EE2016 – EXPERIMENT – 4

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

JAWHAR S

EE20B049

AIM OF THE EXPERIMENT:

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

QUESTIONS (FROM THE HANDOUT):

1. To 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.

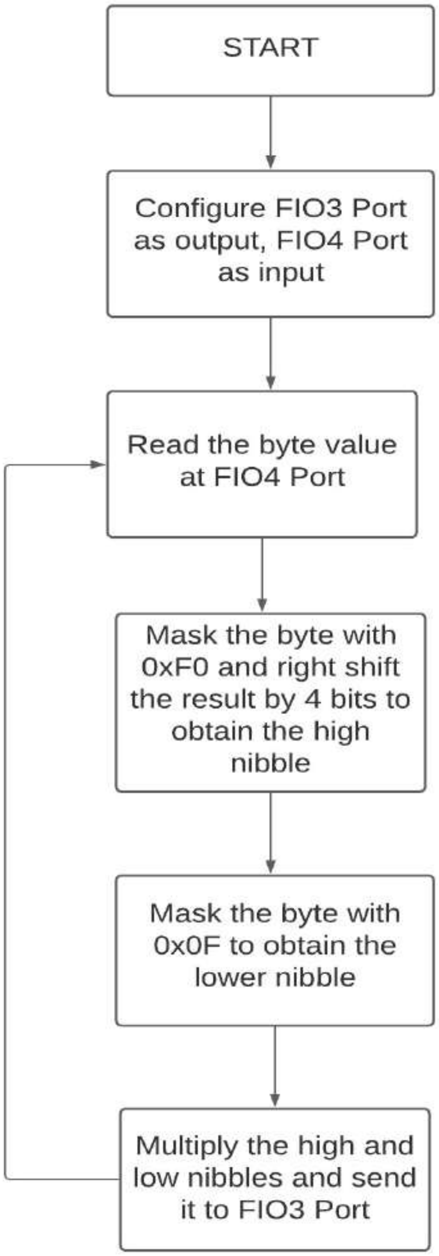
2. To modify the demo code (StpprMtrCntrl.c) supplied to demonstrate the control of stepper motor to rotate in opposite direction.

3. To write a program to rotate the stepper motor by 80 degrees. It is given that step angle of the motor is 2 degrees.

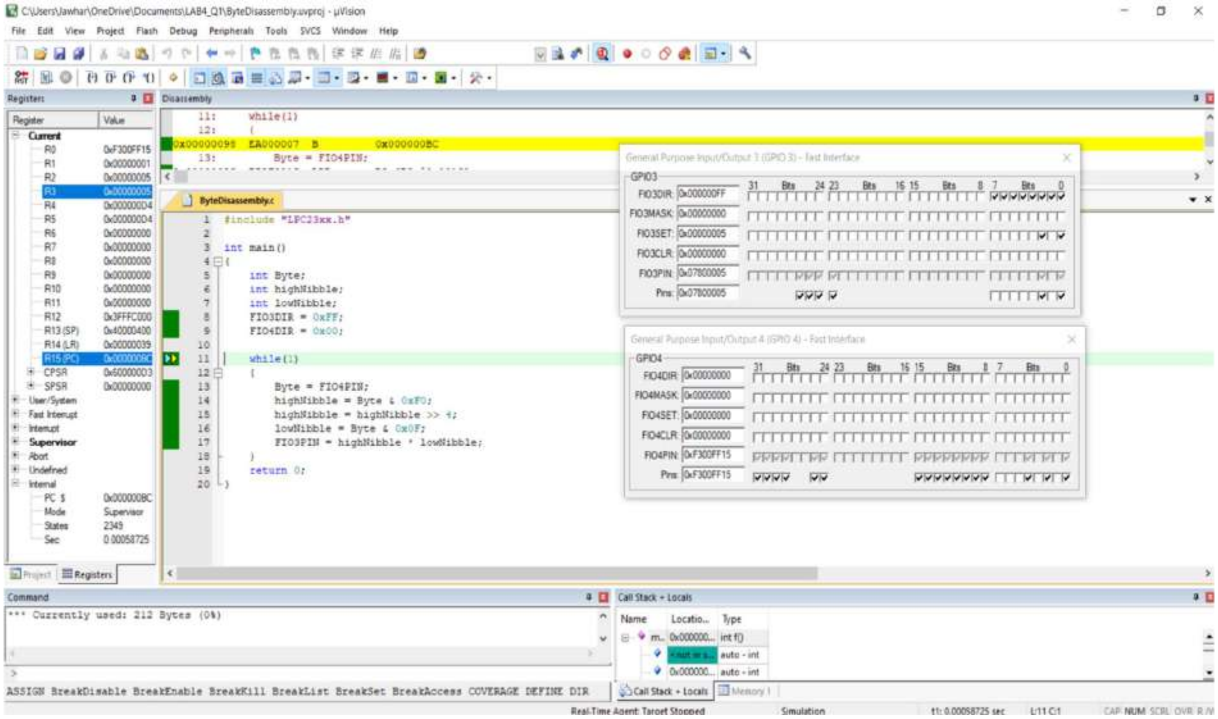
FLOWCHART/LOGIC EXPLANANTION:

