New Paltz
STATE UNIVERSITY OF NEW YORK

# lab 6

## Seven Segment Display

EGC332– Microcontrollers

March 24, 2021

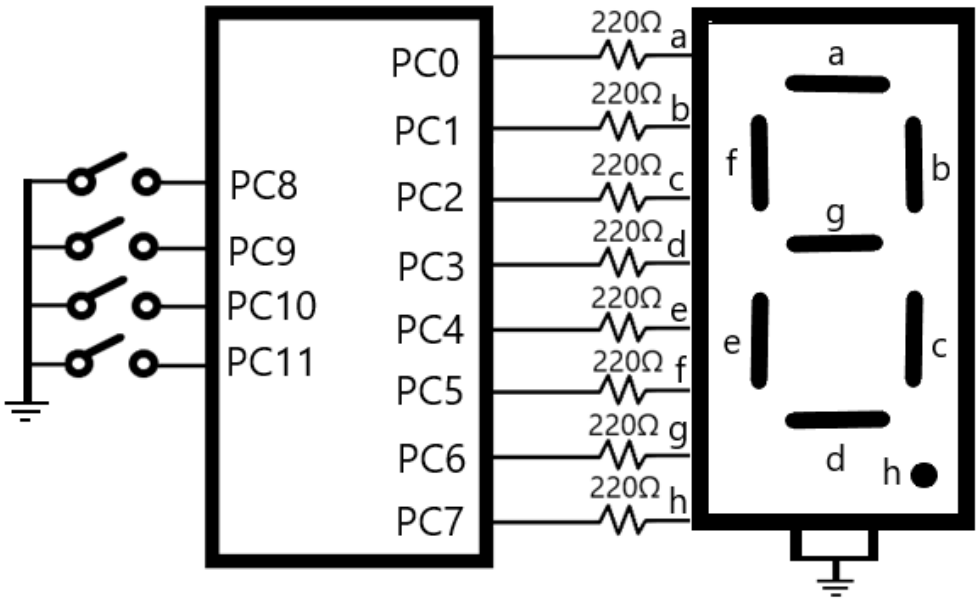
Christopher Lepore

# Introduction

In this lab we designed a BCD to 7 segment display system in C and assembly for a Nucleo-64 microcontroller. This introduced us to setting up and coding a seven-segment display through a STM32F446RE circuit board. This lab also introduced using the logic shift command in C and assembly to arrange the bits so they could be used effectively. The BCD to 7-segment display program takes in a 4-bit input value which then outputs the corresponding preset array value to the 7-segment display. The 7-segment display counter program does the same thing as the previous program, but it uses a counter instead of an input value. The counter starts at 15 and decreases by 1 until it reaches 0 where it jumps back to 15. The array is preset so the display cycles from 9 to 0 and blinks zero 3 times. By designing this BCD to 7-segment display, a deeper comprehension of coding and wiring the 7-segment display with the STM32F446RE circuit board was achieved.

