Educational Objective:

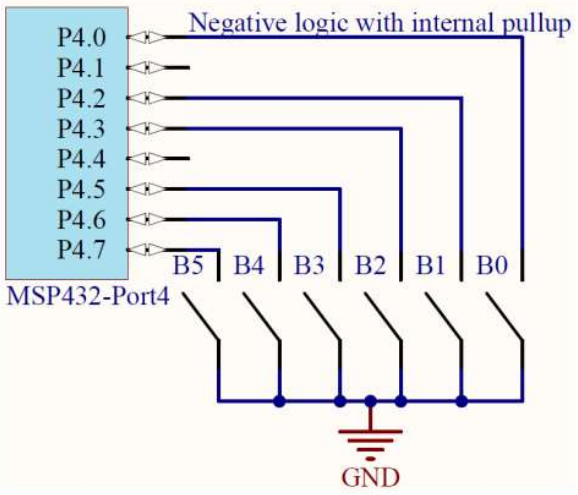
The educational objective of this laboratory assignment is to interface bump sensors and detect a robot collision. You will learn about edge triggered interrupts, global variables and externs. Additionally, you will use your knowledge of pulse width modulated waveforms to control the robot's motors.

Background:

Edge triggered interrupts are a useful feature of microcontrollers that are commonly used in embedded systems. In general, external events can be signified as a change in status. Examples generally include push buttons, power failures, temperature overload, and system faults. If the change in status is managed with an external digital logic signal, it can be connected to a GPIO input and the system can detect an interrupt on a rising or falling edge. This lab will explore using interrupts in conjunction with the bump sensors to determine if the robot has had a collision.

A collision event will cause an interrupt, and reading the status of the bumper switches should occur in an **interrupt service routine (ISR).**

Only ports 1 – 6 can trigger interrupts using the MSP432. The bump sensors are connected on the RLSK robot through port 4 as shown below. Recall that the push buttons use negative logic and an internal pullup.



When observing the robot from the front, the left most bump sensor is B0, making the right most sensor B5.

Review the lecture notes on interrupts and interrupt handlers before completing this lab.

Pre-Laboratory:

1. Read and understand the requirements for this lab.
2. Create a new project in Code Composer and start a new main.c.
3. Pull down the Motor.c support file from myCourses and copy all of your motor functions from Lab 4 into this file. Once you have done that, copy that file into the inc project in your CCS workspace. This file should include the motor control functions that you created using TimerAs, including your TimerInit() and MotorInit() functions. .
4. This lab requires your robot to change direction after colliding with a barrier. Draw a state diagram that performs the following tasks.
   a. Continue forward until collision is detect.
   b. Once collision is detected, go backward for about a half of a second.
   c. Turn the direction opposite of the detected bump. If the collision was a right side collision, turn left. If both right and left detected you may choose a default.
   d. Continue forward until a new collision is detected.
5. Submit your Motor.c file and state machine diagram to the dropbox in MyCourses prior to your lab section. Also, put your code on a flash drive if you do not have a laptop to bring to lab.

Procedure:

1. Pull down the BumpInt.c file from myCourses provided by TI. You will be creating your bumper support functions here.
   a. Write a function **BumpInt_Init()** to initialize the bump sensors. Set the port pins, enable internal pull resisters, configure for a pullup, and enable edge triggered interrupts.
   b. Write an IRQ handler for port 4. This is done by creating a function with the name **PORT4_IRQHandler().**
      i. In this handler you must maintain a count. Assign the increment or decrement values for each button as shown below.

Button Pressed	Value
B0	+3
B1	+2
B2	+1
B3	-1
B4	-2
B5	-3

      ii. Use the P4IV register to determine which button was pressed.
      iii. Don't forget to clear the interrupt flag before leaving the IRQ handler.

2. Test your design and verify that each button press works properly by counting from 0 to 6, then back down to 0. Do this using a breakpoint in the IRQ handler and hovering over the count variable you are modifying.
3. **Demonstrate your design to the Professor or TA for a signoff.**
    a. Must run in debug mode and count the following sequence.
    b. 0 -> 3 -> 5 -> 6 -> 5 -> 3 -> 0
4. Once you have confirmed your interrupt handler is working properly code the state machine you designed in the prelab.
    a. Remember to include an isNewState variable so you are not continually updating the PWM pulse width for the motors.
    b. Use the printf() function to help determine your current state and if a collision was detected. This will help with debugging.
    c. Don't forget a default case.
5. Test your design by placing an obstruction in front of your robot. **Once the robot is performing as expected receive a signoff.**

Post Lab:

Take the Post lab quiz in MyCourses before your next lab section.

Signoffs and Grade:

Name:_____

Component	Signoff	Date
IRQ Detection – Counts 0 to 6 to 0		
Direction Changed After Collision		

==========

Component	Received	Possible
Prelab		20
Signoffs		60
Post-lab Quiz		20
Penalties • after the start of lab session -10pts • after 1 week late -25 pts • no signoffs after 2 weeks late	-	
Total		100