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

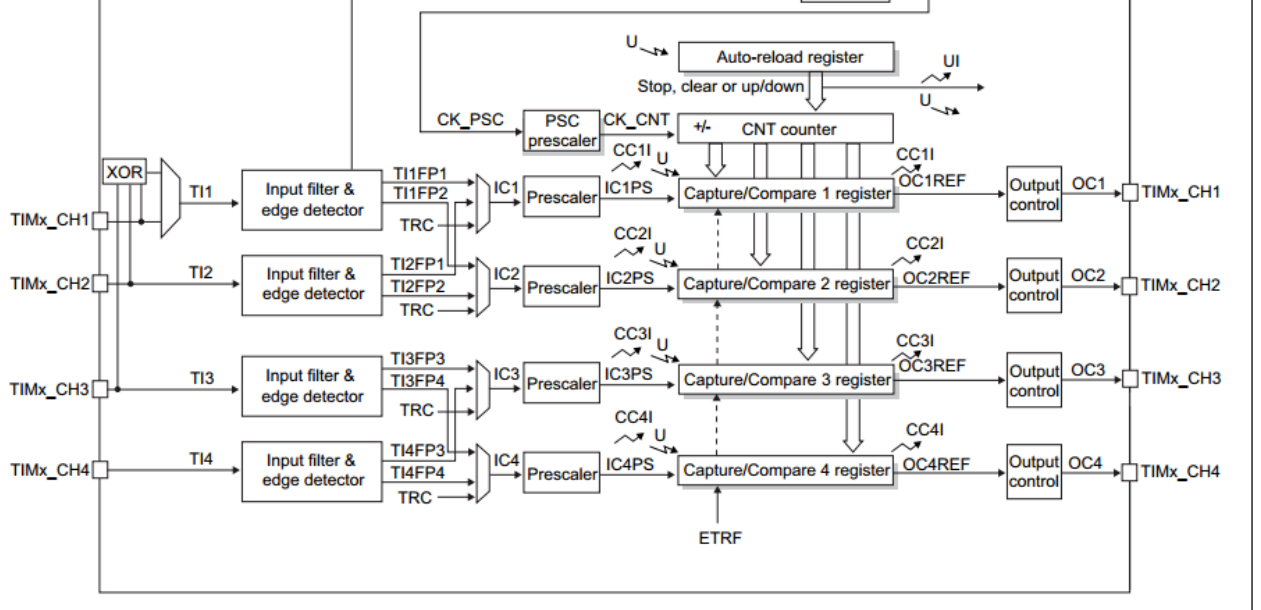
1. Background theory

2.1. Timer and Counter

Please refer to the lecture of 009-MCSL-CounterTimer

2.2. Timer PWM output mode

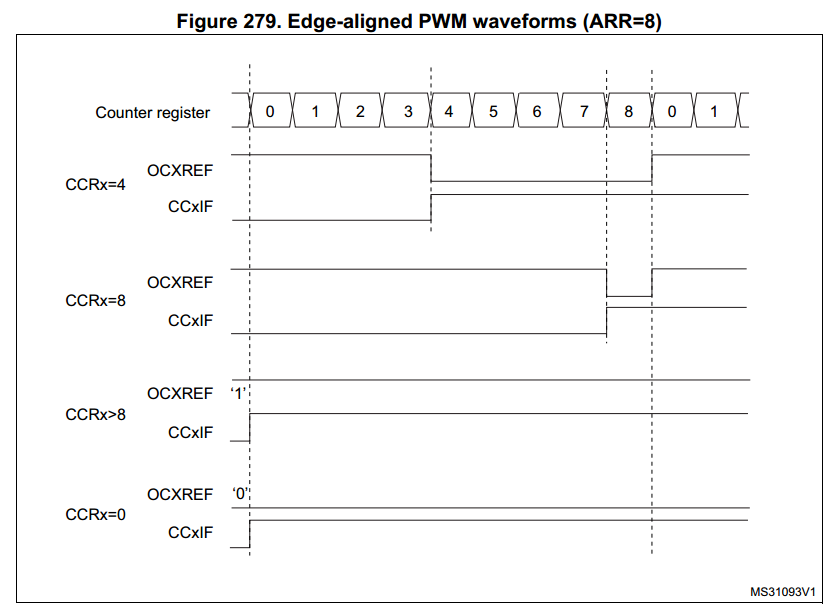
In the system of STM32, when we need to use Timer to generate the output, we are primarily through setting the capture/compare mode register(TIMx\_CCMR1) and TIMx\_CCRx registers and use TIMx\_CCER to activate it.

