EE2016 LAB EXPERIMENT 4 – STEPPER MOTOR AND LED CONTROL

EE20B056

AIM:

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) (ii) Stepper motor control using Vi Microsystem's ViARM 7238 development board. (EMULATION)

TASK 1 – LED CONTROL:

Write a program (in C) to dis-assemble a byte into two nibbles from the DIP switch states, multiply and display the product in the LED.

CODE:

```
#include "LPC23xx.h"


int main()

{

  int a;

  int highByte;

  int lowByte;

  FIO3DIR = 0xFF; // Fast GPIO PIN 3 is set to output (LEDs)

  FIO4DIR = 0x00; // Fast GPIO PIN 4 is set to input (DIP switches)

  while(1)

  {

    a = FIO4PIN; // Given byte read to ‘a’ through Fast GPIO PIN4

    highByte = a & 0xF0;

    highByte = highByte >> 4; // higher nibble extracted into ‘highbyte’ by masking and shifting

    lowByte = a & 0x0F; // lower nibble extracted into ‘lowByte’ by direct masking

    FIO3PIN = highByte * lowByte; // nibbles multiplied and result is put in PIN3 (LEDs)

  } //loop used to give live results as the inputs change

  return 0;

}
```

LOGIC: Get the given byte by reading the DIP switch states into a integer variable ‘a’ and disassemble them into higher nibble and lower nibble respectively into variables ‘highByte’ and ‘lowByte’ multiply and display the result in the fast GPIO pin 3 which light the LED appropriately.

(Step by Step program flow is explained in the comments)

TASK 2 – STEPPER MOTOR CONTROL:

Modify the demo code (StpprMtrCntrl.c) supplied to demonstrate the control of stepper motor to rotate in opposite direction.

CODE:

```
#include "LPC23xx.h"


void delay() // delay function for 256*256 clock cycles

{

   int i, j;

   for(i = 0;i < 0xFF; i++)

      for(j = 0;j < 0xFF; j++);

}


int main()

{

   IODIR0 = 0xFFFFFFFF; // GPIO PIN0 set to output which is connected to the stepper motor


   while(1) // loop is set for true always to get indefinite rotation

   {

      IOPIN0 = 0x00000280;

      delay();

      IOPIN0 = 0x00000240;

      delay();

      IOPIN0 = 0x00000140;

      delay();
```

```
        IOPIN0 = 0x00000180;

        delay();

// given pulse sequence is reversed and followed by delay as given to achieve the desired anti clock

// wise rotation

    }

    return 0;

}
```

LOGIC:

Stepper motor is controlled through the four pulses given successively one way clockwise and the other way anticlockwise by exploiting the slight offset between the teeth and the successive pulse electromagnets. By reversing the pulse sequence in the given program we can achieve counter-clockwise rotation.

(Step by Step program flow is explained in the comments)

TASK 3 – STEPPER MOTOR CONTROL(given angle rotation):

rotate your stepper motor 180° with the step size of 5°

CODE:

```
#include "LPC23xx.h"


void delay() // delay function for 256*256 clock cycles

{

    int i, j;


    for(i = 0;i < 0xFF; i++)

        for(j = 0;j < 0xFF; j++);

}


int main()

{

    IODIR0 = 0xFFFFFFFF;// GPIO PIN0 set to output which is connected to the stepper motor

            int count = 1;
```

```
  while(count<=9) // assuming step size is 5deg and total rotation required is 180deg, 180/(5*4) = 9
  {
    IOPIN0 = 0x00000240;
    delay();
    IOPIN0 = 0x00000140;
    delay();
    IOPIN0 = 0x00000180;
    delay();
    IOPIN0 = 0x00000280;
    delay();
                    count =count +1; iterable incremented
  }
  return 0;
      }
```

LOGIC:

Here the stepper motor works the same as said in TASK 2 only difference being that it rotates to the given angle 180 degrees given step being 5 degrees for each pulse. 5 degrees per pulse means that going through the sequence once which has four pulses will produce a rotation of 4*5 degrees = 20 degrees. And so, 180/20 = 9 times the sequence must be repeated to achieve a rotation of 180 degrees.

Learnings from the experiment:

1)Learnt how to interface stepper motor

2)Learnt how to use C programs to implement LPC2378 ARM processor

3)Learnt how to get the input and project the output using i/o registers