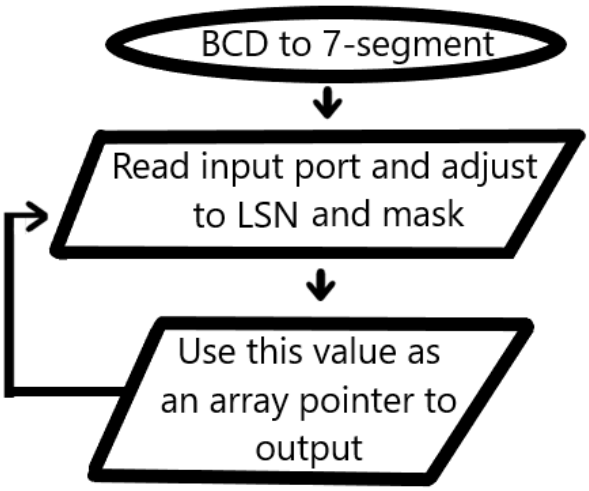
## Design

### Hardware (Block Diagrams)

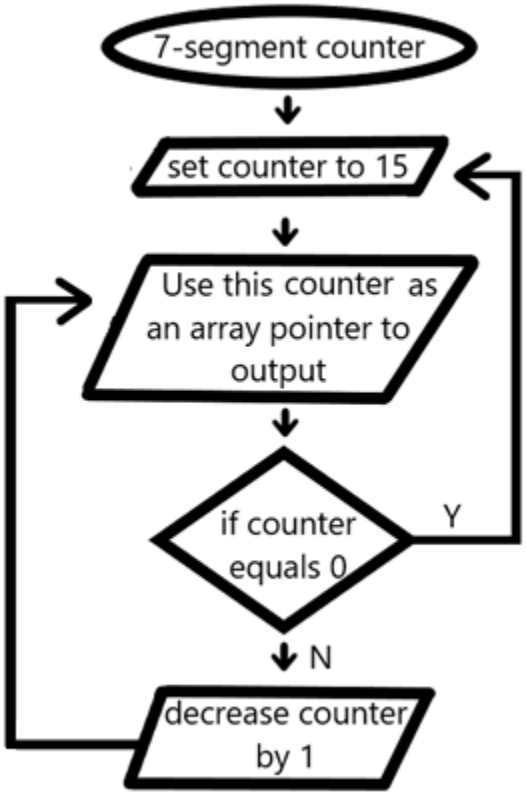



For the BCD to 7 segment display circuit, port PC0 to PC7 were set to output mode and then each connected to the seven-segment display from a to h accordingly. The common of the display are connected to ground. There are 220-ohm resistors in between them to prevent the LEDs of the display from burning out. Port PC8 to PC11 were set to input mode with pull up resistors to control the display. The BCD counter program uses an internal counter instead of the input ports.

Software (Flow Charts)




In BCD to 7 segment display program, the code will cycle through reading the input ports then adjusted the value to be used as an array pointer. The value array finds will then be outputted to the 7-segment display. This cycle will repeat forever.

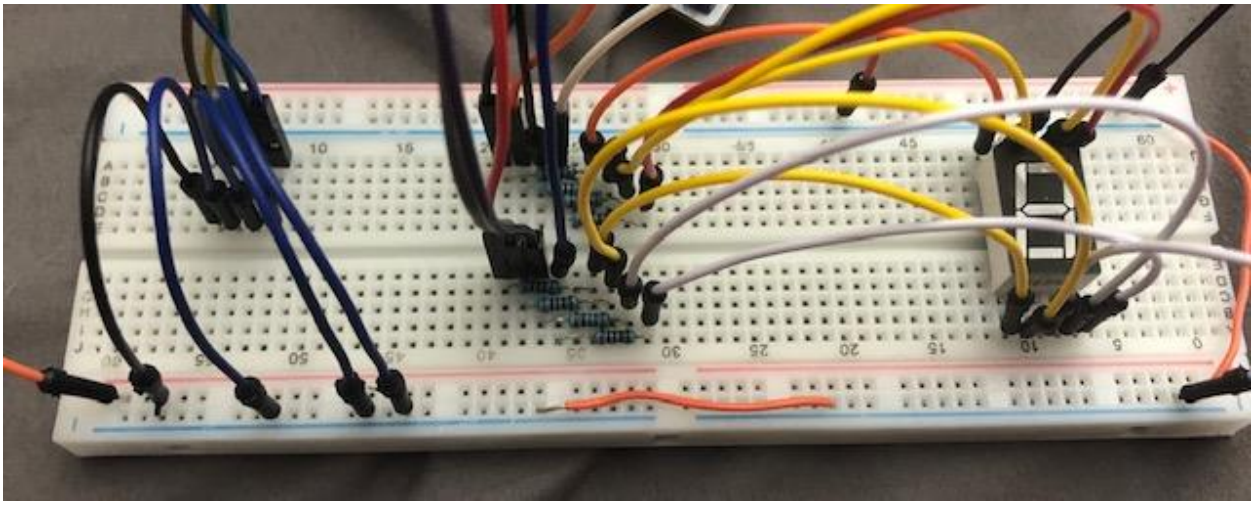



In the 7-segment display counter program, the code will first set a counter to 15. Then the code will cycle through using the counter as an array pointer which will output the array value to the 7-segment display and then decreased by 1. This

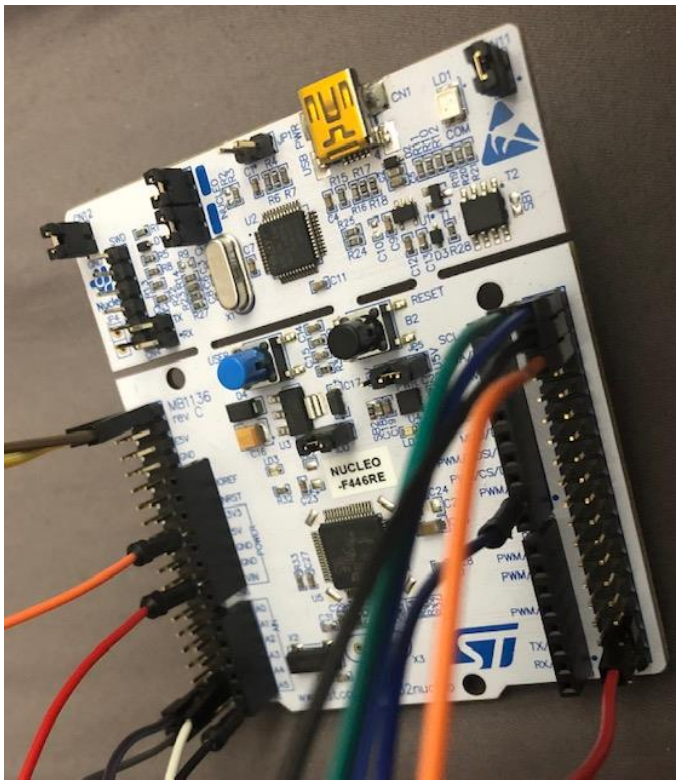
cycle will continue until the counter equals 0 where it will be set to 15 and continue the cycle.

