

San José State University
Department of Computer Engineering
CMPE 146-03, Real-Time Embedded System Co-Design, Fall 2020

Lab Assignment 5

Due date: 10/25/2020, Sunday

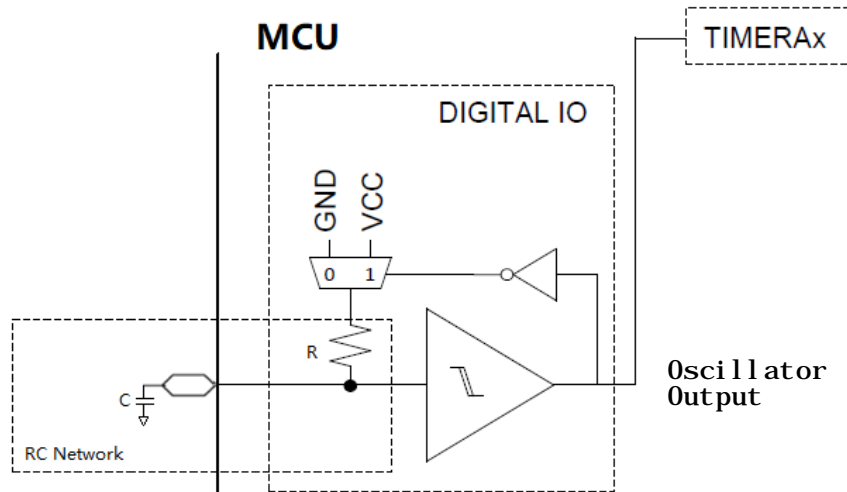
1. Description

In this assignment, we will be familiarized with the GPIO's capacitive-touch capability and the Timer_A module on the MSP432 MCU. We will use the capacitive-touch capability as an input method to control an LED.

2. Exercise 1

In this exercise, we will create an oscillator at a GPIO port and observe its behavior.

A GPIO port can be configured to form an oscillator. A RC network is used. The resistance mainly comes from the built-in pull-up-pull-down resistor in the port. The capacitance does not come from a specific capacitor. Instead, it is a collection of “hidden” capacitance from the MCU itself, connecting wires, circuit board, etc.



To form an oscillator on the MCU, we'll need to enable the capacitive-touch feature at the GPIO port. For example, to create an oscillator at Port 4, Pin 1 (P4.1), we can execute the following statements to set up the control register of the feature:

```
CAPTI00CTL = 0;
CAPTI00CTL |= (1 << 8); // Enable CAPTIO
CAPTI00CTL |= 0b0100 << 4; // Choose Port 4
CAPTI00CTL |= 0b0001 << 1; // Choose Pin 1
```

To read the state of the oscillator's output, we can read the control register and do something like this:

```
bool state = CAPTIOCTL & 0x200
```

Details of the capacitive touch control register can be found in Chapter 14 of the MCU technical reference.

Create a small program (duplicate from the *empty* example project) to continually show the state of the oscillator. To see a stream of binary outputs, we can use `printf()` without a newline. Make sure to add a `fflush(stdout)` call after `printf()` so that the output can be seen immediately. In the loop that continually shows the state, add a small delay using the delay function you wrote in the previous lab so that the printing is not too fast. Use a delay of 10 ms. We should see a stream of random 1's and 0's on the debug console.

