EE2016: Microprocessors Lab
Experiment # 7: Introduction to the LPC2378 Microcontroller and Interfacing a Stepper Motor with it

Amizhthni PRK, EE21B015
Nachiket Dighe, EE21B093

November 6, 2022

Contents

List of Figures

1 PART A - Introduction to the LPC2378 Microcontroller

1.1 PART A.1

1.2 Aim of this experiment:

To get acquainted with the LPC2378 microcontroller, from NXP semiconductors. LPC stands for Low Power Consumption. The ARM core here is called the ARM7TDMI-S (where T stands for Thumb Instructions, D for on-chip debugging support, M for multiplier, I for embedded In circuit emulation hardware and S for the synthesizable option based on RTL provided).

Specific goals:

- To cause the LEDs on the ARM-board to blink.

LPC2378 block diagram is as given in 10

1.3 Algorithm and Approach

The method we are using is to count number of 1's and output 1 if number of 1's are odd and output 0 if number of 1's are even.

- We write a code that causes delay between when the LEDs are on and off
- We set a delay of 65026 (0xFF)**2+1, this is an arbitary value
- The loop contains an empty bracket element to cause a delay simply
- We set the direction register to outputs for certain pins
- Here all 8 pins have been assigned high, meaning all pins glow on activation
- All pins show outputs
- When input given is 0XFF00F00F, we see that only those pins set to F glow.
- Since we've given 1 under conditionals, this means that the value remains true forever and the LED blink on and off goes on until power is turned off
- This command sets the pins which where mentioned in the direction register to high, that is they glow
- The LEDS glow for the duration given under delay
- Once the LED stays on for the given duration, we set the pins to low.
- Now the LEDS stay dim for the same duration
- VIDEO: https://drive.google.com/file/d/1kIx1ZgDO5i4BRtusLORKKcBs2WM9v-e-/view?usp=sharing

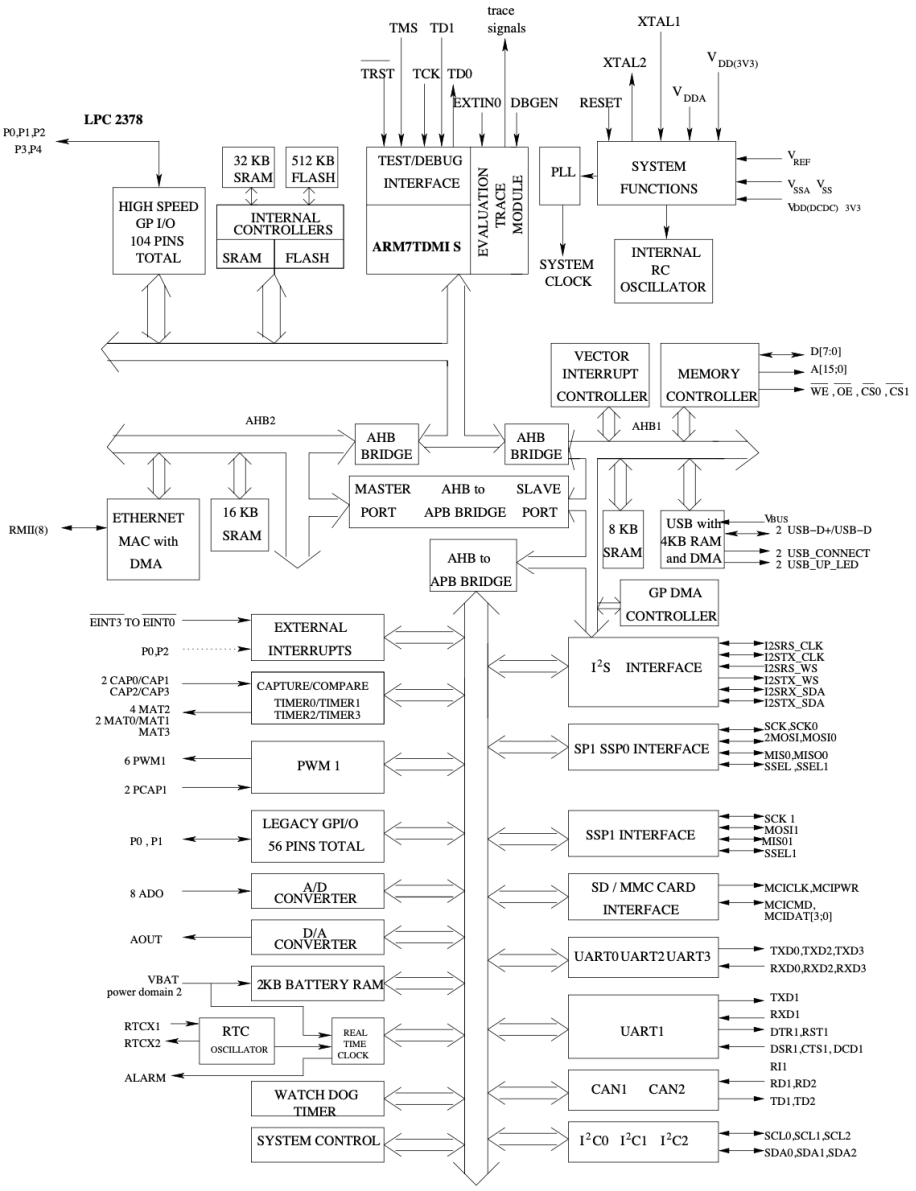


Figure 1: LPC2378 block diagram

1.4 Code

The code for blinking a given set of LEDs on the LPC board is given in listing 1.

```
#include "LPC23xx.h" //to input the module of the given LPC board


void delay(void)//We write a code that causes delay between when the LEDs are on and off
{
        int i;//initialise variable in int
        for(i=0;i<0xfe01;i++)// we set a delay of 65026 {(0xFF)**2+1}, this is an arbitary value
        {} //simply to cause delay
}


int main(void)
{
        FIO3DIR = 0xFFFFFFFF;//we set the direction register to outputs for certain pins
                             //Here all 8 pins have been assigned high, meaning all pins glow on activation
                             //All pins show outputs
                             //when input given is 0XFF00F00F, we see that only those pins set to F glow.
        while(1)//Since we've given 1 under conditionals, this means that the value remains true forever and the LED blink on and off goes on until power is turned off
        {
                        FIO3PIN = 0xFF; //This pin sets the pins which where mentioned in the direction register to high, that is they glow
                        delay();// The LEDS glow for the duration given under delay
                        FIO3PIN = 0x00;//Once the LED stays on for the given duration, we set the pins to low.
                        delay();//Now the LEDS stay dim for the same duration
        }
        return 0;
}
```

