UNIVERSIDAD NACIONAL DE COLOMBIA

Tercer proyecto
TIMERS

Alejandro Ramírez Jaramillo

Universidad Nacional de Colombia sede Manizales Email: alramirezja@unal.edu.co

Introducción

Los temporizadores (Timers) son periféricos del microcontrolador que permiten generar un reloj con una frecuencia distinta del reloj del micro, esto haciendo uso de preescaladores y contadores, tenga en cuenta que la frecuencia del Timer será menor o igual que la frecuencia del reloj de sistema. Para este proyecto se hará uso del Timer 2 de la tarjeta STM32L476 y el led 2 conectado al pin 5 del puerto A, cargando el registro del preescalador para obtener distintas frecuencias del Timer. Este ejemplo fue diseñado para una tarjeta STM32L476 y se hará en el lenguaje de programación assembler.

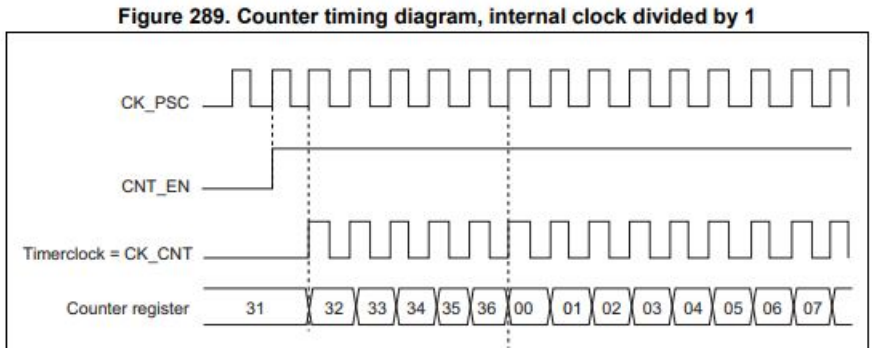

Figure 1. Frecuencia TIMER respecto al reloj interno, RM0351 Reference Manual, Pag. 1016

Contenido

1 Registros

Address:

+ RCC: 0x40021000
+ GPIOA: 0x48000000
+ TIM2: 0x40000000
+ NVIC: 0xE000E100

1.1 Inicialización de los relojes de los periféricos

1.1.1 RCC_AHB2ENR

+ Offset:0x4C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNG EN	HASHE N	AESEN (1)
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DCMIE N	ADCEN	OTGFS EN	Res.	Res.	Res.	GPIOIE N	GPIOH EN	GPIOG EN	GPIOF EN	GPIOE EN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN
	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 2. Distribución bits AHB2ENR, RM0351 Reference Manual, Pag. 251

Para el ejercicio el usuario usará el led 1 que se encuentra en la tarjeta, que está conectado al pin 5 del puerto A.

1.1.2 RCC_APB1ENR1

+ Offset:0x58

Registro que habilita o deshabilita el reloj para ciertos periféricos, con este se habilitara el reloj del TIMER 2 que se va a usar en el ejercicio.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1 EN	OPAMP EN	DAC1 EN	PWR EN	Res.	CAN2 EN	CAN1 EN	CRSEN	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN(1)	USART3 EN	USART2 EN	Res.
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Res.	Res.	WWD GEN	RTCA PBEN	LCD EN	Res.	Res.	Res.	TIM7 EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2 EN
rw	rw			rs	rw	rw				rw	rw	rw	rw	rw	rw

1. Available on STM32L45xxx and STM32L46xxx devices only.

Figure 3. Distribución bits APB1ENR1, RM0351 Reference Manual, Pag. 253

1.2 Configuración de entradas y salidas

1.2.1 GPIOx_MODER

+ Offset:0x00

Con este registro establezca el pin 5 del puerto A como una salida.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 4. Distribución bits MODER, RM0351 Reference Manual, Pag. 305

1.2.2 GPIOx_ODR

+ Offset: 0x14

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 5. Distribución bits ODR, RM0351 Reference Manual, Pag. 307

Para poner en HIGH un pin específico basta con escribir un uno en el bit cuya posición corresponda con el número del pin.

