UNIVERSIDAD NACIONAL DE COLOMBIA

Cuarto proyecto
PWM

Alejandro Ramírez Jaramillo

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

Introducción

PWM (Pulse width modulation) es un tipo de modulación que consiste en cambiar el ciclo útil de una onda cuadrada para controlar la energía que se va a entregar a una carga, también puede usarse para transmitir información, pero esta aplicación no es muy popular. La tarjeta STM32L476 permite generar un PWM internamente usando los TIMERS de la tarjeta, de los cuales podremos controlar características como la frecuencia y el ciclo útil usando los registros correspondientes y produciendo esta señal en los pines de la tarjeta.

Este ejemplo consiste en aumentar el ciclo útil del PWM cada vez que se presione el botón de la tarjeta hasta llegar a 100, entonces disminuirlo cada vez que se presione el mismo botón hasta que el ciclo útil se igual a cero y comenzar el proceso de nuevo. Este ejemplo fue diseñado para una tarjeta STM32L476 y se hará en el lenguaje de programación Assembler.

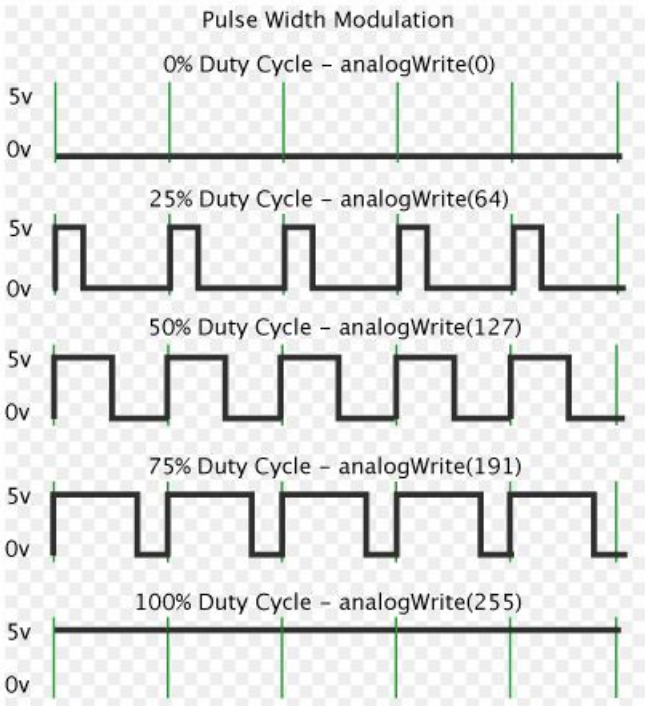

Figure 1. PWM con distintos valores de ciclo útil

Contenido

1 Registros

Address:

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

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.	RNGEN	HASHEN	AESEN (1)
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DCMIEN	ADCEN	OTGFSEN	Res.	Res.	Res.	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	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 registro 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
LPTIM1EN	OPAMPEN	DAC1EN	PWREN	Res.	CAN2EN	CAN1EN	CRSEN	I2C3EN	I2C2EN	I2C1EN	UART5EN	UART4EN(1)	USART3EN	USART2EN	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
SPI3EN	SPI2EN	Res.	Res.	WWDGEN	RTCAPBEN	LCDEN	Res.	Res.	Res.	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
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 para que funcione como "Alternate function", para que pueda usarse como salida de PWM.

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_AFR

+ Offset:0x20

GPIOx alternate function selection, se escoge la función dependiendo de TIMER usado. Este registro se divide en dos: Low y High, lo cuales dependen de los pines que se vayan a seleccionar (en este caso es el pin 5 por lo que se usa Low).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
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
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 5. Distribución bits AFR, RM0351 Reference Manual, Pag. 309

Bits 31:0 **AFSEL[7:0][3:0]:** Alternate function selection for port x I/O pin y (y = 7 to 0)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Figure 6. Distribución bits AFR, RM0351 Reference Manual, Pag. 310

Para saber cual es la función correspondiente para el TIMER 2 en el pin 5 se acude a la tabla de funciones alternativas que se encuentra en la datasheet de la tarjeta (en las referencias se encuentra el link para descargarla).

