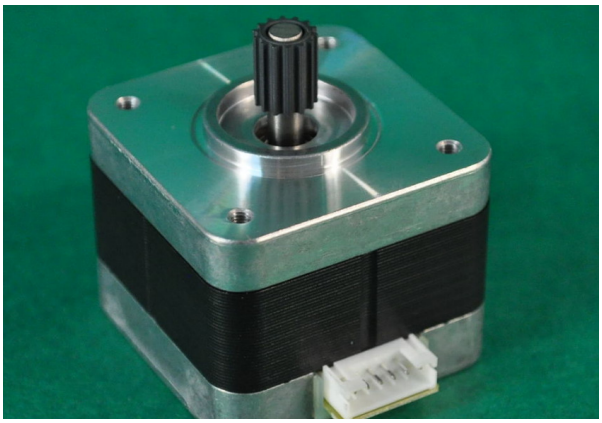
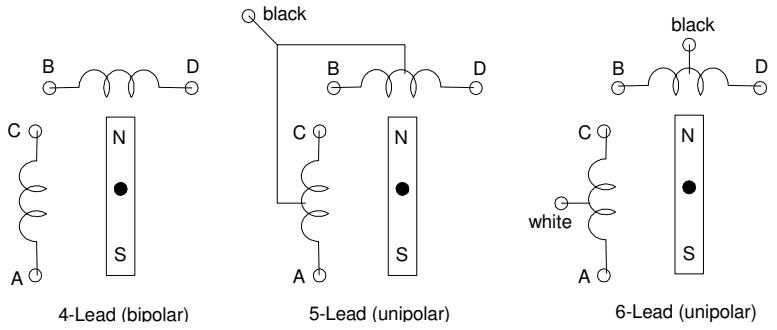
Stepper Motors in C



Unipolar (6 lead) stepper motorr. $1.50 from ebay
100 steps per rotation. 10.4V / 300mA / 0.28Nm holding torque

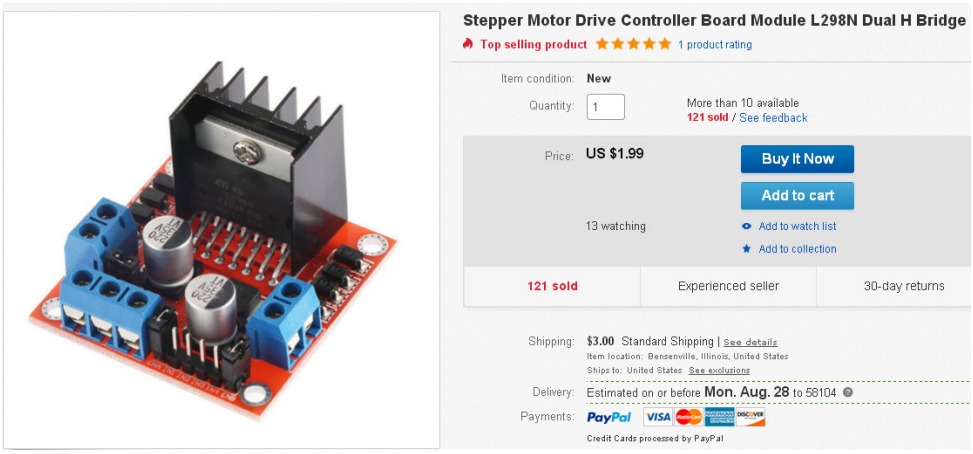
A stepper motor is a digital motor with two phases and 4, 5, or 6 leads. These leads connect to two sets of electromagets. The main difference is the addition of a center tap for the two electromagnets:



The motors we're using are either 6-lead (with the center tap not wired up so it looks like a 4-lead motor) or 5-lead.