Listing 1: Code for blinking some LEDs on the ARM board

1.5 Outputs

Figure 2: LEDS in ON status, with time stamp of video

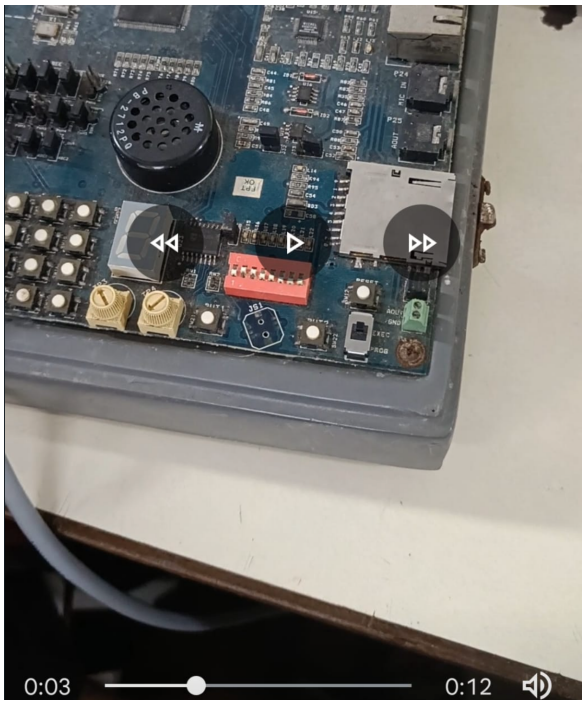


Figure 3: LEDS in OFF status, with time stamp of video

1.6 PART A.2

1.7 Aim of this experiment:

In the previous task, we focussed on output on the LEDs. In this task, we will take input. In particular, read the settings of an 8-way DIP (Dual Inline Package) switch and display it on the LEDs. Use FIO4DIR AND FIO4PIN for data input.

1.8 Algorithm

- Initialise a Temporary variable that stores status of required input pin, from the DIP SWITCH.
- FIO4DIR = 0x00;FI04DIR refers to the input taken from the DIP Switches.
- This data is then fed to the LEDs through the FI03 direction registers.
- FIO3DIR = 0xFF; FIO3DIR refers to the output to be shown on the LEDs, it has been set to high FF because of Output.
- Loop goes on until Power is turned off
- The temporary variable stores the status of the input DIP Switch.
- The same value from the variable is transferred to the output pins that is the LEDs.
- VIDEO: https://drive.google.com/file/d/1fXVCcEZheliiHfjTi8q-ZMuQyEUaRhyn/view?usp=sharing

1.9 Code

The code used for controlling the blinking of an LED by a DIP Switch is given under 2.

```
include "LPC23xx.h" //including LPC238 MODULE
//code by AMIZHTHNI AND NACHIKET DIGHE

int main()

{
    int TEMP; //Temporary variable that stores status of required input pin
    FIO3DIR = 0xFF;//FIO3DIR refers to the output to be shown on the LEDs,
    ↪ it has been set to high FF because of Output
    FIO4DIR = 0x00;//FIO4DIR refers to the input taken from the DIP
    ↪ Switches.


    while(True) //Loop goes on until Power is turned off
    {
        TEMP = FIO4PIN; //The temporary variable stores the status of the
    ↪ input DIP Switch
        FIO3PIN = TEMP; //The same value from the variable is transferred
    ↪ to the output pins that is the LEDs.
    }
    return 0;
}
```

Listing 2: Code for controlling the blinking of an LED by a DIP Switch

1.10 Inference

- When the DIP Switch is set to high, i.e 1. It glows much brighter than normal
- When set to low / 0, it looks dimmed out, giving out a faint glow.
- The video illustrates this best.

