Lab 4 Report
CSE 4560 Embedded System

Name: Daniel Delgado Acosta Coyote ID:006896598

I Goals

Goals of this Lab

- **Learn the difference between polling and interrupts**
- **Configure and implement hardware interrupts**
- **Learn how to debug code running on the microprocessor**
- **Configure and use hardware timers**
- **Understand and implement pulse-width modulation (PWM)**

II Lab Questions, Processes and Program

Task 1: Interrupts and Debugging

Task 1.1: Interrupts with Buttons

The goal of this task is to setup interrupts such that pressing button S1 toggles the green LED and pressing button S2 toggles the blue LED.

Task 1.2: Breakpoints and Stepping Through Code

In this task you are going to use the IDE's debugger functionality to step through the code while it is executed on the Launchpad. Furthermore, you see how to work with breakpoints. We re-use the code of Task 1.1 as a sample program.

Press the push button S1 on the Launchpad. Why does the green LED not turn on?
The green LED does not turn on because of the code stops at the breakpoint where the output pin is set to that LED.

Task 1.3: Interrupt Trace

In this task, you will record a trace of all interrupts that occur while the program is executed on the MSPEXP432P401R. The trace contains the source of the interrupts and the timestamps when the interrupts happened. As a sample program, we use the program of Task 1.1.

1) what are the sources of the interrupts?
The sources of interrupts are the buttons s1 and s2 which are hardware interrupts.
2) did the interrupts occur as you predicted based on your design?
Yes
3) do the timestamps of interrupts match with your designed pattern?
Yes
4) how this interrupt profiling experiment deepen your understanding about interrupts?
I am now able to better understand how code interacts with hardware to create a task or operation using interrupts.

