Temporizadores y contadores en tiempo real: El módulo Timer0 y el prescaler del PIC

Creado: 19/03/2004
Actualizado: 19/03/2004

By [-Ali-] #pic (irc hispano)

1. Introducción

A menudo al utilizar un microcontrolador nos encontramos con la necesidad de contar o generar eventos cada cierto tiempo. Para ayudar en este tipo de tareas, es habitual que los microcontroladores dispongan de circuitos internos para ello. Este circuito, es comúnmente denominado Timer/Counter (Temporizador/Contador) aunque también es habitual encontrarlo con el nombre de RTCC (Real Time Clock Counter).

Nos vamos a centrar en el Timer/Counter de 8 bits habitual en los microcontroladores PIC 16F84, denominado en la hoja de especificaciones como **módulo Timer0** (en otros modelos es posible encontrar adicionales módulos de 8 ó 16 bits cuyo el funcionamiento básico es el mismo).

Antes de explicar el funcionamiento y uso del Timer0, vamos de definir los siguientes conceptos para evitar confusiones:

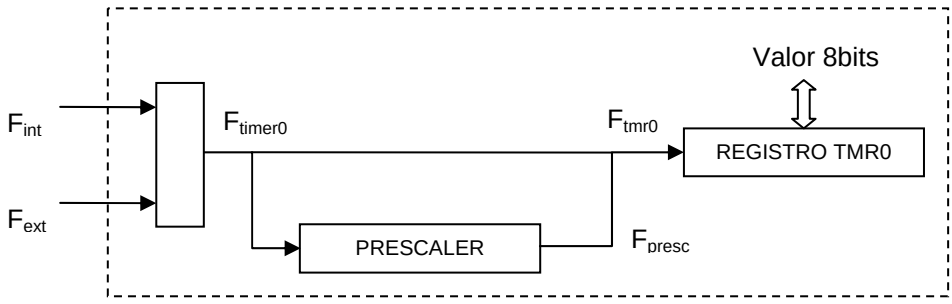
- *Frecuencia de oscilación* (F_{osc}): Frecuencia de trabajo externa del PIC (un cristal de cuarzo, un resonador, etc.).
- *Frecuencia interna de instrucciones* (F_{int}): Frecuencia del reloj interno de instrucciones generada a partir de la frecuencia de oscilación externa. Para los microcontroladores PIC no coincide con la F_{osc}, siendo un cuarto de esta:

$$F_{\text{int}} = \frac{F_{osc}}{4}$$

2. Estructura del Timer0

El Timer0 tiene cuatro componentes básicos:

- La entrada de reloj F_{timer0} (desde la patilla RA4/T0CKI o el reloj interno de instrucciones)
- Un circuito divisor de frecuencias programable o **prescaler**.
- Un registro contador TMR0.
- Los flags de interrupción utilizados por TMR0: GIE, T0IE y T0IF.



Esquema del modulo contador Timer0

3. Funcionamiento del Timer0

El Timer0 funciona como un temporizador o contador, según la procedencia de la señal de reloj que recibe. Debemos señalar que en ambos caso funciona de la misma forma, solo que el origen de la señal de entrada veremos que es mas adecuado usarlo para un cometido u otro.

En el caso que dicha señal provenga del reloj interno de instrucciones (F_{int}), el Timer0 se utiliza para generar interrupciones periódicas a través de una cuenta programada, pues conocemos la frecuencia de funcionamiento y en base al valor cargado en el contador podemos temporizar tiempos.

En el caso que dicha señal sea de una fuente externa al microcontrolador (F_{ext}), es especialmente útil para contar el número de pulsos que dicha señal genera en el tiempo ya que cada pulso de dicha señal incrementa el TMR0.

3.1. Entrada de reloj del modulo Timer0

La señal de reloj para el módulo Timer0 se puede obtener de dos formas: a través de la patilla de contador del microcontrolador (RA4/T0CKI), o bien utilizando el reloj interno de instrucciones. En el diagrama anterior se han denominado F_{ext}, F_{int}, respectivamente.

En el caso del reloj interno de instrucciones, debemos recordar que por diseño de los microcontroladores PIC, se obtiene dividiendo entre 4 la frecuencia de oscilación externa del microcontrolador PIC, es decir, $F_{int} = F_{osc}/4$.

3.2. El prescaler

El prescaler es un circuito que permite modificar la frecuencia del reloj de entrada del Timer0, dividiendo esta y generando una nueva señal de menor frecuencia a la salida que será la señal de reloj de entrada al registro TMR0.

El prescaler del registro TMR0 es activado a través de 4 bits en el registro OPTION, permitiendo dividir la frecuencia de una señal por 1, 2, 4, 8, 16, 32, 64, 128 o 256.
En caso de utilizar un divisor por 1, la señal de salida es la de entrada sin ningún cambio.

Por ejemplo, si usamos un oscilador externo de 4Mhz, entonces el reloj interno de instrucciones funciona a F_{int} = 4Mhz /4 = 1 Mhz. Si esta señal la pasamos por el prescaler configurado para una división por 4, la señal de salida del prescaler será de F_{presc} = 250 Khz.

