CSE 331/EEE 332 (Microprocessor Interfacing & Embedded System Lab)

Lab 08 : Stepper motor Interfacing with 8051 Microcontroller using Proteus
Rotate Stepper Motor – Both Half and Full Step (clockwise and anti-clockwise)
Instructions: DJNZ, SJMP, LJMP, ACALL, LCALL

Lab Instructor : Rokeya Siddiqua

Topics to be covered in class today:

- Microcontroller - 8051 interfacing with Stepper Motor
- Rotate Stepper Motor - Both Half and Full Step (clockwise and anti-clockwise directions)
- Instructions: DJNZ, SJMP, LJMP, ACALL, LCALL

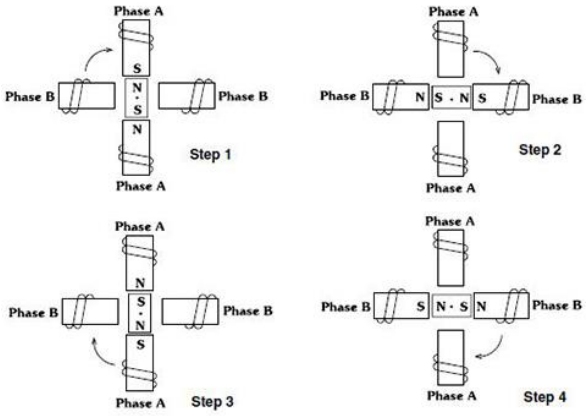
Stepper Motor

A stepper motor is a device that translates electrical pulses into mechanical movement in steps of fixed step angle.

- The stepper motor rotates in steps in response to the applied signals.
- It is mainly used for position control.
- It is used in disk drives, dot matrix printers, plotters and robotics and process control circuits.

Structure

Stepper motors have a permanent magnet called rotor (also called the shaft) surrounded by a stator. The most common stepper motors have four stator windings that are paired with a center-tap. This type of stepper motor is commonly referred to as a four-phase or unipolar stepper motor. The center tap allows a change of current direction in each of two coils when a winding is grounded, thereby resulting in a polarity change of the stator.



Interfacing

Even a small stepper motor requires a current of 400 mA for its operation. But the ports of the microcontroller cannot source this much amount of current. If such a motor is directly connected to the microprocessor/microcontroller ports, the motor may draw large current from the ports and damage it. So, a suitable driver circuit is used with the microprocessor/microcontroller to operate the motor.

Motor Driver

Circuit (ULN2003) Stepper motor driver circuits are available readily in the form of ICs. ULN2003 is one such driver IC which is a High-Voltage High-Current Darlington transistor array and can give a current of 500mA. This current is sufficient to drive a small stepper motor. Internally, it has protection diodes used to protect the motor from damage due to back emf and large eddy currents. So, this ULN2003 is used as a driver to interface the stepper motor to the microcontroller.

Operation

The important parameter of a stepper motor is the step angle. Stepper Motor can be operated in two types of sequences:

Steps	Angle (degree)	Number of steps
Half Step	45	8
Full Step	90	4

Half step Sequence

Step	A	B	A'	B'	Hex Value
0	1	0	0	0	08H
1	1	1	0	0	0CH
2	0	1	0	0	04H
3	0	1	1	0	06H
4	0	0	1	0	02H
5	0	0	1	1	03H
6	0	0	0	1	01H
7	1	0	0	1	09H

Full step Sequence

Step	A	B	A'	B'	Hex codes
0	1	1	0	0	0CH
1	0	1	1	0	06H
2	0	0	1	1	03H
3	1	0	0	1	09H

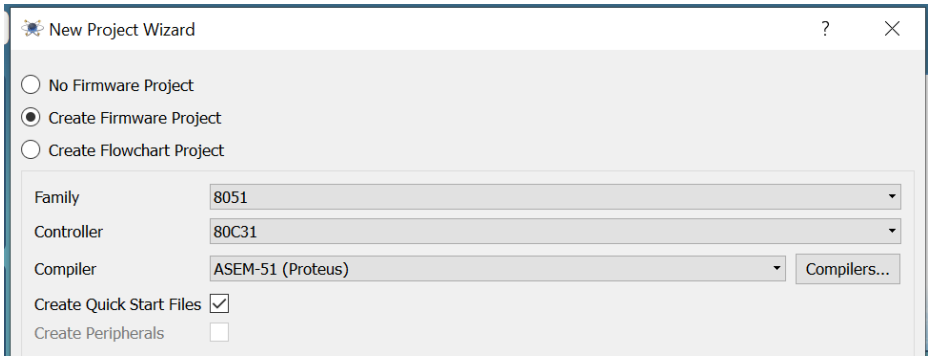
Note: The hex code mainly depends on the construction of the stepper motor.

