Práctica 2: Timer + Teclado matricial

Prof. Alberto A. Del Barrio

1 Objetivos

En esta segunda práctica continuaremos aplicando los conceptos aprendidos en la primera sesión, tanto de configuración del entorno de desarrollo como del controlador de interrupciones de la placa S3CEV40. Además:

- Aprenderemos a utilizar los temporizadores y el teclado matricial.
- Utilizaremos estos periféricos en combinación con los leds, 7-segmentos y pulsadores.

2 Temporizadores

El S3C44B0X tiene 6 temporizadores de 16 bits, conectados según indica la figura 1. Cada uno puede ser configurado para operar por interrupciones o por DMA. Los temporizadores 0, 1, 2, 3 y 4 tienen asociada la capacidad de generar una onda cuadrada de la frecuencia con la que haya sido programado el temporizador, modulada por ancho de pulsos (PWM, pulse width modulation), son las seales TOUT* de la figura 1. Estas señales pueden sacarse por algún pin para transmitir una señal por modulación de ancho de pulsos, controlar un motor, etc. Esta funcionalidad no está disponible en el temporizador 5, que sólo puede utilizarse para temporización interna y no tiene pin de salida.

Los temporizadores son contadores descendentes que pueden ser inicializados con un determinado valor. Una vez configurados e inicializados, cada ciclo de reloj interno se decrementan. Cuando llegan a 0 generan una interrupción que podemos utilizar para realizar algunas tareas de forma periódica.

Como podemos ver en la figura 1, cada par de temporizadores (0-1, 2-3, 4-5) comparte un módulo de pre-escalado y un divisor de frecuencia. El divisor de los 4 primeros temporizadores tiene cinco señales divididas distintas: 1/2; 1/4; 1/8; 1/16 y 1/32. Los temporizadores 4 y 5 tienen un divisor con cuatro señales de reloj divididas: 1/2; 1/4; 1/8 y 1/16, y una entrada adicional TCLK/EXTCLK, que sirve para realizar la cuenta de una señal distinta que la señal de reloj (por ejemplo para contar pulsos externos al sistema, obtenidos de un pin). Como vemos, los divisores se alimentan con la salida de los módulos de pre-escalado, con lo que la frecuencia de cuenta se construye con un proceso de división en dos etapas, como detallaremos posteriormente.

Cada temporizador (excepto el 5) tiene un par de registros asociados, *TCNTn* y *TCMPn*. El registro de cuenta *TCNTn* se inicializa a un determinado valor y se decrementa en cada ciclo mientras el temporizador esté activo. El registro de comparación *TCMPn* se inicializa a otro valor y como su propio nombre indica se emplea para comparar con el registro de cuenta. El resultado de esta comparación se emplea para controlar la señal de salida. Al

