Sensing & Control Projects for the BBC

Thomas Nunns

Micro books

Sensing & Control Projects for the BBC

Thomas Nunns

First Published 1984

MICROBOOKS
443 Millbrook Road
Southampton
SO1 0HX

ISBN 0-946705-03-8

Printed by Litho Techniques (Kenley) Ltd
46-50 Godstone Road, Whyteleafe, Surrey

Sole U.K. Distributors:
John Wiley & Sons Ltd, Baffins Lane,
Chichester, Sussex.

CONTENTS

INTRODUCTION

This book is for anybody who has a BBC micro and has looked at the row of connectors underneath it and at the sockets on the back panel and has wondered what they are for.

It is for the person who likes experimenting but doesn't like soldering irons and who knows nothing about electronics.

It is for someone who wants to find out more about the computer's potential and is willing to spent a little time experimenting.

If you have never used the computer as a link to the outside world then you will not have realised how versatile the BBC micro can be.

You need no experience of electronics - you don't even need a soldering iron. But that doesn't mean the projects are trivial - they have been designed around components which can be fitted together without soldering.

By the time you reach the end of the book you will have fed signals into the BBC to tell it all about its environment, and sent signals out from the BBC to control lights, toy cars - even robots.

The book is in two sections - input using the Analogue to Digital converter and output from the User Port - and each section is complete in itself. However, the projects within each section do follow on and it is not recommended that you jump about between them.

Kits containing all the parts needed to complete the projects are available - details can be found at the back of the book. Most of the programs are quite short and can be typed into the computer in a few minutes but a disc or cassette tape containing all the programs is also available if required.

THE ANALOGUE TO DIGITAL CONVERTER INTERFACE

At the back of the BBC micro there is a socket labelled **analogue in** and all the projects in the first section of the book will use this port (*port* is the computer term for socket).

Unfortunately the plug which connects to this port is rather expensive. To avoid having to connect a new plug to every project we will start by building an interface board (*interface* is the computing word for connection).

This interface board will be used with each project so you must start by building it.

You will need -

a 15 way D plug - IDC type (see below)
1 metre of 15 way ribbon cable
an eight way connector (terminal) block
some PVC tape
a block of wood about 10cm x 7cm x 2cm

The plug is a fifteen way *D-type* connector. One sort of D-type plug has to have its connections soldered to fifteen tiny pins. Although these plugs are slightly cheaper than the type we will use, they are very difficult to connect and unless you are reasonably capable at using a soldering iron they are not recommended.

The type of plug used in this interface is an *Insulation Displacement Connector* (IDC). This plug is squashed onto a piece of ribbon cable causing the pins to pierce the insulation and make a contact. It is very much easier to fit than the solder type and gives a much stronger connection.

The cable which connects the plug to the board is a piece of 15 way ribbon cable, but 15 way is not a standard size so you may have to cut down a piece of 16 or 20 way ribbon cable. The cable is held against the teeth at the back of the plug and the backplate is pushed down onto the ribbon cable to make the teeth push through the insulation and make contact with the wires inside. You will need to use a vice to squeeze the parts of the plug together with

the ribbon cable in between (figure 1). There are many different designs of plug but most have an extra piece which fits over the ribbon cable when it is folded back over the plug to hold the cable and make the connection even stronger (figure 2).

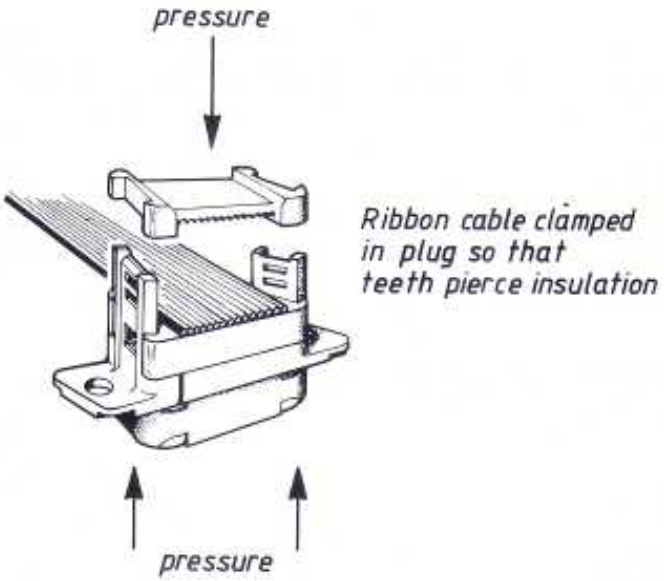


Figure 1

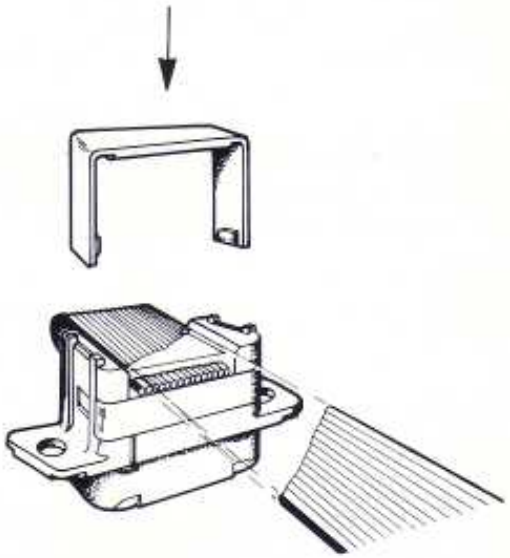


Figure 2

We are only going to use two channels and one switch on the analogue port (if you want to use all four channels and both switches a diagram of the connections is given at the end of the book). As well as the two channels and the switch, one of the wires will be connected to the reference voltage while the other will be connected to ground (earth).

Carefully take a knife and split the other end of the ribbon cable making sure you do not cut through the insulation and expose the wires inside.

The pins in the plug are numbered from 1 to 15. The numbers will be very small and will probably be on the pin side. Find out which side of the ribbon cable is connected to pin 1 and lay the cable flat with the wire connected to pin 1 on the left (figure 3). You will see that the wires are not numbered from 1 to 15 because of the way that the pins in the plug are arranged. It is important that you refer to the diagram to make sure you are connecting the correct wire.

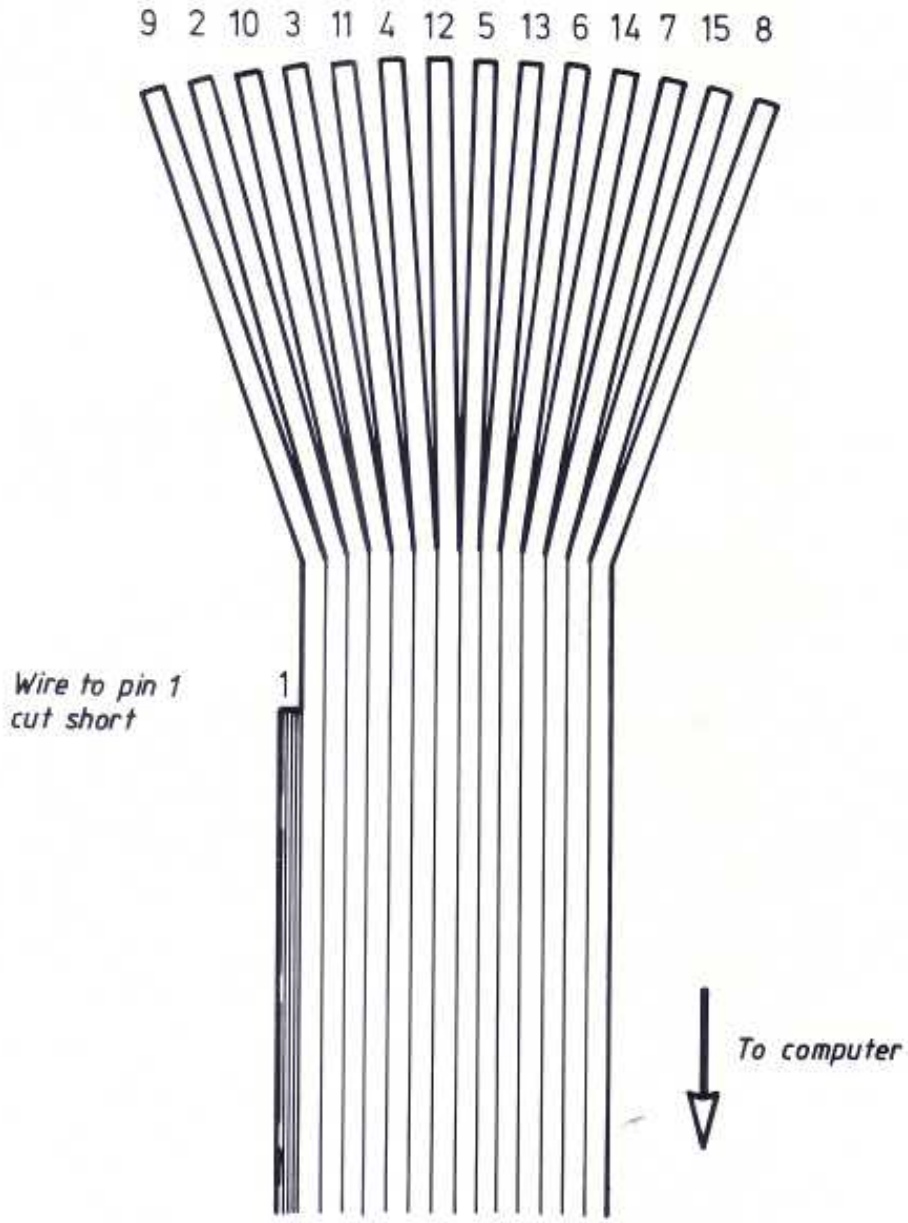
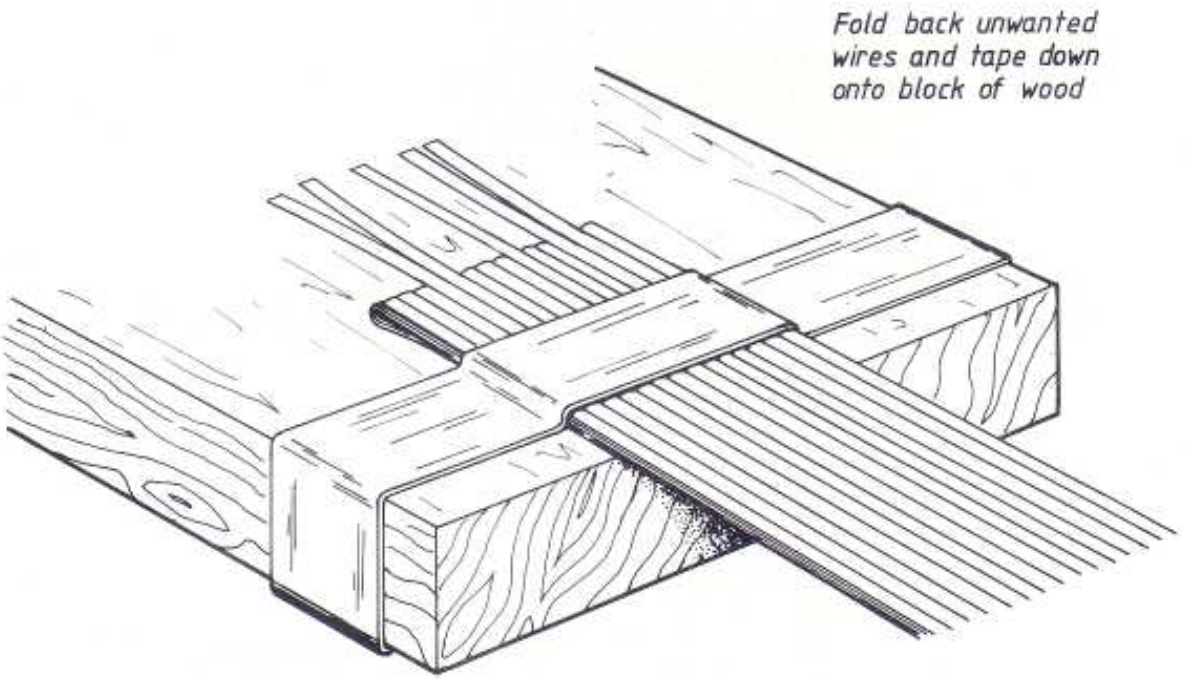


Figure 3

The one wire which *MUST* be avoided is number 1 which carries 5 volts. It will become clear later why we need to avoid this wire, but for the moment peel back the wire about 6cm and cut it off. Stick a piece of plastic tape over the cut end to make sure that it cannot touch any of the other wires.

Connections are made to wires 3, 7, 11, 13 and 15. The other wires are not needed for these projects but do not cut them right off as you may wish to enlarge your interface later. They can be folded back over the ribbon cable and taped down (figure 4). The wires are fixed into an eight way connector block strip as shown in figure 5. Extra lengths of wire will be needed to join connectors 2, 3 and 6 and connectors 5 and 8. If the connector block is screwed onto a small piece of wood the ribbon cable can be bound onto the wood using plastic tape and this will ensure that the wires cannot be pulled away accidentally.

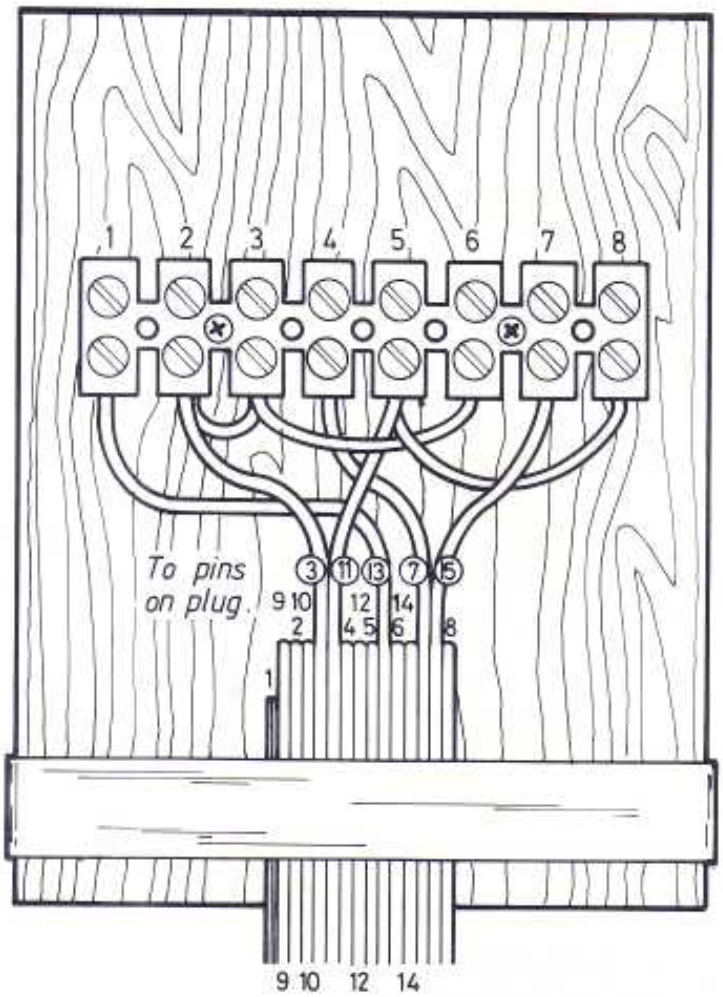
Figure 4



When stripping the ends of the wire remove about 5mm of the insulation and fold the exposed strands back over the plastic. This will allow the connector block screws to hold the plastic as well as the wire which will give much more strength. With ribbon cable you will be able to strip the insulation from the wire using your finger nails. This is the best method to use as it ensures that none of the wires inside are broken.

Figure 5 shows the pin connections and the position of each wire in the connector block with the extra links. The extra links will make it easy to connect different components to each channel without having to put more than one wire into a connector.

Figure 5



Wire 1 cut off - other unused wires folded under

Number the connectors as shown. The projects will refer to these numbers when you have to make a connection to the interface.

The actual computer connection for each part of the connector block is as follows -

connector 1 - switch
connector 2 - ground
connector 3 - ground
connector 4 - channel 2
connector 5 - reference voltage
connector 6 - ground
connector 7 - channel 1
connector 8 - reference voltage

Once the interface is ready you can go on to try the first project which also tests the unit to make sure that all your connections work.

As long as you haven't made a connection to pin 1 and you don't start connecting the unit to a battery you won't damage the computer if you have made any mistakes, but you should get into the habit of checking your work very carefully before plugging it in.

Project 1

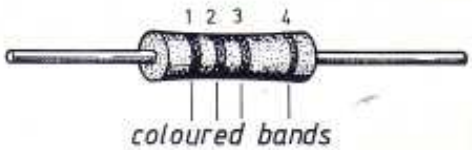
TESTING THE INTERFACE USING A LIGHT DEPENDENT RESISTOR

In this first project the Light Dependent Resistor will be used to measure light levels and the results will be displayed on the computer screen. At the same time the interface will be tested to make sure it is working correctly.

THE EQUIPMENT

You will need -

the A to D interface you have made
a Light Dependent Resistor (LDR) type ORP12
a two-core lead with a pair of connectors at one end
a 1K (1000 ohm) resistor coloured brown, black, red (figure 6)



Band 1= initial digit for value
Band 2= second digit for value
Band 3= number of noughts to add to the two digits
Band 4= tolerance (can be ignored for these projects)

eg. Band 1 - yellow = 4
Band 2 - violet = 7
Band 3 - orange = 3
= 47 plus three noughts
= 47,000 ohms
= 47k ohms

Colour values

Black	= 0
Brown	= 1
Red	= 2
Orange	= 3
Yellow	= 4
Green	= 5
Blue	= 6
Violet	= 7
Grey	= 8
White	= 9

Figure 6

The *Light Dependent Resistor* is joined to the lead using the pair of connectors (figure 7). Connect the other ends of the lead to the connector block on the interface - one half of the lead goes to connector 6 and the other half to connector 7. The *resistor* must be fitted between connectors 7 and 8 (figure 8).

