Example 2 – Button Input Capture

The second example builds upon the first, replacing a timer with more advanced GPIO in the form of buttons S1 and S2. We used GPIO in example 1 to toggle the LEDs on and off, and will do the same in this example, but we will also use the GPIO functions to set up the ports connected to the buttons as inputs that trigger interrupts. In the interrupt service routine (ISR), the LEDs will be toggled, depending on which button is pressed.

This example once again begins with a comment block explaining the example and the inclusion of the driver library. We skip setting up the clocks unlike the previous example, so the MSP430 will run at its default 1MHz. The GPIO is set up starting on line 38. Ports P1.0 and P4.0 are set as outputs for the LEDs as before. Ports P5.5 and P5.6 are set as inputs with internal pull-up resistors. As you can see from the schematic, the buttons are switches to ground, so in order to get a good high-to-low transition that the MSP430 can recognize, a pull-up resistor is required. On line 44, the interrupt edge is selected. When the buttons are not being pressed, they are connected to 3.3V via the internal pull-up resistor, and when the buttons are pressed, they are connected to ground. So, if we want to trigger an interrupt when the button is pressed, we need to select the high-to-low edge as our interrupt source.

After the ports are set up, we clear the interrupt flags so that when we enable interrupts, we don't get any phantom interrupts. Then the MSP430 is set to low-power mode 4 and interrupts are enabled. Notice the power level is much lower than in previous examples as no clocks are needed for a timer.

The last part of code is the interrupt service routine (ISR). Again, this code is only run when an interrupt is triggered. The only major difference between this ISR and the ISR in example 1 is that this ISR can be triggered by multiple sources, i.e. either button. To differentiate between sources, a switch statement is used on the P5IV register, which contains the source of the interrupt. The LEDs act as a sort of two-bit counter from 00 (both off) to 11 (both on) in binary. One button increments the counter and the other resets it to 00 (both LEDs off).

```
1 #pragma vector=PORT5_VECTOR
2 __interrupt
3 void Port_5(void) {
4     switch (__even_in_range(P5IV, 16)) {
5     case 12:
6         //P1.0 = toggle
7         GPIO_toggleOutputOnPin(GPIO_PORT_P1, GPIO_PIN0);
8         break;
```