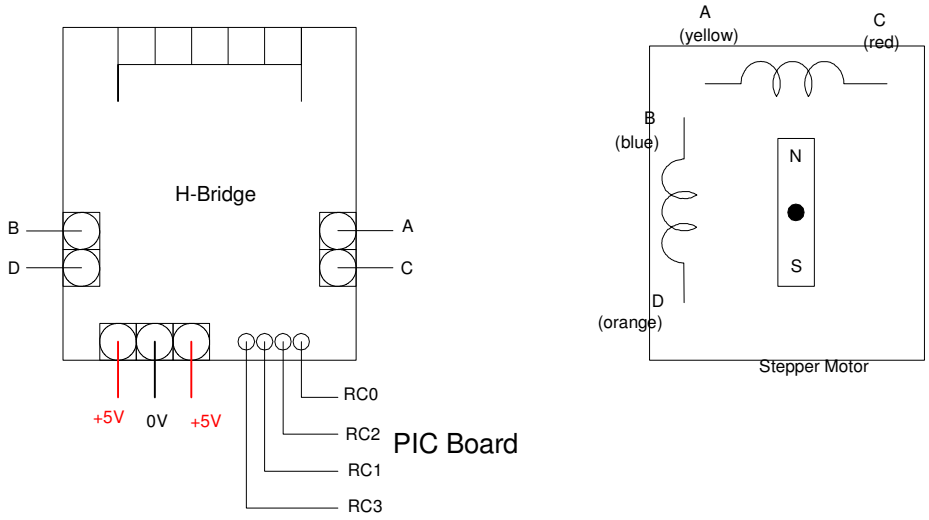
Case 1: 5V Operation

The stepper motor in your kits is rated at +10VDC. It will operate at 5V as well, just without as much holding torque. Each phase has a resistance of 30 Ohms, resulting in the stepper motor drawing 300mA. This is a little more than the 25mA rated for the PIC, so we need an H-bridge like the one in your kit:



H-Bridge capable of up to 24V, 3A (out), 3V to 6V in

To wire this H-bridge to your PIC board, use the following schematic:



Wiring for the H-Bridge. Note that PORTC pins alternate on the input

Recalling from previous lectures, to step the motor using full stepping (100 steps per rotation), the sequence sent to PORTC is

```
PORTC

0 0 0 1
0 0 1 0
0 1 0 0
1 0 0 0
repeat
```

For half-stepping (200 steps per rotation), the sequence is

```
PORTC

0 0 0 1
0 0 1 1
0 0 1 0
0 1 1 0
0 1 0 0
1 1 0 0
1 0 0 0
1 0 0 1
repeat
```

Sample Code:

Full-Stepping Every 100ms (Stepper1.C)

A simple way to walk through the 1/0 pattern is to use a table. The following program cycles through the table with each step taking 100ms

```
// Stepper1.C
// Drive a stepper motor at 100ms/step

// Global Variables
const unsigned char MSG[16] = "Stepper1.C      ";
unsigned char TABLE[4] = {1, 2, 4, 8};


   while(1) {
      STEP = STEP + 1;
      PORTC = TABLE[STEP % 4];
      LCD_Move(1,0);
      LCD_Out(STEP, 5, 0);
      Wait_ms(100);
      }
```

On PORTC, you should see the following bit pattern with it changing every 100ms.

```
RC3  RC2  RC1  RC0
 0    0    0    1
 0    0    1    0
 0    1    0    0
 1    0    0    0
 (repeat)
```

If you want to step the other way, simple decrement STEP every 100ms.

If you want to step faster, change the wait() function.

Adjust the speed the stepper motor steps using push buttons:

- RB4: Forward, 30ms / step
- RB3: Forward, 100ms / step
- RB2: Stop
- RB1: Reverse, 100ms / step
- RB0: Reverse, 30ms / step