1.11 Output figures

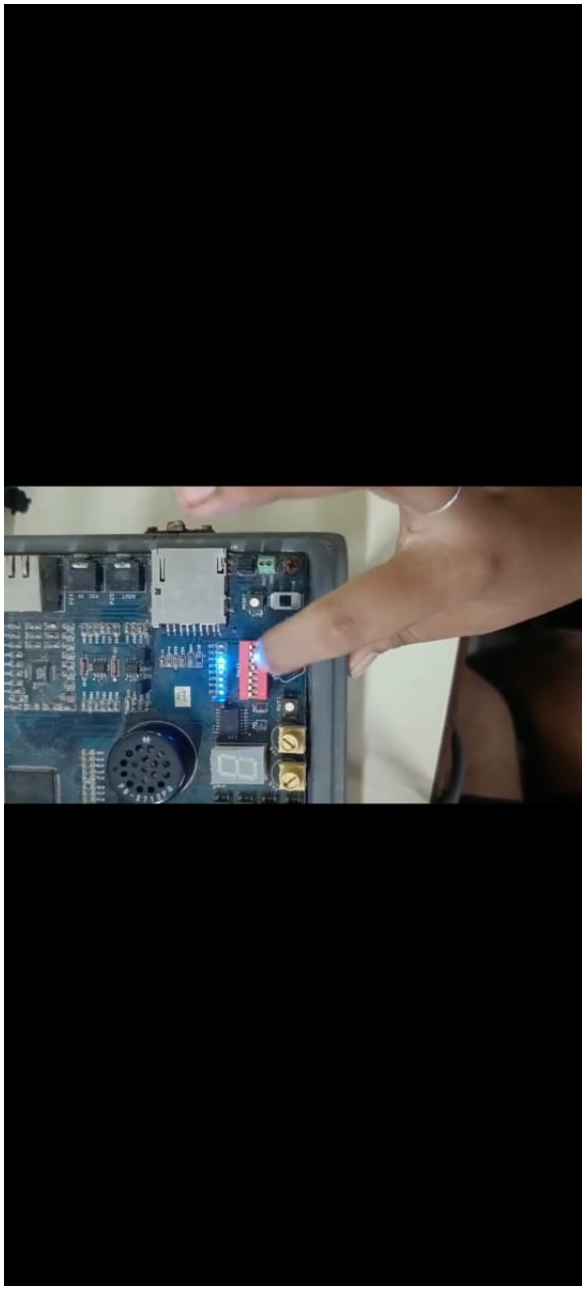


Figure 4: Output for controlling the blinking of an LED by a DIP Switch

1.12 PART A.3

Multiplying inputs and showing on LEDs

1.13 Aim of this experiment:

- Write a C program to read a DIP switch, split into two nibbles (4 bits), multiply them and display the product on the LEDs.

1.14 Code

The code used for multiplying lower and upper nibbles and outputting on LEDs 3.

```
include "LPC23xx.h"                  //including LPC238 MODULE
//code by AMIZHTHNI AND NACHIKET DIGHE
  int
main ()
{

  FIO3DIR = 0xFF;                  //FIO3DIR refers to the output to be shown
↪  on the LEDs,
//it has been set to high FF because of Output
  FIO4DIR = 0x00;                  //FIO4DIR refers to the input taken from
↪  the DIP Switches

  int TEMP, a, b, PROD;                  //Temporary variable that stores
↪  status of required input pin
  //A and B are the two nibbles
  //PROD stores the product

  while (True)                           //Loop goes on until Power is turned
↪  off
    {
      TEMP = FIO4PIN;                  //The temporary variable stores the
↪  status of the input DIP Switch
      a = TEMP & 0xF0;                  //The & (bitwise AND) in C or C++
↪  takes two numbers as operands and does AND on every bit of two numbers.
↪  The result of AND is 1 only if both bits are 1.
      //We have been given two nibbles, that is 8 bits which corresponds to
↪  just two in hexadecimal
      //When and operation is used, if a is say 1001 1001, this 1001 will
↪  be 9 and first of hexadecimal. and 1001 will again 9. and the second
↪  bit is 9 too. Thus total value is 99.
      //Now,  when 0XF0 & A= 0XF0 & 0X99. Since the operation is bitwise,
↪  only 0X90 is stored.
      a = a >> 4;                  //Takes two numbers, right shifts the bits
↪  of the first operand, the second operand decides the number of places
↪  to shift. In other words right shifting an integer b\xb\ with an
↪  integer b\yb\ denoted as b\(x>>y)b\ is equivalent to dividing x with
↪  2^y.
      //This shifts the bits by four positions to the right
```

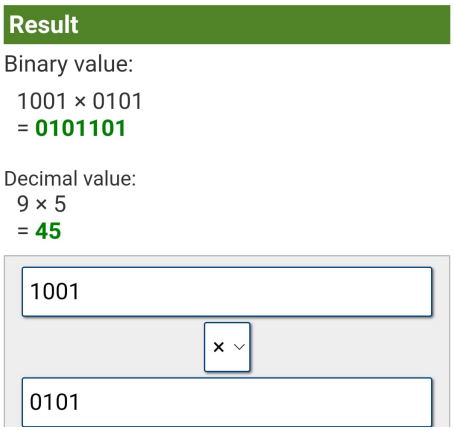
```
24        //From oour example we have. a=0X90. right shift by four places in
    ↪  binary corresponds to shift by 1 in hexadecimal. Thus A becoomes 0X9.
    ↪  Thus,  the upper nibble is stored in A
25
26        b = TEMP & 0x0F;
27        //let TEMP=0x99. now, TEMP & 0X0F is 0X09. This directly is the lower
    ↪  nibble, and does. not require shifting
28        //Now, clearly, a and b store the upper and lower nibbles
    ↪  respectively, and all that is left is normal mulitplication
29        PROD = a * b;                 //Normal multiplication operation of the
    ↪  variables storing the upper and lower nibbles
30        FIO3PIN = PROD;                 //The same value from the variable is
    ↪  transferred to the output pins that is the LEDs.
31      }
32    return 0;
33  }
```

Listing 3: Multiplying lower and upper nibbles and outputting on LEDs

1.15 Approach and Code Algorithm

- FIO3DIR refers to the output to be shown on the LEDs, it has been set to high FF because of Output.
- FI04DIR refers to the input taken from the DIP Switches.
- Temporary variable that stores status of required input pin.
- A and B are the two nibbles.
- PROD stores the product.
- Loop goes on until Power is turned off.
- The temporary variable stores the status of the input DIP Switch.
- The (bitwise AND) in C or C++ takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1.
- We have been given two nibbles, that is 8 bits which corresponds to just two in hexadecimal.
- When and operation is used, if a is say 1001 1001, this 1001 will be 9 and first of hexadecimal. and 1001 will again 9. and the second bit is 9 too. Thus total value is 99.
- Now, when 0XF0 A= 0XF0 0X99. Since the operation is bitwise, only 0X90 is stored.
- Takes two numbers, right shifts the bits of the first operand, the second operand decides the number of places to shift.
- This shifts the bits by four positions to the right.
- From our example we have. a=0X90. Right shift by four places in binary corresponds to shift by 1 in hexadecimal. Thus A becomes 0X9.