1.3 Configuración del TIMER

1.3.1 TIMx_PSC

+ Offset: 0x28

TIMx prescaler, con este registro se establece el preescalador del TIMER x, el cual funciona como un divisor de frecuencia programable. La frecuencia del reloj a la salida del preescalador esta dada por la ecuación:

$$CK\_CNT = \frac{f_{CK\_PSC}}{PSC[15:0]+1}$$

Donde:

- f_{CK_PSC} es la frecuencia de entrada al prescaler (Clock Prescaler)
- CK_CNT es la frecuencia de salida del prescaler (Clock Counter)
- PSC[15:0] es el valor contenido en el registro PSC.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 6. Distribución bits TIMx_PSC, RM0351 Reference Manual, Pag. 1076

1.3.2 TIMx_ARR

+ Offset: 0x2C

TIM x auto-reload register, almacena el valor hasta el cual va a contar el TIMER, funciona distinto si el contador trabaja de forma ascendente o descendente .

- Contador Ascendente: El valor del contador aumenta hasta alcanzar el valor contenido en el registro ARR, después de alcanzarlo el contador se pone en cero y vuelve a comenzar.
- Contador Descendente: El valor del contador disminuye hasta llegar a cero, entonces se carga en el contador el valor contenido en el registro ARR y vuelve a empezar.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 7. Distribución bits TIMx_ARR, RM0351 Reference Manual, Pag. 1077

1.3.3 TIMx_DIER

+ Offset: 0x0C

TIM x DMA/interrupt enable register, habilita y deshabilita distintas interrupciones del TIMER, entre las cuales se encuentra la interrupción de actualización que habilitaremos para el proyecto.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 8. Distribución bits TIMx_DIER, RM0351 Reference Manual, Pag. 975

1.3.4 TIMx_CR1

+ Offset: 0x00

TIM control register 1, este registro se usará para habilitar el contador de el TIMER.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 9. Distribución bits TIMx_CR1, RM0351 Reference Manual, Pag. 969

1.3.5 TIMx_SR

+ Offset: 0x10

TIMx status register, contiene banderas de eventos del TIMER, para este ejercicio se hace uso de la interrupción de actualización y por lo tanto es la bandera que se deberá limpiar al ejecutar la interrupción para establecer como ejecutada una solicitud de interrupción.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6IF	CC5IF
														rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Figure 10. Distribución bits TIMx_SR, RM0351 Reference Manual, Pag. 977

1.3.6 TIMx_CNT

+ Offset: 0x24
+ Reset Value: 0x0000 0000

TIM x counter, en los primeros 16 bits de este registro [15:0] se encuentra el valor del contador del Timer, puede ser leído y sobrescrito.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 11. Distribución bits TIMx_CNT, RM0351 Reference Manual, Pag. 990

1.3.7 NVIC_IPR7

Interrupt priority register. Las interrupciones dependiendo de su tipo y entrada tienen una posición y prioridad determinada, con esa información puedo usar este registro para dar prioridad a esa interrupción.

