MCSL Lab07

STM32 Clock and Timer

0410001
Hong-Shuo Chen

1. Experiment Purpose

- Understand how to use and modify clock source of STM32
- Understand the principle of using timer of STM32
- Understand the principle of using PWM of STM32

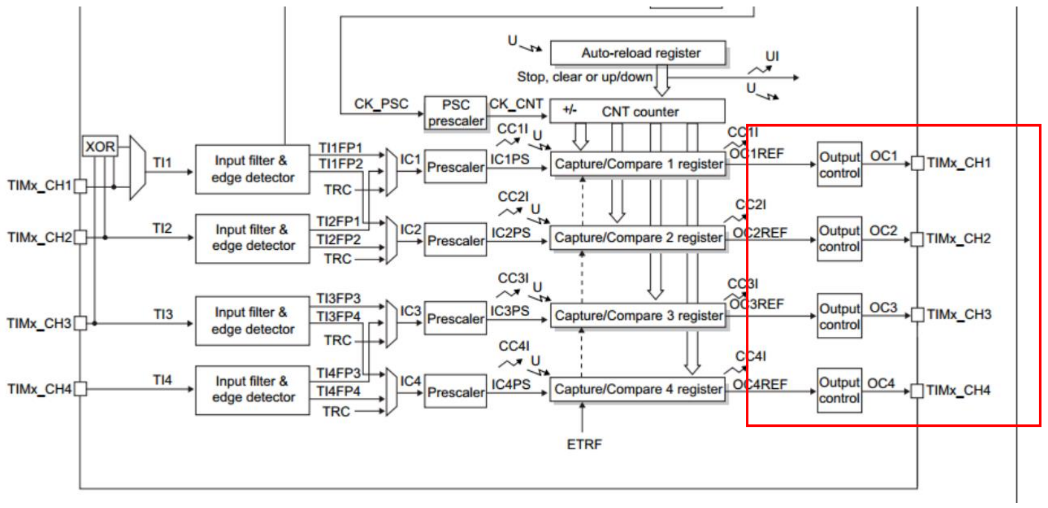
2. Background theory

2.1. Timer and Counter

Please refer to the lecture of 009-MCSL-CounterTimer

2.2. Timer PWM output mode

In the system of STM32, when we need to use Timer to generate the output, we are primarily through setting the capture/compare mode register(TIMx_CCMR1) and TIMx_CCRx registers and use TIMx_CCER to activate it.



Generally, PWM is divided into two modes, which respectively are mode1 and mode2, and the correspondent output on the counter which is in counting up mode is:

- PWM mode1: Channel is active as long as TIMx_CNT<TIMx_CCR1 else inactive.
- PWM mode2: Channel is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

In the other hand, it can be divided into Combined PWM mode and Asymmetric PWM mode depending on the usage of the timer.

Figure 279. Edge-aligned PWM waveforms (ARR=8)