(a) Inputs: 1001, 1001



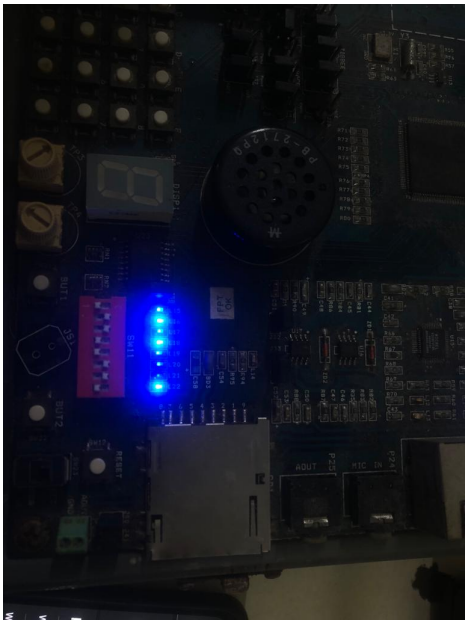
(b) Corresponding output

Figure 5: Three Different Outputs

- As a result, the upper nibble is stored in A
- Let TEMP=0x99. Now, TEMP 0X0F is 0X09. This directly is the lower nibble, and does. not require shifting
- Now, clearly, a and b store the upper and lower nibbles respectively, and all that is left is normal multitplication.
- Normal multiplication operation of the variables storing the upper and lower nibbles
- The same value from the variable is transferred to the output pins that is the LEDs.

1.16 Inferences

- The first output is explained below here:
- The input given by us in the DIP controller is 10010101.
- The two required nibbles are 1001 and 0101.
- Their product is 45 = 00101101. This is outputted in the LED switches. The dim LEDs represent 0 and the brighter ones represent 1. Light intensity determines whether it is 0 or 1.



(a) Inputs: 1001, 0101

Result

Binary value:

1001 × 1001
= **01010001**

Decimal value:

9 × 9
= **81**

1001

×

1001

(b) Corresponding output

(c) Inputs: 1001, 1011

Result

Binary value:

1001 × 1011
= **01100011**

Decimal value:

9 × 11
= **99**

1001

×

1011

(d) Corresponding output

Figure 6: Three Different Outputs

2 PART B - Stepper motor interfacing

- In this experiment, we will interface a stepper motor to the LPC2378 microcontroller. We will begin by describing how a stepper motor works.
- We use a *Permanent Magnet Stepper motors*.
- The electromagnets are energized through the contact pins of the microcontroller. To make the motor shaft rotate, the stator is given an excitation, which attracts the rotor teeth (magnetically) to the stator electromagnet.
- Figure is as shown below.
- The lab experiment will be performed using the Wave Drive control of Stepper motor.
- Only one winding is energized per sequence
- Pulsed waveforms in the correct sequence are used to create the electromagnetic fields required to drive the motor.
- Pulsed waveforms in the correct sequence are used to create the electromagnetic fields required to drive the motor. The stepper motor driver circuit has two major tasks:

(1) To change the current and flux direction in the phase windings. This is achieved by taking a centre-tapped wire connection between each pair of phase windings and is termed as a Unipolar connection.

(2) To drive a controllable amount of current and adjust the drive sequence according to the speed requirements.