## Verification

### Nucleo and breadboard circuit



Breadboard wiring




NUCLEO circuit board

## Summary and conclusion

The BCD to 7 segment display program for C and assembly cycles through obtaining a 4-bit input value from port PC8 to PC11 which is then shifted to the right by 8 bits. This value is then used as an array pointer and the value it points to is outputted to the 7-segment display through port PC0 to PC7 which represent a to h on the display. The BCD counter uses a decrementing variable for the array pointer which cycles from 15 to 0. The array is set up, so the display goes from 9 to 0 and flashes zero 3 times. These programs introduced us to setting up and coding a seven-segment display through a STM32F446RE circuit board and using the logic shift command in C and assembly. This was done by analyzing the pin schematic of a 7-segment display to set up the wiring for this project and by testing the LSR command in assembly. Due to the lessons above this lab has prepared students to begin designing and coding more complex projects with the 7-segment display.

# Appendix (Programs)

## BCD to 7 Segment Display using C:

```
#include "stm32f4xx.h"

int main(void) {
   RCC->AHB1ENR =  4;              /* enable GPIOC clock */

   GPIOC->PUPDR =  0x00550000;       /* set pin to input mode              PC 8 - 11 */
   GPIOC->MODER =  0x00005555;       /* set pin to output mode                   PC 0 - 7 */

unsigned int lookup[16] =
{0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7C,0x07,0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,0x71};
    while(1) {
        int i = ((GPIOC->IDR >> 8) & 0x0F);
        GPIOC->ODR = lookup[i];
   }
}
```

## BCD to 7 Segment using assembly:

```
        EXPORT  __Vectors
        EXPORT  Reset_Handler
        AREA    vectors, CODE, READONLY

__Vectors   DCD     0x10010000  ; 0x20008000    ; Top of Stack
        DCD     Reset_Handler             ; Reset Handler

RCC_AHB1ENR equ 0x40023830
GPIOC_MODER equ 0x40020800
GPIOC_ODR   equ 0x40020814

GPIOC_PUPDR equ 0x4002080C
GPIOC_IDR   equ 0x40020810

        AREA    PROG, CODE, READONLY
Reset_Handler
        ldr     r4, =RCC_AHB1ENR    ; enable GPIOC clock
        mov     r5, #4
        str     r5, [r4]

        ldr     r4, =GPIOC_PUPDR
        ldr     r5, =0x00550000       ; set pin to input mode (pull up resistor)         PC 8 - 11
        str     r5, [r4]
```

```
        ldr     r4, =GPIOC_MODER
        ldr     r5, =0x00005555             ; set pin to output mode        PC 0 - 7
        str     r5, [r4]

        ldr     r0, =GPIOC_IDR      ;input port
        ldr     r1, =GPIOC_ODR      ;output port
        ADR     r8, SevenSeg

again   ldr     r3, [r0]            ; read switches
        LSR     r3, r3, #8          ; shifts input bits 8 to the right
        bic     r3, r3, #0xF0       ; isolate PC8 - PC11 , clear PC0 - PC7
        ldrb    r4, [r8, r3]        ; gets value from array
        strb    r4, [r1]            ; stores value in output
        b       again

SevenSeg    DCB
0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7C,0x07,0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,0x71

        end
```

## 7 Segment Display countdown using C:

```
#include "stm32f4xx.h"

void delayMs(int n);

int main(void) {
  RCC->AHB1ENR =  4;              /* enable GPIOC clock */

  GPIOC->MODER =  0x00005555;     /* set pin to output mode                  PC 0 - 7 */

  unsigned int lookup[16] =
{0x00,0x3F,0x00,0x3F,0x00,0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7C,0x07,0x7F,0x6F};
  int i;
  while(1){

      for(i=15; i>0; i--)
      {
         GPIOC->ODR = lookup[i];
         delayMs(500);
      }
  }
}

/* 16 MHz SYSCLK */
void delayMs(int n) {
```

```
    int i;

    /* Configure SysTick */
    SysTick->LOAD = 16000;  /* reload with number of clocks per millisecond */
    SysTick->VAL = 0;       /* clear current value register */
    SysTick->CTRL = 0x5;    /* Enable the timer */

    for(i = 0; i < n; i++) {
       while((SysTick->CTRL & 0x10000) == 0) /* wait until the COUNTFLAG is set */
           { }
    }
    SysTick->CTRL = 0;      /* Stop the timer (Enable = 0) */
}
```