One of the complicating issues with mechanical switches is they can bounce (oscillate on and off) when touched and when released. The contact bounce varies from switch to switch and from time to time, but usually bouncing is a transient event lasting less than 5 ms. We can eliminate the effect of bounce if we design software that waits at least 10 ms between times we read the switch values.

To interface a switch we connect it to a pin (e.g., Figure 8.3) and initialize the pin as an input. The initialization function will enable the clock, set the direction register to input, turn off the alternative function, and enable the pin. Notice the software is friendly because it just affects PA5 without affecting the other bits in Port A. The input function reads Port A and returns a true (0x20) if the switch is pressed and returns a false (0) if the switch is not pressed. Figure 8.4 shows how we could build this circuit with a protoboard and a LaunchPad.

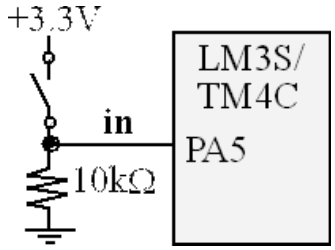

Figure 8.3. Interface of a switch to a microcomputer input.

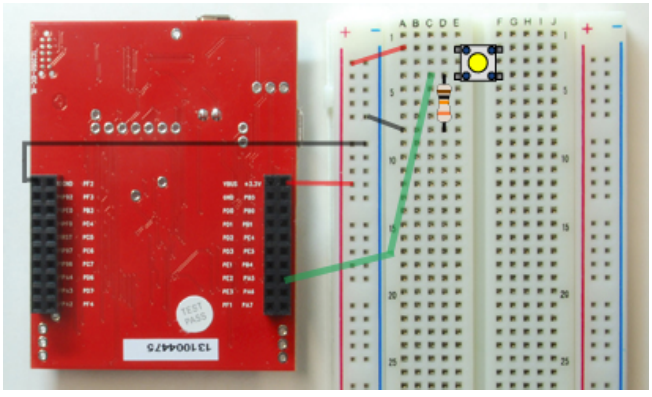

*Figure 8.4. Construction of the interface of a switch to a microcontroller input. The brown-black-orange resistor is 10k. The switches in the lab-kit should plug into the protoboard. The switch is across the two pins that are closer to each other. It doesn't matter what color the wires are, but in this figure the wires are black, red and green. The two black wires are ground, the red wire is +3.3V, and the green wire is the signal **in**, which connects the switch to PA5 of the microcontroller.*

The software in Program 8.1 is called a **driver**, and it includes an initialization, which is called once, and a second function that can be called to read the current position of the switch. Writing software this way is called an **abstraction**, because it separates what the switch does (Init, On, Off) from how it works (PortA, bit 5, TM4C123). The first input function uses the bit-specific address to get just PA5, while the second reads the entire port and selects bit 5 using a logical AND.

```
#define PA5   (*((volatile unsigned long *)0x40004080))
void Switch_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000001;     // 1) activate clock for Port A
  delay = SYSCTL_RCGC2_R;           // allow time for clock to start
                                    // 2) no need to unlock GPIO Port A
  GPIO_PORTA_AMSEL_R &= ~0x20;      // 3) disable analog on PA5
  GPIO_PORTA_PCTL_R &= ~0x00F00000; // 4) PCTL GPIO on PA5
  GPIO_PORTA_DIR_R &= ~0x20;        // 5) direction PA5 input
  GPIO_PORTA_AFSEL_R &= ~0x20;      // 6) PA5 regular port function
  GPIO_PORTA_DEN_R |= 0x20;         // 7) enable PA5 digital port
}
unsigned long Switch_Input(void){
  return PA5; // return 0x20(pressed) or 0(not pressed)
}
unsigned long Switch_Input2(void){
  return (GPIO_PORTA_DATA_R&0x20); // 0x20(pressed) or 0(not pressed)
}
```

Program 8.1. Software interface for a switch on PA5 (C8_Switch).

Maintenance Tip: When interacting with just some of the bits of an I/O register it is better to modify just the bits of interest, leaving the other bits unchanged. In this way, the action of one piece of software does not undo the action of another piece.

In this class, we call software that only affects the necessary bits as **friendly**. By the way, we have invented this term, so when communicating with people outside of this class, you will need to define what you mean by friendly.

Help

(https://plus.google.com
/108235383044095082735/posts)

(http://youtube.com/user/edxonline)
© 2014 edX, some rights reserved.

Terms of Service and Honor Code -
Privacy Policy (https://www.edx.org/edx-privacy-policy)

Help