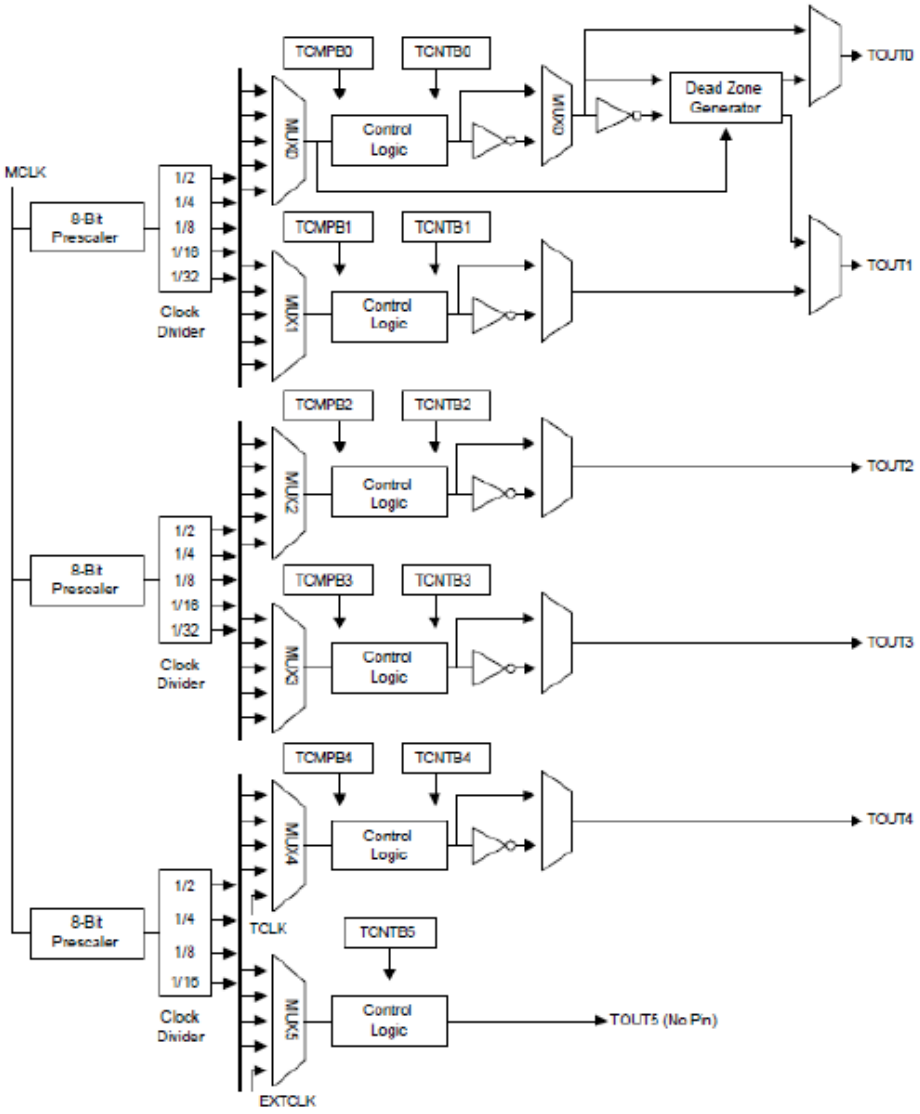

Figura 1: Esquema de interconexión de los temporizadores del S3C44B0X

comienzo de cada cuenta esta señal tiene un valor 0; cuando el contador alcanza el valor del registro de comparación la señal toma el valor 1. De esta forma podemos controlar exactamente la anchura de los pulsos (PWM), es decir el número de ciclos que debe estar a 0 y a 1.

Los temporizadores pueden funcionar en dos modos: auto-reload y one-shot. En modo auto-reload cuando el temporizador llega a 0, su valor inicial se vuelve a cargar automáticamente y se comienza una nueva cuenta atrás. Esto sirve para generar de forma muy precisa interrupciones periódicas. Por contra, en el modo one-shot cuando el contador llega a 0 se detiene la cuenta.

La figura 2 representa el diagrama temporal de un ejemplo de funcionamiento habitual de

