

ICT 5307 : Embedded System Design

Lecture 12 Interfacing Motors

Professor S.M. Lutful Kabir

BUET

DC Motor

- DC (Direct Current) Motors are two wire (power & ground), continuous rotation motors.
- When you supply power, a DC motor will start spinning until that power is removed.
- Most DC motors run at a high RPM (revolutions per minute), examples being computer cooling fans, or radio controlled car wheels!
- The speed of DC motors is controlled using pulse width modulation (PWM).
- Each pulse is so rapid that the motor appears to be continuously spinning.

The Servo Motor

- Servo motors are generally an assembly of four things: a DC motor, a gearing set, a control circuit and a position-sensor.
- The position of servo motors can be controlled more precisely than those of standard DC motors, and they usually have three wires (power, ground & control).
- Power to servo motors is constantly applied, with the servo control circuit regulating the draw to drive the motor.
- Servo motors do not rotate freely like a standard DC motor.
- Instead the angle of rotation is limited to 180 Degrees (or so) back and forth.
- Servo motors receive a control signal that represents an output position and applies power to the DC motor until the shaft turns to the correct position, determined by the position sensor.

Servo Motor (continued.....)

- PWM is used for the control signal of servo motors.
- However, unlike DC motors it's the duration of the positive pulse that determines the position, rather than speed, of the servo shaft.
- A neutral pulse value dependent on the servo (usually around 1ms) keeps the servo shaft in the center position.
- Increasing that pulse value will make the servo turn clockwise, and a shorter pulse will turn the shaft anticlockwise.
- The servo control pulse is usually repeated every 20 milliseconds.
- When a servo is commanded to move, it will move to the position and hold that position, even if external force pushes against it.
- The servo will resist from moving out of that position, with the maximum amount of resistive force the servo can exert.

