# MCSL Lab06

## STM32 Keypad Scanning

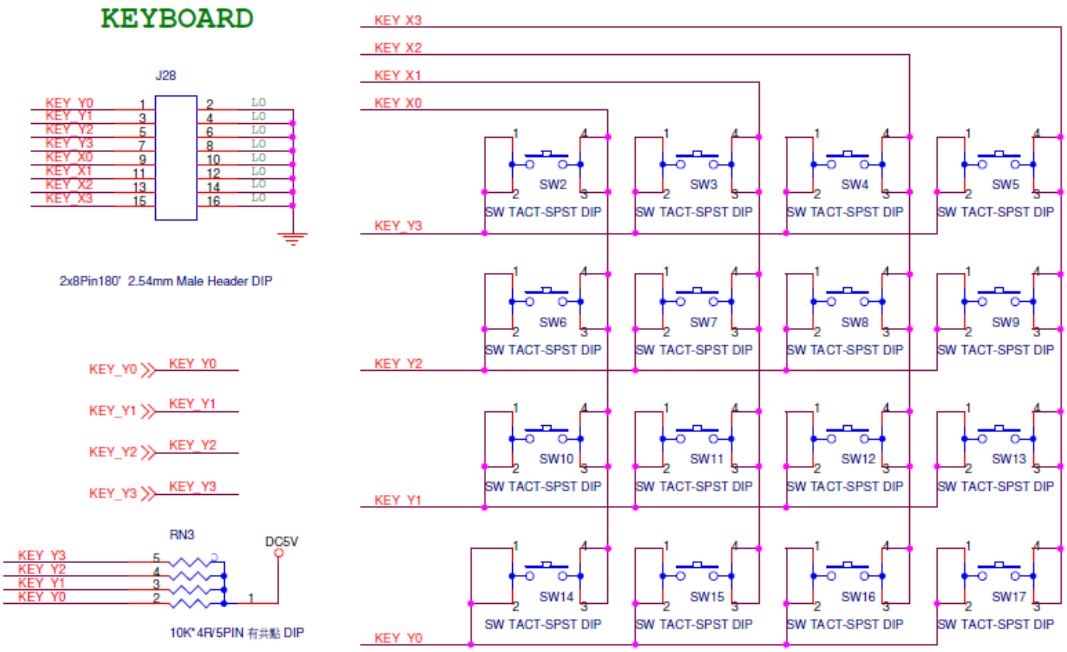
0410001
Hong-Shuo Chen

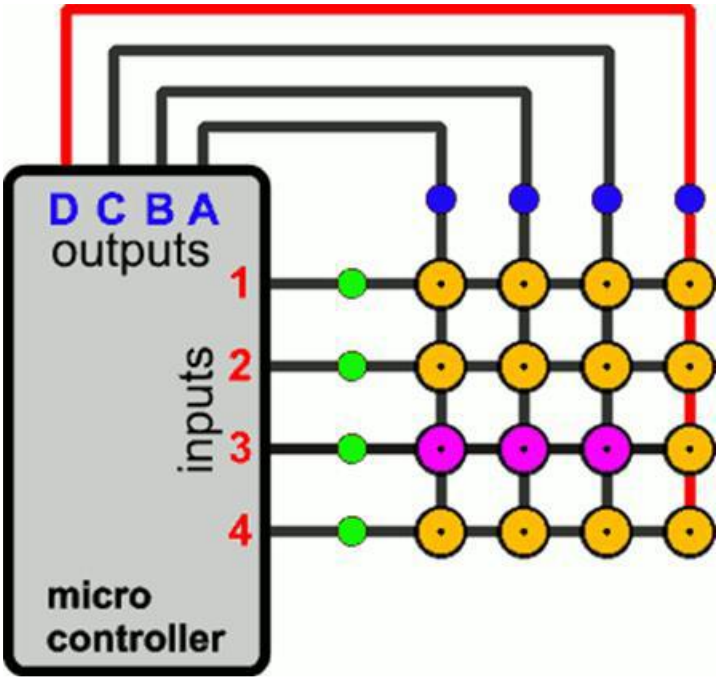
## 1. Experiment Purpose

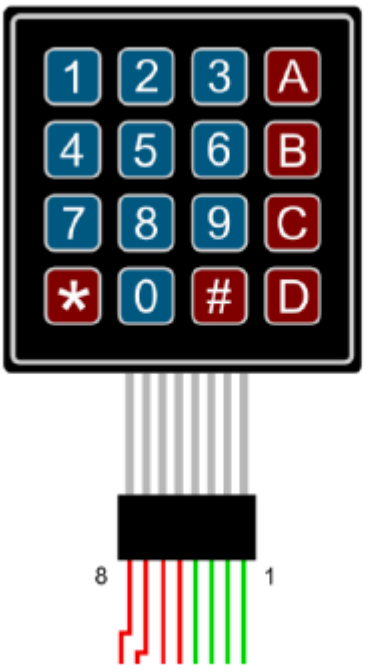
- Understand the principle of using STM32
- Understand how to use C code to control STM32
- Design the program of the 7-Seg LED and keypad

