EE23B022RprtLab7

Deepak Charan S

October 2024

1 Objective

Interfacing of DIP Switch, LED and stepper motor using C coding and ARK's LPC2148 embedded ARM development board.

2 Equipments/Software Required:

1. ARK's LPC2148 embedded ARM development board and accessaries
2. RS-232 cable
3. Windows PC with Keil microvision 5, Flash magic and Burn o-mat.
4. USB - serial converter
5. Stepper motor

3 General Procedure:

1. Go through the handout to understand the working of LPC2148 board.
2. Download KEIL Microvision 5 and flash magic on a windows PC.
3. build a new project in KEIL Microvision 5 with the right configurations and create a new C file in the Source File section
4. Write the proper C code to accomplish all the desired objective.
5. Burn the code into the board using Flash Magic (By using the .hex file that would have been automatically generated when we built the project)

6. Verify if the code works by checking for multiple test cases.

7. Repeat this procedure for all the tasks

4 LEDs and DIP Switch Tasks:

4.1 Blinking the LEDs on the board

4.1.1 Implementation:

- The 10-13 and 15-18 pins of the IOPIN register are used for LEDs which we set as output by setting the appropriate value of IO0DIR. Pin 5 (Controls LED Driver) should also be set in IO0DIR
- Inside the main function we create an infinite while loop wherein we alternate the bits of the LED pin (with a delay in between), creating a blinking effect

4.1.2 Code:

```
#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
```

```
20 #define LED2_ON      LED_IOPIN |= (unsigned long)(LED_D1)
       ;        // LED2 ON
21 #define LED3_ON      LED_IOPIN |= (unsigned long)(LED_D2)
       ;        // LED3 ON
22 #define LED4_ON      LED_IOPIN |= (unsigned long)(LED_D3)
       ;        // LED4 ON
23 #define LED5_ON      LED_IOPIN |= (unsigned long)(LED_D4)
       ;        // LED5 ON
24 #define LED6_ON      LED_IOPIN |= (unsigned long)(LED_D5)
       ;        // LED6 ON
25 #define LED7_ON      LED_IOPIN |= (unsigned long)(LED_D6)
       ;        // LED7 ON
26 #define LED8_ON      LED_IOPIN |= (unsigned long)(LED_D7)
       ;        // LED8 ON
27
28 void delay(void){
29     for(int i=0;i<0xff;i++) for(int j=0;j<0xff;j++);
30 }
31 int main (void)
32 {
33
34   IO0DIR |= LED_DATA_MASK;           // GPIO Direction
        control -> pin is output
35     IO0DIR |= LED_DRIVER_OUTPUT_EN;      // GPIO
         Direction control -> pin is output
36     IO0CLR |= LED_DRIVER_OUTPUT_EN;
37
38
39     while(1)
40     {
41     int value=0xff;
42
43     if(value & BIT(0)) LED8_OFF;
44     if(value & BIT(1)) LED7_OFF;
45     if(value & BIT(2)) LED6_OFF;
46     if(value & BIT(3)) LED5_OFF;
47
48     if(value & BIT(4)) LED4_OFF;
49     if(value & BIT(5)) LED3_OFF;
50     if(value & BIT(6)) LED2_OFF;
51     if(value & BIT(7)) LED1_OFF;
52
53     delay();
```

```
54
55      value=0xff;
56
57      if(value & BIT(0)) LED8_ON;
58      if(value & BIT(1)) LED7_ON;
59      if(value & BIT(2)) LED6_ON;
60      if(value & BIT(3)) LED5_ON;
61
62      if(value & BIT(4)) LED4_ON;
63      if(value & BIT(5)) LED3_ON;
64      if(value & BIT(6)) LED2_ON;
65      if(value & BIT(7)) LED1_ON;
66
67      delay();
68
69      }
70
71    return 0;
72  }
```

Video for Blinking LED is provided here

4.2 Read the DIP Switch positions and display the byte via LEDs

4.2.1 Implementation:

1. bits 16-19 and 22-25 of PO belong to the DIP Switch and we should accordingly enable it as input. Setting LED pins as output is directly done through *set_led_port_output ()* function present in *led.h*

