EE 444

Embedded Systems Design

Lab 1: *MSP430: Clocks and Digital I/O*

*2/07/2017*

**Chic J O’Dell**

Does your solution work the way it’s supposed to work? □YES □NO1

1If your answer is NO, please explain in your report.

Instructor/TA comments and grading

# Objective and Background

Become familiar with the TI MSP-EXP430F5438A experimenter’s board, CrossStudio, and the process of downloading, executing, and debugging programs on MSP430-based boards. In addition, I will be using Digital IO capabilities of MSP430, timers, and interrupts to measure time intervals.

# Equipment

* Crossworks
* TI MSP-EXP430F5438A board
* Agilent mixed-signal oscilloscope

# Procedure

## Part 1: PMM

In order to use MCLK at 25 MHz it is necessary to modify the MSP430 core voltage as seen in the data sheet Figure 3. This was accomplished by using the provided IncrementVcore() and DecrementVcore() functions. After setting the core voltage to the highest level “3” UCSCTL2,1, and 4 are set to allow for 25 MHz clock pulse as seen in Figure 4. UCSCTL2 sets the frequency multiplication factor necessary to have a 25 MHz output. UCSCTL1 sets the DCO tap to a setting that will support the desired frequency (25 MHz) chosen from pg 26 of the MSP430F5438A datasheet. UCSCTL4 sets the output of the FLLREFCLK to DCOCLKDIV to the input of MCLK so we can see our signal on an MCLK test point.

## Part 2: Button

In order to write a program to measure the time between two button presses it was necessary to use timer interrupts (Timer\_A) as well as the port two interrupt to catch the time and button push respectively, as seen in Figure 2. Using the provided switch on the TI MSP\_EXP430F5438 breakout board connected to P2.6 to throw an interrupt when pushed. Inside the P2 interrupt routine the TA0CCTL0 CCIS0 interrupt is manually thrown by toggling CCIS0 to GND or VCC. This causes TA0CCTL0 to store the current time of the button push. The current time is then stored in a global variable which can be seen in the debuggers “global’s” window, as seen in INSERT FIGURE. Also I have set the input of TA0CCTL0 to be fed from ACLK which I have divided in order to provide more time for me to hit the button twice before overrunning the timer buffer.

* *Provide a short description of how you did what you did in this lab, especially if there were different ways of doing it.*
* *Don’t copy the lab assignment’s procedure. Instead, give a summary of the steps you took to design and implement what you created in lab, in your own words.*
* *Make sure you mention all the wonderful things you did that were not specifically mentioned in the assignment.*

# Results

## Part 1: PMM

Line 21 to 28 in code, Figure 1 I have added the IncrementVcore() and DecrementVcore() function calls such that I can run the Crossworks Debugger and check the voltage levels at each PMMCOREV setting 0 through 3, seen in Table 1. These values were probed using a DMM and an open test point on the MSP breakout board.

Table 1: PMMCOREV measurements

|  |  |  |
| --- | --- | --- |
| VCORE Voltage | | |
| #increments | Observed | Expected |
| 3 | 1.934 | 1.85 |
| 2 | 1.823 | 1.75 |
| 1 | 1.619 | 1.55 |
| 0 | 1.416 | 1.35 |

Throughout all PMMCORE settings my clock produced 25MHz though the data sheet clearly stated it should not be able to do so. Is this possibly because I was not doing any “heavy lifting” with the MSP?

## Part 2: Button

Using the Crossworks debugger to capture changes in the global variable “time” I was able to calculate a button push using Eq (1) where time1 and time2 are the timer interrupt count values captured from TA0CCTL0 upon the first and second button pushes and freq is ACLK = 1024 Hz (32768 divided by 32).