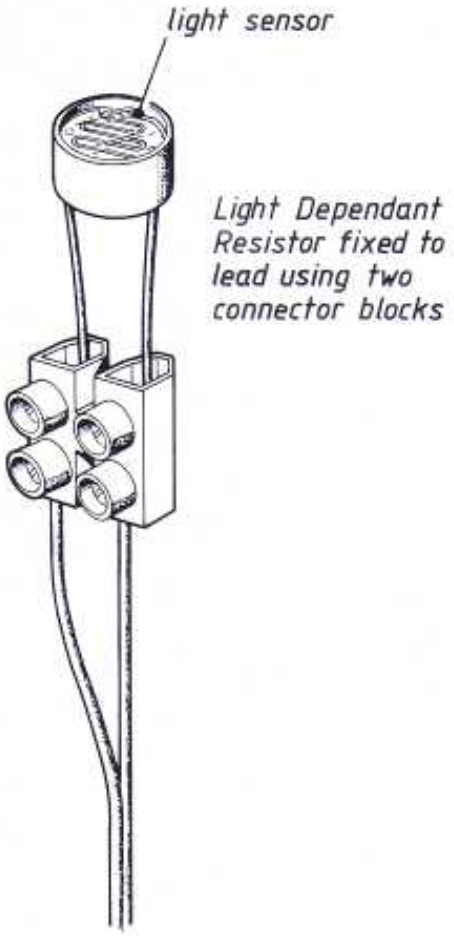


Figure 7

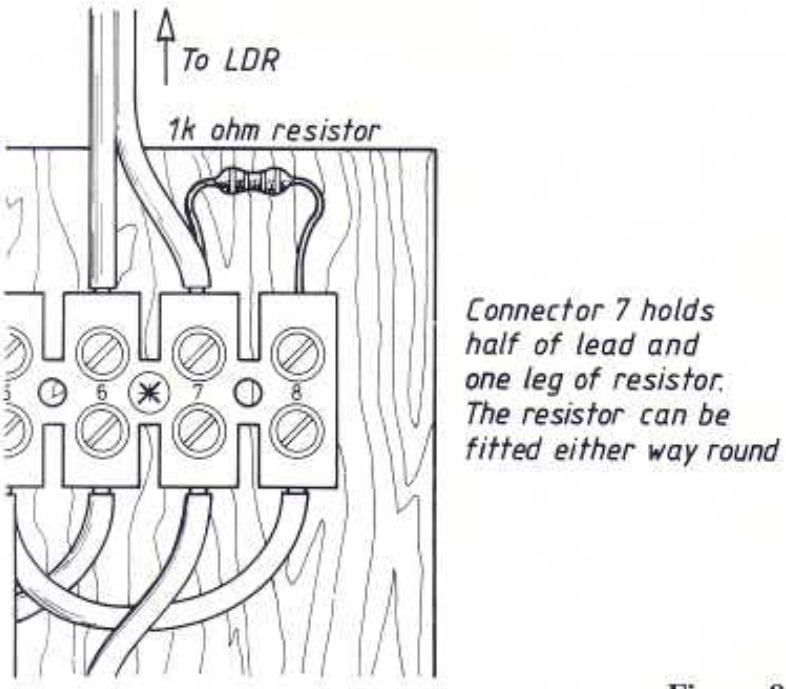


Figure 8

A small electrical signal (less than 1.8 volts) will flow through the Light Dependent Resistor and will be *read* by the computer's Analogue to Digital converter. The LDR resists the flow of electricity. How much resistance it produces depends on the amount of light falling on the face of the LDR. The more light that falls on the LDR the less its resistance will be and, as a result, the *reading* by the computer will change.

The computer's Analaogue to Digital converter will produce a number between 0 and 65,535 depending on the voltage it is reading. A signal of 1.8 volts will produce a reading of 65,535 and a signal of 0 volts will produce a reading of 0. If the signal is greater than 1.8 volts it is possible that the A to D chip in the computer will be destroyed. Luckily the computer provides a signal of 1.8 volts (the voltage reference signal from pin 11) and provided this is always used as a power source no harm can come to the computer. That is why the 5 volt line was cut off when you were building the interface - if this voltage was sent back to the computer serious damage could result.

THE PROGRAM

Program 1 displays the level of light reaching the LDR as a moving line on the screen.

The first two lines set the mode and switch off the cursor. The rest of the program loops continuously until **BREAK** or **ESCAPE** is pushed. The value of the reading is found by the expression **ADVAL(1)** where the 1 tells the computer which channel to read. To make the readings more accurate and more manageable they are divided by 64 by the expression **DIV 64**.

The actual reading in the computer falls as the light intensifies - the opposite of the result one would expect. The result is therefore taken away from 1024 to reverse the reading and make the line on the screen rise as more light reaches the LDR.

Program 1:

```
 10 MODE1
 20 VDU23;8202;0;0;0;
 30
 40 REPEAT
 50 CLS
 60 PRINT TAB(1,1);"Level="
 70 PRINT TAB(18,1);"LIGHT"
 80 PRINT TAB(18,31);"DARK"
 90 MOVE 0,1024-ADVAL(1) DIV 64
100
110 FOR count%=4 TO 1280 STEP4
120 level%=1024-ADVAL(1) DIV64
130 DRAW count%,level%
140 PRINT TAB(7,1);level%;" ";
150 NEXT
160
170 UNTIL FALSE
```

USING THE PROJECT

Plug the interface into the **analogue in** socket at the back of the computer and run the program.

The computer screen will continuously draw a line showing the light the LDR is receiving. In the top left hand corner a figure will show the value being read by the computer divided by 64.

If a hand is placed over the LDR the line should fall to the bottom of the screen and if a bright light is shone on the LDR the line will rise. The value will never quite reach the maximum of 1024 because of the 1K resistor but this helps to stabilise the results.

To leave the program push **ESCAPE**. (Don't forget to save the program on tape or disc before pushing **BREAK**.)

WHAT TO DO IF IT DOESN'T WORK

Remove the plug from the **analogue in** socket and run the program. If the program is running properly the words *Level*, *LIGHT* and *DARK* should be printed on the screen and a wobbly line should be drawn continously across the screen. If the program isn't working you must correct it before carrying on as it is impossible to test the interface without some suitable software in the computer. If there is a mistake in the program correct it then plug the interface back in and try again.

Check the wiring of the interface very carefully against the final diagram in the previous chapter (figure 5). Be particularly careful to make sure that you have identified pin 1 on the plug and that it is not connected to the interface, and that you have counted the strands of the ribbon cable correctly.

Save Program 1, remove the lead and the resistor from the interface, push **BREAK**, type in the following three lines of program and run them. Make sure that the interface is plugged in at the back of the computer.

```
10 REPEAT
20 PRINT ADVAL(1),ADVAL(2)
30 UNTIL FALSE
```

The computer should print two numbers side by side continuously. Using a bent paper clip connect together connectors 6 and 7. The left hand number should fall below 100. Now join together connectors 7 and 8 - the left hand number should rise to about 65,000. Try the same with connectors 3 and 4, and then 4 and 5. The right hand number should show the same results. If your results are different you will have connected the ribbon cable to the connector block incorrectly. Refer back to the diagram to trace your mistake.

FURTHER TESTS FOR THE INTERFACE

You should try these tests even if the first project worked correctly first time. They will check the rest of the interface to make sure it is working correctly.

Change the program lines 90 and 120 so that the figure in brackets after **ADVAL** is 2 instead of 1. Remove the lead with the LDR and the resistor and replace them in connectors 3, 4 and 5 so that the resistor is between connectors 4 and 5 and the lead is attached to connectors 3 and 4. Run the new program. The result should be exactly the same as before but you are now using channel 2 instead of channel 1.

Finally, to test the switch, type in the follow three lines and run them -

```
10 REPEAT
20 PRINT ADVAL(0) AND 1
30 UNTIL FALSE
```

When run a line of noughts should appear vertically on the screen. Connect together connectors 1 and 2 with a piece of wire - the noughts should change to ones. This test shows that the switch is working correctly.

If any of the tests fail you should refer back to the diagram in the last chapter to find where you have made a mistake. You cannot experiment with other sensors until you are sure that the interface is working correctly.

AN IDEA TO TRY

The LDR is a **sensor** - it senses how much light it can *see*. Various colours reflect light better than others. Place the LDR on a flat surface about 2cm away from a vertical surface - a propped up book is quite suitable (figure 9). Make sure that you are not casting a shadow on the LDR.

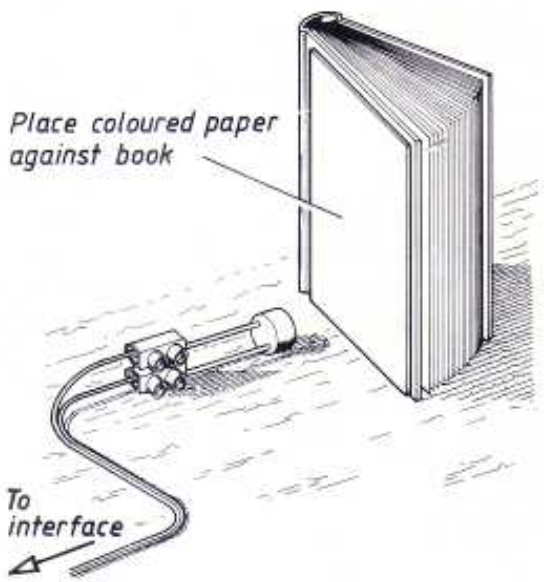


Figure 9

Try placing different coloured pieces of paper against the book. The light the LDR receives will be reflected by the paper. Using this test you should be able to find out which colours reflect the most light and which are the least reflective.

Project 2

USING THE LDR TO MEASURE TIME

In project 1 the Light Dependent Resistor was used to measure light levels. In this project two Light Dependent Resistors will be used but they will only be sensing changes in light levels. Using the computer's TIME function the LDRs can measure accurately quite small time intervals.

THE EQUIPMENT

You will need -

the A to D interface
two Light Dependent Resistors
two two-core leads with connectors at one end
two 1K (1000 ohm) resistors coloured brown, black, red
a 'push-to-make' switch (if necessary this can be made using a paper clip).

Connect the LDRs to the leads using the connectors in the same way as you did in Project 1. Connect the other ends of the leads to the connector block - one lead goes to connectors 6 and 7 and the other lead goes to connectors 3 and 4. The resistors must be fitted between connectors 7 and 8 and between connectors 4 and 5 (figure 10).

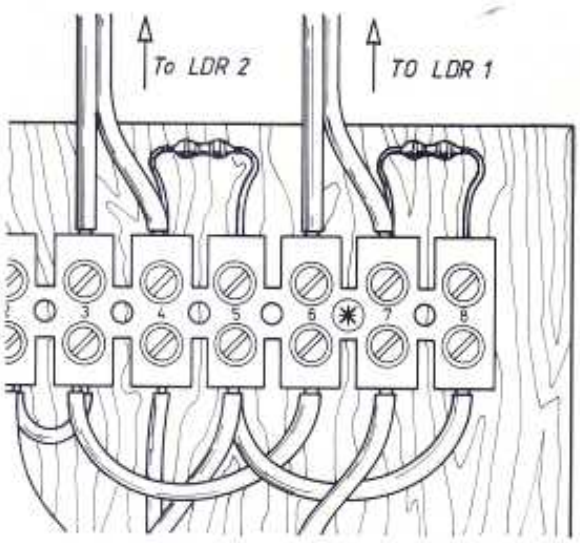


Figure 10

You will also need a *push-to-make* switch between connectors 1 and 2. The pins on the back of the switch should fit into the two connectors so no extra wiring is necessary. This switch can be made using a paper clip - one end is fixed into connector 1 and when the switch has to be *made* the other end is touched against connector 2 and then released.

THE PROGRAM

The program uses the **ADVAL(Channel)** function to collect data but now two channels are being used so **ADVAL(1)** will collect data from one channel and **ADVAL(2)** from the other. The program also uses **ADVAL(0) AND 1** which makes the computer check to see if the switch is made. The command in line 30 - ***FX16,2** - tells the computer that only two channels are being used, this speeds up reaction time.

The program prints *Waiting for set signal ...* and then does nothing until the switch is made. When the computer senses that the switch has been made it remembers the light level at each sensor and prints *Ready ...* on the screen. As soon as the light level changes at LDR 1 the computer starts its clock and prints *Clock running ...* on the screen. The clock remains running until the light level at LDR 2 changes. The clock is then stopped and the time printed on the screen.

The program is only sensing changes in light so the clock can be started and stopped by increasing or decreasing the light. The variable **sense%** sets the amount of change needed to trigger the clock - if **sense%** is a larger number a greater change in the light level will be needed to start and stop the clock.

Program 2:

```
 10 MODE7
 20 sense%=10
 30 *FX16,2
 40
 50 REPEAT:CLS
 60 PRINT TAB(0,3)"Waiting for set signal ... ";
 70
 80 REPEAT
 90 switch%=ADVAL(0) AND 1
100 UNTIL switch%>0
110 set1%=ADVAL(1) DIV 64
120 set2%=ADVAL(2) DIV 64
130 PRINT TAB(0,6)"Ready ... ";
140
150 REPEAT
160 UNTIL ABS(set1%-ADVAL(1) DIV 64)>sense%
170 TIME=0
180 PRINT TAB(0,9)"Clock running ... ";
```

```
190
200 REPEAT
210 UNTIL ABS(set2%-ADVAL(2) DIV 64)>sense%
220 PRINT TAB(0,15)"Time = ";TIME/100;" seconds"
230
240 PRINT TAB(0,22)"Push the space bar to continue ... ";
250 *FX15,1
260 REPEAT:G=GET:UNTILG=32
270
280 UNTIL FALSE
```

USING THE PROJECT

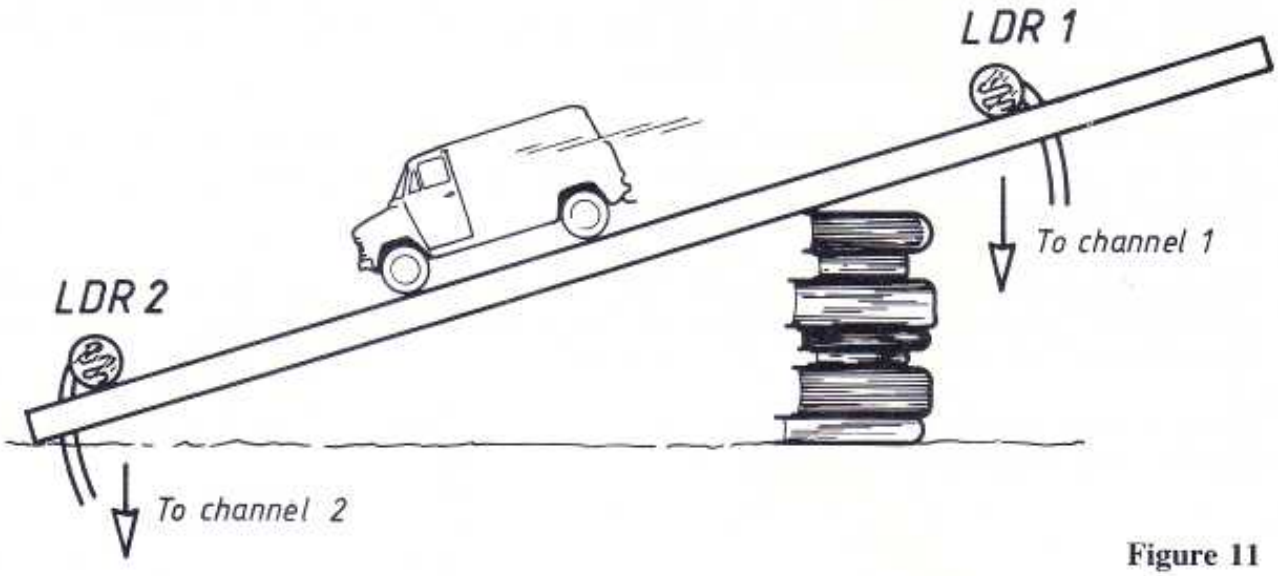


Figure 11

Build a slope about one metre long as shown in figure 11. Fix the LDRs as shown with the LDR which is connected to channel 1 at the top of the slope. When all is ready run the program. The computer screen will show *Waiting for set signal ....*

Join connectors 1 and 2 for a moment with the switch. The computer will remember the light levels at each sensor and will print *Ready ...* on the screen. Run a small car or LEGO vehicle down the slope so that it passes close to the face of both sensors.

As the vehicle passes the first sensor the computer will print *Clock running ...* on the screen. As it passes the second sensor the computer will print the time the vehicle has taken to run down the slope.

To repeat the program push the space bar. To leave the program push **ESCAPE**. (Don't forget to save the program on tape or disc before pushing **BREAK**.)

WHAT TO DO IF IT DOESN'T WORK

If the clock starts or stops running before the vehicle has passed the sensors the it means the light level has changed. Are people moving in front of a window? Has a light been turned on or off? Has a hand been waved in front of a sensor by mistake?

If the sensors continue to trigger the clock try making the variable **sense%** in the program larger.

If the sensors do not start or stop the clock it may be that the vehicle is not passing close enough to them to register a suitable change in the light level. Try using a track to ensure that the vehicle runs straight and true and close to the LDRs. If the system still does not work try making the variable **sense%** smaller.

To make the system almost foolproof two torches can be positioned so that they are shining across the slope on to the LDRs. The light level will then be much higher and the change much more definite when the vehicle passes.

If the program doesn't wait until the *set* signal is sent make sure that the switch has not permanently joined connectors 7 and 8. If a paper clip is being used it should only join the two connectors for a moment and then be removed.

If the computer prints an error message on the screen you have probably made a typing mistake when entering the program.

SOME IDEAS TO TRY

Try increasing the angle of the slope and seeing if the time decreases because the vehicle has travelled faster.

Make a LEGO sledge and see how much slower it is than a vehicle on wheels.

Try covering the slope with different materials - a towel, sandpaper, cellophane, cotton sheeting. Each material produces friction which will change the speed of the vehicle. Try

the sledge - does it get stuck on the sandpaper?

Can the speed of the vehicle be worked out if the sensors are exactly one metre apart and the computer has recorded the time?

SOME OTHER USES FOR THE PROJECT

The two LDRs and the computer measure Time. They are especially useful if the time interval is so short that it cannot be measured accurately using a stopwatch.

Experiments to measure the effect of gravity on an object are possible if the object is large enough to trigger the sensors as it falls.

The sensors can be used to time reactions - if the first sensor is triggered out of sight by someone putting a hand over it, a second person can see how quickly the other sensor can be covered after *Clock running ...* appears on the screen.

Find some VERY long pieces of low resistance wire and place a sensor at each end of a running track to accurately time races!

Project 3

COUNTING

In this project two Light Dependent Resistors are used to make an automatic counting system.

THE EQUIPMENT

You will need -

the A to D interface
two Light Dependent Resistors
two two-core leads with connectors at one end
two 1K (1000 ohm) resistors coloured brown, black, red
a push-to-make switch

The equipment used for this project is identical to that used for project 2. The LDRs and their leads together with the resistors should be connected as figure 10.

THE PROGRAM

The BBC microcomputer has the facility for long variable names which makes programs much easier to understand. However, the computer can work faster if it is using its own variables - the *resident integer variables* A% to Z%. In this program it is important that the computer reacts to changes in signals from the sensors as quickly as possible.

The program uses some of the resident integer variables and their uses are described in lines 60 to 140. The two flags, F% and G%, are needed to ensure that the program doesn't continue counting when the light at a sensor is reduced. They are set to one in lines 300 and

320 when the light level changes and are not set back to 0 until the light level returns to its original value. L% holds the amount of change needed to trigger the count in the same way that **sense%** did in program 2 - this can be altered to make the system more or less sensitive.

