ECE0202: Embedded Systems and Interfacing

Lab 3: Branching, 7-Segment Displays (in assembly)
Due: 3/7/21 at 11:59 PM

Objectives

- **Write assembly code that counts up from 0-9 in a loop**
- **Display the current number as the count is happening on a 7-segment display**
- **Include the blue button to reset the count to 0 when pressed**
- **A breadboard circuit where the Nucleo board is connected to a 7-segment display driver chip and the current number count is displayed on a 7-segment display.**

Deliverables – total 100 points

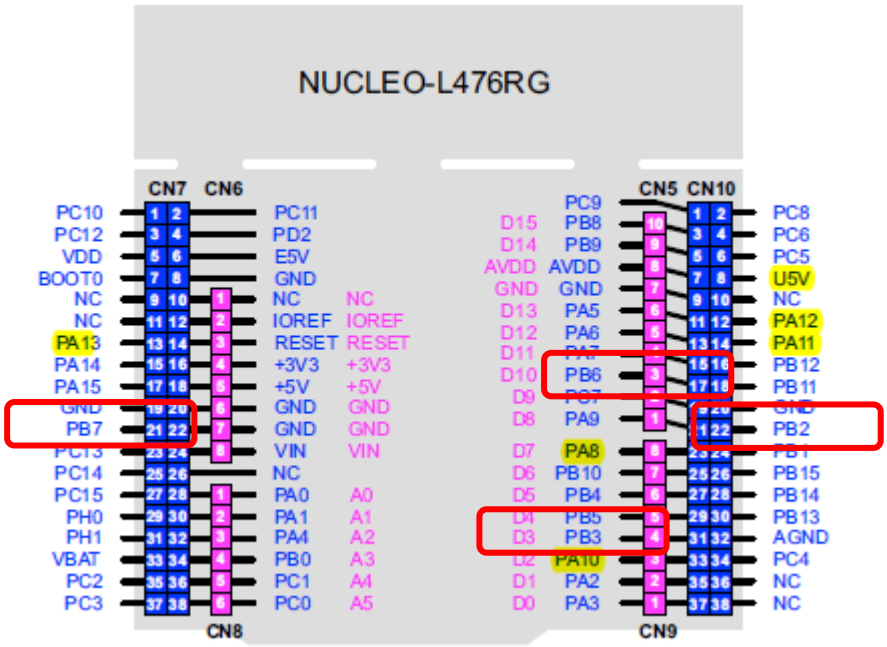
- **(50 points) Submission of code that is able to count upwards from 0-9 in a loop and export the current number via output GPIO pins, as well as resetting the count to 0 when the blue button is pressed on an input GPIO pin.**
- **(50 points) A Demonstration of operational circuit and program to the instructor or a TA.**
- **(10 points extra credit) Code that performs the following operation:**
    - **The blue button counts down by 2 on the 7-segment display with each click**

Please submit your code as *.s files and your schematic as a pdf

Getting Started

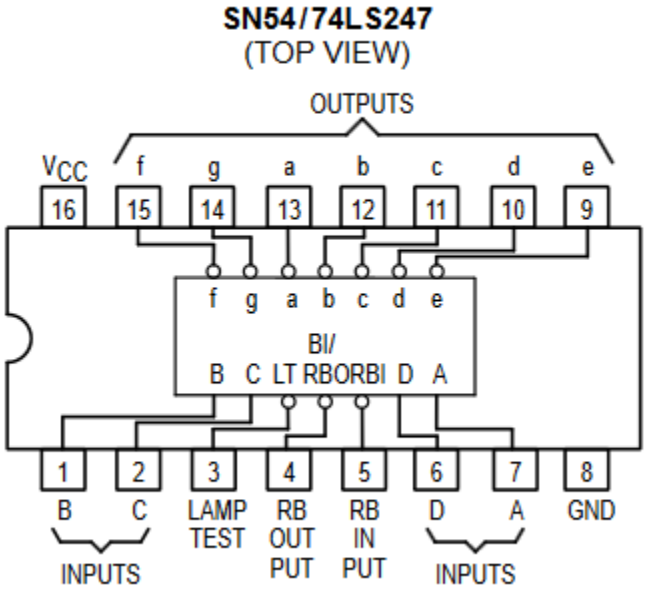
- This project will be written in assembly rather than in C code.
- Download the template from Canvas.
- Write the code for this lab in the “main.s” file.

GPIO pins



In this lab, you will need to use both input and output GPIO pins. The methods for configuring GPIO pins have been covered in the lectures and in Lab 2. The best pins to use as general-purpose output pins when interfacing with external devices are GPIOB 2, 3, 6, and 7 (circled in the diagram above).