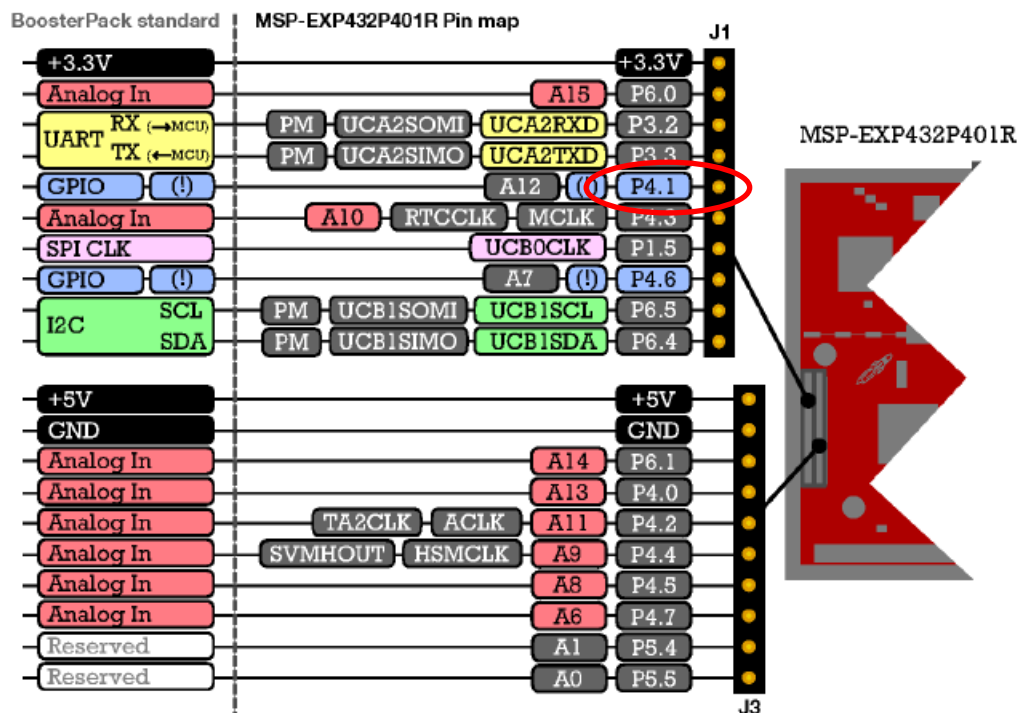
Lab Report Submission

1. Show the `main()` function.
2. Show the debug console outputs.
3. Provide the program listing in the appendix.

3. Exercise 2

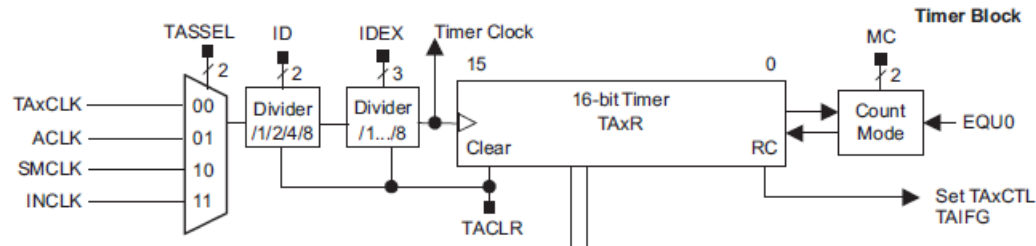
In this exercise, we will use Timer_A to measure the frequency of the oscillator we created in the previous exercise. We will change the frequency by touching the GPIO pin with our finger.

There are many GPIO pins we can get access to on the LaunchPad's connectors. For example, Port 4, Pin 1 is connected to Connector J1.



The P4.1 oscillator output can be routed to the A2 counter/timer in the Timer_A module. We will use the 16-bit counter to sense the change of the capacitance at the GPIO pin. The P4.1 oscillator provides a clock signal to the counter. We measure how many clock pulses it receives in a specific time window. If the count changes significantly, it is most likely caused by a change in the capacitance at the pin induced by a human's finger touch on the connector pin.

Each counter in the Timer_A module accepts one of the four input clock signals. The P4.1 oscillator output is routed to the module as the INCLK signal.



We can use the Timer_A DriverLib functions to set up the counter to count the incoming pulses. The following statements configure the counter to count in continuous mode and start up the counter to count.

```
Timer_A_ContinuousModeConfig timer_continuous_obj;
timer_continuous_obj.clockSource = TIMER_A_CLOCKSOURCE_INVERTED_EXTERNAL_TXCLK;
timer_continuous_obj.clockSourceDivider = TIMER_A_CLOCKSOURCE_DIVIDER_1;
timer_continuous_obj.timerInterruptEnable_TAIE = TIMER_A_TAIE_INTERRUPT_DISABLE;
timer_continuous_obj.timerClear = TIMER_A_DO_CLEAR;
MAP_Timer_A_configureContinuousMode(TIMER_A2_BASE, &timer_continuous_obj);
MAP_Timer_A_startCounter(TIMER_A2_BASE, TIMER_A_CONTINUOUS_MODE);
```

Note that `TIMER_A_CLOCKSOURCE_INVERTED_EXTERNAL_TXCLK` refers to the INCLK input clock in the schematic. We set the clock divider to divide by 1, essentially not dividing because we want to be able to measure with high precision. We do not use interrupt. We clear the counter and disable counting when we configure it.