+ Adress offset: 0x400 + 0x04*x
+ Reset value: 0x0000 0000

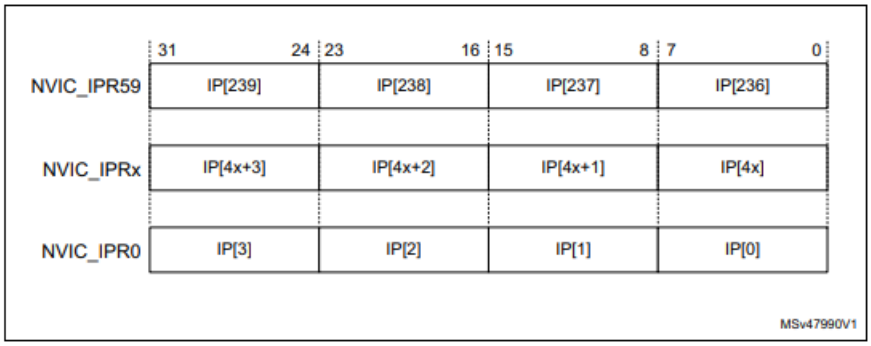

Figure 12. Distribución bits NVIC_IPR1, PM0214 Programming Manual, Pag. 215

27	34	settable	TIM1_CC	TIM1 capture compare interrupt	0x0000 00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000 00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000 00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000 00B8

Figure 13. Prioridad interrupciones, RM0351 Reference Manual, Pag. 398

Para dar prioridad a una interrupción, se carga en el registro IPRx un valor de x tal que al hacer la operación del bit IP el resultado sea la posición de la interrupción mostrada en la tabla anterior.

1.3.8 NVIC_ISERx

Interrupt set-enable register x, registro encargado de habilitar las interrupciones, dado el número de interrupciones disponibles se dividen en 8 grupos (x=0,1,2,...7). Como sabemos la interrupción del TIM2 está en la posición 28 así que se usaría ISER0.

+ Adress offset: 0x100 + 0x04*x
+ Reset value: 0x0000 0000

NVIC_ISER0 bits 0 to 31 are for interrupt 0 to 31, respectively

NVIC_ISER1 bits 0 to 31 are for interrupt 32 to 63, respectively

....

NVIC_ISER6 bits 0 to 31 are for interrupt 192 to 223, respectively

NVIC_ISER7 bits 0 to 15 are for interrupt 224 to 239, respectively

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SETENA[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETENA[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Figure 14. Interrupciones habilitadas, PM0214 Programming Manual, Pag. 210

+0: Interrupción deshabilitada.
+1: Interrupción habilitada.

2 Ejemplo TIMER

El TIMER que se usará en el ejercicio encenderá y apagará un led cada cierta cantidad de tiempo, esto lo hace alternando el valor de un registro cada vez que pasa ese tiempo, pero es necesario aclarar como cuenta ese tiempo. Para el ejemplo a continuación se tiene un ARR=36 y un preescalador de 2.

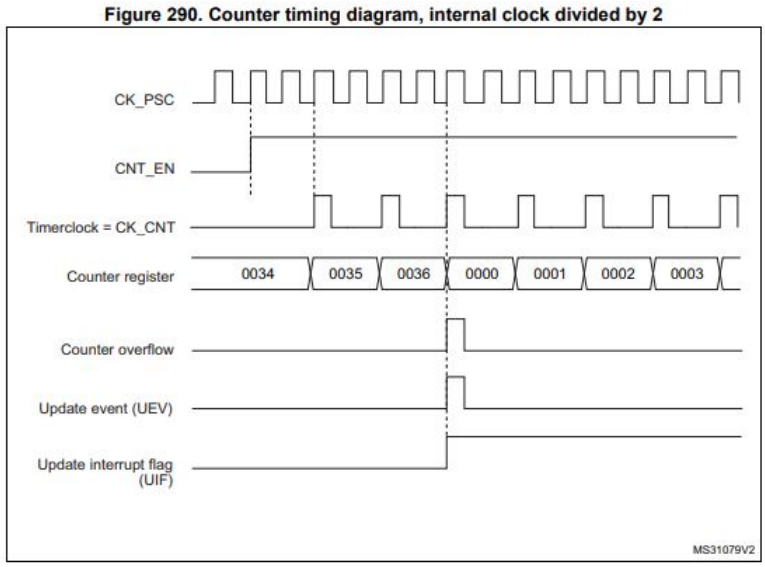

Figure 15. RM0351 Reference Manual, Pag. 1017

Se explicaran las señales de cada fila:

