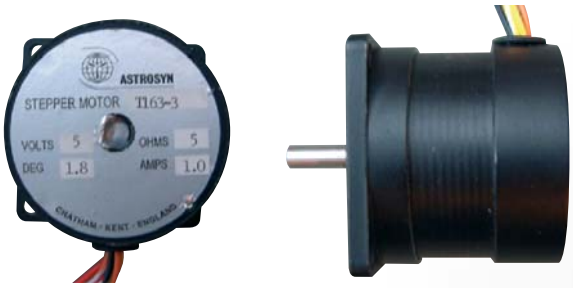
PICs and Stepper Motor control

Introduction to Stepper Motors

PICs have been in fairly consistent use for several years in schools and yet we still see colleagues struggling with old style dedicated stepper motor chips and trying to generate suitable clock pulses. To revitalise stepper motor work in school follow this guide to programming, wiring and controlling stepper motors using TEP's Chipfactory.

A typical motor available to schools is the T 163-3 stepper motor, an industry standard motor with four separate coils it consists of a permanent magnet rotor revolved by energising each coil or pair of coils in a defined sequence. This provides 'steps' of 1.8°, hence the name stepper motor. 1.8° provides a motor with 200 steps to complete one revolution. These motors are readily available both commercially and as surplus stock from a number of suppliers. The price of them continues to fall as larger volumes of surplus stock hits the market, making them increasingly viable alternatives to permanent magnet motors. They are used widely in scanners, copiers and printers and can be typically seen in D&T workshops driving axis drives on CNC machines. They lend themselves well to accurate positional control in projects and assignments. 'Specs' are generally labelled on motor cases.



Most motors are best described as **'Unipolar'** motors and will have 5, 6 or 8 wires leading from the casing. Some surplus motors are Bipolar and have only two coils. Also many surplus motors do not have a coil wiring diagram supplied. It can seem a hassle, however using a multi-meter set to low range resistance the required pairs of wires to each coil can be identified. A reading between 1-10Ω indicates a pair of wires to a coil. A much higher reading indicates an open circuit and no connection. Once all four coils pairs are identified you will still need to determine the sequence for the four coils by 'hot wiring' each pair of wires in turn to a supply and observing the step and direction of the motor output to determine the sequence of the four inputs. Of course supplied diagrams with the motors is easier but even those we have found are not infallible. Most leads will be colour coded and easy to identify.

Five or six wire motors will be connected as per the diagrams (Fig1). Eight wire motors will have two connections per coil.

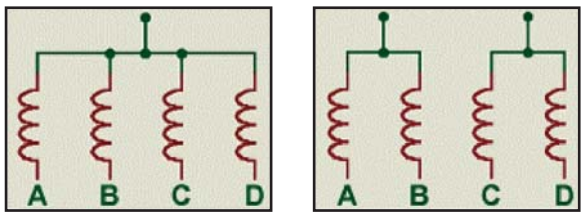
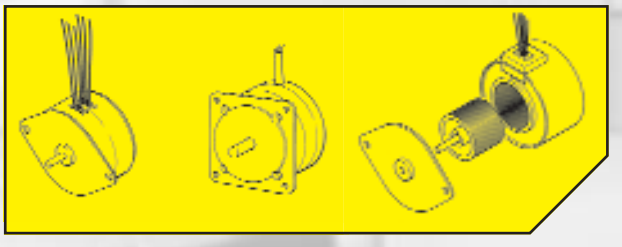


Fig1

Other surplus motors tend to have a step angle of 7.5° which give 48 steps per revolution. Operated by 5-6 volts each coil has such a low resistance they 'draw' around 1 amp when energised. Thus it is advisable to provide a separate power supply for the motor driver and stepper motor so that the voltage of the PIC controller is not dropped causing the program to fault. Stepper motors do not have a high torque output and are best suited to small loads and precise control. Gearing the output using a worm drive is an ideal way of preventing excessive load causing motor slip and allows more accurate angular control and increase torque. Stepper motors are 'open feedback' and provide no inherent feedback of position and so if the output is impeded they can lose position. Hence the need for reset and homing switches on CNC machines that use them.

Due to their rotor inertia if high speed step pulses are applied instantly the motor under-steps and loses position. The same is true if at high speed the step signal is switched off again the motor will overstep and go beyond a desired stop position. Hence stepper motors need to be 'stepped' precisely and sped up and slowed down from and to a maximum step speed. Position is accurately determined by using the PICs counting ability to add or subtract pulses from a nominal 200 steps to ensure exact position. Unipolar motors are essentially two sets of coils assembled back to back with a heavy 'rotor' in the centre. They will happily accept 200-300 steps per second or at a push up to 500 steps this is broadly 90-150 rpm. The slower the rpm, the greater the torque. This is the reverse of continuous current DC motors that require higher revolutions to maintain or increase torque. Full steps of 7.5° and 1.8° can be improved on by **'half stepping'**. Industrial applications can involve even smaller step angles being achieved called **micro-stepping**. Overpowering the stepper motor by a 100% increase in rated supply voltage is also possible and gives much greater torque but will make the motor 'warm up' and prematurely expire.



