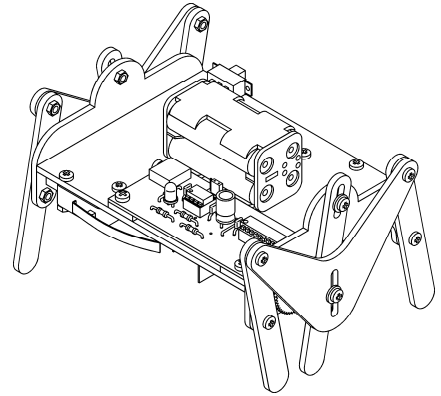


# ***BEEBLE***

## **DESCRIPTION**

***BEEBLE*** is a six-legged user programmable device. When it hits an object, ***BEEBLE*** backs away, turns around and continues in another direction, imitating the actions of a real beetle. ***BEEBLE*** also changes direction when it doesn't bump into anything for some time.

***BEEBLE*** has a lever arm microswitch at either end to detect an obstruction. It is driven by two motors, each with its own gearbox, and is controlled by a PICAXE-08M microcontroller.



## **INVESTIGATION**

This project provides a number of different aspects of the ***BEEBLE*** for investigation. Some ideas are listed below, in "Further Development" (at the end of this unit), as well as in the Theory section.

- The ***BEEBLE*** you design can be evaluated for individuality, walking speed and smoothness, component layout and space efficiency.
- Create your own unique ***BEEBLE*** design based on our drawings, which focus on component relationships, rather than dimensions. This provides scope for individual variation.
- Evaluate the suitability of various materials. For example: Aluminium, Perspex and PVC.
- Investigate adding more "intelligence" or different sequences to the program.
- Investigate other devices that could use two bi-directional motors and one switch input, such as the ***DIZZY***.

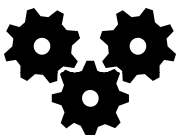
## **1. COMPONENTS REQUIRED**

### **1.1 COMPONENTS SUPPLIED.** The following components are supplied in the kit:

1 x Printed Circuit Board (Dizzy PCB)	1 x Header strip - 3 pins
1 x PICAXE-08M (microcontroller IC)	1 x Jumper (Header Socket 2 pin)
1 x L293D (motor driver IC)	2 x Microswitch - long lever
1 x IC Socket (8 pin)	1 x Sliding Switch (small)
1 x IC Socket (16 pin)	1 x Battery Holder - 4AA
1 x Red LED (Light Emitting Diode)	4 x Bolt - M3 x 20mm
5 x Capacitor - 0.1uF (monolithic or equivalent)	4 x Nut - M3
1 x Capacitor - 100uF (electrolytic)	12 x Bolt - M3 x 16mm
1 x Resistor - 220 Ohms (Red-Red-Brown-Gold)	12 x M3 "Nyloc" Nut
2 x Resistor - 10k Ohms (Brown-Black-Orange-Gold)	16 x M3 (0.5mm thick) Flat Washer
1 x Resistor - 22k Ohms (Red-Red-Orange-Gold)	4 x Self-Tapping Screw - 2.6mm x 4mm
1 x Stereo socket - 3.5mm	

### **2 x Sets of Multi-ratio Gearbox "kits for kits" (each containing):**

1x Multi-ratio Gearbox case	2x 50T/10T Spur gears (white)
1x 4.5V Electric Motor (round)	1x 50T/10T Spur gear (yellow)
2x 2.5 dia x120 long steel rod	2x 12T Pinions 2.4 hole
1x 3mm inner dia 1.0 thick Washer	1x 10T Pinion 1.9 hole
2x M2.6x 4 self-tapping Screws	



Revised: 18th April 2009

**SCORPIO TECHNOLOGY VICTORIA PTY. LTD.**

A.B.N. 34 056 661 422

17 Inverell Ave., Mt. Waverley, Vic. 3149

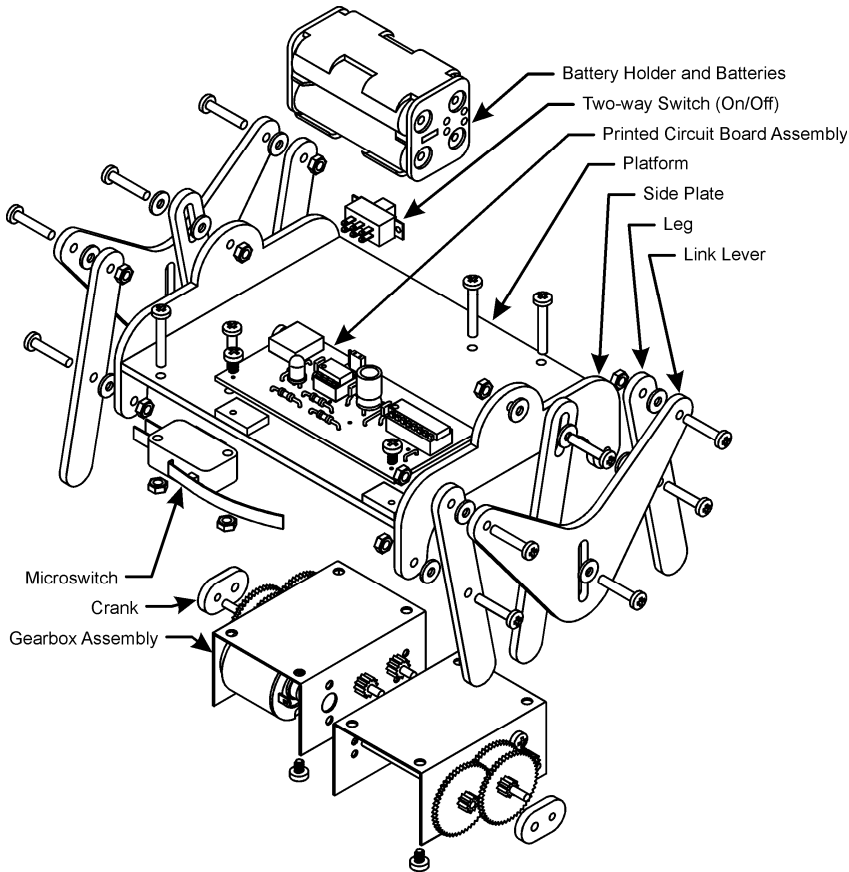
Tel: (03) 9802 9913 Fax: (03) 9887 8158

[www.scorpiotechnology.com.au](http://www.scorpiotechnology.com.au)