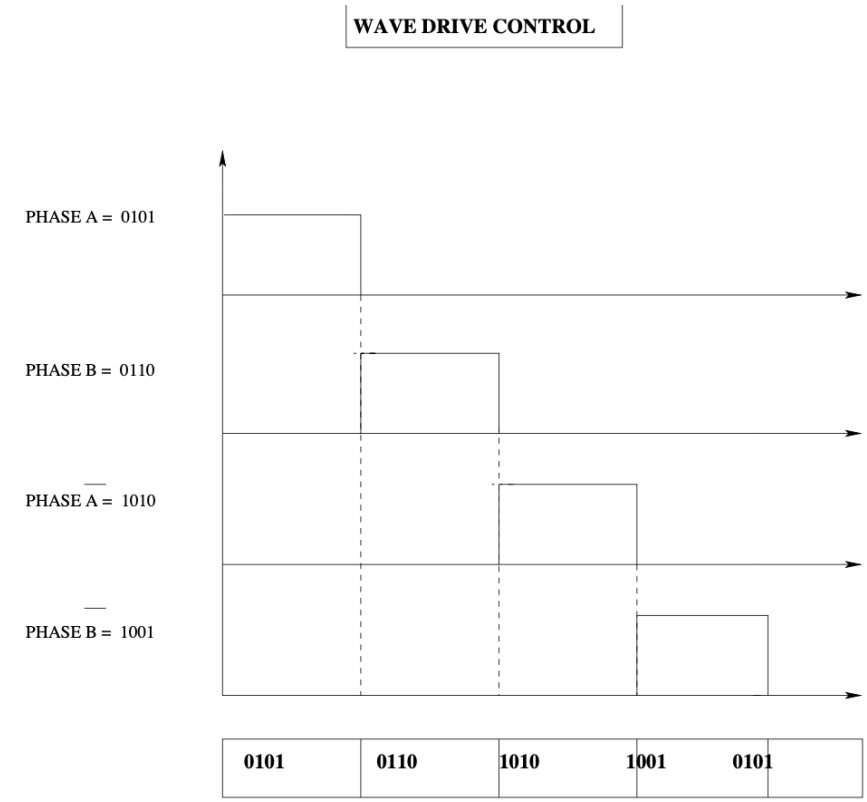


Figure 7: Wave Drive control

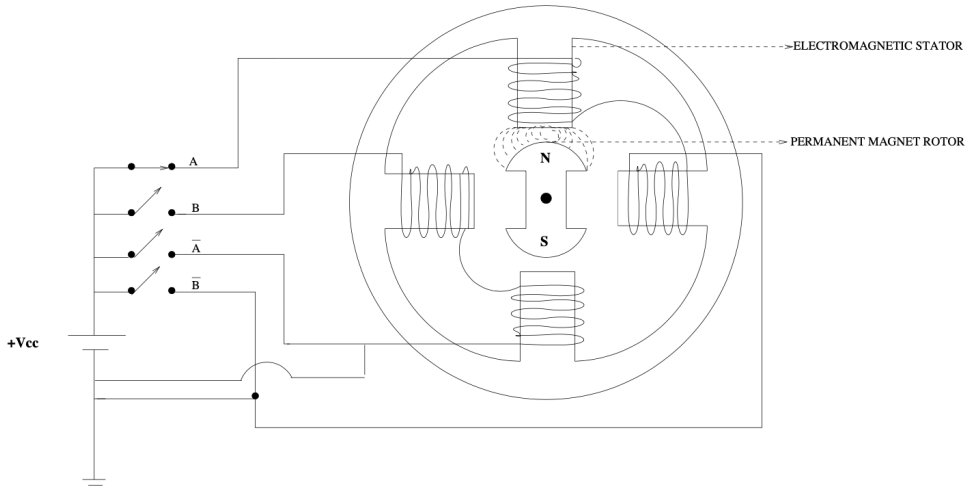


Figure 8: Stepper motor working

2.1 PART B.1

2.2 Aim of this experiment:

The first task in this experiment involves completion of the program given below to make the motor rotate in a specific direction at a fixed speed.

2.3 Algorithm and Code explanation

- The upper limit of the for loop iterations determines the duration of delay
- The direction register IODIR0 has been set to output at all of the 8 pins.
- There are 4 states in Wave drive control.Square-wave drive is the simplest drive method. It rotates a rotor by switching the ON/OFF state of the power element according to the rotation angle of the rotor and then changing the current direction of the stator coil.

  The 4 states are: 0101,0110,1010,1001=(5,6,10,9)
- Assign the phase sequence for the stator to enable the motor to rotate in a given direction. In particular, use the IOPIN0 register to set the phase excitation
- The inputs we give in C code can be understood this way:
- 0x00000280 = 0000 0000 0000 0000 0000 0010 1000 0000 ← 10
- 0x00000240 = 0000 0000 0000 0000 0000 0010 0100 0000 ← 9
- 0x00000140 = 0000 0000 0000 0000 0000 0001 0100 0000 ← 5
- 0x00000180 = 0000 0000 0000 0000 0000 0001 1000 0000 ← 6
- Hence the LSBs of 3rd nibble and MSBs of 2nd nibble decide the waveform used as input for stepper motor. We thus control the direction of motor rotation by the order in which these inputs are given to the motor.
- These phases in this particular example rotate the motor in the clockwise sense.

2.4 Code

The code used for rotating a stepper motor is given under 4.

```
1  // ARM- to turn stepper motor and complete rotations, using C code
2
3  #include "LPC23xx.h"                  //invoking the modules of the required
   ↪  arm board //Include the header file(s)
4
5  void
6  delay ()                          //This delay() statement determines how
   ↪  long the motor stays in a particular direction/state.
7  {
8    int tim;
9    //variable initialisation
10   for (tim = 0; tim < 0xFFFF; tim++)
11     {
```

```
12      }
13    //The upper limit of the for loop iterations determines the duration of
      ↪  delay
14  }
15
16  int
17  main ()
18  {
19    IODIR0 = 0xFFFFFFFF;                  //The direction register IODIR0 has
      ↪  been set too output at all of the 8 pins
20
21    while (1)                           //Loop goes on until power goes off
      ↪  //execute this loop for ever (until a break is encountered
22      //     or an interrupt occurs)
23      {
24        //There are 4 states in Wave drive control.Square-wave drive is the
      ↪  simplest drive method. It rotates a rotor by switching the ON/OFF state
      ↪  of the power element according to the rotation angle of the rotor and
      ↪  then changing the current direction of the stator coil.
25        //The 4 states are: 0101,0110,1010,1001=(5,6,10,9)
26
27        //Assign the phase sequence for the stator to enable the motor
28  //  to rotate in a given direction. In particular, use the
29        // IOPIN0 register to set the phase excitation.
30        // (Refer wave drive control diagram attached).
31        IOPIN0 = 0x00000280;          //Excitation for Phase 1
32        //  Select bits 6 to 9 of IOPIN0 (count from 0)
33        //  to enable phase excitation
34        delay ();
35        //This delay() statement determines how long the motor stays in a
      ↪  particular direction/state.
36        IOPIN0 = 0x00000240;          //Excitation for Phase 2
37        delay ();
38        //This delay() statement determines how long the motor stays in a
      ↪  particular direction/state.
39        IOPIN0 = 0x00000140;          // Excitation for Phase 3
40        delay ();
41        //This delay() statement determines how long the motor stays in a
      ↪  particular direction/state.
42        IOPIN0 = 0x00000180;          // Excitation for Phase 4
43        delay ();
44        //This delay() statement determines how long the motor stays in a
      ↪  particular direction/state.
45
46      }
47    return 0;
48  }
```

Listing 4: Code for rotating a stepper motor

2.5 Output Figures

:

