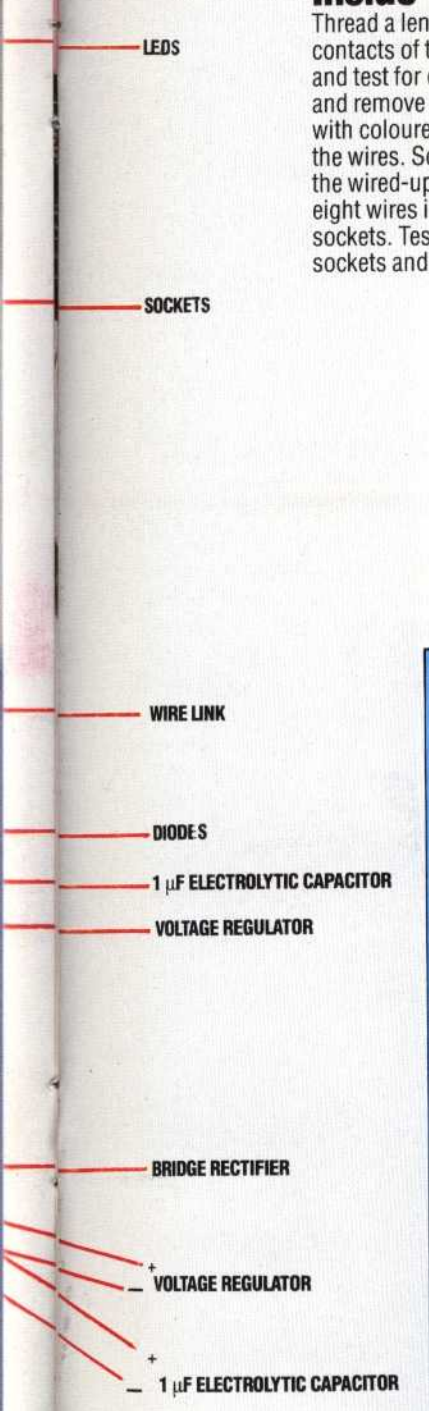

Inside The Box

Thread a length of tinned wire as shown through the contacts of the sockets. Solder it to each contact, and test for continuity. Take 20 cm of ribbon cable and remove three wires, leaving a nine-way ribbon with coloured edge stripe. Bare and tin the ends of the wires. Solder the coloured wire to the leftmost of the wired-up sockets. Now solder the remaining eight wires in order to the contacts of the other sockets. Test for continuity between each of the sockets and the end of the wire connected to it

Building The Circuit Board

The board shown has 30 tracks with 45 holes and fits our box exactly. Follow the layout illustrations carefully, and you should have no trouble building the board. Use as little solder as possible, and take care not to bridge the tracks; check continually that you are placing the right components in the right places. The diodes, electrolytic capacitor, bridge rectifier and voltage regulator must all be connected in the direction shown — any other orientation will damage them, so study their plus/minus markings. All components are heat-sensitive, so don't 'overcook' them with the iron. When fitting the minicon and power sockets take care to locate the pins in the right board holes, but don't bend the pins roughly. Use the wires you removed from the ribbon cable for the 'jump leads'.

When everything is in place on the board, cut the copper tracks exactly as shown. You can buy a special tool for this, or you can hold a drill bit in your fingers and twirl it in a hole, cutting the copper gradually. Don't leave shreds of copper on the board. Solder the ribbon cables onto the board. The orientation of the socket and LED leads is shown by the coloured stripe, but the user port lead needs some thought — the two earth lines must be in holes 1 and 10 (counting from the edge of the board), and the signal lines must go in order into holes 2 to 9 so that the least significant line is closest to the edge of the board.

Lastly, using the board as a template, cut slots in the box sides to accommodate the sockets and the user port lead

```
 10 REM * BBC REACTION TIME **
 15 :
 20 DDR=&FE62: DATREG=&FE60
 30 ?DDR=127: REM LINES 0-6 OUTPUT
 40 ?DATREG=127: REM LEDS OFF
 50 :
 60 CLS: PRINT "GET READY"
 70 DELAY=3000+RND(9000)
 80 FOR I=1 TO DELAY:NEXT:REM DELAY LOOP
 85 FOR D=1 TO 200:NEXT D
 90:
 97 REPEAT UNTIL ?DATREG AND 128=1
100 ?FE60=0: REM TURN LEDS ON
110 TIME =0: REM INIT TIMER
120 REPEAT
130 UNTIL ?DATREG AND 128=0: REM S/W ON
140 :
150 PRINT "TIME TAKEN="TIME/100"SECS"
160 END
```

The program makes use of TIME, a reserved variable that returns a value corresponding to the number of hundredths of a second since TIME was last set to zero. Making use of the logical AND in line 130 isolates bit 7 (value 128) so that it can be tested independently of the other bits in the data register (see page 66). When the switch is thrown, the value of a bit 7 changes from one to zero.

A similar program can be written for the Commodore 64 using its internal timer, TI. TI works differently from TIME, returning a value in sixtieths of a second since the machine was turned on. To use it we must take the value of TI at the start of the interval to be timed and subtract it from the value of TI at the end.

```
10 REM CBM 64 ** REACTION TIMER **
20 :
30 DDR=56579: DATREG=56577
40 POKE DDR,127: REM LINES 0-6 OUTPUT
50 POKE DATREG,127: REM LEDS OFF
60 :
70 PRINT CHR$(147): REM CLEAR SCREEN
80 PRINT "GET READY"
90 DE=3000+INT(9000*RND(1))
100 FOR N=1 TO DE:NEXT: REM DELAY LOOP
110 :
120 POKE DATREG,0: REM TURN LEDS ON
130 T=TI: REM TAKE START TIME
140 IF PEEK(DATREG) AND 128<>0 THEN 140
150:
160 TM=(TI-T)/60: REM CALC INTERVAL
170 PRINT "TIME TAKEN=";TM;"SECS"
180 END
```

In future instalments of the course, we will look at the construction of higher current outputs that are sufficient to drive electric motors, and we will design software to control bi-directional and variable speed motors.

Experiments For You To Try

1) Write a program that will light one LED at a time in sequence from left to right.
2) Write a program to make the LEDs light in sequence on lines 0 to 6 (as in question 1), but include a switch on line 7 to change the sequence direction. Can you alter your program to operate with a 'train' of three LEDs?
3) Write a program to simulate a dice throw, using six LEDs and one switch.
4) Write a program to simulate the action of a traffic light, using three LEDs.
5) Write a program to count the number of 'cars' (pulses on a switch) arriving while a traffic light is on red and to change the lights when the number of cars exceeds 10 or if one minute has elapsed since the last change.