Line 40 allows the **ESCAPE** key to be pushed to restart the program. Do not include this line until you are sure that the rest of the program is correct. To leave the program **BREAK** has to be pushed but the program can be recovered by typing **OLD**.

Program 3:

```
 10 MODE 7
 20 VDU 23;8202;0;0;0;
 30 *FX16,2
 40 ON ERROR RUN
 50
 60 L%=20:REM sensitivity setting
 70 A%=0:REM channel 1
 80 B%=0:REM channel 2
 90 C%=0:REM count
100 F%=0:REM flag for channel 1
110 G%=0:REM flag for channel 2
120 S%=0:REM switch result
130 Y%=0:REM channel 1 setting
140 Z%=0:REM channel 2 setting
150
160 PRINT TAB(1,3);"Waiting for the set signal ... ";
170 REPEAT
180 S%=ADVAL(0) AND 1
190 UNTIL S%=1
200 A%=ADVAL(1) DIV 64
210 B%=ADVAL(2) DIV 64
220 Y%=A%+L%
230 Z%=B%+L%
240 CLS
250 PRINT TAB(10,10);"Count = ";
260
270 REPEAT
280 A%=ADVAL(1) DIV 64
290 B%=ADVAL(2) DIV 64
300 IF A%>Y% AND F%=0 C%=C%+1:F%=1
310 IF Y%-A%>0 F%=0
320 IF B%>Z% AND G%=0 C%=C%-1:G%=1
330 IF Z%-B%>0 G%=0
340 PRINT TAB(18,10);C%;" ";
350 UNTIL FALSE
```

When the program is run it waits until the switch is pushed to set the light levels.

Once the switch has been pushed the computer looks for a decrease in the light levels at the sensors. If the light decreases at sensor 1 the count increases and if the light decreases at sensor 2 the count decreases.

Fix the sensors so that they are facing a constant light source - a torch or a light bulb - before pushing the switch. It is important that the light source does not vary as the signal from the LDR must be able to return to its original value after an object has passed in front of the sensor (figure 12).

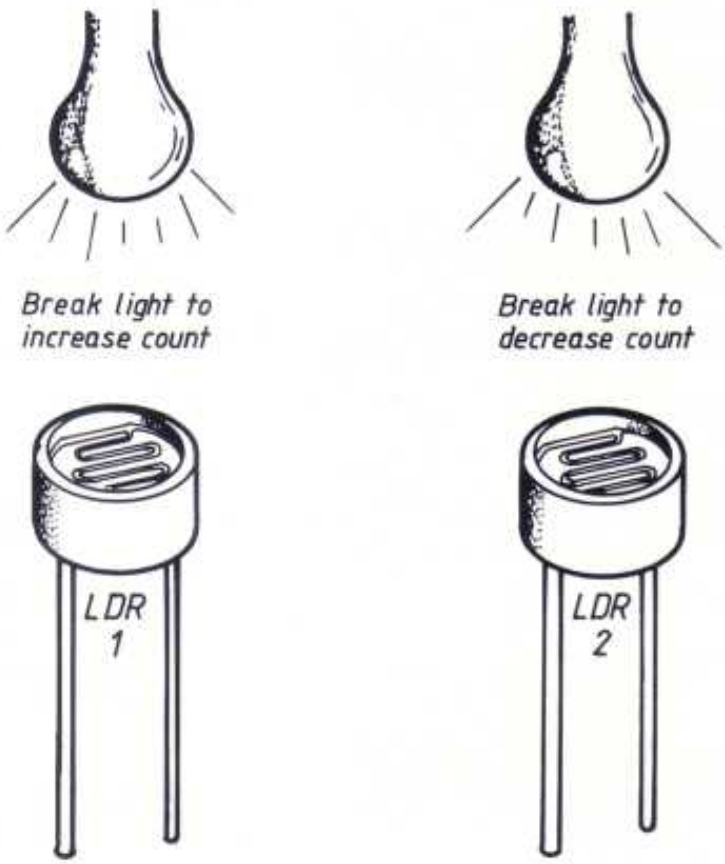


Figure 12

As objects pass between the lights and the LDRs the count will increase or decrease. The system should work quickly enough to count the fingers of an outspread hand passed in front of sensor 1.

To restart the count or reset the light levels push **ESCAPE**.

WHAT TO DO IF IT DOESN'T WORK

If project 2 worked correctly it is unlikely that the sensors or the interface will be at fault unless a wire has been pulled out of a connector.

Most problems will be caused by unwanted variations in the light. Make sure that the sensors are fixed firmly so that they are pointing directly towards the light - in between two large books is usually a satisfactory solution. Daylight is by far the brightest light so try to avoid pointing the sensors towards a window unless it is a cloud-free day.

SOME IDEAS TO TRY

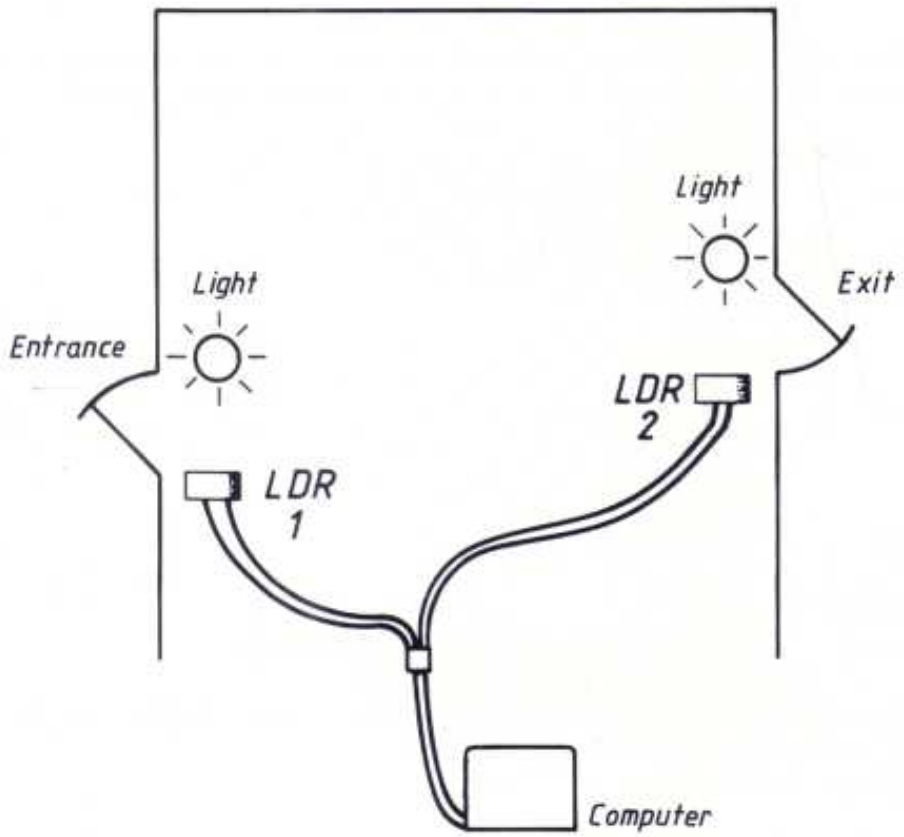


Figure 13

The sensors can be positioned as shown in figure 13 in a room with two doors to keep a count of the number of people in the room. You will have to make sure that nobody leaves by the entrance or enters via the exit or the results will be incorrect.

The system is not fast enough to count accurately at high speed - for example the pages of a book being flicked in front of it - but it can count quite quickly if the objects are clearly defined. Try dropping pencils quickly between the sensor and the light source to find out how fast the computer reacts. The variable L% can be adjusted for optimum reactions.

Using only one sensor, see if you can alter the program so that it prints a welcoming message on the screen when a person passes. If the system is used at a party it can even tell the guests how many other people have arrived (or left)!

Project 4

USING A THERMISTOR TO MEASURE TEMPERATURE

The thermistor is a device which produces a resistance dependent upon its temperature. This project uses a thermistor to produce a continuous graph of the temperature on the screen.

THE EQUIPMENT

You will need -

the A to D interface
a 15K ohm disc, rod or bead thermistor (see below)
a two-core lead with connectors at one end
a 15K (15,000 ohm) resistor coloured brown, green, orange
a push-to-make switch
an ice cube and a centigrade thermometer

A *Thermistor* is similar to a Light Dependent resistor in that it creates a resistance which varies according to its environment. However its resistance is dependent on temperature rather than light - it could be called a Temperature Dependent Resistor instead of a Thermistor.

There are many different types of thermistor and they are usually described by their resistance at 25 C. The thermistor recommended for this project has a resistance of 15K ohm at 25 C but any thermistor between about 1K ohm and 27K ohm can be used providing that the value of the resistor is the same as that of the thermistor.

The thermistor is connected to the lead in the same way as the LDR was in project 1. The lead is fixed into connectors 6 and 7 on the interface and the 15K resistor is fixed between connectors 7 and 8.

If the thermistor is a small disc or bead the wires from it will be very close together. If they are squashed together and make contact no harm will result to the computer but the results will be incorrect. It is a good idea to wrap a small piece of tape around one of the leads to insulate it from the other. Figure 14 shows the equipment set up and ready for use.

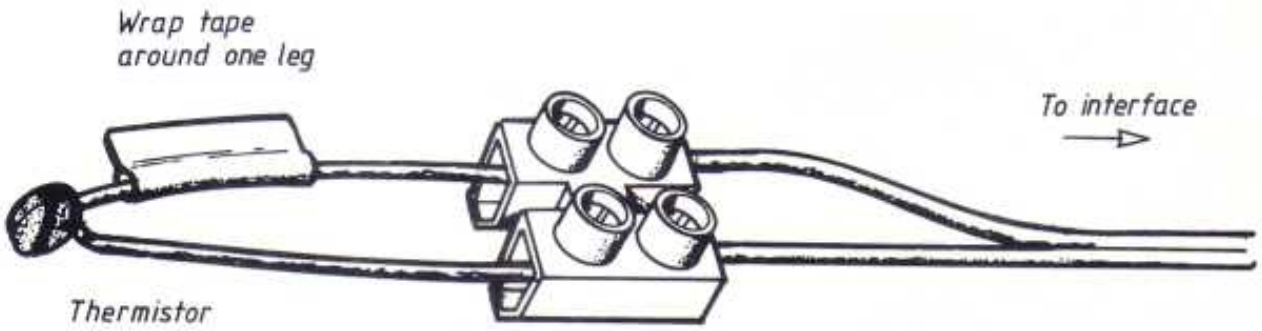


Figure 14

THE PROGRAM

The program is similar to the first program for the LDR. A line is continuously drawn across the screen but at a slower speed as the reaction from a thermistor will be very much slower than the reaction from a LDR.

There is little point in measuring temperature without some sort of relative scale so the first part of the program - lines 70 to 180 - draws and numbers a scale on the screen.

Unfortunately, the sort of inexpensive thermistors used for the project are not very accurate and even two of the same value can vary greatly. Rather than altering the height of the line on the screen the vertical position of the scale is controlled by the variable **height%** which is set in line 30. By increasing or decreasing the value of **height%** the scale can be repositioned higher or lower on the screen.

The variable **factor** - line 40 - is used to expand or contract the scale. Its use is explained in *USING THE PROJECT*. Note that **factor** is not an integer variable as it holds decimal values so it is not followed by a % sign.

Lines 210 to 230 form the loop which draws the line across the screen according to the value read from the thermistor. As with the LDR an opposite effect is created as the resistance of the thermistor decreases as the temperature rises. To make the line on the screen rise as the temperature increases the value read by **ADVAL(1)** is subtracted from 800.

Program 4:

```
 10 MODE 0
 20 VDU 23;8202;0;0;0;
 30 height%=120
 40 factor=1.0
 50
 60 REPEAT
 70 CLS
 80 VDU5
 90 VDU 29,0;312+height%;
100 FOR C%=-40 TO 50 STEP 10
110 MOVE 0,((C%*8)+12)*factor
120 PRINT;(C%/10)+4;"-";
130 NEXT
140 VDU4
150 MOVE 32,424*factor
160 DRAW 32,-320*factor
170 DRAW 1280,-320*factor
180 VDU26
190 MOVE 32,800-ADVAL(1) DIV 64
200
210 FOR count%=48 TO 1280
220 DRAW count%,800-ADVAL(1) DIV 64
230 NEXT
240
250 UNTIL FALSE
```

USING THE PROJECT

The first thing you must do is to calibrate the screen display. This involves comparing the temperature read by the thermistor with a known temperature.

Hold the thermistor and a thermometer in the palm of your hand and wait for the temperatures to settle. The line on the screen should be registering 30 to 40 degrees. Compare the temperature read by the thermometer with the thermistor's reading and alter the variable **height%** to move the scale up or down until the reading from the thermistor is the same as the temperature shown by the thermometer. Reducing **height%** by 20 will move the scale down (and therefore the temperature reading up) by about 2 degrees, and increasing **height%** by 20 will move the scale up and the temperature down by the same amount.

When you are satisfied that the temperature of the thermometer and the thermistor read the same put the thermistor on an ice cube and see if the line on the screen falls below or above 0 degrees. Be careful not to get the thermistor wet as this will affect the reading.

The variable **factor** stretches or squashes the scale if it is larger or smaller than 1.0. If the line has fallen below the 0 degree line increase **factor** to 1.1 and try again, if the line has not reached 0 degrees reduce **factor** to 0.9. If you want to create a very precise measuring device you will have to use two places of decimals when adjusting **factor**.

The scale is based on 40 degrees and increasing **factor** expands the scale outwards from this point. As body temperature is around 35 degrees this is a suitable temperature to use for the first calibration.

If **factor** has to be altered by a large amount the variable **height%** will also need to be re-adjusted so you will have to repeat the previous stages again. By adjusting the two variables you should be able to create a fairly accurate temperature display on the screen.

Figure 15 shows the relationships between **height%** and **factor**.

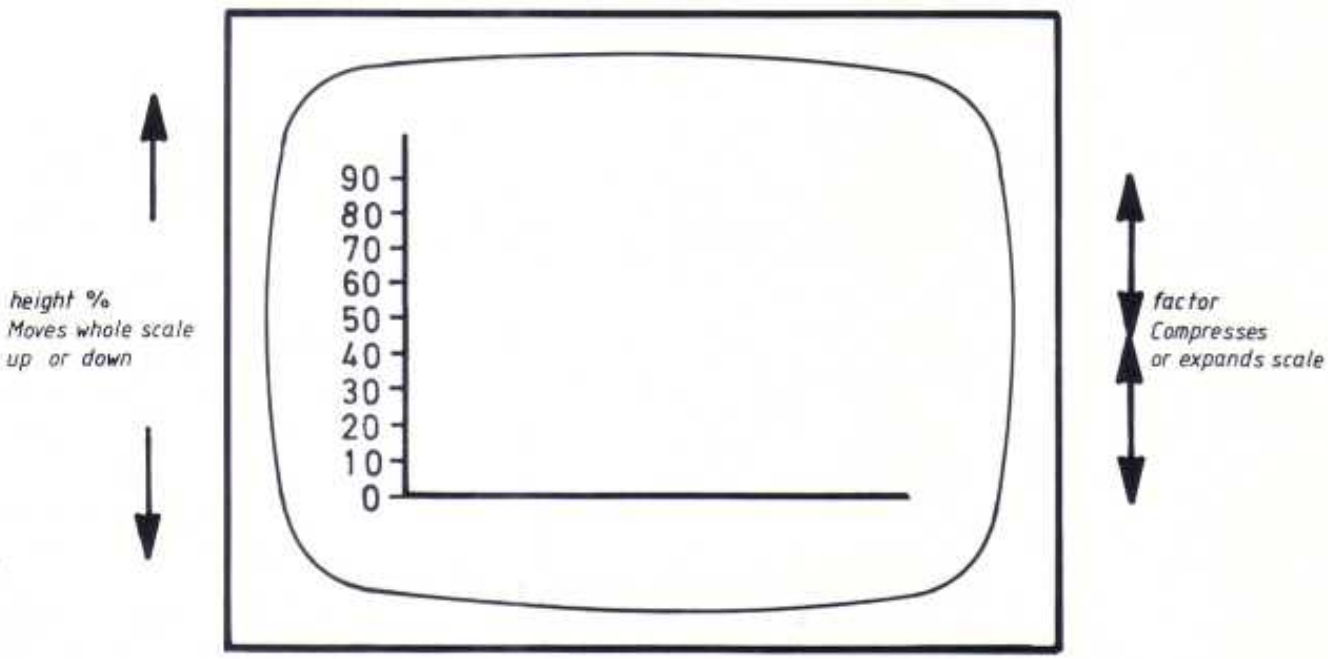


Figure 15

WHAT TO DO IF IT DOESN'T WORK

Thermistors are reliable devices designed to withstand extreme temperatures so it is unlikely that the device will be at fault.

Check to make sure that the value of the resistor is the same as the value of the thermistor.

If the line on the screen is much too low to allow adjustment by **height%** increase the figure '800' in lines 190 and 220 to 1000. This will move the relative position of the line higher on the screen and will still allow **height%** and **factor** to be used to adjust the scale.

AN IDEA TO TRY

The thermistor can be used as an alarm to warn of high or low temperatures. If an extra line is inserted in the loop

```
225 PRINT TAB(1,1);ADVAL(1) DIV 64
```

the actual reading by the computer's analogue to digital converter will be shown in the top left hand corner of the screen.

Make a note of the reading at which you wish the alarm to be triggered and add

```
227 IF ADVAL(1) DIV 64>=??? REPEAT:VDU7:UNTIL FALSE
```

where **???** is the level at which the alarm should sound (if the alarm is warning of low temperatures >=??? will have to read <=???). Once the critical temperature has been reached the computer will *beep* until **ESCAPE** is pushed.

Project 5

"SEESAW" - A GAME USING A TILT SWITCH

Project 5 is an amusing game which involves balancing a ball on a seesaw. The seesaw is controlled by a Tilt Switch.

THE EQUIPMENT

You will need -

the A to D interface
a tilt switch
a two-core lead with connectors at one end
a 1K (1,000 ohm) resistor coloured brown, black, red
a push-to-make switch

A *Tilt Switch* is a glass or resin encapsulated globule of mercury which connects together the two leads inside it when it is at an angle greater than horizontal (figure 16).