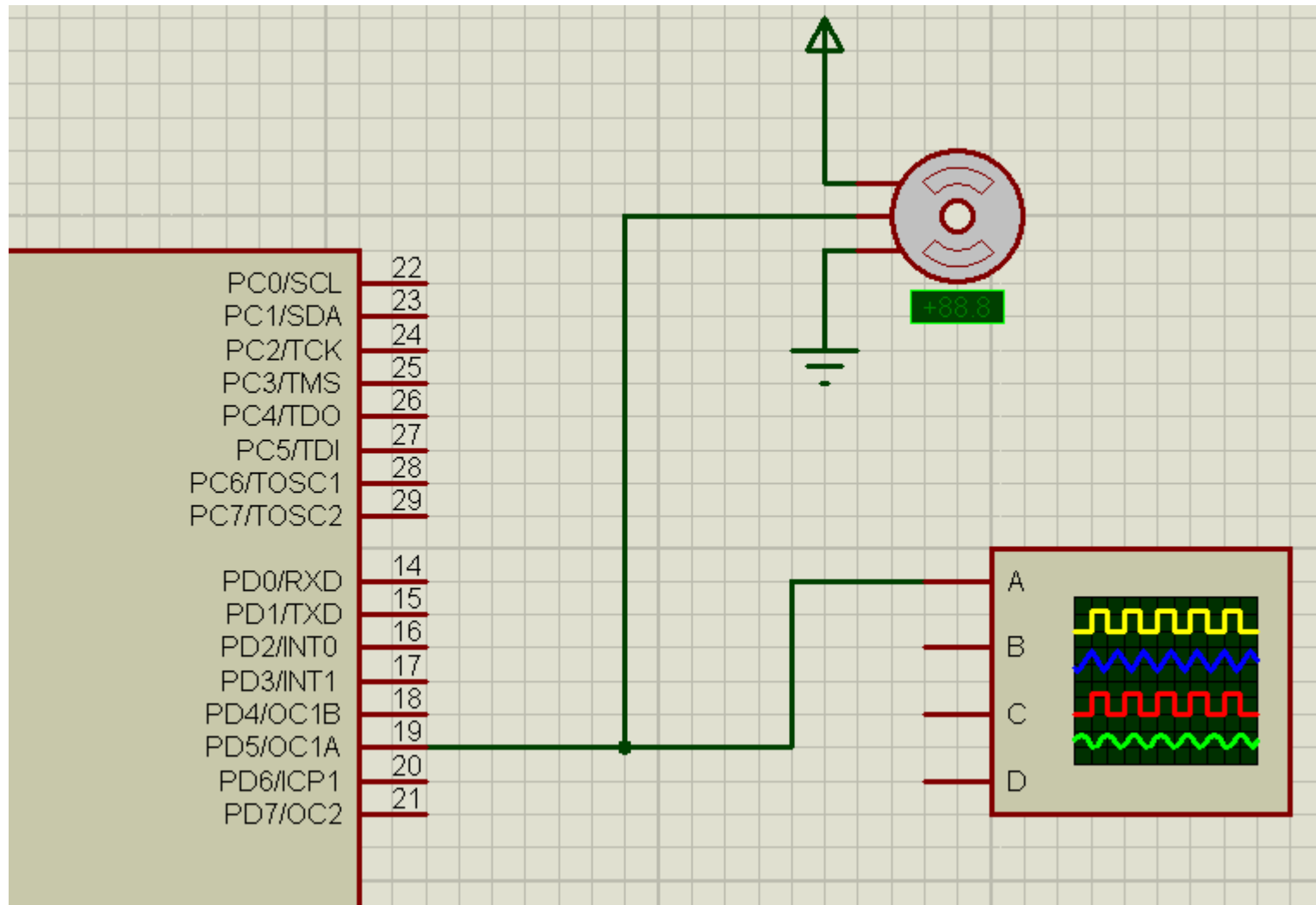
Stepper Motor

- Stepper motors utilize multiple toothed electromagnets arranged around a central gear to define position.
- Stepper motors require an external control circuit or micro controller to individually energize each electromagnet
- When electromagnet 'A' is powered it attracts the gear's teeth and aligns them, slightly offset from the next electromagnet 'B'.
- When 'A' is switch off, and 'B' switched on, the gear rotates slightly to align with 'B', and so on around the circle,
- Each rotation from one electromagnet to the next is called a "step", and thus the motor can be turned by precise pre-defined step angles through 180 or full 360 Degree rotation.

Stepper Motor (continued....)

- Stepper motors are available in two varieties; unipolar or bipolar.
- Bipolar motors are the strongest type of stepper motor and usually have four or eight leads.
- They have two sets of electromagnetic coils internally, and stepping is achieved by changing the direction of current within those coils.
- Unipolar motors, identifiable by having 5,6 or even 8 wires, also have two coils, but each one has a center tap.
- Unipolar motors can step without having to reverse the direction of current in the coils, making the electronics simpler.

Interfacing a Servo Motor in Proteus



Properties of the Servo Motor

The image shows a software dialog box titled "Edit Component" with a standard Windows-style title bar (minimize, maximize, close buttons). The dialog is used to configure the properties of a servo motor component. It contains several input fields for numerical values, checkboxes for visibility and simulation settings, and a list for other properties. The "Component Reference" and "Component Value" fields are empty. The "Minimum Angle" is set to 0.0, "Maximum Angle" to +180, "Rotational Speed" to 50, "Minimum Control Pulse" to 1m, and "Maximum Control Pulse" to 2m. Each of these five fields has a "Hidden" checkbox (all are unchecked) and a "Hide All" button with a dropdown arrow. Below these is a section for "Other Properties" with a large empty text area. At the bottom, there are four checkboxes: "Exclude from Simulation" (unchecked), "Exclude from PCB Layout" (checked), "Edit all properties as text" (unchecked), "Attach hierarchy module" (unchecked), and "Hide common pins" (unchecked). On the right side of the dialog, there are three buttons: "OK", "Help", and "Cancel".

