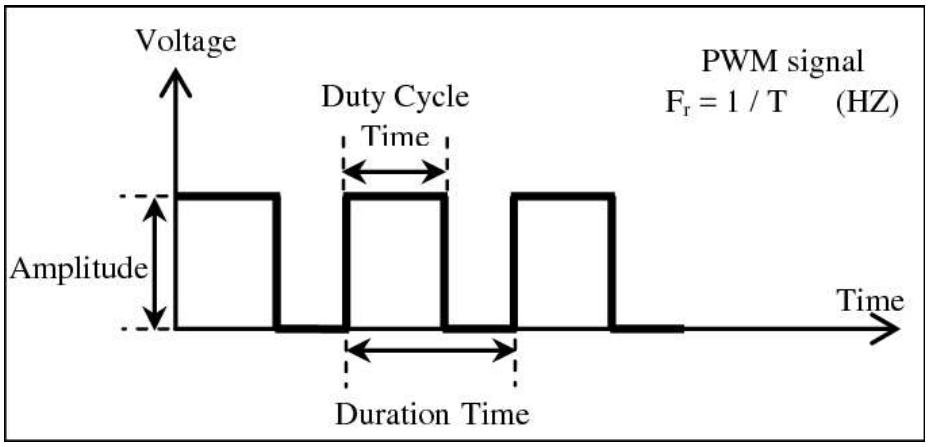
Working with STM32 and Timers part 2: PWM mode, fading 2 LEDs



In the previous guide (here), we discussed how to use the timer to generate accurate delay. In this guide, we shall look at another application for timer which is to generate PWM signal on two channels and fading two leds.

In this guide, we will cover the following:

- What is PWM
- Configure the timer and GPIO to generate PWM signal
- Connection
- Code
- Demo

1. What is PWM:

PWM is a technique used to emulate “analog signal” by rapidly turning on-off the pin. This allow the mcu to vary the power delivered to the load such as motor (will be covered later). The PWM has three main characteristics;

- Frequency:

which describe the duration time of the entire signal

- Duty cycle

The term duty cycle describes the proportion of ‘on’ time to the regular interval or ‘period’ of time; a low duty cycle corresponds to low power, because the power is off for most of the time. Duty cycle is expressed in percent, 100% being fully on.

- Amplitude

Which is the voltage level of the PWM (3.3v for STM32f4).

For more details, please check this wikipedia article (here)

2. Configure the timer and GPIO to generate PWM Signal:

First we need to locate which pins connected to TIMER2_CH1 and TIMER2_CH2.

According to the datasheet of STM32F411, PA0 and PA1 are connected to TIM2_CH1 and TIM2_CH2. Hence, we need to enable clock access to GPIOA as following: (for details refer to this topic)

C

```
RCC->AHB1ENR|=RCC_AHB1ENR_GPIOAEN;
```

The we need to configure PA0 and PA1 as alternate function as following:

C

```
GPIOA->MODER|=(1<<1)|(1<<3);
```

After that, wee need to select AF1 for PA0 and PA1 since it is responsible to TIM2 as following

C

```
GPIOA->AFR[0]|=(1<<0)|(1<<4);
```

After that, we enable clock access to TIMER2 as following

C

```
RCC->APB1ENR|=RCC_APB1ENR_TIM2EN; //enable clock access to tim2
```

Hence we determine the prescaller and the ARR

C

```
TIM2->PSC=0; //set prescaller to 0 (no divider)
TIM2->ARR=255; //set the maximum count value
TIM2->CNT=0; //seset the current count
```

with prescaler of 0 and ARR of 255 we will get 62.5KHz (16000000/255)

Then we need to configure capture/compare mode register (CCMR)

The capture/compare mode register has two register (CCMR1 and CCMR2)

- CCMR1 is for channel 1 and channel 2 of the timer
- CCMR2 is for channel 3 and channel 4 of the timer

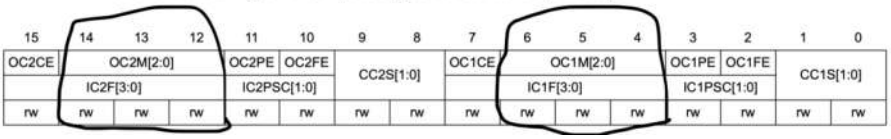
Hence we need CCMR1

13.4.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. Take care that the same bit can have a different meaning for the input stage and for the output stage.



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]	
	IC2F[3:0]			IC2PSC[1:0]					IC1F[3:0]			IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

For OC1M and OC2M we need to set them as PWM

Bits 6:4 **OC1M**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.
000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).
001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.
100: Force inactive level - OC1REF is forced low.
101: Force active level - OC1REF is forced high.
110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0) as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF=1).
111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

Note: In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

C

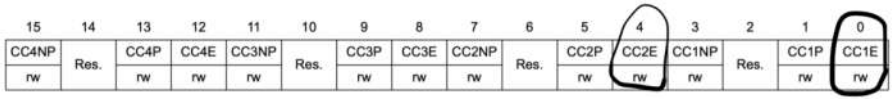
```
TIM2->CCMR1=(1<<5)|(1<<6)|(1<<13)|(1<<14); //configure the pins as PW
```