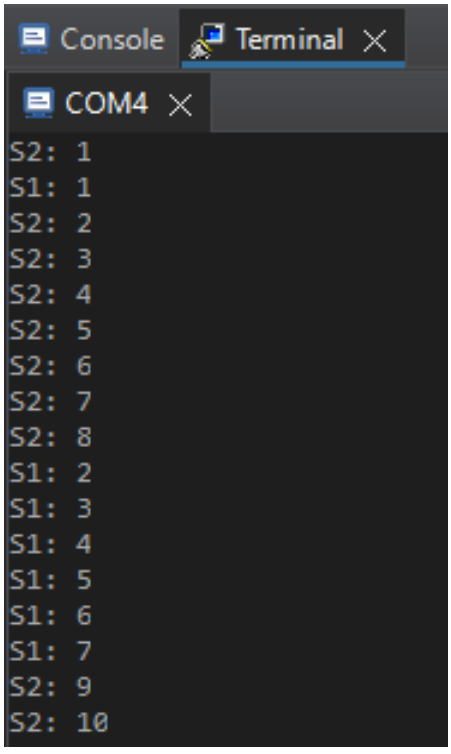
III Answers and Results

Task 1.1: Interrupts with Buttons

Changed Code

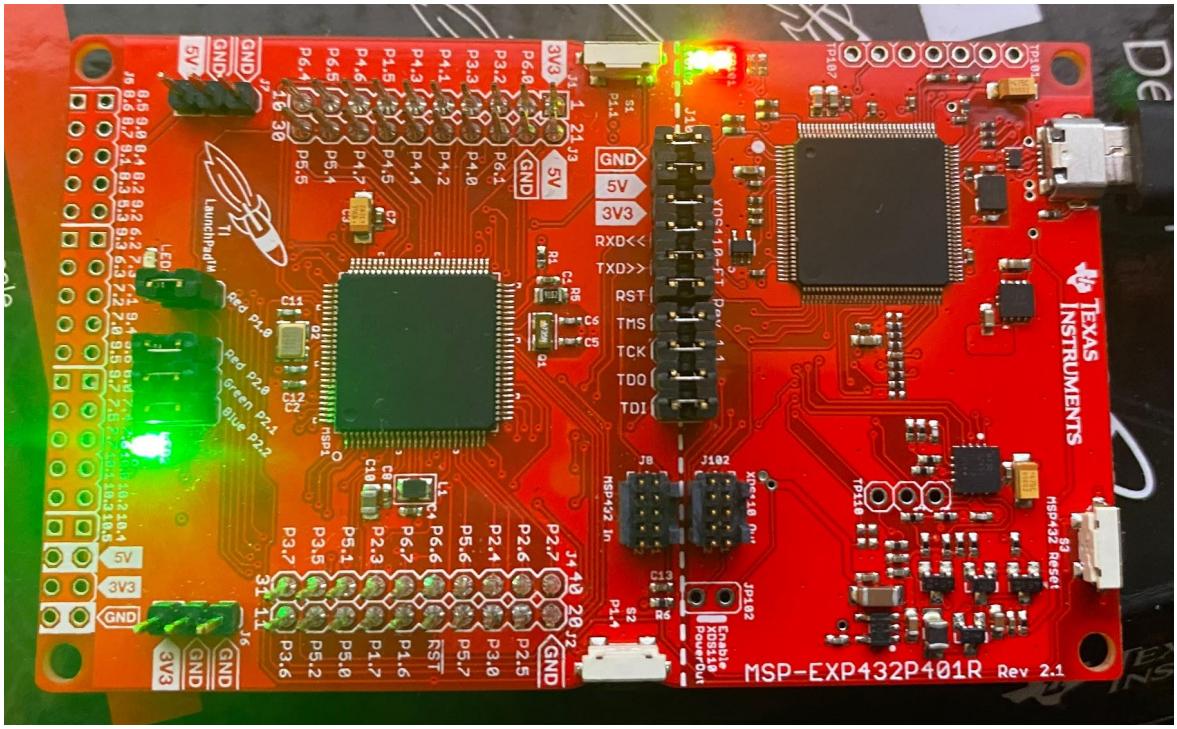
```
15  /* Setup UART */
16  uart_init(UART_BAUDRATE);
17
18  //// Placeholder 1 ////////////////////////////////////////////////////////
19    /* Configuring P2.1 (LED2 green) and P2.2 (LED2 blue) as output */
20    GPIO_setAsOutputPin (GPIO_PORT_P2, GPIO_PIN1);
21    GPIO_setAsOutputPin (GPIO_PORT_P2, GPIO_PIN2);
22    ////////////////////////////////////////////////////////////////////////////
23
24    //// Placeholder 2 ////////////////////////////////////////////////////////
25    /* Configuring P1.1 (Button S1) and P1.4 (Button S2) as an input */
26    GPIO_setAsInputPinWithPullUpResistor (GPIO_PORT_P1, GPIO_PIN1);
27    GPIO_setAsInputPinWithPullUpResistor (GPIO_PORT_P1, GPIO_PIN4);
28    ////////////////////////////////////////////////////////////////////////////
29
30    //// Placeholder 3 ////////////////////////////////////////////////////////
31    /* Configure interrupts for buttons S1 and S2 */
32    GPIO_interruptEdgeSelect (GPIO_PORT_P1, GPIO_PIN1, GPIO_LOW_TO_HIGH_TRANSITION);
33    GPIO_clearInterruptFlag (GPIO_PORT_P1, GPIO_PIN1);
34
35    GPIO_interruptEdgeSelect (GPIO_PORT_P1 , GPIO_PIN4, GPIO_LOW_TO_HIGH_TRANSITION);
36    GPIO_clearInterruptFlag (GPIO_PORT_P1, GPIO_PIN4);
37    ////////////////////////////////////////////////////////////////////////////
38
39    //// Placeholder 4 ////////////////////////////////////////////////////////
40    /* Enable interrupts for both GPIO pins */
41    GPIO_enableInterrupt (GPIO_PORT_P1, GPIO_PIN1);
42    GPIO_enableInterrupt (GPIO_PORT_P1, GPIO_PIN4);
43    /* Enable interrupts on Port 1 */
44    Interrupt_enableInterrupt (INT_PORT1);
45    /* Enable interrupts globally */
46    Interrupt_enableMaster();
47    ////////////////////////////////////////////////////////////////////////////
48
49    //// Placeholder 5 ////////////////////////////////////////////////////////
50    /* Initially turn LEDs off */
51    GPIO_setOutputLowOnPin (GPIO_PORT_P2, GPIO_PIN1);
52    GPIO_setOutputLowOnPin (GPIO_PORT_P2, GPIO_PIN2);
53    ////////////////////////////////////////////////////////////////////////////
```

```
85  /* Get the content of the interrupt status register of port 1 */
86  do {
87  status = GPIO_getEnabledInterruptStatus(GPIO_PORT_P1);
88
89  //// Pseudocode 1 (Task 1.1) ////////////////////////////////////////////
90  //
91   if (status & GPIO_PIN1) {
92       GPIO_toggleOutputOnPin(GPIO_PORT_P2, GPIO_PIN1);
93       button1Flag = true;
94       GPIO_clearInterruptFlag(GPIO_PORT_P1, GPIO_PIN1);
95
96   }
97   else if (status & GPIO_PIN4) {
98       GPIO_toggleOutputOnPin(GPIO_PORT_P2, GPIO_PIN2);
99
100      button2Flag = true;
101      GPIO_clearInterruptFlag(GPIO_PORT_P1, GPIO_PIN4);
102  }
103 //
104 // Hint: Use the provided status variable and the bit masks GPIO_PIN1 and
105 //       GPIO_PIN4 to figure out if button S1 or S2 have been pressed.
106 ////////////////////////////////////////////////////////////////////////////
107 } while(status);
108 }
```