LS247N BCD to 7-segment display drivers



The datasheet for the LS247N chips has been posted on Canvas. Only the highlights are given here.

You have most likely derived the truth tables for converting a 4-bit BCD number into on/off states for a 7-segment display. Luckily, these truth tables have been implemented in chips and you do not have to implement this logic yourself.

Given a 4-bit BCD number (exported onto GPIO pins) where D = bit 3 (MSB), C = bit 2, B = bit 1, and A = bit 0 (LSB), the inputs to the chip are given in the overhead diagram above.

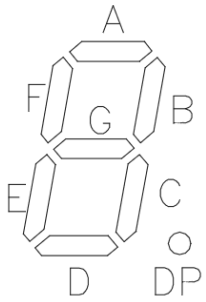
D is the Most Significant Bit (MSB) and A is the Least Significant Bit (LSB).

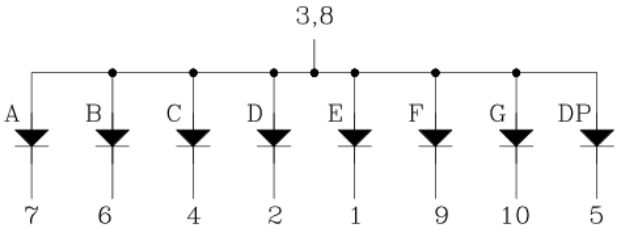
Pins 9-15 are open-collector (the BJT equivalent of open-drain) outputs used to sink current for the LEDs in the seven-segment display.

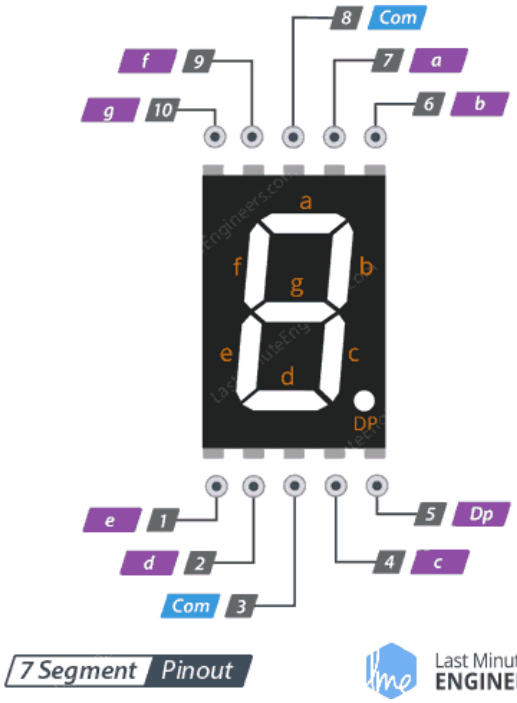
7-segment displays

A 7-segment display has 7 LEDs and is able to display the numbers 0-9 as well as some English characters (though that capability won't be used here). A diagram of the LED layout as well as the naming system (a-g) is given below.

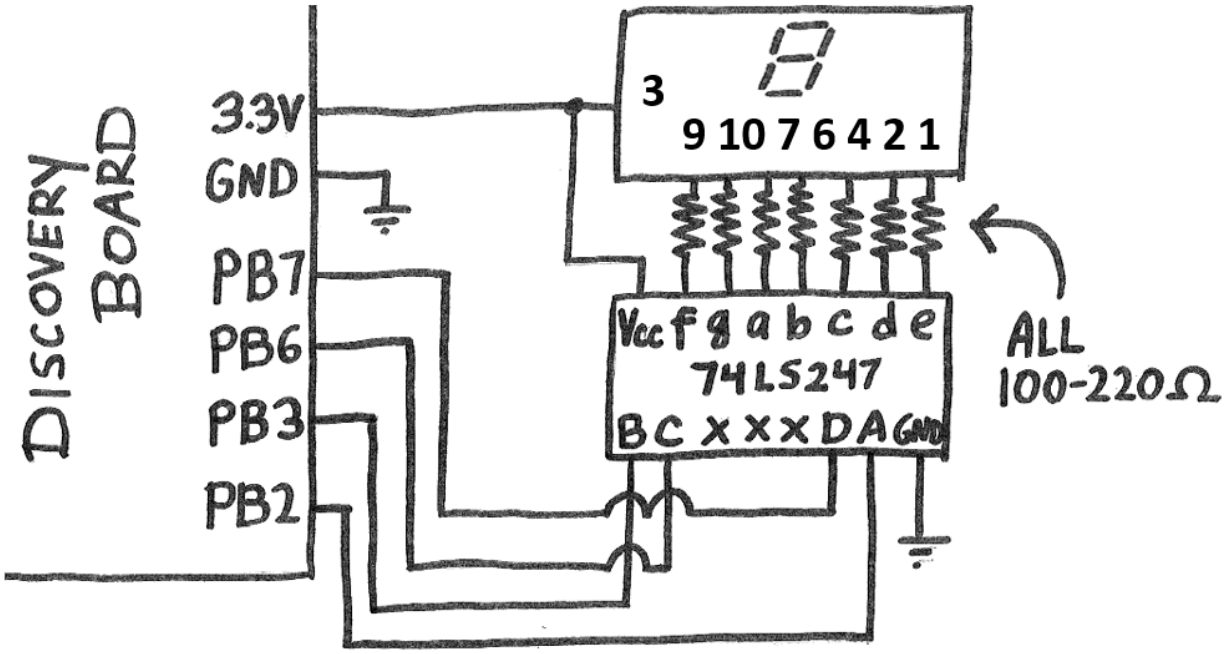
This 7-segment display is a common-anode device, meaning that there is a single connection to a positive voltage for all of the 7 LEDs, but they each have their own connection to a negative voltage (you must put a current limiting resistor in between each LED's cathode and ground, or in this case, the outputs from the LS247N chip). The electrical schematics of the 7-segment display are given below. You can ignore the DP LED as we will not be using it.