2. in the main function, there is an infinite while loop wherein we read the DIP Switch data and accordingly display the value through LEDs

4.2.2 Code:

```
1
2   #include "LPC214x.H"       /* LPC214x definitions */
3   #include "led.h"
4   #include "delay.h"
5
6   #define DIP_SW_D0 (1 << 16)      // P0.16
7   #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;
}


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
       ;}
        delay_mSec(10);
    if(sw_status & DIP_SW_D1){ LED2_OFF;} else{ LED2_ON
       ;}
```

```
47          delay_mSec(10);
48      if(sw_status & DIP_SW_D2){ LED3_OFF;} else{ LED3_ON
            ;}
49          delay_mSec(10);
50      if(sw_status & DIP_SW_D3){ LED4_OFF;} else{ LED4_ON
            ;}
51          delay_mSec(10);
52
53      if(sw_status & DIP_SW_D4){ LED5_OFF;} else{ LED5_ON
            ;}
54          delay_mSec(10);
55      if(sw_status & DIP_SW_D5){ LED6_OFF;} else{ LED6_ON
            ;}
56          delay_mSec(10);
57      if(sw_status & DIP_SW_D6){ LED7_OFF;} else{ LED7_ON
            ;}
58          delay_mSec(10);
59      if(sw_status & DIP_SW_D7){ LED8_OFF;} else{ LED8_ON
            ;}
60          delay_mSec(10);
61
62      }
63
64  //     return 0;
65  }
```

4.3 Read the bits of the DIP switches, split it into 2 nibbles and display their sum.

4.3.1 Implementation:

1. Initial part is same as the previous task
2. for adding the two nibbles, we store the DIP Switch readings as two separate nibbles (and accordingly shift them to get proper 4 bit numbers)
3. We then add them and display the sum through LEDs

4.3.2 Code:

```
1
2   #include "LPC214x.H"       /* LPC214x definitions */
```

```
3  #include "led.h"
4  #include "delay.h"
5
6  #define DIP_SW_D0 (1 << 16)      // P0.16
7  #define DIP_SW_D1 (1 << 17)      // P0.17
8  #define DIP_SW_D2 (1 << 18)      // P0.18
9  #define DIP_SW_D3 (1 << 19)      // P0.19
10
11 #define DIP_SW_D4 (1 << 22)      // P0.22
12 #define DIP_SW_D5 (1 << 23)      // P0.23
13 #define DIP_SW_D6 (1 << 24)      // P0.24
14 #define DIP_SW_D7 (1 << 25)      // P0.25
15
16 #define DIP_SW_DIR       IO1DIR
17 #define DIP_SW_PIN       IO1PIN
18
19 #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)
20
21 void set_dipswitch_port_input( void )
22 {
23     DIP_SW_DIR &= ~(DIP_SW_DATA_MASK);
24 }
25
26 unsigned long read_dip_switch( void )
27 {
28     return DIP_SW_PIN;
29 }
30
31 int main (void)
32 {
33     unsigned long sw_status;
34
35     set_led_port_output();
36     set_dipswitch_port_input();
37
38     while(1)
39     {
40         sw_status = read_dip_switch();
41     int first_nibble = sw_status & 0x000F0000;
42     int second_nibble = (sw_status>>6) & 0x000F0000;
43     int sum = first_nibble + second_nibble;
```

```
    sum = sum >> 16;

    if(sum & 0x00000001){ LED5_ON;} else{ LED5_OFF;}
    if(sum & 0x00000002){ LED4_ON;} else{ LED4_OFF;}
    if(sum & 0x00000004){ LED3_ON;} else{ LED3_OFF;}
    if(sum & 0x00000008){ LED2_ON;} else{ LED2_OFF;}
    if(sum & 0x00000010){ LED1_ON;} else{ LED1_OFF;}
    { LED6_OFF;}
    { LED7_OFF;}
    { LED8_OFF;}

    delay_mSec(10);


    }

//    return 0;
}
```