1. QUESTION 1:

Flowchart for Question 1


START
Configure FIO3 Port as output, FIO4 Port as input
Read the byte value at FIO4 Port
Mask the byte with 0xF0 and right shift the result by 4 bits to obtain the high nibble
Mask the byte with 0x0F to obtain the lower nibble
Multiply the high and low nibbles and send it to FIO3 Port

Here, we configure the FIO4 port as input to detect the instantaneous state of the DIP switches. There are 8 switches and hence 8 states or 1 byte is used for detecting the states of the switches. The program then disassembles the byte into the higher and lower nibble according to the algorithm given in the flowchart. Then the product of the two nibbles is sent to the FIO3 port as a result of which the LEDs blink according to the objective of the problem statement.


C:\Users\Jawhar\OneDrive\Documents\LAB4_Q1\ByteDisassembly.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
ByteDisassembly.c
Project
Registers
Command
Call Stack + Locals
Memory 1

Register	Value
Current	
R0	0xF300FF15
R1	0x00000001
R2	0x00000005
R3	0x00000005
R4	0x00000004
R5	0x00000004
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x3FFFC000
R13 (SP)	0x40000400
R14 (LR)	0x00000039
R15 (PC)	0x000000BC
CPSR	0x60000003
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x000000BC
Mode	Supervisor
States	2349
Sec	0.00058725

```
    11:     while(1)
    12:     {
0x00000098  EA000007  B          0x000000BC
    13:         Byte = FIO4PIN;
```

```
#include "LPC23xx.h"

int main()
{
    int Byte;
    int highNibble;
    int lowNibble;
    FIO3DIR = 0xFF;
    FIO4DIR = 0x00;

    while(1)
    {
        Byte = FIO4PIN;
        highNibble = Byte & 0xF0;
        highNibble = highNibble >> 4;
        lowNibble = Byte & 0x0F;
        FIO3PIN = highNibble * lowNibble;
    }
    return 0;
}
```

General Purpose Input/Output 3 (GPIO 3) - Fast Interface

GPIO3

	Value
FIO3DIR:	0x000000FF
FIO3MASK:	0x00000000
FIO3SET:	0x00000005
FIO3CLR:	0x00000000
FIO3PIN:	0x07800005
Pins:	0x07800005

General Purpose Input/Output 4 (GPIO 4) - Fast Interface

GPIO4

	Value
FIO4DIR:	0x00000000
FIO4MASK:	0x00000000
FIO4SET:	0x00000000
FIO4CLR:	0x00000000
FIO4PIN:	0xF300FF15
Pins:	0xF300FF15

```
*** Currently used: 212 Bytes (0%)
```

```
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR
```

