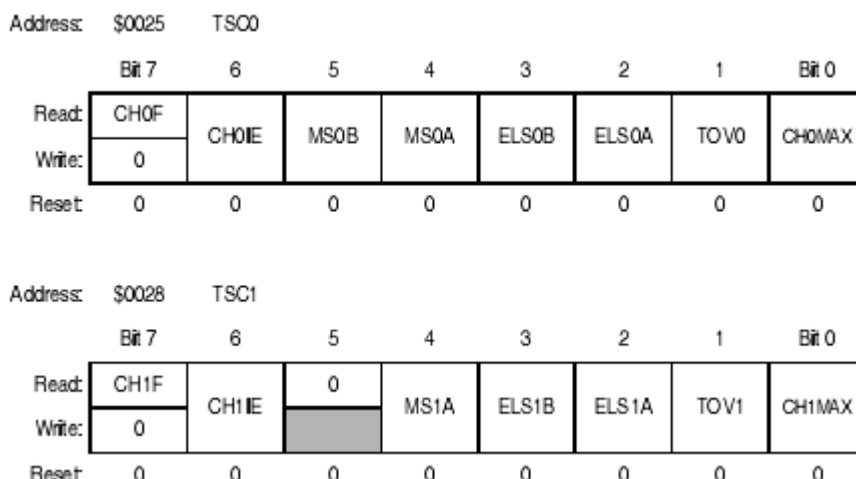


Figura 6: Registros TSC0 y TSC1

La configuración de los registros anteriores obedecen a la siguiente tabla:

MSxB	MSxA	ELSxB	ELSxA	Mode	Configuration
X	0	0	0	Output Preset	Pin under Port Control; Initial Output Level High
X	1	0	0		Pin under Port Control; Initial Output Level Low
0	0	0	1	Input Capture	Capture on Rising Edge Only
0	0	1	0		Capture on Falling Edge Only
0	0	1	1		Capture on Rising or Falling Edge
0	1	0	1	Output Compare or PWM	Toggle Output on Compare
0	1	1	0		Clear Output on Compare
0	1	1	1		Set Output on Compare
1	X	0	1	Buffered Output Compare or Buffered PWM	Toggle Output on Compare
1	X	1	0		Clear Output on Compare
1	X	1	1		Set Output on Compare

Figura 7: Tabla de selección de modo (mode), flanco (edge) y nivel (level)

De esta tabla surge el valor a escribir en el registro del canal correspondiente, para seguir con el ejemplo, si se desea configurar un PWM unbuffered en el canal cero, con clear en output compare y toggle en overflow deberíamos hacer:

```

bclr MS0B,TSC0 ;0:1 en MS0B:MS0A para PWM unbuffered
bset MS0A,TSC0
bset TOV0,TSC0 ;1 en TOV0 para toggle on overflow
bset ELS0B,TSC0 ;1:0 en ELS0B:ELS0A para clear on output compare
bclr ELS0A,TSC0
    
```

El escribir bit a bit este registro no es caprichoso, se debe hacer así; escribir este registro haciendo mov o sta provoca que la salida de PWM se active con un ciclo de 100% provocando efectos no deseados.

Además no es conveniente configurar la conmutación (toggle) en output compare pues no se podría obtener un duty de 0% y si llegase a ocurrir algún problema el PWM no podría volver a normalizarse

De estos registros falta mencionar los bits 6 y 7 que al igual que en el registro TSC son respectivamente el bit de habilitación de interrupción (ChxIE) y la bandera de overflow (ChxF). El bit 0 (CHxMAX) lo veremos en el apartado siguiente.

● **Arrancar el contador**

Se puede hacer en cualquier parte del programa donde se necesite y simplemente hay que poner a cero el bit TSTOP del registro TSC, a partir de ese momento, el contador comenzará a incrementarse y se producirá la acción de output compare configurada cuando llegue al valor en TCHxH:TCHxL y la de desbordamiento cuando se iguale a TMODH:TMODL. En este punto además reiniciará la cuenta desde cero.