4.4 Read the bits of the DIP switches, split it into 2 nibbles and display their product.

4.4.1 Implementation:

Everything is the same as the previous task. We just need to do multiplication instead of summation and display the value through the LEDs

4.4.2 Code:

```

#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
```

```
14 #define DIP_SW_D7 (1 << 25)     // P0.25
15
16 #define DIP_SW_DIR      IO1DIR
17 #define DIP_SW_PIN      IO1PIN
18
19 #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)
20
21
22 void set_dipswitch_port_input( void )
23 {
24     DIP_SW_DIR &= ~(DIP_SW_DATA_MASK);
25 }
26
27 unsigned long read_dip_switch( void )
28 {
29     return DIP_SW_PIN;
30 }
31
32 int main (void)
33 {
34     unsigned long sw_status;
35
36     set_led_port_output();
37     set_dipswitch_port_input();
38
39     while(1)
40     {
41     sw_status = (read_dip_switch());
42     int first_nibble = sw_status & 0x000F0000;
43     first_nibble=first_nibble>>16;
44     int second_nibble = (sw_status>>6) & 0x000F0000;
45     second_nibble=second_nibble>>16;
46     int sum = first_nibble * second_nibble;
47
48     if(sum & 0x00000001){ LED8_ON;} else{ LED8_OFF;}
49     if(sum & 0x00000002){ LED7_ON;} else{ LED7_OFF;}
50     if(sum & 0x00000004){ LED6_ON;} else{ LED6_OFF;}
51     if(sum & 0x00000008){ LED5_ON;} else{ LED5_OFF;}
52     if(sum & 0x00000010){ LED4_ON;} else{ LED4_OFF;}
53     if(sum & 0x00000020){ LED3_ON;} else{ LED3_OFF;}
54     if(sum & 0x00000040){ LED2_ON;} else{ LED2_OFF;}
```

```
55      if(sum & 0x00000080){ LED1_ON;} else{ LED1_OFF;}
56
57      delay_mSec(10);
58
59      }
60
61  //     return 0;
62  }
```

4.5 Photos

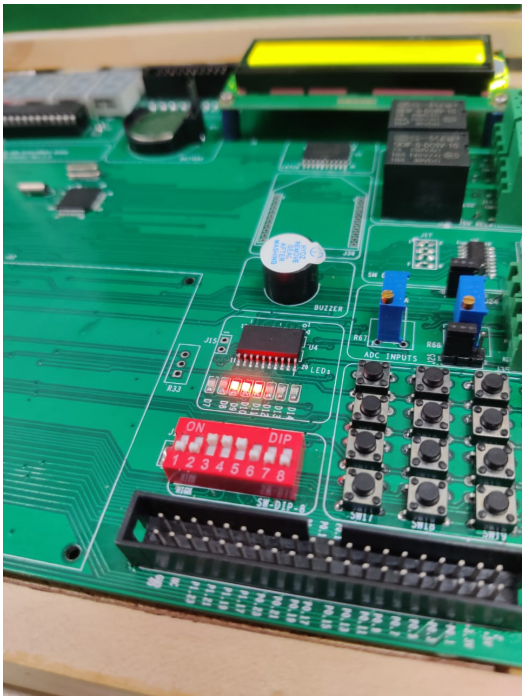

Figure 1: Displaying the status of each DIP switch

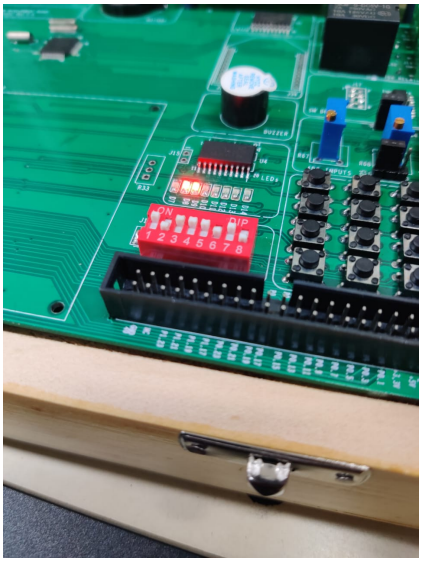

Figure 2: Adder test case A (1010 + 0010 = 1100)

