NAME: ROHIT KUMAR

ROLL NO.: EE20B111

# EE2016 Microprocessor Lab & Theory Jul - Nov 2021

EE Department, IIT Madras.

(Lab Experiment-4 Report)

# TITLE: ARM C-Interfacing - Emulation of Switch LED and Stepper Motor Control

## 1.) 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 and

(ii) Stepper motor control using Vi Microsystem's ViARM 7238 development board. Due to ongoing pandemic, we carried out only emulated version of this experiment.

## 2.) Equipments, Hardwares/Softwares Required:

1. ARM ViARM 2378 development board and accessories

2. RS-232 cable

3. Keil microvision 5

4. USB- serial converter (this is a must when the PC loaded with Keil doesn't have a serial port).

5. Flash magic

6. Burn o-mat

7. Stepper motor

## 3.) Questions from the handout (Problem definitions):

1. 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.

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

(a) Identify the signal to the stepper motor and demonstrate it to your TA.

3. Rotate the shaft of stepper motor through some angle (64°) and stop.

## 4.) Solutions to the problems:

*Problem-1: Dis-assemble a byte and displaying the product in LED:*

### Code part:

```
#include "LPC23xx.h" //including the required header file
int main()
{
        FIO3DIR = 0xFF ; //Directing(fast I/O) the Port3(LEDs, P3.0-P3.7) as output
        FIO4DIR = 0x00 ; //Directing(fast I/O) the Port4(DIP Switches, P4.0-P4.7) as input
        int temp; //used to sore input
        int LowNibble; //Last 4 bits of the input(size is 8 bits)
        int HighNibble; //First 4 bits of the input
        while(1) //keep on doing
        {
                temp = FIO4PIN; //temp stores the input given through Port4
                LowNibble = temp & 0x0F; //Extracting the lownibble
                HighNibble = (temp & 0xF0) >> 4;//Extracting the high nibble and properly aligning it
                FIO3PIN = HighNibble*LowNibble; //P3.7 to P3.0 displays the required product
        }
        return 0;
}
```

Logic part:

The input is in 8bits and is stored in **temp**. To get the least significant 4 bits (last 4 bits i.e., low nibble), we simply carry out bitwise AND (&) operation between temp and 0b00001111(equivalent to 0x0F), using the property of "&" (AND): = {1 & X = X, 0 & X = X}. Therefore, the low nibble retains as it is and the high nibble becomes zero, result is stored in "LowNibble".

Similarly, high nibble (first 4 bits) can be obtained with similar logic as said above. Firstly, we carry out bitwise AND operation (&) between temp and 0b11110000(equivalent to 0xF0) , next the required high nibble is obtained by right shifting the above result obtained by four decimal places and then storing the result in "HighNibble".

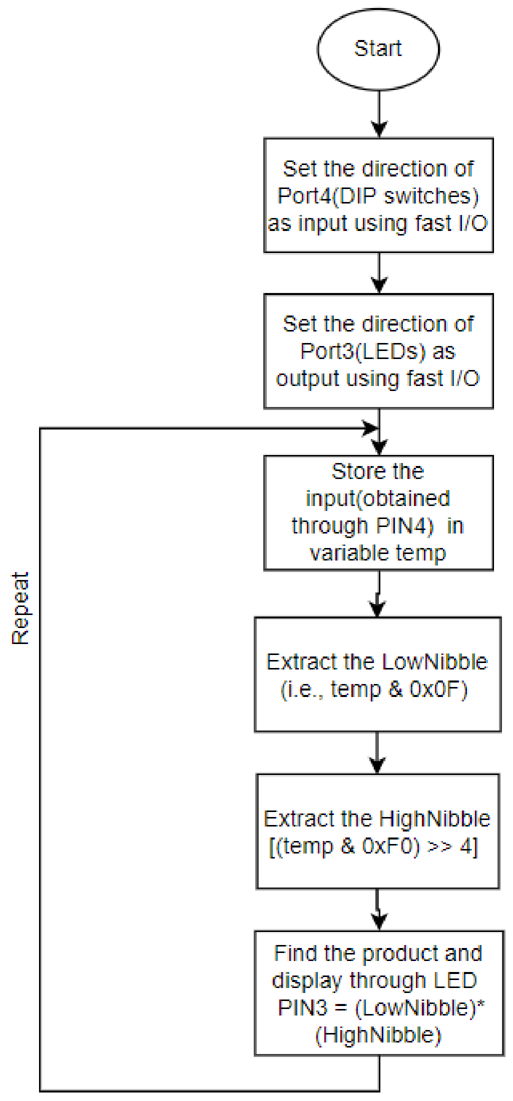
Now, we multiply the nibbles and give at as an output to the LED port, accomplishing the task.