## 1.2 ADDITIONAL REQUIREMENTS

- 1.2.1 The following items are available from us, but need to be ordered separately: 2.3mm, 2.6mm and 3.5mm diameter drill bits; and 1.5 volt AA batteries (4 required).
- 1.2.2 Electric hook-up wire in assorted colours.
- 1.2.3 Material for the platform and legs. We used 3.0mm PVC sheet. (For plastic sheet refer to the Telstra Yellow Pages under the heading "Plastics Fabricators.")
- 1.2.4 A PICAXE serial interface cable. This can be bought or made (Refer section 6.1).
- 1.2.5 To install the PICAXE programming editor software requires a PC running Windows 95 or later with approximately 20MB free space. Any PC that runs the Windows operating system will work in textual 'BASIC' mode, however a Pentium 4 processor or later is recommended for graphical flowcharting (we also used an iMac running OSX, Parallels and Windows XP with a generic USB to serial adapter.)
- 1.2.6 A PC with 9-pin serial (RS-232) interface (the PC may require an USB to serial adapter.)
- 1.2.7 The PICAXE editor, which can be downloaded for free from [www.rev-ed.co.uk](http://www.rev-ed.co.uk) or from [www.picaxe.co.uk](http://www.picaxe.co.uk).

## 2. DESIGN CONSIDERATIONS



### DESIGN CONCEPT

- 2.2.2 Determine a method of attaching the vertical side pieces to the platform (we used PVC adhesive, with small offcuts added to reinforce the joints - not shown in the drawings).
- 2.2.3 Position both gearboxes with their driving shafts in line with the centre of the platform, and a suitable distance apart (what happens if they're not in the centre?). Define the axle shaft length.
- 2.2.4 Determine positions and attaching methods for: the PCB assembly (we glued spacers to the platform and used self-tapping screws to attach the PCB); the on/off switch; the microswitches and the battery holder (preferably near the centre of gravity).  
Hint: Investigate alternate attachment methods, such as: hot-melt glue, screws, double-sided tape and hook-and-loop tape ("Velcro" or equivalent).

## 2.1 PLANNING

- 2.1.1 *BEETLE* consists of a platform to which the components are attached. Legs and a link lever are located on each side of the platform.
- 2.1.2 Before starting construction, plan, lay out and draw the components. Look at the *BEETLE* as a complete unit, and not just as separate parts. Our drawings can be used as a starting point for your design.

## 2.2 DESIGN

- 2.2.1 Determine dimensions for the platform, side pieces, legs and leg links.

2.2.7 Determine a suitable operating speed. The *MULTI-RATIO GEARBOX* kit provides a choice of 3 gear ratios. Before starting, the desired gearbox ratio (which affects the device's speed) must be chosen, as this defines the parts to be used, and the assembly procedure (as detailed in Section 5: Assembling the Gearbox).

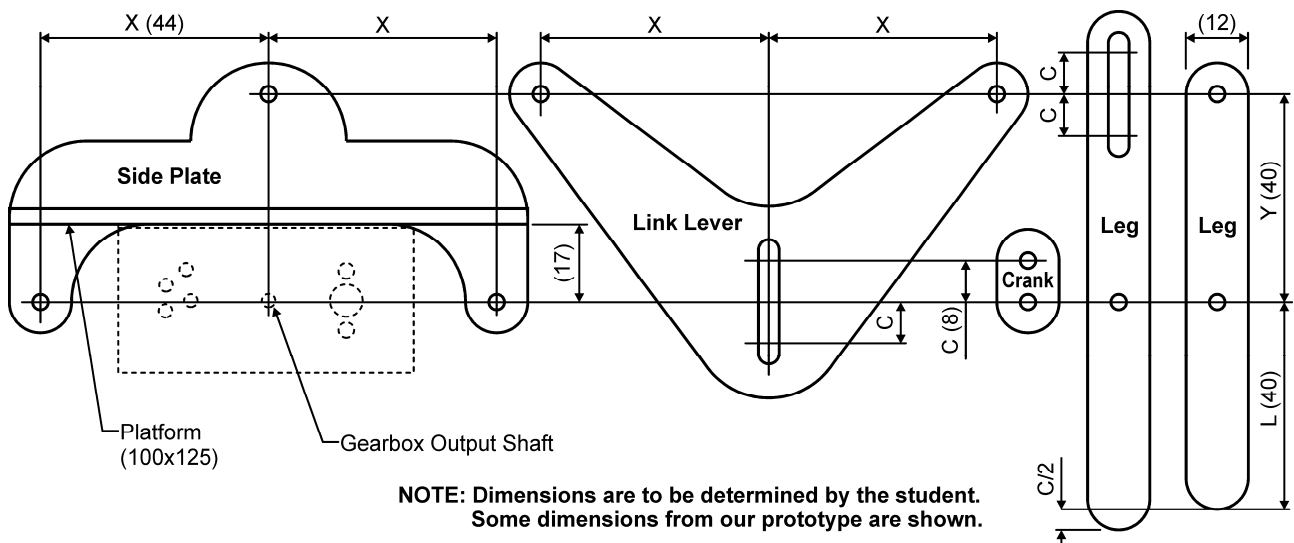
Note: for details on gear ratios etc, refer to the section on Assembling the Gearbox

Note: For our *BEEBLE* we used the TRIPLE-reduction version of the gearbox.