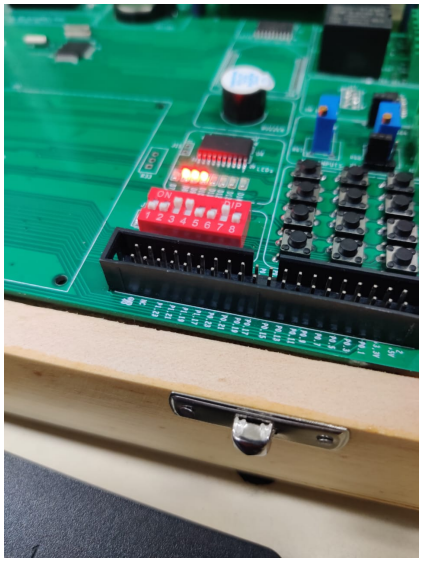

Figure 3: Adder test case B (1011 + 0011 = 1110)

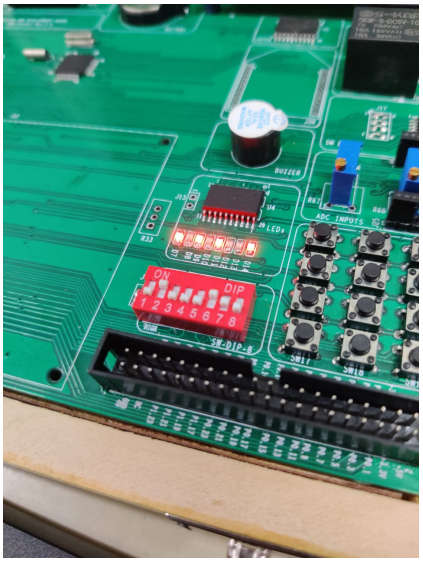

Figure 4: Multiplier test case A (1101 * 1101 = 10101001)

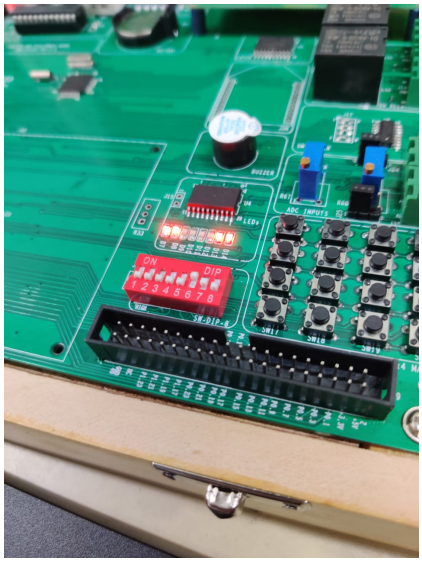

Figure 5: Multiplier test case B (1111 * 1101 = 11000011)

5 Working with a Stepper Motor:

5.1 Reversing the Direction:

5.1.1 Implementation:

1. we first set pins PO.0 - PO.15 as GPIO Ports and set pins P0.4 to P0.7 as output using PINSEL0 and IO0DIR respectively.

2. the code *StpprMtrCntrlInLPC2148Mdle.c* provided in Moodle makes the motor move in one direction. However, if we invert the array, we can make the motor move in the other direction (motor coil activation is done in reverse order now).

