Lab 2 Report
CSE 4560 Embedded System

Name: Daniel Delgado Acosta Coyote ID:006896598

I Goals (give the objectives and purposes of this lab)

The goal of this lab is to get a hang of using code composer studio using the MSP-EXP432P401R LaunchPad development kit. By completing each task in this lab we are able to get a better understanding of future labs.

II Lab Questions, Processes and Program (introduce and summarize the questions and tasks in the lab, provide the details of your method and processes to obtain the solutions along with source codes)

Task 2 - Track the number of toggling operations:
Task 2.1: Flash your very first program!
Start the execution and observe the functionality of the program. What is the purpose of the application?
The code toggles/blinks a red led on the board.

Task 3: Adaptive toggling delay:
Observe that the program never executes the "task completed" print statement. What is the problem?
The problem is that variable i is set to 16 bits which is too limited to complete the task, setting it to uint32_t solves that problem.

What is the effect of defining the delay variable as static? What would happen if we don't?
By defining the delay variable as static the effect is that the variable can not change, therefore if we don't define it as static it is dynamic and can be changed to 0 after every loop.

Task 4: Use your own function:
What is the difference between the two function techniques? How are they called?
When calling by value, the function returns an update value whereas calling by reference will update the value in memory using the address. Calling by value is done by calling the function definition whereas calling by reference is done by using "&".

Task 5: Taking control of the LEDs:
Task 5.1: Static setup of the toggling LEDs
Build and flash the application. What happens?
The red led is toggled/blinking and a counter is printed in the console.

What LEDs will toggle if you set activeLED to 137? To 257? Try to predict then verify.
Predictions: setting it to 137 will produce binary 1000 1001 which should toggle the red led and the blue rgb led. 257 will produce binary 0000 0001 which should toggle the red led alone.
Verification: Yes, my prediction was correct.

III Answers and Results (provide the answers, results, and analysis and give explanations necessary)

Task 2 - Track the number of toggling operations:
Changed Code

```
while (1)
{
  // Toggle an LED
  toggle(LED_RED);

  //
  // TODO: Print and increment the counter
      uart_println("counter = %u",counter); //prints counter to console
      counter += 1; //increments counter


  // Delay of ~166ms
  for (i = 0; i < 50000; i++);
}
}
```

Console

workspace_v9 - lab2/ta

File Edit View Projec

Terminal

COM5

```
counter = 991
counter = 992
counter = 993
counter = 994
counter = 995
counter = 996
counter = 997
counter = 998
counter = 999
counter = 1000
counter = 1001
counter = 1002
counter = 1003
counter = 1004
counter = 1005
counter = 1006
counter = 1007
counter = 1008
counter = 1009
counter = 1010
counter = 1011
counter = 1012
counter = 1013
counter = 1014
counter = 1015
counter = 1016
counter = 1017
counter = 1018
counter = 1019
counter = 1020
counter = 1021
counter = 1022
```

Task 3: Adaptive toggling delay

Changed Code

```
    // Delay
    uint32_t delay = 0;  // Erased static, Declare the initial toggling delay
    delay = delay + 3000;       // Increase it by ~10ms
    for (i = 0; i < delay; i++);
  }
```

Console

```
Problems  Terminal
COM5
counter = 0
counter = 1
counter = 2
counter = 3
counter = 4
counter = 5
counter = 6
counter = 7
counter = 8
counter = 9
counter = 10
counter = 11
counter = 12
counter = 13
counter = 14
counter = 15
counter = 16
counter = 17
counter = 18
counter = 19
counter = 20
counter = 21
counter = 22
counter = 23
counter = 24
Congrat's! You completed task 3 :-)
```

Task 4: Use your own function

Changed code

```
uint32_t print_and_increment(uint32_t value);  // Function prototype declaration
void print_and_increment(uint32_t *value)
{
    uart_println("counter = %u", *value); //prints counter
    *value += 1; //increments value
}


void task_4(void)
{
  // Declare a delay counter
  volatile uint32_t i;

  // Define a toggling counter
  uint32_t counter = 0;

  // Toggle the LED 25 times, with an increasing toggling delay
  while (counter < 25)
  {
    // Toggle an LED
    toggle(LED_RED);

    //
    // TODO: add function call
    print_and_increment(&counter); //calls void function
    //

    // Delay
    static uint32_t delay = 0;  // Declare the initial toggling delay
    delay = delay + 3000;       // Increase it by ~10ms
    for (i = 0; i < delay; i++);
  }

  // Your task consist in reaching this print statement.
  uart_println("Congrat's! You completed task 4 :-) ");

  while (1)
  {
    /* An empty loop preventing the program from exiting */
  }
}


uint32_t print_and_increment(uint32_t value)
{
  //
  // TODO: Implement the function here
    uart_println("counter = %u", value); // prints counter
    value += 1; // increments value
    return value; // returns value
  //
}
```

