EE2016_MUP_LAB_EXPERIMENT_07_ Switch LED Stepper Motor Control

MIRUDHULA J | EE23B046

Drive link for all the experiments with video reference and reports:

MUP LAP

https://drive.google.com/drive/folders/1ir0SKb8wvSX2joF7MPM4rSHpboF_FUIA?usp=sharing

OBJECTIVES:

- Using C-interfacing, use C-programming, to implement the following tasks:

  (i) Read the status (binary position) of the switch and use the LEDs (8 LEDs are provided) to display the status of each of the 8-bit DIP switch

  (ii) Stepper motor control using Vi Microsystem's ViARM 7238 development board. Due to the ongoing pandemic, only an emulated version of this experiment is intended here.

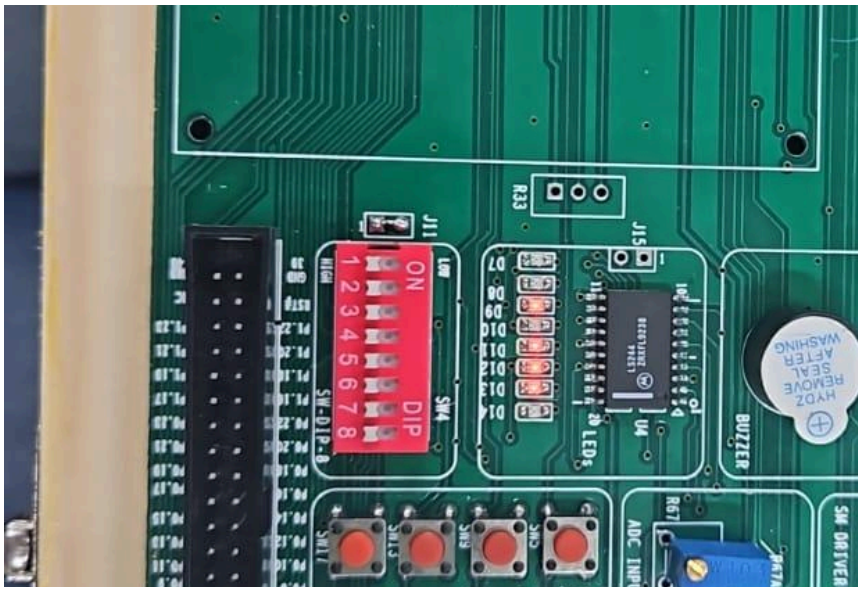
CODE USED:

PROBLEM_1: Write a C-program to display last three digits of your roll number (in hex) in LEDs on the ARM-board. (Your roll number is in decimal, note).

```
      //LPC2148- C program to display a value on the LEDs
#include "LPC214x.h"                    /* LPC21xx definitions */

 #define LED_IOPIN              IO0PIN
#define BIT(x) (1 << x)

#define LED_D0        (1 << 10)               // P0.10
#define LED_D1        (1 << 11)               // P0.11
#define LED_D2        (1 << 12)               // P0.12
#define LED_D3        (1 << 13)               // P0.13

#define LED_D4        (1 << 15)               // P0.15
#define LED_D5        (1 << 16)               // P0.16
#define LED_D6        (1 << 17)               // P0.17
#define LED_D7        (1 << 18)               // P0.18
#define LED_DATA_MASK                         ((unsigned long)((LED_D7 | LED_D6 | LED_D5 |
LED_D4 | LED_D3 | LED_D2 | LED_D1 | LED_D0)))
        #ifndef LED_DRIVER_OUTPUT_EN
#define LED_DRIVER_OUTPUT_EN (1 << 5)        // P0.5
#endif
#define LED1_ON               LED_IOPIN |= (unsigned long)(LED_D0);              // LED1 ON
```

```
#define LED2_ON             LED_IOPIN |= (unsigned long)(LED_D1);          // LED2 ON
#define LED3_ON             LED_IOPIN |= (unsigned long)(LED_D2);          // LED3 ON
#define LED4_ON             LED_IOPIN |= (unsigned long)(LED_D3);          // LED4 ON
#define LED5_ON             LED_IOPIN |= (unsigned long)(LED_D4);          // LED5 ON
#define LED6_ON             LED_IOPIN |= (unsigned long)(LED_D5);          // LED6 ON
#define LED7_ON             LED_IOPIN |= (unsigned long)(LED_D6);          // LED7 ON
#define LED8_ON             LED_IOPIN |= (unsigned long)(LED_D7);          // LED8 ON

int main (void)
{

  IO0DIR |= LED_DATA_MASK;                          // GPIO Direction control -> pin is output
        IO0DIR |= LED_DRIVER_OUTPUT_EN;                 // GPIO Direction control -> pin is
output
        IO0CLR |= LED_DRIVER_OUTPUT_EN;


        while(1)
        {
                int value=0x2E;

        if(value & BIT(0)) LED8_ON;
        if(value & BIT(1)) LED7_ON;
        if(value & BIT(2)) LED6_ON;
        if(value & BIT(3)) LED5_ON;

        if(value & BIT(4)) LED4_ON;
        if(value & BIT(5)) LED3_ON;
        if(value & BIT(6)) LED2_ON;
        if(value & BIT(7)) LED1_ON;
        }

  return 0;
}
/******************************************************************************
**                      End Of File
******************************************************************************/
```