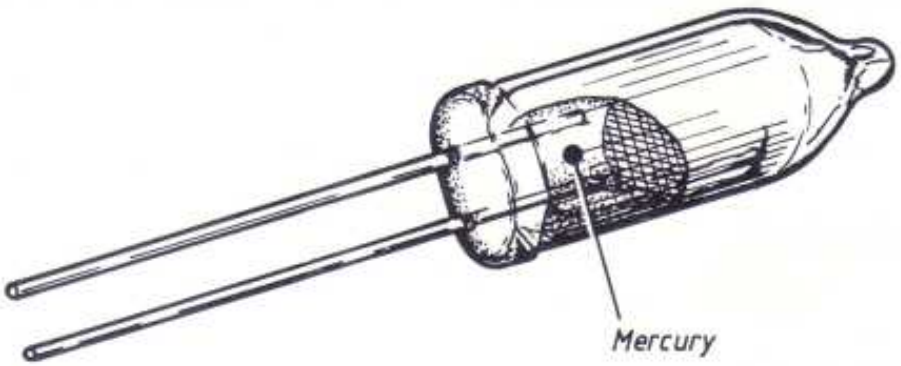


Figure 16

Because it reacts very quickly to movement it can be used to control the computer via the *analogue in* port.

The tilt switch contains mercury which is a toxic substance so the device should never be opened and should be treated with care.

Connect the tilt switch to the lead joined to connectors 6 and 7 and fix a resistor between connectors 7 and 8. As the computer will only be looking for high or low readings the exact value of the resistor is not important.

THE PROGRAM

The program is longer than the previous programs as the game is displayed graphically on the screen but it is worth the effort of typing in as the resulting game is very entertaining.

Line 30 tells the computer that only one channel of the A to D converter will be used. Lines 50 to 110 set up two arrays which hold the vertical and horizontal positions for the ball on the seesaw, this saves having to work out the positions each time the ball moves. Line 220 calls **PROCscreen** to set up the initial display.

The main loop is contained in lines 240 to 290. Lines 260 and 270 check to see if the tilt switch has been moved and call **PROCbeam** and **PROCroll** if necessary. The program uses **GCOL3,3** to overdraw both beams each time **PROCbeam** is called but because one was already drawn in the original screen display only one beam is ever showing.

PROCroll calls **PROCball** to draw the ball on the seesaw and checks to make sure it hasn't fallen off the end. If it has the program waits until the push to make switch is pressed - lines 300 to 320 - before restarting.

The variable **speed%** is decreased each time the beam is moved. In line 280 it is compared with the BBC micro's **TIME** function to work out if the ball should roll. As **speed%** decreases so the speed of the game increases as less time is spent between each movement of the ball.

Program 5:

```
10 MODE 1
20 VDU 23;8202;0;0;0;
30 *FX16,1
40
50 DIM H%(10)
60 DIM V%(10,1)
70 FOR C%=0 TO 10
80 H%(C%)=284+C%*60
```

```
 90 V%(C%,0)=632-C%*20
100 V%(C%,1)=432+C%*20
110 NEXT
120
130 GCOL3,3
140 flag%=1
150 OH%=H%(5)
160 OV%=V%(5,1)
170 TIME=0
180 speed%=50
190 C%=5
200 finish%=FALSE
210 score%=0
220 PROCscreen
230
240 REPEAT
250 channel%=ADVAL(1) DIV 64
260 IF channel%>512 AND flag%<>0 PROCbeam:flag%=0:PROCroll
270 IF channel%<512 AND flag%<>1 PROCbeam:flag%=1:PROCroll
280 IF TIME>speed% C%=C%+1-(flag%*2):PROCroll:TIME=0
290 UNTIL finish%=TRUE
300 REPEAT
310 IF (ADVAL(0) AND 1)=1 RUN
320 UNTIL FALSE
330
340 DEF PROCbeam
350 SOUND 1,-15,score%/5,1
360 MOVE 300,400:DRAW 900,600
370 MOVE 300,600:DRAW 900,400
380 IF speed%>2 speed%=speed%-2
390 ENDPROC
400
410 DEF PROCroll
420 IF C%<0 OR C%>10 SOUND 0,-15,100,10:finish%=TRUE:ENDPROC
430 score%=ABS(5-C%)+score%
440 PRINT TAB(7,1);score%
450 PROCball(OH%,OV%)
460 PROCball(H%(C%),V%(C%,flag%))
470 OH%=H%(C%)
480 OV%=V%(C%,flag%)
490 ENDPROC
500
510 DEF PROCscreen
520 MOVE 600,492
530 DRAW 560,400
540 PLOT 85,640,400
550 MOVE 300,400
560 DRAW 900,600
570 PROCball(OH%,OV%)
580 PRINT TAB(1,1);"Score=0"
```

```
590 ENDPROC
600
610 DEF PROCball(X%,Y%)
620 VDU5
630 MOVE X%,Y%
640 PRINT"O"
650 VDU4
660 ENDPROC
```

PLAYING THE GAME

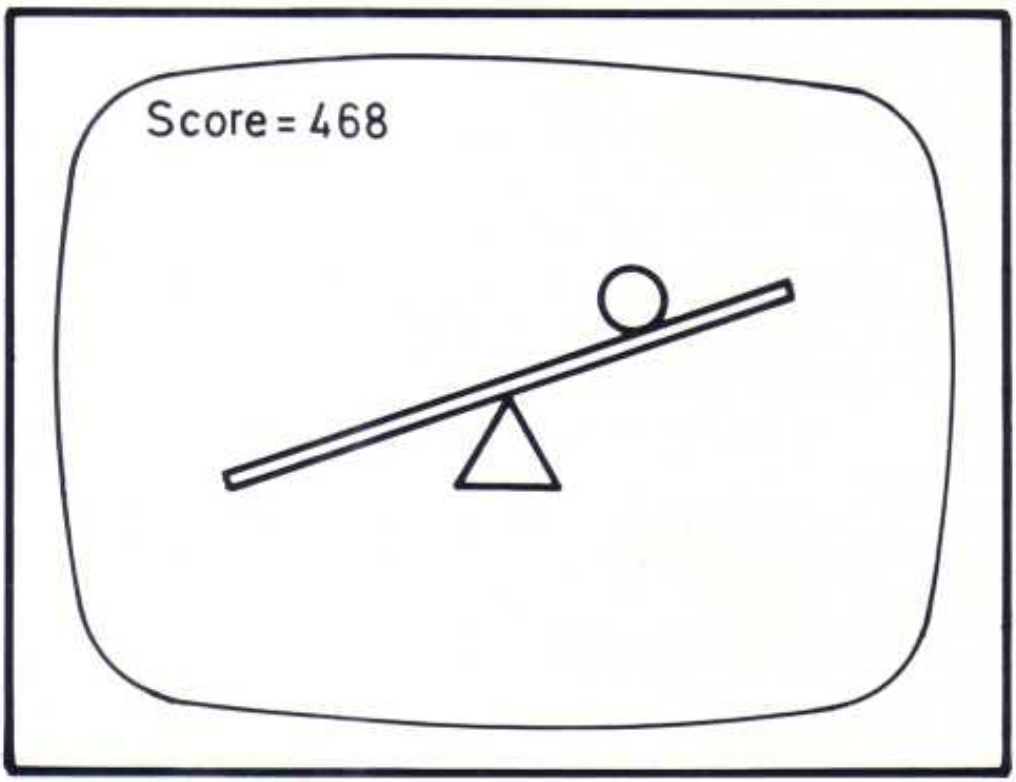


Figure 17

The game starts with the seesaw being drawn on the screen (figure 17). Whichever way the seesaw is tilted the ball will roll, your task is to stop it from rolling off the end of the beam by moving the tilt switch to an angle greater or less than horizontal.

Each time the seesaw is moved the speed at which the ball rolls increases until it is rolling as fast as possible. With each movement of the ball the score increases - how much it increases by depends on how near to an end of the beam the ball is.

To gain a high score do not tilt the beam until the ball is near the end and remember that with each tilt the speed will increase.

To restart the program when the ball has fallen off the seesaw push the switch on the interface.

WHAT TO DO IF IT DOESN'T WORK

Save the program on tape or disc and type in:

```
10 REPEAT
20 PRINT ADVAL(1) DIV 64
30 UNTIL FALSE
```

The computer should print either 0 or 1023 depending on the angle of the tilt switch. If this does not work check all connections very carefully.

As the tilt switch is either on or off very little can go wrong with it. It is likely that any problems will be due to errors made when typing in the program or poor connections on the interface.

Project 6

A DRAWING SYSTEM USING TWO POTENTIOMETERS

This project introduces potentiometers and uses them to operate a point to point drawing system on the screen.

THE EQUIPMENT

You will need -

the A to D interface
two 2K linear potentiometers
two two-core leads with connectors at one end
two 2.2K (2,200 ohm) resistors coloured red, red, red
a push-to-make switch

Potentiometers, or *pots* as they are commonly called, are variable resistors. Pots are used in most electronic equipment as a method of control of the output by the user - the volume and tone controls on a radio are pots. Inside a pot is a wiper which is moved along or round a carbon track which presents a resistance to the electrical signal. The resistance depends on how far along the track the signal has to travel.

Pots can be *Linear* or *Logarithmic*. For this project linear pots should be used as they produce an increase in resistance directly related to the amount of movement of the wiper.

A pot is described by its maximum resistance i.e. a 2K pot can produce any resistance between 0 ohms and 2K ohms. The resistors used need to be the same as or slightly greater than the resistance of the pot.

The pots are fixed to the ends of the leads by the connector blocks and the resistors are fitted between connectors 7 and 8 and 5 and 4.

There will be three terminals on the pot and to make the pot *work* in a clockwise direction the left and middle terminals should be used. If the right and middle terminals are used the pot will still work but in reverse.

If some of the plastic insulation at the end of the connector is cut away you will be able to fix the two terminals of the pot directly into the connectors (figure 18). If the terminals on the pot are too wide to fit into the connectors they can be squashed using a pair of pliers.

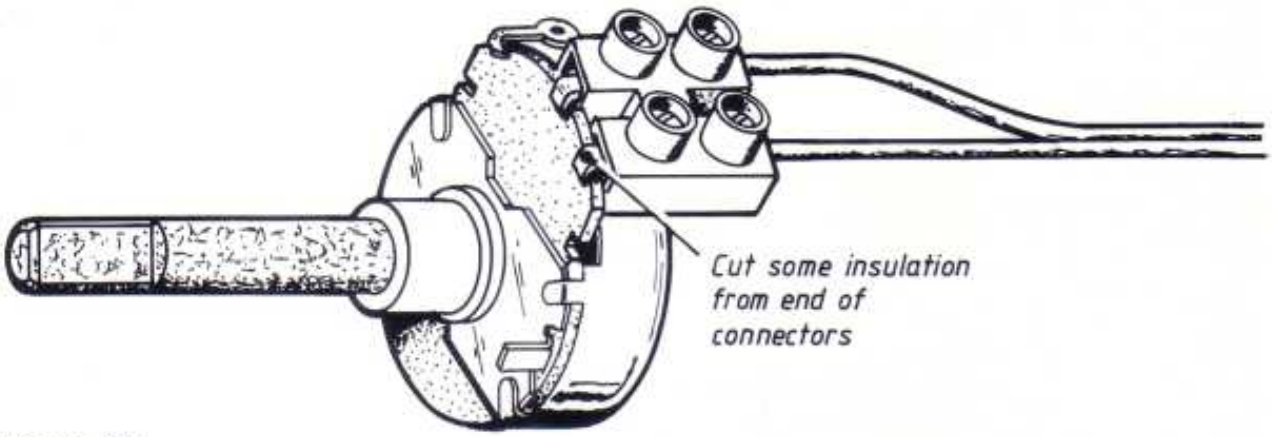


Figure 18

THE PROGRAM

The program prints a + on the screen which is moved around by the pots. When the switch on the interface is pushed a line is drawn from the end of the previous line to the present position of the +.

PROCpoint prints the + on the screen in its present and previous positions using **GCOL3,3**. **X%** and **Y%** hold the present vertical and horizontal positions and **OX%** and **OY%** hold the previous positions.

The last point which has been drawn to is held by **SX%** and **SY%** and these variables are updated when a new line is drawn by **PROCdraw**.

The non integer variable **factor** is used to adjust the readings from the A to D converter so that the range of the pots is contained within the full screen.

The value for **X%** and **Y%** is worked out by the functions **FNX** and **FNY** which read the A to D converter. The value for **X%** is multiplied by 1.25 to allow for the rectangular shape of the screen.

Line 140 checks for a ket pressed on the computer to signal *move* or *clear the screen*.

Program 6:

```
 10 MODE1
 20 PRINT TAB(0,30);"Space bar=Clear";
 30 PRINT TAB(20,30);"Any other key=Move ";
 40 factor=2
 50 X%=FNX
 60 Y%=FNY
 70 SX%=X%
 80 SY%=Y%
 90 PROCpoint(X%,Y%)
100
110 REPEAT
120 OX%=X%
130 OY%=Y%
140 I%=INKEY(1)
150 X%=FNX
160 Y%=FNY
170 PROCpoint(OX%,OY%)
180 PROCpoint(X%,Y%)
190 IF (ADVAL(0) AND 1)=1 PROCdraw
200 IFI%>32 SX%=X%:SY%=Y%
210 UNTIL I%=32
220 RUN
230
240 DEF PROCpoint(A%,B%)
250 GCOL3,3
260 VDU5
270 MOVE A%-16,B%+16
280 PRINT;"+";
290 VDU4
300 ENDPROC
310
320 DEF PROCdraw
330 GCOL0,2
340 MOVE SX%,SY%
350 DRAW X%,Y%
360 SX%=X%
370 SY%=Y%
380 ENDPROC
390
400 DEF FNX:=(ADVAL(1) DIV64)*factor*1.25
410
420 DEF FNY:=(ADVAL(2) DIV64)*factor
```

HOW IT WORKS

To understand the theory behind what is happening imagine that connector 8 is a tap with a hose connected to it and that the tap is turned on. There are two routes the water can take. One is out of a hole in the hose at connector 7 and the other is out of the end of the hose at connector 6.

When the end of the hose is blocked (i.e. the pot is turned so that it is producing a high resistance) the water will escape through the hole at connector 7. But when the pot is turned so that it is producing no resistance the water will take the easiest path - out of the end of the hose at connector 6.

The electrical signal behaves in the same way as the imaginary water, it will only escape through the A to D converter if that is the easiest route it can take. As soon as it can escape directly to ground through connector 6 it will flow in that direction.

When the pot is partly turned some of the electricity will be able to escape through connector 6 and some will still flow through connector 7 producing an intermediate reading (figure 19).

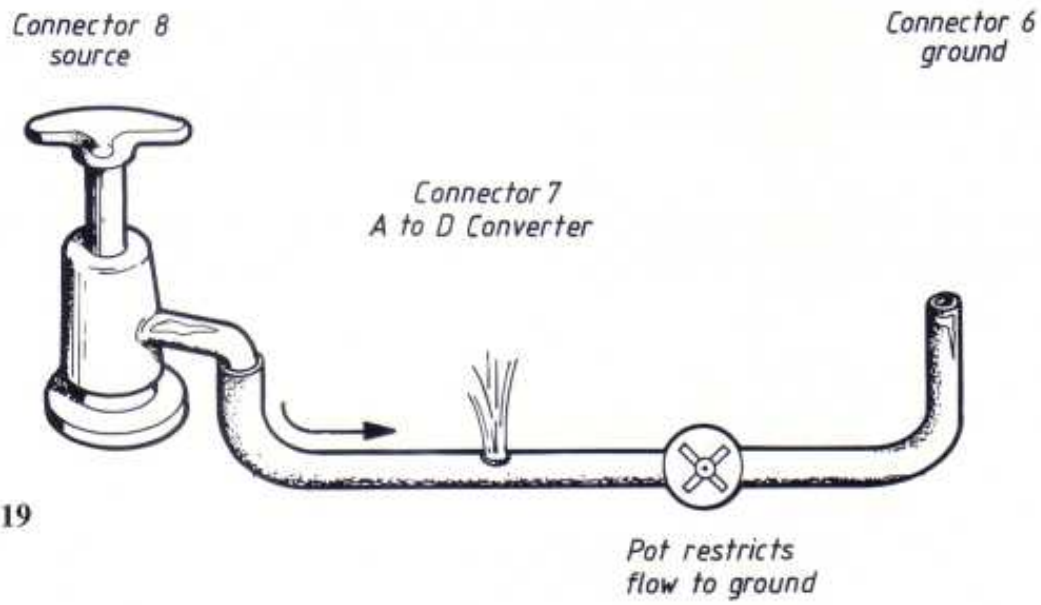


Figure 19

USING THE PROJECT

The program produces a screen drawing aid. The controls are -

Pot connected to connectors 6 and 7 - **VERTICAL POSITION**
Pot connected to connectors 3 and 4 - **HORIZONTAL POSITION**
Push to make switch - **DRAW**
Space bar - **CLEAR THE SCREEN**
Any other key - **MOVE WITHOUT DRAWING A LINE**

The cross is moved using the pots, and the push to make switch draws from the *previously drawn to* point to the present position. If no line is required push the space bar rather than the push to make switch and the present position of the cross will become the *previously drawn to* point.

Any other key clears the screen - this is useful at the start to allow the cross to be placed in the desired position.

If a full movement of the pots fails to move the cross to the top right hand corner of the screen try increasing the value of **factor**. If, however, the cross can be moved right off the screen then decrease the value of **factor**.

Some reasonably accurate drawings can be constructed using the system if care is taken when positioning the cross. The cross may be inclined to wobble slightly but this will be due to poor contacts in the pots and cannot be cured except by buying more accurate and therefore more expensive potentiometers.

SOME IDEAS TO TRY

Can the software be changed to allow certain keys to change the colour of the line? **IF** statements will have to be added to the main loop and a variable for the colour will need to be defined. Remember that **GCOL3,?** rather than **GCOL0,?** is being used - some peculiar effects may result otherwise. Remember also that **I%** will contain the *ASCII code* of the key pressed.

Can you add another symbol, perhaps *, to show where the *previously drawn to* point is? This symbol will have to be erased and repositioned when a new line is drawn.

AND NOW TRY

If you have worked through all the six analogue to digital converter projects you will have measured time, light, and temperature and will have controlled the screen display using devices other than the computer's keyboard. But the projects should not be an end in themselves - try altering the programs to produce different results from the equipment or altering the equipment to produce different results from the programs.

The uses that can be found for the analogue to digital converter channels are endless and are only limited by the imagination. Provided the rule that the input voltage is kept below 1.8 volts is followed - which is easily adhered to if you use the reference voltage only - you can experiment as much as you like.
Some possible suggestions for projects are made below and these in turn should suggest other uses for the A to D port. Some of these projects may be quite difficult to set up and program but the satisfaction gained when the system works is always a just reward. To implement some of these projects you will need to use all four channels of the BBC micro's A to D converter, details of how to extend the interface can be found in Appendix 2.

* **A MINI WEATHER STATION** - you have already measured light and temperature but the LDR could also be used to measure wind speed if it counted the revolutions of an anemometer over a fixed time. Perhaps a pot could be used to record wind direction or two thermistors - one wet, one dry - used to measure relative humidity. Can the data all be shown on the screen at the same time?

