**EGR 226: Microcontroller Programming and Applications**

**Winter 2021**

Instructor: Professor Trevor Ekins

Lab 4: Digital Inputs and Outputs

Sam Wieneke

February 17, 2021

**Contents**

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

**Appendices**

1. **Source Code: Part 1**

**Source Code: Part 2**

1. **Objectives**

The objectives from this lab were to develop a program for the MSP432 that interfaces with the pushbutton switch to control the sequence of lighting different colored LED’s, to develop a program for the MSP432 to generate time intervals for controlling the LED lighting, and to implement a programmed method to accommodate switch bounce.

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: Sequencing Colored LED’s using pushbutton**

Part 1 involved driving the LED’s on the MSP432 to change when the onboard push button was pressed. A debouncing function was needed for the button to be functional. The program was a loop and could go on forever if the user pleased and the lights will be in a sequential order.

**Part 2: Sequencing Colored LED’s using timer and pushbutton**

Part 2 involved controlling the LED sequence using a time delay. The program sequenced through the color as long as the button was pushed and when it was released, it would stop on the color it was on.

1. **Procedure**

**Part 1**

This code was created by making a large looping function that will run forever, and within that loop, another loop that runs when the button is pushed. There are if statements within this loop that will turn off a color and turn on another color, each color will increment a counter which will determine which if statement should be next. There is a debounce function that will eliminate any bouncing caused by pushing the button.

**Part 2**

Part 2 was fairly similar to part 1, but in this part of the lab the colors had to be changing by themselves when the button was pushed and would stop on the color if the button was released. A function delay cycles, was used to determine how long the light will stay on before moving to the next color. This function is shown in Figure 4.1.

Figure 4.1: Delay Cycles

\_delay\_cycles(2000000);

This function will keep the light on for 1 second.

This code was created by having on main looping function that ran while the button was pushed. Within this loop, there were if statements that determined what color would be displayed along with a delay, so the lights do not change instantly. There was also a counter which allowed for the changing of the lights and worked as a loop where it would go back to the first color when it reached a certain number.

1. **Results/ Discussion**

**Lab Results**

To demonstrate this lab, part 1 had to be shown to the instructor as well as part 2. For part 1, the button had to be pushed to change the color and it didn’t change on its own. For part 2, the lights should change for as long as the button was held and be able to stop on the color if the button was released.

**Prelab:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **I/O** | **Port Pin** | **Pin Direction** | **1?** | **0?** |
| **LED\_1** | **P1.0** | **0** | **On** | **Off** |
| **LED\_RED** | **P2.0** | **0** | **On** | **Off** |
| **LED\_GREEN** | **P2.1** | **0** | **On** | **Off** |
| **LED\_BLUE** | **P2.2** | **0** | **On** | **Off** |
| **Switch Button 1** | **P1.1** | **1** | **Pull Up Res.** | **Pull Down Res.** |
| **Switch Button 2** | **P1.4** | **1** | **Pull Up Res.** | **Pull Down Res.** |

1. **Conclusion/ Future Work**

This lab was really the first in-depth dive into the MSP432. Lab 3 was very basic and didn’t require much coding or thinking. A challenge faced was changing the debounce function to a higher time length because the time wasn’t long enough and the light would flash very fast and not how it was supposed to. This was solved by adding more time in the debounce function. Another problem faced was setting up the code for part 1. I originally had it setup as 4 while loops within the main while loop but then changed it to if else statements because code composer seemed to like that better and it made more sense with the application. Something that could’ve been done to make the lab more smoothly in the future would to setup a way to be able to change the brightness of the lights as well for more effect.