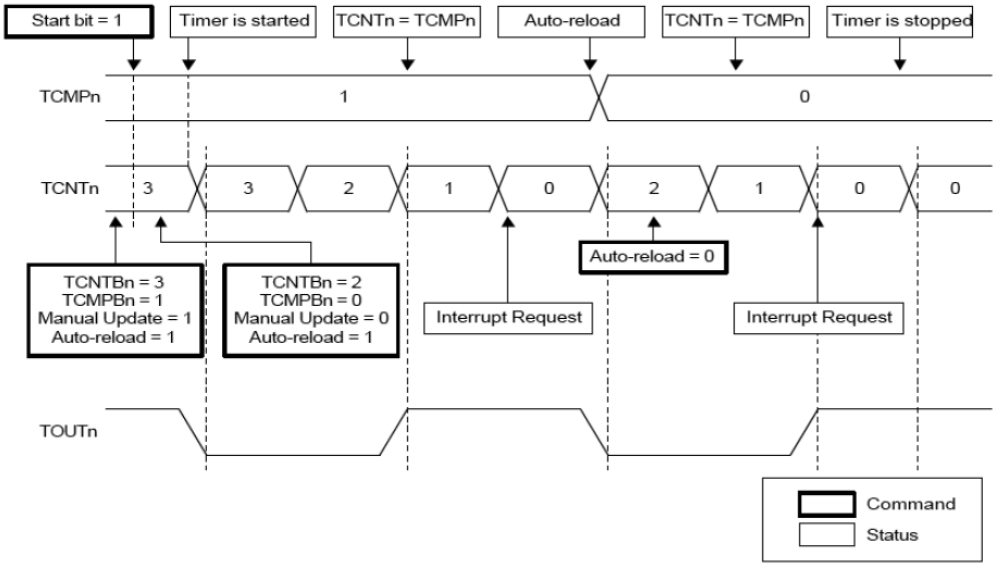

Figura 2: Diagrama temporal de los temporizadores del S3C44B0X

los temporizadores. Se utiliza un sistema de doble buffer que permite al usuario modificar el valor de recarga del temporizador mientras está contando, sin alterar o parar el contador. Normalmente el temporizador se inicializa escribiendo un valor en el registro de cuenta *TCNTn* y activando el *manual update bit*, como veremos más adelante. Mientras el contador está en marcha, se modifica el valor del registro de buffer asociado *TCNTBn* y el valor de la cuenta no se verá alterado. Este nuevo valor se utilizará para recargar *TCNTn* cuando la cuenta llegue a cero (si está en modo auto-reload). Lo mismo sucede con el registro de comparación *TCMPn*, que será recargado a partir del registro de buffer *TCMPBn*.

El valor actual de cuenta puede ser leído del registro de observación *TCNTOn*. En cambio, si se lee *TCNTBn* no se obtiene el valor actual de la cuenta sino el valor de recarga.

2.1 Inicializando los temporizadores

Como el valor del registro *TCMPBn* se copia en el registro de cuenta *TCNTn* sólo cuando *TCNTn* alcanza cero, el temporizador no puede ser inicializado simplemente escribiendo en *TCMPBn*. Para que el valor que escribimos en *TCMPBn* pase directamente a *TCNTn* debemos activar el *manual update bit*. La secuencia de inicialización del temporizador será por tanto:

- Escribir los valores iniciales de cuenta y comparación en *TCNTBn* y *TCMPBn* respectivamente.
- Activar el *manual update bit* del temporizador.

- Activar el bit de comienzo o *start bit* al mismo tiempo que se desactiva el *manual update bit.*

El control de los temporizadores (puesta en marcha, parada, actualización, modo autoreload, manual update, etc) se realiza mediante el registro *TCON*. Su funcionalidad puede encontrarse en el manual [1].

2.1.1 El reloj en los temporizadores

Además de inicializar el valor de cuenta, tendremos que establecer la frecuencia efectiva del reloj de los mismos por medio del preescalado y del divisor. Dicha frecuencia F sería:

$$F = MCLK/((preescalado + 1) * divisor) \quad (1)$$

donde, *MCLK* es la señal interna de reloj, el valor de preescalado está en el intervalo 0-255 y el valor del divisor puede ser: 2; 4; 8; 16; 32.

El valor de preescalado para los temporizadores se configura en el registro *TCFG0*, descrito en la tabla de la figura 3. El valor del divisor se configura usando el registro *TCFG1*. Se deja como ejercicio consultar el manual [1] para configurar el divisor.

Por tanto, la configuración inicial de un timer, en este caso el timer0, se haría con un código similar al del cuadro 1

Código 1: Ejemplo de inicialización de un timer

```
void timer_init(void){
  /* Config. del controlador de interrupciones */
  // Configurar las lineas como de tipo IRQ
  // Habilitar int. vectorizadas y la linea IRQ (FIQ no)
  // Enmascarar todas las lineas excepto Timer0 y el bit global
  /* Establece la rutina de servicio para TIMER0 */
  pISR_TIMER0=(unsigned)timer_ISR;
  /* Configurar el Timer0 */
  /* El resto de los timers se dejan a la config. por defecto) */
  // pre-escalado = 255
  // divisor = 1/2
  // TCNTB0 = 65535
  // TCMPB0 = 12800
  // establecer manual_update
  // iniciar timer y activar modo auto-reload
}
```

Función	Bits	Descripción
Longitud de la zona muerta	[31:24]	Estos 8 bits determinan la zona muerta. La unidad de tiempo de la zona muerta es la misma que la del temporizador 0.
Pre-escalado 2	[23:16]	Estos ocho bits determinan el factor de pre-escalado de los temporizadores 4 y 5.
Pre-escalado 1	[15:8]	Estos ocho bits determinan el factor de pre-escalado de los temporizadores 2 y 3.
Pre-escalado 0	[7:0]	Estos ocho bits determinan el factor de pre-escalado de los temporizadores 0 y 1.

