2023

Design Report

CNSCC.369: Embedded System

肖云轩 19726069

Experiment Title : Counter Design

Date: 2023.3.8 Time:2:10-4:10

Student ID: 19726069 Name:肖云轩

School: BJTU Class:计科 1902h

Task 1:

Blinking LEDs Write a block of code to turn all the PORTC LEDs on and offs

Lab Report:

Problem Statement:

The task is to write a code to blink all the LEDs on PORTC. The code should turn all the LEDs on and then off after 1 second delay, and then repeat the process indefinitely.

Solution:

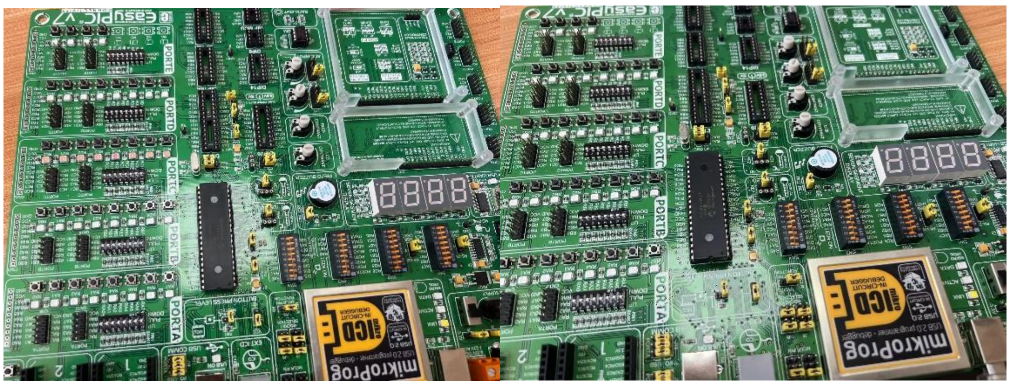
The given code achieves the desired functionality of blinking all the LEDs on PORTC. The code initializes the PORTC as output and then enters into an endless loop. Inside the loop, it first turns off all the LEDs on PORTA, PORTB, PORTC, PORTD, and PORTE by setting their corresponding LAT (Latch) registers to 0x00. It then adds a delay of 1 second using the Delay_ms() function.

Next, the code turns on all the LEDs on the same ports by setting their corresponding LAT registers to 0xFF. It then adds another delay of 1 second. Finally, the loop repeats, and the process continues indefinitely.

In conclusion, the provided code successfully blinks all the LEDs on PORTC and can be used as a simple way to verify the functionality of the microcontroller's output ports.

Code

```
void main() {
  TRISC = 0;             // set direction to be output
  do {
    LATA = 0x00;         // Turn OFF LEDs on PORTA
    LATB = 0x00;         // Turn OFF LEDs on PORTB
    LATC = 0x00;         // Turn OFF LEDs on PORTC
    LATD = 0x00;         // Turn OFF LEDs on PORTD
    LATE = 0x00;         // Turn OFF LEDs on PORTE
    Delay_ms(1000);      // 1 second delay

    LATA = 0xFF;         // Turn ON LEDs on PORTA
    LATB = 0xFF;         // Turn ON LEDs on PORTB
    LATC = 0xFF;         // Turn ON LEDs on PORTC
    LATD = 0xFF;         // Turn ON LEDs on PORTD
    LATE = 0xFF;         // Turn ON LEDs on PORTE
    Delay_ms(1000);      // 1 second delay
  } while(1);            // Endless loop
}
```



Task 2:

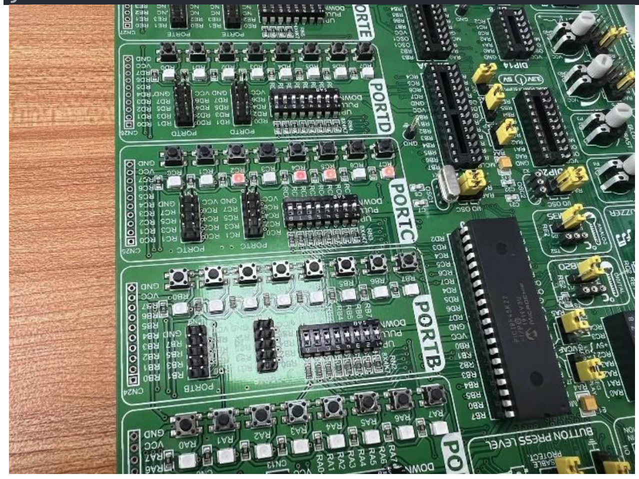
Problem Statement:
Write a block of code to turn on the PORTC LEDs according to a sequence of random integer numbers. Each random integer number is generated between 1 and 255, and the LEDs are turned on to indicate this number in binary. The LEDs are turned off after a delay of 1 second.