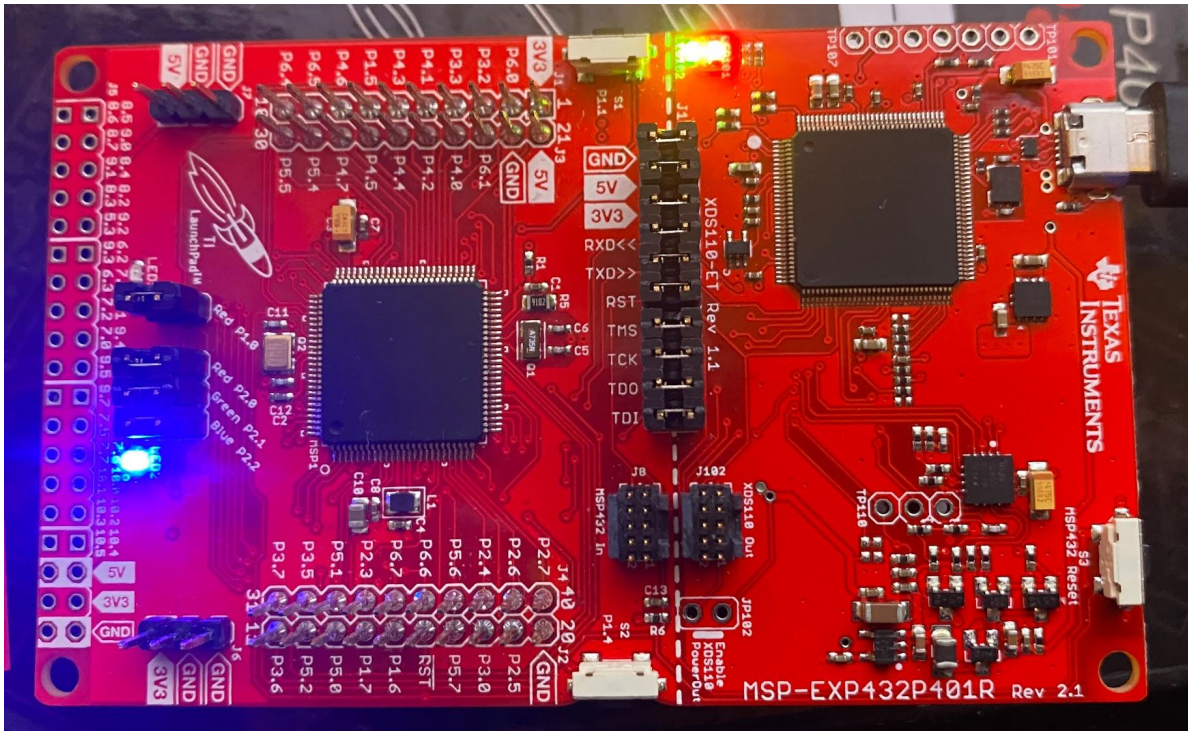


Explanation: The green is toggled on the RGB LED if button s1 is toggled and the blue is toggled if button s2 is toggled.