### 2.3 DIMENSIONAL RELATIONSHIPS

Use our drawings as a guide only. Consider designing more "insect-like" body shapes and leg profiles.

Note: these drawings are designed to show the relationships between the various dimensions, rather than defining the sizes that the *BEEBLE* should be.

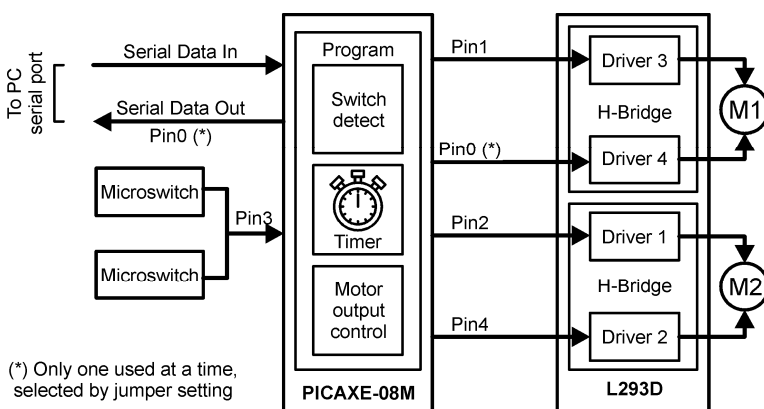


## 3. HOW THE CIRCUIT WORKS (THEORY)

### 3.1 CIRCUIT AND PROGRAM OVERVIEW

NOTE: PICAXE documentation refers to "Input Pins" and "Output Pins", which are not the same as the physical pins on a device. To avoid confusion, in this document "leg" means a physical pin of an integrated circuit and "pin" means a logical input or output.

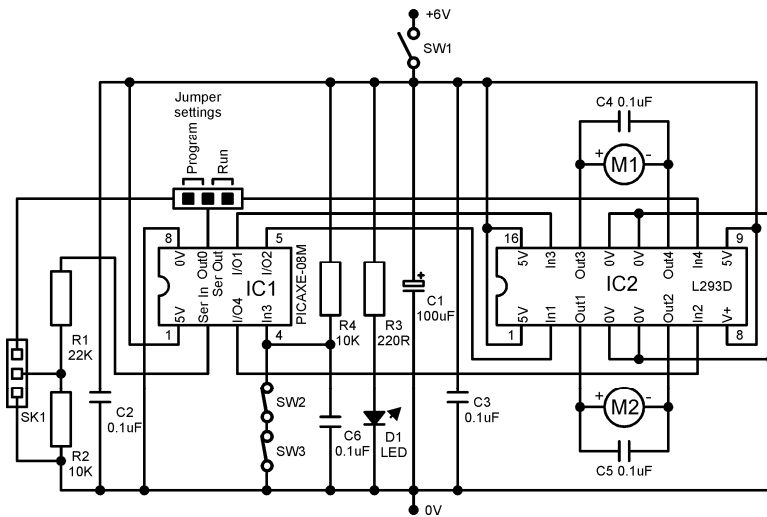
3.1.1 A program can be downloaded into the PICAXE-08M memory to control the motors by input from switches and an internal timer. The program can be customised as desired.



**BLOCK DIAGRAM**

3.1.2 Pin0 on the PICAXE-08M is shared between the download circuit and one of the motor outputs. To download a program, move the jumper to the "program" position. This connects pin0 to the stereo socket. Using the download cable, connect the stereo socket to a serial port on the PC and download the program.

3.1.3 When the program has downloaded, move the jumper to the "run" position. The PICAXE-08M then executes the commands in the program.



**CIRCUIT DIAGRAM**

Each H-bridge circuit (consisting of two "driver" blocks) controls the direction of one motor. Consequently, the motors turn backwards, forwards and stop as commanded by the PICAXE.

### 3.2 ABOUT PICAXE MICROCONTROLLERS

3.2.1 The PICAXE\* is a type of IC (Integrated Circuit) called a microcontroller - another name for a single chip computer. The PICAXE has similar features to a PC: CPU (central processing unit), RAM (random access memory), ROM (read only memory), I/O (input/output) lines, timers and A/D (analogue to digital) converters. \* PICAXE is a trademark of Revolution Education Ltd.

3.2.2 The Flash memory (EEPROM - Electrically Erasable Programmable Read Only Memory) in a PICAXE allows it to be reprogrammed many times (typically 100,000). This means that you can develop a program and constantly check the effects of changes.

3.2.3 A PICAXE program is created using an easy to learn version of the BASIC programming language (our preferred method) or using flowcharting software.

NOTE: The PICAXE is supplied containing 'bootstrap' code that enables you to download your program using the serial cable. Do not substitute the PICAXE with a blank PIC microcontroller.

### 3.3 PICAXE-08M (IC1)

3.3.1 When the program runs, it reads the status of the two microswitches and controls the direction of the two motors. In this case, the PICAXE behaves as a simple neural network.

3.3.2 PICAXE Leg 1 is connected to the positive terminal (+6V) of the power supply (batteries).

3.3.3 PICAXE Leg 2 (serial data in) is used only when transferring a program from a serial port on your PC (COM1: to COM4:) to the PICAXE. The 22k Ohm (R1) and 10k Ohm (R2) resistors must be present for reliable operation. Do not substitute other resistor values.