Duplicate the program in the previous exercise. Add the necessary code to see how many counter ticks will be recorded in 2 ms. Basically, before the measurement, you clear the counter with a DriverLib function `MAP_Timer_A_clearTimer(TIMER_A2_BASE)` so that it counts up from 0 again. Delay for 2 ms. Then read the counter register to get the count by calling `MAP_Timer_A_getCounterValue(TIMER_A2_BASE)`.

Print the counter value to the debug console. Convert it to frequency (in Hz) and print it to the debug console. The value tells us the current frequency of the oscillator.

In the program, repeat the measurement in a forever loop. After one measurement, delay 500 ms before making another one so that the display is not too rapid. Touch the J1 connector where P4.1 is connected to. Observe how the counter value and frequency change. What would you do to change the frequency slightly and drastically?

Lab Report Submission

1. Show the relevant code to do the measurement and computation of frequency. Explain how the frequency is computed from the counter value.
2. Show the debug console outputs. Indicate where the touch and no-touch outputs are.
3. Report the frequencies computed when the connector is touched or not.
4. Explain the method you use to make the frequency change drastically.
5. Provide the program listing in the appendix.

4. Exercise 3

In this exercise, we will use the capacitive touch feature to turn on and off an LED.

Based on what you have done in the previous exercise, write a program to control the red LED (LED1 on the LaunchPad). When you touch the J1 connector where P4.1 connected to, the program turns on the LED; otherwise, it turns it off.

You can make the control decision based on the frequency of the oscillator that the program constantly measure. For example, it turns off the LED if the frequency is above some threshold. Whatever you do, you cannot hard-code any parameter in the program for the control purpose. You can do some kind of calibration when the program starts up, before the control loop is entered. You can safely assume that there is no human touch on the J1 connector during calibration. Once you have determined the parameter(s) in the program, print it (or them) to the debug console.

In the control loop, print the measurement results, just like what was done in the previous exercise.

Lab Report Submission

1. Show the relevant code to determine the control parameter(s) and the control of the LED. Explain how they are done.
2. Show the debug console outputs. Display the control parameter(s). Indicate where the touch and no-touch outputs are.
3. Provide the program listing in the appendix.

5. Exercise 4

In this exercise, we will look at the oscillation at another port.

Duplicate the project in Exercise 2. Instead of using P4.1, use P3.4. Print the oscillation frequency to the debug console. We are only interested in the no-touch frequency here. Compare the frequencies at the two different ports. Describe the differences (if any) and provide an explanation. You may want to consult the LaunchPad's schematic for a clue.




Lab Report Submission

1. Show relevant code to set up to for P3.4.
2. Show the debug console outputs.
3. Provide an explanation with supporting evidence or data on your observation regarding the frequency differences (if any) between the two ports.
4. Provide the program listing in the appendix.

6. Submission Requirements

Write a formal report that contains at least what are required to be submitted in the exercises. Submit the report in PDF to Canvas by the deadline. Include source code in your report. As for the report contents, do not use screenshots to show your codes. In the report body, list the relevant codes or screenshots only for the exercise you are discussing. Put the entire program listing in the appendix. Use single-column format.

7. Grading

Lab 5 Rubric									
Criteria	Ratings						Pts		
Exercise 1	2 pts 2 Correct implementation. Program listing in appendix.		1 pts 1 Showed main(). Showed bit stream on debug console.		0 pts 0 Not attempted or reported.		2 pts		
Exercise 2	3 pts 3 Correct implementation. Described and explained method to change the oscillator frequency. Program listing in appendix.		2 pts 2 Reported reasonable touch and no-touch frequencies. Correct method of measurements. Provided explanations on methods used.		1 pts 1 Showed relevant code. Showed outputs on debug console with touch and no-touch indications.		0 pts 0 Not attempted or reported.		3 pts
Exercise 3	3 pts 3 Correct implementation. Program listing in appendix.	2 pts 2 Proper calibration method used. Provided explanations on method used.		1 pts 1 Showed relevant code. Showed outputs on debug console with control parameters and measurements.		0 pts 0 Not attempted or reported.		3 pts	
Exercise 4	2 pts 2 Correct implementation. Provided reasonable observation and explanation with supporting evidence or data. Program listing in appendix.			1 pts 1 Showed relevant code and outputs on debug console. Provided observation and explanation.		0 pts 0 Not attempted or reported.		2 pts	
Total Points: 10									

Grading Policy

The lab's grade is determined by the report. If you miss the submission deadline for the report, you will get zero point.