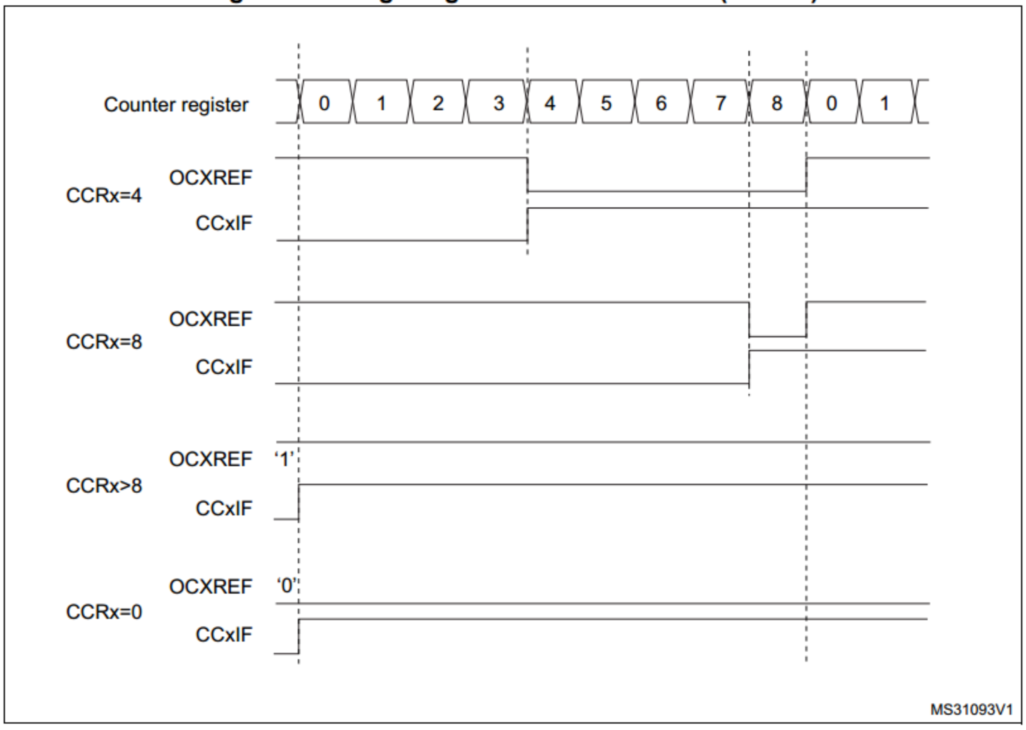
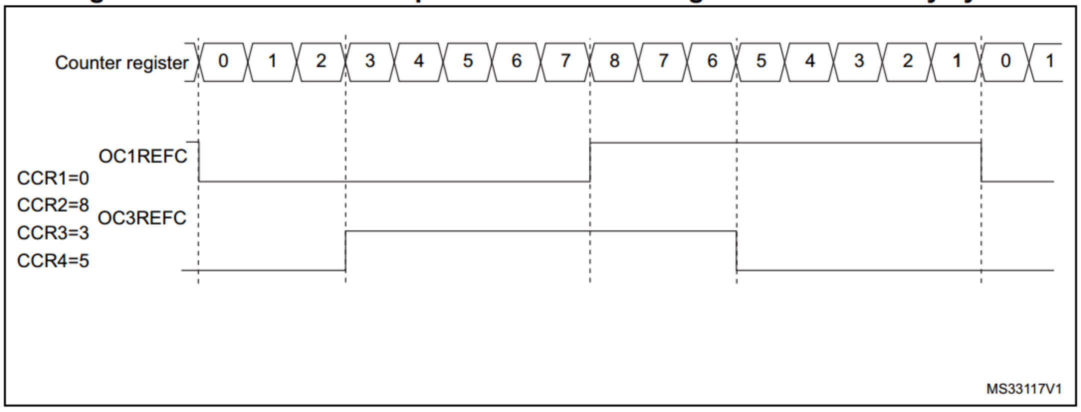


Figure 281. Generation of 2 phase-shifted PWM signals with 50% duty cycle



3. Experiment Procedure

3.1. Lab 7.1: Modify system initial clock(20%)

```
main.c
void GPIO_init();
void 4MHz_delay_1s();
void SystemClock_Config(){
   //TODO: Change the SYSCLK source and set the corresponding
Prescaler value.
}

int main(){
  SystemClock_Config();
  GPIO_init();
  while(1){
    if (user_press_button())
    {
      //TODO: Update system clock rate
    }
    GPIOA->BSRR = (1<<5);
    4MHz_delay_1s ();
    GPIOA->BRR = (1<<5);
    4MHz_delay_1s ();
  }
}
```

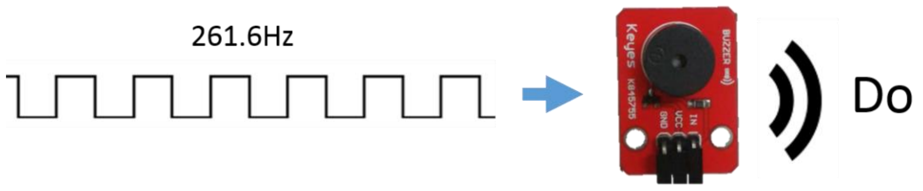
To modify the PLL configuration, proceed as follows:

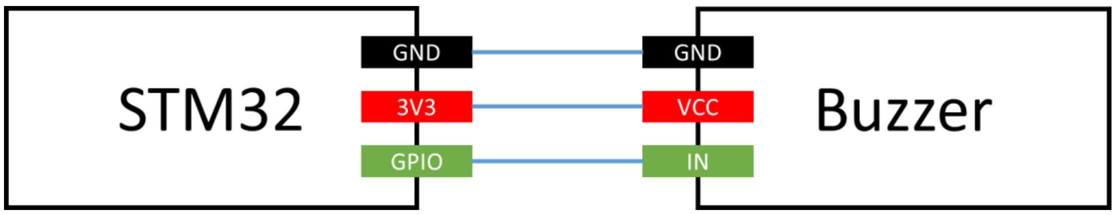
1. Disable the PLL by setting PLLON to 0 in *Clock control register (RCC_CR)*.
2. Wait until PLLRDY is cleared. The PLL is now fully stopped.
3. Change the desired parameter.
4. Enable the PLL again by setting PLLON to 1.
5. Enable the desired PLL outputs by configuring PLLPEN, PLLQEN, PLLREN in *PLL configuration register (RCC_PLLCFGR)*.