3.3.4 PICAXE Leg 3 (pin4 in the program) is connected to one of the inputs to the motor driver IC to control motor M2.

3.3.5 PICAXE Leg 4 (pin3 in the program) is connected to the microswitches (SW2 and SW3). Leg 4 is normally tied to ground through the microswitches, which are wired in series. When either of the microswitches is opened, positive battery voltage appears on Leg 4 through resistor R4.

3.3.6 PICAXE Leg 5 (pin2 in the program) is connected to one of the inputs to the motor driver IC to control motor M2.

3.3.7 PICAXE Leg 6 (pin1 in the program) is connected to one of the inputs to the motor driver IC to control motor M1.

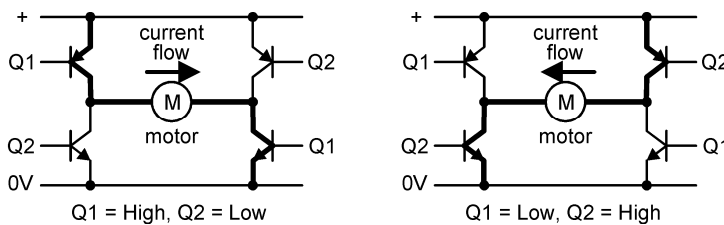
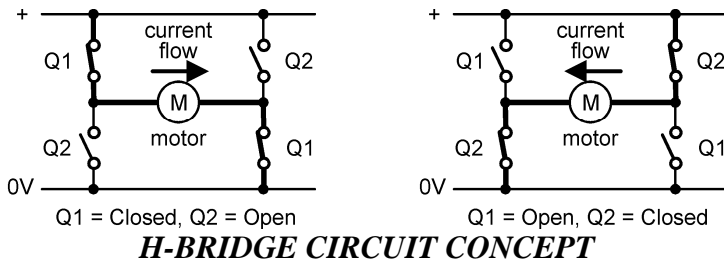
3.1.4 When our program starts running, both motors are set to run in the "forward" direction. The main program loops continually to check the status of both switches on pin3 and an internal timer. When either switch has been pressed or the timer has expired, the PICAXE-08M outputs a timed sequence of high/low voltages on pin0, pin1, pin2 and pin4.

3.1.5 Responding to the high/low voltage sequence on its inputs, the L293D opens/closes the electronic switches in its H-bridge circuits.

- 3.3.8 PICAXE Leg 7 has two functions selected by the jumper on the header strip. In the "program" position, the program can be downloaded into the PICAXE. In the "run" position, Leg 7 (pin0 in the program) is connected to one of the inputs to the motor driver IC to control motor M1.
- 3.3.9 PICAXE Leg 8 is connected to the negative terminal (0V) of the power supply (batteries).

### 3.4 MOTOR DRIVER L293D (IC2)

- 3.4.1 The L293D motor driver (IC2) contains two H-bridge circuits. Each of the "driver" blocks contains transistors that work as electronic switches.



- 3.4.2 Changing the voltage applied to the input pins changes the direction of current flow through the motor.

- Q1 low, Q2 low - motor stop
- Q1 high, Q2 low - motor forward
- Q1 low, Q2 high - motor reverse
- Q1 high, Q2 high - motor stop

- 3.4.3 The motor driver IC is simpler to use and fault-find than an equivalent circuit built using individual transistors and resistors. To view H-Bridge circuits constructed from transistors, see the "Wanderer" teaching unit on the Scorpio Technology website.

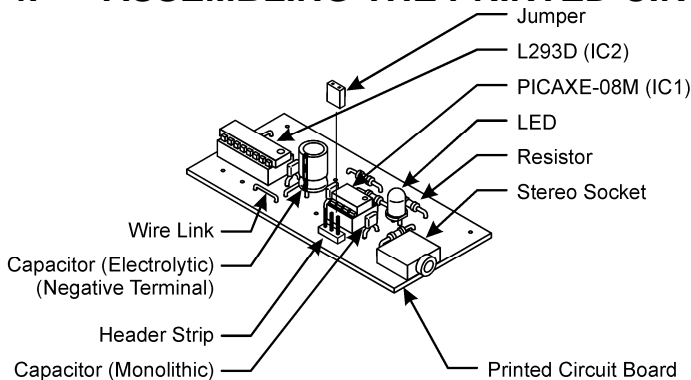
### 3.5 CAPACITORS

- 3.5.1 Capacitor C1 (100uF) smooths battery power caused by motor switching.
- 3.5.2 Capacitors C2 and C3 (0.1uF) smooth the power supply close to the ICs.
- 3.5.3 Capacitors C4 and C5 (0.1uF) reduce electrical motor noise that reaches the PICAXE.
- 3.5.4 Capacitor C6 reduces switch bounce from the microswitches connected to leg 4.

### 3.6 POWER

- 3.6.1 A red LED D1, in series with a 220Ohm resistor is used to indicate the presence of power on the PCB assembly. The LED intensity may change during operation.
- 3.6.1 Power switch SW1 is used to control power to the circuit.