Overall Schematic



Note: devices are not to scale, and pins are not necessarily drawn in their correct locations.

Assembly Register Declarations

The files that you include in your project contain register memory address declarations so that you do not need to look these addresses up in the datasheet yourself. The following list gives all necessary register declarations that you will need to use in this project.

- RCC_BASE: The base address of the Reset and Clock Control (RCC) module
- RCC_AHB2ENR: The offset for the AHB2ENR register in the RCC module
- GPIOC_BASE: The base address of the GPIOC modules
- GPIOB_BASE: The base address of the GPIOB modules
- GPIO_MODER: The offset for the Mode Register (MODER) for any GPIO module
- GPIO_ODR: The offset for the Output Data Register (ODR) for any GPIO module
- GPIO_IDR: The offset for the Input Data Register (IDR) for any GPIO module

Lab 3: 7-Segment Counter in Assembly

•••

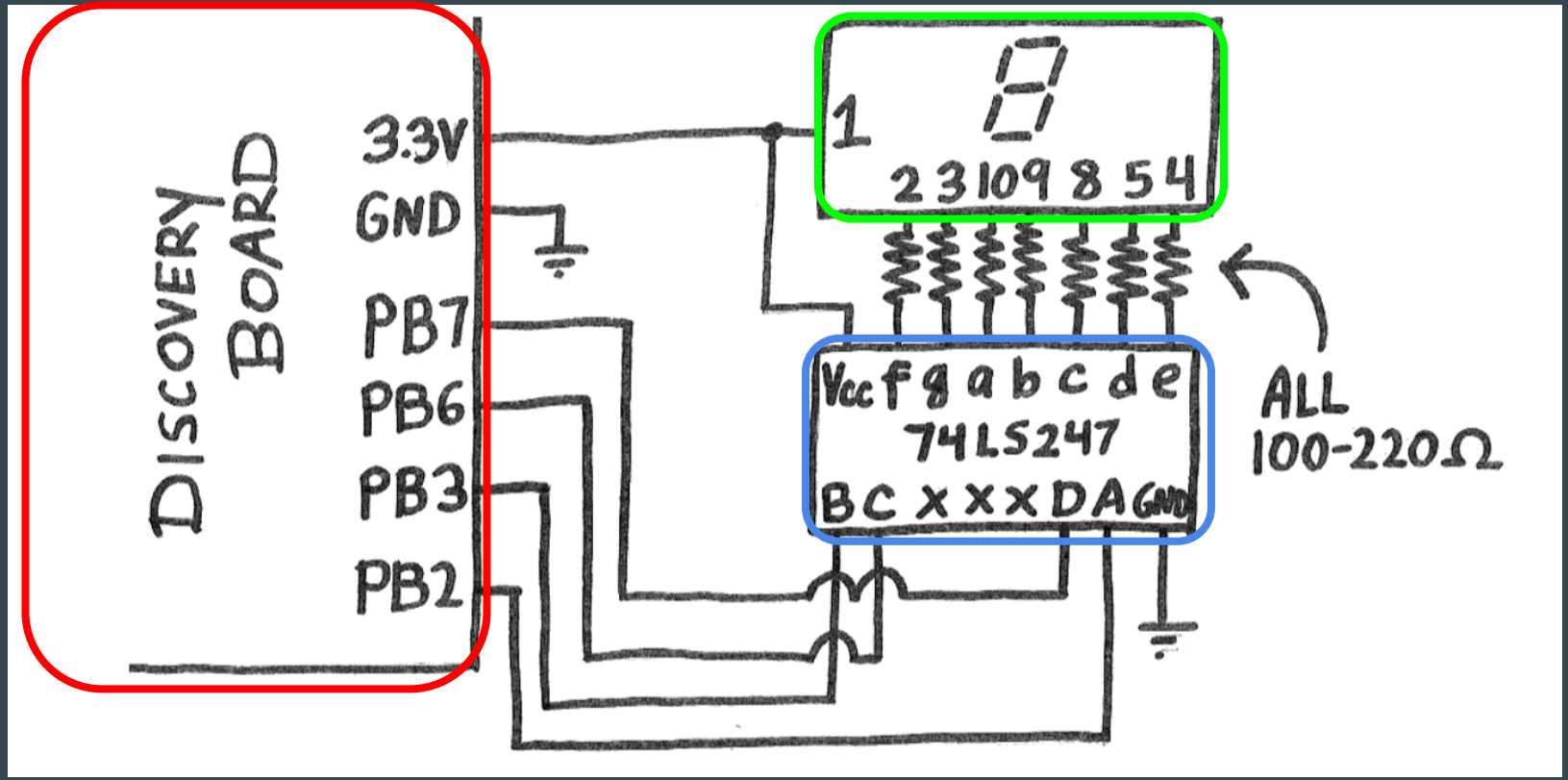
ECE 0202

Lab Outline

1. Download assembly template project from courseweb (same as lab 1)
2. Enable peripheral GPIO pins (4 output, 1 input)
3. Write a loop that updates output pins after a delay
4. Write logic within that loop to check if reset has been pressed

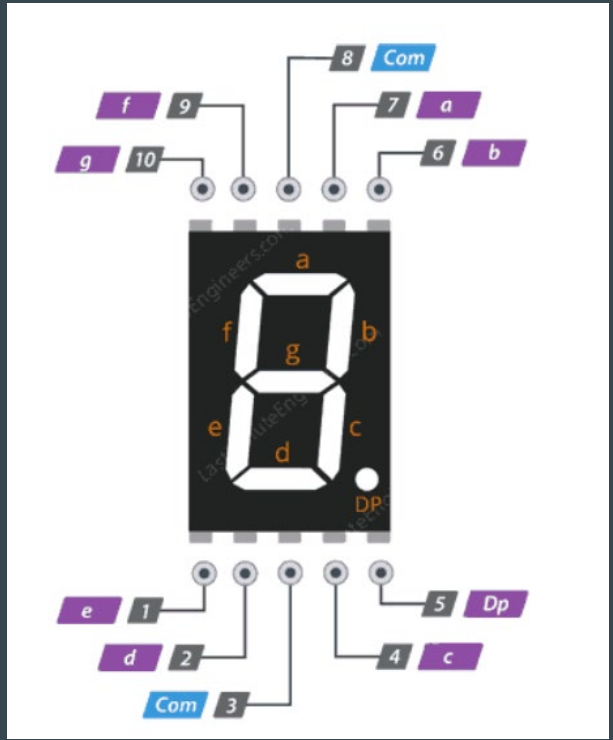
- Show us counting 0-9 on the LCD (checkoff)
- Show us that you have a reset button (checkoff)