OUTPUT:



46 is displayed as output.

PROBLEM_2: Read the DIP positions and display the byte in LED (logical ’1’ by OFF & logical ’0’ for ’ON’). You can see the ON position written on the red colored DIP switches.

HERE 5 FILES ARE USED:

1)SWITCHLED.C

```
#include "LPC214x.H"                        /* LPC214x definitions */
#include "led.h"
#include "delay.h"

//-------------------------------------------------------------------------
-------------------------

#define DIP_SW_D0 (1 << 16)        // P0.16
#define DIP_SW_D1 (1 << 17)        // P0.17
#define DIP_SW_D2 (1 << 18)        // P0.18
#define DIP_SW_D3 (1 << 19)        // P0.19

#define DIP_SW_D4 (1 << 22)        // P0.22
#define DIP_SW_D5 (1 << 23)        // P0.23
#define DIP_SW_D6 (1 << 24)        // P0.24
#define DIP_SW_D7 (1 << 25)        // P0.25

#define DIP_SW_DIR           IO1DIR
#define DIP_SW_PIN           IO1PIN

#define DIP_SW_DATA_MASK     (DIP_SW_D7 | DIP_SW_D6 | DIP_SW_D5 |
DIP_SW_D4 | DIP_SW_D3 | DIP_SW_D2 | DIP_SW_D1 | DIP_SW_D0)

//-------------------------------------------------------------------------
-------------------------
```

```
void set_dipswitch_port_input( void )
{
      DIP_SW_DIR &= ~(DIP_SW_DATA_MASK);
}

unsigned long read_dip_switch( void )
{
      return DIP_SW_PIN;
}

/*****************************************************************************
****
**   Main Function  main()
*****************************************************************************
****/
int main (void)
{
      unsigned long sw_status;

      set_led_port_output();
      set_dipswitch_port_input();

      while(1)
      {
            sw_status = read_dip_switch();
/*
            if(sw_status & DIP_SW_D0){ LED1_OFF;} else{ LED1_ON;}
            if(sw_status & DIP_SW_D1){ LED2_OFF;} else{ LED2_ON;}
            if(sw_status & DIP_SW_D2){ LED3_OFF;} else{ LED3_ON;}
            if(sw_status & DIP_SW_D3){ LED4_OFF;} else{ LED4_ON;}

            if(sw_status & DIP_SW_D4){ LED5_OFF;} else{ LED5_ON;}
            if(sw_status & DIP_SW_D5){ LED6_OFF;} else{ LED6_ON;}
            if(sw_status & DIP_SW_D6){ LED7_OFF;} else{ LED7_ON;}
            if(sw_status & DIP_SW_D7){ LED8_OFF;} else{ LED8_ON;}
*/

            if(sw_status & DIP_SW_D0){ LED1_OFF;} else{ LED1_ON;}
                        delay_mSec(10);
            if(sw_status & DIP_SW_D1){ LED2_OFF;} else{ LED2_ON;}
                        delay_mSec(10);
            if(sw_status & DIP_SW_D2){ LED3_OFF;} else{ LED3_ON;}
                        delay_mSec(10);
            if(sw_status & DIP_SW_D3){ LED4_OFF;} else{ LED4_ON;}
                        delay_mSec(10);

            if(sw_status & DIP_SW_D4){ LED5_OFF;} else{ LED5_ON;}
                        delay_mSec(10);
            if(sw_status & DIP_SW_D5){ LED6_OFF;} else{ LED6_ON;}
                        delay_mSec(10);
            if(sw_status & DIP_SW_D6){ LED7_OFF;} else{ LED7_ON;}
                        delay_mSec(10);
            if(sw_status & DIP_SW_D7){ LED8_OFF;} else{ LED8_ON;}
```

```
                        delay_mSec(10);

            delay_mSec(100);
      }

//    return 0;
}

/*****************************************************************************
*****
**                              End Of File
*****************************************************************************
****/
```