* **A REED SWITCH** is a pair of contacts enclosed in a glass tube. If a magnet is passed close to the tube the reeds part. A reed switch can be used to record the position of a vehicle carrying a magnet as it passes. If a number of reed switches are connected in series with different value resistors but are all connected to the same channel can the computer decide which particular switch has been broken?

* Wire itself has a resistance and this resistance increases with the distance along the wire that the electrical signal has to travel. Using a fairly high resistance wire can you make a measuring device which displays the results on the screen? Can you attach a spring to this measure to take pressure readings?

* Can you mount the potentiometers used in project 6 so that they form a joystick. The stick itself will have to be able to move north, south, east and west adjusting the appropriate pot as it moves.

SOME HINTS

When starting a project do not worry too much about writing a perfect program - the software can be tidied up at the end - it is more important to get the system working.

If the hardware seems to be producing peculiar results try-

```
10 REPEAT
20 PRINT ADVAL(channel number) DIV64
30 UNTIL FALSE
```

Often the actual readings by the A to D coverter can give an indication whether the problem lies with the hardware or the software.

The A to D chip in the BBC microcomputer is only accurate to ten bits. Always use **DIV 64** after an **ADVAL** call.

Don't make the leads from the interface too long or the resistance of the lead will affect the reading.

If you haven't already read it, read the **ADVAL** entry in the keywords section of the *User Guide* on page 202, and also the paragraphs on analogue input on page 467.

There is a endless scope for experimentation using the BBC's A to D converter. *Happy sensing!*

THE USER PORT INTERFACE

Underneath the BBC microcomputer there is a twenty way connector labelled **user port**. Eight of the pins in the connector carry lines which are switchable by the computer, two are *handshake* lines, and the others are 5 volt supply lines or are connected to ground.

The pins are connected to a chip inside the computer called the **6522 Versatile Interface Adaptor** (VIA). This chip is truly versatile - it contains two timers and sixteen addressable, eight-bit registers - and can offer a wide range of facilities, but for these projects we will only be concerned with four of the switchable lines and ground.

The switchable lines can be used to feed signals into the computer or to send signals out to an external device connected to the user port. We will be using the lines to send signals out from the computer to switch on and off motors and lights connected to the interface.

THE INTERFACE COMPONENTS

You will need -

a 20 way IDC socket
1 metre of 16 way ribbon cable
a 16 way IDC dual-in-line connector
a piece of 'breadboard'
four 1N4001 diodes
a short piece of solid core (0.6mm) equipment wire

The components used for the interface are designed to be soldered onto printed circuit boards. However, if they are fixed permanently into place they cannot be reused so we will use a piece of **Breadboard** to build the interface on (figure 20).

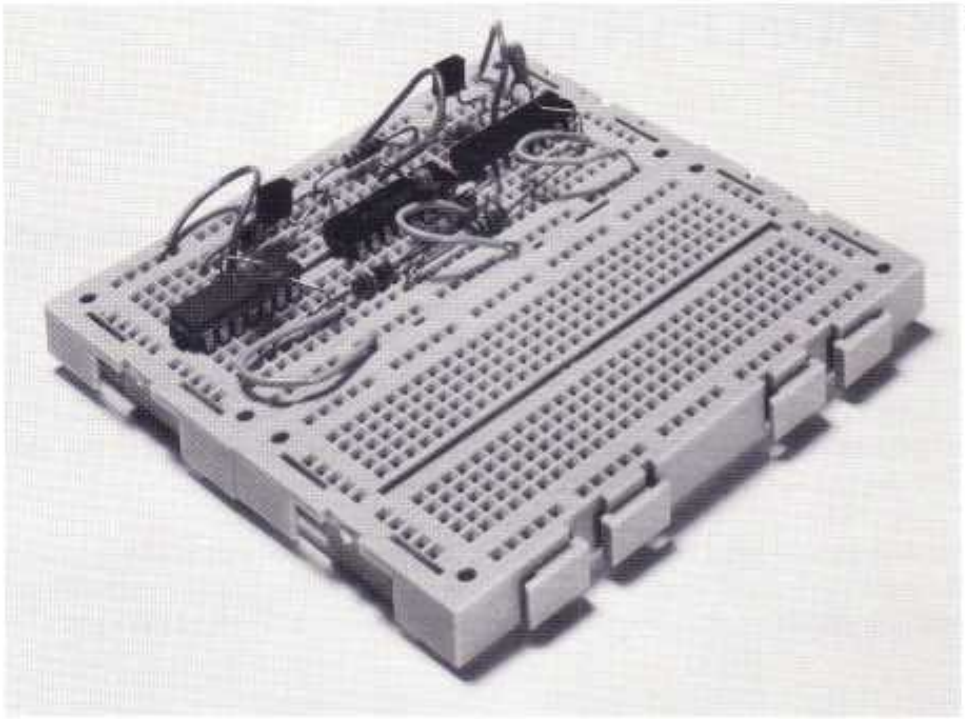


Figure 20

Photo: shows two blocs

Breadboard is a block of plastic with internal electrical connections on which integrated circuits (chips) and other components can be arranged to make a circuit, but its great advantage lies in the fact that the components can be removed, replaced and rearranged as required.

Breadboard is quite expensive to buy but it is almost indestructable and will last as long as you have experiments to use it for and, as none of the components have to be soldered to the board, they also can be reused time and time again.

The most commonly available make of breadboard is *Verobloc* made by **BICC-VERO** so all illustrations and instructions will refer to this design, but other types of breadboard may be used although the layout of components might need to be adapted to suit.

MAKING THE LEAD

As with the **analogue in** port, the socket used to plug devices into the user port is of the **Insulation Displacement Type**.

(The terms *plug* and *socket* can be rather confusing as a socket is usually thought of as being fixed to the equipment and a plug attached to a lead. Using the correct terminology a plug has pins sticking out and a socket doesn't. In this case the plug is fixed to the BBC micro and the socket is on the end of the ribbon cable - the opposite of the A to D interface.)

The socket is *keyed* so that it can only be fitted into the plug on the computer one way round. The key will be a block of plastic sticking out in the middle of one side. Sockets which aren't keyed should never be used as they can be plugged in the wrong way round which could damage either the computer or the equipment at the other end of the cable.

Hold the socket so that the key is on the side pointing away from you and place the 16 way ribbon cable on the leftmost teeth so that the four teeth at the right of the plug will not be connected (figure 21). Put the top on the plug and squash it all together in a vice to make the teeth pierce the ribbon cable.

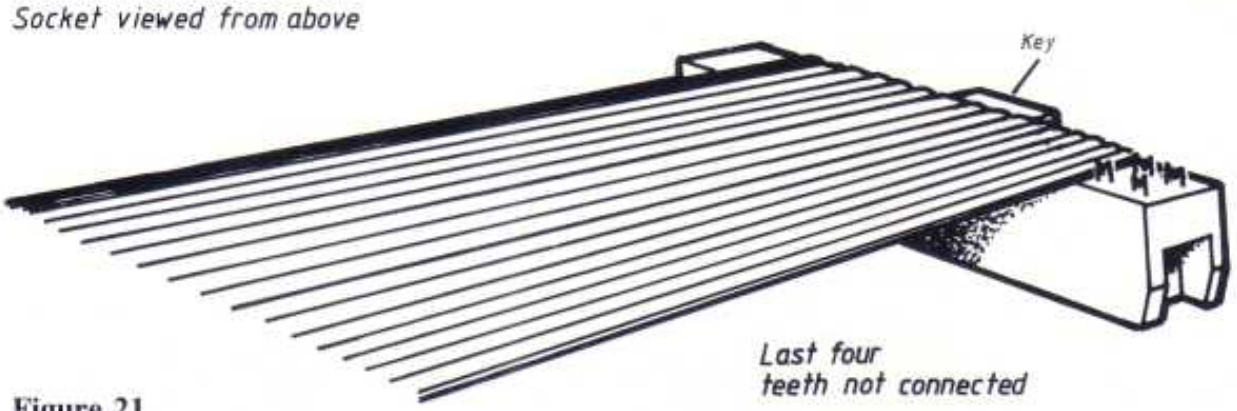


Figure 21

Check again to make sure that the four teeth on the right of the plug are not connected and that your plug and cable is connected as shown in figure 21.

At the other end of the ribbon cable the 16 way **d.i.l. IDC connector** is fitted (*d.i.l.* stands for dual-in-line). The connector will transfer the signals in the ribbon cable to two rows of pins which will plug directly into the breadboard. Lay the ribbon cable flat with the socket pointing upwards and fit the connector the opposite way around so that its pins are pointing downwards (figure 22).

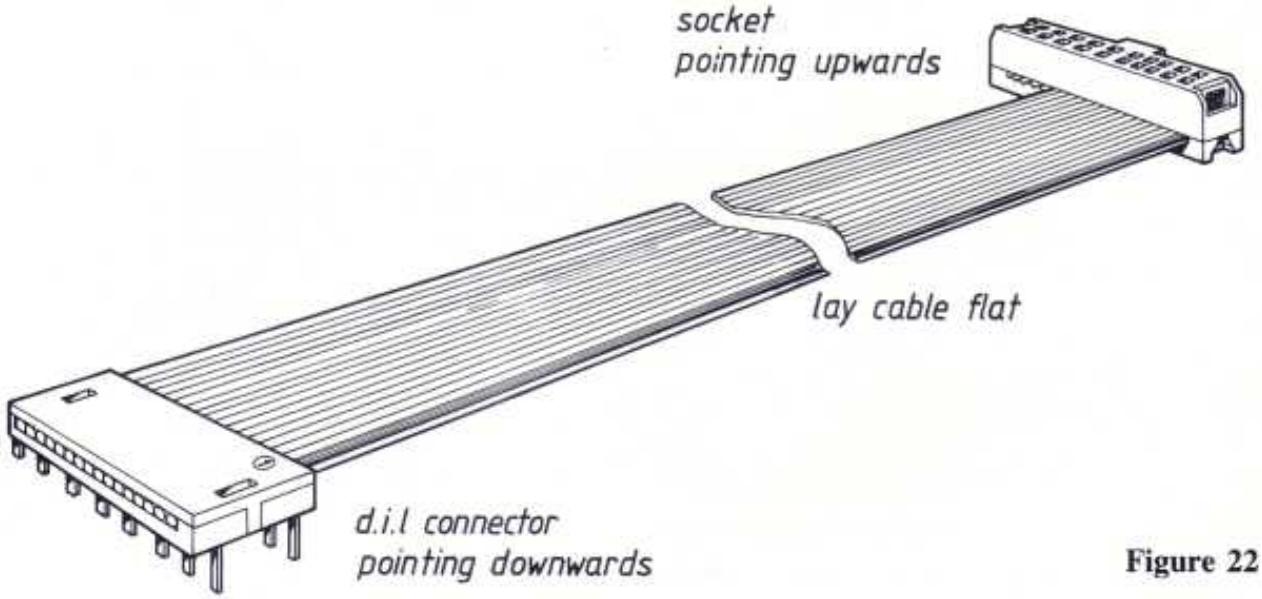


Figure 22

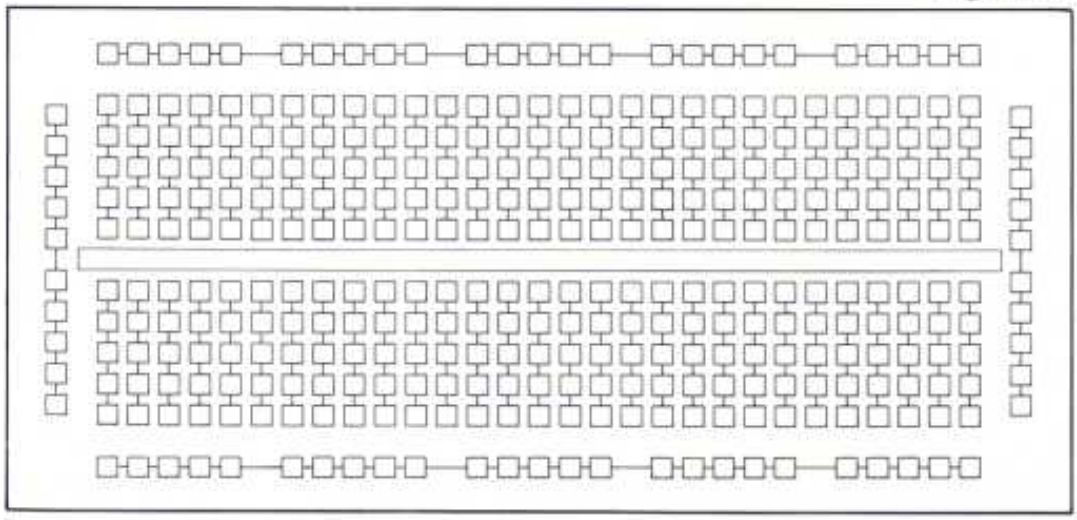
The connector is fitted in the same way as the socket by squashing the cable onto the teeth. However, you will have to place a small piece of wood or metal between the pins when you put the connector in the vice or they will get squashed as well.

Before you continue check that your lead is exactly the same as the lead shown in figure 22.

THE BREADBOARD END

Figure 23 shows which holes on the breadboard are connected to which. The strips along the top, bottom and ends are called *rails* and are used for carrying the positive or negative supply. The rows of holes leading out from the space in the centre are used for the components.

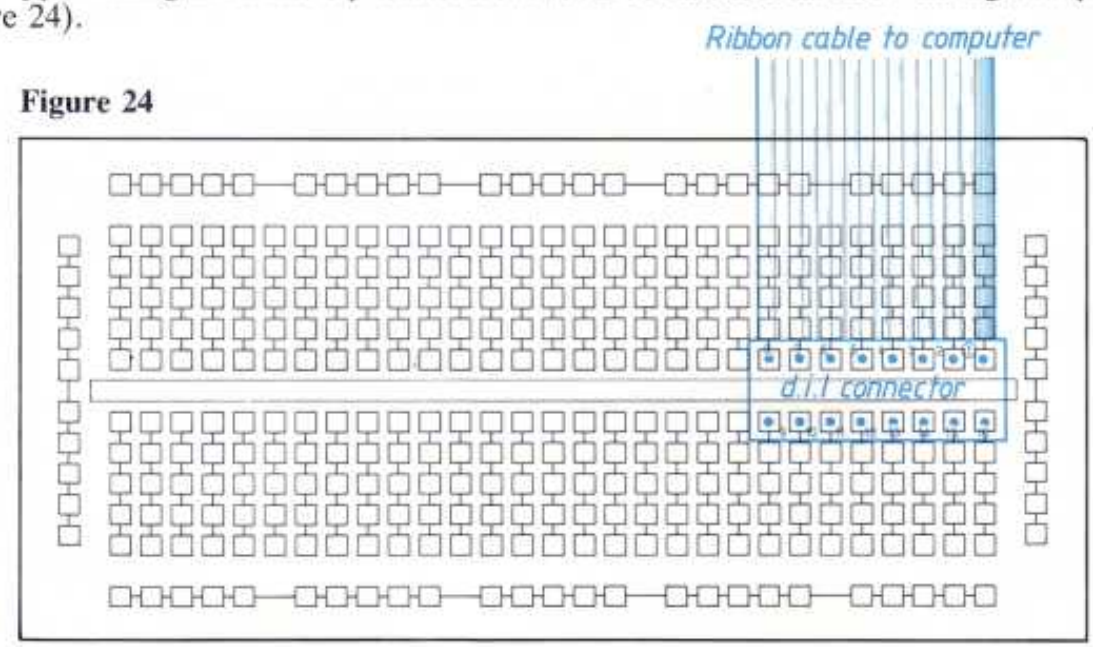
Figure 23



Do not plug the ribbon cable into the computer yet.

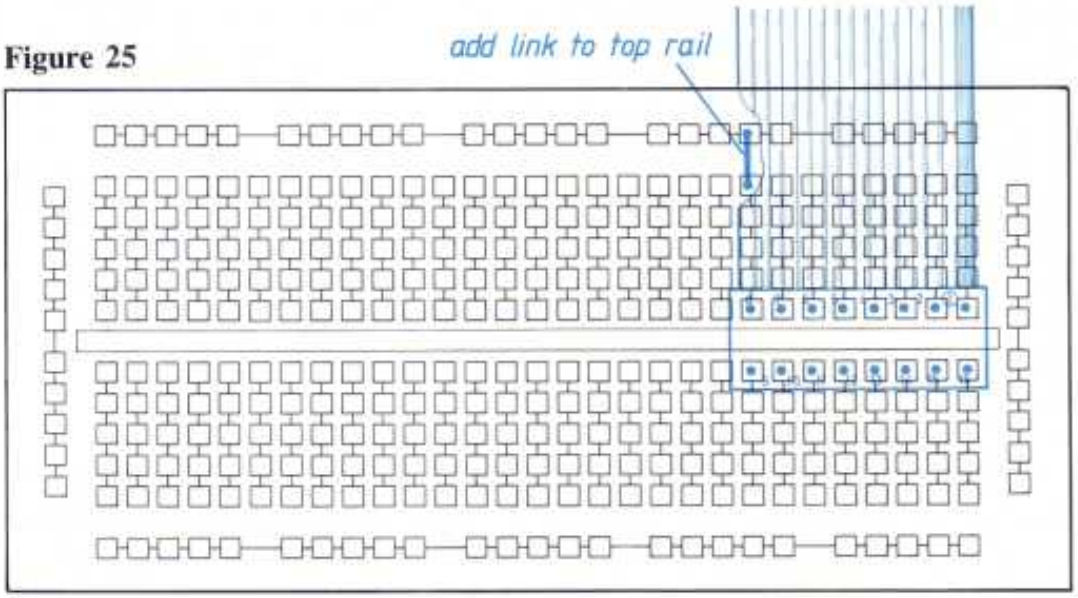
Plug the d.i.l. connector into the breadboard so that it bridges the central area with the cable approaching from the top of the board, and so that it is as far to the right as possible (figure 24).

Figure 24



All the pins of the d.i.l. connector above the central area are connected to the computer's ground via the ribbon cable. Join one of these pins to the top rail by putting a short piece of equipment wire into another hole in one of the rows and into a hole on the top rail (figure 25).

