EE337 Microprocessors Laboratory

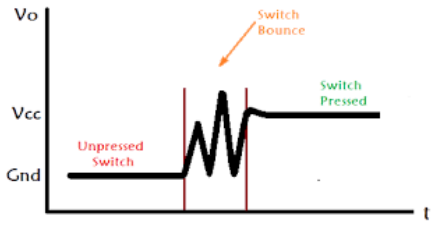
Wadhwani Electronics Laboratory
Electrical Engineering IIT Bombay

Problem set: 5 Date: February 6, 2025

Aim - Design and implement a key debouncing mechanism to ensure reliable detection of switch states.

In this lab, you will write an embedded-c program to understand the concept of debouncing. You will configure some port pins as inputs, some as output and demonstrate the concept of debouncing.

Key Debouncing: It is a technique used to eliminate unwanted multiple signals caused by the mechanical bouncing of a key or switch when pressed or released. When a key is actuated, it does not make a clean transition but instead oscillates between ON and OFF states before settling. Debouncing can be implemented in hardware using capacitors or in software using time delays or algorithms to filter out false signals. It ensures only a single valid key press is registered, improving input accuracy in keyboards, switches, and micro-controller applications.



Implement a debounce algorithm using a delay to ensure that only stable switch states are detected.

1. [20 points] Functional Requirements:

   - Each switch is mapped to a corresponding LED.
   - When a switch transitions from OFF to ON and remains stable after the debounce delay, turn ON the corresponding LED and turn OFF all others.
   - When a switch transitions from ON to OFF and remains stable after the debounce delay, briefly turn ON all LEDs as a visual indication and then turn them off after 1 second.
   - If a switch changes position (ON to OFF or OFF to ON) before the debounce delay expires, discard the change and maintain the previous state.
   - If no switches are pressed or released, all LEDs should remain OFF.
   - Continuously monitor the state of the switches.

   Timing:

- The debounce delay should be implemented as 2 seconds (just for understanding of the debouncing concept, in practice it is 10ms to 20ms).
- The duration for all LEDs ON (after an OFF-to-ON transition) must be 1 second.

Configure GPIO ports for switches as inputs (P1.0 to P1.3).
Configure GPIO ports for LEDs as outputs (P1.4 to P1.7).

Hints

- ```
  // Use P1.0{P1.3 (SW1 to SW4) as input pins to read the state of the switches
  // Use P1.4{P1.7 (LED1 to LED4) as output pins to control LEDs.
  // For ex.
  sbit sw1 = P1^0;
  sbit LED1 = P1^4;
  ```

- You can use following function to generate delay in milliseconds. The value of 'time' will decide duration of delay. If 10 is passed as an argument to the function, it will give you a delay of 10 milli-seconds

  ```
  void msdelay(unsigned int time)
  {
          int i,j;
          for(i=0;i<time;i++)
          {
                  for(j=0;j<382;j++);
          }
  }
  ```

- You may use this template to start with your first Embedded.c code

  ```
  #include <at89c5131.h>
  sbit LED1 = P1^4;
  // Add other port bits
  // You can write your functions here

  void main()
  {
      /*
      Add your code here
      */


  }
  ```
  ```

TA Checkpoints

1 Correct Implementation of the 2-Second Debounce Delay, 1-Second LED Delay, and Switch-to-LED Mapping (5 Points)

2 OFF-to-ON State Transition Handling (5 Points)

3 ON-to-OFF State Transition Handling (5 Points)

4 Handling Intermediate State Changes (Switch Position Change Before Debounce) (5 Points)