2)LED.H:

```
#include <LPC214x.H>                /* LPC21xx definitions */

#define BIT(x)    (1 << x)

#define LED_D0    (1 << 10)         // P0.10
#define LED_D1    (1 << 11)         // P0.11
#define LED_D2    (1 << 12)         // P0.12
#define LED_D3    (1 << 13)         // P0.13

#define LED_D4    (1 << 15)         // P0.15
#define LED_D5    (1 << 16)         // P0.16
#define LED_D6    (1 << 17)         // P0.17
#define LED_D7    (1 << 18)         // P0.18

#define LED_IOPIN       IO0PIN
#define LED_CTRL_DIR    IO0DIR
#define LED_CTRL_SET    IO0SET
#define LED_CTRL_CLR    IO0CLR


#define LED_DATA_MASK             ((unsigned long)((LED_D7 | LED_D6 |
LED_D5 | LED_D4 | LED_D3 | LED_D2 | LED_D1 | LED_D0)))
#define LED_ODD_DATA_MASK         ((unsigned int)(LED_D7 | LED_D5 |
LED_D3 | LED_D1))
#define LED_EVEN_DATA_MASK        ((unsigned int)(LED_D6 | LED_D4 |
LED_D2 | LED_D0))

/*-------------------------------------------------------------------------
----------*/

#define LED1_ON         LED_IOPIN |= (unsigned long)(LED_D0);         //
LED1 ON
#define LED2_ON         LED_IOPIN |= (unsigned long)(LED_D1);         //
LED2 ON
#define LED3_ON         LED_IOPIN |= (unsigned long)(LED_D2);         //
LED3 ON
```

```
#define LED4_ON        LED_IOPIN |= (unsigned long)(LED_D3);         //
LED4 ON
#define LED5_ON        LED_IOPIN |= (unsigned long)(LED_D4);         //
LED5 ON
#define LED6_ON        LED_IOPIN |= (unsigned long)(LED_D5);         //
LED6 ON
#define LED7_ON        LED_IOPIN |= (unsigned long)(LED_D6);         //
LED7 ON
#define LED8_ON        LED_IOPIN |= (unsigned long)(LED_D7);         //
LED8 ON


#define LED1_OFF LED_IOPIN &= ~(unsigned long)(LED_D0);        // LED1 OFF
#define LED2_OFF LED_IOPIN &= ~(unsigned long)(LED_D1);        // LED2 OFF
#define LED3_OFF LED_IOPIN &= ~(unsigned long)(LED_D2);        // LED3 OFF
#define LED4_OFF LED_IOPIN &= ~(unsigned long)(LED_D3);        // LED4 OFF
#define LED5_OFF LED_IOPIN &= ~(unsigned long)(LED_D4);        // LED5 OFF
#define LED6_OFF LED_IOPIN &= ~(unsigned long)(LED_D5);        // LED6 OFF
#define LED7_OFF LED_IOPIN &= ~(unsigned long)(LED_D6);        // LED7 OFF
#define LED8_OFF LED_IOPIN &= ~(unsigned long)(LED_D7);        // LED8 OFF

/*-------------------------------------------------------------------------
----------*/

#ifndef LED_DRIVER_OUTPUT_EN
#define LED_DRIVER_OUTPUT_EN (1 << 5)   // P0.5
#endif

#ifndef LCD_DRIVER_OUTPUT_EN
#define LCD_DRIVER_OUTPUT_EN (1 << 7)   // P0.7
#endif

/*-------------------------------------------------------------------------
----------*/

void Led_On_All(void);
void Led_Off_All(void);
void Led_Toggle(void);
void Turn_On_Led(int value);
void set_led_port_output( void );
```

