MCSL Lab08

STM32 Interrupt and Exception

0410001

Hong-Shuo Chen

1. Experiment Purpose

* Understand how to set STM32 SysTick timer
* Understand the principle and setting of STM32 NVIC and External interrupt

1. Background theory

Please refer to the lecture.

1. Experiment Procedure

3.1. Lab 8.1: SysTick timer interrupt & DS18B20 & One wire protocol

* Implement a SysTick interrupt handler. Show current temperature on 7-segment display when interrupt has been triggered.
* Understand OneWire Protocol and implement onewire.c, onewire.h
* Understand DS18B20 Protocol and implement ds18b20.c, ds18b20.h
* It is fine to show integral part of temperature value only.
* Enable or disable SysTick timer when user push user button.

Notes: Set SysTick clock source to 10MHz. System timer triggers interrupt every 2 seconds. Please use NVIC interrupt mask register to enable or disable Systick timer. Please reference Programming manual section 4.3 for NVIC registers.



void SystemClock\_Config(){

//TODO: Setup system clock and SysTick timer interrupt

}

void SysTick\_Handler(void) {

//TODO: Show temperature on 7-seg display

}

int main(){

SystemClock\_Config();

GPIO\_init();

while(1){

if(user\_press\_button())

{

//TODO: Enable or disable Systick timer

}

}

}

3.2. Lab8.2: Keypad external interrupt(30%)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | X0 | X1 | X2 | X3 |
| Y0 | 1 | 2 | 3 | 10 |
| Y1 | 4 | 5 | 6 | 11 |
| Y2 | 7 | 8 | 9 | 12 |
| Y3 | 15 | 0 | 14 | 13 |

**char** key\_value = 0;

**void EXIT\_Setup**(){

//**TODO**: Setup EXTI interrupt

}

**void SystemClock\_Config**(){

//**TODO**: Setup system clock and SysTick timer interrupt

}

**void SysTick\_Handler**(**void**) {

//**TODO**: Scan the keypad column

}

**void EXTIx\_IRQHandler**(**void**){

//**TODO**: Read the keypad row value

}

**int main**(){

SystemClock\_Config();

GPIO\_init();

EXTI\_Setup();

**while**(1){

display(key\_value,2);

}

}

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:

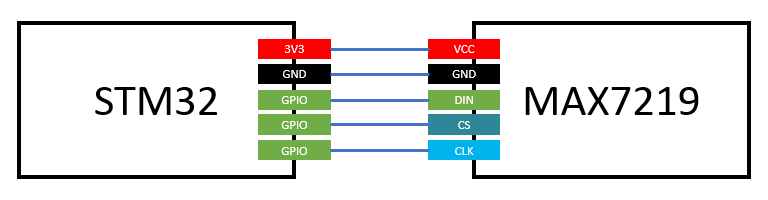
PLLSRC (HSI16) \* PLLN / PLLM / PLLR / HPRE = HCLK

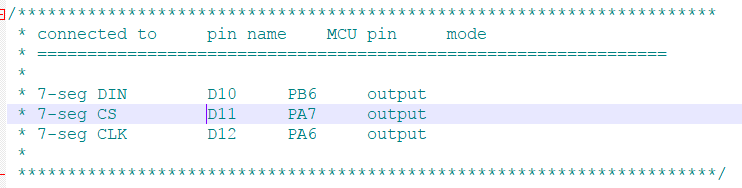
The first thing is to switch the SYSCLK to the original MSI by changing the value of RCC\_CFGR register, and then turn the PLL off by configuring the RCC\_CR. We can change the value of RCC\_PLLCFGR only after turning off the PLL. The value of RCC\_PLLCFGR can be calculated according to the table above.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



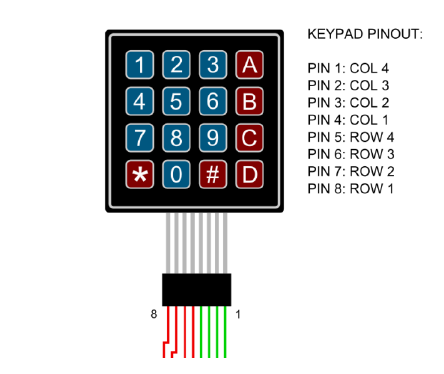


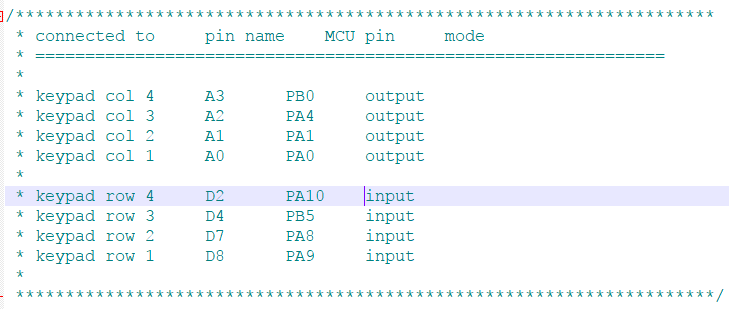
Code Explanation:

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.