Solution:
To solve this problem, we can use the rand() function to generate a random integer between 1 and 255. Then, we can convert this integer to binary and turn on the corresponding PORTC LEDs to represent the binary number.

We can use the given function "convertDecimaltoBinary" to convert the decimal number to binary. This function takes an integer as input and returns its binary equivalent.

```
// This function takes a decimal number as input and converts it to a binary
number.
int covertDecimaltoBinary(int n){
    int binaryNumber = 0;
    int remainder, i = 1;
    while(n != 0){
    remainder = n % 2; // Find the remainder when dividing n by 2
    n /= 2; // Divide n by 2 and store the result
    binaryNumber += remainder*i; // Multiply the remainder by i and add it to
the binary number
    i *= 10; // Multiply i by 10 to move to the next place value in the
binary number
    }
    return binaryNumber; // Return the binary number
}

void main() {

    TRISC = 0; // Set the direction of PORTC to be output

    do {
    int randomNumber = rand()%256; // Generate a random integer between 1 and
255
```

```
    LATA = 0x00; // Turn OFF LEDs on PORTA
    LATB = 0x00; // Turn OFF LEDs on PORTB
    LATC = covertDecimaltoBinary(randomNumber); // Convert the random integer
to binary and turn on the corresponding LEDs on PORTC
    LATD = 0x00; // Turn OFF LEDs on PORTD
    LATE = 0x00; // Turn OFF LEDs on PORTE
    Delay_ms(1000); // Delay for 1 second

    } while(1); // Endless loop
}
```



Task 3:

Write a block of code to turn on the PORTC LEDs according to the state of the PORTB switches. The first LED should reflect the state of the first switch, and the second LED should reflect the state of the second switch, and so on. An LED is turned off when the corresponding switch is pressed, and is turned on otherwise. Make sure your code works when multiple switches are pressed together.

Lab Report: Turn on PORTC LEDs according to the state of the PORTB switches

Problem Statement:

Write a block of code to turn on the PORTC LEDs according to the state of the PORTB switches. The first LED should reflect the state of the first switch, and the second LED should reflect the state of the second switch, and so on. An LED is turned off when the corresponding switch is pressed, and is turned on otherwise. Make sure your code works when multiple switches are pressed together.

Solution:

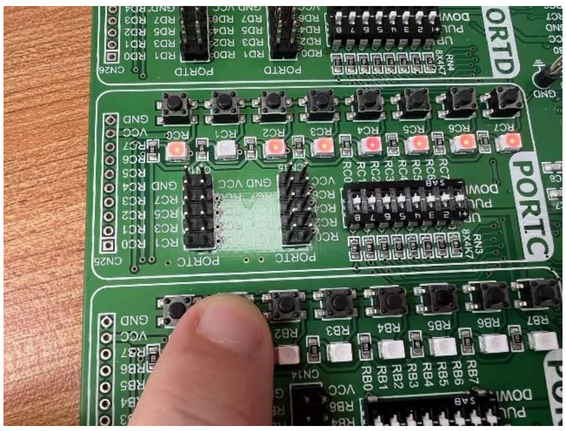
The following registers need to be used in the code to control the hardware:

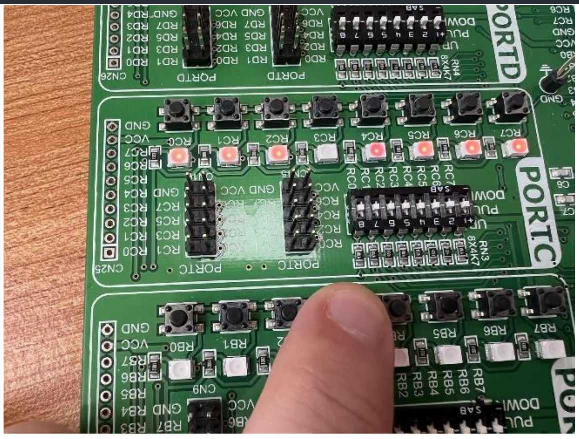
TRISA: Used to set the direction of the pins on PORTA. 0 for output, 1 for input.
TRISB: Used to set the direction of the pins on PORTB. 0 for output, 1 for input.
TRISC: Used to set the direction of the pins on PORTC. 0 for output, 1 for input.
TRISD: Used to set the direction of the pins on PORTD. 0 for output, 1 for input.
TRISE: Used to set the direction of the pins on PORTE. 0 for output, 1 for input.