3)<u>LED.C:</u>

```
#include <LPC214x.H>                        /* LPC21xx definitions */
#include "led.h"

int led_counter;


/**
*************************************************************************
*****************
  Function Name :set_led_port_output()
```

```
  Description   :

** parameters:              None
** Returned value:          None

  Note          :
*******************************************************************************
******************
*/
void set_led_port_output( void )
{
// Enable LED PINs
      IO0DIR |= LED_DATA_MASK;                // GPIO Direction control ->
pin is output
      IO0DIR |= LED_DRIVER_OUTPUT_EN;         // GPIO Direction control ->
pin is output
      IO0CLR |= LED_DRIVER_OUTPUT_EN;         // GPIO Port Output Clear
register -> LED HARDWARE_DRIVER(74LVC244) ENABLE -> P0.5 goes LOW

// Disable LCD PINs
      IO0DIR |= LCD_DRIVER_OUTPUT_EN;         // GPIO Direction control ->
pin is output
      IO0SET |= LCD_DRIVER_OUTPUT_EN;         // GPIO Port Output Clear
register -> LCD HARDWARE_DRIVER(74LVC244) DISABLE -> P0.7 goes HIGH
}

//-----------------------------------------------------------------------
------------

void Led_On_All(void)
{
      LED_IOPIN |= LED_DATA_MASK;
}

void Led_Off_All(void)
{
      LED_IOPIN &= ~(LED_DATA_MASK);
}

//-----------------------------------------------------------------------
------------

void Led_Toggle(void)
{
      led_counter++;

      if( led_counter & 0x01 )     /* alternate the LED display */
      {
            LED_IOPIN &= ~(LED_ODD_DATA_MASK);
            LED_IOPIN |= LED_EVEN_DATA_MASK;
      }
      else
```

```
	{
		LED_IOPIN &= ~(LED_EVEN_DATA_MASK);
		LED_IOPIN |= LED_ODD_DATA_MASK;
	}
}

//-------------------------------------------------------------------------
------------

void Turn_On_Led(int value)
{
/*
	int Lower_Nibble, Higher_Nibble;

	Lower_Nibble = (value & 0x0f);
	Higher_Nibble = ((value >> 4) & 0x0f);

	LED_IOPIN &= ~(LED_DATA_MASK);
	LED_IOPIN |= Higher_Nibble << 15;
	LED_IOPIN |= Lower_Nibble  << 10;
*/

	Led_Off_All();

	if(value & BIT(0)) LED8_ON;
	if(value & BIT(1)) LED7_ON;
	if(value & BIT(2)) LED6_ON;
	if(value & BIT(3)) LED5_ON;

	if(value & BIT(4)) LED4_ON;
	if(value & BIT(5)) LED3_ON;
	if(value & BIT(6)) LED2_ON;
	if(value & BIT(7)) LED1_ON;

}

//-------------------------------------------------------------------------
------------
```

4)DELAY.H:

```
#ifndef _DELAY_H
#define _DELAY_H

void delay_mSec(int);
void delay_10uSec(void);
void delay_100uSec(void);

#endif
```