3.2. Lab7.2: Timer(30%)

```
main.c
#include "stm32l476xx.h"
#define TIME_SEC 12.70
extern void GPIO_init();
extern void max7219_init();
extern void Display();
void Timer_init( TIM_TypeDef *timer)
{
    //TODO: Initialize timer
}
void Timer_start(TIM_TypeDef *timer){
    //TODO: start timer and show the time on the 7-SEG LED.
}
int main()
{
   GPIO_init();
   max7219_init();
   Timer_init();
   Timer_start();
   while(1)
   {
     //TODO: Polling the timer count and do lab requirements
   }
}
```

3.3. Lab7.3 Music keypad(35%)

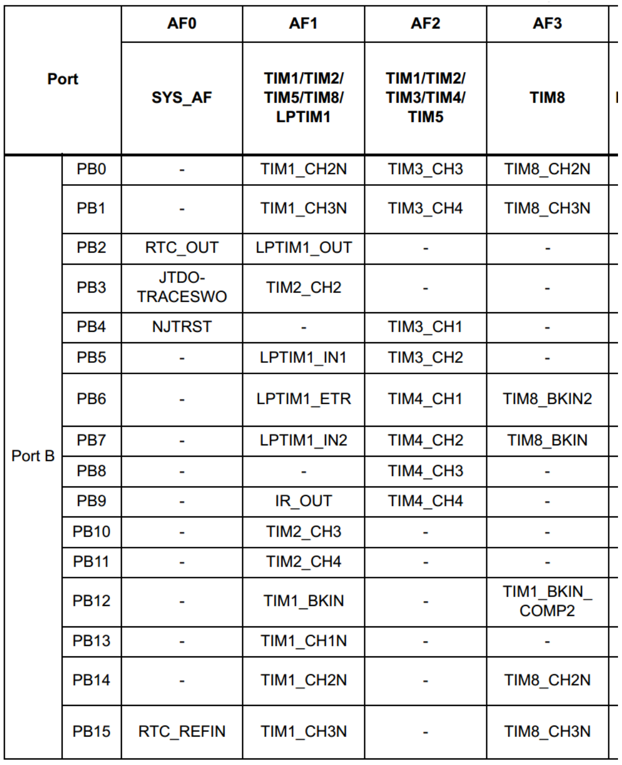




Each value of corresponding button is given below.

	X0	X1	X2	X3
Y0	Do	Re	Mi	
Y1	Fa	So	La	
Y2	Si	HDo		
Y3				

音名	Do	Re	Mi	Fa	So	La	Si	HDo
頻率(Hz)	261.6	293.7	329.6	349.2	392.0	440.0	493.9	523.3



Port		AF0	AF1	AF2	AF3
		SYS_AF	TIM1/TIM2/ TIM5/TIM8/ LPTIM1	TIM1/TIM2/ TIM3/TIM4/ TIM5	TIM8
Port B	PB0	-	TIM1_CH2N	TIM3_CH3	TIM8_CH2N
	PB1	-	TIM1_CH3N	TIM3_CH4	TIM8_CH3N
	PB2	RTC_OUT	LPTIM1_OUT	-	-
	PB3	JTDO-TRACESWO	TIM2_CH2	-	-
	PB4	NJTRST	-	TIM3_CH1	-
	PB5	-	LPTIM1_IN1	TIM3_CH2	-
	PB6	-	LPTIM1_ETR	TIM4_CH1	TIM8_BKIN2
	PB7	-	LPTIM1_IN2	TIM4_CH2	TIM8_BKIN
	PB8	-	-	TIM4_CH3	-
	PB9	-	IR_OUT	TIM4_CH4	-
	PB10	-	TIM2_CH3	-	-
	PB11	-	TIM2_CH4	-	-
	PB12	-	TIM1_BKIN	-	TIM1_BKIN_ COMP2
	PB13	-	TIM1_CH1N	-	-
	PB14	-	TIM1_CH2N	-	TIM8_CH2N
	PB15	RTC_REFIN	TIM1_CH3N	-	TIM8_CH3N

```
extern void GPIO_init();
void GPIO_init_AF(){
//TODO: Initial GPIO pin as alternate function for buzzer. You can
choose to use C or assembly to finish this function.
}
void Timer_init(){
    //TODO: Initialize timer
}
void PWM_channel_init(){
    //TODO: Initialize timer PWM channel
}
int main(){
   GPIO_init();
   GPIO_init_AF();
   Timer_init();
   PWM_channel_init();
    //TODO: Scan the keypad and use PWM to send the corresponding
frequency square wave to buzzer.
     }
```