- CK_PSC: Reloj interno del microcontrolador, su frecuencia depende de como este configurado y de la alimentación de la tarjeta.
- CNT_EN: Habilita el reloj CK_CNT, comienza el conteo n ciclos después, siendo n el divisor del reloj (PSC[15:0]+1).
- CK_CNT: Reloj del contador, con una frecuencia igual a $\frac{f_{CK\_PSC}}{PSC[15:0]+1}$, en cada ciclo se aumenta en 1 el contador del timer.
- Counter register: Registro que almacena el contador del TIMER, aumenta en 1 cada ciclo del CK_CNT, cuando alcanza el valor de ARR se reinicia al ciclo siguiente (para el caso ascendente) o si llega a 0 toma el valor de ARR en el ciclo siguiente (para el caso descendente).
- Counter overflow: Cuando el contador del TIMER se reinicia, se genera una señal de que ya se alcanzó el valor máximo.
- Update Event: Anteriormente se había habilitado la interrupción de actualización, este evento (señal) es el que genera esa interrupción para que se ejecute el código.
- Update Interrupt Flag: Es la bandera que indica la solicitud de interrupción de actualización, es limpiada usando el registro SR.

3 Código Blink Led con Timer

El código que se estudiado encenderá y apagará el led que se esta conectado al pin 5 del puerto A cada cierta cantidad de tiempo definido por el TIMER.
Operaciones:

- orr → OR lógico
- and & = → AND lógico

3.1 Dirección de registros

Se inicializan los registros usando los valores de offset y direcciones que se encuentran en la sección Registros.

```
// You can find the base addresses for all peripherals from Memory Map section
// RM0351 on page 78. Then the offsets can be found on their relevant sections.

// Constants
.equ     PRESCALER,      1999

//       RCC   base address is 0x40021000
//   AHB2ENR register offset is 0x4C
.equ     RCC_AHB2ENR,   0x4002104C // RCC AHB2 peripheral clock reg

//       RCC   base address is 0x40021000
//   APB2ENR register offset is 0x58
.equ     RCC_APB1ENR1,   0x40021058 // RCC APB1 peripheral clock reg

//       GPIOA base address is 0x48000000
//   MODER register offset is 0x00
//   ODR   register offset is 0x14
.equ     GPIOA_MODER,   0x48000000 // GPIOA port mode register
.equ     GPIOA_ODR,     0x48000014 // GPIOA output data register

//       TIM2  base address is 0x40000000
//   PSC   register offset is 0x28
//   ARR   register offset is 0x2C
//   DIER  register offset is 0x0C
//   CR1   register offset is 0x00
//   SR    register offset is 0x10
//   CNT   register offset is 0x24
.equ     TIM2_PSC,   0x40000028 // TIM2 prescaler register
.equ     TIM2_ARR,   0x4000002C // TIM2 auto-reload register
.equ     TIM2_DIER,  0x4000000C // TIM2 interrupt enable register
.equ     TIM2_CR1,   0x40000000 // TIM2 control 1 register
.equ     TIM2_SR,    0x40000010 // TIM2 status register
.equ     TIM2_CNT,   0x40000024 // TIM2 counter register

.equ     NVIC_IPR7,    0xE000E41C
.equ     NVIC_ISER0,   0xE000E100
```

3.2 Configuración GPIOs

Debido a que se hará uso del led que esta conectado a la tarjeta es necesario activar el reloj para el puerto A y configurar el pin 5 de este puerto (al cual esta conectado el led) como una salida. Primero activamos el reloj para este periférico:

```
ldr r6, = RCC_AHB2ENR       // Load peripheral clock reg address to r6
ldr r5, [r6]                // Read its content to r5
orr r5, 0x1                         // Set bit 0 to enable GPIOA clock
str r5, [r6]                // Store result in peripheral clock register
```

Después configuramos el pin 5 del puerto A como una salida:

```
ldr r6, = GPIOA_MODER       // Load GPIOA MODER register address to r6
ldr r5, [r6]                // Read its content to r5
and r5, 0xFFFFF7FF          // Write 01 to bits 11, 10 for P5
str r5, [r6]                // Store result in GPIOA MODER register
```