Console

Problems Terminal

COM5

```
counter = 0
counter = 1
counter = 2
counter = 3
counter = 4
counter = 5
counter = 6
counter = 7
counter = 8
counter = 9
counter = 10
counter = 11
counter = 12
counter = 13
counter = 14
counter = 15
counter = 16
counter = 17
counter = 18
counter = 19
counter = 20
counter = 21
counter = 22
counter = 23
counter = 24
Congrat's! You completed task 4 :-)
```

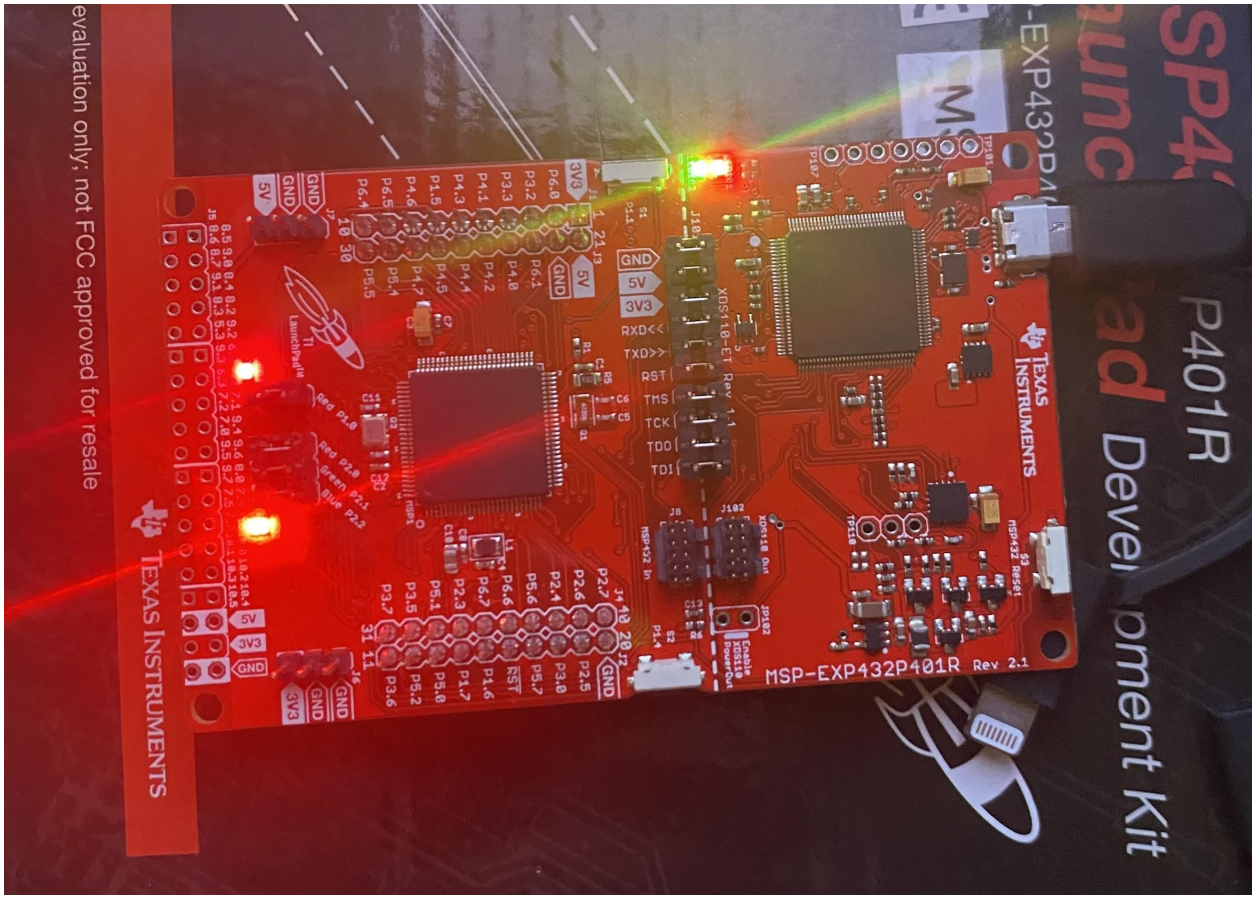
Task 5.1: Static setup of the toggling LEDs

Changed code

```
// TODO: Change the definition of activeLED
//
    uint8_t activeLED = 3;                  // Using decimal
    uint8_t activeLED = 0x03;               // Using hex
    uint8_t activeLED = 0b00000011;         // Using binary
    uint8_t activeLED = LED_RED | RGB_RED;  // Using boolean

  while(1)
  {
```