Change the hard numbers in the previous code to variables, which are changed based upon the button you press:

```
// Stepper2.C
#include <pic18.h>

// Global Variables
const unsigned char MSG[16] = "Stepper2.C     ";
unsigned char TABLE[4] = {1, 2, 4, 8};
int STEP, N, TIME;

// Subroutine Declarations
#include "LCD_PortD.C"

// Main Routine

void main(void)
{
   unsigned int i, MS, DIR;

   TRISA = 0;
   TRISB = 0xFF;
   TRISC = 0;
   ADCON1 = 0x0F;

   DIR = 0;
   MS = 100;
   STEP = 0;

   LCD_Init();
   LCD_Move(0,0);
   for (i=0; i<16; i++) LCD_Write(MSG[i]);
   Wait_ms(100);


   while(1) {
      if(RB0) { DIR = -1;  MS =  30; }
      if(RB1) { DIR = -1;  MS = 100; }
      if(RB2) { DIR =  0;  MS = 100; }
      if(RB3) { DIR =  1;  MS = 100; }
      if(RB4) { DIR =  1;  MS =  30; }

      STEP = STEP + DIR;
      PORTC = TABLE[STEP % 4];
      LCD_Move(1,0);
      LCD_Out(STEP, 5, 0);
      Wait_ms(MS);
      }
   }
```

Case 3: Position Control via push buttons

Have the stepper motor go to the following positions based upon which button is pressed:

- RB4: 100 steps (360 degrees)
- RB3: 75 steps (270 degrees)
- RB2: 50 steps (180 degrees)
- RB1: 25 steps (90 degrees)
- RB0: 0 steps (0 degrees)

```
// Stepper3.C
// Position control of a stepper motor

// Global Variables
const unsigned char MSG1[16] = "REF             ";
const unsigned char MSG2[16] = "STEP            ";

unsigned char TABLE[4] = {1, 2, 4, 8};

// Subroutine Declarations
#include <pic18.h>
#include "LCD_PortD.C"


// main routine
void main(void) {
   unsigned int i, REF, STEP ;

   TRISA = 0;
   TRISB = 0xFF;
   TRISC = 0;
   ADCON1 = 0x0F;

   STEP = 0;
   REF   = 100;

   LCD_Init();
   LCD_Move(0,0);  for (i=0; i<16; i++) LCD_Write(MSG1[i]);
   LCD_Move(1,0);  for (i=0; i<16; i++) LCD_Write(MSG2[i]);
   Wait_ms(100);

   while(1) {
      if(RB0) REF = 0;
      if(RB1) REF = 25;
      if(RB2) REF = 50;
      if(RB3) REF = 75;
      if(RB4) REF = 100;

      if (STEP < REF) STEP = STEP + 1;
      if (STEP > REF) STEP = STEP - 1;

      PORTC = TABLE[STEP % 4];
      LCD_Move(0,8);  LCD_Out(REF, 5, 0);
      LCD_Move(1,8);  LCD_Out(STEP, 5, 0);
      Wait_ms(30);
      }
   }
```

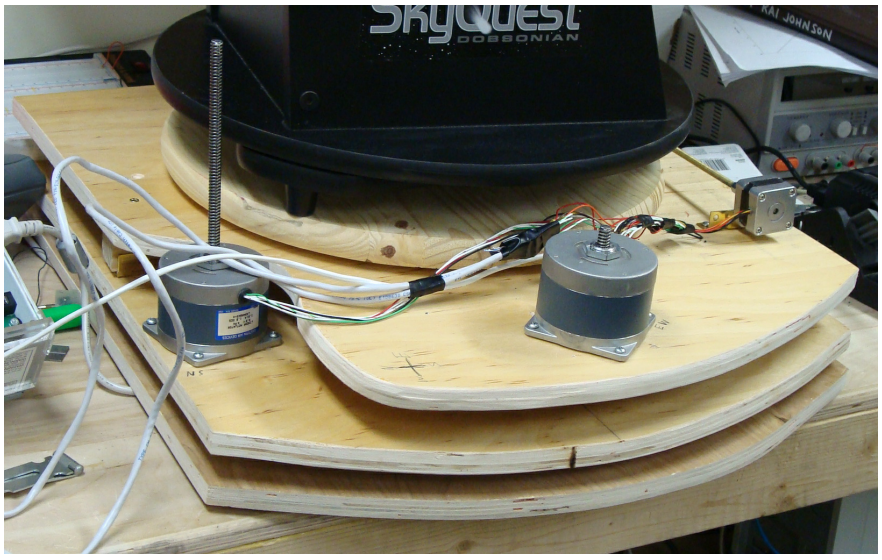
Linear Actuators:

A linear actuator is a stepper motor where the shaft is stationary and the motor rotates around the shaft. In addition, the shaft is a screw, converting rotation to displacement.