The motor cases have suitable flanges and mounting plates making them easy to fix onto surfaces in projects. The other advantage of course is when not being stepped they stop, unlike continuous current motors that can run on long after the supply is switched off!

Continued overleaf

PIC Control of Stepper Motors

All the above advice holds true no matter which electronic drive system you choose below is a described method using a PIC project board and the 16F627 or 16F84 PIC. With the stepper motor only needing four inputs a lower cost 8pin PIC method is also possible and is planned as a future PICAXE article.

Equipment

For this project work you will need:

- 1 High Power project board
- 1 PIC 16F627 or 16F84
- 1 Motor Driver Chip L293D
- 2 3AA Battery Boxes
- 6 AA Batteries
- 2 Push to make switches
- 2 Battery Snap connectors
- 1 Unipolar Stepper Motor (5 volt)

The simplest control with a stepper motor is to turn each coil on and then off in turn as in figure 2 after four steps the cycle is repeated so the program for simple rotation is quite straightforward. From figure 3 you can see each pair of coils can be energised to increase the turning torque in four steps which are then repeated.

Step	Coil A	Coil B	Coil C	Coil D
1	ON	OFF	OFF	OFF
2	OFF	OFF	ON	OFF
3	OFF	ON	OFF	OFF
4	OFF	OFF	OFF	ON

Fig 2

Step	Coil A	Coil B	Coil C	Coil D
1	ON	OFF	OFF	ON
2	ON	OFF	ON	OFF
3	OFF	ON	ON	OFF
4	OFF	ON	OFF	ON

Fig 3

Step	Coil A	Coil B	Coil C	Coil D
1	ON	OFF	OFF	OFF
2	ON	OFF	ON	OFF
3	OFF	OFF	ON	OFF
4	OFF	ON	ON	OFF
5	OFF	ON	OFF	OFF
6	OFF	ON	OFF	ON
7	OFF	OFF	OFF	ON
8	ON	OFF	OFF	ON

Fig 4

To run the stepper motor in reverse is to simply reverse the step sequence so 4 is followed by 3 followed by 2 followed by 1. This sequence forward and reverse is full step, to **'half step'** requires us to consider combining both single and pairs of coils in turn as in figure 4. Note that the total number of steps is now doubled to eight.

So that the rotation of the output shaft can be readily seen, a pulley or pointer should be attached. In practice a pulley or pinion gear is likely to be attached and fixed with a grub screw.

After identifying the wiring correctly it is a straightforward task to connect the motor to the project board. Solder the two battery snaps to each of the power pads at the bottom of the board, two on each side of the reset button. Take care to ensure the polarity of the leads red to positive and black to negative. There is a link resistor (R9) at the top left hand side of the board that needs removing with side cutters. This is so the PIC and the motor run off separate supplies.

The four individual coil wires are the – **supply** and are soldered or connected with terminal blocks to outputs 0,1,2 and 3 which are the FET driven outputs on the lower right hand side of the project board. The remaining wires are the + **supply** wires and can be connected to a common supply V at the top right hand side of the project board. The L293D motor driver chip can be inserted carefully in the right hand side DIL socket with the chip's notch to the top.

The two push to make switches are attached to short lengths of insulated stranded wire and soldered to inputs 2 and 3 respectively on the left hand side of the board. You are now ready to program.

A test-drive program is the simplest one first to check the motor and project board wiring.

```
Test-Drive Program

00    out   000
01    out   009
02    wait  001
03    out   005
04    wait  001
05    out   006
06    wait  001
07    out   010
08    wait  001
09    goto  001
```

```
Program description

sends outputs 0 and 3 high
0.1 second delay
sends outputs 0 and 2 high
0.1 second delay
sends outputs 1 and 2 high
0.1 second delay
sends outputs 1 and 3 high
0.1 second delay
Loop to line 1
```

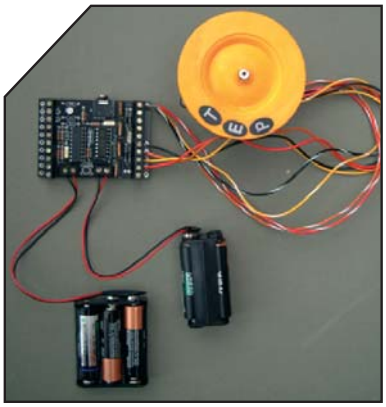
Hopefully you should now have a stepping motor that takes about 20 seconds to complete a free running revolution either clockwise or anti-clockwise. Any 'dodging' back and forth with the motor means it is wired up out of sequence to the board and will require some rewiring.