4. Results and Discussion

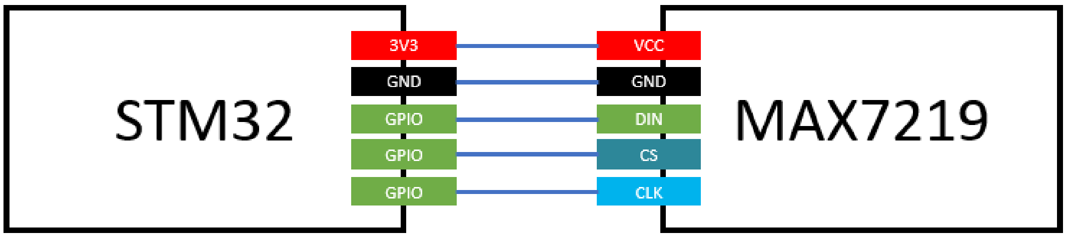
4.1. Lab 7.1: Max7219 displayer (10%)

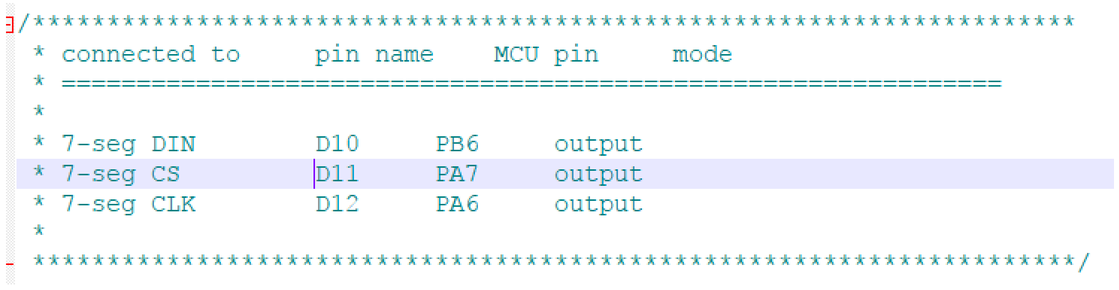
Hardware Design:
set PA5 as output mode(LED)
set PC13 as input mode(button)

Code Explanation:
In this lab, I use the PLL clock as the SYSCL, and use HSI16 as PLL's source clock. In order to create many different HCLK, I adjust the PLLN, PLLM, PLLR, and HPRE in the RCC_PLLCFGR register and RCC_CFGR register according to the formula:
PLLSRC (HSI16) * PLLN / PLLM / PLLR / HPRE = HCLK
The first thing is to switch the SYSCLK to the original MSI by changing the value of RCC_CFGR register, and then turn the PLL off by configuring the RCC_CR. We can change the value of RCC_PLLCFGR only after turning off the PLL. The value of RCC_PLLCFGR can be calculated according to the table above. After setting up the value of RCC_PLLCFGR, I turn the HSI16 and PLL on, switching the SYSCLK to PLL, so we have configured the SYSCLK.
The final step is to set the HPRE (AHB prescaler) by changing the value of RCC_CFGR, then the HCLK for AHB bus is configured to the desired frequencies.

4.2. Lab7.2: Timer(30%)

Hardware Design:





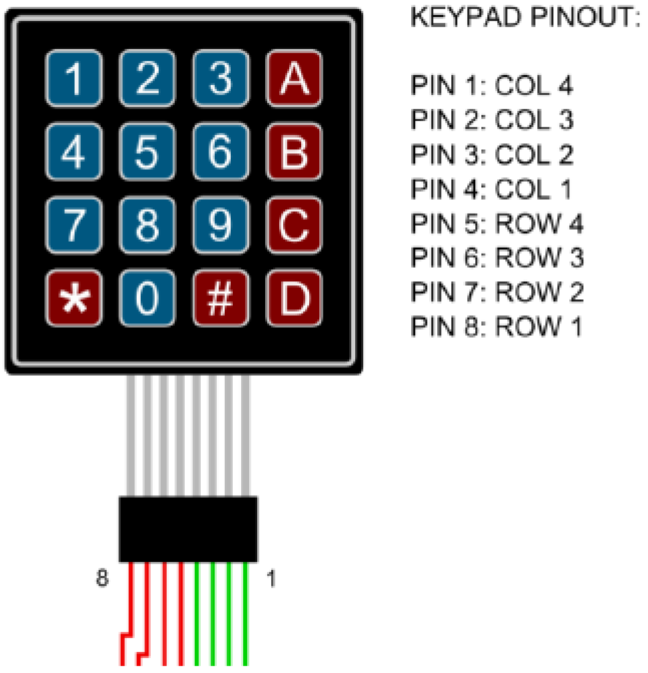
```
/**********************************************************************
 * connected to     pin name    MCU pin     mode
 * ===============================================================
 *
 * 7-seg DIN        D10     PB6     output
 * 7-seg CS         D11     PA7     output
 * 7-seg CLK        D12     PA6     output
 *
 **********************************************************************/
```