Eq(1)

# 6. Conclusion

* *What was the point of this lab?*
* *What did you learn from this lab?*

# 7. Attachments

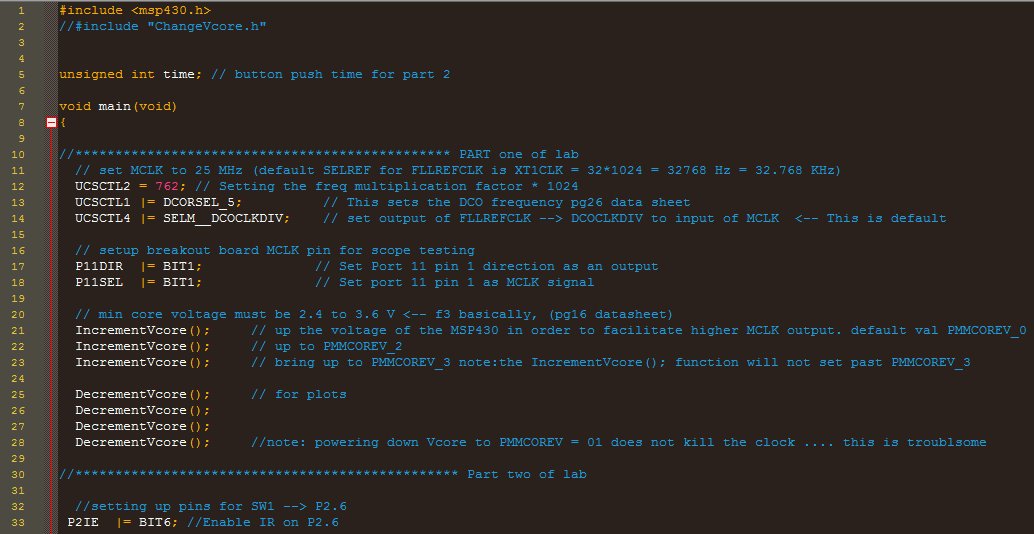


Figure 1: Lab 2 PMM code (pt1/2).

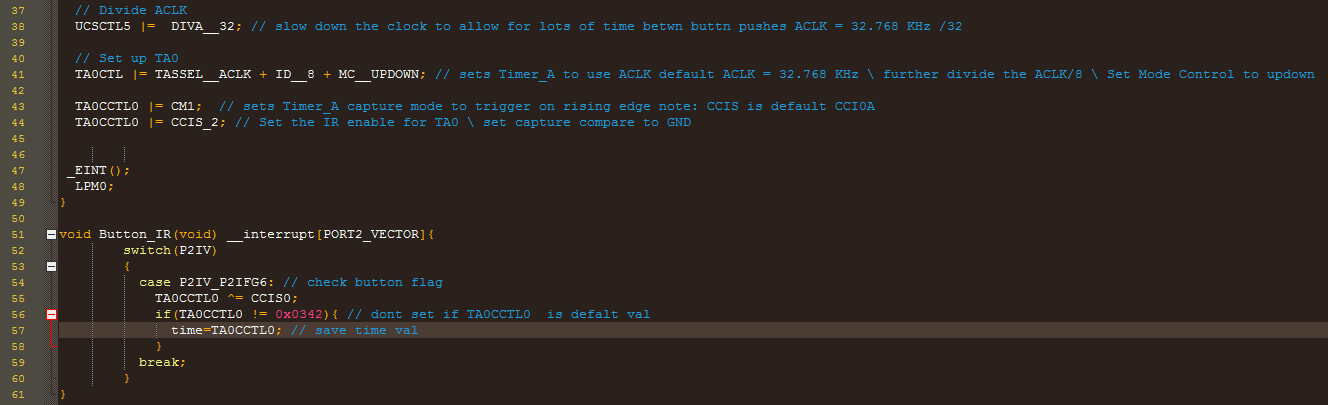


Figure 2:Lab 2 button code (pt2/2).

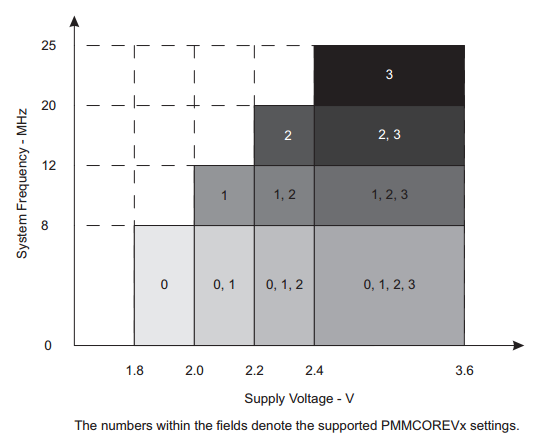


Figure 3: Frequency vs Supply Voltage.