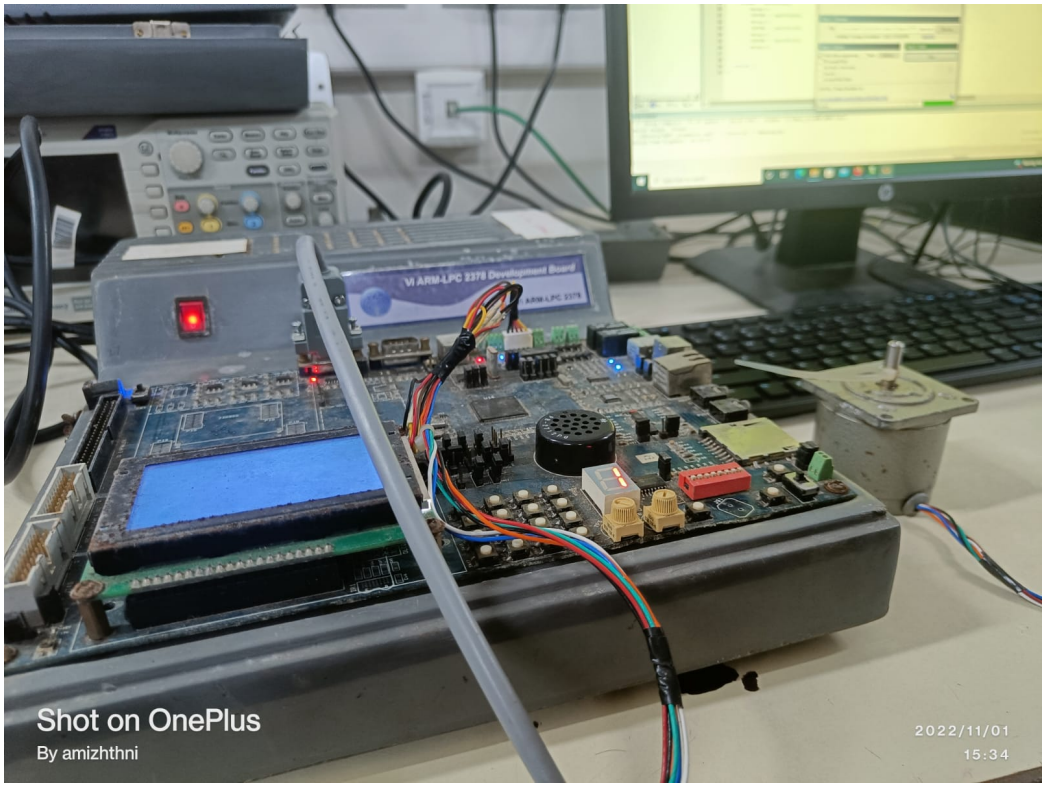


Figure 9: Motor rotation

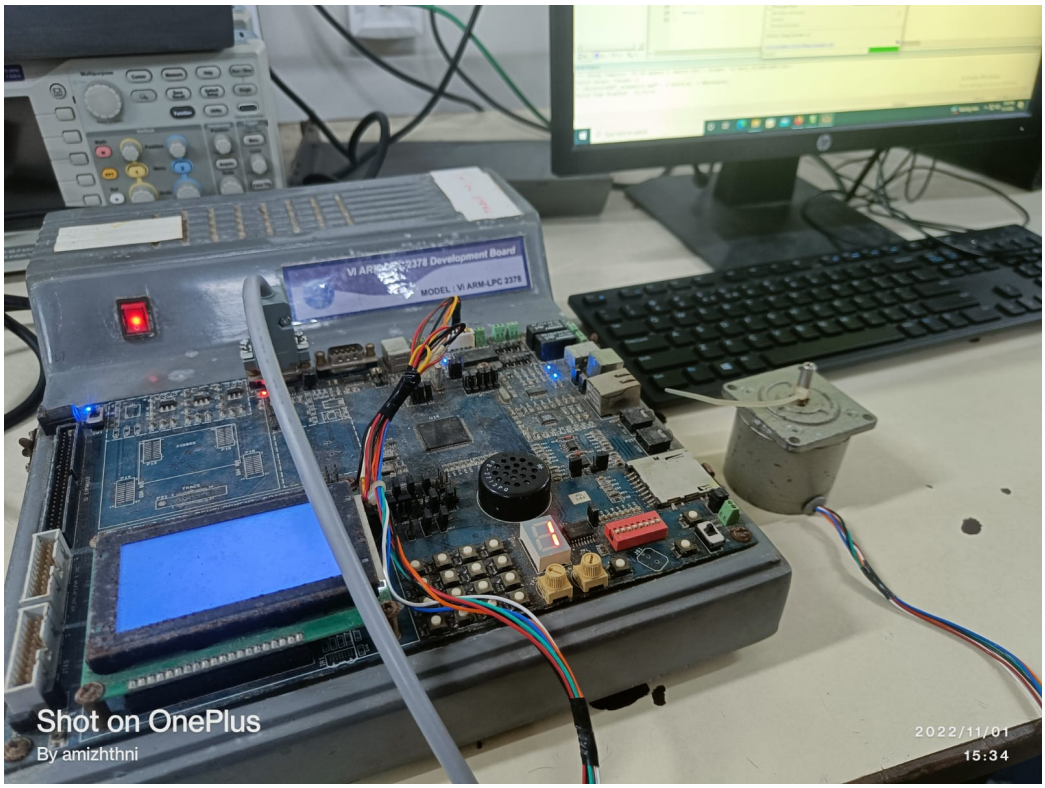


Figure 10: Motor rotation

2.6 PART B.2

2.7 Aim of this experiment:

Modify the program given in Task 1 to cause rotation of the stepper motor in both clockwise and anti-clockwise directions. That is, the motor should make a few rotations in clockwise direction, stop and then make a few rotations in the anti-clockwise direction.

2.8 Algorithm and Code explanation

- The upper limit of the for loop iterations determines the duration of delay
- The direction register IODIR0 has been set to output at all of the 8 pins.
- There are 4 states in Wave drive control.Square-wave drive is the simplest drive method. It rotates a rotor by switching the ON/OFF state of the power element according to the rotation angle of the rotor and then changing the current direction of the stator coil.