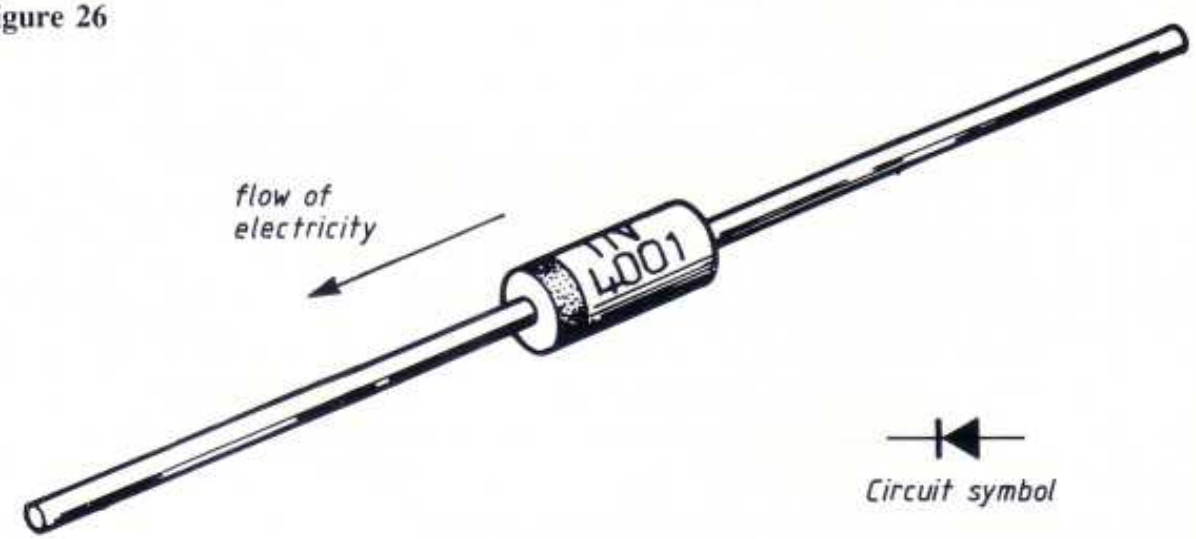
Figure 25



The pins below the central area are connected to the computer's switchable lines. We are going to use the first four lines only in these projects.

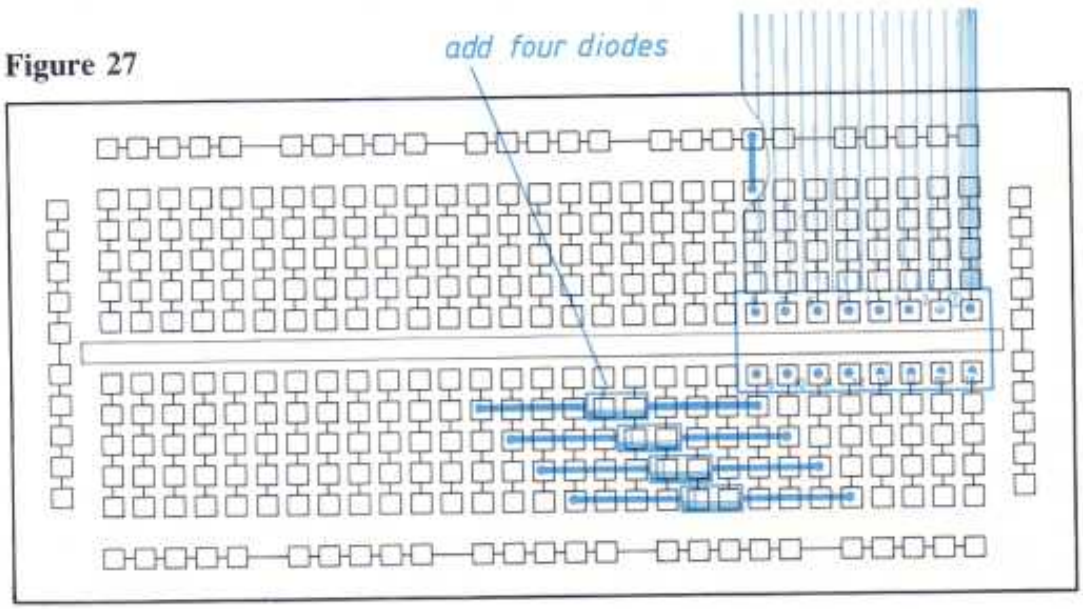
A **diode** will be attached to each of these lines. A diode is a component which allows electricity to flow in one direction but not in the other (figure 26). One end of the diode will be marked by a band. The electricity can flow through the diode from the unmarked end to the marked end but not in the other direction. As we do not want to let any electricity flow back to the computer the diodes must be fitted with the unmarked end nearest to the d.i.l. connector.

Figure 26



The diodes will take the signals from the rows into which the d.i.l. connector is fitted to rows nine places further to the left. Figure 27 shows the diodes in place.

Figure 27



Check your finished interface very carefully to make sure it is the same as figure 28.

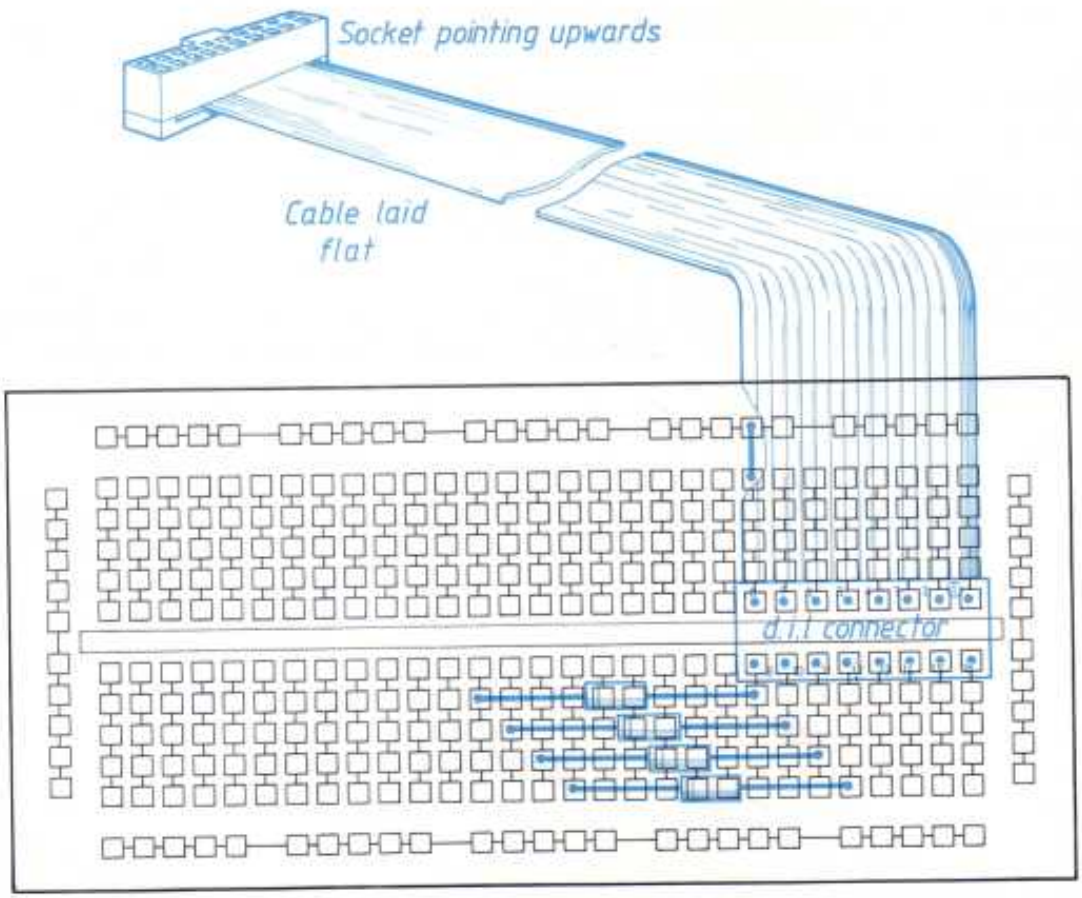


Figure 28

When you are satisfied all is correct go on to try project 7.

Project 7

TESTING THE USER PORT INTERFACE - A SOUND AND LIGHT METRONOME

In this project the User Port interface is tested by using the computer to light an LED. The method of programming the computer to switch the lines from the User Port is also introduced.

THE EQUIPMENT

You will need -

the User Port interface
a 470 ohm resistor coloured yellow, violet, brown
a Light Emitting Diode (LED)

Connect the 470 ohm resistor across the central channel between diode connected to line 1 and the row of contacts on the other side (figure 29).

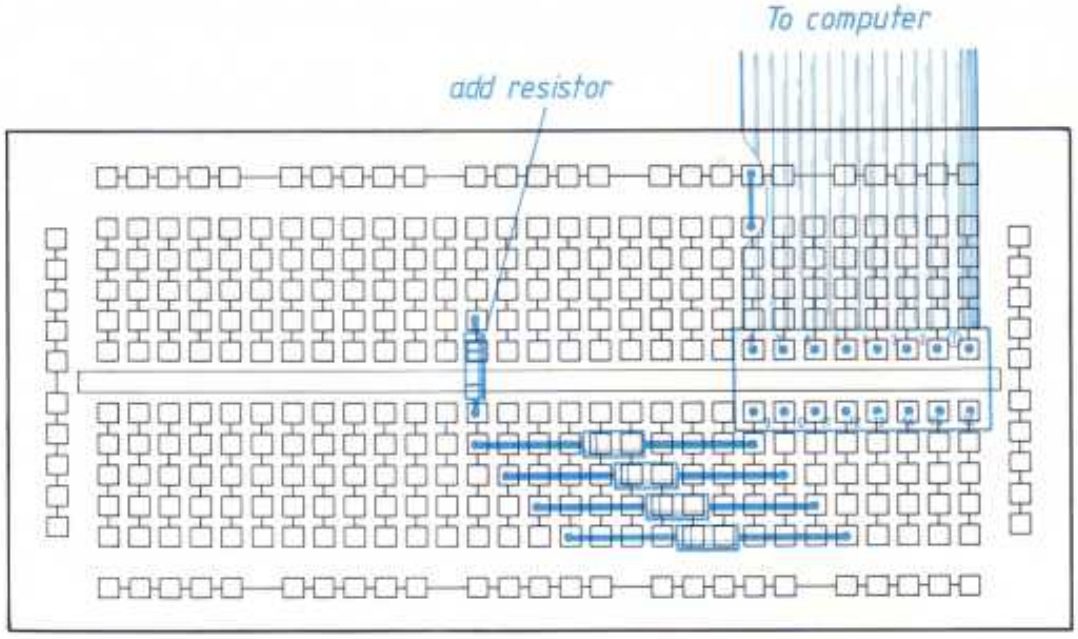
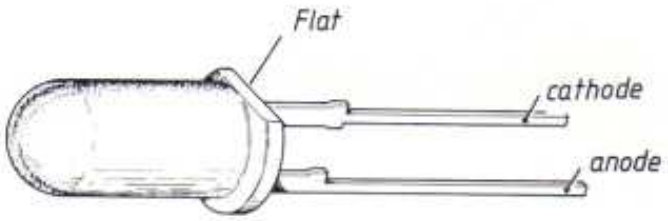


Figure 29

A **LED (Light Emitting Diode)** is similar to a diode in that it only allows electricity to flow in one direction but in doing so light is produced. A LED must always have a resistor in series with it - in this case the resistor which you have already fitted. The LED has one side which is marked by a flat edge which is the same as the mark on the end of a diode. The leg of the LED on the flat side is called the **cathode** (figure 30).

Figure 30



The electricity is going to flow from the resistor through the LED and to the top rail which is ground, so the LED must be fitted with the cathode leg in the top rail and the other leg (called the **anode**) in the same row as the resistor (figure 31).

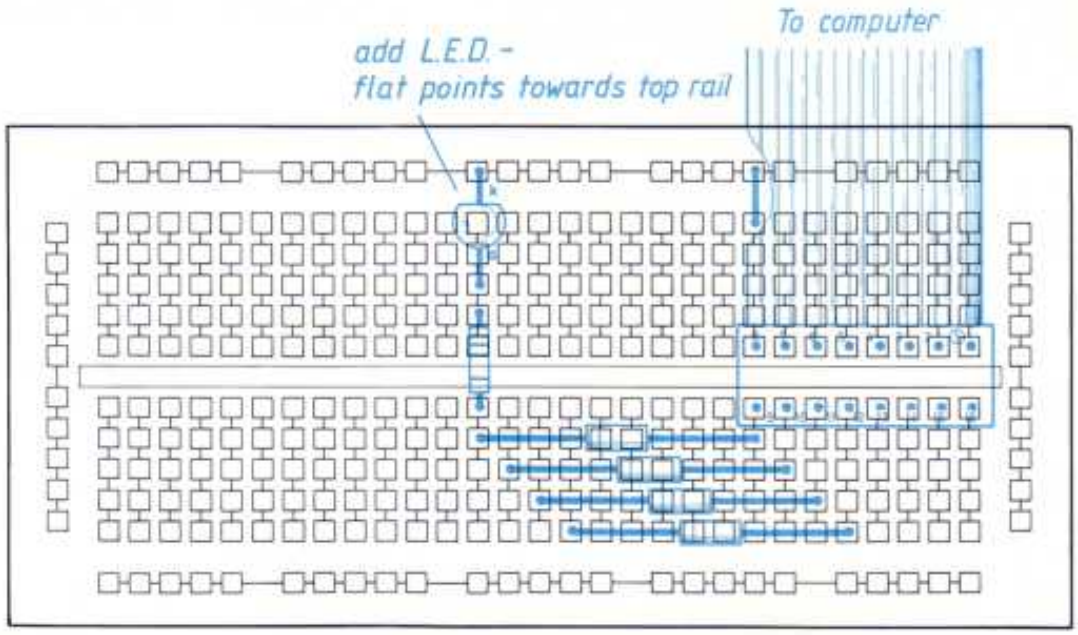


Figure 31

Check your equipment very carefully and make sure it is exactly the same as figure 31 before going on.

Before we plug the ribbon cable into the User Port we must tell the computer that we are going to use the lines from the User Port as outputs rather than inputs - i.e. we are going to send messages to the interface. One of the registers in the 6522 VIA is called the **Data Direction Register (DDR)** and when this register contains 255 the *direction* is set so that all the lines are sending signals out.

Type **?&FE62=255** and push RETURN. The **?** means *put into memory at address* and **&FE62** is the address of the DDR. So **?&FE62=255** means *put into address &FE62 the value 255* or set the DDR for all the lines to be outputs.

Now plug the ribbon cable into the User Port.

The LED may or may not light because we haven't given a value to the register in the VIA which switches the lines on and off. This register is at address **&FE60** in the computer's memory so type **?&FE60=1** and push RETURN. The LED should light. Now type **?&FE60=0** and the LED should go out.

If the LED does not light go back and check the interface and the wiring of the ribbon cable very carefully. You will have to find out where you have made a mistake before continuing.

Typing **?&FE60=??** each time we want to switch on or off the light is a waste of the computer's power - the light could be controlled much more easily by an ordinary switch.

Program 7,1 sets the DDR to output then starts an endless loop which switches line 1 on, waits, switches line 1 off, waits - and then repeats the sequence. The wait period is set by the **INKEY** value.

Program 7,1:

```
10 ?&FE62=255:REM Set DDR to output
20
30 REPEAT
40 ?&FE60=1:REM Line 1 on
50 I=INKEY(20)
60 ?&FE60=0:REM Lines off
70 I=INKEY(20)
80 UNTIL FALSE
```

Program 7,1 switches the LED on and off at regular intervals. Program 7,2 however waits until the key *0* or *1* is pressed before switching off or on the light. Pushing key *0* switches off the light and pushing key *1* switches it on. ***FX15,1** in line 80 clears the keyboard buffer in case a key has been held down.

Program 7,2:

```
10 ?&FE62=255:REM Set DDR to output
20 ?&FE60=0:REM All lines off
30
40 REPEAT
50 I$=INKEY$(5)
60 IF I$="1" ?&FE60=1
70 IF I$="0" ?&FE60=0
80 *FX15,1
90 UNTIL FALSE
```

A SOUND/LIGHT METRONOME

Using **?&FE60=0** and **?&FE60=1** the computer can switch on or off the light at regular intervals and so become a metronome. Program 7,3 allows you to change the speed of the metronome by pushing *F* for faster and *S* for slower.

Line 40 addresses the DDR to set all the User Port lines to output and line 50 turns all the lines off. **S%** is the variable which holds the delay between pulses and this is set to 50 when the program starts. Each unit of S% is worth one fiftieth of a second between pulses, so the start value of 50 is one pulse per second or sixty pulses a minute.

The endless loop of the program starts at line 80. Lines 90 to 110 check to see if *F* or *S* have been pushed and adjust the variable **S%** accordingly. Line 120 stops **S%** falling too low as at speeds above about 600 beats a minute it is not possible to see the light switching on and off.

Lines 140 and 150 switch on line 1 of the User Port and the computer's sound synthesiser. Line 160 waits for the length of time set by **S%** before continuing.

The LED is then turned off, line 170, and the sound is stopped by ***FX15,0** which flushes the keyboard and the sound buffers. The delay in line 190 is the same as the delay in line 160 so the computer is silent and the light is off for the same length of time as the sound and the light were on.

Program 7,3:

```
10 MODE 7
20 PRINT TAB(6,10);"Beats/min. = ";
30
40 ?&FE62=255:REM Set DDR to output
50 ?&FE60=0:REM All lines off
60 S%=50:REM S% holds the speed
70
80 REPEAT
90 I$=INKEY$(0)
100 IF I$="f" OR I$="F" S%=S%-2
```

```
110 IF I$="s" OR I$="S" S%=S%+2
120 IF S%<6 S%=6
130 PRINT TAB(19,10);INT(3000/S%);" ";
140 ?&FE60=1:REM Line 1 on
150 SOUND 1,-5,200,255
160 TIME=0:REPEAT UNTIL TIME>=S%
170 ?&FE60=0:REM All Lines off
180 *FX15,0
190 TIME=0:REPEAT UNTIL TIME>=S%
200 UNTIL FALSE
```

TESTING THE REST OF THE INTERFACE

By altering line 140 in Program 7.3 the other lines connected to the User Port can be switched on. However, the values for the first four lines are 1, 2, 4 and 8 and not 1, 2, 3 and 4 as might be expected (this is explained more fully in the next project).

Move the resistor and the LED one row to the right and alter line 140 of the program to read **?&FE60=2** and run the program again to test line 2 (figure 32).