Eg: If input, temp = 0b10101010 (0xAA) then LowNibble = 0b1010 (0xA), HighNibble = 0b1010(0xA)

Output = product (low and high nibbles) = (0x0b00001010) *(0b00001010) = 0b01100100(0x64)

In this case, P3.2, P3.5, P3.6 is only HIGH i.e., corresponding PIN LEDs glow!

Flowchart part:

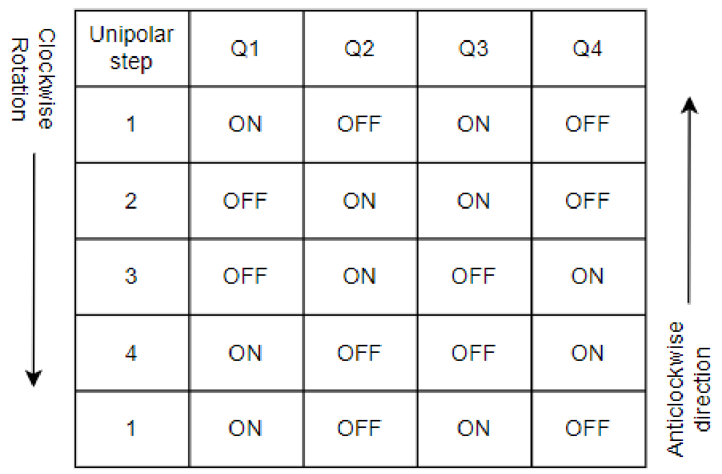

Start
Set the direction of Port4(DIP switches) as input using fast I/O
Set the direction of Port3(LEDs) as output using fast I/O
Store the input(obtained through PIN4) in variable temp
Extract the LowNibble (i.e., temp & 0x0F)
Extract the HighNibble [(temp & 0xF0) >> 4]
Find the product and display through LED PIN3 = (LowNibble)* (HighNibble)
Repeat

*__Problem-2: Modify the demo code to reverse the sense of rotation:__*

Code part:

```
#include "LPC23xx.h"

void delay(void)

{

        int i,j; //generating a delay function with some delay

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

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

}

int main(void)

{

        IODIR0 = 0xFFFFFFFF; //Direction to set Port0 as output

        while(1) //keep on doing

        {

                IOPIN0=0x00000280; //based on the demo code

                delay(); //adding some delay

                IOPIN0=0x00000240; //the values for clockwise were 280,180,140,240

                delay();

                IOPIN0=0x00000140; //so values for anticlockwise would be 280,240,140,180

                delay();

                IOPIN0=0x00000180;

                delay();

        }

        return 0;

}
```

Logic part:


Clockwise Rotation
Anticlockwise direction

| Unipolar step | Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|---|
| 1 | ON | OFF | ON | OFF |
| 2 | OFF | ON | ON | OFF |
| 3 | OFF | ON | OFF | ON |
| 4 | ON | OFF | OFF | ON |
| 1 | ON | OFF | ON | OFF |

