Indian Institute of Technology Bombay
CS684/CS308 Embedded Systems

Interrupts and Timers

E.R.T.S. Lab

1 Lab Objective

This lab will introduce you to the use of Timers and Interrupts on the TM4C123GH6PM.

2 Pre-requisite

1. Lab 1: Interface led and both the switches.
2. Reference material: Please go through student guide and lab manual (Chapter 4) given in the resource section before you proceed further.

3 Problem Statement

1. Use sw1 to change the color of the led (R⇒ G⇒ B⇒ R....) where you should press the switch just once instead of long press in Lab 1. Use switch debouncing mentioned below in the procedure to differentiate between switch bounce and actual key press.
2. Use sw2 to increment a global variable once for each button press. Check if the variable always increments by one (adjust the time interval of 10 ms if you wish to)

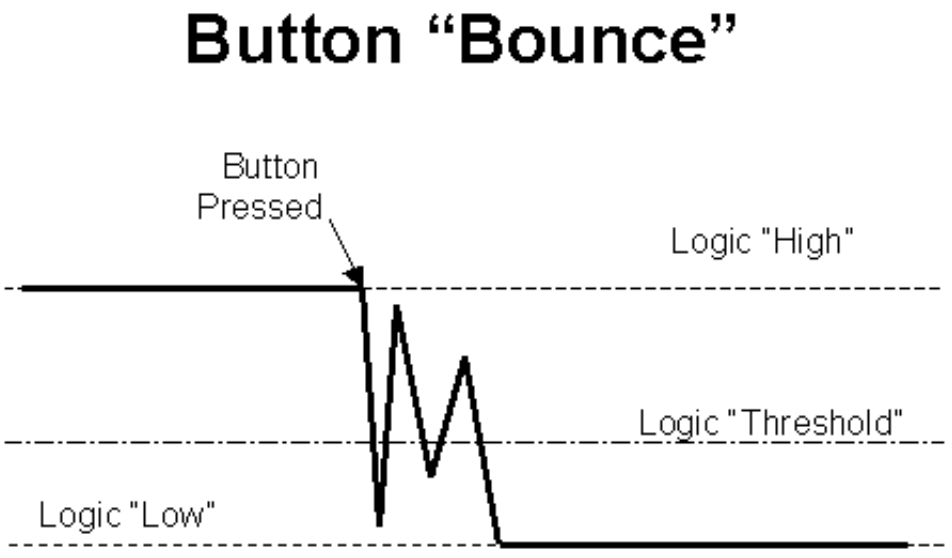

Figure 1: Button Bounce Waveform
*

4 Relevant Theory

This lab will use a timer to generate interrupts and we will write a timer interrupt service routine (ISR) that implements a state machine to perform **“software button debouncing”**.

Button Bounce: From the above diagram it is clear that when you press the switch the signal bounces for a finite amount of time before settling down to its final logic state. This is due to the tendency of any two metal contacts in an electronic device to generate multiple signals as the contacts close or open. The bounce period is 10 to 20 ms.

Solution: There are several ways to do button debouncing which includes modifications in both hardware and software. In this lab we will explore software debouncing. In software debouncing too, there are a couple of ways to solve this problem viz.

1. **Polling the switch plus introducing finite delays:** In polling method the controller is busy polling the switch hence it will not be able to perform any other tasks.
2. **Using interrupts and timers:** If we want to use this time to do other tasks, then we can use interrupts and timers.

Switch debouncing using interrupts and timers: We will check for the status of the switch at finite intervals (say 10 ms). This time interval will be generated using timers. When the 10 ms interval is over the timer should generate an interrupt and the code in the interrupt service routine (ISR) will be executed. In the interrupt service routine, we

will implement a state machine. Here the idea is that if we find the switch pressed for two successive intervals (of 10 ms) then we confirm that the switch is pressed and we will detect the next key press only if we confirm that the switch is released. This will ensure that if a switch press is only detected once.