Edit Component

Component Reference:

Component Value:

Hidden: ☐

Hidden: ☐

Minimum Angle: Hide All ▾

Maximum Angle: Hide All ▾

Rotational Speed: Hide All ▾

Minimum Control Pulse: Hide All ▾

Maximum Control Pulse: Hide All ▾

Other Properties:

☐ Exclude from Simulation

☒ Exclude from PCB Layout

☐ Edit all properties as text

☐ Attach hierarchy module

☐ Hide common pins

The Calculations

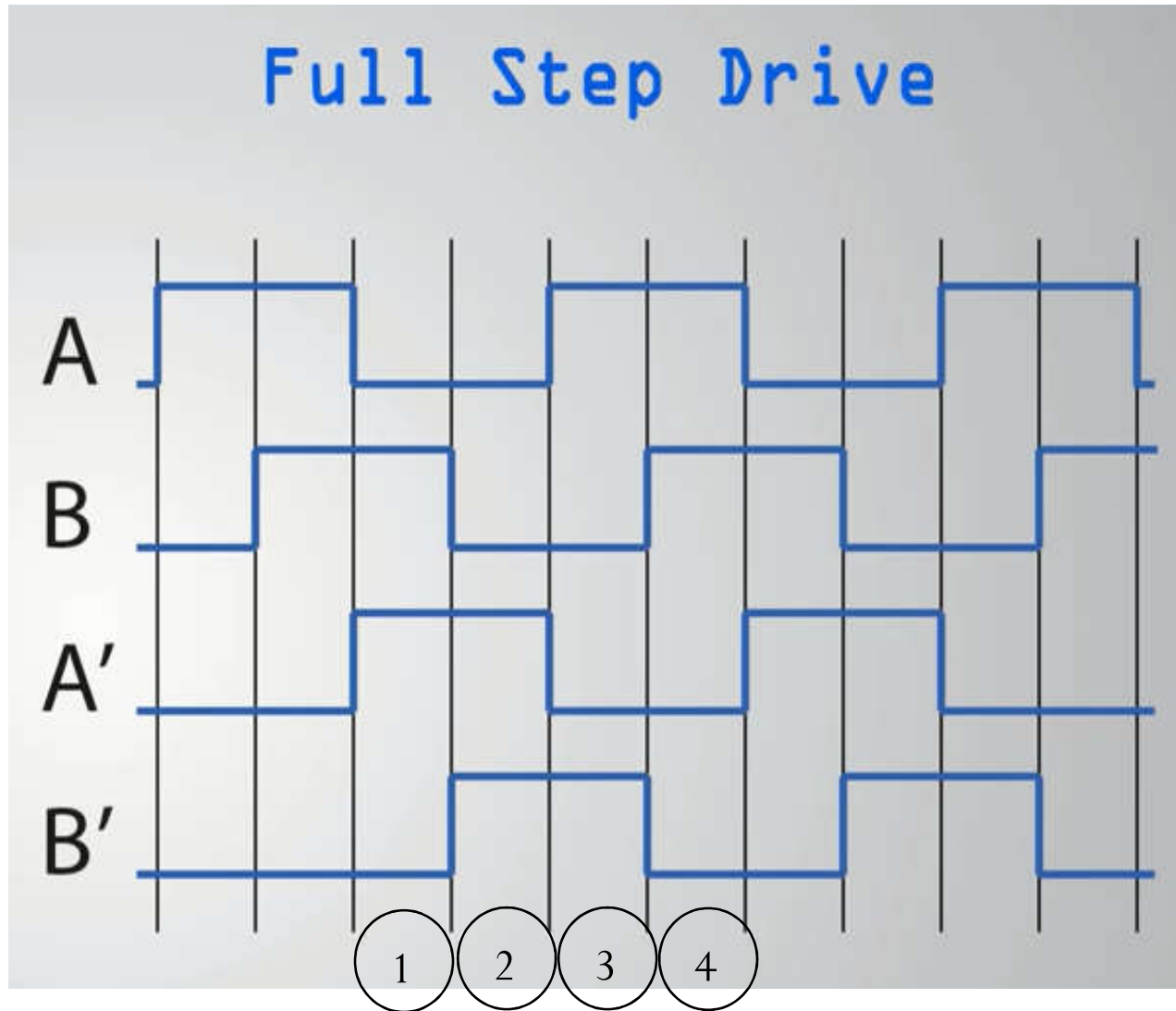
- The minimum time of the pulse = 1 ms
- The maximum time of the pulse = 2 ms
- Time period of the PWM signal = 20 ms ($f=50\text{Hz}$)
- Prescalar=8,
- So $f_{\text{osc}}=16\text{MHz}/8=2\text{MHz}$
- $T_{\text{tick}}=0.5\mu\text{s}$
- No. of pulse to make 20 ms $=20\text{e-}3/0.5\text{e-}6=40000=0\text{x}9\text{C}40$
- No. of pulse to make 1 ms $=1\text{e-}3/0.5\text{e-}6=2000$
- No. of pulse to make 2 ms $=2\text{e-}3/0.5\text{e-}6=4000$
- To start with
ICR1H=0x9C and ICR1L=0x40, and OCR1A=2000;