Por ello no debemos confundir la frecuencia de trabajo del modulo Timer0, con la frecuencia de trabajo del registro TMR0 del Timer0: la primera es la frecuencia base utilizada por el modulo, mientras que la segunda es dicha frecuencia modificada que alimenta al registro TMR0.

3.3. El registro TMR0

El registro TMR0 es un corazón del módulo Timer0. Es un registro contador de 8 bits (podemos contar hasta 256 valores, entre 0 y 255) que a cada ciclo de su señal de reloj (F_{tmr0}) **incrementa automáticamente su contenido**.

La finalización de la cuenta se detecta cuando el contador pasa por 0: cuando el valor del registro TMR0 pasa de 255 a 0 (0xFF a 0x00), se activa el flag T0IF para indicar que ocurrió un desbordamiento ("overflow") y el registro continua incrementándose normalmente con cada pulso del prescaler.

Notas:

1. Si el microcontrolador esta dormido (mediante una instrucción SLEEP) y se utiliza como señal de reloj del modulo Timer0 la frecuencia interna de instrucciones, el Timer0 esta desactivado y no se incrementará el contador. Por tanto jamás se producirá ninguna interrupción por desbordamiento que permita salir del estado SLEEP.

2. Si la fuente de la señal de reloj del modulo Timer0 es externa, si se puede producir interrupción por desbordamiento, ya que aunque el Timer0 este desactivado el contador no depende activamente del microcontrolador sino que se incrementa a cada pulso de la señal externa. En este caso si se puede salir del estado SLEEP a través de la interrupción.

3.4. Flags de interrupción afectados

En el funcionamiento del Timer0 se ven afectados los siguientes flags de interrupción: GIE, T0IE, T0IF.

Si los flags GIE y T0IE están activados cuando el flag T0IF esta activo, se genera una interrupción, el bit GIE es automáticamente borrado para temporalmente prevenir que ocurran otras interrupciones mientras la rutina de interrupción esta siendo ejecutada, y el PIC salta hacia el "vector de interrupción" en la dirección de código 0x04.

La rutina de servicio de interrupción en esa localización debería comprobar el flag T0IF para determinar el origen de la interrupción y, si ocurrió una interrupción por desbordamiento, borrar el flag T0IF (debemos hacerlo nosotros pues no es automático), para evitar que el PIC vuelva a la rutina de interrupción cuando las interrupciones sean de nuevo habilitadas.

En este punto de la rutina de interrupción es donde deberíamos recargar el Timer0 con cualquier valor que deseemos, pues aunque el contador sigue incrementándose simultáneamente a la ejecución de instrucciones, no puede generar mas interrupciones al estar estas deshabilitadas.

Cuando se ha terminado de manejar la interrupción, debe finalizar con una instrucción RETFIE que automáticamente activará el bit GIE para habilitar las interrupciones y

devolverá el control al programa principal en la instrucción siguiente donde ocurrió la interrupción.

4. Cálculo de temporizaciones

Supongamos que el módulo Timer0 se alimenta con una señal de frecuencia F_{timer0}. Dicha señal pasa por el prescaler generando a su salida una nueva señal de frecuencia:

$$F_{presc} = \frac{F_{timer0}}{prescaler} \qquad T_{presc} = \frac{1}{F_{presc}}$$

(si el divisor del prescaler es 1, observamos que el efecto es como si la señal no se hubiera pasado a través del prescaler, en correspondencia al diagrama anteriormente mostrado).

Esta señal es el reloj del TMR0 que parte de un valor inicial V_{tmr0}, que se incrementa en cada ciclo del reloj. Cuando haya pasado $V_{tmr0} * T_{presc}$ segundos, el registro TMR0 se desbordará al pasar de 255 a 0 generando una interrupción.

En contra de lo que cabe pensar esto no es correcto; el TMR0 está contando lo que **falta para llegar al desbordamiento**, no el valor almacenado en él; es decir, si el TMR0 vale 8, el contador no contará 8 valores sino que se incrementará hasta alcanzar 255 y luego pasará a 0:

8, 9, 10, 11, 12, …, 253, 254, 255, 0

Desbordamiento

Luego el valor real que se cuenta es 256- V_{tmr0} y el paso por cero se realiza cada (256- V_{tmr0}) $*$ T_{presc} segundos. Es decir, el retraso que genera el Timer0 es:

$$Retraso_{tmr0} = (256 - V_{tmr0}) * T_{presc}$$

Generalizando para un contador de n bits:

$$Retraso_{tmr0} = ((2^n - V_{tmr0}) * prescaler) / F_{timer0} \quad [segundos]$$

La frecuencia de paso por cero de:

$$F_{tmr0} = 1/Retraso_{tmr0} = F_{timer0} / ((2^n - V_{tmr0}) * prescaler)$$

Partiendo de esta ecuación podemos obtener el valor inicial de TMR0 para que generar una interrupción por desbordamiento en un tiempo determinado:

$$V_{tmr0} = 2^n - ((Retraso_{tmr0} * F_{timer0}) / prescaler)$$

Donde F_{timer0} es la frecuencia interna de trabajo del Timer0 (la frecuencia interna de instrucciones o una señal de reloj externa).