Step 1-->0x00000280

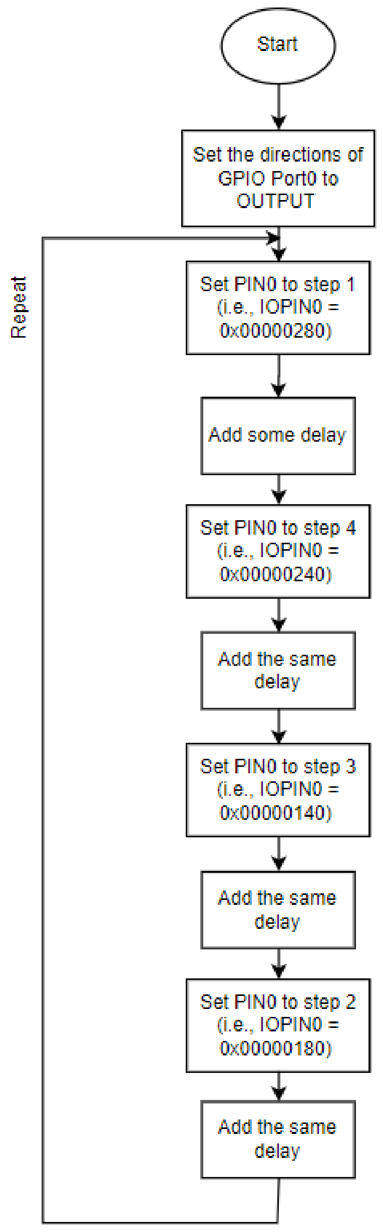
Step 2-->0x00000180

Step 3-->0x00000140

Step 4-->0x00000240

Clearly from the table, the sequence for anti-clockwise direction sense of rotation is step 1,4,3,2. Hence, done like that with some finite and equal delay between each steps.

Flowchart part:


Start
Set the directions of GPIO Port0 to OUTPUT
Repeat
Set PIN0 to step 1 (i.e., IOPIN0 = 0x00000280)
Add some delay
Set PIN0 to step 4 (i.e., IOPIN0 = 0x00000240)
Add the same delay
Set PIN0 to step 3 (i.e., IOPIN0 = 0x00000140)
Add the same delay
Set PIN0 to step 2 (i.e., IOPIN0 = 0x00000180)
Add the same delay

Problem-2(a): Identify the signal to the stepper motor and demonstrate it to your TA.

Solution: The signal can be seen through oscilloscope, which can't be done in emulation-based experiment. However, as demonstrated by TA, the signal given is a unit step (i.e., sudden jump is given at some point).

*Problem-2: Rotate the stepper motor through some angle:* (say, the angle θ = 64°)

Code part:

```
#include "LPC23xx.h"
void delay()
{
        int i, j;  //generating delay function with some delay
        for(i = 0;i < 0xFF; i++)
            for(j = 0;j < 0xFF; j++);
}
 int main()
{
        IODIR0 = 0xFFFFFFFF; //Direction to set Port0 as output
        for(i=0; i<0x08;i++) //This finite loops will make it rotate through finite angle and stop
        {
                IOPIN0 = 0x00000280;
                delay(); //calling delay function
                IOPIN0 = 0x00000240;
                delay();
                IOPIN0 = 0x00000140;
                delay();
                IOPIN0 = 0x00000180;
                delay();  //each for loop makes it rotate for 8°
        }
return 0;
}
```

Logic part:

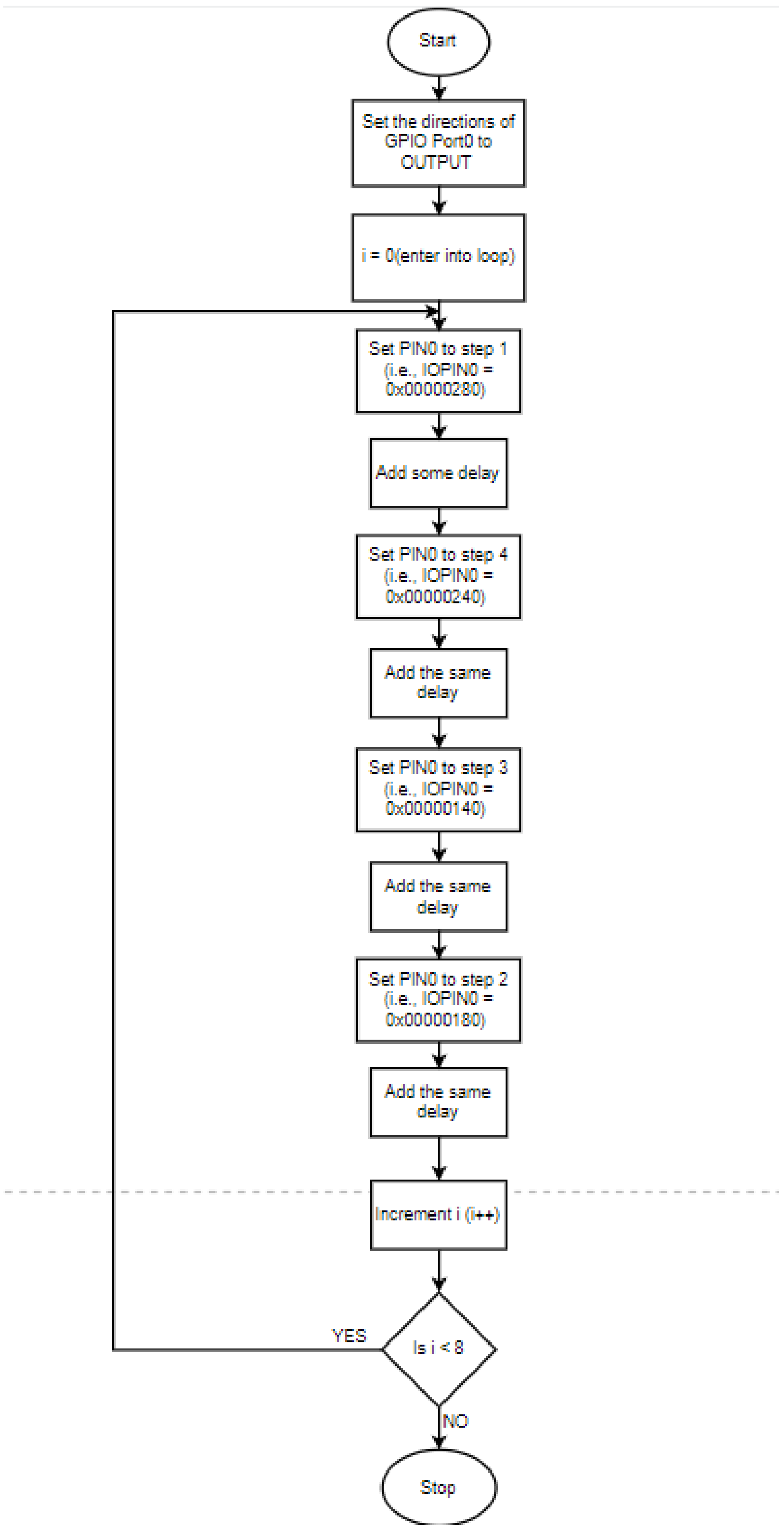
Now, each for each step applied the approximate rotation of stepper motor = 2°

Total number of steps applied in each for loop = 4 (280,240,140,180)

Total number of loops in for loop = 8 (i=0,1,2,3,4,5,6,7)

∴ Total angle rotated = 8 x (4 x 2°) = 64°, as required.

Flowchart part:


Start
Set the directions of GPIO Port0 to OUTPUT
i = 0(enter into loop)
Set PIN0 to step 1 (i.e., IOPIN0 = 0x00000280)
Add some delay
Set PIN0 to step 4 (i.e., IOPIN0 = 0x00000240)
Add the same delay
Set PIN0 to step 3 (i.e., IOPIN0 = 0x00000140)
Add the same delay
Set PIN0 to step 2 (i.e., IOPIN0 = 0x00000180)
Add the same delay
Increment i (i++)
Is i < 8
YES
NO
Stop

## 5.) Inferences/Learning from the experiments:

- Learnt how to implement specific tasks through ARM-C Interfacing.
- Learnt about various ports and accessories in ViARM 2378 development board such as DIP Switches, LEDs. Learnt the two modes of it i.e., programming and execution mode.
- Learnt how to use fast I/O operations, and how to give directions to ports to act as input/output on our choice, i.e., to deal with I/O devices and their connections.
- Learnt how a stepper motor works basically. And how to connect it to our development board. Also, learned what output signal to be given to stepper motor for particular sense of rotation.
- Learnt how to create a C file and its corresponding hex file. And how to burn it into the processor (programming mode) and execute it (execution mode).
- Observed in the demonstration (by TA) that what kind of signa is given to stepper motor through oscilloscope.