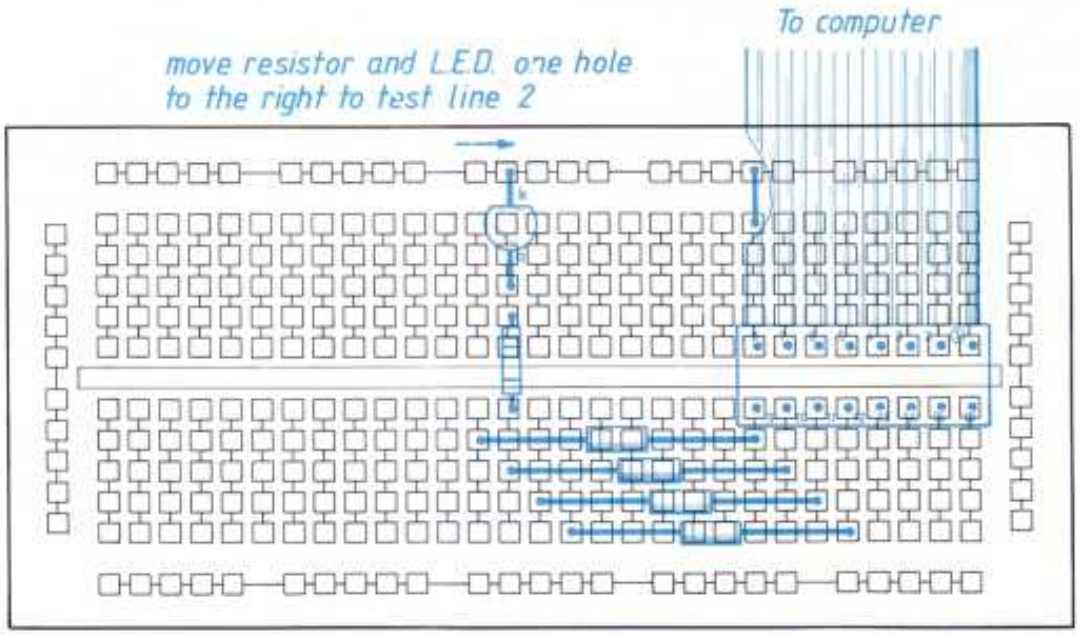


Figure 32

To test line 3 the LED and the resistor have to be moved to the right again and line 140 of the program altered to read **?&FE60=4**. Similarly move the components and alter line 140 to **?&FE60=8** to test line 4 from the User Port.

When you are sure that the four lines from the User Port are working correctly try Project 8 which uses three LEDs to make a set of traffic lights.

Project 8

TRAFFIC LIGHTS

Project 8 uses three LEDs to make a set of traffic lights. Programming techniques for the User Port are explained and the operator EOR is introduced to selectively switch the lines.

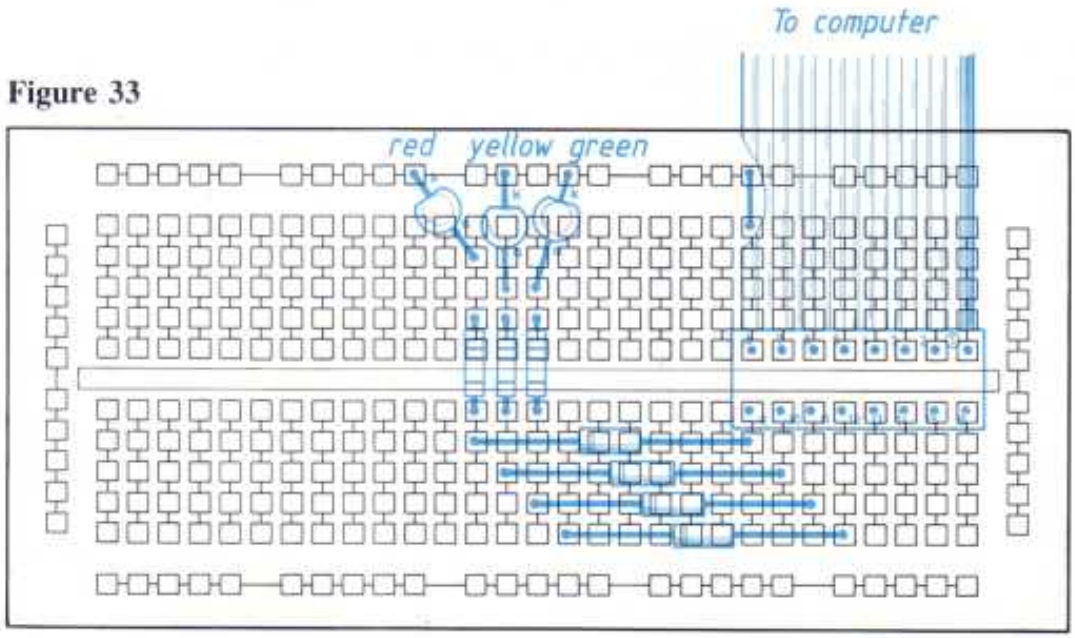
THE EQUIPMENT

You will need -

the User Port interface
three 470 ohm resistors (yellow, violet, brown)
four LEDs - if possible two red, one yellow and one green

Connect three resistors and three of the LEDs to the first three lines from the user port as shown in figure 33. If you have different coloured LEDs connect a red one to line 1, a yellow one to line 2 and a green one to line 3.

Figure 33



To test that they are all working type **?&FE62=255** and push RETURN to set the DDR to output, then type **?&FE60=7** and push RETURN to switch the three LEDs on. **?&FE60=0** will switch the LEDs off. If any of the LEDs do not work check that the *flat* on the LED is pointing towards the rail which carries ground. If the LED still does not light test it in a different row (LEDs, like light bulbs, do *go* although they should have a much longer life than an ordinary bulb).

Lines 60 and 70 in program 7,2 in the last project are only useful if one output line is being used. If more than one line is in use everytime the computer reads the instruction **?&FE60=0** all the lines will be switched off.

The computer has an *operator* which is written **EOR (Exclusive OR)** which can be used to alter a particular line in the register and leave the rest unchanged. Program 8,1 works in exactly the same way as program 7,2 but lines 60 and 70 of program 7,2 have been replaced with one line which uses **EOR** to *toggle* the light on line 1. When any key is pushed the signal on line 1 from the User Port becomes the opposite of its previous state - if it was off it is switched on, if it was on it is switched off.

Program 8,1:

```
10 ?&FE62=255:REM Set DDR to output
20 ?&FE60=0:REM All lines off
30
40 REPEAT
50 I$=INKEY$(5)
60 IF I$>"" ?&FE60=?&FE60 EOR 1
70 *FX15,1
80 UNTIL FALSE
```

An understanding of **EOR** and the values of each *bit* of an eight bit number helps when programming the user port.

Value in register	Line 1	Line 2	Line 3	Line 4
0	off	off	off	off
1	on	off	off	off
2	off	on	off	off
3	on	on	off	off
4	off	off	on	off
5	on	off	on	off
6	off	on	on	off
7	on	on	on	off
8	off	off	off	on
9	on	off	off	on
10	off	on	off	on
11	on	on	off	on
12	off	off	on	on
13	on	off	on	on
14	off	on	on	on
15	on	on	on	on

Figure 34

Each line is one bit of the eight bit register at address &FE60. The first line has a value of 1, the second line 2, the third line 4, the fourth line 8. So typing **?&FE60=2** will switch on line 2, typing **?&FE60=3** will switch on lines 1 and 2, typing **?&FE60=4** will switch on line 3 and so on (figure 34). The value of each line is double the value of the previous line so the series 1, 2, 4, 8, 16, 32, 64, 128 controls the eight lines.

When we want to alter the value of a particular line we do not want to affect any of the other lines. If lines 1, 2 and 3 are on the register will hold the value 7. To switch off line 2 we can use **7 EOR 2** which equals 5 - i.e. lines 1 and 3 on and line 2 off. If lines 1, 2, 3 and 4 are on and we want to switch off line 3 only we use **15 EOR 4** which equals 11.

In the same way **EOR** will switch on a line which is off. If lines 1 and 3 are on the register will hold the value 5, by writing **5 EOR 2** the register will hold 7 so lines 1, 2 and 3 will be on.

However, there is no need to write the first number because it will always be the present value of the register so it can be replaced with **?&FE60**. Therefore **?&FE60 EOR 1** will always make line 1 the opposite of its present state - if it's on it will turn off, if it's off it will turn on. In the same way **?&FE60 EOR 2** will alter line 2 to its opposite state, **?&FE60 EOR 4** will alter line 3 to its opposite state and **?&FE60 EOR 8** will alter line 4 in the same way.

If the effect of **EOR** is new to you try using it in the direct mode. Type **?&FE62=255** to set the DDR so that all the lines are outputs and plug in the interface. Type **?&FE60=7** to turn on the three LEDs and then experiment to find out how the state of a line changes using **?&FE60=?&FE60 EOR 'the line's value'**. In English this means '*make the register at address &FE60 equal what the register at &FE60 was Exclusive OR'd with the value of the chosen line*'. The **?** symbol both reads the value of the register and writes the new value to it.

All this may seem rather complicated but it is worth taking a few minutes to try to understand the effect of the operator EOR as it is very useful when controlling registers directly.

TRAFFIC LIGHTS

Program 8,2 turns on and off the three LEDs in the same sequence as a set of traffic lights. The starting point for the program is Amber before Red and line 20 sets the output register so that line 2 is switched on and the other lines are off.

The **FOR/NEXT** loop - lines 60 to 100 - operates the lights according the information read from the DATA statements in lines 130 to 160. The first item of data in each case is the value to be **EOR**ed with the user port register and the second item is the delay before the next change of the lights.

Figure 35 shows the sequence of values **EOR**ed with the register and the resulting effect on the state of the lines.

Old register value	EORed with Line 1		Line 2	Line 3	New register value
(Start)		off	on	off	2
2	3	on	off	off	1
1	2	on	on	off	3
3	7	off	off	on	4
4	6	off	on	off	2

Figure 35

Program 8,2:

```
10 ?&FE62=255:REM All lines output
20 ?&FE60=2:REM Amber on, Red and Green off
30
40 REPEAT
50 RESTORE 130
60 FOR C%=1 TO 4
70 READ Lines%,Wait%
80 ?&FE60=?&FE60 EOR Lines%
90 I%=INKEY(Wait%)
100 NEXT
110 UNTIL FALSE
120
130 DATA 3,500
140 DATA 2,175
150 DATA 7,500
160 DATA 6,175
```

MORE PROGRAMS TO TRY

Add another resistor to the diode attached to line 4 of the interface and another LED from the resistor to the ground rail (figure 36).

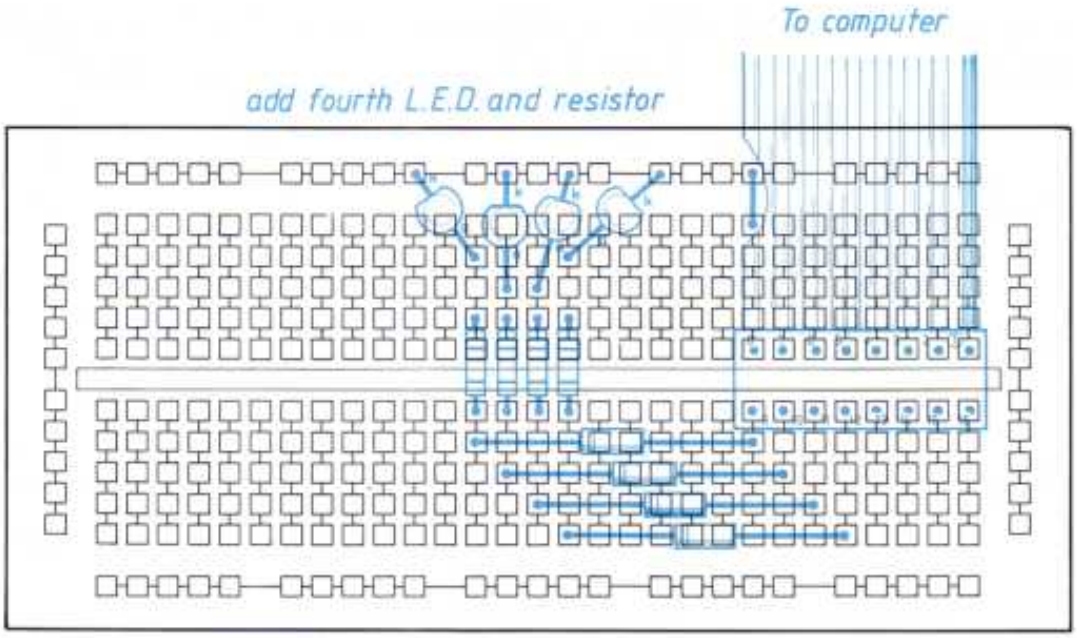


Figure 36

Try the following short programs. See if you can work out what will happen before you run them.

Program 8,3:

```
10 ?&FE62=255
20 ?&FE60=5
30 REPEAT
40 ?&FE60=?&FE60 EOR 15
50 I%=INKEY(30)
60 UNTIL FALSE
```

Program 8,4:

```
10 ?&FE62=255
20 ?&FE60=9
30 REPEAT
40 ?&FE60=?&FE60 EOR 15
50 I%=INKEY(30)
60 UNTIL FALSE
```

Program 8,5:

```
10 ?&FE62=255
20 ?&FE60=1
30 REPEAT
40 ?&FE60=?&FE60*2 MOD 15
50 I%=INKEY(7)
60 UNTIL FALSE
```

These programs can be used to experiment with the output lines by altering the values sent to address **&FE60** and watching the results. The more you experiment the more you will understand the workings of the eight bit register.

Finally try program 8,6 which displays the value of the register and the state of each line on the screen as the register is altered.

Program 8,6:

```
10 MODE 7
20 VDU 23;8202;0;0;0;
30 ?&FE62=255:REM All lines output
40 PROCScreen
50 Count%=0
60
70 REPEAT
80 ?&FE60=Count%
90 FOR A%=0 TO 3
100 Status%=(Count% DIV2^A%) AND 1
110 IF Status% PRINT TAB(A%*10,10);CHR$(129);" ON " ELSE PRINT
TAB(A%*10,10);CHR$(132);" OFF"
120 NEXT
130 PRINT TAB(25,18);Count%;" ";
140 Count%=(Count%+1) MOD16
150 REPEAT
160 I%=INKEY(60)
170 UNTIL I%=-1 OR I%=13
180 UNTIL FALSE
190
200 DEF PROCScreen
210 FOR A%=0 TO 3
220 PRINT TAB(A%*10,6);"Line ";A%+1;TAB(A%*10,7);"";
230 NEXT
240 PRINT TAB(7,18);"Output register = ";
250 PRINT TAB(0,23);CHR$(130);"Push RETURN to speed up sequence";
260 PRINT TAB(3,24);CHR$(130);"Push any other key to hold";
270 ENDPROC
```

Project 9

A MORE POWERFUL SWITCHING UNIT

Project 9 introduces one method of using the computer to switch on and off a piece of external equipment. A battery powered vehicle is driven at a variable speed controlled by the computer.

THE EQUIPMENT

You will need -

the User Port interface
a Darlington Driver Array integrated circuit (chip)
a sub-minature relay (6 to 12 volts)
about 20cm of equipment wire
a power supply unit producing between 6 and 12 volts
a length of two-core wire for the lead to the vehicle
a battery powered vehicle with an on off switch

Remove the LEDs and resistors used for project 8 but do not remove the four diodes. Do not plug the interface into the computer until you have added all the extra components and have checked your work to make sure it looks right. Try and get into the habit of unplugging equipment from the computer everytime you are going to work on it.

THE POWER SUPPLY

The electrical power available from the switchable lines from the User Port can do little more than light an LED or send a signal to another external Integrated Circuit (chip). To control any equipment which requires more than a few milliamps to drive it we must use an external power supply. But when we use an external power supply we must make sure that we are not going to send high voltages back into the computer as this would cause considerable damage.

The equipment used for this project and project 10 is not suitable for switching mains supply voltages.

The most convenient form of power supply is the type of transformer built onto a 13 amp mains plug. These are available for about £5.00 and offer a range of outputs (usually 3, 4.5, 6, 7.5, 9 and 12 volts) which are selected by a switch on the unit.

Alternatively you can use the type of power supply unit which is designed to run an electric train set or slot car racing game - however, if you are using a train set control unit, you must find out first which terminal is positive and which is negative and make sure you do not set the transformer to reverse.

Batteries can be used but the cost of replacing them when they run out will eventually be a great deal more than buying a suitable power supply unit.

9 volts DC is an ideal supply for these projects but any voltage between 6 volts DC and 12 volts DC can be used depending on the relay's requirements.

You may need to prepare the lead from the supply to make sure it fits securely into the breadboard. The lead may have a jack plug fitted to the end of it in which case you will have to cut if off and strip the ends of the wire. To make the lead stiff enough to push into the breadboard you can *tin* the wires with solder or, alternatively, join a length of equipment wire to each part by twisting the wires together and insulating them with plastic tape.

It is important to make sure that you know which part of the lead from the power supply is positive and which is negative, and to insert the wires into the breadboard the correct way round. The lead may be marked where it leaves the transformer or the wires themselves may be colour coded. When you are sure which part of the lead from the power supply is positive fix it into the left hand end rail. The negative part of the lead is fixed into the top rail which already contains the ground connection from the d.i.l. connector (figure 37).

Figure 37

THE EXTRA CHIP

The **Darlington Array** I.C. is used to switch the current from the power supply on and off. Although the interface already contains protection for the computer in the form of the diodes, the Darlington Array adds to that protection by removing the switching process one stage further from the computer.

Darlington Arrays are available with seven or eight stages and with or without **R-in**(input resistors). Because the space on the breadboard is limited the seven stage array is the most suitable. You should use a chip with R-in as this can be connected directly to the BBC micro's user port whereas a chip without R-in will need additional external resistors.

The chip actually contains the equivalent of 21 diodes, 14 transistors and 21 resistors which act as seven switches. When a signal is received from the computer the appropriate transistors in the chip act as a switch and the current flows. The chip actually works in an unexpected way because it *sinks* rather than *sources* the current.

The chip should be fitted into the breadboard so that it bridges the central gap and is as near to the d.i.l. connector as possible. One end of the chip will be marked with a cut-out or a dot above the first pin and this end should be furthest from the d.i.l. connector (figure 38).