The stepper motors we have use a 3/8 16 screw, meaning 16 rotations move the screw one inch. Using half-stepping, this works out to

$$\left(200\ \frac{\text{step}}{\text{rotation}}\right)\left(16\ \frac{\text{rotations}}{\text{inch}}\right) = 3200\ \frac{\text{steps}}{\text{inch}}$$

These motors are also able to apply up to 350 lb force, so they're useful for lifting heavy loads with great precision. For example, you can build an equatorial platform with these. Here, the motors tilt the platform left and right, forward and back to track stars.



Microstepping:

A stepper motor is actually an AC synchronous machine:

- If you send a square wave to the motor, it jumps from one angle to another: it is operating as a stepper motor. For the motor above, it has 100 steps per rotation.
- If you send a sine wave to the motor, it smoothly moves from one angle to another: it is operating as an AC synchronous machine. For the motor above, it rotates once per 25 cycles:

$$\left(\frac{100 \text{ steps}}{\text{revolution}}\right)\left(\frac{1 \text{ cycle}}{4 \text{ steps}}\right) = \left(\frac{25 \text{ cycles}}{\text{revolution}}\right)$$

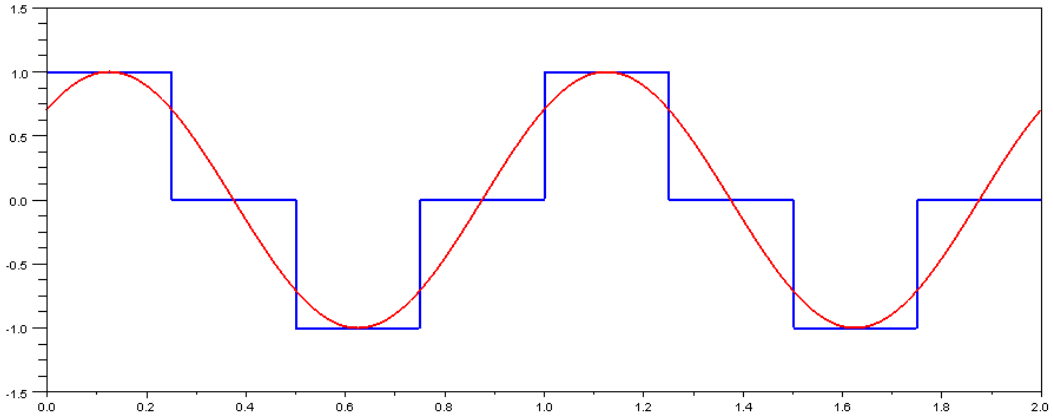
If you apply a 60Hz sine wave to the above motor (sine on phase AC, cosine on phase BD), it will spin at

$$\left(\frac{60 \text{ cycle}}{\text{sec}}\right)\left(\frac{1 \text{ revolution}}{25 \text{ cycles}}\right) = 2.4 \text{ rps} = 144 \text{ rpm}$$

For example, if you look at the voltage applied to phase AC, when using full stepping the voltage V_{AC} looks like the blue line in the following figure (shown for two cycles):

```
A B C D
1 0 0 0      VAC = 1
0 1 0 0      VAC = 0
0 0 1 0      VAC = -1
0 0 0 1      VAC = 0
repeat
```

If you use a sine wave instead, you'll be able to position the motor inbetween steps. This is shown in the red line below:



Signal sent to V_{AC} when using full steps (blue line) or microstepping (red line).
V_{BD} would be 90 degrees out of phase (cosine vs. sine).