Chapter Six: Interfacing a touch sensor

1. Sensors

As humans, we often take for granted our amazing perceptual systems. We see a cup sitting on a table, automatically reach out to pick it up and think nothing of it. At least, we are not aware of thinking much of it. In fact, accomplishing the simple task of drinking from a cup requires a complex interplay of sensing, interpretation, cognition, and coordination, which we understand only minimally.

So, how does your robot “talk” to the outside world? Through sensors. Sensors are of various types: simple bump sensors that tell your robot that it has hit something, temperature sensors, light sensors, rotation sensors, sonar or even cameras! This chapter gives some hints on using a touch sensor. The main issue involved with using sensors is eliminating noise from sensor information.

2. Interfacing a Touch Sensor

Using a touch sensor is simple, because it just tells the robot that it has hit something. Therefore, we can use simply use a switch. If the switch is closed, then the robot has hit something. The following diagram shows a Normally Open (NO) switch:

![Diagram of a Normally Open (NO) switch]

Figure 1. A simple touch sensor for the CalBOT.

If the switch is not pressed, then NC and C are connected (NC – “normally closed”). If the switch is pressed, then NO and C are connected as shown below (NO – “normally open”).
Figure 2. Operation of the switch. If the switch is open, as shown in the left diagram, then C and NC are connected. If the switch is closed, C and NO are connected.

Thus, using this switch as a bump sensor is simple. Connect the C input from the switch to any pin on the Kit-CON connector that you can configure as an input pin. Connect the NO input from the switch to digital $V_{CC}$ (the +5 V on the Kit-CON connector). To make sure that you have a 0 when the switch is open, connect the NC input from the switch to GND on the Kit-CON connector. This is shown in figure 3. **DO NOT CONNECT THE C INPUT TO DIGITAL $V_{CC}$ AND THE NC INPUT TO GND. THIS WILL SHORT THE BOARD!**

Figure 3. One possible interface for the switch

The 5 Kohm resistor is to prevent you from accidentally shorting the board.
3. Conditional Branches

How do you check in software if the sensor has hit something? In other words, how do you check if something is true or not? You can use the `if-else` statements as shown in the program below.

```c
#include <reg167.h>
void main(void)
{
    int a = 2;
    if(a == 0)
    {
        a = a + 3;
    }
    else if(a == 1)
    {
        a = a + 4;
    }
    else
    a = a + 5;
}
```

*Program 1. if-else if-else illustration*

When you run this program, the value of `a` will change to 7. This is because, the value of "a" is 2, and C will ignore the `if` and the `else-if` and just execute the final `else` case. *Many a C programmer has made the mistake of using single equals (=) instead of double equals (==) in the if-else if-else cases. Please do not fall into this trap!* Let us write a simple program to test a bump sensor. Assuming this sensor is connected to PORT8_4, the program is shown below.

```c
#include <reg167.h>
void kludgyWait(void) {
    float i;
    for(i = 0;i < 2500;i+=0.1) {
    ;
    }
}
void main(void)
{
    DP8 = 0x0F; /* Bit 4 is 0, since P8_4 is configured as input */
    for(;;)
    {
        if(((P8 & 0x10) && (P8 & 0x10) && (P8 & 0x10)) == 1) {
            /* switch has hit, we will do something */
            kludgyWait(); /* Wait till the above code finishes */
        }
    }
}
```

4. Switch debouncing

The only trick in the above code is checking for an actual switch hit. Since there is a lot of noise due to the motors, the C167 may pick up the noise spikes as switch inputs. In order to avoid this we check if we have a switch hit multiple times. The duration of the noise spike is usually much smaller than the time taken to check for four switch hits\(^1\). This procedure is called *switch debouncing*. Another use of debouncing is eliminating error due to mechanical limitations of your switch.

5. Summary

Well, this chapter touched the tip of an iceberg. The world of robotic sensors is fascinating. It is a separate research subject in itself. A simple touch sensor (like the one described in this chapter) is enough to implement simple obstacle avoidance behavior. The last chapter in the manual shows you common problems encountered and gives you final pointers on the different ports in the C167.

---

\(^1\) This is an extremely kludge method for eliminating noise. We could use bypass capacitors and Schmitt triggers, but they are beyond the scope of this class.