**EGR 226: Microcontroller Programming and Applications**

**Winter 2021**

Instructor: Professor Trevor Ekins

Lab 3: Introduction to MSP432

Sam Wieneke

February 10, 2021

**Contents**

1. **Objectives**
2. **Equipment**
3. **Introduction**
4. **Procedure**
5. **Results/ Discussion**
6. **Conclusion and Future Work**

**Appendices**

1. **Source Code: Part 2**

**Source Code: Part 3**

1. **Objectives**

The objectives of this laboratory are to become familiar with Code Composer, to install and run firmware on the MSP432 that interfaces to external I/O devices, to learn how to set breakpoints and watch memory locations on the MCU using debugging features of CCS, and to explore a GUI between a PC connected to a MSP432.

1. **Equipment**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Part | Description | Model | Measured Value | Notes |
| Code Composer Studios | Texas Instruments programming environment | Version 9.3.0 | N/A | N/A |
| GitLab | Remote Repository for code maintenance | N/A | N/A | Makes collaboration on team projects and code very convenient. |
|  |  |  |  |  |

1. **Introduction**

**Part 1: Out of Box Demo**

Part 1 involved unboxing the MSP432 and opening a demo that is already programmed into the MSP432. It allowed for the user to change the color and frequency of the flashing light.

**Part 2: Debugging on MCU**

Part 2 involved creating a new CCS project and actually writing some code in main.c. The code was already given, it just needed to be transferred into CCS. The code was designed to flash the LED every 20000 cycles and would do until the program was stopped.

**Part 3: LED flash**

Part 3 involved taking the code from part 2 and modifying it to be able to control the rate of the flashing LED without it being hardcoded as in part 2.

1. **Procedure**

**Part 1**

This part of the lab required plugging in the MSP432 into the computer to start and then finding the out of box demo to experiment with changing the colors and frequency of the LED. There was not much to this part, it just required following the steps to find the demo and running the demo with the MSP432 being plugged in correctly.

**Part 2**

In this section of the lab, a new CCS project was opened and the code that was given was typed into main.c. The code was then built and ran and if you changed the hardcoded timer, it would change the frequency of the LED. The powerpoint slides were then opened and the tasks within them were completed. These tasks expanded the knowledge of the debugging process which was explored thoroughly by following the powerpoint.

**Part 3**

Part 3 was very similar to part 2, with a few minor changes. A new CCS project was created and the code from part 2 was copied over. For this code, it had to be changed so that the timer could be changed while it was running, not hardcoded into the program. To do this, the timer had to be edited as shown in Figure 4.1.

Figure 4.1: Timer with variable

**while** (1) {

P1->OUT ^= BIT0; //toggle LED status

**for**(i=q; i>0; i--); //Crude delay

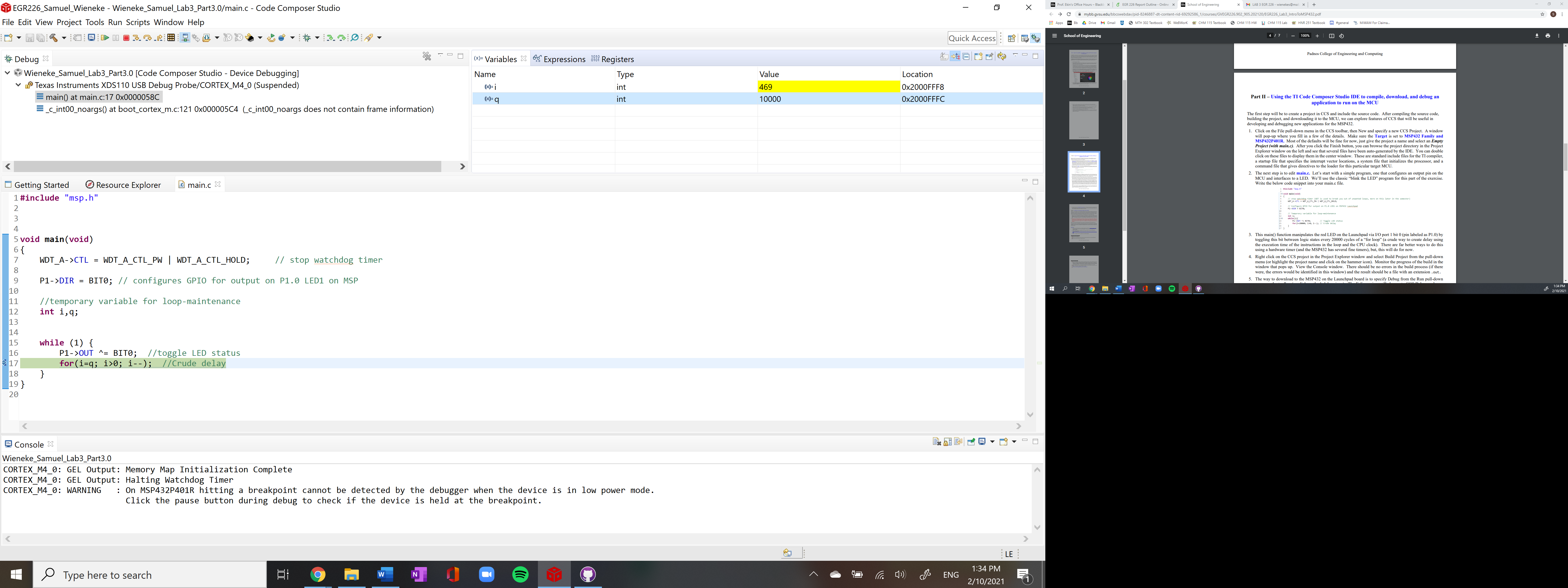
The “q” in the code was able to be changed during the program which would allow for changing the timer without rebuilding the program every time.