Figura 3: Registro TCFG0

3 Teclado matricial

La placa S3CEV40 dispone un teclado matricial. En este tipo de teclados las teclas están dispuestas según un array bidimensional como el mostrado en la figura 4. Como puede observarse, el teclado es un array de líneas horizontales y verticales, junto con 16 interruptores (SB1-SB16) que al ser pulsados conectan una línea horizontal con una líea vertical. Se trata por lo tanto de un dispositivo extremadamente sencillo, que requiere de una lógica adicional para su uso.

En ausencia de pulsación, el controlador de la placa detecta una tensión de alta ('1' lógico). Cuando una tecla es pulsada, el interruptor asociado cortocircuita la columna con la fila correspondiente, de forma que la columna tendrá una tensión de baja ('0' lógico). Esta tensión de baja, provocará que la señal *EINT1* tome el valor de baja. Si el puerto G y el controlador de interrupciones se configuran de manera adecuada este flanco de bajada desencadenará una interrupción IRQ.

Para detectar la pulsación de una tecla, utilizaremos la técnica de *scanning*. Esta técnica consiste en enviar una tensión de baja por una línea horizontal y una tensión alta por el resto de líneas horizontales. Cuando una línea vertical tome la tensión baja, entonces la tecla que se encuentre en la intersección de ambas líneas, horizontal y vertical, será la tecla pulsada.

La dirección de control de teclado es la 0x06000000. A esta dirección la llamaremos *keyboard_base*. Para detectar una pulsación hay que aplicar el método de scanning sumándole un desplazamiento a *keyboard_base*. Una vez la dirección esté calculada, leeremos el valor del bus de datos. El valor leído estará compuesto por 4 bits. Si dicho valor es 0xF (todo '1's), sabremos que la fila no ha sido pulsada. Si es distinto de 0xF, la fila habrá sido pulsada y la posición que tenga un '0' identificará la columna del teclado matricial. La correspondencia entre los desplazamientos en la dirección y el dato leido viene dada por la figura 5. De esta manera, si en la dirección 0x060000FD hemos leído 0xD, sabremos unívocamente que hemos

