# MCSL Lab05

# MAX7219 and 7-Seg LED

0310004
Kuan-Yen Chou

## 1. Purpose

- Understand how to use MAX7219.
- Design programs using 7-Seg LED.

## 2. Steps

### 2.1 Max7219 and 7-Seg LED practice – without code B decode mode

Here is the code.

### 2.2 Max7219 and 7-Seg LED practice – use code B decode mode

Here is the code.

### 2.3 Max7219 and 7-Seg LED practice – Fibonacci Sequence

Here is the code.

## 3. Results and Analysis

### 3.1 Max7219 and 7-Seg LED practice – without code B decode mode

In this lab, I use PB3 to PB5 as the three pins connected to the MAX7219. PB3 is the data input pin, PB4 is the clock pin, and PB5 is the load pin. Therefore, the first thing to do is to initialize those three pins in port B, and then to initialize the MAX7219. There are many configurable settings of the MAX7219 that we can tune. In this first practice, we are going to use the 'no decode' mode, so that we can display the letters that are specified in the 'arr' variable. The 'scan limit' is set to 0, which causes the 7-Seg LED to show only one digit. The 'intensity' can be set from 0x0 (the darkest) to 0xf (the brightest), and I set it to be 7.

As described in the class, I put the register address in r0, and the register data in r1. After setting up the values in r0 and r1, branch to the 'max7219_send' subroutine to send the whole message or command serially to the data input (PB3), and adjust the clock (PB4) accordingly. Finally, signal the load (PB5) to inform that the data transmission is finished. When the initialization of MAX7219 is done, the remaining part is much simpler. Set the r0 to 1 in order to show the first digit on the 7-Seg LED, and load the data from the 'arr' variable one byte at a time.

### 3.2 Max7219 and 7-Seg LED practice – use code B decode mode

Based on the previous practice, this practice is to display a given number on the 7-Seg LED, using the 'code B decode' mode. After the initialization of GPIO port B and the MAX7219, I send the number to be displayed one digit at a time. So I have to do some procedure similar to the process of division by the power of ten.

I send one digit at a time, from the most significant digit to the least significant digit. Using the 'get_digit' subroutine, which actually divides the displayed number by the powers of ten and returns the one-digit quotient stored in r1, I can consequently show the digits to the 7-Seg LED.

### 3.3 Max7219 and 7-Seg LED practice – Fibonacci Sequence

Beside the initialization routines, there are three chief routines that are executed repeatedly in a loop in this program of practice. 'check_btn' checks whether the USER button is pressed. If the button is not pressed, then do nothing. While the button is being pressed, then wait for a time of debounce, and compute how many cycles are passed when the button is pressed. If the time exceeds approximately one second, then it will reset the values; otherwise, it will add N by 1.

'fib' routine is called then to compute the fib(N) and store the value in the fibN. This procedure is mostly the same as the second practice in the Lab02. The biggest difference is that it begins with 0 rather than 1.

Finally, the job of 'display' routine is to display the value of fibN on the 7-Seg LED. It first checks whether the value is in the appropriate range. If it's out of range, then show -1 on the display. Otherwise, obtain the length of fibN through the 'fibN_len' subroutine, and then show the fibN in the same way as the previous practice.

## 4. Reviews and Applications

It is quite innovative to know how these techniques and communication mechanisms are designed. I learned a lot from the lab practices, though I wish we could do something more practical and useful. I think that the most difficult part for me is the control of the button, and designing the program control flow in order to let the program be responsive quickly. However, it is still exciting that we are going to write C code in the next lab. I think assembly is interesting and fascinating, but C is my favorite.