Ciclos de trabajo de 100% y 0%, como alcanzarlos

El bit 0 del registro TSCx, CHxMAX permite alcanzar un ciclo de actividad de 100% o 0% dependiendo si se controla el ancho de la parte activa o de la parte no activa del ciclo de trabajo para evitar los errores que se producen cuando se igualan los puntos de output compare y overflow, pues si esto sucede, la conmutación por desborde tiene precedencia y la acción por output compare no se produce; entonces se obtiene a la salida un PWM de la mitad de la frecuencia original con un duty de 50%, efecto no deseado.

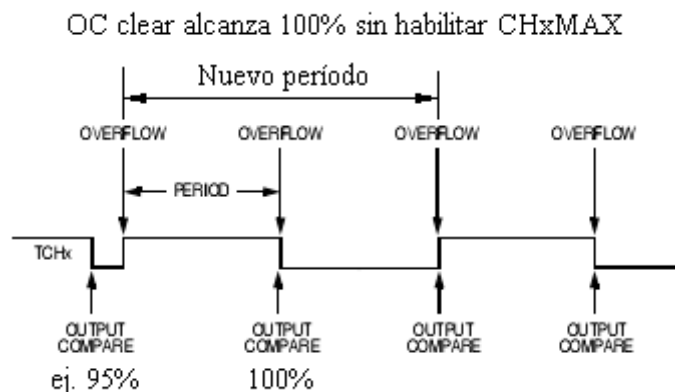


Figura 8: Efecto cuando se desea 100% de duty sin habilitar CHxMAX

Poner en uno el bit CHxMAX es la única forma de llegar a un duty de 100% (o 0% si se controla el ancho de la parte no activa), aunque también tiene algún defecto (existe una latencia) es sin duda la mejor opción para el funcionamiento deseado.

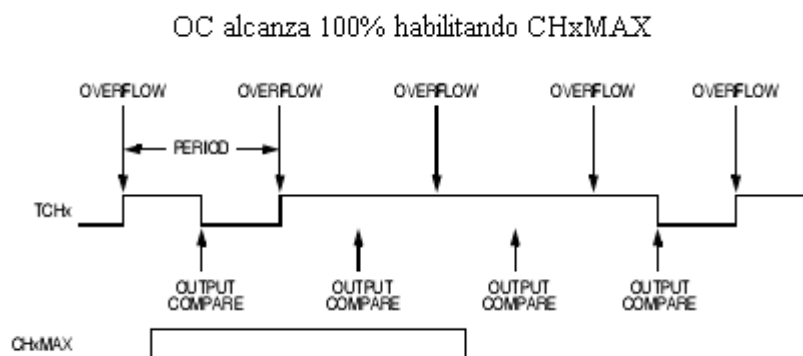


Figura 9: Efecto de setear CHxMAX para 100% de duty

Para permitir que el duty sea de 0% cuando se controla el ancho del pulso activo (o 100% si se controla el ancho de la parte no activa) simplemente basta con deshabilitar la conmutación por desborde (toggle on overflow) haciendo 0 el bit TOVx del registro TSCx correspondiente.

Conclusión

El módulo de timer es muy útil para realizar tareas relacionadas con el tiempo, ya sea interactuando con el exterior para medir o generar señales en función del tiempo o como método de generar demoras o bases de tiempos en aquellos micros (como los de esta familia) que no cuentan con un módulo de timebase dedicado.

En especial, las prestaciones que se alcanzan con las distintas configuraciones de PWM que se pueden obtener hacen que sea posible encontrar la adecuada según nuestra aplicación y la complejidad del programa que se requiera.

A continuación se agrega a modo de ejemplo un programa que utiliza ambos canales de timer configurados como PWM unbuffered en forma bipolar comandados por un mismo potenciómetro conectado en el canal 5 (PTB5) del conversor AD con el que se puede controlar, a través de un puente H, velocidad y sentido de giro de un motor de DC.