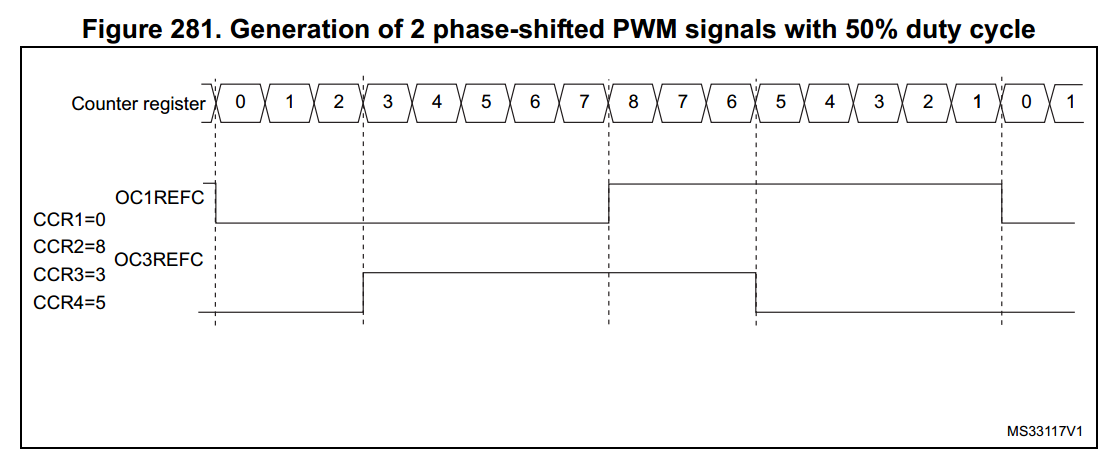


Generally, PWM is divided into two modes, which respectively are mode1 and mode2, and the correspondent output on the counter which is in counting up mode is:

* 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.





1. 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);

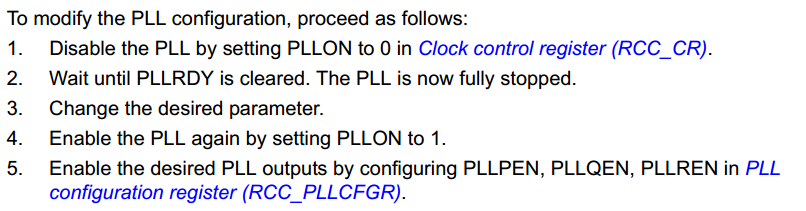
4MHz\_delay\_1s ();

GPIOA->BRR = (1<<5);

4MHz\_delay\_1s ();

}

}



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)