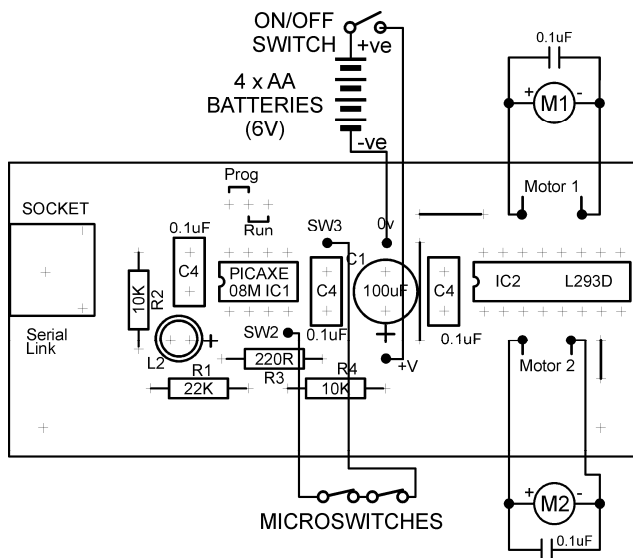
## 4. ASSEMBLING THE PRINTED CIRCUIT BOARD ASSEMBLY



### PCB COMPONENT TERMINOLOGY

- 4.1 Solder the resistors to the PCB. Trim the resistor leads and solder three of these to the PCB as wire links. The wire links are shown as straight lines on the PCB overlay.
- 4.2 Solder the remaining components to the PCB in this order: IC sockets, three-pin header strip, stereo socket, LED, capacitors and electrolytic capacitor. Trim the component leads as required.

CAUTION: Take care in the orientation of IC sockets, LED and electrolytic capacitor.



**PRINTED CIRCUIT BOARD WIRING DIAGRAM**

Unsoldering and replacing damaged or wrongly positioned components will waste time. During soldering, do not overheat the PCB and components.

**WARNING:** The electrolytic capacitor will be damaged and may cause injury if it is installed in the wrong direction and power is applied.

4.3 Do not insert the ICs into their sockets yet. These will be inserted during testing.

**NOTE:** For wiring, use different coloured wires to assist in tracing wires during fault finding. When soldering wires, strip a short piece of insulation from the end of the wire, twist the strands and "tin" them with solder.

## 5 ASSEMBLING THE GEARBOX

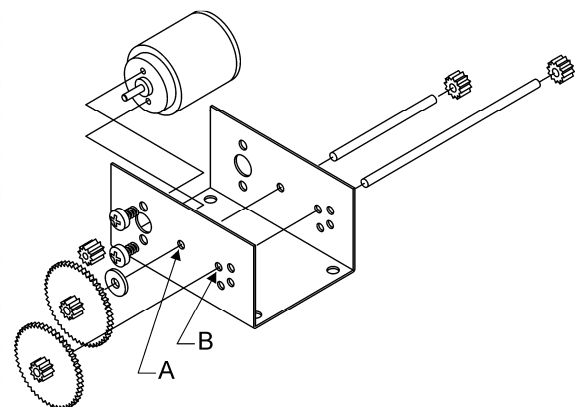
### 5.1 GENERAL

NOTES:

- the white gears, with the 1.9mm holes, are press fit onto the motor.
- the white gears, with the 2.4mm holes are press fit on to the gearbox shafts.
- The yellow spur gears free wheel on the gearbox shaft and have a 2.6mm diameter hole.
- The 12T pinions are used as locators.

5.1.1 Solder a 0.1uF capacitor across each motor's terminals.

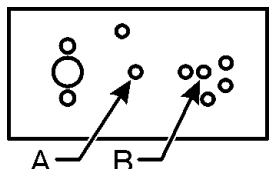
5.1.2 Assemble both gearboxes the same way.



**The MULTI-RATIO GEARBOX Assembly**

The choice of ratios available at the Gearboxes' „Output” shaft are:

<u>GEARBOX STAGE / Reduction ratio</u>	<u>OUTPUT SHAFT</u>	<u>RATIO</u>
Single reduction	Hole A	1:5
Double reduction	Hole B	1:25
Triple reduction	Hole A	1:125



The motor, under load, turns at approximately these speeds.

- with 3 Volts (2xAA batteries) 6,500 RPM
- with 6 Volts (4xAA batteries) 12,600 RPM

### 5.2 ASSEMBLING THE GEARBOX

**HINT:** Before starting assembly, cut (and de-burr) the steel rod to the designed length.

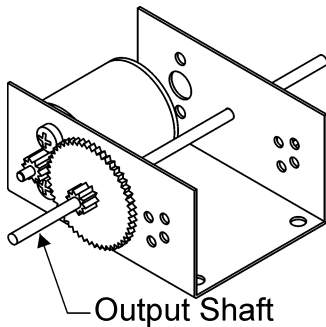
5.2.1 Assemble the steel rod, and all the gears, to the gearcase - as shown in the appropriate drawing. The gears can be assembled onto the shaft/s with a help of small hammer.

5.2.2 Press the 10T pinion onto the motor shaft.

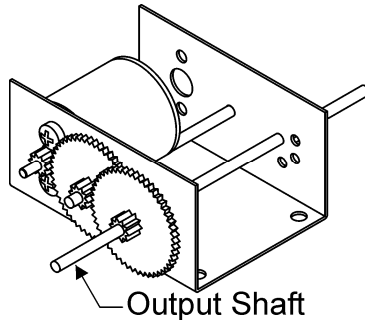
Hint: Place the gear on the bench, insert the motor shaft into the worm gear's hole and gently tap the end of the shaft (where it exits the motor) with a small hammer. Stop when the worm gear is 3mm from the motor's body. **WARNING:** Don't just push the motor down by hand as this can push the motor armature out of its bearings and jam the motor.