Bibliografía

De la Web de Freescale (www.freescale.com):

Datasheet MC68HC08JL3

Datasheet MC68HC08JL8

Datasheet MC68HC08JL16

Notas de aplicación AN2701

Notas de aplicación AN2475

;Ejemplo de control bipolar de dos canales PWM unbuffered con potenciómetro y ciclo 100%
 ;Autor: Marcelo Vieytes
 ;Charla brindada en el Instituto Nacional Superior del Profesorado Técnico (INSPT) 25/10/2007
 ;Basado en MCU MC68HC08JL3

#include 'jl3regs.inc'

RAMStart equ \$0080 ;Inicio sector memoria RAM (JL y JK iguales)
 RAMEnd equ \$00FF ;Fin memoria RAM (JL y JK iguales)
 ROMStart equ \$EC00 ;Inicio sector memoria ROM (para JK cambiar por \$F600)
 Vectors equ \$FFDE ;Inicio vectores de interrupción

TOF equ 7
 TOIE equ 6
 TSTOP equ 5
 TRST equ 4

MS0B equ 5
 MS0A equ 4
 ELS0B equ 3
 ELS0A equ 2
 TOV0 equ 1
 CH0MAX equ 0

MS1A equ 4
 ELS1B equ 3
 ELS1A equ 2
 TOV1 equ 1
 CH1MAX equ 0

COCO equ 7

org RAMStart

valor rmb 1

org ROMStart

main:

rsp ;Resetea Stack
 mov #\$11,CONFIG1 ;Apaga módulo COP y LVI

*** Esta porción de código inicializa toda la RAM en cero ***

Init_RAM:

ldhx #RAMStart ;Carga HX con dirección inicio RAM

LOOP_RAM:

clr ,x ;Borra posición apuntada por HX
 aix #\$01 ;Apunta HX a la posición siguiente
 cphx #RAMEnd+1 ;Compara con la posición siguiente a la última
 bne LOOP_RAM ;Si no es igual (no pasó por la última) vuelve a ejecutar