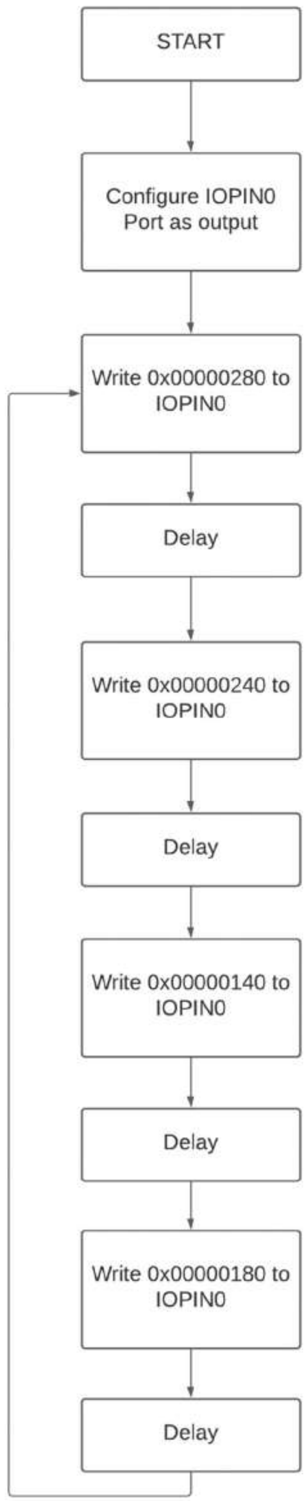
Name	Locatio...	Type
m...	0x000000...	int f()
	<not in s...	auto - int
	0x000000...	auto - int

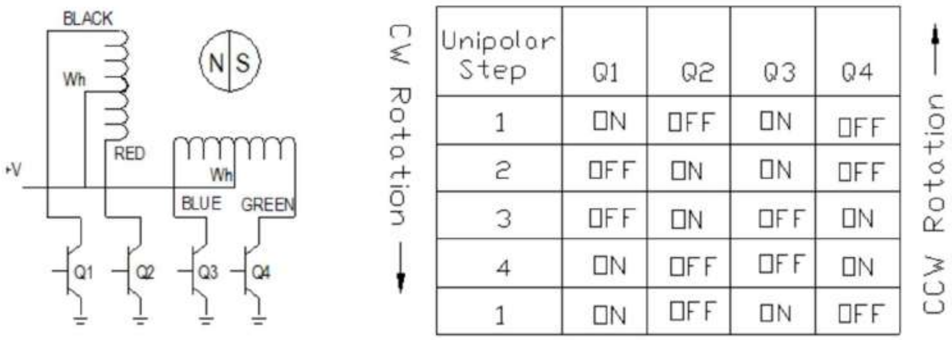
Real-Time Agent: Target Stopped Simulation t1: 0.00058725 sec L:11 C:1 CAP NUM SCRL OVR R/W

As we can see, the byte at FIO4 was manually given as 0001 0101 and after executing the while loop once, 0000 0101 was written into FIO3 (Since 0001 x 0101 = 0000 0101).

2. QUESTION 2

Flowchart for Question 2


START
Configure IOPIN0 Port as output
Write 0x00000280 to IOPIN0
Delay
Write 0x00000240 to IOPIN0
Delay
Write 0x00000140 to IOPIN0
Delay
Write 0x00000180 to IOPIN0
Delay