Table 17. Alternate function AF0 to AF7[1]

Port		AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7
		SYS_AF	TIM1/TIM2/ TIM5/TIM8/ LPTIM1	TIM1/TIM2/ TIM3/TIM4/ TIM5	TIM8	I2C1/I2C2/I2C3	SPI1/SPI2	SPI3/DFSDM	USART1/ USART2/ USART3
Port A	PA0	-	TIM2_CH1	TIM5_CH1	TIM8_ETR	-	-	-	USART2_CTS
	PA1	-	TIM2_CH2	TIM5_CH2	-	-	-	-	USART2_RTS_ DE
	PA2	-	TIM2_CH3	TIM5_CH3	-	-	-	-	USART2_TX
	PA3	-	TIM2_CH4	TIM5_CH4	-	-	-	-	USART2_RX
	PA4	-	-	-	-	-	SPI1_NSS	SPI3_NSS	USART2_CK
	PA5	-	TIM2_CH1	TIM2_ETR	TIM8_CH1N	-	SPI1_SCK	-	-
	PA6	-	TIM1_BKIN	TIM3_CH1	TIM8_BKIN	-	SPI1_MISO	-	USART3_CTS
	PA7	-	TIM1_CH1N	TIM3_CH2	TIM8_CH1N	-	SPI1_MOSI	-	-
	PA8	MCO	TIM1_CH1	-	-	-	-	-	USART1_CK

Figure 7. Funciones alternativas, Datasheet STM32L476xx, Pag. 92

1.3 Configuración del TIMER

1.3.1 TIMx_PSC

+ Offset:0x28

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

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 8. 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 .

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 9. Distribución bits TIMx→ARR, RM0351 Reference Manual, Pag. 1077

1.3.3 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 10. Distribución bits TIMx→CR1, RM0351 Reference Manual, Pag. 969

1.3.4 TIMx_CCR1

+ Offset:0x34

TIMx capture/compare register 1, si el canal esta configurado como salida el valor en este registro se cargara en el registro de captura/comparación (reemplazando el valor precargado). Este valor es el que se comparara con el valor en el contador TIMx_CNT y determinará el ciclo útil.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 11. Distribución bits TIMx→CCR1, RM0351 Reference Manual, Pag. 991

1.3.5 TIMx_CCMR1

+ Offset:0x18

TIMx capture/compare mode register 1 [alternate], este registro se usará para configurar el modo de funcionamiento de la comparación de la salida (PWM) con los bits [6:4] y habilitar la precarga del registro TIMx_CCMR1 con el bit 3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 12. Distribución bits TIMx→CCMR1, RM0351 Reference Manual, Pag. 982

1.3.6 TIMx_CCER1

+ Offset:0x20

TIM1/TIM8 capture/compare enable register, se usa para habilitar un canal especifico de captura y comparación. Para el ejercicio se usa el canal CC1 como salida, el cual se habilita con el bit 0 de este registro.

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.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 13. Distribución bits TIMx→CCER1, RM0351 Reference Manual, Pag. 987

2 PWM

Controlar la energía que se le entrega a una carga sirve para controlar su funcionamiento según sea necesario. Para ilustrar mejor la utilidad de un PWM tomemos como ejemplo un led cualquiera, los leds requieren de un cierto nivel de voltaje para alcanzar la corriente suficiente y funcionar, por lo tanto hay rangos de voltajes para los cuales el led no enciende lo que limita el control sobre el brillo .

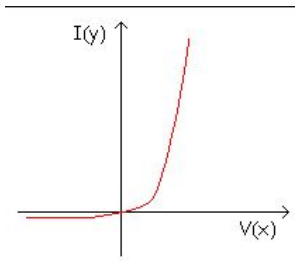

Figure 14. Curva característica diodo

La solución para este problema es alimentar el led con una señal de PWM cuyo voltaje máximo sea el que haga brillar todo lo posible en DC. Entonces para ajustar el brillo del led solo se necesita aumentar o disminuir el ciclo útil, esto es el porcentaje del período de la señal en que hay un voltaje distinto de cero.