Components:

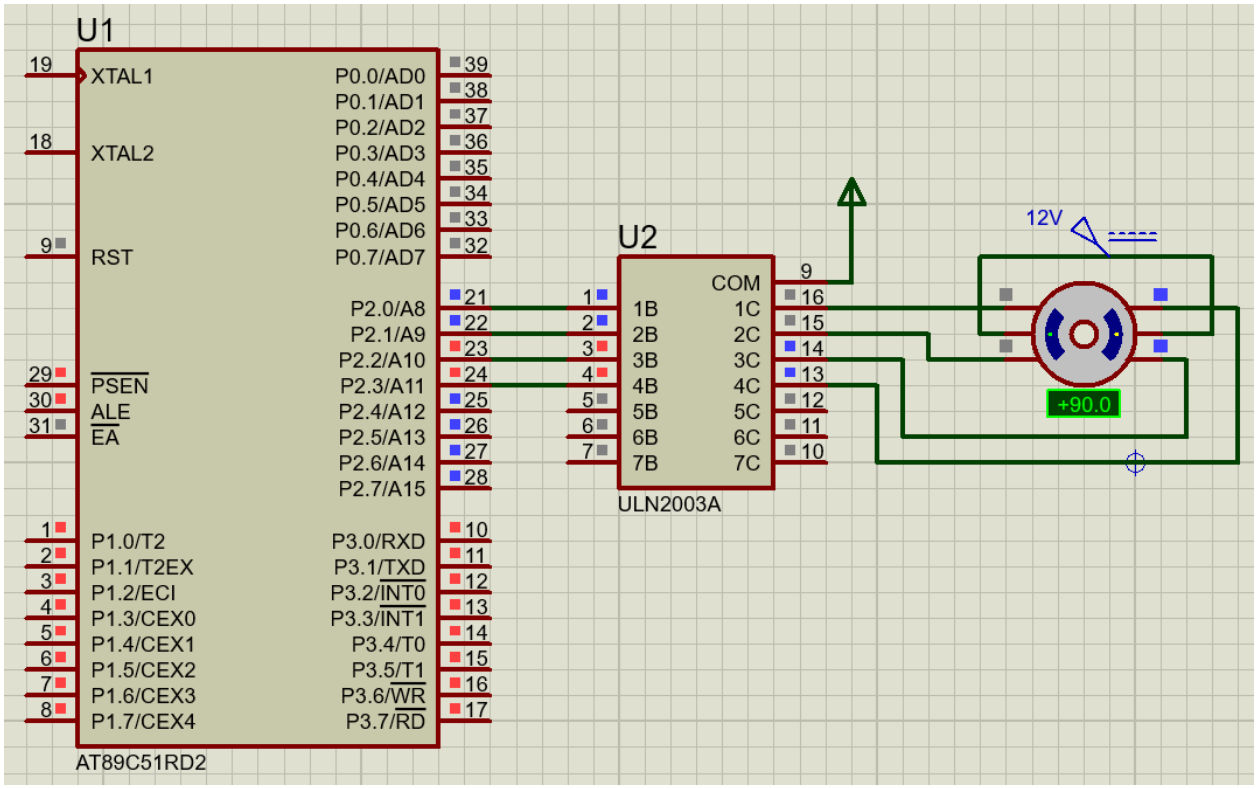
Component Details	Specification
Microcontroller 8051	AT89C51RD2
Stepper Motor	MOTOR-STEPPER (Unipolar)
Motor Driver	ULN2003A