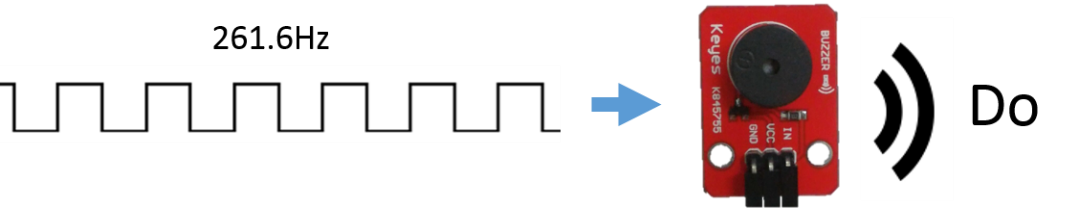
{

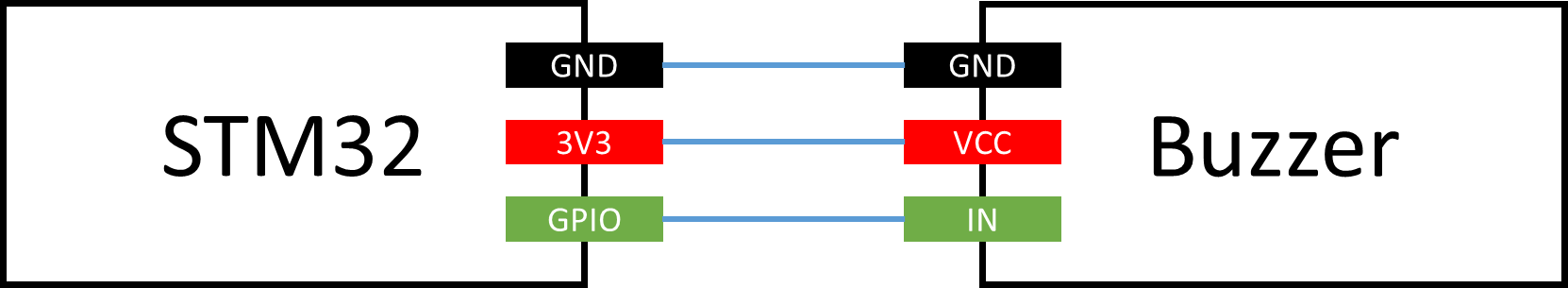
//**TODO**: Polling the timer count and do lab requirements