The Code for Interfacing A Servo Motor

```
#include <mega32.h>
#include <delay.h>
int i=0;
void main(void)
{
  DDRD.5=1;
  TCCR1A=(1<<COM1A1) |
  (0<<COM1A0) | (0<<COM1B1) |
  (0<<COM1B0) | (1<<WGM11) |
  (0<<WGM10);
  TCCR1B=(0<<ICNC1) |
  (0<<ICES1) | (1<<WGM13) |
  (1<<WGM12) | (0<<CS12) |
  (1<<CS11) | (0<<CS10);
  ICR1H=0x9C;
  ICR1L=0x40;
```

```
  while (1)
  {
    OCR1A=1999;
    for (i=0;i<200;i++)
    {
      OCR1A=OCR1A+10;
      delay_ms(10);
    }
    delay_ms(2000);
    for (i=200;i>0;i--)
    {
      OCR1A=OCR1A-10;
      delay_ms(10);
    }
    delay_ms(2000);
  }
}
```

Interfacing a Stepper Motor (Full Step)



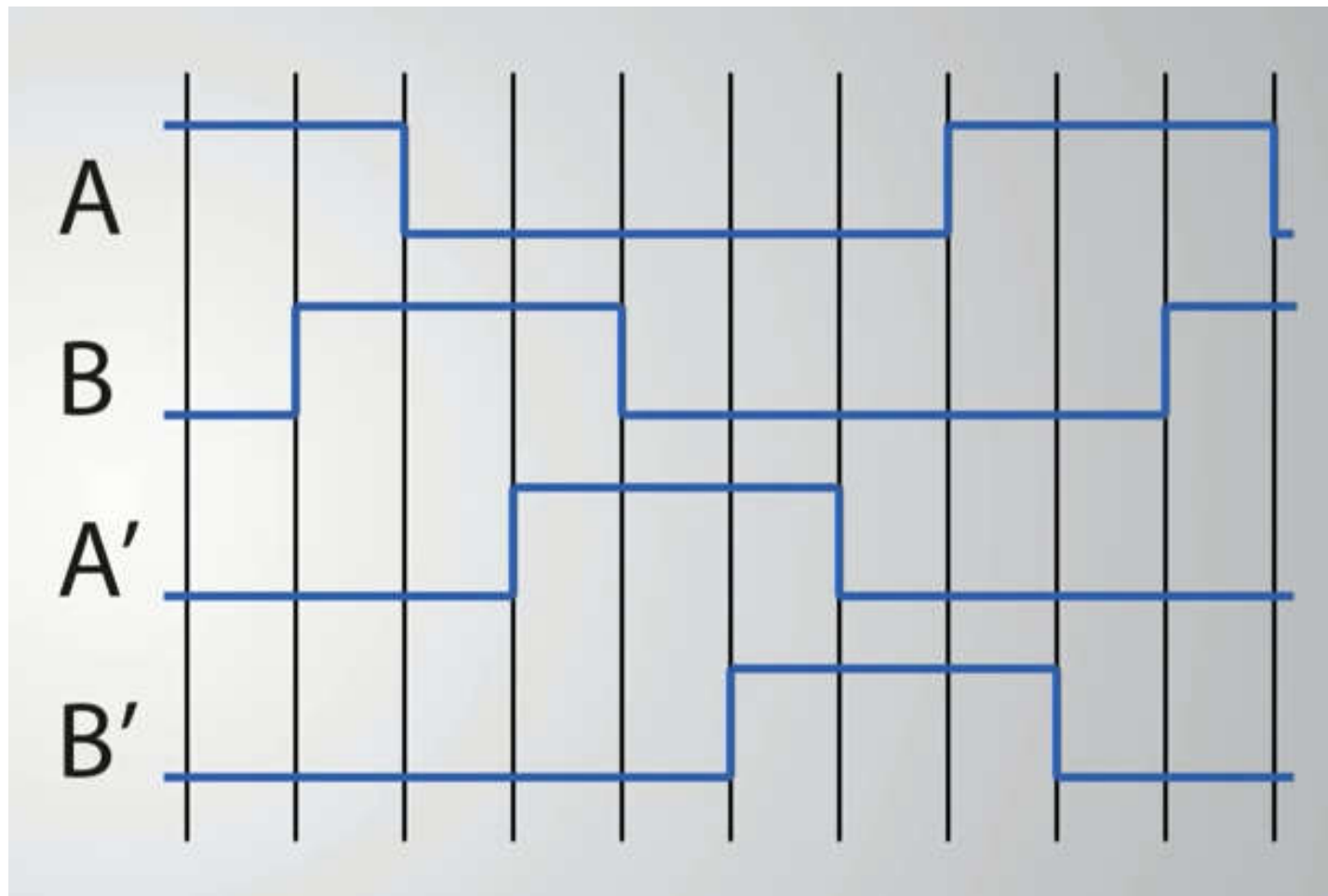
1-0110

2-0011

3-1001

4-1100

Interfacing a Stepper Motor (Half Step)