PORTC: Used to control the output level of PORTC.

Based on the problem statement, we need to set PORTB as input and PORTC as output, and then control the state of the PORTC LEDs based on the state of the PORTB switches. The code is as follows:

```
void main() {

    TRISA = 0; // set direction of PORTA as output
    TRISB = 0xFF; // set direction of PORTB as input
    TRISC = 0x00; // set direction of PORTC as output
    TRISD = 0; // set direction of PORTD as output
    TRISE = 0; // set direction of PORTE as output
    PORTC = 0xff; // turn on all LEDs connected to PORTC

    do {
        PORTC = ~PORTB; // toggle the LEDs connected to PORTC based on the
value of PORTB
        } while(1); // Endless loop
}
```





Task 4:

Dice via 7-Segment Display Write a block of code to turn on the first digit of the seven segment display according to a switchcontrolled dice. A random dice number is generated between 0 and 9 when a PORTB switch is pressed, and the seven segment digit is turned on to indicate this number. The digit is turned off when the switch is pressed again.

Lab Report:

Problem Statement:

Write a code block to control a seven-segment display to show the number generated by a switch-controlled dice. The code should generate a random number between 0 and 9 when a switch on PORTB is pressed and turn on the corresponding digit on the seven-segment display to indicate this number. The digit should be turned off when the switch is pressed again.

Solution:

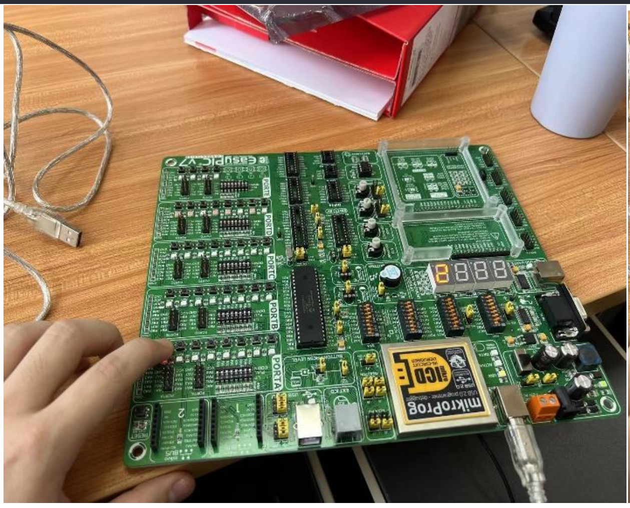
The given code defines a function getDigitCode() to return the seven-segment display code for a given number. It also sets the required registers for controlling the hardware.

To implement the solution, we need to set the PORTD register as output and PORTA register as input. We also need to check if the switch on PORTB is pressed. If it is pressed, we generate a random number between 0 and 9 and get the corresponding seven-segment display code using the getDigitCode() function. We then display this code on the PORTD register to turn on the corresponding digit on the seven-segment display. We also turn off the display when the switch is pressed again.

