

# BUGGY (TECHNOLOGICAL STUDIES COURSES)

---

MOD001 Buggy (Technological Studies Courses)

## Contents:

- CHI007 Buggy Kit
- Programming Editor software on CDROM
- PICAXE Download Cable

*Also required* - computer running Windows 95/98/2000/ME/NT

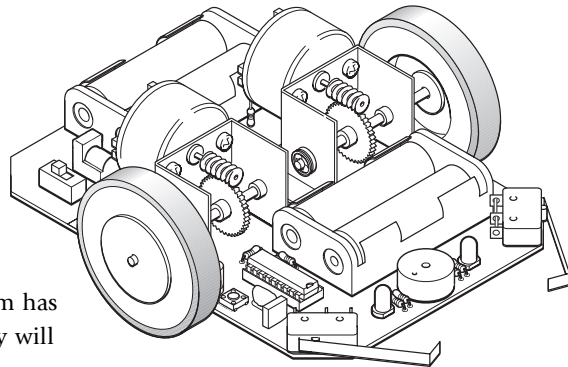
The buggy is not compatible with the Acorn software.

## Description:

The MOD001 buggy is designed for schools teaching Standard Grade Technological Studies.

The buggy is different to all other models, in that it is battery powered (4xAA cells) and has the microcontroller 'on-board'. This makes it totally 'remote' - once the program has been downloaded the cable can be removed and the buggy will work without any connection to the computer or external power supply.

The buggy is pre-fitted with a PICAXE-18 microcontroller. This mimics the Stamp Controller behaviour so that students can write BASIC programs and download them directly into the buggy, without having to wire the buggy to the Stamp Controller by an 'umbilical cord'. Therefore the Stamp Controller is **not used** when programming the buggy.



## Instructions:

1. Assemble the CHI007 buggy as per instructions in the kit.
2. Install the Programming Editor software from the CDROM.
3. Run the Programming Editor software. Make sure the software is in PICAXE-18 mode (from the View-Options menu).
4. Connect the cable to the serial port of the computer. Make sure the correct serial port (COM1 to COM4) is selected within the software (from the View>Options menu).
5. Type in a BASIC program (example overleaf)
6. Make sure the buggy is fitted with the PICAXE-18 microcontroller. Connect the cable to the buggy, and make sure the buggy is switched on (note that it may be necessary to prop up the buggy to stop it moving if an existing program is already in memory).
7. Click Run from the PICAXE menu to start the program download.

FOR FURTHER NOTES ON HOW TO USE THE PICAXE-18 BASIC PROGRAMMING LANGUAGE SEE THE MANUALS PROVIDED ON THE CDROM.

**Important Notes:**

- Ensure you have installed the **latest version** of the Programming Editor from the CDROM enclosed. The system will not work with older versions of the software.
- When using the software the 'PICAXE-18' microcontroller mode (View menu > Options) **must be selected** when programming the buggy. The BASIC language is identical to the Stamp Controller BASIC.
- Follow the assembly instructions for the buggy given in the leaflet. However use LK3 for the wire link for the piezo sounder.
- The motor connections on the buggy are identical to the push-pull driver motor connections on the Output Driver board.
- DO NOT reprogram the PICAXE-18 microcontroller in a PIC programmer. This will render it un-useable within the buggy.
- The **buggy download cable** can be safely connected to the end of the Stamp Controller download cable, so students **do not** have to swap cables at the back of the computer.
- If desired the buggy can be used with a PIC16F84A in Advanced Higher courses. However in this case a 3-pin 4MHz ceramic resonator must be soldered in position X1. This does not affect normal operation with the PICAXE-18 microcontroller in the socket.

**Buggy Connections:**

output pin allocation

(using binary notation %76543210)

76 = motor B (same as Output Driver module)

54 = motor A (same as Output Driver module)

3 = not used

2 = LED B

1 = LED A

0 = piezo sounder (make sure LK3 is used on buggy PCB)

input pin allocation

pin1 = micro-switch sensor B

pin0 = micro-switch sensor A

## Sample BASIC Programs:

The first sample program uses the binary notation "let pins =" for controlling outputs. If desired these commands can be replaced by multiple "high" and "low" commands as in example 2.

### Example 1

```
`start going forwards
```

```
`testing switches as you go
```

```
main:
```

```
    let pins = %01010000  
    if pin0 = 1 then left  
    if pin1 = 1 then right  
    goto main
```

```
`left switch hit
```

```
`so stop, light LED, beep, reverse, turn
```

```
left:
```

```
    let pins =%00000100  
    sound 0, (100,150)  
    let pins =%10100100  
    pause 2000  
    let pins =%10010100  
    pause 1500  
    goto main
```

```
`right switch hit
```

```
`so stop, light LED, beep, reverse, turn other way
```

```
right:
```

```
    let pins =%00000010  
    sound 0, (50,150)  
    let pins =%10100010  
    pause 2000  
    let pins =%01100010  
    pause 1500  
    goto main
```

**Example 2**

```
`start going forwards
```

```
`testing switches as you go
```

```
main:
```

```
    low 7          ` motor forwards  
    high 6  
    low 5  
    high 4  
    low 2          ` leds off  
    low 1  
    if pin0 = 1 then left  
    if pin1 = 1 then right  
    goto main
```

```
`left switch hit
```

```
`so stop, light LED, beep, reverse, turn
```

```
left:
```

```
    low 6  
    low 4  
    high 2  
    sound 0, (100,150)  
    high 5  
    high 7  
    pause 2000  
    low 5  
    high 4  
    pause 1500  
    goto main
```

```
`right switch hit
```

```
`so stop, light LED, beep, reverse, turn other way
```

```
right:
```

```
    low 6  
    low 4  
    high 1  
    sound 0, (50,150)  
    high 5  
    high 7  
    pause 2000  
    low 7  
    high 6  
    pause 1500  
    goto main
```