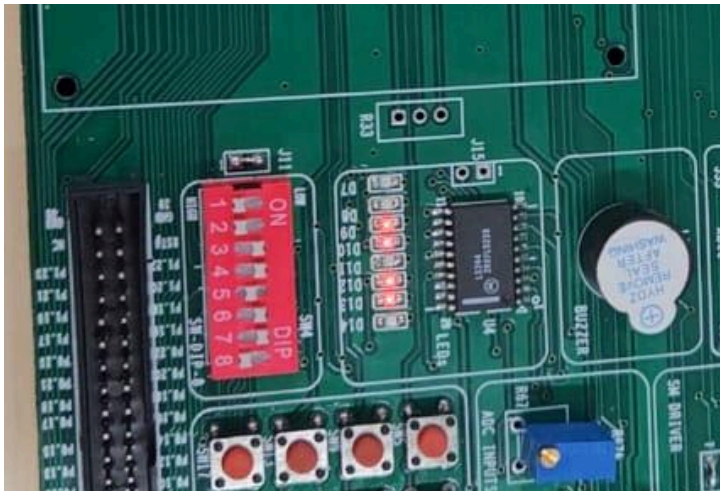
5)DELAY.C:

```
#include "delay.h"
```

```
/*-------------------------------------------------------------------------
----------
                          void delay_10uSec(void)

-------------------------------------------------------------------------
--------

 * Return value  : none

 * description :
         This function is used generate a approximate delay in 10uSec.

-------------------------------------------------------------------------
---------*/

void delay_10uSec(void)           // pr_note:~10 uSec
{
  int j=0,i=0;

  for(j=0;j<10;j++)
  {
    /* At 60Mhz, the below loop introduces
    delay of 10 us */
    for(i=0;i<10;i++);
  }
}


/*-------------------------------------------------------------------------
----------
                          void delay_100uSec(void)

-------------------------------------------------------------------------
--------

 * Return value  : none

 * description :
         This function is used generate a approximate delay in 100uSec.

-------------------------------------------------------------------------
---------*/

void delay_100uSec(void)          // pr_note:~100 uSec
{
  int j=0,i=0;

  for(j=0;j<100;j++)
  {
    /* At 60Mhz, the below loop introduces
    delay of 10 us */
    for(i=0;i<10;i++);
```

```
  }
}


/*-------------------------------------------------------------------------
----------
                          void delay_mSec(int dCnt)

-------------------------------------------------------------------------
---------
 * I/P Arguments: int
 * Return value  : none

 * description:
     This function is used generate delay in ms.
     It genarates a approximate delay of 1ms for each count,
     if 1000 is passed as the argument then it generates delay of apprx
1000ms(1sec)
-------------------------------------------------------------------------
---------*/

void delay_mSec(int dCnt)          // pr_note:~dCnt mSec
{
  int j=0,i=0;

  while(dCnt--)
  {
        for(j=0;j<1000;j++)
        {
          /* At 60Mhz, the below loop introduces
          delay of 10 us */
          for(i=0;i<10;i++);
        }
  }
}
```

OUTPUT:



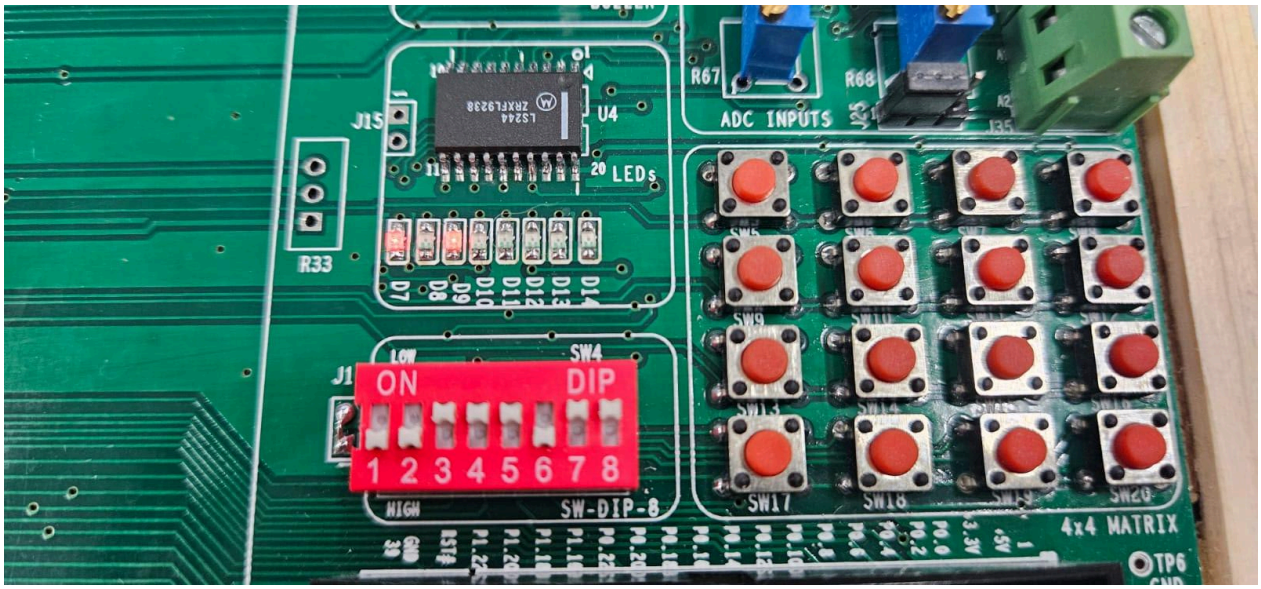
PROBLEM_3: Write a C program to read a DIP switch, split into two nibbles (4 bits), display the sum in the LEDs.

```
#include "LPC214x.H"                        /* LPC214x definitions */
#include "led.h"
#include "delay.h"

//-------------------------------------------------------------------------
-------------------------

#define DIP_SW_D0 (1 << 16)        // P0.16
#define DIP_SW_D1 (1 << 17)        // P0.17
#define DIP_SW_D2 (1 << 18)        // P0.18
#define DIP_SW_D3 (1 << 19)        // P0.19

#define DIP_SW_D4 (1 << 22)        // P0.22
#define DIP_SW_D5 (1 << 23)        // P0.23
#define DIP_SW_D6 (1 << 24)        // P0.24
#define DIP_SW_D7 (1 << 25)        // P0.25

#define DIP_SW_DIR           IO1DIR
#define DIP_SW_PIN           IO1PIN

#define DIP_SW_DATA_MASK     (DIP_SW_D7 | DIP_SW_D6 | DIP_SW_D5 |
DIP_SW_D4 | DIP_SW_D3 | DIP_SW_D2 | DIP_SW_D1 | DIP_SW_D0)

//-------------------------------------------------------------------------
-------------------------

void set_dipswitch_port_input( void )
{
      DIP_SW_DIR &= ~(DIP_SW_DATA_MASK);
}

unsigned long read_dip_switch( void )
{
      return DIP_SW_PIN;
}

/*************************************************************************
****
**   Main Function  main()
*************************************************************************
****/
int main(void)
{
    unsigned long sw_status;
    unsigned char lower_nibble, upper_nibble, sum;

    set_led_port_output();        // Set LED pins as output
    set_dipswitch_port_input();   // Set DIP switch pins as input
```

```
    while (1)
    {
        // Read DIP switch state
        sw_status = read_dip_switch();

        // Extract lower 4 bits (nibble) from bits P0.16 to P0.19
        lower_nibble = (sw_status >> 16) & 0x0F;

        // Extract upper 4 bits (nibble) from bits P0.22 to P0.25
        upper_nibble = (sw_status >> 22) & 0x0F;

        // Compute the sum of the two nibbles
        sum = lower_nibble + upper_nibble;

        // Display the sum on the first 4 LEDs
        if (sum & 0x01) { LED1_ON; } else { LED1_OFF; }
        if (sum & 0x02) { LED2_ON; } else { LED2_OFF; }
        if (sum & 0x04) { LED3_ON; } else { LED3_OFF; }
        if (sum & 0x08) { LED4_ON; } else { LED4_OFF; }
       if (sum & 0x10) { LED1_ON; } else { LED1_OFF; }
        if (sum & 0x20) { LED2_ON; } else { LED2_OFF; }
        if (sum & 0x40) { LED3_ON; } else { LED3_OFF; }
        if (sum & 0x80) { LED4_ON; } else { LED4_OFF; }

        delay_mSec(100);  // Add a delay to avoid rapid updates
    }
}


/*****************************************************************************
*****
**                            End Of File
*****************************************************************************
****/
```

This is the main C code, the remaining files i.e., led.h, led.c, delay.h, delay.c are the same.

OUTPUT:



Sum of the two numbers 0011 and 0010, the result 0101 is displayed by the leds.