5.2.3 Secure the motor to the gearbox case using the two self-tapping screws.

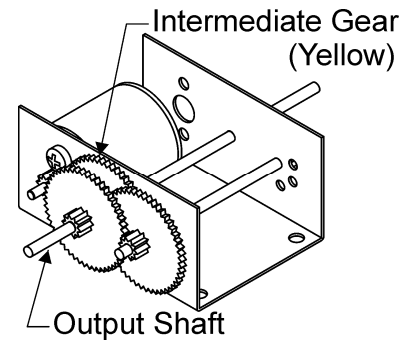
5.2.4 Solder suitable lengths of wire to the motor terminals.



**SINGLE REDUCTION**  
(Low ratio = high speed)



**DOUBLE REDUCTION**



**TRIPLE REDUCTION**  
(High ratio = low speed)

## 6 ASSEMBLING THE *BETTLE*

### 6.1 ASSEMBLY

NOTE: Use a 2.3mm drill bit for pilot holes for self-tapping screw threads and for holes that have a press fit for the 2.5mm steel rod. The 2.6mm drill bit is used for drilling clearance holes for the gearbox shafts, and the 3.5mm drill for M3 screws, which attach the microswitches.

- 6.1.1 Attach the legs and link levers to the side plates. Use M3 Bolts, washers and "Nyloc" nuts (the Nyloc nut is a locking nut, and does not need to be done up tightly). Ensure that all components move freely.
- 6.1.2 Fit the crank to each gearbox. Attach the crank to the corresponding link lever and leg.
- 6.1.3 Position the gearbox assemblies on the platform. Drill holes for self-tapping screws in the platform to match the position of 2 diagonal holes in each gearbox case.
- 6.1.4 Attach both motor and gearbox assemblies to the platform.
- 6.1.5 Mount the PCB assembly to the platform.
- 6.1.6 Form the lever arms of the microswitches so that they form a gentle curve.
- 6.1.7 Attach the microswitches to the platform (using bolts and nuts).
- 6.1.8 Attach the battery holder to the platform.

### 6.2 WIRING

NOTE: Refer to the PCB wiring diagram for more details.

- 6.2.1 Solder the motor wires to the "Motor 1" and "Motor 2" connections on the PCB.
- 6.2.2 Solder a length of wire between the "Normally Closed" terminals (usually marked with "NC" or "2") on both microswitches. Solder a length of wire between the "Common" terminal (usually marked with a "C" or a "1") on one microswitch to "SW2" on the PCB. Solder a length of wire between the "Common" terminal on the other microswitch to "SW3" on the PCB.
- 6.2.3 Solder wires from the battery holder to the power switch. The red wire is positive and the black wire is negative. Solder a length of wire between the switch and the negative power connection (0V) and another length of wire to the positive power connection (V+).
- 6.2.4 Attach the on/off switch to the platform.

## 7. ELECTRICAL TESTING

- 7.1 Inspect soldering for short circuits and poor "wetting" of component leads or pads.
- 7.2 Insert four 1.5 Volt AA batteries into the battery holder. Move the power switch to "on". Check that the LED illuminates (this shows that power is available).
- 7.3 If the LED does not illuminate:
  - Check that the batteries are properly inserted in the battery holder.
  - Check that the battery voltage is above 5.5 volts. (If low, replace the batteries.)
  - Check that battery voltage is present between legs 1 and 8 on IC1. Leg 1 should be positive.
  - Check the wiring against the wiring diagram.
  - Check the values of the resistors against the circuit diagram.
  - Check that the LED is the right way around.
  - Check that the LED is working by using a 220 Ohm resistor and 6Volt (battery) power.
- 7.4 Move the power switch to "off". Check the orientation of the ICs - the end with leg 1 is identified with a notch or dimple at one end. Line up the legs of each IC with its IC socket holes and press down firmly. Do not use the letters/numbers on the IC to identify leg numbers.

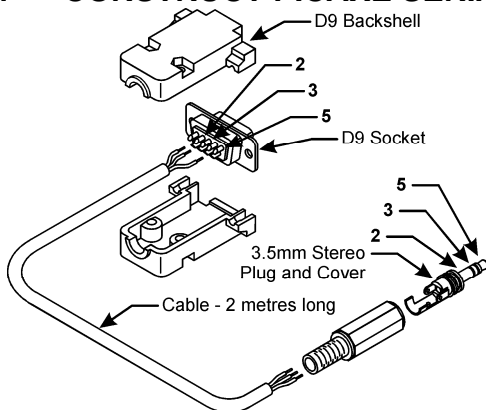
NOTE: It may be necessary to bend the IC legs slightly to line them up with the socket holes.

CAUTION: ICs will be damaged if they are installed in the wrong direction or if power supply (battery) connections are reversed.

- 7.5 Position the jumper on the header strip so that it is in the "run" position. Move the power switch to "on". Check that the LED illuminates (this checks that power is available). If the PICAXE is not programmed, the motors will not turn and will not respond when the microswitches are pressed. Note: At this stage nothing will appear to be working.

## 8. PROGRAMMING THE PICAXE

### 8.1 CONSTRUCT PICAXE SERIAL CABLE (IF REQUIRED)



*PICAXE serial cable construction*

If required, construct a PICAXE serial cable (shown).

### 8.2 INSTALL PICAXE EDITOR

NOTE: PICAXE editor needs to be installed once.

- 8.2.1 Start up and log into your PC. (Some PCs require that you log in as the 'Administrator' to install software.)
- 8.2.2 Download and run the installation file from the software page at [www.rev-ed.co.uk](http://www.rev-ed.co.uk) or from [www.picaxe.co.uk](http://www.picaxe.co.uk).

- 8.2.3 Follow the on-screen instructions to install the PICAXE editor.
- 8.2.4 Insert the PICAXE serial cable into a 9 pin serial port (COM1) on your PC. If your PC does not have a 9 pin serial port, a USB to serial adapter is needed. Take note of the "COM" port number that you are using. (The serial ports on some PCs are labelled A and B rather than numbers – usually A is COM1 and B is COM2).

