EGR: 226 Microcontroller Programming and Applications

Winter 2021

Instructor Prof. Trevor Ekin

**Lab 3: Introduction to MSP432**

Gabriel Gasbarre

February 10th, 2021

[1. Objectives 3](#_Toc63868928)

[2. Equipment 3](#_Toc63868929)

[3. Introduction 3](#_Toc63868930)

[3.1: Part 1 – Exploring the capabilities of the MSP432 3](#_Toc63868931)

[3.2: Part 2- Using the Code Composer Studio to compile, download, and debug an application to run on the MSP432. 4](#_Toc63868932)

[3.3 Part 3 – Creating your own program two flash the red LED 4](#_Toc63868933)

[4. Procedure 4](#_Toc63868934)

[4.1: Part 1 - Exploring the capabilities of the MSP432 4](#_Toc63868935)

[4.2: Part 2- Using CCS 5](#_Toc63868936)

[4.3: Part 3- Creating your own program to flash the LED 5](#_Toc63868937)

[5. Results 6](#_Toc63868938)

[5.1 Part 1 Results 6](#_Toc63868939)

[5.2: Part 2 Results 6](#_Toc63868940)

[5.3: Part 3 Results 6](#_Toc63868941)

[6. Conclusions. 7](#_Toc63868942)

# Objectives

There were 4 main goals of Lab exercise 3. First off, was to become familiar with Code Composer Studio. This is the basic developmental system for creating and loading programs to run on the MSP432. Next, was to install and run firmware on the MSP432 that interfaces to external output LED’s on the MSP432. Learning how to effectively set and watch breakpoints for debugging was a third main component of the lab goals, with Graphical User Interface exploration between a PC and the MASP432 being the last.

# Equipment

|  |  |  |
| --- | --- | --- |
| **Part** | **Description** | **Model** |
| CCS (Code Composer Studio) | Integrated development environment to develop applications for Texas Instruments embedded processors. | 10.0.00010 |
| MSP432 | Mixed-signal microcontroller family from Texas Instruments. | MSP432P401x |
| EGR:226 Lab 3 Exercise | Introduction to the MSP432 | N/A |

# Introduction

## 3.1: Part 1 – Exploring the capabilities of the MSP432

For part 1, students were asked to investigate the pre-programmed demo that comes with the MSP432. This demo allows the user to alter the onboard LED through pushbutton switches and a GUI though CCS. The GUI allows for color, blink rate, and light intensity of the on-board LED to be manipulated.

## 3.2: Part 2- Using the Code Composer Studio to compile, download, and debug an application to run on the MSP432.

For part 2, students were asked to create a new CCS project that will blink the on-board red LED using a pre-written program provided in the lab document. This should also teach students how to compile, debug, and load programs onto the MSP432.

## 3.3 Part 3 – Creating your own program two flash the red LED

For part 3, students were asked to build upon part 2 in order to create a project that will ultimately show a student’s ability to alter a variable within an application without rebuilding the program. This will be done by altering the delay variable of a function.

# Procedure

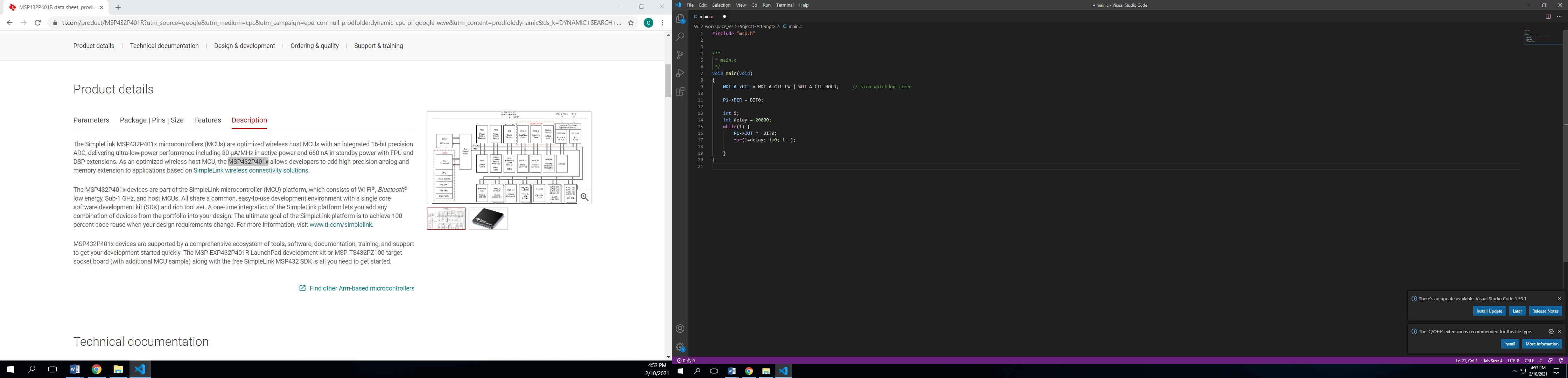
## 4.1: Part 1 - Exploring the capabilities of the MSP432