## 2. Background theory

The circuit diagram of keypad is given below. You're supposed to use 4 input pins and 4 output pins. Use output pins to determine which row you're scanning. For example, when output value of KEY X0~3 is 1000 and input value of KEY Y0~3 is 1000, then we can say that SW14 is pressed.






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

# 3. Experiment Procedure

## 3.1. Lab 6.0: Max7219 displayer (10%)

Modify your code in lab5.2 to make it callable by C. Add a C file to complete the code given below, display your student ID on 7-Seg LED.

```
//These functions inside the asm file
extern void GPIO_init();
extern void max7219_init();
extern void max7219_send(unsigned char address, unsigned char data);
/**
* TODO: Show data on 7-seg via max7219_send
* Input:
* data: decimal value
* num_digs: number of digits will show on 7-seg
* Return:
* 0: success
* -1: illegal data range(out of 8 digits range)
*/
int display(int data, int num_digs)
{
}
void main()
{
int student_id = 01234567;
GPIO_init();
max7219_init();
display(student_id, 8);
}
```

### 3.2. Lab6.1: Keypad Scanning

Use 4 input GPIO pins and 4 output GPIO pins to connect with keypad. Show the corresponding number of pressed button on 7-Seg LED.

```
#include “stm32l4xx.h”
//TODO: define your gpio pin
#define X0 GPIOXX
#define X1
#define X2
#define X3
#define Y0
#define Y1
#define Y2
#define Y3
unsigned int x_pin = {X0, X1, X2, X3};
unsigned int y_pin = {Y0, Y1, Y2, Y3};
/* TODO: initial keypad gpio pin, X as output and Y as input
*/
void keypad_init()
{
}
/* TODO: scan keypad value
* return:
* >=0: key pressed value
* -1: no key press
*/
char keypad_scan()
{
}
```

Each value of corresponding button is given below

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

3.3. Lab6.2 Deal with single or multiple buttons (30%)
Show pressed button of keypad on 7-Seg LED. Each value of corresponding button is given below.

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

When multiple buttons are pressed, show the sum of values that buttons pressed representing. If shown value is greater than 99999999, don't modify the number showing on 7-Seg LeD until button C is pressed. Example video link is given above.

3.4. Lab6.3 Deal with single or multiple buttons (BONUS)
Design a calculator first doing multiplication and division then do addition and subtraction. Requirements are given below.

Input value should be in the range of 1~999. If input value is already 3 digits, don't give any responds to button pressed after that.

When operator is pressed, clear the number shown on 7-Seg LED and wait for next number input.

If operator is pressed more than 1 time, answer output should be right though.

After "equal" is pressed, show the answer(negative number and number greater than 999 should be shown).

Example video link is given above.
https://goo.gl/rn8srq

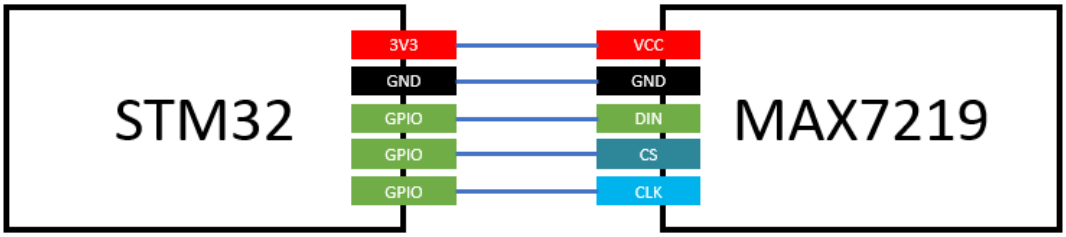
Each value of corresponding button is given below.

| | X0 | X1 | X2 | X3 |
|---|---|---|---|---|
| Y0 | 1 | 2 | 3 | + |
| Y1 | 4 | 5 | 6 | - |
| Y2 | 7 | 8 | 9 | * |
| Y3 | = | 0 | C | / |

## 4. Results and Discussion

### 4.1. Lab 6.0: Max7219 displayer (10%)

Hardware Design:




I use PA5 as DIN, PA6 as CS and PA7 as CLK to control MAX7219, which is following the lab5.