1-0110

2-0010

3-0011

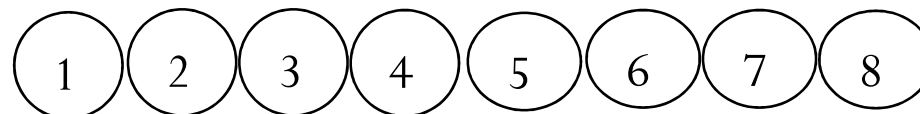
4-0001

5-1001

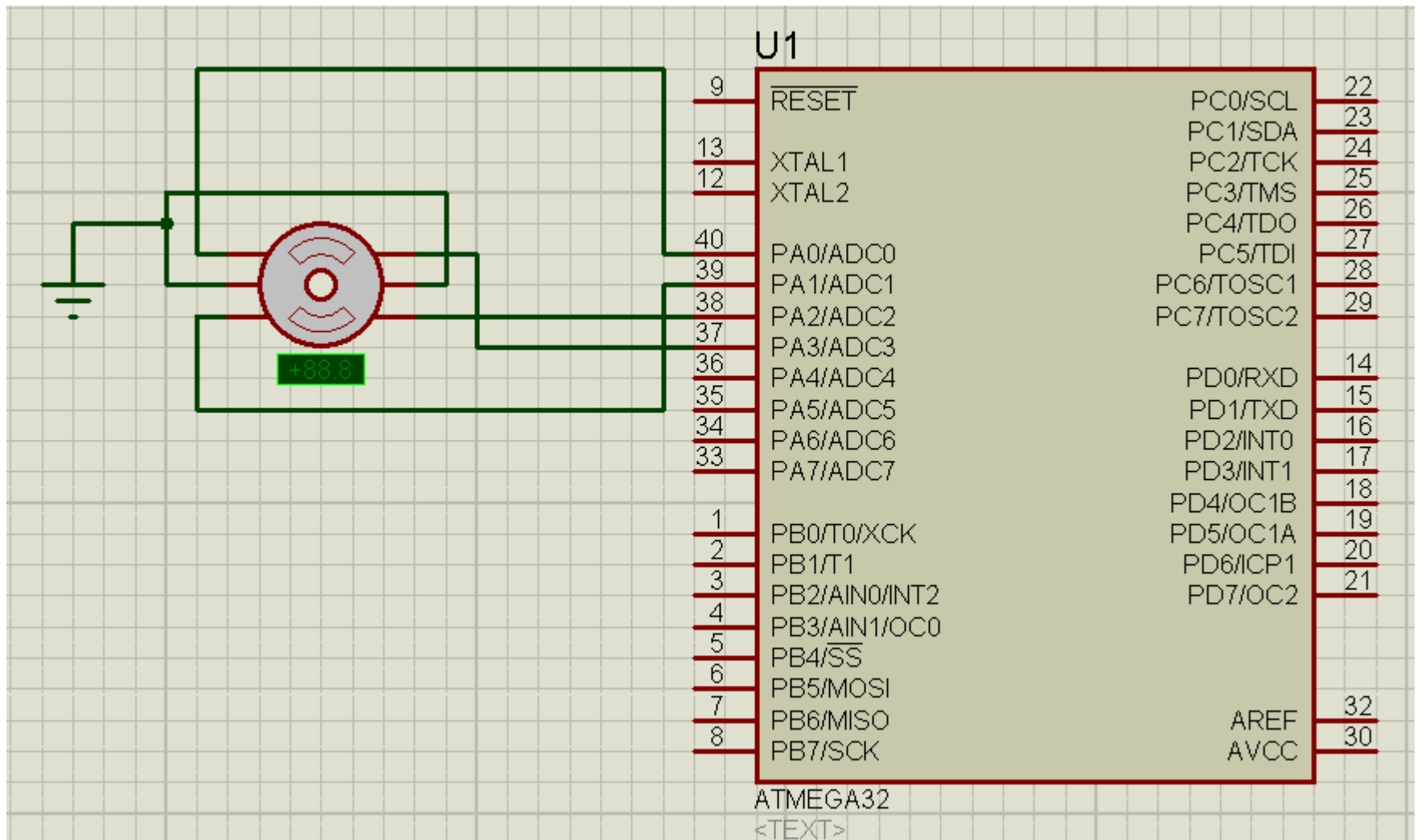
6-1000

7-1100

8-0100



Interfacing a Stepper Motor in Proteus



Properties of the Stepper Motor

Edit Component

Component Rference:

Component Value:

LISA Model File:

Nominal Voltage:

Step Angle:

Maximum RPM:

Coil Resistance:

Coil Inductance:

Hidden: ☐

Hidden: ☐

Other Properties:

☐ Exclude from Simulation

☒ Exclude from PCB Layout

☐ Edit all properties as text

☐ Attach hierarchy module

☐ Hide common pins

The Code for Interfacing for Full Stepping

```
#include <mega32.h>
#include <delay.h>
void main(void)
{
    DDRA=0xFF;
    while (1)
    {
        full_step();
    }
}
```

```
void full_step(void)
{
    PORTA=0b00000110;
    delay_ms(1000);
    PORTA=0b00000011;
    delay_ms(1000);
    PORTA=0b00001001;
    delay_ms(1000);
    PORTA=0b00001100;
    delay_ms(1000);
}
```

The Code for Interfacing for Half Stepping

```
#include <mega32.h>
#include <delay.h>
void main(void)
{
    DDRA=0xFF;
    while (1)
    {
        half_step();
    };
}
```

```
void half_step(void)
{
    PORTA=0b000000110;
    delay_ms(1000);
    PORTA=0b000000010;
    delay_ms(1000);
    PORTA=0b000000011;
    delay_ms(1000);
    PORTA=0b000000001;
    delay_ms(1000);
    PORTA=0b000001001;
    delay_ms(1000);
    PORTA=0b000001000;
    delay_ms(1000);
    PORTA=0b000001100;
    delay_ms(1000);
    PORTA=0b000000100;
    delay_ms(1000);
}
```


Thanks