### 8.3 START PICAXE EDITOR

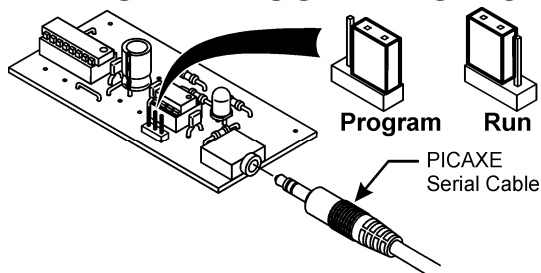
- 8.3.1 Click Start>Programs>Revolution Education>Programming Editor to start the software.
- 8.3.2 If the Options screen does not automatically appear, click the View>Options menu. On the 'Mode' tab select PICAXE-08M mode. On the 'Serial Port' tab select the appropriate serial COM port (usually COM1) then click OK.
- 8.3.3 The PICAXE programming editor software is ready to use.

## 8.4 EDIT PROGRAM

8.4.1 Copy our program into the PICAXE programming editor software. The program and flowchart are at the end of this document.

8.4.2 Save the program. You can keep the original file. For changes use different file names.

## 8.5 TRANSFER PROGRAM TO PICAXE



8.5.1 Move the power switch to "off".

8.5.2 Position the jumper to the "program" position.

8.5.3 Connect the PICAXE serial cable between your PC and the stereo socket on the PCB.

8.5.4 Move the power switch to "on".

### CONNECTING PICAXE SERIAL CABLE

8.5.5 Run the PICAXE Programming Editor software and transfer our program to the PCB. When the program has finished downloading, one (or both) of the motors will begin to turn.

8.5.6 If the Programming Editor software gives an error message stating that program cannot be transferred to the PICAXE:

- Carry out the checks listed in Section 5, Electrical Testing.
- Check that the cable is fully inserted into the stereo socket.
- Check that the cable is inserted into the correct COM port (usually COM1) on your PC.
- Check that the PICAXE Programming Editor software is set to the appropriate COM port.

8.5.7 Move the power switch to "off".

8.5.8 Disconnect the PICAXE serial cable from the stereo socket on the PCB.

8.5.9 Position the jumper to the "run" position.

## 9 FUNCTIONAL TEST

NOTE: The functional test can only be performed after the program is loaded into the PICAXE.

9.1 Move the power switch to "on". Both motors should move in the same direction. To change the direction of a motor, swap its wires.

9.2 Press one of the microswitches. The motors should rotate backwards, turn in opposite directions and continue in the backwards direction. Press the other microswitch. The motors should rotate forwards, turn in opposite directions and continue in the forward direction.

9.3 Do not press the microswitches for a period of time (about 15 seconds). After this time, the motors go through a change of direction sequence.

9.4 Move the power switch to "off". While pressing one of the microswitches, move the power switch to "on". The motors should move backwards and forwards and then stop. Repeat for the other microswitch. (This mode was developed to stop the motors turning before downloading a program.) Move the power switch to "off".

9.5 Move the power switch to "on". Place *BEEBLE* on the floor and observe it bumping into objects and altering its course.

NOTE: The L293D will become warm with continuous use.

## 10 FURTHER DEVELOPMENT

We encourage you to change the program after you have determined that our program works with your *BEEBLE*. You may use the following investigations to customise the way in which your *BEEBLE* interacts with its environment.

NOTE: For programming language details, from the PICAXE "Programming Editor" help menu, open "PICAXE Manual 2 - BASIC Commands".

10.1 A flowchart is a graphical representation of the program. When the program encounters an "end" statement, the program will stop until the power is cycled (turned off, then turned on). What condition needs to be filled for the program to end?

10.2 A number of parameters are setup when the program starts running. One at a time, change the values of "wlng" (wait long), "wstp" (wait stop), "wbrf" (wait brief) and "await" (wait timer) in the following program lines.

```
symbol wlng = 2000 'wait long = 2 seconds
symbol wstp = 500 'wait stop = 0.5 seconds
symbol wbrf = 100 'wait brief = 0.1 seconds
let await = 15000 '15 second timer if switch not pressed
```

As a starting point, double and then halve each parameter. (For example, change the value of "wlng" from 5000 to 10000). What effect does each of these parameters have?

10.3 When moving forwards the following pair of lookup tables after label "skip\_forward:" controls what happens after *BEEBLE* bumps into an object.

```
lookup aloop, (mstp, mrev, mstp, mlft, mstp), apins
lookup aloop, (wstp, wlng, wstp, wlng, wstp), atime
```

When moving backwards, the following pair of lookup tables after label "skip\_reverse:" controls what happens when *BEEBLE* bumps into an object.

```
lookup aloop, (mstp, mofd, mstp, mlft, mstp), apins
lookup aloop, (wstp, wlng, wstp, wlng, wstp), atime
```

The top line of each pair of lookup tables specifies motor directions and the bottom line specifies the time interval. Available parameters are listed in the "symbols" section of the program.