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

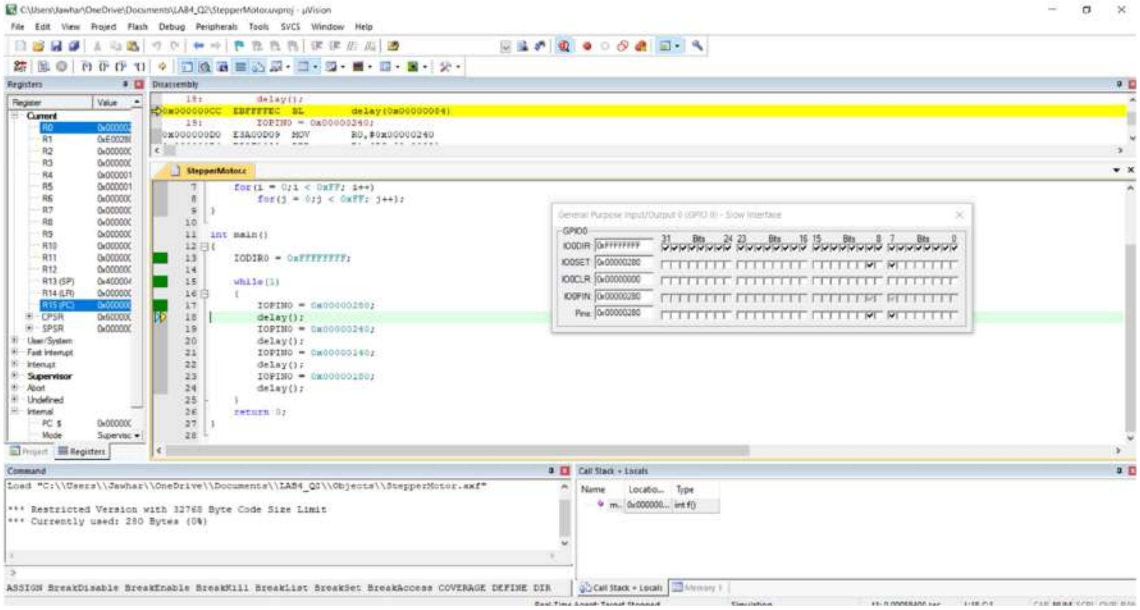
Figure 5: Wiring diagram and step sequence for unipolar motor

To rotate the stepper motor counter clockwise, we need to write corresponding values as shown in the above table to the 6th,7th,8th and 9th bits of the IOPIN0 register.

(Q1 – 9th bit;   Q2 – 8th bit;   Q3 – 7th bit;   Q4 – 6th bit)

Hence, we have to follow the order 1,4,3,2,1,….

When we write 0x00000280 to IOPIN0, the 9th and 7th bits are set as shown below (Unipolar step – 1).