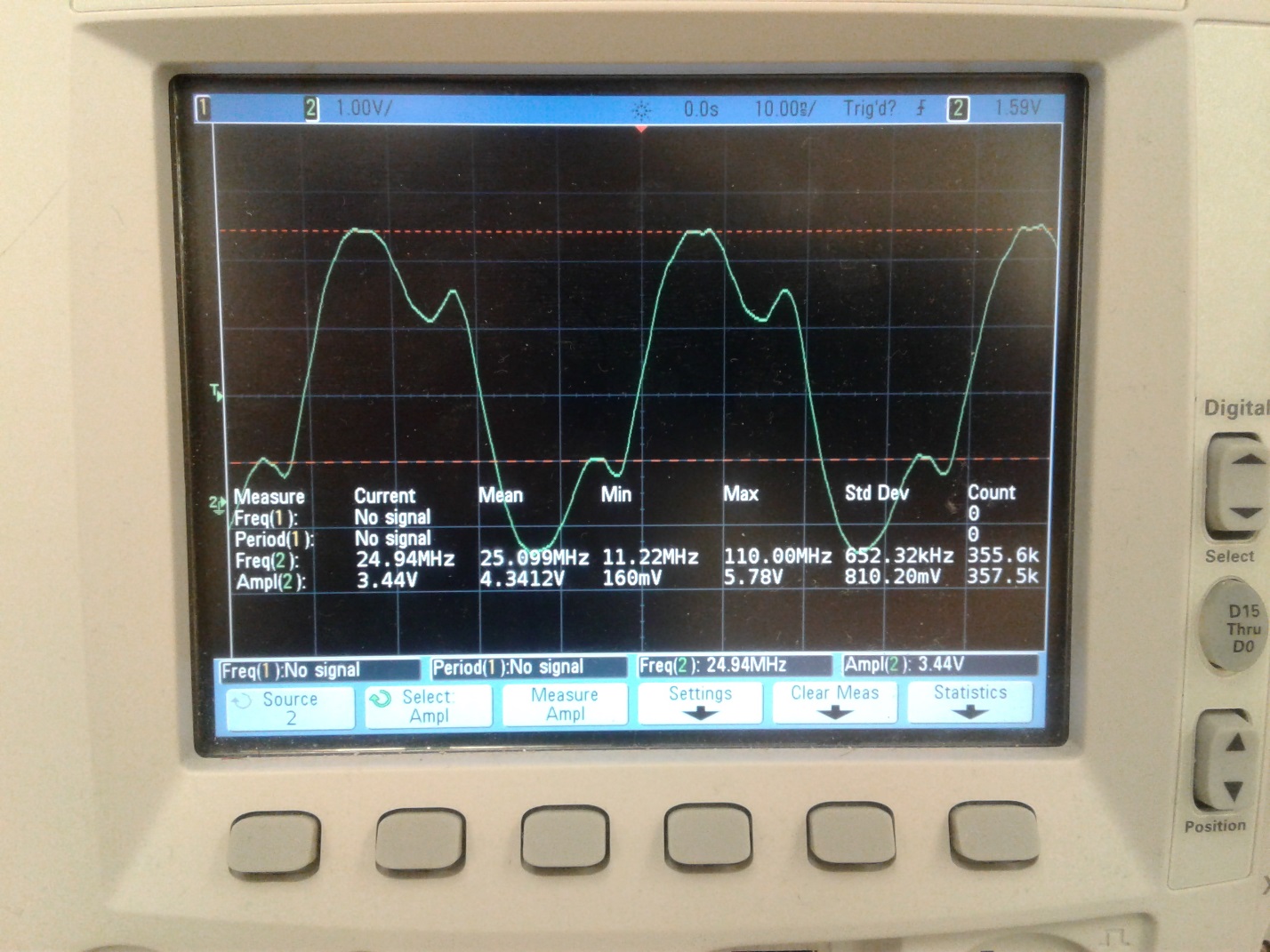


Figure 4: Part 1 25 MHz MCLK output.