Step 1: Create New Project

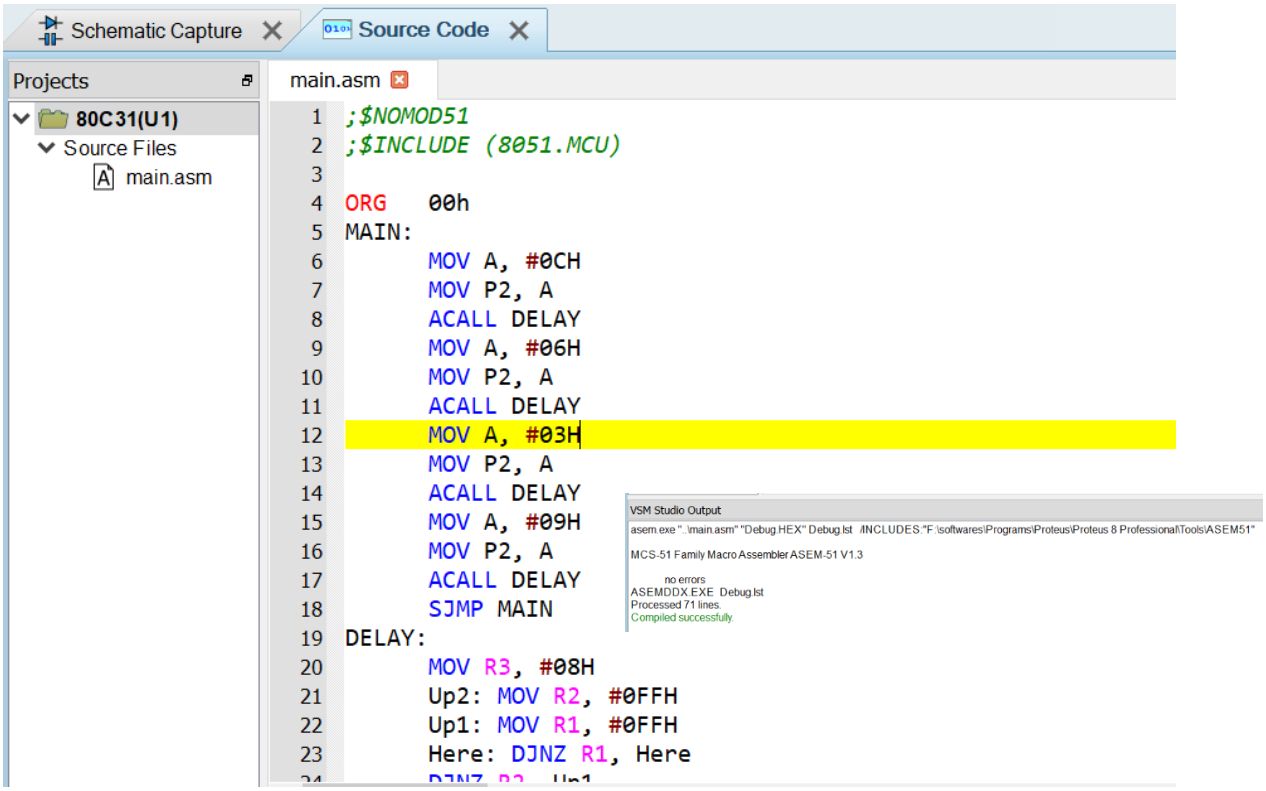
Open Proteus software. *File* >> *New Project* >> *Project Name* and *Path*



Step 2: Simulation



Step 3: Code



ASSEMBLY LANGUAGE PROGRAM (8051):

;$NOMOD51
;$INCLUDE (8051.MCU)

ORG 00h ; sets the program origin to memory address 00h

MAIN:

MOV A, #0CH ; Load the value 0CH (binary 1100) into the accumulator (A)

MOV P2, A ; Move the value from the accumulator (A) to Port 2 (P2)

ACALL DELAY ; Call a subroutine named DELAY

MOV A, #06H

MOV P2, A

```
    ACALL DELAY
    MOV A, #03H
    MOV P2, A
    ACALL DELAY
    MOV A, #09H
    MOV P2, A
    ACALL DELAY
    SJMP MAIN
DELAY:
    MOV R3, #08H ; outer loop counter
    Up2:       ; Label for the outer loop
        MOV R2, #0FFH  ; inner loop counter
        Up1:         ; Label for the inner loop
            MOV R1, #0FFH
            Here:       ; Label for the innermost loop
                DJNZ R1, Here ; Decrement and Jump
        DJNZ R2, Up1
    DJNZ R3, Up2
    RET


END
```

Instructions:

Instructions	Details
DJNZ (Decrement and Jump if Not Zero)	The DJNZ instruction is typically used within loops to decrement a specified register and jump to a specified label if the result of the decrement operation is not zero. DJNZ register, label *Example:* MOV R1, #10 ; Initialize R1 with the value 10 Loop: DJNZ R1, Loop ; Decrement R1 and jump back to "Loop" if R1 is not zero
SJMP (Short Jump) => (00 H -- FF H)	SJMP is used for short-range jumps within the current code segment or program block. SJMP typically allows you to jump to a location within a limited range, often within -128 to +127 bytes of the current instruction. If you're working in a small section of code and need to branch to a nearby instruction, you would use SJMP.
LJMP (Long Jump) => (0000 H -- FFFF H)	LJMP is used for long-range jumps within the entire code space or program memory. LJMP allows you to jump to any address within the entire program memory, which is useful for branching to distant parts of the code. If you need to jump to a different code segment located anywhere in memory, you would use LJMP.
LCALL (Long Call) => (0000 H -- FFFF H)	LCALL is used for long-range subroutine calls within the entire code space. LCALL allows you to call a subroutine located anywhere in program memory, and it typically pushes the return address onto the stack. When you need to call a subroutine located elsewhere in memory and then return to the calling instruction, you would use LCALL.
ACALL (Absolute Call) => (00 H -- FF H)	ACALL is used for absolute subroutine calls within a limited address range. ACALL typically allows you to call subroutines within a specific range, often from 00H to FFH. When you need to call a subroutine located within a restricted address range, you would use ACALL.

Simulation (YouTube) Link: https://www.youtube.com/watch?v=mP-NHtD0PNs&t=578s&ab_channel=WikiNote

Task 1: Rotate the stepper motor anti-clockwise (Full Step).

Task 2: Rotate the stepper motor clockwise (Half Step).

Task 3: Rotate the stepper motor anti-clockwise (Half Step).