  The 4 states are: 0101,0110,1010,1001=(5,6,10,9)
- Assign the phase sequence for the stator to enable the motor to rotate in a given direction. In particular, use the IOPIN0 register to set the phase excitation
- The inputs we give in C code can be understood this way:
- 0x00000280 = 0000 0000 0000 0000 0000 0010 1000 0000 ← 10
- 0x00000240 = 0000 0000 0000 0000 0000 0010 0100 0000 ← 9

- 0x00000140 = 0000 0000 0000 0000 0000 0001 0100 0000 ← 5
- 0x00000180 = 0000 0000 0000 0000 0000 0001 1000 0000 ← 6
- Hence the LSBs of 3rd nibble and MSBs of 2nd nibble decide the waveform used as input for stepper motor. We thus control the direction of motor rotation by the order in which these inputs are given to the motor.
- These phases in this particular example rotate the motor in the clockwise sense.
- We then write the same code with the phases in the opposite order.
- This makes sure that the motor rotates in the opposite direction, which is anti clockwise in this direction.
- We set x as 55. It takes 55 steps by the stepper motor to complete one revolution so here we made the stepper. This observation was made by taking various values of x and the best estimate was considered.
- We first follow the order 280,240,140,180.
- And then follow 180,140,240,280.
- Clearly the output is when the motor rotates a near full rotation. in the clockwise direction and turns around and completes a full rotation in the anticlockwise direction.

2.9 Code

The code used for rotating a stepper motor is given under 5.

```
/* ARM C program to turn stepper motor in clockwise and anti clockwise
 ↪  direction alternatively*/

#include "LPC23xx.h"//Include the header file

static void delay()
//This delay() statement determines how
//long the motor stays in a particular direction/state.
{
    int i;
    //variable initialisation
    for(i=0;i<0x0FFF;i++)
    //The upper limit of the for loop iterations determines the duration of
        //delay
    {}
}

int main ()

{
  IODIR0 = 0xFFFFFFFF;
  //The direction register IODIR0 has
//been set too output at all of the 8 pins
  int x = 55;//We set x as 55.
```

```
24   //It takes 55 steps by the steppermotor to complete one revolution so
     ↪ here we made the stepper. This observation was made by taking various
     ↪ values of x and the best estimate was considered.
25   //to move for 55 steps in clockwise direction and then in anticlockwise
     ↪ direction.
26   while(x>0)
27   {
28     IOPIN0 = 0x00000280;
29     //Excitation for Phase 1
30         delay();
31         //This delay() statement determines how long the motor stays in a
     ↪ particular direction/state/phase.
32         IOPIN0 = 0x00000240;
33         //Excitation for Phase 2
34         delay();
35         //This delay() statement determines how long the motor stays in a
     ↪ particular direction/state/phase.
36         IOPIN0 = 0x00000140;
37         //Excitation for Phase 3
38         delay();
39         //This delay() statement determines how long the motor stays in a
     ↪ particular direction/state/phase.
40         IOPIN0 = 0x00000180;
41         //Excitation for Phase 4
42         delay();
43         //This delay() statement determines how long the motor stays in a
     ↪ particular direction/state/phase.
44                 x--;
45                 //decrements, to make sure there are just 55 steps, for a
     ↪ given delay.
46
47   }
48          x = 55;
49   while(x>0)
50   {
51       /*NOTE THAT THE ORDER OF PHASES HAVE BEEN INVERTED SO AS TO MAKE THE
     ↪ MOTOR ROTATE IN THE OPPOSITE DIRECTION*/
52
53     IOPIN0 = 0x00000180;
54     //Excitation for Phase 1
55         delay();
56         //This delay() statement determines how long the motor stays in a
     ↪ particular direction/state/phase.
57         IOPIN0 = 0x00000140;
58         //Excitation for Phase 2
59         delay();
60         //This delay() statement determines how long the motor stays in a
     ↪ particular direction/state/phase.
61         IOPIN0 = 0x00000240;
62         //Excitation for Phase 3
```

```
63          delay();
64          //This delay() statement determines how long the motor stays in a
    ↪  particular direction/state/phase.
65          IOPIN0 = 0x00000280;
66          //Excitation for Phase 4
67          delay();
68          //This delay() statement determines how long the motor stays in a
    ↪  particular direction/state/phase.
69                  x--;
70                  //decrements, to make sure there are just 55 steps, for a
    ↪  given delay.
71
72    }
73    return 0;
74  }
```

Listing 5: Code for rotating a stepper motor clockwise and anticlockwise

2.10 Output

Check the video: https://drive.google.com/file/d/1UCOtBGyYmUvfRnauzM4ed_-v_grWqFqW/view?usp=sharing

2.11 PART B.3

2.12 Aim of this experiment:

The program in Task 1 causes the motor to rotate at approximately 90 rpm. Write a program which will allow the motor to rotate at four different speeds. That is, it should rotate at (say) 30 rpm for one complete revolution, then at (say) 50 rpm for another revolution, then at 70 rpm and finally at 90 rpm for the last revolution

2.13 Algorithm and Code explanation

- The upper limit of the for loop iterations determines the duration of delay
- The direction register IODIR0 has been set to output at all of the 8 pins.
- There are 4 states in Wave drive control.Square-wave drive is the simplest drive method. It rotates a rotor by switching the ON/OFF state of the power element according to the rotation angle of the rotor and then changing the current direction of the stator coil.