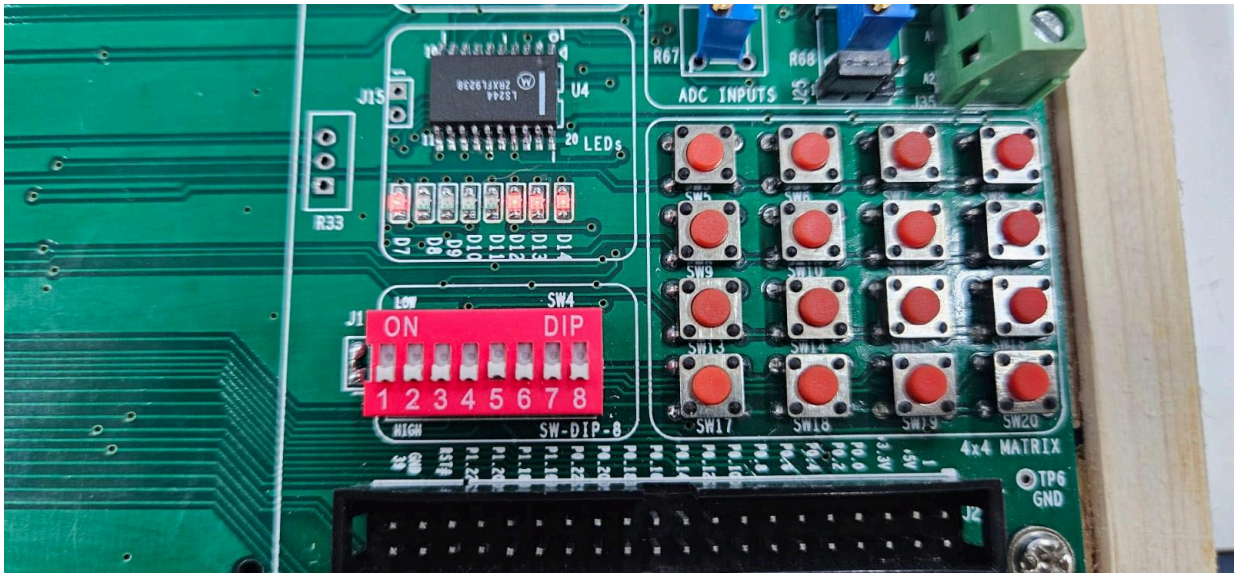
PROBLEM_4: Write a C program to read a DIP switch, split into two nibbles (4 bits), display the product in the LEDs.

```
#include "LPC214x.H"                        /* LPC214x definitions */
#include "led.h"
#include "delay.h"

//-------------------------------------------------------------------------
-------------------------

#define DIP_SW_D0 (1 << 16)        // P0.16
#define DIP_SW_D1 (1 << 17)        // P0.17
#define DIP_SW_D2 (1 << 18)        // P0.18
#define DIP_SW_D3 (1 << 19)        // P0.19

#define DIP_SW_D4 (1 << 22)        // P0.22
#define DIP_SW_D5 (1 << 23)        // P0.23
#define DIP_SW_D6 (1 << 24)        // P0.24
#define DIP_SW_D7 (1 << 25)        // P0.25

#define DIP_SW_DIR           IO1DIR
#define DIP_SW_PIN           IO1PIN

#define DIP_SW_DATA_MASK     (DIP_SW_D7 | DIP_SW_D6 | DIP_SW_D5 |
DIP_SW_D4 | DIP_SW_D3 | DIP_SW_D2 | DIP_SW_D1 | DIP_SW_D0)

//-------------------------------------------------------------------------
-------------------------

void set_dipswitch_port_input( void )
{
      DIP_SW_DIR &= ~(DIP_SW_DATA_MASK);
}
```

```
unsigned long read_dip_switch( void )
{
      return DIP_SW_PIN;
}

/*****************************************************************************
****
**   Main Function  main()
*****************************************************************************
****/
int main(void)
{
    unsigned long sw_status;
    unsigned char lower_nibble, upper_nibble;
    unsigned char product;

    set_led_port_output();        // Set LED pins as output
    set_dipswitch_port_input();   // Set DIP switch pins as input

    while (1)
    {
        // Read DIP switch state
        sw_status = read_dip_switch();

        // Extract lower 4 bits (nibble) from bits P0.16 to P0.19
        lower_nibble = (sw_status >> 16) & 0x0F;

        // Extract upper 4 bits (nibble) from bits P0.22 to P0.25
        upper_nibble = (sw_status >> 22) & 0x0F;

        // Compute the product of the two nibbles (up to 8 bits)
        product = lower_nibble * upper_nibble;

        // Display the product on the LEDs
        if (product & 0x01) { LED1_ON; } else { LED1_OFF; }
        if (product & 0x02) { LED2_ON; } else { LED2_OFF; }
        if (product & 0x04) { LED3_ON; } else { LED3_OFF; }
        if (product & 0x08) { LED4_ON; } else { LED4_OFF; }
        if (product & 0x10) { LED5_ON; } else { LED5_OFF; }
        if (product & 0x20) { LED6_ON; } else { LED6_OFF; }
        if (product & 0x40) { LED7_ON; } else { LED7_OFF; }
        if (product & 0x80) { LED8_ON; } else { LED8_OFF; }

        delay_mSec(100);  // Add a delay to avoid rapid updates
    }
}

/*****************************************************************************
*****
**                            End Of File
*****************************************************************************
****/
```