4.1. Limitaciones de tiempo

En caso de la temporización los valores que podemos obtener están limitados por los divisores del prescaler, el valor máximo capaz de contar TMR0 y frecuencia de trabajo del Timer0.

El mayor retraso se puede obtener con el mayor divisor del prescaler (256), un valor 0 para el TMR0 (contará desde 0 a 255 antes del desbordamiento, es decir, 256 incrementos). La frecuencia de trabajo no la podemos conocer a priori así que no la usaremos directamente.

Por ejemplo, suponiendo una F_{osc} de 4 Mhz, obtenemos un retraso máximo de:

$$Retraso_{tmr0} = (256 * 256) / 1\ Mhz = 65{,}566\ ms$$

5. Ejemplos prácticos

Ejemplo 1: Elegir V_{tmr0} para generar un retraso de 1.5 ms usando un cristal del 10 Mhz.

$V_{tmr0} = 2^n – ((Retraso_{tmr0} * F_{int}) / prescaler)$
$V_{tmr0} = 256 – ((1.5\ ms * (10\ Mhz / 4)) / prescaler)$
$V_{tmr0} = 256 – (3750 / prescaler)$

Sólo resta dar valores al prescaler hasta obtener un valor adecuado, por ejemplo:

si prescaler = 256, V_{tmr0} = 242 (redondeado) ⇨ Retraso = 400 ns * 256 * (256-242) = 1,4336 ms
si prescaler = 128, V_{tmr0} = 227 (redondeado) ⇨ Retraso = 400 ns * 128 * (256-227) = 1,4848 ms
si prescaler = 64, V_{tmr0} = 197 (redondeado) ⇨ Retraso = 400 ns * 64 * (256-197) = 1,5104 ms
si prescaler = 32, V_{tmr0} = 139 (redondeado) ⇨ Retraso = 400 ns * 32 * (256-139) = 1,4976 ms
etc.

Estos valores deben entenderse como una guía y pueden ser aumentados o disminuidos sin mas que variar al valor contenido de TMR0, para obtener mayor o menor retraso.

Así podemos escoger prescaler = 64 y Vtmr0 = 197, dándonos un exceso de 0,01 ms.

Ejemplo 2: Generar un retraso de 1 segundo.

Vemos que con los máximos valores del TMR0 no podemos alcanzar el valor del retraso deseado, salvo cambiando la frecuencia de oscilación. Pero está limitada por el fabricante y no es factible poder usar frecuencias mayores. En este caso, la máxima frecuencia de oscilación de un microcontrolador PIC normal es de 4 Mhz, por lo que obtendríamos un retraso de 65,566 ms nada más.

Para solventar este problema, debemos utilizar otro mecanismo. Este se basa en utilizar el Timer0 como base para contar sobre otro registro, es decir, utilizar la interrupción por desbordamiento generada por el paso por cero del Timer0, como "señal de reloj" para decrementar otro registro hasta cero.

Supongamos que configuramos el prescaler para una división de 1:32 y una frecuencia de oscilación F_{osc} = 4 Mhz.

La frecuencia F_{presc} que obtenemos será:

$F_{presc} = F_{int} / 32 = (F_{osc} / 4) / 64 = 15625$ Hz

Si almacenamos en el TMR0 el valor 131, se alcanzará el cero después de 256-131=125 incrementos de TMR0, por lo que obtenemos una frecuencia de paso por cero del TMR0 de:

$F_{tmr0} = F_{presc} / V_{tmr0} = 15625$ Hz / 125 = 125 Hz

Es decir, cada segundo el TMR0 pasa por cero 125 veces generando una interrupción por desbordamiento cada vez. El siguiente paso es almacenar en el registro auxiliar un valor tal que, decrementando este una vez por cada paso por cero del registro TMR0, el registro auxiliar pase por cero cuando ha transcurrido 1 segundo.

$F_{tmr0} / V_{aux} = 1$ seg ⇨ Vaux = 125 Hz / 1 seg = 125

En definitiva, **buscamos un valor de frecuencia del prescaler (F_{presc}) que dividido por el valor del Timer0 (V_{tmr0}) de una frecuencia que sea multiplo de la que queremos generar. De esta forma tendremos la máxima exactitud al no tener decimales en los cálculos.** Pero no siempre es posible encontrar dicho múltiplo.

Este este resultado se podría haber obtenido igualmente aplicando sucesivamente la formula anteriormente calculada, pues el registro auxiliar utilizado funciona igual que el Timer0 con la unica diferencia que la frecuencia de entrada es la frecuencia de paso por cero del TMR0 y que **no hay un prescaler** (prescaler = 1).

F_{tmr0} = (4 Mhz / 4) / ((256-131)*64)) = 125 Hz