Part 1 was the simpler half of the lab experiment. First, we plug in the MSP432 and install the drivers. Next, we launch CCS and navigate to the Out Of Box Experience GUI. This GUI allows for remote control of the LED color and blink rate from the connected PC via USB. After some toying around with the MSP432, students were to investigate the source code of the program. This will be interesting to return to when we inevitably start creating our own functions and programs for controlling LED’s.

## 4.2: Part 2- Using CCS

Part 2 invites students to create their first project using CCS. First, we create a new project with the MSP432P401R as the target. Next, we configure an output pin on the MCU and interface to an LED by copying the provided code in the lab document. This code toggles a bit between every 20000 cycles of a ”for” loop. Next, we build and debug the program, then download it to the MSP432. Testing the program is as easy as selecting ‘resume’ from the Run pull-down menu and watching your program get to work.

## 4.3: Part 3- Creating your own program to flash the LED

 Part 3 allows students to take what they have learned from the previous parts and combine it in order to create a program that will modify the delay by a variable amount. For myself, this was as simple as altering the “for” loop to instead of repeating every 20,000 cycles, to repeat by a number of cycles dictated by the variable “delay.” This in turn would force the program to be occupied for a longer/shorter amount of time deepening on the number of repetitions, ultimately making the on board LED blink faster / slower.

Above is a snippet of the code used that allowed for the altercation of the delay variable, and ultimately the increase/ decrease of number of repetitions for the ‘for’ loop.

# Results

## Part 1 Results

This part of the lab was not super significant in the knowledge it brought of how to use the MSP432, however it was interesting and significant in displaying the power of the MSP432. I also learned here that certain color combinations also alter the intensity of the light, which may be useful for future programs.

## 5.2: Part 2 Results

Part 2 of the lab was very exciting as it was my first opportunity to personally alter the MSP432. Here I learned about how the clocks and delay functions of the MSP432 are not entirely necessary, however they are almost surely more reliable. This is because when occupying the MSP432 with a loop of any kind, it cannot progress further into the program without completing every repetition. Although this may seem like a reliable way to make a clock or delay for the MSP, it is unreliable as it relies entirely upon the processing power and compiling of the program. If the same program were to be loaded onto an even faster microcontroller, the delay could be half the time. In the opposite circumstance, it could even take twice as long. Still, it is an interesting way to introduce delay into our functions.

## 5.3: Part 3 Results

Part 3 of the program was straightforward, however it gave me some trouble along with a couple learning opportunities. After setting ‘i,’ the increment variable for the ‘for’ loop equal to my delay value, I thought I had figured out how to vary the program without rebuilding by modifying the delay value within the “variables” tab of CCS while the program was paused. Though this was in fact the case and the correct way to do it, I found that often when changing the delay variable I did not hit the ‘enter’ key afterwards and it did not update the value within the program. A silly mistake but it proves the importance of knowing exactly what you are doing because even if you have the right idea, the execution may simply just not be there.

# Conclusions.

I personally feel like I took quite a bit away from this lab in the sense that it provides great example of the power of the MSP432. Running through over 20,000 repetitions of a ‘for’ loop in under a second is quite impressive. I do, however wish I had spent a little more time looking inside of the source code for the out of box experience demo, as im sure it has many useful functions that I could learn from. It will almost surely be something that I come back to.