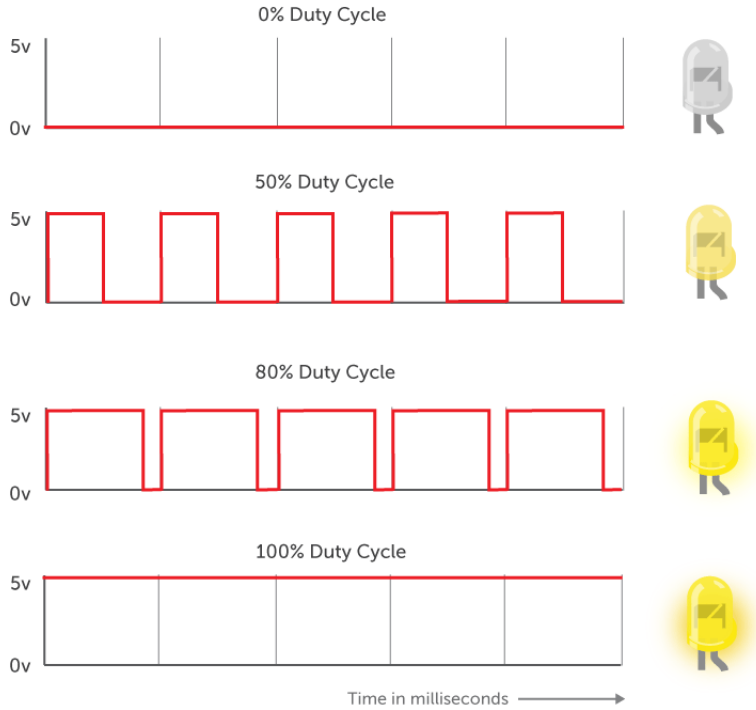

Figure 15. Brillo led para distintos valores de ciclo útil

Como se ve en la imagen anterior al aumentar el ciclo útil el led ilumina con

mayor intensidad, cuando el ciclo útil es del 100% el led llega a su máximo brillo y en 0% permanece apagado. Entonces ¿Por qué sucede esto? Los sistemas se resisten a los cambios repentinos, de forma que cuando una señal tiene una frecuencia muy alta el sistema no puede reaccionar inmediatamente a esta y lo hace de manera gradual.
Si la frecuencia del PWM fuera muy baja simplemente se vería el led parpadeando, cuando la frecuencia aumente el parpadeo se hará tan rápido que sera imperceptible y si aumenta aun más el led no alcanzara a apagarse antes de volver a recibir un voltaje que lo haga encenderse y así sucesivamente, lo que provoca que no brille al máximo ni este apagado, si no que tome un valor intermedio que dependerá de cuanto tiempo se le entrega energía y cuanto no.

3 Código PWM

El ejercicio consiste en programar el Timer para generar una señal de PWM con un ciclo útil determinado. A continuación se explicará como interactúan algunos con la señal de PWM:

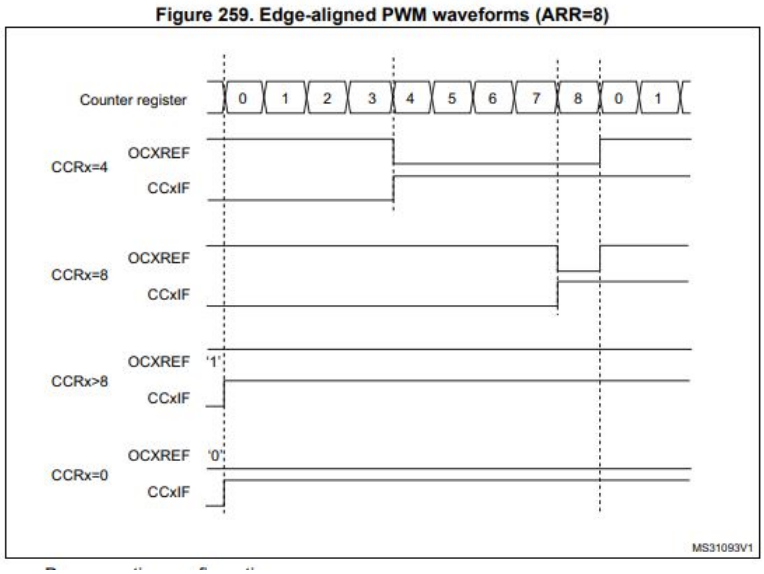

Figure 16. Señal de PWM generada con la STM32L476

Para este ejemplo el valor cargado en el registro ARR es 8, el periodo de la señal PWM dura hasta que el contador del Timer alcance el valor en ARR, cuando esto pasa el contador se reinicia y la señal PWM se repite. Para simplificar podemos decir que el PWM tiene un periodo de 9 (valor en el registro ARR más uno, pues el contador inicia desde cero) y sabiendo esto podemos escoger el valor del ciclo útil que se cargará en el registro CCR1.

$$Ciclo\ útil(\%) = \frac{CCR1}{ARR + 1} * 100$$

Entonces en la imagen se ilustra como seria la señal de PWM con periodo 9 para distintos valores de ciclo útil, se resalta el hecho que para un ciclo útil igual a cero la señal permanece en 0 todo su periodo mientras que para cualquier valor de ciclo útil que sea mayor o igual al periodo, se tendrá una salida de voltaje constante.

