Experiment-7

Introduction to the LPC2378 Microcontroller

Batch-35

G Vikas , EE21B051

K Sarath Chandra , EE21B073

PART-1

In this experiment, we will write C programs for various tasks.

The LPC2378 Microcontroller:

The LPC2378 microcontroller to be used in this experiment is from NXP semiconductors. LPC stands for Low Power Consumption. The LPC2378 is based on the ARM7TDMI-S CPU so we will discuss aspects of ARM below.

The ARM and ARM7TDMI-S:

In this experiment, we will use an ARM7-based processor, namely LPC2378. 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). ARM7TDMI was the first of a range of processors introduced in 1995 by ARM. ARM7TDMI is the first core to include the Thumb instruction set. The Thumb instruction set contains 16-bit instructions. These serve as a 'compact shorthand' for a subset of the 32-bit instructions of the ARM. For 32-bit systems, ARM instructions are recommended for higher performance. Thumb instructions are appropriate for memory constrained systems

Task-1:

Complete the program to cause the LEDs on the ARM-board to blink.

Code:

```
#include "LPC23xx.h"
int main ()
{
    int i=1;
    while(1)
    {
        FIO3DIR=0x00FF;
        FIO3PIN=0x00FF;
        for ( i ; i<=10000 ; i++)
        {

        }
        FIO3PIN=0x0000;
        for ( i ; i<=20000 ; i++)
        {

        }
        return 0;
    }
}
```

Explanation:

1. We are using for loop to switch the state of the LEDs.
2. In the first FOR loop the value of FIO3PIN is high, and in the next loop it's low.
3. This turns the LEDs to on and off. This causes the LEDs to blink.
4. In the video in the results we can see the LEDs blinking.

Task-2:

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.

Code:

```
#include "LPC23xx.h"

int main()
{
    int a;
    FIO3DIR = 0xFF;
    FIO4DIR = 0x00;
    while(1)
    {
        a = FIO4PIN;
        FIO3PIN = a;
    }
    return 0;
}
```

Explanation:

1. For the input in the ViARM kit we should get the respected output. To be simple we just gave the output as the input only.
2. The while loop in the code will transfer the data from the input to processor to the LEDs whether to on or off them.
3. First we assign the input to FIR4DIR pin and this pin assigns the value to FIR3DIR pin which is connected to LEDs.
4. This is how the input value is processed by the processor and shown by LEDs.

Task-3:

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

Code:

```c
#include "LPC23xx.h"

int main()
{
    int a;
    int highByte;
    int lowByte;
    FIO3DIR = 0xFF;
    FIO4DIR = 0x00;

    while(1)
    {
        a = FIO4PIN;
        highByte = a & 0xF0;
        highByte = highByte >> 4;
        lowByte = a & 0x0F;
        FIO3PIN = highByte * lowByte;
    }
    return 0;
}
```

Explanation:

1. We are using a while loop in the code to multiply the given numbers bit by bit.
2. The first 4 switches of the ViARM kit act as one 4 bit number. While the next 4 switches of the ViARM kit act as the next 4 bit number.
3. We are assigning the numbers to FIO3DIR and FIO4DIR and multiplying them and showing the results in the LEDs above them.

PART-2

In this experiment, we will interface a stepper motor to the LPC2378 microcontroller. We will begin by describing how a stepper motor works.

How does a stepper motor work ?

A stepper motor is a brushless electric motor that creates discrete rotation from the electrical input pulses. Such a motor is also called a fractional horsepower motor and the absence of a commutator (therefore wear and tear) adds to the robustness. While there are various types of stepper motors, the ones used in the lab experiment are called Permanent Magnet Stepper motors. These have multiple toothed electromagnets (called stator) arranged around a central gear-shaped permanent magnet (termed as rotor) as shown in Figure 1. 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. When the rotor teeth are aligned with stator 1 (as shown in Figure 1), they are slightly offset from stator 2. When the next excitation is given, the rotor aligns with the stator 2. This process continues until the rotor makes one complete revolution. Each of these rotations is known as a “Step” with an integer number of steps making one full revolution.

Waveforms that can drive a stepper motor

The following are the possible drive control methods for a stepper motor.

- Wave Drive control
- Full Step Drive control
- Half Step Drive control

The lab experiment will be performed using the Wave Drive control of Stepper motor.

Task-1:

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.

Code:

```c
#include "LPC23xx.h"

void delay()
{
     int i;
     for(i=0;i<0xFFFF;i++)
     {}
}
int main ()
{
     IODIR0 = 0xFFFFFFFF;

     while(1)
     {
          IOPIN0 = 0x00000280;
          delay();
          IOPIN0 = 0x00000240;
          delay();
          IOPIN0 = 0x00000140;
          delay();
          IOPIN0 = 0x00000180;
          delay();
     }
     return 0;
}
```

Explanation:

1. As we are using permanent magnet stepper motors in the lab. We can rotate the motor by assigning all magnets in the same orientation.
2. We are doing the same in the code above. We just assign all magnets in the motor in the same phase that will rotate the shaft.
3. We used a while loop to do this in the code as you can see.