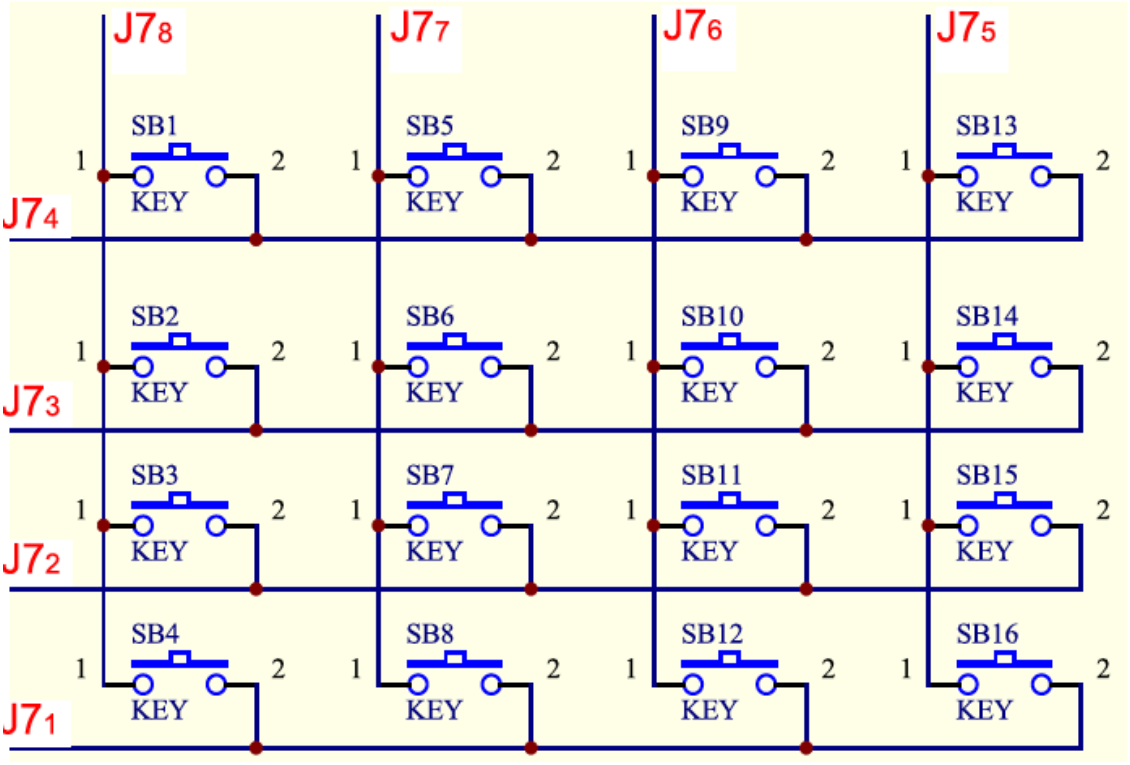

Figura 4: Circuito del teclado matricial 4x4

pulsado la tecla identificada como *SB9* en la figura 4. El significado que le demos a esa tecla ya dependerá del programador. En principio, si seguimos una disposición de los datos natural, en la primera fila (*SB1*, *SB5*, *SB9* y *SB13*) deberíamos mapear los valores 0, 1, 2 y 3, respectivamente.

3.1 El problema de los rebotes

Cuando se pulsa una tecla, no se origina una transición instantánea y estable de niveles de tensión entre los bornes del pulsador, como correspondería a una conmutación ideal. Esto

Dirección	**Dato**				
	0x7	0xB	0xD	0xE	0xF
0xFD	SB1	SB5	SB9	SB13	-
0xFB	SB2	SB6	SB10	SB14	-
0xF7	SB3	SB7	SB11	SB15	-
0xEF	SB4	SB8	SB12	SB16	-

Figura 5: Correspondencia entre dirección de lectura y dato leído para cada una de las teclas del teclado matricial

es debido a los rebotes. Obsérvese que este hecho también sucede con los pulsadores.

Para detectar correctamente una tecla es necesario esperar a que terminen los rebotes de presión. Una forma posible de hacerlo consiste en esperar durante un tiempo superior al tiempo de rebote de presión mínimo previsto (*trp*) después de haber detectado la pulsación (es decir, una vez se da servicio a la RTI asociada). Posteriormente se procederá a la identficación de la tecla. Para este tipo de teclado supondremos $trp = 20$ms.

Asimismo, se debe proceder también a la eliminación de los rebotes de depresión. Como no conocemos el momento en el que el usuario va a dejar de pulsar la tecla, la forma más segura de evitar los rebotes de depresión consiste en detectar la transición de depresión, y sólo entonces esperar durante un tiempo superior al tiempo de rebote de depresión (*trd*). Esto permitirá eludir los rebotes de depresión antes de dar por finalizado el proceso de identficación. Para detectar la transición de depresión se esperará a que la línea *EINT1* tome el valor '1' y después esperaremos $trd = 100$ms. Nótese que para leer el valor de *EINT1* consultaremos el bit 1 de *PDATG*.

4 Desarrollo de la práctica

4.1 Parte guiada

En esta primera parte completaremos un programa que ilumina los LEDs de manera alterna a intervalos regulados mediante interrupciones del temporizador 0.

Para ello descargaremos el código correspondiente a la práctica 2 y crearemos un proyecto nuevo siguiendo los mismos pasos que en la práctica anterior. Añadiremos la carpeta *commonEclipse* y los ficheros *main.c*, *timer.c*, *leds.c* e *init.asm*.

Nótese que en el fichero *timer.c* tenemos tanto las rutinas de inicialización como de tratamiento para el *timer0*.

4.2 Parte no guiada

Esta segunda parte constará a su vez de dos subsecciones. En la primera, trabajaremos con el teclado y el 8-segmentos, mientras que la segunda será un proyecto libre.

4.2.1 Teclado y 8-segmentos

Añadiremos los ficheros *keyboard.c* y *8led.c* al proyecto y completaremos el código de inicialización y de tratamiento de interrupción para el caso del teclado. Nótese que también habrá que modificar el *main* para inicializar el teclado matricial.

4.2.2 Proyecto libre

Después de las prácticas 1 y 2 el alumno debería estar familiarizado tanto con el entorno de desarrollo como con algunos componentes básicos de la placa S3CEV40. Por tanto, se propone el desarrollo de un proyecto que haga uso de todos los elementos estudiados hasta ahora (8-segmentos, leds, pulsadores, timers y teclado matricial) utilizando el sistema de interrupciones. En el caso de los timers, será obligatorio usar al menos 3 timers diferentes.

Bibliografía

[1] S3C44B0X RISC Microprocessor. Accesible en el Campus Virtual.