From capture/compare enable register we need to enable channel1 and channel2

13.4.9 TIMx capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000



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

C

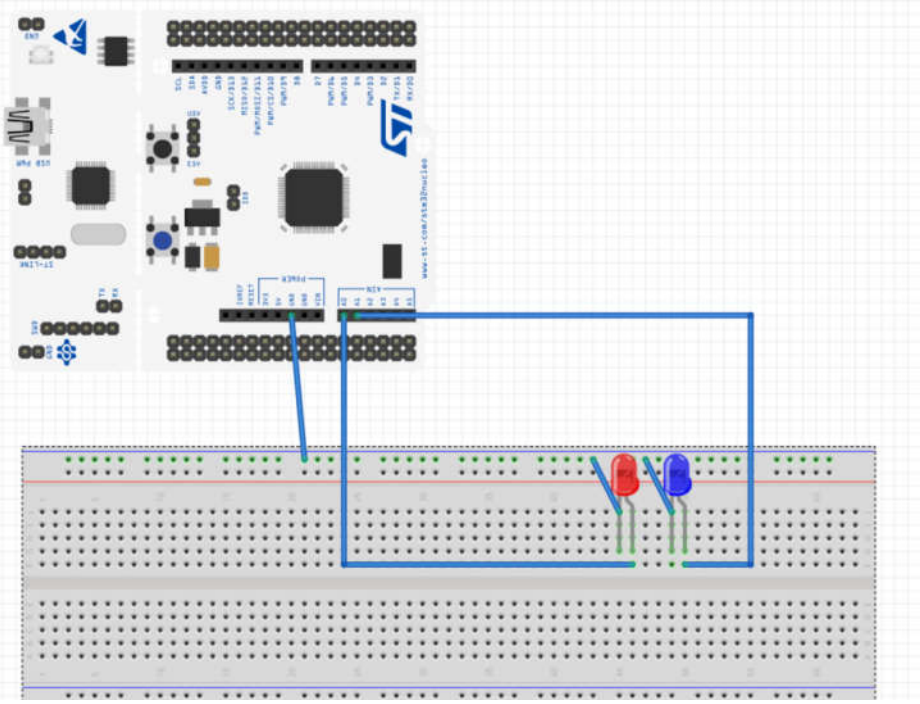
```
TIM2->CCER|=0x11; //enbale channel1 and channel2
```

Finally, we enable the timer from control register 1 (CR1)

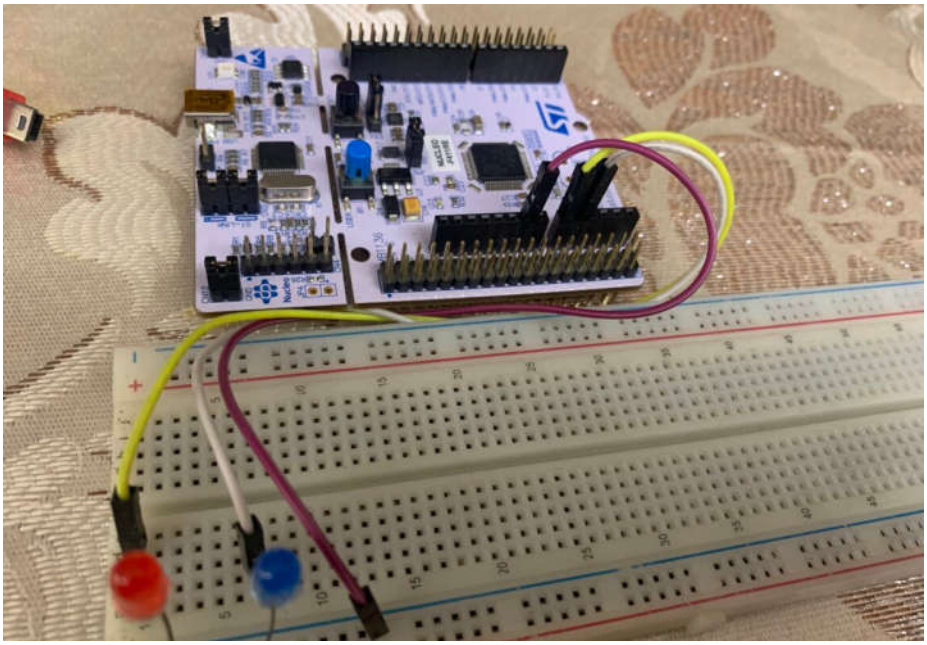
C

```
TIM2->CR1=1; //enable timer
```

3. Connection



Connection



actual connection

4. Code:

C

```c
#include "stm32f4xx.h"                  // Device header
void GPIO_Init(void);
void Timer2_init(void);
void delay(int ms );
#define rate 255
int main(void)
{

GPIO_Init();
Timer2_init();
```

```
while(1)
{
for (int i=0;i<rate;i++){
TIM2->CCR1=i;
TIM2->CCR2=rate-i;
delay(5);
}
for (int i=255;i>0;i--){
TIM2->CCR1=i;
TIM2->CCR2=rate-i;
delay(5);
}

}
}

void GPIO_Init(void)
{
RCC->AHB1ENR|=RCC_AHB1ENR_GPIOAEN;
GPIOA->AFR[0]|=(1<<0)|(1<<4);
GPIOA->MODER|=(1<<1)|(1<<3);

}
void  Timer2_init(void){
RCC->APB1ENR|=RCC_APB1ENR_TIM2EN; //enable clock access tto tim2
TIM2->PSC=0; //set prescaller to 0 (no divider)
TIM2->ARR=255; //set the maximum count value
TIM2->CNT=0; //seset the current count
TIM2->CCMR1=(1<<5)|(1<<6)|(1<<13)|(1<<14); //configure the pins as PW
TIM2->CCER|=0x11; //enbale channel1 and channel2
TIM2->CR1=1; //enable timer
}


void delay(int ms)

{

int i;

for(; ms>0 ;ms--)

{

for(i =0; i<3195;i++);

}

}
```

5. Demo:

0:00 / 0:06