1. **Results/ Discussion**

**Code Display**

Figure 5.1 shows the window where the timer can be changed and the code functioning properly.

Figure 5.1: Code Running



In the top right corner of the screen, “q” can be changed to change the frequency of the LED flashing, without having to rebuild the program every time the frequency needs to be changed.

**Questions**

1. Why do you think TI decided to use an Eclipse foundation for CCS? Describe the major features of the Eclipse IDE platform. What other devices are supported by an Eclipse based IDE?

I think TI decided to use an Eclipse foundation for CCS because it has a good user interface and easy to navigate. An Eclipse IDE platform contains a basic workspace and an extensible plug-in system for customizing the environment. Many devices are supported by an Eclipse based IDE.

1. How is a software breakpoint established in Code Composer Studio? How does this differ from a hardware breakpoint?

A software breakpoint is established in CCS by double clicking the editor margin. A software breakpoint is different from a hardware breakpoint because hardware breakpoints have a maximum amount meaning there are only a certain amount for each microcontroller where as software breakpoints generally have no maximum amount.

1. What is a watchpoint? How does it differ from a standard breakpoint?

A watchpoint is a type of breakpoint that monitors activity at a memory assress. A standard breakpoint is an intentional stopping or pausing place in a program for debugging purposes.

1. **Conclusion/ Future Work**

This lab was the first interaction with the MSP432. It involved using a demo code to tinker around with the light color and frequency, it also involved taking a prewritten code by the instructor and changing the hardcoded value of the timer to change the frequency. The final part of the lab took the code from the second part and required adding another variable to be able to change the frequency, so it was not hardcoded.

This lab was not very challenging compared to previous labs. There was minor difficulty determining how to setup another variable, so the code does not need to be rebuilt but it was figured out fairly quickly. There was not any future work that could be implemented because the code was given and there was only a minor change needed to change the code from being hardcoded to a variable that can be changed while the code was running.

**Appendix A**

Part 2 Source Code:

**#include** "msp.h"

**void** **main**(**void**)

{

WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD; // stop watchdog timer

P1->DIR = BIT0; // configures GPIO for output on P1.0 LED1 on MSP

//temporary variable for loop-maintenance

**int** i;

**while** (1) {

P1->OUT ^= BIT0; //toggle LED status

**for**(i=10000; i>0; i--); //Crude delay

}

}

Part 3 Source Code:

**#include** "msp.h"

**void** **main**(**void**)

{

WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD; // stop watchdog timer

P1->DIR = BIT0; // configures GPIO for output on P1.0 LED1 on MSP

//temporary variable for loop-maintenance

**int** i,q;

**while** (1) {

P1->OUT ^= BIT0; //toggle LED status

**for**(i=q; i>0; i--); //Crude delay

}

}