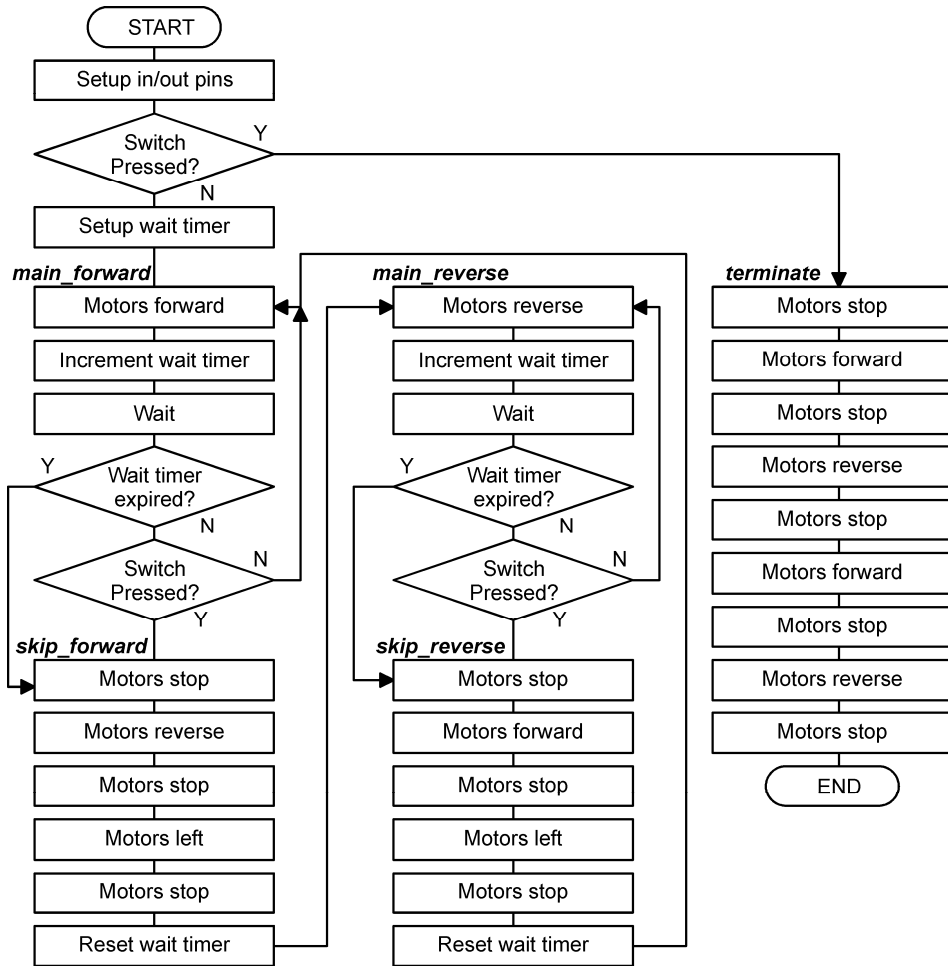
If adding/deleting the number of data entries, in the following program line change the value after "to" by the same amount.

```
for aloop = 0 to 4
```

The loop variable "aloop" controls which values will be loaded into "apins" and "atime". When "aloop" = 0, the first data entry is loaded; when "aloop" = 1, the second data entry is loaded; when "aloop" = 2, the third data entry is loaded; and so on.

Investigate creating new sequences of actions by changing data in the lookup tables. Why should there be a pause before changing the leg direction?

Congratulations, you have successfully built your own unique *BEEBLE !!!* - *HAVE FUN!*



PROGRAM FLOW CHART

## THE PROGRAM

```
'BEETLE
'(C)2007 PVA Tecwrite - Peter Aleksejevs

'PIN SETTINGS
'pin0 = leg 7 = output = Left motor connection
'pin1 = leg 6 = output = Left motor connection
'pin2 = leg 5 = output = Right motor connection
'pin3 = leg 4 = input = Front and rear switch (1 = pressed)
'pin4 = leg 3 = output = Right motor connection

'Symbols
symbol mfwd = %00010010 'motor forward
symbol mlft = %00010001 'motor left
symbol mrev = %00000101 'motor reverse
symbol mrgt = %00000110 'motor right
symbol mstp = %00000000 'motor stop
symbol wlng = 5000 'wait long = 5 seconds
symbol wstp = 500 'wait stop = 0.5 seconds
symbol wbrf = 100 'wait brief = 0.1 seconds
symbol aloop = b0 'loop counter variable
symbol apins = b1 'which pins are to be output
symbol atime = w1 'time interval for pins to be output(in milliseconds)
symbol acntr = w2 'change direction if switch not pressed for a while
symbol await = w3 'wait timer

'Initial setup
    let dirs = %00010111
    if pin3 = 1 then terminate      'Switch pressed when power turned on.
    let await = 15000              '15 second timer if switch not pressed
    let await = await / wbrf      'Convert into loop cycles

main_forward:
    let pins = mfwd
    let acntr = acntr + 1
    pause wbrf
    if acntr >= await then skip_forward
    if pin3 = 0 then main_forward 'check for switch

skip_forward:
    for aloop = 0 to 4
        lookup aloop, (mstp,mrev,mstp,mlft,mstp), apins
        lookup aloop, (wstp,wlng,wstp,wlng,wstp), atime
        let pins = apins
        pause atime
    next aloop
    let pins = mrev
    pause wlng
    let acntr = 0

main_reverse:
    let pins = mrev
    let acntr = acntr + 1
    pause wbrf
    if acntr >= await then skip_reverse
    if pin3 = 0 then main_reverse 'check for switch

skip_reverse:
    for aloop = 0 to 4
        lookup aloop, (mstp,mfwd,mstp,mlft,mstp), apins
        lookup aloop, (wstp,wlng,wstp,wlng,wstp), atime
        let pins = apins
        pause atime
    next aloop
    let pins = mfwd
    pause wlng
    let acntr = 0
    goto main_forward

terminate:
    for aloop = 0 to 8
        lookup aloop, (mstp,mfwd,mstp,mrev,mstp,mfwd,mstp,mrev,mstp), apins
        let pins = apins
        pause wbrf
    next
end
```

## PROGRAM LISTING