AIM: To implement stepper motor functionality using Atmega 16 microcontroller

Tools: Keil uVision 4, ISIS Proteus 7

Theory: Physically, a stepper motor (or step motor) is a brushless DC electric motor that divides a full rotation into a number of equal steps. The motor's position can then be commanded to move and hold at one of these steps without any feedback sensor (an open-loop controller), as long as the motor is carefully sized to the application. Practically, the functionality of a stepper motor can be simulated by changing the input pattern to the stepper motor at fixed instants of time, thereby simulating the 'stepping'. This is achieved in our project by pressing the forward and reverse button provided. Thus the motion of the motor is controlled by these buttons. The angle of rotation of the motor is also displayed with the help of a display provided.

The ATmega16 is a low-power CMOS 8-bit micro-controller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega16 achieves throughputs approaching 1MIPS per MHz allowing the system designer to optimize power consumption versus processing speed. The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

Algorithm:
1) Start
2) Define the pin configuration bits and the pin definitions.
3) Define the hex values necessary to produce required rotation.
4) Initialize the Stack Pointer and Configure the ports.
5) If rotation is desired in the forward direction, give positive values to the ports.
6) If rotation is desired in the reverse direction, give negative values to the ports.
7) Keep repeating 5 and 6.
8) Stop

Code:

```
#include <avr/io.h>
#include <util/delay.h>
```

```
#include <avr/pgmspace.h>
#define uchar unsigned char
#define uint unsigned int

// Low level port/pin definitions
#define sbit(x,PORT) (PORT) |= (1<<x)
#define cbit(x,PORT) (PORT) &= ~(1<<x)
#define pin(x,PIN) (PIN)&(1<<x)

// Pins definition
#define pos pin(0,PINC)
#define neg pin(1,PINC)
#define out PORTD

uchar PROGMEM turn[] =
{0x02,0x06,0x04,0x0c,0x08,0x09,0x01,0x03};

int main(void)
 { uchar i=0;

   // Initialize Stack Pointer
   SPL = 0x5f;
   SPH = 0X04;

   // Configure Ports
   DDRD = 0xff;
   DDRC = 0x00;
   out  = 0xff;

//Motor action loop
   while(1)
    {
// Clockwise rotation
      if(!(pos))
       { i = i<8? i+1: 0;
         out = pgm_read_byte(&turn[i]);
         _delay_ms(50);
       }
// Anticlockwise rotation
     else if(!(neg))
            { i = i>0? i-1: 7;
              out = pgm_read_byte(&turn[i]);
              _delay_ms(50);
            }
```

```
        }
    }
```

Conclusion: The AVR family of controllers was studied in detail and the stepper motor project was implemented using the ATmega16 microcontroller. The inputs were manipulated and the effect was observed on the output window.