Basic Express
Nota de aplicación

Programación del Timer1 para la captura de flancos

El temporizador Timer1 y el pin de entrada

El procesador BasicX incluye un temporizador integrado denominado Timer1. Este temporizador puede utilizarse para numerosas funciones, una de las cuales es medir el retardo de tiempo hasta que tiene lugar un incremento o decremento del nivel lógico en el pin de entrada de captura.

¿Porqué se debe utilizar el Timer1 para la captura de flancos (edge capture)? La principal ventaja es que el procesador no debe estar dedicado exclusivamente a esperar la transición del pin.

En comparación, podemos considerar la llamada al sistema RCtime, que realiza una función similar. RCtime le permite configurar un pin de entrada/salida de uso general como input-tristate (impedancia alta) y medir el retardo para una transición. El problema es que RCtime dedica el procesador a la medición del tiempo. El reloj de tiempo real, la conmutación de tareas y el tráfico de la red se suspenden durante esta función.

El temporizador Timer1 no tiene este problema. Mientras que Timer1 espera que se produzca un evento de este tipo, el procesador puede seguir realizando el resto de las tareas.

Por otro lado, este enfoque tiene sus desventajas. Timer1 debería utilizarse únicamente cuando no se produzcan conflictos con otros recursos del sistema, como los puertos serie Com2 o Com3, así como los procedimientos InputCapture y OutputCapture, todos ellos dependientes del Timer1.

Puede preguntarse por qué no podemos utilizar el procedimiento InputCapture para la captura de flancos. Hay dos problemas – en primer lugar, la función primaria de InputCapture es medir el ancho de los pulsos, y no registra el retardo para iniciar la primera transición. En segundo lugar, el procedimiento se colgará si la transición nunca se produce.

Capacidades del temporizador Timer1

Podemos superar estos problemas utilizando el Timer1. Este temporizador es capaz de operar con 5 frecuencias discretas de marcado que oscilan desde aproximadamente 7,20 kHz a 7.37 MHz, lo que implica que puede utilizar el Timer1 para medir los intervalos de tiempo con resoluciones entre 136 ns y 139 µs.

Los siguientes recursos se utilizan para acceder al Timer1 para la captura de flancos (edge capture):

Tipo	Nombre	Descripción
Byte	TCNT1L	Timer/Counter1 Low Byte
Byte	TCNT1H	Timer/Counter1 High Byte
Byte	TCCR1B	Timer/Counter1 Control Register B
Byte	TCCR1A	Timer/Counter1 Control Register A
Byte	TIFR	Timer/Counter Interrupt Flag register
Byte	ICR1L	T/C 1 Input Capture Register High Byte
Byte	ICR1H	T/C 1 Input Capture Register High Byte

Consulte los siguientes ficheros PDF para obtener más información sobre el Timer1, que en realidad recibe el nombre Timer/Counter1. Este fichero se encuentra en el material de documentación de instalación de BasicX y del chip Atmel presentes en el procesador BasicX.

```
BX-01: File AT90S4414_8515.pdf, página 31.

BX-24, BX-35: File AT90S_8535.pdf, página 32.
```

Advertencia -- Timer1 debería utilizarse únicamente cuando no se produzcan conflictos con otros recursos del sistema, como los puertos serie Com2 o Com3, así como los procedimientos InputCapture y OutputCapture, todos ellos dependientes del Timer1.
.

Programación del Timer1 para la captura de flancos

Iniciación

Necesitamos inicializar el temporizador antes de empezar a utilizarlo. El primer paso es poner a cero el registro de control del temporizador Timer1 denominado TCCR1A. Esto desconecta el Timer1 de los pines de salida OC1A y OC1B, e inhabilita el funcionamiento de PWM:

```
Register.TCCR1A = 0
```

El siguiente paso es detener el temporizador:

```
Register.TCCR1B = 0
```

A continuación, debemos poner a cero los dos bytes del contador de Timer1. El byte alto debe escribirse en primer lugar, seguido del byte bajo:

```
Register.TCNT1H = 0
Register.TCNT1L = 0
```

También debemos poner a cero el Input Capture Flag 1 (ICF1), que se hace escribiendo 1 en el bit del registro TIFR:

```
Const ICF1 As Byte = bx00001000  ' BX-01
Const ICF1 As Byte = bx00100000  ' BX-24, BX-35

Register.TIFR = ICF1
```

Debería ponerse a cero el indicador de desbordamiento Timer/Counter1 (TOV1), de nuevo escribiendo el bit adecuado del TIFR:

```
Const TOV1 As Byte = bx10000000 ' BX-01
Const TOV1 As Byte = bx00000100 ' BX-24, BX-35

Register.TIFR = TOV1
```