C:\Users\Jawhar\OneDrive\Documents\LAB4_Q2\StepperMotor.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
StepperMotor.c
Command
Call Stack + Locals
Project
Registers
Memory 1

```
       18:         delay();
0x000000CC  EBFFFFEC  BL        delay(0x00000084)
       19:         IOPIN0 = 0x00000240;
0x000000D0  E3A00D09  MOV       R0,#0x00000240
```

```
    for(i = 0;i < 0xFF; i++)
        for(j = 0;j < 0xFF; j++);
}

int main()
{
    IODIR0 = 0xFFFFFFFF;

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

General Purpose Input/Output 0 (GPIO 0) - Slow Interface

GPIO0

		31 Bits 24	23 Bits 16	15 Bits 8	7 Bits 0
IO0DIR	0xFFFFFFFF	☑☑☑☑☑☑☑☑	☑☑☑☑☑☑☑☑	☑☑☑☑☑☑☑☑	☑☑☑☑☑☑☑☑
IO0SET	0x00000280	□□□□□□□□	□□□□□□□□	□□□□□□☑□	☑□□□□□□□
IO0CLR	0x00000000	□□□□□□□□	□□□□□□□□	□□□□□□□□	□□□□□□□□
IO0PIN	0x00000280	□□□□□□□□	□□□□□□□□	□□□□□□☑□	☑□□□□□□□
Pins	0x00000280	□□□□□□□□	□□□□□□□□	□□□□□□☑□	☑□□□□□□□

Register	Value
R0	0x000002
R1	0xE0028
R2	0x00000C
R3	0x00000C
R4	0x000001
R5	0x000001
R6	0x00000C
R7	0x00000C
R8	0x00000C
R9	0x00000C
R10	0x00000C
R11	0x00000C
R12	0x00000C
R13 (SP)	0x40000<
R14 (LR)	0x00000C
R15 (PC)	0x00000C
CPSR	0x60000C
SPSR	0x00000C
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000C
Mode	Supervisc

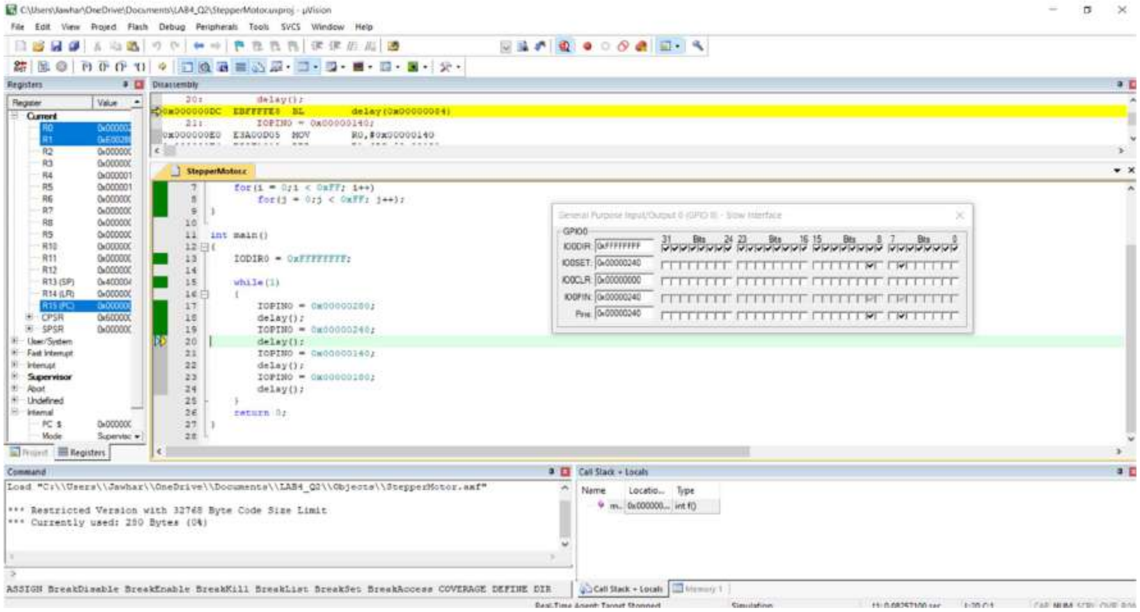
```
Load "C:\\Users\\Jawhar\\OneDrive\\Documents\\LAB4_Q2\\Objects\\StepperMotor.axf"

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 280 Bytes (0%)
```

```
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR
```

Name	Locatio...	Type
m...	0x000000...	int f()

When we write 0x00000240 to IOPIN0, the 9th and 6th bits are set as shown below (Unipolar step - 4).



C:\Users\Jawhar\OneDrive\Documents\LAB4_Q2\StepperMotor.uvproj - µVision

Disassembly

```
    20:             delay();
0x000000DC  EBFFFFE8  BL        delay(0x00000084)
    21:             IOPIN0 = 0x00000140;
0x000000E0  E3A00D05  MOV       R0,#0x00000140
```

StepperMotor.c

```
    for(i = 0;i < 0xFF; i++)
        for(j = 0;j < 0xFF; j++);
}

int main()
{
    IODIR0 = 0xFFFFFFFF;

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

General Purpose Input/Output 0 (GPIO 0) - Slow Interface

GPIO0

		Bits 31–24	Bits 23–16	Bits 15–8	Bits 7–0
IO0DIR:	0xFFFFFFFF	☑☑☑☑☑☑☑☑	☑☑☑☑☑☑☑☑	☑☑☑☑☑☑☑☑	☑☑☑☑☑☑☑☑
IO0SET:	0x00000240	□□□□□□□□	□□□□□□□□	□□□□□□☑□	□☑□□□□□□
IO0CLR:	0x00000000	□□□□□□□□	□□□□□□□□	□□□□□□□□	□□□□□□□□
IO0PIN:	0x00000240	□□□□□□□□	□□□□□□□□	□□□□□□☑□	□☑□□□□□□
Pins:	0x00000240	□□□□□□□□	□□□□□□□□	□□□□□□☑□	□☑□□□□□□