Board when s1 is toggled

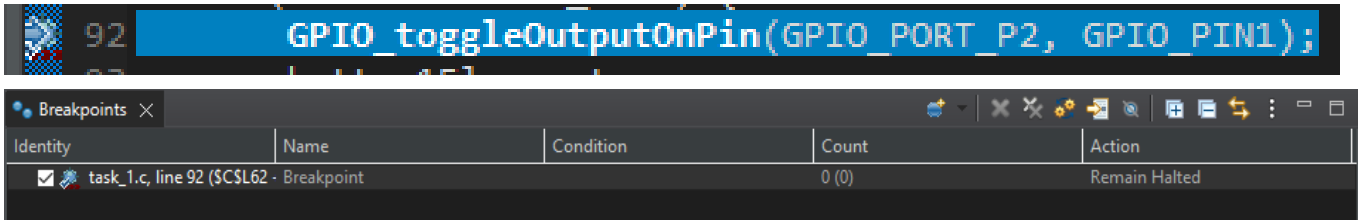


Board when s2 is toggled



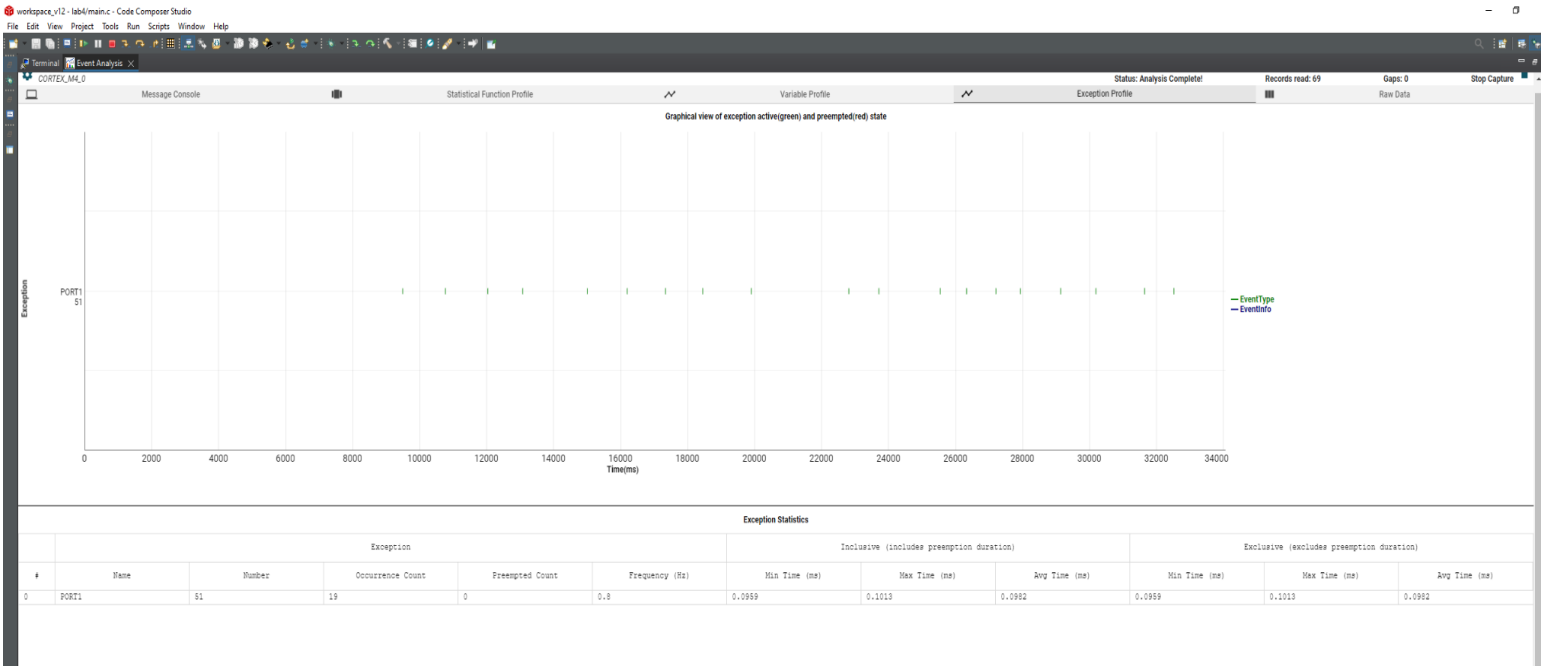
Task 1.2: Breakpoints and Stepping Through Code

Breakpoint



Task 1.3: Interrupt Trace

Event Analyis (Exception Profile)



IV Problem

The first issue I had was in task 1.1 because I kept getting an error about target configuration when trying to compile. I solved this issue by moving lab 4 into a different file and opening it from there. The second issue I had was trying to find the tools menu bar but I resolved this by switching from simple mode to default mode.

V What have you learned

During this lab, I learned about polling and how to toggle an LED using the s1 and s2 buttons. I also larned about breakpoints when debugging and how to make a graph of the event analysis when running the program.