5.1.2 Code:

```
1
2  /*5V Stepper Motor
3
4  Connector J16 connect with stepper motor  as per below
       mentioned configuration
5
6  Pin - 1  : BLUE
7  Pin - 2  : PINK
8  Pin - 3  : YELLOW
9  Pin - 4  : ORANGE
10 Pin - 5  : Red (Motor Vcc)
11
12 Motor Pins:
13 P0.4 to P0.7
14 */
15
16 #include <LPC214x.h>                          /* LPC21xx
       definitions */
17
18 void delay_mSec(int);
19
20 int main (void)
21 {
22      int i;
23 unsigned char steps[4] = {0x03, 0x06, 0x0c, 0x09};  //
       standard step sequence for stepper motor
24 signed char x = 0;
```

```
25
26
27      PINSEL0 = 0x0;
                  // Pin function Select -> P0.0 to P0.15
           -> GPIO Port
28      IO0DIR |= 0xF0;                          // Set stepper
            motor pins as output in IO0 port
29      delay_mSec(10);
30
31      while(1)
32      {
33      for(i=0;i<2500;i++)
34      {
35          IO0PIN = (steps[x++] << 4); //send the 4 bit
               step value to stepper motor lines connected
               to IO0 port
36          if(x > 3)
37              x = 0;
38
39          delay_mSec(2);
40      }
41      }
42      return 0;
43  }
44
45  void delay_mSec(int dCnt)        // pr_note:~dCnt mSec
46  {
47    int j=0,i=0;
48    while(dCnt--)
49    {
50        for(j=0;j<1000;j++)
51        {
52          /* At 60Mhz, the below loop introduces
53          delay of 10 us */
54          for(i=0;i<10;i++);
55        }
56    }
57  }
58
59
60  }
```

Please click the link to view the video of the motor moving as per the one given in Moodle and the reverse direction

5.2 Gauging the Speed of the Motor:

5.2.1 Implementation:

1. We can change the speed of the motor by changing the delay between changing steps.

2. We need to change this delay within limits only as if we reduce the delay too much, the motor won't have time to move to the next step before it changes, or if it is too slow, the motion won't be smooth anymore.

5.2.2 Code:

```
1
2  /*5V Stepper Motor
3
4  Connector J16 connect with stepper motor  as per below
       mentioned configuration
5
6  Pin - 1  : BLUE
7  Pin - 2  : PINK
8  Pin - 3  : YELLOW
9  Pin - 4  : ORANGE
10 Pin - 5  : Red (Motor Vcc)
11
12 Motor Pins:
13 P0.4 to P0.7
14 */
15
16 #include <LPC214x.h>                          /* LPC21xx
       definitions */
17
18 void delay_mSec(int);
19
20 int main (void)
21 {
22      int i;
23 unsigned char steps[4] = {0x03, 0x06, 0x0c, 0x09};  //
       standard step sequence for stepper motor
24 signed char x = 0;
25
26
27     PINSEL0 = 0x0;
                 // Pin function Select -> P0.0 to P0.15
```

```
        -> GPIO Port
28      IO0DIR |= 0xF0;                         // Set stepper
            motor pins as output in IO0 port
29      delay_mSec(10);
30
31      while(1)
32      {
33      for(i=0;i<2500;i++)
34      {
35          IO0PIN = (steps[x++] << 4); //send the 4 bit
               step value to stepper motor lines connected
               to IO0 port
36          if(x > 3)
37              x = 0;
38
39          delay_mSec(4);  // Slow
40          //delay_mSec(1); // Fast
41      }
42      }
43      return 0;
44  }
45
46  void delay_mSec(int dCnt)       // pr_note:~dCnt mSec
47  {
48    int j=0,i=0;
49    while(dCnt--)
50    {
51        for(j=0;j<1000;j++)
52        {
53          /* At 60Mhz, the below loop introduces
54          delay of 10 us */
55          for(i=0;i<10;i++);
56        }
57    }
58  }
59
60
61  }
```

5.3 Max Speed of the Motor:

5.3.1 Implementation:

1. To find the maximum speed, we reduced the delay in steps of 0.01 mSec until the motor stopped working properly.

2. At a delay of 0.84 the motor reached the maximum speed of 1200 pulses or 6 rotations per second approximately.

5.3.2 Code:

The code remains the same except for changing the delay to 0.83 mSec.

5.4 Number of Steps required for 360 deg rotation:

5.4.1 Implementation:

1. As the step angle of the motor is 1.8 degrees, the number of steps required for 360 deg rotation is 200 (360/step angle).

6 Interpretations of result

1. Interfacing of DIP Switch, LED and stepper motor using C coding and ARK's LPC2148 embedded ARM development board was demonstrated in this experiment.

Note: I handled the stepper motor part of the assignment

7 References:

Handouts and the Code provided in Moodle