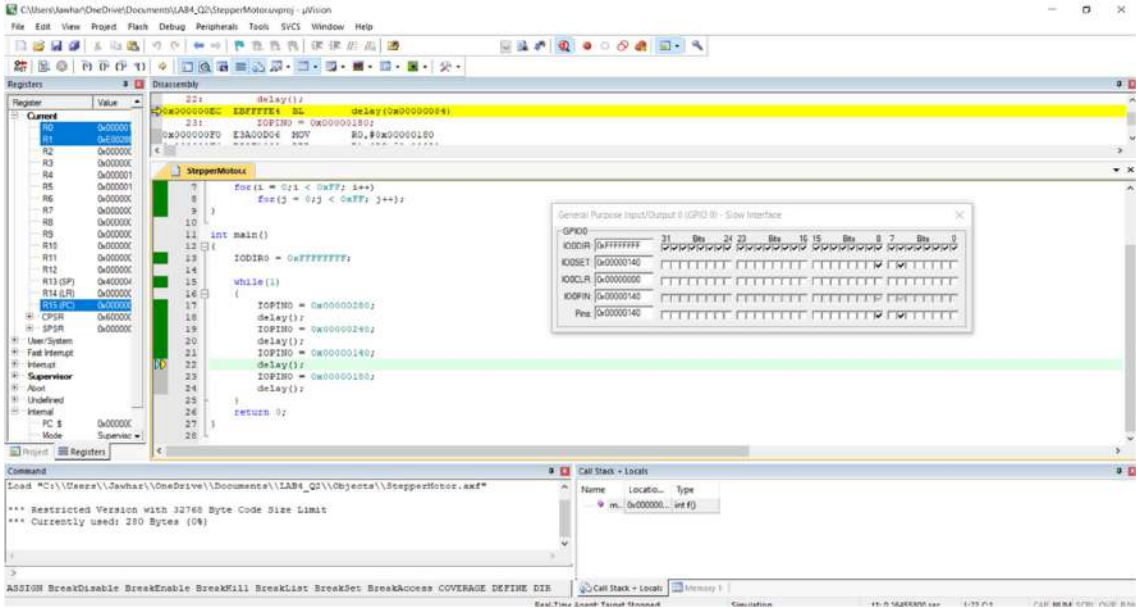
Command

```
Load "C:\\Users\\Jawhar\\OneDrive\\Documents\\LAB4_Q2\\Objects\\StepperMotor.axf"

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 280 Bytes (0%)
```

Name	Locatio...	Type
m...	0x000000...	int f()

When we write 0x00000140 to IOPIN0, the 8th and 6th bits are set as shown below (Unipolar step - 3).



C:\Users\Jawhar\OneDrive\Documents\LAB4_Q2\StepperMotor.uvproj - µVision

Disassembly

```
    22:             delay();
0x000000EC  EBFFFFE4  BL        delay(0x00000084)
    23:             IOPIN0 = 0x00000180;
0x000000F0  E3A00D06  MOV       R0,#0x00000180
```