  The 4 states are: 0101,0110,1010,1001=(5,6,10,9)
- Assign the phase sequence for the stator to enable the motor to rotate in a given direction. In particular, use the IOPIN0 register to set the phase excitation
- The inputs we give in C code can be understood this way:
- 0x00000280 = 0000 0000 0000 0000 0000 0010 1000 0000 ← 10
- 0x00000240 = 0000 0000 0000 0000 0000 0010 0100 0000 ← 9
- 0x00000140 = 0000 0000 0000 0000 0000 0001 0100 0000 ← 5
- 0x00000180 = 0000 0000 0000 0000 0000 0001 1000 0000 ← 6
- Hence the LSBs of 3rd nibble and MSBs of 2nd nibble decide the waveform used as input for stepper motor. We thus control the direction of motor rotation by the order in which these inputs are given to the motor.
- These phases in this particular example rotate the motor in the clockwise sense.
- **QUESTION SPECIFIC ALGORITHM:**
- We initiate five different *delay* functions.
- Each delay upper limit value gives us a different speed of rotation.
- **Thought process:** We first set upper limit as 0X0FFF. We then find the time it takes to finish a rotation, and convert it to RPM and figure out that it takes 6RPM.
- Now, to simplify the problem, we divide the upper limit by 5. This increases speed five fold. That is, the motor has to complete the same angle in 5 times lesser speed. Now RPM is 30.
- We then Multiply 0x033 by $\frac{3}{5}$ to get 0x01EB, which corresponds to 50RPM.
- We keep doing this to reach different RPMs of 70 and 90.

- We then use the same code from part1 but with different DELAY values to get the motor to rotate at different speeds but in the same direction.
- We set x as 55. It takes 55 steps by the stepper motor to complete one revolution so here we made the stepper. This observation was made by taking various values of x and the best estimate was considered.
- This has been arrived at by multiple trials and timing experiments. The output has been attached as a link below.

2.14 Code

The code used for rotating a stepper motor is given under 6.

```
/* ARM C program to turn stepper motor with increasing speeds*/

//code by AMIZHTHNI AND NACHIKET

#include "LPC23xx.h"//Include the header file

void delay1()
//This delay() statement determines how long the motor stays in a particular direction/state

//To get 6RPM
{
    int i;
    for(i=0;i<0x0FFF;i++) //6
    {}
}
void delay2()
//This delay() statement determines how long the motor stays in a particular direction/state

//To get 30 RPM
{
    int i;
    for(i=0;i<0x0333;i++) //30
    {}
}
void delay3()
//This delay() statement determines how long the motor stays in a particular direction/state

//To get 50 RPM
{
    int i;
    for(i=0;i<0x01EB;i++)  //50
    {}
}
void delay4()
```

```
//This delay() statement determines how long the motor stays in a particular direction/state

//To get 70 RPM
{
    int i;
    for(i=0;i<0x015F;i++) //70
    {}
}
void delay5()
//This delay() statement determines how long the motor stays in a particular direction/state

//To get 90 RPM
{
    int i;
    for(i=0;i<0x0111;i++) //90
    {}
}
int main ()

{
  IODIR0 = 0xFFFFFFFF;
  int x = 55;
  /*It takes 55 steps by the steppermotor to complete one revolution so
here we made the stepper. This observation was made by taking various
values of x and the best estimate was considered.
to move for 55 steps in clockwise direction and then in anticlockwise
direction*/
  while(x>0)
  {
                                IOPIN0 = 0x00000280;
        delay1();
        IOPIN0 = 0x00000240;
        delay1();
        IOPIN0 = 0x00000140;
        delay1();
        IOPIN0 = 0x00000180;
        delay1();
                                x--;

  }
        x = 55;
          /*It takes 55 steps by the steppermotor to complete one revolution so
here we made the stepper. This observation was made by taking various
values of x and the best estimate was considered.
to move for 55 steps in clockwise direction and then in anticlockwise
direction*/
        while(x>0)
```

```
  {
                                IOPIN0 = 0x00000280;
        delay2();
        IOPIN0 = 0x00000240;
        delay2();
        IOPIN0 = 0x00000140;
        delay2();
        IOPIN0 = 0x00000180;
        delay2();
                                x--;

  }
        x = 55;
          /*It takes 55 steps by the steppermotor to complete one
↪  revolution so
here we made the stepper. This observation was made by taking various
values of x and the best estimate was considered.
to move for 55 steps in clockwise direction and then in anticlockwise
direction*/
        while(x>0)
  {
                                IOPIN0 = 0x00000280;
        delay3();
        IOPIN0 = 0x00000240;
        delay3();
        IOPIN0 = 0x00000140;
        delay3();
        IOPIN0 = 0x00000180;
        delay3();
                                x--;

  }
        x = 60;
          /*It takes 55 steps by the steppermotor to complete one
↪  revolution so
here we made the stepper. This observation was made by taking various
values of x and the best estimate was considered.
to move for 55 steps in clockwise direction and then in anticlockwise
direction*/
        while(x>0)
  {
                                IOPIN0 = 0x00000280;
        delay4();
        IOPIN0 = 0x00000240;
        delay4();
        IOPIN0 = 0x00000140;
        delay4();
        IOPIN0 = 0x00000180;
        delay4();
                                x--;
```

```
130
131   }
132         x = 55;
133           /*It takes 55 steps by the steppermotor to complete one
      ↪  revolution so
134 here we made the stepper. This observation was made by taking various
135 values of x and the best estimate was considered.
136 to move for 55 steps in clockwise direction and then in anticlockwise
137 direction*/
138         while(x>0)
139   {
140                                   IOPIN0 = 0x00000280;
141         delay5();
142         IOPIN0 = 0x00000240;
143         delay5();
144         IOPIN0 = 0x00000140;
145         delay5();
146         IOPIN0 = 0x00000180;
147         delay5();
148                                   x--;
149
150   }
151   return 0;
152 }
```

Listing 6: Code for rotating a stepper motor in increasing speeds

2.15 Output

Check the video:
https://drive.google.com/file/d/1jaR0_pr4EE9P0S4y8kKHKS0-mWiD0gwe/view?usp=sharing

3 MASTER DRIVE LINK

Find below the link of all the photos, videos taken during the experiment, in an unordered manner.
https://drive.google.com/drive/folders/1ziGpOLeDdLC4GhkBBq1JBiryEp6GOm59?usp=share_link