3.1 Inicialización

Se inicializan los registros que se van a usar con su direcciones y offsets.

```
//        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
//   AFRL  register offset is 0x20
.equ     GPIOA_MODER,   0x48000000 // GPIOA port mode register
.equ     GPIOA_AFRL,    0x48000020 // GPIOA alternate data register

//       TIM2  base address is 0x40000000
//   PSC   register offset is 0x28
//   ARR   register offset is 0x2C
//   CR1   register offset is 0x00
//   CCMR1 register offset is 0x18
//   CCR1  register offset is 0x34
//   CCER  register offset is 0x20
.equ     TIM2_PSC,   0x40000028 // TIM2 prescaler register
.equ     TIM2_ARR,   0x4000002C // TIM2 auto-reload register
.equ     TIM2_CR1,   0x40000000 // TIM2 control 1 register
.equ     TIM2_CCMR1, 0x40000018 // TIM2 capture/compare mode register
.equ     TIM2_CCR1,  0x40000034 // TIM2 capture/compare register
.equ     TIM2_CCER,  0x40000020 // TIM2 capture/compare enable register
```

Y se activan los relojes para los periféricos que se utilizaran: GPIOA y TIM2.

```
// Enable GPIOA and GPIOC Peripheral Clock (bit 0 and 2 in AHB2ENR register)
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

// Enable TIMER2 Controller Clock (bit 0 in APB1ENR1 register)
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
```

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) con una función alternativa.

```
// Make GPIOA Pin5 as output pin (bits 1:0 in MODER register)
ldr r6, = GPIOA_MODER       // Load GPIOA MODER register address to r6
ldr r5, [r6]                // Read its content to r5
and r5, 0xFFFFFBFF          // Write 10 to bits 11, 10 for P5
str r5, [r6]                // Store result in GPIOA MODER register
```

Entonces se escoge la función alternativa que se usará en el pin 5, según la tabla mostrada en la sección del registro AFRL.

```
// Make GPIOA Pin5 as alternate pin (bits 1:0 in AFRL register)
ldr r6, = GPIOA_AFRL        // Load GPIOA AFRL register address to r6
ldr r5, [r6]                // Read its content to r5
orr r5, 0x00100000          // Write 0001 to bits 23, 22, 21, 20 for P5
str r5, [r6]                // Store result in GPIOA AFRL register
```

3.3 Configuración TIMER 2

Se establece el valor del prescalador y el valor limite del contador, definiendo la frecuencia del PWM.

```
// Make TIMER2 prescaler register (bits 15:0 in PSC register)
ldr r6, = TIM2_PSC          // Load TIM2 PSC register address to r6
ldr r5, = PRESCALER         // load 4 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
```

```
// Make TIMER2 auto-reload register (bits 31:0 in ARR register)
ldr r6, = TIM2_ARR          // Load TIM2 ARR register address to r6
ldr r5, [r6]                // Read its content to r5
and r5, 0x50                    // Write 80 ARR = IN_interrpt/T_timer
str r5, [r6]                // Store result in TIM2 ARR register
```

Después se escoge el ciclo útil, se configura para que el canal 1 trabaje como PWM y se habilitan el canal.

```
// Make TIMER2 capture/compare register (bits 31:0 in CCR1 register)
ldr r6, = TIM2_CCR1         // Load TIM2 CCR1 register address to r6
ldr r5, [r6]                // Read its content to r5
orr r5, 0x8                     // Set duty cycle
str r5, [r6]                // Store result in TIM2 CCR1 register

// Make TIMER2 capture/compare mode register (bits 31:0 in CCMR1 register)
ldr r6, = TIM2_CCMR1        // Load TIM2 CCMR1 register address to r6
ldr r5, [r6]                // Read its content to r5
orr r5, 0x68                    // Set PWM mode
str r5, [r6]                // Store result in TIM2 CCMR1 register

// Make TIMER2 capture/compare enable register (bits 15:0 in CCER register)
ldr r6, = TIM2_CCER         // Load TIM2 CCER register address to r6
ldr r5, [r6]                // Read its content to r5
orr r5, 0x1                     // enable output
str r5, [r6]                // Store result in TIM2 CCER register
```

Para terminar se habilita el contador para comenzar a generar la señal:

```
// Make TIMER2 enable register (bits 15:0 in CR1 register)
ldr r6, = TIM2_CR1          // Load TIM2 CR1 register address to r6
ldr r5, [r6]                // Read its content to r5
orr r5, 0x1                     // Set bit 0 to enable timer
str r5, [r6]                // Store result in TIM2 CR1 register
```

4 Ejercicios prácticos

1. Disminuya la frecuencia del PWM hasta unos pocos Hz y describa que sucede.

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.
+ STM32l476rg datasheet.