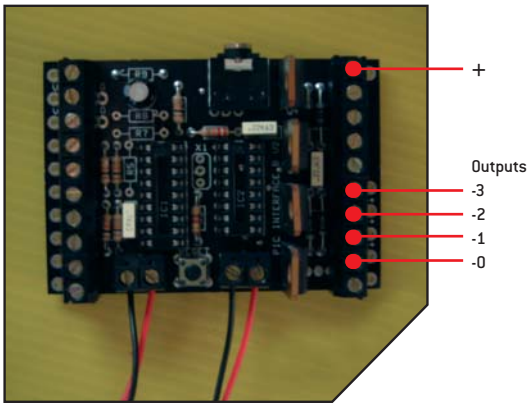
```
PTM Program

00          out   000
01    if 2 on goto  003
02          goto  000
03          out   009
04          wait  001
05          out   005
06          wait  001
07          out   006
08          wait  001
09          out   010
10          wait  001
11          goto  001
```

Adding a couple of extra lines to the program will give us push button control, with the PTM switch on input 2.
Notice how line 01 looks for a signal and if switched on goes through the program stepping the motor four times and looping back to see if the switch is on or off again. Line 2 simply sends the program in a loop to keep checking for an input on input 2.

Faster stepping needs to be considered as one revolution every 20 seconds or 3 rpm is slow!






```
Simple Forward/Reverse

00          out     000
01    if 2 on goto  004
02    if 3 on goto  013
03          goto    000
04          out     009
05          wait    001
06          out     005
07          wait    001
08          out     006
09          wait    001
10          out     010
11          wait    001
12          goto    000
13          out     010
14          wait    001
15          out     006
16          wait    001
17          out     005
18          wait    001
19          out     009
20          wait    001
21          goto    000
```

Controlling forward and reverse steps is equally easy by including a second PTM switch on input 3 and simply reversing the sequence in a second subroutine. This of course only gives accuracy to the nearest 4 steps or 7.2° . You would need more **if** statements between each step to get directional control accuracy down to one step of 1.8°

Our final task in this installment is to look at step counting, one of the really important benefits of using a stepper motor. So with 200 steps per revolution we can accurately position the output shaft at any angle as a multiple of 1.8° eg: 3.6v, 9°, 27°, 54°, 180° etc.

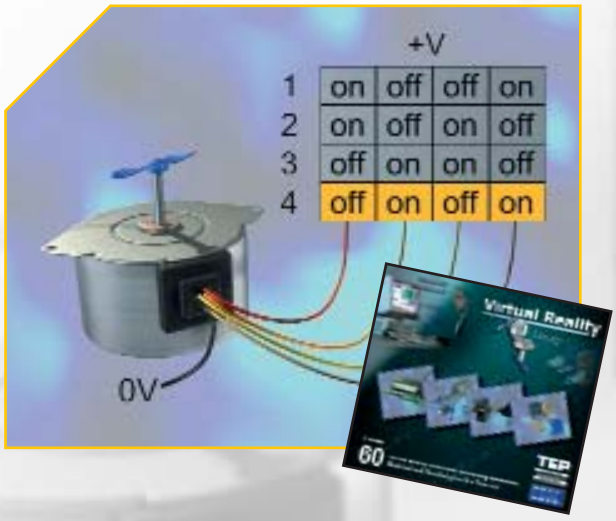
So for a selected angle of 180° that is 180/360 X 200 steps= 100 steps (remember: there are four sequences in full stepping a motor so 100/4= 25 counts up or down). There are many angles that are not going to possible with a 1.8° motor for example common angles like 60° and 120° as they are not multiples of 1.8°. Counting up or down requires the use of a variable in this case x or y. We will take a closer look as step counting in the next issue.

What sort of stepper motor projects?

How about a turntable that tracks the sun or moon, a steerable radio mast or controlling the launchpad for the TEP rocketfactory? As well as the more obvious robot arms and legs or sorting ball bearings into a bag or lighting filters in front of a spotlight or a useful dividing table...The possibilities are endless.

The most useful motor featured in this article is available from **Teaching Resources** and is **order code: EW2 017** and operates on a 5 volt supply.

There is also a really useful animation on stepper motors on the **TEP Virtual Reality CD-ROM** along with a host of other VR images on circuits and processes too.



Nick will be pleased to hear from staff using stepper motors in projects and is available at: **nickbaldwin@enterprise.net**