**Appendix A**

Part 1 Source Code:

**#include** "msp.h"

**int** **DebounceSwitch1**(**void**);

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

{

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

P1->SEL1 &= 0xFD; // configure P1.1 as simple I/O

P1->SEL0 &= 0xFD;

P1->DIR &= 0xFD; // P1.1 set as output pin \*/

P1->REN |= 0x02;

P1->OUT |= 0x02;

P2->SEL1 &= ~1; // configure P2.0 as simple I/O

P2->SEL0 &= ~1;

P2->DIR |= 1; // P2.0 set as output pin \*/

P2->SEL1 &= ~2; // configure P2.1 as simple I/O

P2->SEL0 &= ~2;

P2->DIR |= 2; // P2.1 set as output pin \*/

P2->SEL1 &= ~4; // configure P2.2 as simple I/O

P2->SEL0 &= ~4;

P2->DIR |= 4; // P2.2 set as output pin \*/

**int** i=0;

**while** (1) {

**while**(DebounceSwitch1())

{

**if**(i==0)

{

P2->OUT &= ~2;

P2->OUT &= ~4;

P2->OUT ^= 1;

i++;

}

**else** **if**(i==1)

{

P2->OUT &= ~4;

P2->OUT &= ~1;

P2->OUT ^= 2;

i++;

}

**else** **if**(i==2)

{

P2->OUT &= ~1;

P2->OUT &= ~2;

P2->OUT ^= 4;

i++;

}

**else** **if**(i==3)

{

P2->OUT &= ~1;

P2->OUT &= ~2;

P2->OUT &= ~4;

i=0;

}

}

}

}

**int** **DebounceSwitch1**(**void**)

{

**int** pin\_Value = 0; //initialize variable as low

**if** ((P1IN & 0x02) == 0x00) //check of button pushed

{ \_\_delay\_cycles(300000); //pause for 10 m-sec for switch bounce

**if** ((P1IN & 0x02) == 0x00) //check of button is still pushed

pin\_Value = 1;

}

**return** pin\_Value; //return 1 if pushed- 0 if not pushed

}

Part 2 Source Code:

**#include** "msp.h"

**int** **DebounceSwitch1**(**void**);

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

{

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

P1->SEL1 &= 0xFD; // configure P1.1 as simple I/O

P1->SEL0 &= 0xFD;

P1->DIR &= 0xFD; // P1.1 set as output pin \*/

P1->REN |= 0x02;

P1->OUT |= 0x02;

P2->SEL1 &= ~1; // configure P2.0 as simple I/O

P2->SEL0 &= ~1;

P2->DIR |= 1; // P2.0 set as output pin \*/

P2->SEL1 &= ~2; // configure P2.1 as simple I/O

P2->SEL0 &= ~2;

P2->DIR |= 2; // P2.1 set as output pin \*/

P2->SEL1 &= ~4; // configure P2.2 as simple I/O

P2->SEL0 &= ~4;

P2->DIR |= 4; // P2.2 set as output pin \*/

**int** i=0;

P2->OUT &= ~BIT0;

P2->OUT &= ~BIT1;

P2->OUT &= ~BIT2;

P2->OUT |= BIT0;

**while** (1) {

**while**((P1->IN & BIT1) != BIT1)

{

**if**(i==4)

{

i=0;

}

**if**(i==0)

{

P2->OUT &= ~BIT2;

P2->OUT |= BIT0;

\_delay\_cycles(2000000);

}

**else** **if**(i==1)

{

P2->OUT &= ~BIT0;

P2->OUT |= BIT1;

\_delay\_cycles(2000000);

}

**else** **if**(i==2)

{

P2->OUT &= ~BIT1;

P2->OUT |= BIT2;

\_delay\_cycles(2000000);

}

i++;

}

}

}

**int** **DebounceSwitch1**(**void**)

{

**int** pin\_Value = 0; //initialize variable as low

**if** ((P1IN & 0x02) == 0x00) //check of button pushed

{ \_\_delay\_cycles(300000); //pause for 10 m-sec for switch bounce

**if** ((P1IN & 0x02) == 0x00) //check of button is still pushed

pin\_Value = 1;

}

**return** pin\_Value; //return 1 if pushed- 0 if not pushed

}