A state machine is described below for implementing software debouncing. In this state machine there are three states viz. Idle, Press and Release for a switch. There are two transition paths in each state. Any state transition condition is checked after a fixed interval of time which is set by a timer (in this case it is 10 ms).

Note: As the bounce period is 10 ms to 20 ms you can change this time interval based on your experimental trials. It is possible that different switches have different bounce periods.

IDLE
If the key is pressed, enter press state
else remain in IDLE state

PRESS
If key is still pressed, then enter release state and make flag 1 indicating that key was pressed.
else return to idle state (de-bouncing period)

RELEASE
If the key is released, then enter an idle state
else remain in release state until key is released.

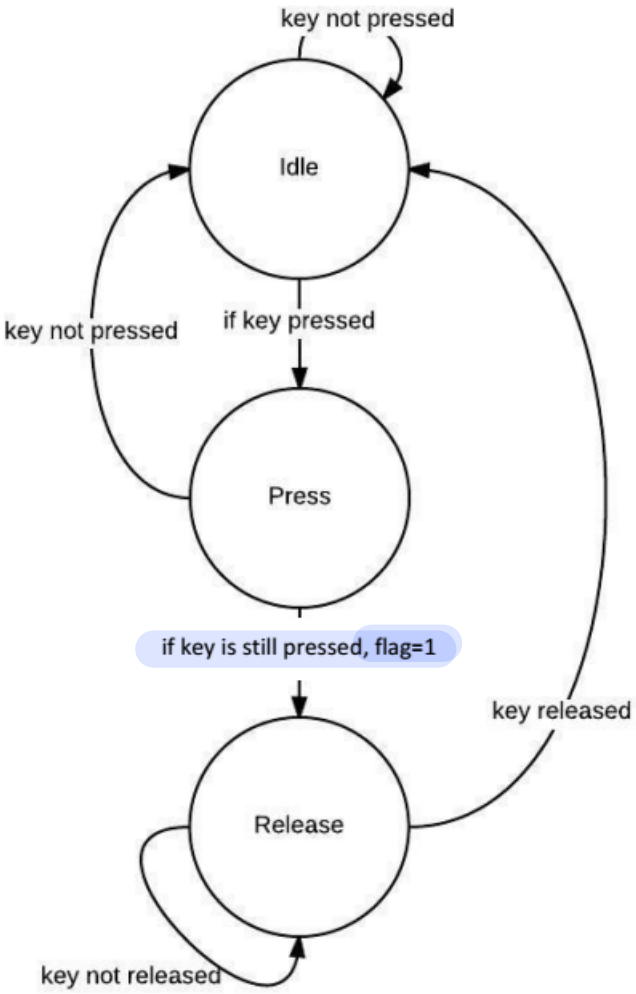

Figure 2: Switch debouncing state diagram

5 Procedure

1. Configure and initialize Timer 0 and set the time interval to 10ms. Use the lab exercise in student guide and lab manual (Chapter 4) as a starting point.

2. Make a user defined function called "detectKeyPress()" and call this in the interrupt handler. This function will return 1 only if a key press is detected according to the state machine. The prototype of the function is given below.

```
unsigned char detectKeyPress()
{
return flag;
}
```

3. Describe the state machine**Figure:2** in this function and each time the function is called from the interrupt handler the machine should make a transition to the next stage. Make sure that the previous state (Scope of variables) of the system is maintained each time the function is called.

4. Write a code for debouncing both sw1 and sw2.

5. Verify that both the switches are de-bounced.

Note: The action associated with the switches should be performed exactly once for each switch press. Long press of the switch should not lead to multiple triggers.

6 Demo and Submissions

- Show the output to your TA and explain if this approach helps in debouncing the switches.
- Document your code. You will be asked to upload your codes on GitHub.
- It is recommended that you make your code modular and use this debouncing method in upcoming labs.