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 C.

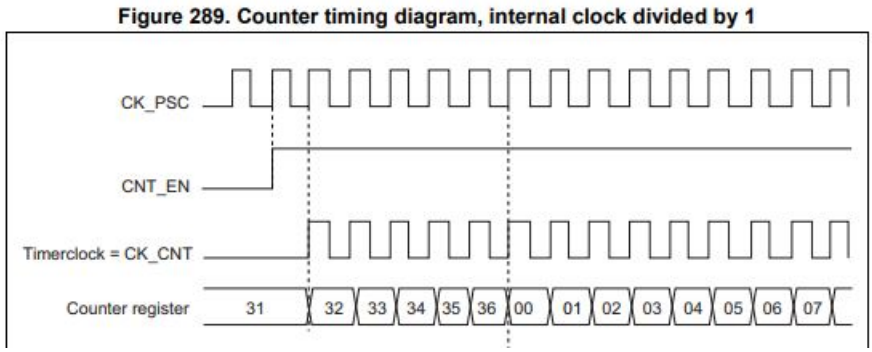

Figure 1. Frecuencia TIMER respecto al reloj interno, RM0351 Reference Manual, Pag. 1016

Contenido

1 Registros

1.1 Inicialización de los relojes de los periféricos

1.1.1 RCC→AHB2ENR

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

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

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. 303

1.2.2 GPIOx→ODR

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. 305

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

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. 1074

1.3.2 TIMx→ARR

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. 1074

1.3.3 TIMx→DIER

TIM x DMA/interrupt enable register, habilita y deshabilita distintas interrupciones del TIMER, entre las cuales se encuentra la actualización de interrupción 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. 972

1.3.4 TIMx→CR1

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. 966

1.3.5 NVIC_SetPriority(TIM2_IRQn, 2)

Establece la prioridad de una interrupción, en este caso al TIMER 2 se le va a asignar un nivel de prioridad 2.

1.3.6 NVIC_EnableIRQ(TIM2_IRQn)

Habilita la interrupción del TIMER 2, para que se ejecute una sección de código especifica cada vez que el TIMER tiene un sobre flujo.

1.3.7 TIMx_SR

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→DIER, RM0351 Reference Manual, Pag. 975

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.

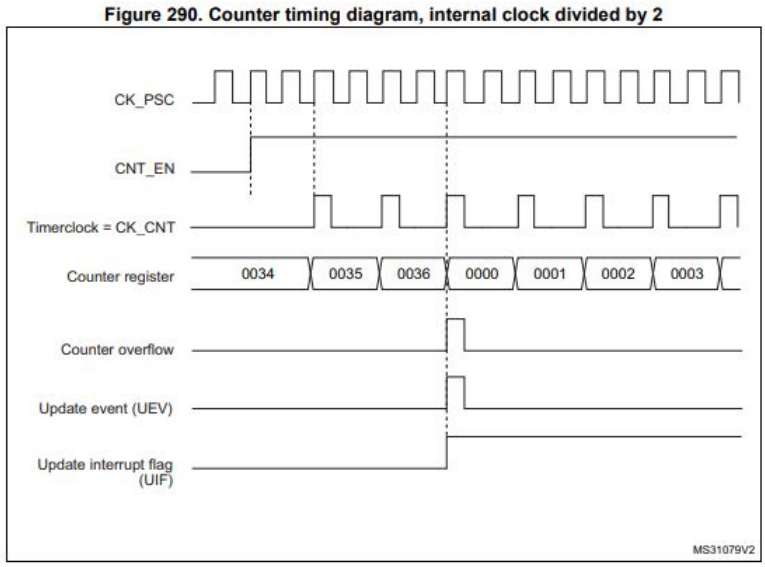

Figure 11. RM0351 Reference Manual, Pag. 1017

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

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:

- $|=$ $\longrightarrow$ OR lógico
- $\&=$ $\longrightarrow$ AND lógico
- $\wedge=$ $\longrightarrow$ XOR lógico

3.1 Inicialización

Se importan las librerías que se usaran y se defina la variable COUNT que se va a usar después.

```
* Libraries include
**********************************************/
#include "stm32l476xx.h"
#include <stdbool.h>
/**********************************************
* definition variables
**********************************************/
#define COUNT 10
```

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.

```
// enable GPIOA clock
RCC->AHB2ENR = 0x1;
// Make GPIOA Pin5 as output pin (bits 1:0 in MODER register)
GPIOA->MODER &= 0xFFFFF3FF;
GPIOA->MODER |= 0x00000400;
```

3.3 Configuración TIMER 2

Habilitamos el reloj para el TIMER.

```
// enable TIM2 clock
RCC->APB1ENR1 |= (1 << 0);
```

Se establece el valor del preescalador y el contador (usando la variable COUNT), definiendo la frecuencia con la que el TIMER ejecutará la interrupción.

```
// 40 Khz / n + 1 =  timer clock speed
TIM2->PSC = 40680;
// set total count
TIM2->ARR = COUNT;
```

Se habilita esta interrupción y se le asigna una prioridad.

```
NVIC_SetPriority(TIM2_IRQn, 2); // Priority level 2
// enable TIM2 IRQ from NVIC
NVIC_EnableIRQ(TIM2_IRQn);

// Enable Timer 2 module (CEN, bit0)
TIM2->CR1 |= (1 << 0);
```

3.4 Manejador TIMER 2

Es la sección de código que se ejecutará cada vez que haya una interrupción por actualización. En primer lugar verifica que la interrupción que fue habilitada es la interrupción de actualización ($TIM2 \rightarrow DIER \,\&\, 0x01$), después comprueba que la interrupción actual sea a causa de esta a través de la bandera de la interrupción específica y si es así, limpia la bandera con un cero.
Después de esto se aplica XOR al pin de salida para apagarlo o encenderlo. Al finalizar vuelve a la ejecución del código original mientras espera la siguiente señal de interrupción.

```
void TIM2_IRQHandler(void)
{

    // clear interrupt status
    if (TIM2->DIER & 0x01) {
        if (TIM2->SR & 0x01) {
            TIM2->SR &= ~(1U << 0);
        }
    }

    GPIOA->ODR ^= 0x0020;


}
```

4 Ejercicios prácticos

1. Usando la ecuación del reloj del timer (CK_CNT) calcule cual es la frecuencia del encendido y apagado del Led.

2. Usando el preescalador del TIMER duplique el tiempo que el led pasa encendido/apagado.

3. Use la variable COUNT para disminuir a la mitad el tiempo que el led pasa encendido/apagado.

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.