*** Fin inicialización de RAM ***

```

        clrh                ;Inicializa índice y acumulador
        clrx
        clra
        jsr    Init_Ports    ;Inicializa puertos
        jsr    Init_ADC      ;Inicializa módulo ADC
        jsr    Init_TIM      ;Inicializa timer
        bclr   TSTOP,TSC     ;Enciende contador
loop:
        lda    #$25          ;Selecciona el canal de entrada (Canal 5, PTB5)
        jsr    ADC_Conv      ;Realiza conversión
        sta    valor         ;Guarda valor obtenido
        cmp    #$FF          ;Compara con valor máximo
        beq    max_value     ;Si es igual, salta
        cmp    #$00          ;Compara con el mínimo
        beq    min_value     ;Si es igual, salta
        bset   TOV0,TSC0     ;Habilita toggle en overflow (TIM0)
        bset   TOV1,TSC1     ;Habilita toggle en overflow (TIM1)
        bclr   CH0MAX,TSC0   ;Deshabilita máximo (TIM0)
        bclr   CH1MAX,TSC1   ;Deshabilita máximo (TIM1)
        mov    #$00,TCH0H    ;Pone 0 en parte alta registro de canal (TIM0)
        mov    valor,TCH0L   ;Pone el valor hallado AD en parte baja
        mov    #$00,TCH1H    ;Pone 0 en parte alta registro de canal (TIM1)
        mov    valor,TCH1L   ;Pone el valor hallado AD en parte baja
        jsr    DELAY_10mS    ;Demora 10mS entre conversiones
        bra    loop
min_value:
        bclr   TOV0,TSC0     ;Deshabilita toggle en overflow (TIM0)
        bclr   TOV1,TSC1     ;Deshabilita toggle en overflow (TIM1)
max_value:
        bset   CH1MAX,TSC1   ;Habilita máximo (TIM1)
        bset   CH0MAX,TSC0   ;Habilita máximo (TIM0)
        bra    loop

;*** Subrutina de conversión A/D ***
ADC_Conv:
        sta    ADSCR          ;Inicia conversión continua (indica canal de conversión)
        brclr  COCO,ADSCR,*   ;Espera a que COCO = 1 (Fin conversión)
        lda    ADR            ;Borra COCO haciendo una lectura
        brclr  COCO,ADSCR,*   ;Desecha la primer lectura y realiza otra
        mov    #$1F,ADSCR     ;Apaga el conversor
        lda    ADR            ;Recupera el resultado de la conversión
        rts

;*** Fin subrutina ***

;*** Subrutina inicialización PWM ***
;* Fosc=9,8304MHz, Fpwm=10KHz (9,6KHz), PS=1 *
Init_TIM:
        bset   TSTOP,TSC     ;Detiene el contador seteando TSTOP
        bset   TRST,TSC      ;Resetea contador y preescaler seteando TRST
        mov    #$00,TMODH    ;Configura #$00FF en TMODH:TMODL para período de 100uS (104uS)
        mov    #$FF,TMODL
    
```

```

    bclr    MS0B,TSC0    ;0:1 en MS0B:MS0A para PWM unbuffered
    bset    MS0A,TSC0
    bset    TOV0,TSC0    ;1 en TOV0 para toggle on overflow
    bset    ELS0B,TSC0    ;1:0 en ELS0B:ELS0A para clear on output compare
    bclr    ELS0A,TSC0

    bset    MS1A,TSC1    ;1 en MS1A para PWM unbuffered (ya es MS0B=0)
    bset    TOV1,TSC1    ;1 en TOV1 para toggle on overflow
    bset    ELS1B,TSC1    ;1:1 en ELS1B:ELS1A para set on output compare
    bset    ELS1A,TSC1

    rts
;*** Fin subrutina ***

;*** Subrutina inicializa módulo conversor AD ***
Init_ADC:
    mov     #$60,ADCLK    ;Configura reloj para conversión (prescaler 8)
    rts
;*** Fin subrutina ***

;*** Subrutina inicialización de puertos ***
Init_Ports:
    mov     #$00,PTA      ;Inicializa Puerto A con valores conocidos
    mov     #$00,DDRA     ;Configura Puerto A como entrada
    mov     #$00,PTB      ;Inicializa Puerto B con valores conocidos
    mov     #$00,DDRB     ;Configura Puerto B como entrada
    mov     #$00,PTD      ;Inicializa Puerto D con valores conocidos
    mov     #$00,DDRD     ;Configura puerto D como entrada
    rts                  ;Sale de subrutina
;*** Fin subrutina ***

;*** Demora 10 mS ***
DELAY_10mS:
    ldx     #$5B
LOOP_3:
    lda     #$25
LOOP_1:
    decx
    beq     LOOP_0
    bra     LOOP_1
LOOP_0:
    decx
    beq     LOOP_2
    bra     LOOP_3
LOOP_2:
    rts
;*** Fin subrutina ***

;*** Subrutina fantasma que intercepta todas las interrupciones no esperadas ***
dummy_isr:
    rti
;*** Fin subrutina ***

```

```

org     Vectors
dw     dummy_isr     ;ADC_Int Vector interrupción conversor AD
dw     dummy_isr     ;KBI_Int Vector interrupción teclado
dw     dummy_isr     ;No usado
dw     dummy_isr     ;No usado
dw     dummy_isr     ;No usado
dw     dummy_isr     ;No usado
dw     dummy_isr     ;No usado
dw     dummy_isr     ;No usado
dw     dummy_isr     ;No usado
dw     dummy_isr     ;No usado
dw     dummy_isr     ;TOF_Int Vector interrupción overflow de timer
dw     dummy_isr     ;TIM1_INT Vector interrupción timer canal 1
dw     dummy_isr     ;TIM0_Int Vector interrupción timer canal 0
dw     dummy_isr     ;No usado
dw     dummy_isr     ;IRQ_Int Vector interrupción por IRQ
dw     dummy_isr     ;SWI_Int Vector interrupción por SWI
dw     main          ;RST_Int Vector interrupción por Reset
end
    
```