## 2.3 INTERFACING STEPPER MOTOR WITH LPC1768 MICROCONTROLLER

**Learning Outcomes:**

After studying this interfacing project, learners will be able to:

Understand Stepper Motor Basics

Interface Stepper Motor with LPC1768 Microcontroller

Understand usage of ULN2803 as a Current Driver

Implement Stepper Motor Control Logic

Write Embedded C Code for Stepper Motor Control

### 2.3.1 INTERFACING DIAGRAM

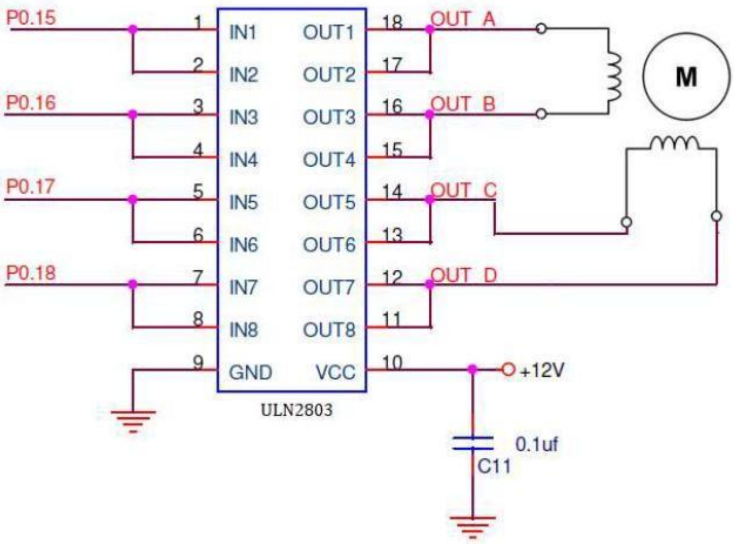


**Fig.: Interfacing Stepper Motor with LPC1768 Microcontroller**

The circuit diagram shows how a stepper motor is interfaced with an **LPC1768** microcontroller using a **ULN2803** Darlington transistor array.

- The LPC1768 microcontroller generates pulse sequences on GPIO pins (P0.15, P0.16, P0.17, P0.18). These signals are sent to the IN1 to IN4 pins of the ULN2803.
- ULN2803 – A Darlington transistor array that acts as a current amplifier. When an input (INx) from the microcontroller is HIGH, the corresponding output (OUTx) is pulled to GND. This completes the circuit for the stepper motor coil, allowing current to flow through it.
- The stepper motor requires a specific sequence of pulses on its coils to rotate in steps.
- The motor windings are energized in a controlled pattern (e.g., Full Step or Half Step m. The stepper motor operates at +12V, which is supplied through the VCC (Pin 10) of ULN2803. The capacitor (C11, 0.1µF) is placed for power stability and noise reduction.

### 2.3.2 DESIGN CALCULATIONS

The total number of steps required to rotate stepper motor is given by:

$$\text{No. of Steps} = \frac{X_0}{1.8_0}$$

Where, Step Angle = $1.8^0$ is the angle the motor moves per step specified in the datasheet.

Examples:

| Degrees | Number of steps |
|---|---|
| One Revolution ($360^0$) | $\text{No.of Steps} = \frac{360}{1.8} = 200$ |
| $90^0$ | $\text{No.of Steps} = \frac{90}{1.8} = 50$ |

### 2.3.3 EXCITATION SEQUENCE FOR STEPPER MOTOR CONTROL

**Wave Drive Mode**, also known as **Single-Coil Excitation**, is a stepper motor control method where only one coil is energized at a time. This mode is simpler and consumes less power but provides **lower torque** compared to **full-step** and **half-step** modes.

<table>
<tr><th>P0.18</th><th>P0.17</th><th>P0.16</th><th>P0.15</th><th>Hex Value</th><th>Clock Sequence</th><th>Anti-Clock Sequence</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0x0000 8000</td><td rowspan="4">↓</td><td rowspan="4">↑</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0x0001 0000</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0x0002 0000</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0x0004 0000</td></tr>
</table>

### 2.3.4 EXAMPLE CODES

**3a) Write a C program to continuously rotate a stepper motor clockwise and counterclockwise, completing a full 360-degree rotation in each direction.**

```
#include <lpc17xx.h>
#define DEGREES 360 void
delay(unsigned int ms)
{
    unsigned int i,j;
for(i=0;i<ms;i++)
for(j=0;j<1275;j++);
}
int main() {
    unsigned int steps = 200*DEGREES/360;
unsigned int i = 0;
```

```
    unsigned int clock[] =
            {0x00008000,0x00010000,0x00020000,0x00040000};
unsigned int anticlock[] =
             {0x00040000,0x00020000,0x00010000,0x00008000};

    LPC_GPIO0->FIODIR|=0x00078000;

    while(1)
    {
        for(i = 0; i < steps; i++)
        {
            LPC_GPIO0->FIOCLR |= 0x00078000;
            LPC_GPIO0->FIOSET = clock[i%4];
delay(200);
        }

        for(i = 0; i < steps; i++)
        {
            LPC_GPIO0->FIOCLR |= 0x00078000;
LPC_GPIO0->FIOSET = anticlock[i%4];
            delay(200);
        }
    }
}
```

**Further exploration:**

Explore full step sequence and half-step sequence.
Try options to reduce the code size.
Try different scenarios ○ Rotating for specific number of times for given angle. ○ Rotate once / fixed number of times / continuously.