This is the main C code, the remaining files i.e., led.h, led.c, delay.h, delay.c are the same.

OUTPUT:



Product of the two numbers 1111 and 1111, the result 11100001 (left is lsb) is displayed by the leds.

PROBLEM_4: Stepper Motor:

The following code is used for the stepper motor.

```
/*5V Stepper Motor

Connector J16 connect with stepper motor  as per below mentioned
configuration

Pin - 1   : BLUE
Pin - 2   : PINK
Pin - 3   : YELLOW
Pin - 4   : ORANGE
Pin - 5   : Red (Motor Vcc)

Motor Pins:
P0.4 to P0.7
*/

#include <LPC214x.h>                        /* LPC21xx definitions */

void delay_mSec(int);

int main (void)
{
      int i;
```

```
unsigned char steps[4] = {0x09, 0x0c, 0x06, 0x03};  //standard step
sequence for stepper motor
signed char x = 0;


     PINSEL0 = 0x0;
          // Pin function Select -> P0.0 to P0.15 -> GPIO Port
     IO0DIR |= 0xF0;                         // Set stepper motor pins
as output in IO0 port
     delay_mSec(10);

     while(1)
     {
     for(i=0;i<2500;i++)
     {
          IO0PIN = (steps[x--] << 4); //send the 4 bit step value to
stepper motor lines connected to IO0 port
          if(x < 0)
               x = 3;

          delay_mSec(1);
     }
     }
    return 0;
}

void delay_mSec(int dCnt)       // pr_note:~dCnt mSec
{
  int j=0,i=0;
  while(dCnt--)
  {
       for(j=0;j<100;j++)
       {
         /* At 60Mhz, the below loop introduces
         delay of 10 us */
         for(i=0;i<10;i++);
       }
  }
}
```

Video file:
https://drive.google.com/drive/folders/18CRwpEnCrKj3ZptpJecro0RXYamTxai3?usp=drive_link

Gauge the Speed of the Stepper Motor:
The speed is controlled by the delay function delay_mSec(1); within the loop. With a delay of 1 millisecond per step and 200 steps per full rotation, it will complete one rotation in approximately 200 milliseconds. The motor speed in rotations per minute (RPM) would be approximately:
Speed (RPM) = 60/0.2 = 300 rpm
Maximum Speed:
To increase the speed, you can reduce the delay in delay_mSec(). The motor's maximum

reliable speed varies based on its specifications, but reducing the delay to the lowest value that still maintains smooth operation (typically around delay_mSec(0)) would give the maximum achievable speed. Testing different delay values can help find the optimal speed without losing steps.

Steps Required for 360° Rotation:

Assuming the stepper motor has a 1.8° step angle, it will need:

360 / 1.8 = 200 steps

This means 200 steps are required for one full 360° rotation. Adjust STEPS_PER_REV accordingly if the motor's step angle differs.

PROCEDURE FOLLOWED:

- Wrote the C programs for the above problems (one at a time).
- Entered the above program in KEIL software, edit and compile / assemble.
- Then the generated hex file is uploaded into the Flash Magic.
- Then it is programmed into the LPC2148 board and generated to view the results.
- Finally, demonstrated its working, to TA

MY CONTRIBUTION:

- I was responsible for 3rd and 4th question coding and implementation.