Task-2:

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.

Code:

```
#include "LPC23xx.h
void delay()
{
    int i;
    for(i=0;i<0x0FFF;i++)
    {

    }
}
int main ()
{
    IODIR0 = 0xFFFFFFFF;
    while(1)
    {
        for(int j=0;j<0x28;j++)
        {
            IOPIN0 = 0x00000280;
            delay();
            IOPIN0 = 0x00000240;
            delay();
            IOPIN0 = 0x00000140;
            delay();
            IOPIN0 = 0x00000180;
            delay();
        }
        for(int j=0;j<0x12;j++)
        {
            delay();
        }
        for(int j=0;j<0x28;j++)
        {
            IOPIN0 = 0x00000180;
            delay();
            IOPIN0 = 0x00000140;
            delay();
            IOPIN0 = 0x00000240;
            delay();
            IOPIN0 = 0x00000280;
            delay();
        }
        for(int j=0;j<0x12;j++)
        {
            delay();
        }
    }
    return 0;
}
```

Explanation:

1. In the previous task we rotated the shaft by orienting all the magnets in the motor in one direction.
2. We can change the direction of the shaft by reversing the orientation of the magnets in the motor.
3. That is what we have done in the code.
4. For some duration we allowed the motor to rotate in a particular direction and then we changed the orientation of the magnets by using the microcontroller.
5. In this process we used FOR loops inside a while loop to change the orientation of the magnets.
6. Note we also included a delay between every directional change.

Task-3:

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.

Code:

```
#include "LPC23xx.h"
void delay4()
{
    int i;
    for(i=0;i<0x02FF;i++)
    {}
}
```

```
void delay3()
{
    int i;
    for(i=0;i<0x05FF;i++)
    {}
}
void delay2()
{
    int i;
    for(i=0;i<0x0AFF;i++)
    {}
}
void delay1()
{
    int i;
    for(i=0;i<0x0FFF;i++)
    {}
}
int main ()
{
    IODIR0 = 0xFFFFFFFF;
    while(1)
    {
        for(int j=0;j<0x35;j++)
        {
            IOPIN0 = 0x00000280;
            delay1();
            IOPIN0 = 0x00000240;
            delay1();
            IOPIN0 = 0x00000140;
            delay1();
            IOPIN0 = 0x00000180;
            delay1();
        }
        for(int j=0;j<0x10;j++)
```

```
{
    delay1();
}
for(int j=0;j<0x35;j++)
{
    IOPIN0 = 0x00000280;
    delay2();
    IOPIN0 = 0x00000240;
    delay2();
    IOPIN0 = 0x00000140;
    delay2();
    IOPIN0 = 0x00000180;
    delay2();
}
for(int j=0;j<0x10;j++)
{
    delay1();
}
for(int j=0;j<0x35;j++)
{
    IOPIN0 = 0x00000280;
    delay3();
    IOPIN0 = 0x00000240;
    delay3();
    IOPIN0 = 0x00000140;
    delay3();
    IOPIN0 = 0x00000180;
    delay3();
}
for(int j=0;j<0x10;j++)
{
    delay1();
}
for(int j=0;j<0x35;j++)
{
```

```
            IOPIN0 = 0x00000280;
            delay4();
            IOPIN0 = 0x00000240;
            delay4();
            IOPIN0 = 0x00000140;
            delay4();
            IOPIN0 = 0x00000180;
            delay4();
        }
        for(int j=0;j<0x10;j++)
        {
            delay1();
        }
    }
    return 0;
}
```

Explanation:

1. The speed of the shaft can be controlled by changing the time for which the magnets in the motor are powered.
2. Thus if the magnets are powered off for a short duration then the speed is high. If they are powered off for a long duration then the speed is low.
3. That is what we have done with the code.
4. We have written it in a way that in every FOR loop there are delays between each magnet. Now the delay decides the speed of the shaft.
5. Initially in the first FOR loop the delay is 0FF which is high, so the speed is less compared to other loops.
6. In this way we increased the speed from 30 rpm to 50 rpm to 70 rpm to 90 rpm, by decreasing the delay in each loop from 0FF to 0AFF to 05FF to 03FF.
7. Note after the shaft reaches the highest speed i.e 90 rpm the loop starts again so the speed again comes to 30rpm

Results:(Part-1&2)

This link contains videos of all six tasks

https://drive.google.com/drive/folders/1-PIUJdtAQURH8MtixStr8LJGb3fpWxRy?usp=sharing