StepperMotor.c

```
    for(i = 0;i < 0xFF; i++)
        for(j = 0;j < 0xFF; j++);
}

int main()
{
    IODIR0 = 0xFFFFFFFF;

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

General Purpose Input/Output 0 (GPIO 0) - Slow Interface

GPIO0

		Bits 31–24	Bits 23–16	Bits 15–8	Bits 7–0
IO0DIR:	0xFFFFFFFF	☑☑☑☑☑☑☑☑	☑☑☑☑☑☑☑☑	☑☑☑☑☑☑☑☑	☑☑☑☑☑☑☑☑
IO0SET:	0x00000140	□□□□□□□□	□□□□□□□□	□□□□□□□☑	□☑□□□□□□
IO0CLR:	0x00000000	□□□□□□□□	□□□□□□□□	□□□□□□□□	□□□□□□□□
IO0PIN:	0x00000140	□□□□□□□□	□□□□□□□□	□□□□□□□☑	□☑□□□□□□
Pins:	0x00000140	□□□□□□□□	□□□□□□□□	□□□□□□□☑	□☑□□□□□□

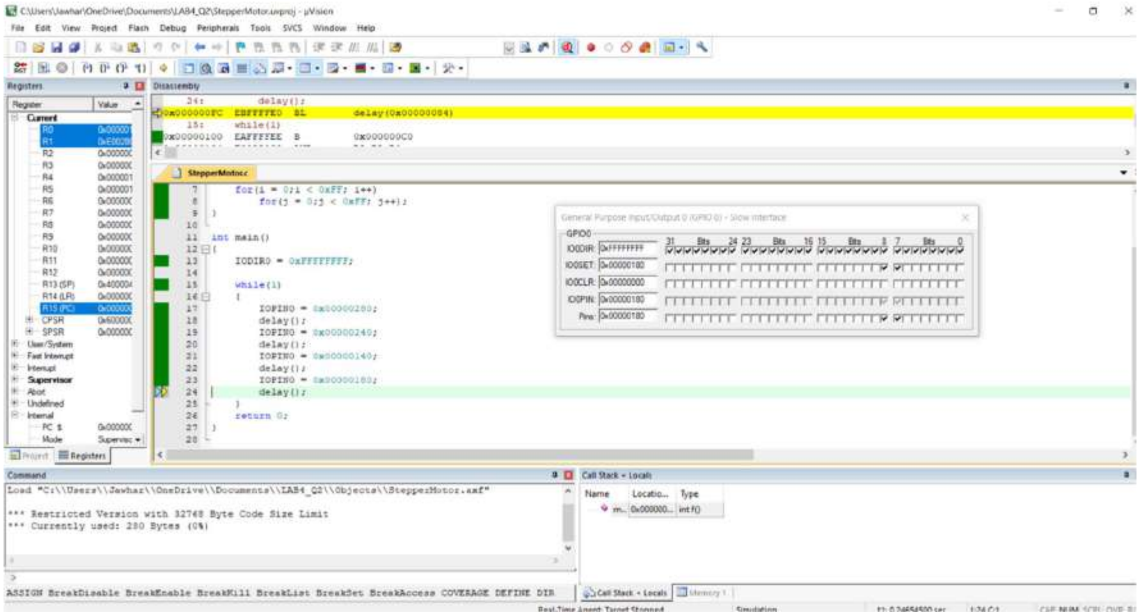
Command

```
Load "C:\\Users\\Jawhar\\OneDrive\\Documents\\LAB4_Q2\\Objects\\StepperMotor.axf"

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 280 Bytes (0%)
```

Name	Locatio...	Type
m...	0x000000...	int f()

When we write 0x00000180 to IOPIN0, the 8th and 7th bits are set as shown below (Unipolar step - 2).


C:\Users\Jawhar\OneDrive\Documents\LAB4_Q2\StepperMotor.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
StepperMotor.c
General Purpose Input/Output 0 (GPIO 0) - Slow Interface
GPIO0
Command
Call Stack + Locals
Name Locatio... Type
Project Registers
Call Stack + Locals Memory 1
Real-Time Agent: Target Stopped
Simulation

```
      24:         delay();
0x000000FC  EBFFFFE0  BL        delay(0x00000084)
      15:     while(1)
0x00000100  EAFFFFEE  B         0x000000C0
```

```
    for(i = 0;i < 0xFF; i++)
        for(j = 0;j < 0xFF; j++);
}

int main()
{
    IODIR0 = 0xFFFFFFFF;

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

IO0DIR: 0xFFFFFFFF
IO0SET: 0x00000180
IO0CLR: 0x00000000
IO0PIN: 0x00000180
Pins: 0x00000180

```
Load "C:\\Users\\Jawhar\\OneDrive\\Documents\\LAB4_Q2\\Objects\\StepperMotor.axf"

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 280 Bytes (0%)
```

```
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR
```