Code Explanation:

This exercise is a simple timer implementation by polling the counter value. The clock source is the original MSI (4MHz), and the prescaler of TIM2 and auto-reload register are set to be 39999 and 99, respectively. Hence, the frequency of the timer is going to be 100 Hz, and the counter is added by one for each 0.01 second. Whenever the value of the counter (or timer) changed, the displayed number 'n' is also added by one until the time limit set by TIME_SEC * 100. The display function used in this exercise is modified for showing the decimal point in the third position of the 7-segment display.

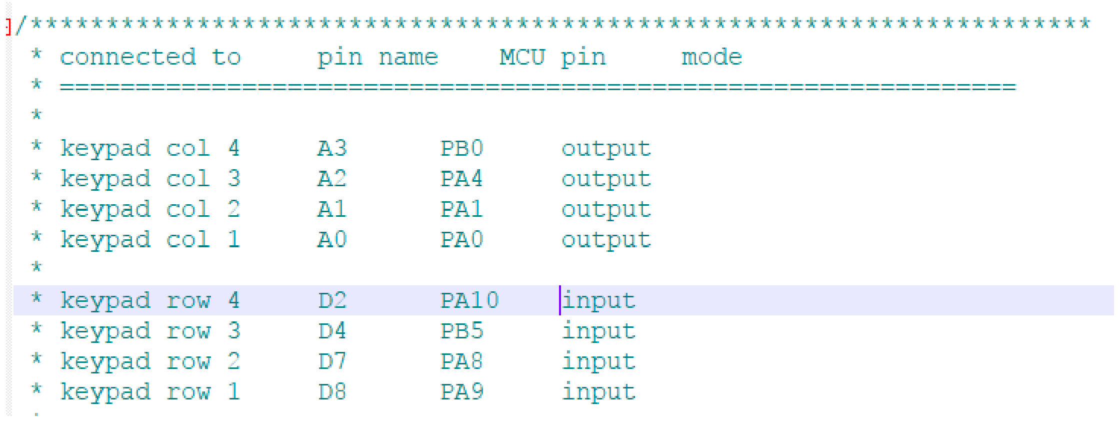
4.3. Lab7.3 Music keypad(35%)

Hardware Design:



KEYPAD PINOUT:

PIN 1: COL 4
PIN 2: COL 3
PIN 3: COL 2
PIN 4: COL 1
PIN 5: ROW 4
PIN 6: ROW 3
PIN 7: ROW 2
PIN 8: ROW 1



```
/*********************************************************************
 * connected to     pin name    MCU pin     mode
 * ============================================================
 *
 * keypad col 4     A3       PB0      output
 * keypad col 3     A2       PA4      output
 * keypad col 2     A1       PA1      output
 * keypad col 1     A0       PA0      output
 *
 * keypad row 4     D2       PA10     input
 * keypad row 3     D4       PB5      input
 * keypad row 2     D7       PA8      input
 * keypad row 1     D8       PA9      input
```

Code Explanation:

This exercise is to generate signals of different frequencies from C4 to C5. The TIM2 is used as a timer as the previous exercise, and the clock source is also the original MSI. The auto-reload register of the timer is fixed to 99. By adjusting the prescaler of the timer, we can generate different frequencies with the following formula:

HCLK / (prescaler + 1) / 100 = desired frequency So, prescaler = HCLK / desired frequency / 100 – 1 The duty cycle is initialized to 50, and can be added or subtracted by 5 by pressing button '#' or button '*'. To be honest, however, I cannot tell the difference between two signals with the same frequencies but different duty cycles.

5. Reviews and Applications

Timer and counter are two very important components in many aspects. This Lab help me totally know how to generate different rate of clock and generate PWM. Timer and counter also use frequently in controlling the motor and PWM. We should learn it carefully.