Board



Task 5.2: Dynamic LED toggling

Changed code

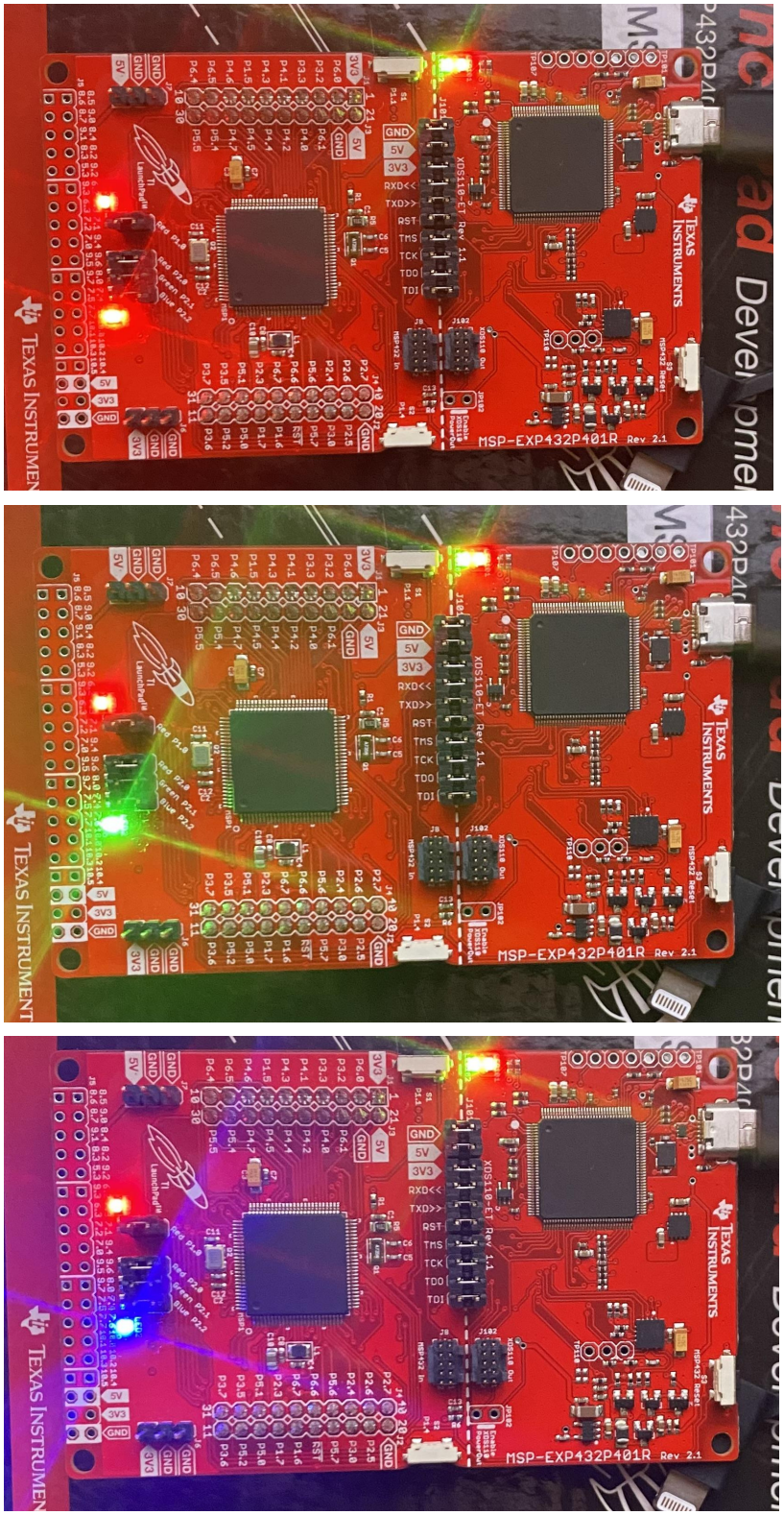
```
while(1)
{
  //
  // TODO: For Task 5.2 (set value for activeLED)
  // Using if statements
     if( counter %3 == 0 )
     {
     activeLED = LED_RED | RGB_RED;
     }
     else if( counter %3 == 1)
     {
     activeLED = LED_RED | RGB_GREEN;
     }
     else if( counter %3 == 2)
     {
     activeLED = LED_RED | RGB_BLUE;
     }

     // Using shift operators
     activeLED = LED_RED | (1 << ( counter %3 + 1) );

  //

  // Toggle an LED
  toggle(activeLED);
```

Board


MSP-EXP432P401R Rev 2.1
TEXAS INSTRUMENTS
XDS110-ET Rev 1.1
GND
5V
3V3
RXD<<
TXD>>
RST
TMS
TCK
TDO
TDI

IV Problem (what kind of problem have you met during the lab, what is the reason for it and how did you solve it?)

I really didn't have too many problems during this lab. One annoying issue I kept having was finding which comm the board was. I solved this issue by finding the comm number in device manager on windows.

V What have you learned

During this lab, I became familiar with coding in C and how to toggle the leds on the board. C is similar C++ however there are subtle difference. I think going forward, what I've learned in this lab will be useful.