3. QUESTION 3:

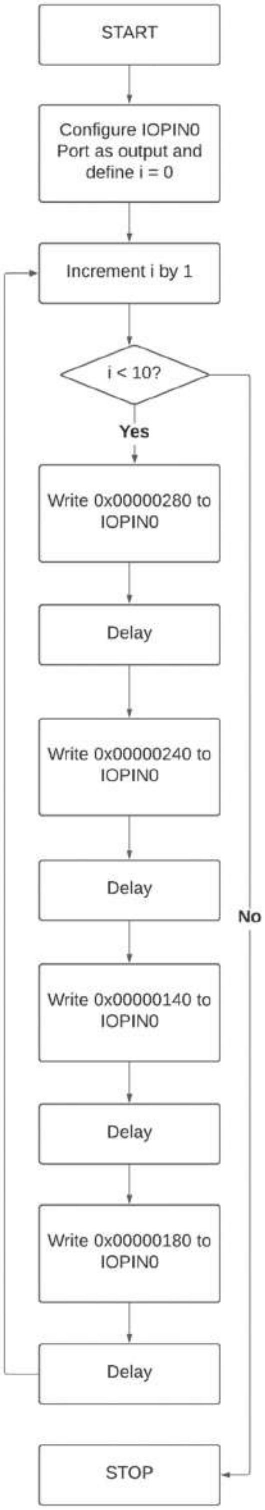
Given the step angle of the motor to be 2 degrees. The objective is to rotate the stepper motor by 80 degrees from its initial position. The logic for this problem is same as that of previous problem except the fact that we have to stop rotating the motor by 80 degrees. So instead of using a while loop in the code, we can use a for loop iterating 10 times and constrain the rotation by 80 degrees. The no. of times that we should iterate can be calculated as follows.

Step angle = 2 degrees

No. of times the stepper motor is rotated within 1 loop = 4

Therefore, to obtain 80 degrees rotation, the no. of times we should iterate the loop = 80/2*4 = 10 times.

Flowchart for Question 3


START
Configure IOPIN0 Port as output and define i = 0
Increment i by 1
i < 10?
Yes
No
Write 0x00000280 to IOPIN0
Delay
Write 0x00000240 to IOPIN0
Delay
Write 0x00000140 to IOPIN0
Delay
Write 0x00000180 to IOPIN0
Delay
STOP

CODES:

1.

```
#include "LPC23xx.h"

int main()
{
   int Byte;
   int highNibble;
   int lowNibble;
   FIO3DIR = 0xFF;
   FIO4DIR = 0x00;

   while(1)
   {
      Byte = FIO4PIN;
      highNibble = Byte & 0xF0;
      highNibble = highNibble >> 4;
      lowNibble = Byte & 0x0F;
      FIO3PIN = highNibble * lowNibble;
   }
   return 0;
}
```

2.

```
#include "LPC23xx.h"
```

```
void delay()
{
    int i, j;

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

int main()
{
    IODIR0 = 0xFFFFFFFF;

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

3.

```
#include "LPC23xx.h"
void delay()
{
    int i, j;

    for(i = 0;i < 0xFF; i++)
        for(j = 0;j < 0xFF; j++);
}
int main()
{
    IODIR0 = 0xFFFFFFFF;

    for(int i = 0;i < 0x0A; i++)
    {
        IOPIN0 = 0x00000280;
        delay();
        IOPIN0 = 0x00000240;
        delay();
        IOPIN0 = 0x00000140;
        delay();
        IOPIN0 = 0x00000180;
        delay();
    }
    return 0;
}
```

INFERENCES/LEARNINGS FROM THE EXPERIMENT:

- Learnt about the ViARM-2378 Development Board, various registers used as I/O ports.
- Learnt how to configure the Kiel software for C-interfacing.
- Learnt how to interface the DIP switches and LEDs in LPC 2378 ARM processor.
- Learnt about the working principle of a stepper motor and the stepper motor driver in the ViARM-2378 Development Board.
- Learnt how to program to rotate the stepper motor in both clockwise and counter clockwise directions.
- Learnt about the step angle of the motor and how to program to rotate the stepper motor by a certain angle.