}

}

3.3. Lab7.3 Music keypad(35%)

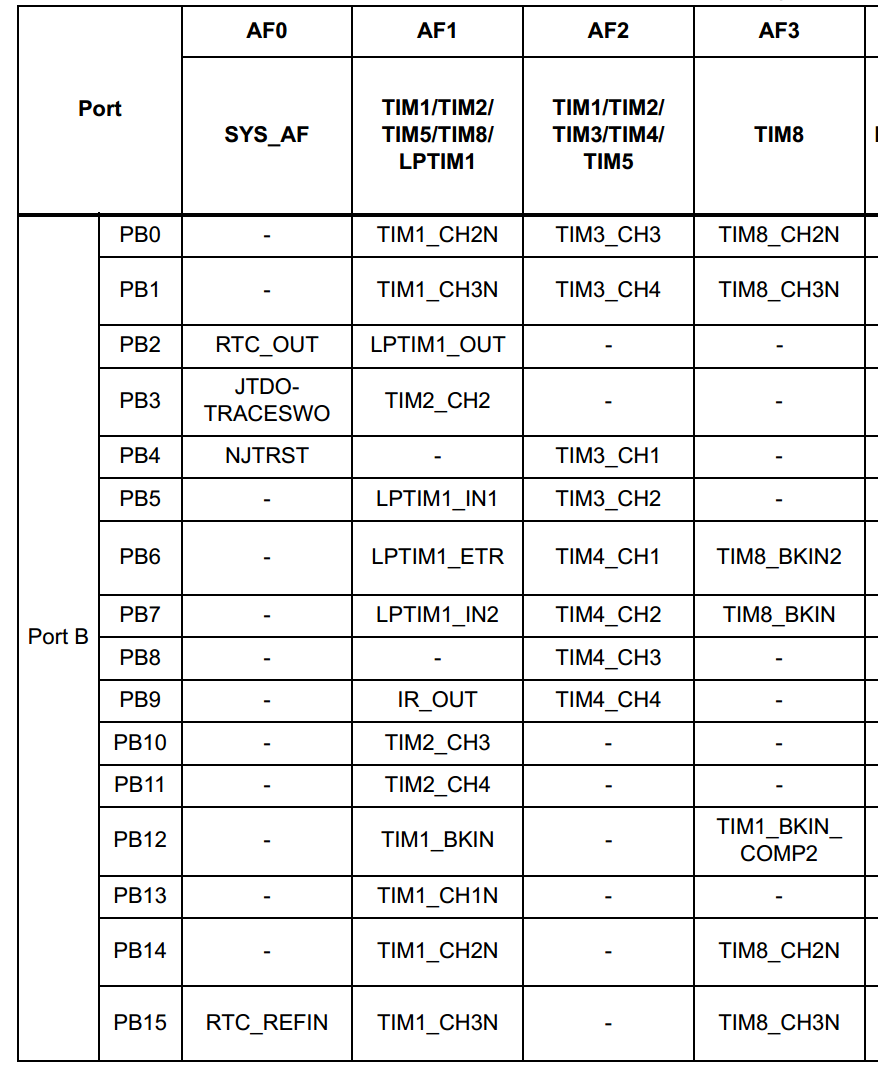




Each value of corresponding button is given below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 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 |



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.

}

1. 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:

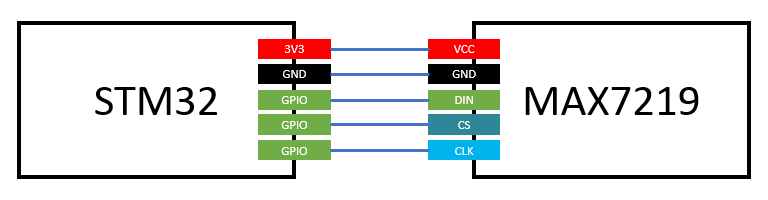
PLLSRC (HSI16) \* PLLN / PLLM / PLLR / HPRE = HCLK

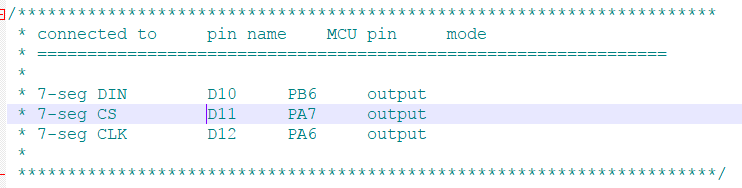
The first thing is to switch the SYSCLK to the original MSI by changing the value of RCC\_CFGR register, and then turn the PLL off by configuring the RCC\_CR. We can change the value of RCC\_PLLCFGR only after turning off the PLL. The value of RCC\_PLLCFGR can be calculated according to the table above. 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:



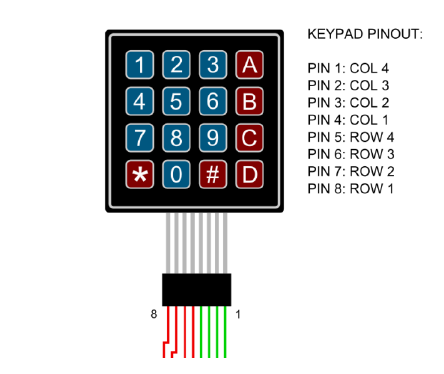


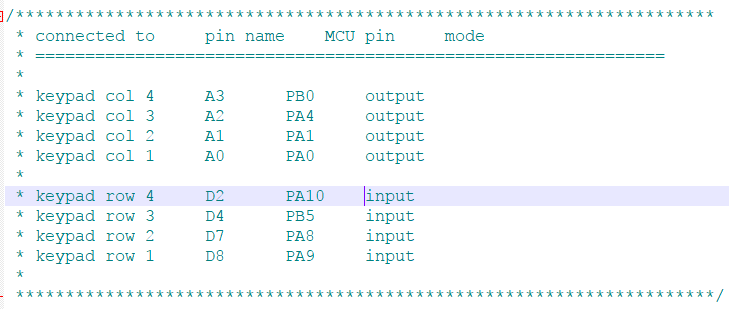
Code Explanation:

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:





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.

1. 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.