3.3 Configuración TIMER 2

Habilitamos el reloj para el TIMER.

```
ldr r6, = RCC_APB1ENR1      // Load peripheral clock reg address to r6
ldr r5, [r6]                // Read its content to r5
orr r5, 0x1                 // Set bit 0 to enable APB1ENR1 clock
str r5, [r6]                // Store result in peripheral clock register
```

Se establece el valor del preescalador, en este caso no se aplica una operación lógica al valor en el registro, sino que se carga una constante en el:

```
ldr r6, = TIM2_PSC          // Load TIM2 PSC register address to r6
ldr r5, = 0x3E7             // load 999 constant to r5 register
                                        // fCK_PSC / (PSC[15:0] + 1) = 4 MHz / n + 1 =
timer clock speed
str r5, [r6]                // Store result in TIM2 PSC register
```

Escogemos el valor hasta el que va a contar el Timer:

```
ldr r6, = TIM2_ARR          // Load TIM2 ARR register address to r6
ldr r5, [r6]                // Read its content to r5
and r5, 0xFA0                   // Write 4000 ARR = IN_interrpt/T_timer
str r5, [r6]                // Store result in TIM2 ARR register
```

Ahora hay que escoger que la fuente de interrupción del Timer, que en este caso es la interrupción de actualización (Update event) que ocurre cuando el contador del Timer alcanza el valor en el registro ARR, cuando esto sucede el contador se reinicia y se activa la bandera del evento de actualización, que provoca que se ejecute la interrupción:

```
ldr r6, = TIM2_DIER         // Load TIM2 DIER register address to r6
ldr r5, [r6]                // Read its content to r5
orr r5, 0x1                     // Set bit 0 to update interrupt enable
str r5, [r6]                // Store result in TIM2 DIER register
```

Se habilita esta interrupción y se le asigna una prioridad.

```
ldr r6, = NVIC_ISER0
ldr r5, [r6]
orr r5, 0x10000000
str r5, [r6]

ldr r6, = NVIC_IPR7
ldr r5, [r6]
orr r5, 0x10
str r5, [r6]
```

Para finalizar se hanilita el contador del Timer:

3.4 Manejador TIMER 2

Es la sección de código que se ejecutará cada vez que haya una interrupción por actualización. Entonces se carga el valor del contador en el registro r5 y se compara con una constante, en caso de ser iguales va a saltar al código con la marca "int" y en caso de no ser igual va a mantener el led apagado.

```
TIM2_IRQHandler:                    /* TIM2 global interrupt
*/
```

```
        ldr r6, = TIM2_CNT
        ldr r5, [r6]
        cmp r5, 0x7D0
        beq     int
```

Cuando salta a "int" enciende el led y limpia la bandera de interrupción de actualización:

```
int:
        ldr r6, = GPIOA_ODR             // Load GPIOA output data register
        ldr r5, [r6]                    // Read its content to r5
        orr r5, 0x0020                  // write 1 to pin 5
        str r5, [r6]                    // Store result in GPIOA output data register

        ldr r7, = TIM2_SR
        ldr r8, [r7]
        and r8, 0xFE
        str r8, [r7]                                    //Clear any pending event on SR

        bx lr
```

4 Ejercicios prácticos

- Configure el Timer para que el led se encienda y se apague cada 3 segundos, para hacer el toggle del led se recomienda usar la operación eor (XOR lógico).
- Usando el Timer genere una señal de PWM conectada al led y configure su duty cicle a: 30%, 50% y 80%.

5 Referencias

+ PM0214 Programming manual, STM32 Cortex-M4 MCUs and MPUs programming manual.
+ RM0351 Reference manual, STM32L4x5 and STM32L4x6 advanced Arm-based 32-bit MCUs.
+ MB1136 Schematic board.