- Submit code (e.g. 'main.s') on courseweb

The Circuit


DISCOVERY BOARD
3.3V
GND
PB7
PB6
PB3
PB2
1
2 3 10 9 8 5 4
Vcc f g a b c d e
74LS247
BC x x x DA GND
ALL
100-220Ω

LCD Pinout

All lowercase letters should be connected to 7 segment display driver through the 220 Ω resistors



Enabling GPIO pins in Assembly

We need to do something called **pre-indexing**

1. Load the base register address
2. Load the value located in base register + offset
3. Change that value with bitwise operations
4. Store the value in the address of base register + offset

*See lecture 10 for more examples
*As well as the PreLab for Lab 2

Equivalent C code: `GPIOE->PUPDR |= 0x1;`

```
LDR r0, =GPIOE_BASE
LDR r1, [r0, #GPIO_PUPDR]
ORR r1, #0x1
STR r1, [r0, #GPIO_PUPDR]
```

Control Logic

```
// My Pseudocode
Setup_GPIO();
while(1){
   delay = 100;
   while (delay !=0) {
      delay -=1;
   }
   count++;
   display(count);
   check(reset);
}
```

Our main loop (after pin initialization)

Some delay operation

After the delay, we update our counter

Once per loop we check our reset button

Potential Steps for the Lab

1. Configure peripherals: GPIO (inputs and outputs)
    a. Verify these using debugger
2. Wire up circuit using breadboard, 7-segment display and driver
    a. Test that circuit works by writing out number like 3 and check that it is on display
3. Write logic for counting up
4. Add reset button