El último paso de inicialización es para definir si nos enfrentamos a una subida/bajada del nivel lógico. El bit Input Capture1 Edge Select (ICES1) está encargado de esta tarea. Debería configurarse el ICES como 1 para la subida del nivel lógico y 0 para la caída del nivel lógico:

```
Const ICES1 As Byte = bx01000000

Register.TCCR1B = ICES1            ' For rising edge.
```

Inicio del Timer1

Una vez que el temporizador Timer1 ha sido configurado, se inicia al escribir unos de los 5 valores enumerados en los 3 bits de valor lógico inferior del registro TCCR1B. Los valores deducibles aparecen a continuación:

Valor TCCR1B	Resolución temporizador (µs)	Frecuencia marcación (Hz)	Rango máximo de tiempo (s)
1	0.135 633 7	7 372 800	0.008 889
2	1.085 069	921 600	0.071 11
3	8.680 555	115 200	0.568 9
4	34.722 22	28 800	2.276
5	138.888 9	7 200	9.102

En este ejemplo, utilizaremos la tasa de marcación de 28.8 kHz:

```
Register.TCCR1B = Register.TCCR1B + 4
```

Esperando una transición

Una vez que ha sido iniciado y arrancado el temporizador, el programa necesita comprobar el indicador ICF1, que se configura automáticamente cuando se produce la transición deseada. Tan pronto como se fije el indicador, el procesador también copia el valor del temporizador en los registros ICR1L y ICR1H. Si la transición no se produce antes de que el temporizador se desborde, el indicador de desbordamiento TOV1 se fija.

El siguiente código esperará una captura de flancos o un desbordamiento, lo que ocurra en primer lugar:

```
Dim HasOverflowed As Boolean, TIFRcopy As Byte

HasOverflowed = False
Do
    TIFRcopy = Register.TIFR

    ' Bail out if timer overflows.
    If ((TIFRcopy And TOV1) = TOV1) Then
        HasOverflowed = True
        Exit Do
    End If

Loop Until ((TIFRcopy And ICF1) = ICF1) ' Stop if edge is detected.
```

Durante este bucle, se siguen ejecutando otras tareas, como por ejemplo la gestión del tráfico de la red, el puerto serie I/O y el reloj de tiempo real. No es necesario que el procesador se dedique exclusivamente a esperar la transición de un pin.

Tenga en cuenta que se hace una copia del registro TIFR en vez de comprobar el registro directamente. Esto es así porque es necesario comprobar dos bits diferentes en el registro. Al hacer una copia, congelamos el patrón de bits del registro, por lo que no tenemos que preocuparnos de los cambios de los bits en los chequeos de captura y desbordamientos.

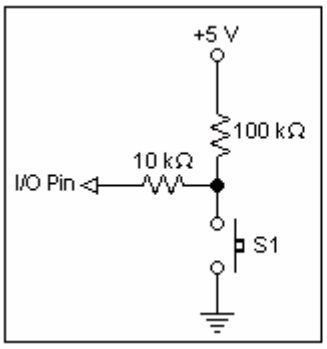
Si no se produce el desbordamiento, el valor del temporizador puede copiarse desde los registros ICR1L y ICR1H. Es importante leer en primer lugar el byte bajo, y en segundo lugar el byte alto de mayor peso:

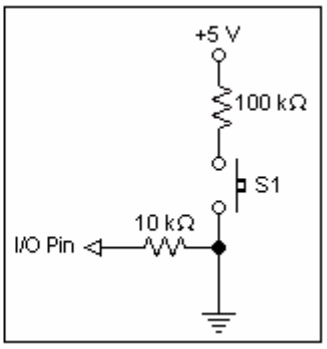
```
Dim LowByte As Byte, HighByte As Byte, Count As New UnsignedInteger

LowByte = Register.ICR1L
HighByte = Register.ICR1H

Count = CuInt(HighByte) * 256 + CuInt(LowByte)
```

El resultado final es *Count*, un valor con el rango de 0 a 65 535 unidades. La conversión de unidades a segundos depende de la tasa de marcación del temporizador Timer1. En nuestro ejemplo de 28.8 kHz, el factor de escala es aproximadamente de 3.47×10^{-5}, que nos da un limite de aproximadamente (65 535 unidades)$(3.47 \times 10^{-5}$ s/unit) = 2.27 segundos previos al desbordamiento.





Código de ejemplo

El código fuente para del programa de ejemplo se proporciona como un fichero independiente. El nombre del fichero es EdgeCaptureExample.bas.

Figura 1 Figura 2

Este ejemplo de programa puede utilizarse con cualquiera de los circuitos de las figuras anteriores conectados a los pines de entrada de captura (pin 31 en el BX-01; pin 12 en el BX-24; pin 20 en el BX-35). Dependiendo de qué circuito utilice y de si el interruptor está normalmente abierto o cerrado, puede personalizar la variable RisingEdgeSelected para especificar una subida / caída del nivel lógico para la captura.

2.00.A

Traducción Española: Alicia Bernal, revisión: Pablo Pompa www.superrobotica.com