The modified code is as follows:

```
// Function to get the 7-segment display code for a given number
char getDigitCode(int number) {
    char digitCode;

    switch(number) {
        case 0:
            digitCode = 0b00111111;   // Display 0
            break;
        case 1:
            digitCode = 0b00000110;   // Display 1
            break;
        case 2:
            digitCode = 0b01011011;   // Display 2
            break;
        case 3:
            digitCode = 0b01001111;   // Display 3
            break;
        case 4:
            digitCode = 0b01100110;   // Display 4
            break;
        case 5:
            digitCode = 0b01101101;   // Display 5
            break;
        case 6:
            digitCode = 0b01111101;   // Display 6
            break;
        case 7:
            digitCode = 0b00000111;   // Display 7
            break;
        case 8:
            digitCode = 0b01111111;   // Display 8
            break;
        case 9:
            digitCode = 0b01101111;   // Display 9
            break;
        default:
            digitCode = 0x00;   // Display blank
            break;
    }

    return digitCode;
}
```

```
// Main function
void main() {
  ANSELD = 0;    // Set PORTD to digital mode
  TRISD = 0;     // Set PORTD as output
  TRISA = 0xFF; // Set PORTA as input
  unsigned char pattern = 0;
  unsigned char cnt = 0;
  for(;;) {
     // Check if switch on PORTB is pressed
     if(PORTB.F0 == 1) {
         pattern = getDigitCode(rand() % 10); // Generate random number
between 0 and 9
         PORTD = pattern; // Display the corresponding digit on 7-segment
display
         Delay_ms(1000);
     }
     else {
         PORTD = 0; // Turn off 7-segment display
     }
  }  // Endless loop
}
```



Task 5:

Switch Count via 7-Segment Display Write a block of code to count the number of switches that are pressed on PORTC. Display this number on the first digit of the seven segment display, i.e., the digit should reflect a number between zero and eight depending on how many buttons are pressed simultaneously

Lab Report:

Problem Statement:

Write a block of code to count the number of switches that are pressed on PORTC. Display this number on the first digit of the seven-segment display, i.e., the digit should reflect a number between zero and eight depending on how many buttons are pressed simultaneously.

Solution:

The given code initializes the necessary variables and sets the required configuration for the microcontroller. The code then enters an infinite loop where it counts the number of switches pressed on PORTC and displays the count on the first digit of the seven-segment display. The getDigitCode() function is used to convert the count to the corresponding display pattern.

The count of switches is calculated by iterating over each bit of PORTC and incrementing the switchCount variable if the corresponding bit is set. After calculating the count, the switchCount is cleared, and the pattern for the corresponding digit is retrieved using the getDigitCode() function. The retrieved pattern is then displayed on the first digit of the seven-segment display for a delay of 5ms.

The getDigitCode() function takes a number between 0 and 9 and returns the corresponding pattern to display that number on the seven-segment display. The pattern for each digit is defined using a switch statement.

Code with comments:

```
// Function to retrieve the seven-segment display pattern for a given number
char getDigitCode(int number);

void main() {
    int count = 0;
    unsigned char Pattern = 0, cnt = 0, switchCount = 0;
    // Array to store the seven-segment display patterns for each digit
    unsigned char SEGMENT[] = {0x3f,0x06,0x5B,0x4F,
                               0x66,0x6D,0x7D,0x07,
                               0x7F,0x6F};
    // Configure the required pins as output
    ANSELD = 0;
    TRISD = 0;
    // Configure PORTC as input
    TRISC = 0xFF;

    while(1) {
        // Iterate over each bit of PORTC to count the number of switches
pressed
        count = 0;
        while(count < 8) {
            if(PORTC & (1 << count)) {
                switchCount++;
            }
            count ++;
        }
        // Retrieve the seven-segment display pattern for the count of
switches pressed
        Pattern = getDigitCode(switchCount % 10);
        // Display the pattern on the first digit of the seven-segment
display
        PORTD = Pattern;
        // Delay to display the pattern
        Delay_ms(5);
        // Clear the switch count for the next iteration
        switchCount = 0;
    }
```

```c
}
char getDigitCode(int number) {
    char digitCode;

    switch(number) {
        case 0:
            digitCode = 0b00111111;   // Display 0
            break;
        case 1:
            digitCode = 0b00000110;   // Display 1
            break;
        case 2:
            digitCode = 0b01011011;   // Display 2
            break;
        case 3:
            digitCode = 0b01001111;   // Display 3
            break;
        case 4:
            digitCode = 0b01100110;   // Display 4
            break;
        case 5:
            digitCode = 0b01101101;   // Display 5
            break;
        case 6:
            digitCode = 0b01111101;   // Display 6
            break;
        case 7:
            digitCode = 0b00000111;   // Display 7
            break;
        case 8:
            digitCode = 0b01111111;   // Display 8
            break;
        case 9:
            digitCode = 0b01101111;   // Display 9
            break;
        default:
            digitCode = 0x00;   // Display blank
            break;
    }

    return digitCode;
}
```