Code Explanation:

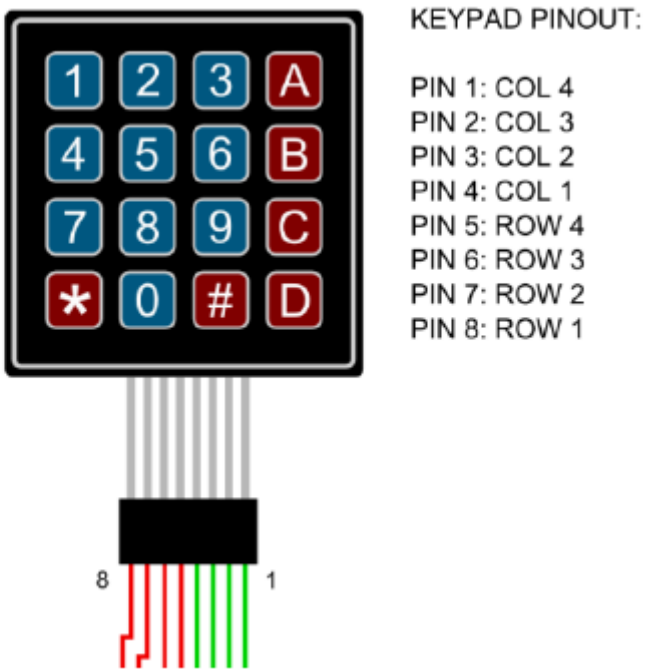
This is the first lab that we use C code to control our board. In the folder *src*, there are two files. They are *function.s* and *main.c* respectively. In the *function.s*, it contains *GPIO_init*, *MAX7219_init* and *MAX7219_send*, which all are wrote by the assembly language. In the *main.c*, we use C and write the function to display our student_id, and include all the functions in the *function.s*.

In the display function, we aim to show the data on 7-seg via max7219_send. The inputs are data which is decimal value and num_digs which is the number of digits will show on 7-seg. We return 0 if it successes. Otherwise, we return -1 due to illegal data range(out of 8 digits range). I also deal with the negative value.

4.2. Lab6.1: Keypad Scanning

Hardware Design:

The part of the 7-seg LED is same as Lab6.1.




PB5,6,7,9 PA8,9,10,12

On the part of the keypad, we need four inputs and four outputs. I use PA8,9,10,12 as outputs, and use PB5,6,7,9 as inputs. Which is corresponding to col1,2,3,4 and row1,2,3,4.

Code Explanation:

The code can be divided into three main parts. They are keypad_init, keypad_scan and display, respectively.

keypad_init: We initialize the GPIO inputs and outputs which we use to control the keypad scanning. We set the outputs as pull-up and medium speed mode, while we set inputs as pull-down and medium speed mode.

keypad_scan: This function scans the value which we pressed. There is a table we store all the value given in the experiment procedure. If someone presses the button, we will detect it. We will wait for a period of short time and detect it again. If the button still be pressed, we will check which button be pressed, and display it, and return the value.

Display: It is the same as Lab6.0.

### 4.3. Lab6.2 Deal with single or multiple buttons (30%)

Hardware Design:
Same as Lab6.1

Code Explanation:
This code is very similar to the Lab6.1. However, we need to deal with the debouncing problem and the multiple button pressing.
I use three array to record the state of each button, which is currentState, lastState and change. I check for each button if there is a change, which means the input is different from the lastState. We delay for a very short period time and check it again whether input is stiil different from the CurrentState or not. If it still different, we change the currentState. If the currentState is high, we add the value to the sum and display the sum. If the C button is pressed, we set the sum to zero.

### 4.4. Lab6.3 Deal with single or multiple buttons (BONUS)

Hardware Design:
Same as Lab6.1

Code Explanation:
The code is also very similar to the Lab6.2, but we still need to deal with some detailed things. I construct a linked list to store the operator and operand. If we press the equal button, we first calculate the multiply and divide operator, and then calculate the add and minus operator and then display the result.

## 5. Reviews and Applications

This Lab is the first lab that we use C code to control our board. Using C code is much more intuitive but we still need to understand the principal of the hardware, so that we can control the hardware in the right way. I think the most impressed thing for me is that I use linked list by myself to store the operand and operator and it does work. We need to remember to deal with the problem of the memory leak. This is really a nice lab. I learned very much.