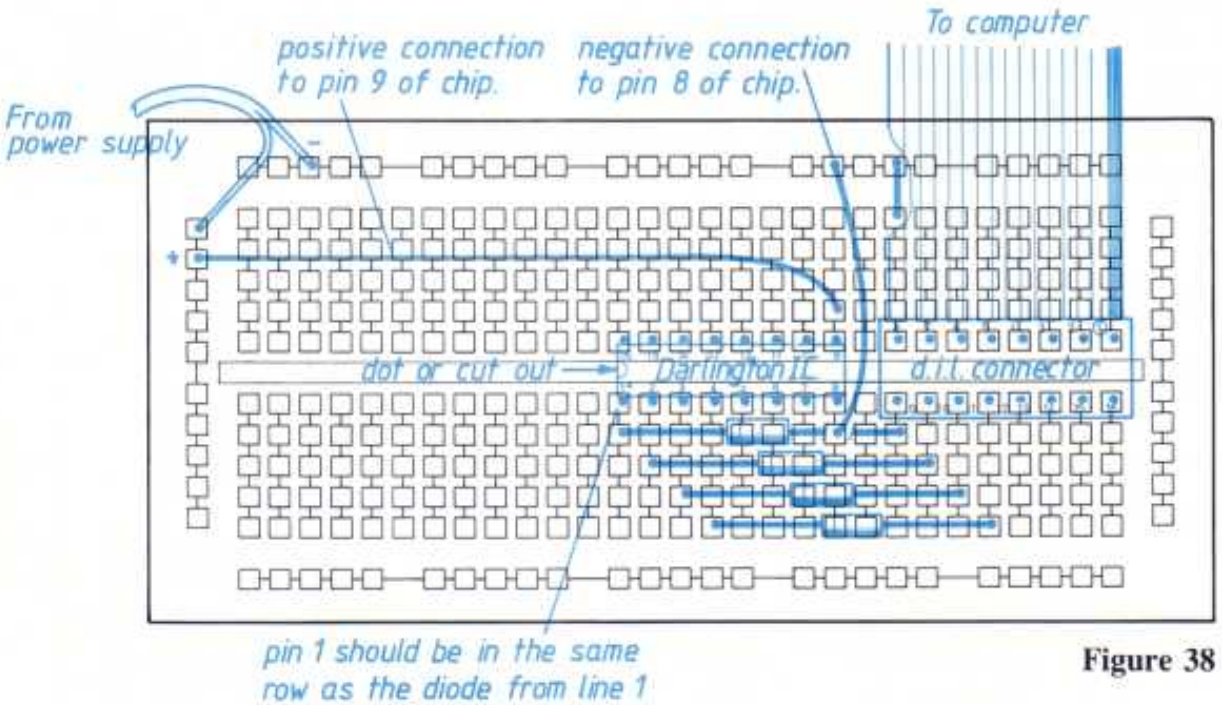


Figure 38

The diodes will already be in place to connect the first four stages of the array to the d.i.l. connector. The two other connections to the chip are positive and ground. Pin nine of the chip is connected to the positive rail at the end of the breadboard and the ground connection is made between pin 8 and the top rail. When connecting the equipment wire to pin eight of the chip you will have to be careful to keep the wire away from the diodes which are crossing over this row of contacts.

THE RELAY

A **Relay** is a switch which is controlled by an electrical signal. A mechanical relay contains a coil which becomes a magnet when a current flows through it causing two contacts to be pulled together. When the current ceases to flow the contacts part.

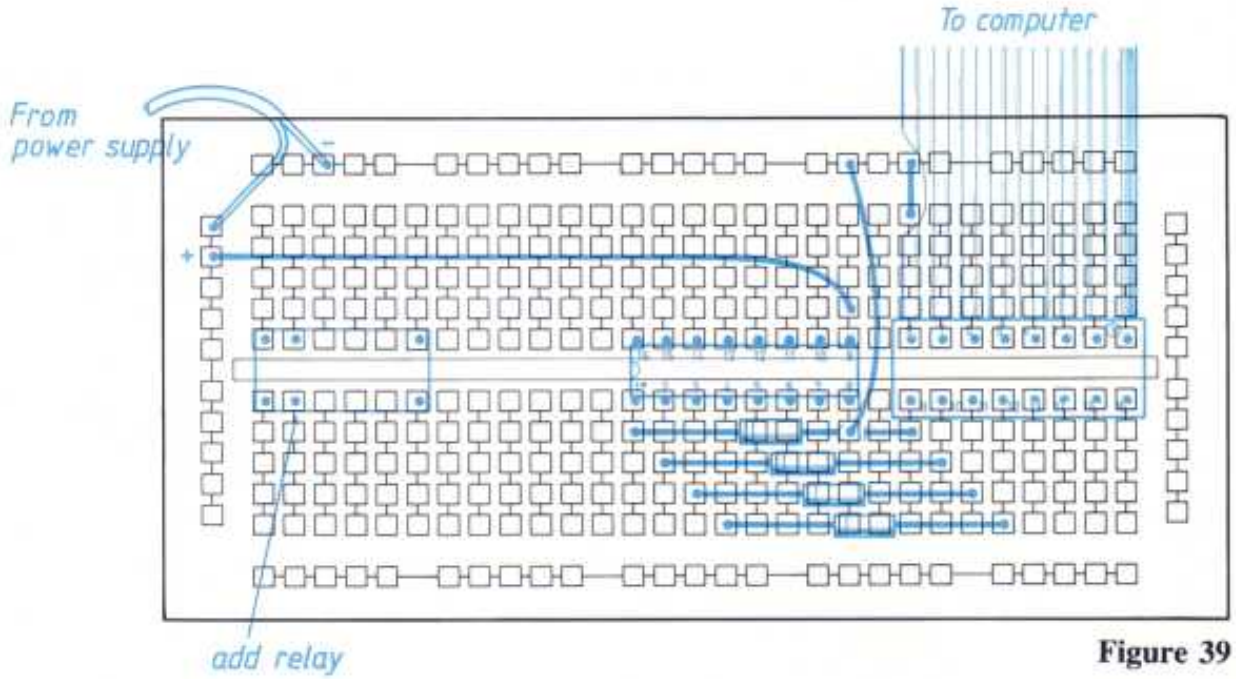


Figure 39

A small sized relay is needed because of the space available on the breadboard, and the pins on the relay must be suitably spaced for mounting onto a printed circuit board. The relay will plug in to the breadboard bridging the central channel (figure 39). Use a relay that has a transparent cover as this allows you to see the contacts moving when the coil is energised. The voltage required by the coil must be one of the voltages available from your power supply although it doesn't have to be exactly the same (a 12 volt relay can usually be switched by a 9 volt supply).

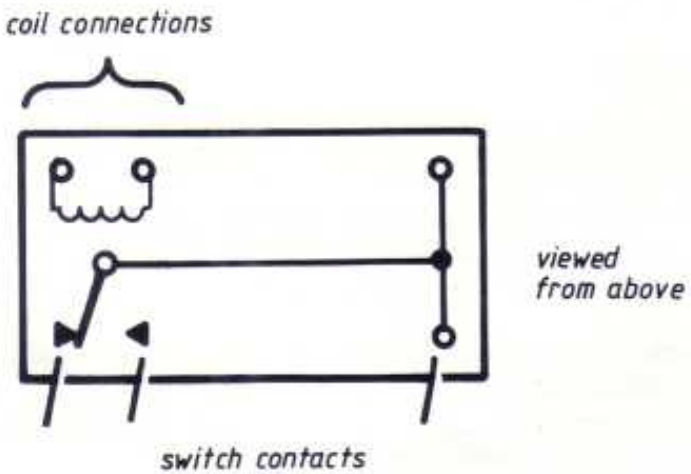


Figure 40

You will have to identify which two pins are connected to the coil (figure 40 shows the pins connected to the coil for the relay supplied in the kit available with the book). Connect one of these pins on the relay to the positive rail and the other pin on the relay to pin sixteen which is the first output of the Darlington Array (figure 41).

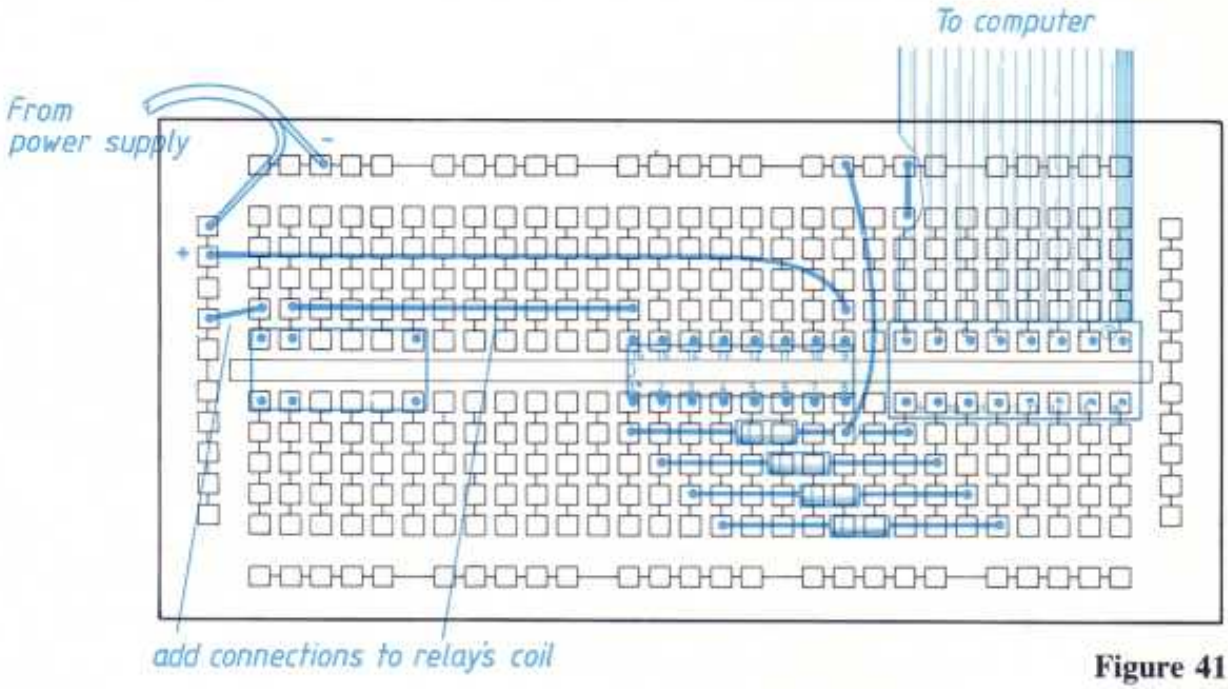


Figure 41

TRYING IT OUT

Check your equipment carefully against figure 41 then plug the ribbon cable into the computer, switch on the power supply and run program 9,1.

Program 9,1:

```
10 ?&FE62=255
20 ?&FE60=0
30 REPEAT
40 ?&FE60=?&FE60 EOR 1
50 I%=INKEY(20)
60 UNTIL FALSE
```

The relay should click open and shut until **ESCAPE** is pushed. Change line 50 to **I%=INKEY(5)** to see how fast the computer can switch on and off equipment (if the **INKEY** value is too small the relay may not be able to keep up with the computer).

If the relay does not work check the equipment very carefully and find out where you have made a mistake before continuing. Is the supply voltage suitable for the relay and is it correctly set on the transformer? Have you made all the connections to the chip? Are you sure which pins on the relay are connected to the coil?

Most battery powered vehicles have an on/off switch but the toy will probably have to be taken to bits to get to the connections. We are going to replace the switch with the relay so that the toy can be switched on and off by the computer. The lead from the relay to the vehicle will have to be quite long to allow adequate movement. Don't worry about the speed of the vehicle as the computer will be controlling it.

A bit of ingenuity may be needed to connect the two wires to the switch and to secure it on the vehicle. It is not necessary to disconnect the existing wires to the switch - just make sure that the switch is in the off position (figure 42).

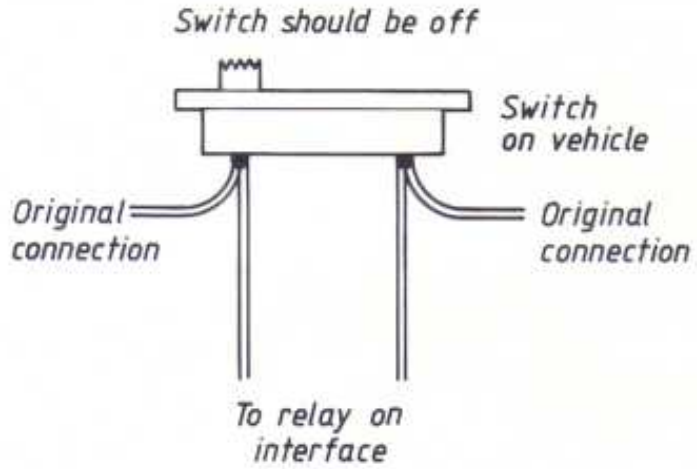


Figure 42

At the other end the lead is connected to the two pins on the relay which are joined when the relay is switched on. By examining the relay you should be able to find out which pins are which (figure 43 shows the connections for the relay supplied in the kit).

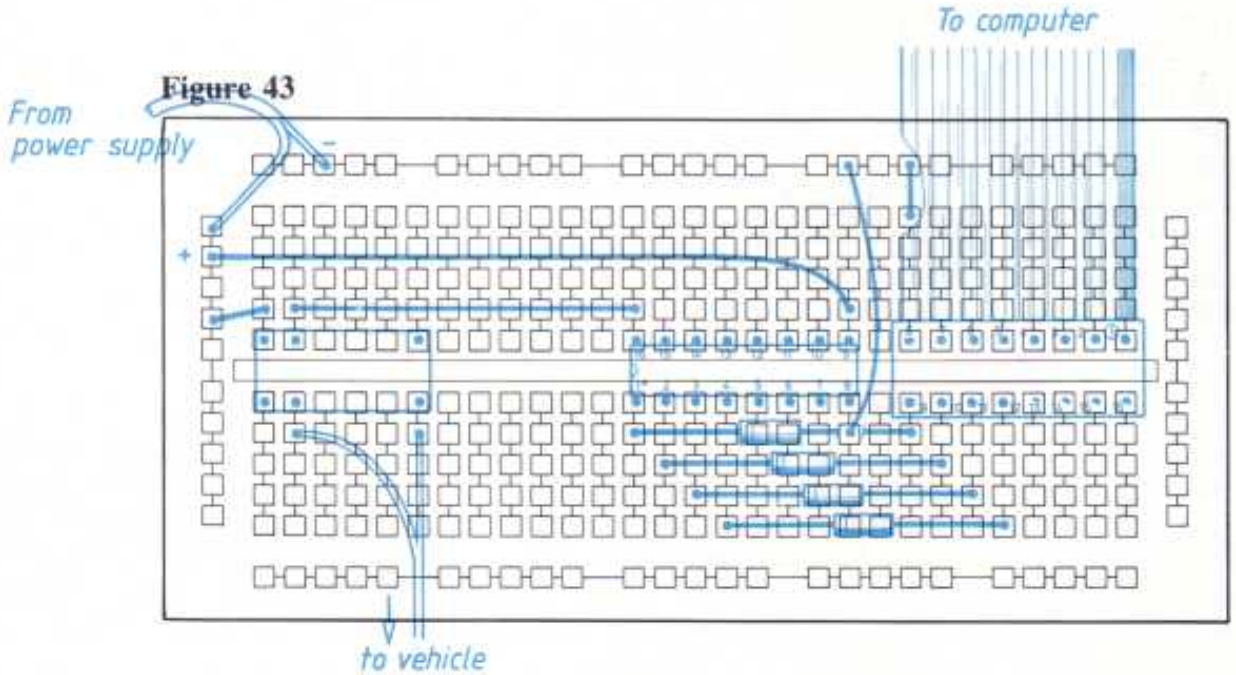


Figure 43

The relay will replace the switch on the vehicle so the moment the computer sends a signal down line 1 the vehicle will be turned on. If the signal is continuous the vehicle will move at full speed but by using the computer to switch on and off the relay very quickly the speed of the vehicle can be controlled.

The variable **speed%** in program 9,2 holds the number of cycles the computer will wait when the relay is turned off (line 150). It will always be a nominal value between 0 and 1000. If the value is high the vehicle will move slowly because there will be a lengthy pause between each burst of power. If the value of **speed%** equals 0 the relay will be switched on almost all the time so the vehicle will move at its maximum speed. The starting value of **speed%** in line 30 can be adjusted to suit the vehicle.

Line 130 allows the **SPACE BAR** to be pressed to stop the vehicle and pressed again to restart it.

Program 9,2:

```
 10 ?&FE62=255 : REM All lines output
 20 ?&FE60=0 : REM All lines off
 30 speed%=500
 40 MODE 1
 50 VDU 23;8202;0;0;0; : REM Cursor off
 60 PROCscreen
 70
 80 REPEAT
 90 I$=INKEY$(0)
100 *FX15,1
110 IF (I$="S" OR I$="s") AND speed%<990 speed%=speed%+10
120 IF (I$="F" OR I$="f") AND speed%>0 speed%=speed%-10
130 IF I$=" " ?&FE60=0:REPEAT:I$=INKEY$(10):UNTIL I$=" "
140 ?&FE60=0 : REM Turn off and wait
150 FOR wait%=1 TO speed%:NEXT
160 ?&FE60=1 : REM Turn back on
170 PRINT TAB(20,10);1000-speed%;" ";
180 UNTIL FALSE
190
200 DEF PROCscreen
210 PRINT TAB(12,10);"Speed = ";
220 PRINT TAB(13,24);"F = faster"
230 PRINT TAB(13,26);"S = slower"
240 PRINT TAB(5,28);"SPACE BAR = stop and start"
250 ENDPROC
```

A word of warning about small motors. The motors fitted in toys are often of poor quality and can produce high frequency spikes which can cause interference to the computer's T.V. or monitor and in the worst cases can cause the computer to *crash*. If the interference is too bad you will either have to use a longer lead to move the vehicle further from the computer or find a different toy to use. You can test the toy before you connect it to the interface by running it near to the computer to find out if it will cause interference.

Project 10

A COMPUTER CONTROLLED SLOT CAR SET

Project 10 sets up an adaptable program which allows the User Port to be programmed using simple statements. The program runs two cars on a slot car track but may be adapted to handle all the eight lines available from the user port.

THE EQUIPMENT

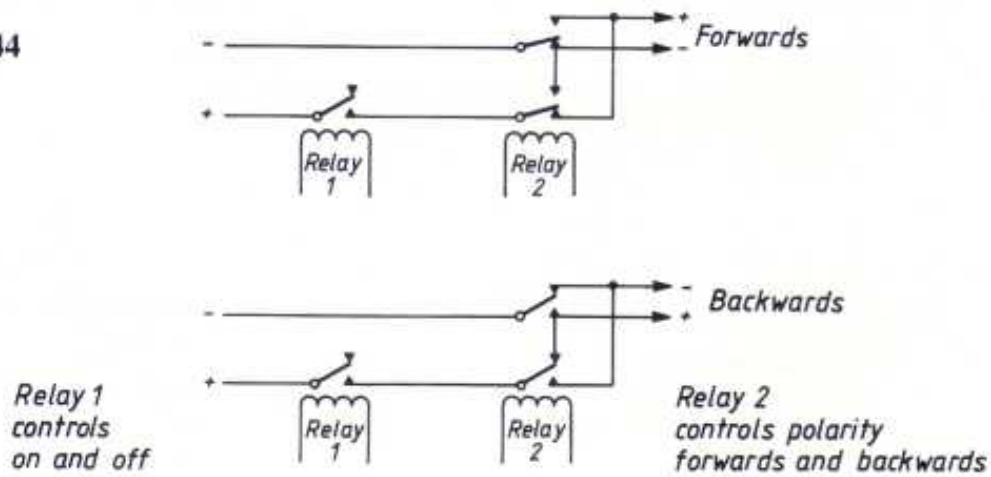
If you have followed the principles introduced in the previous projects you should be able to adapt your system to control any low voltage equipment.

The layout used for one relay can be repeated for however many relays are required.

There are eight lines available from the user port and you can expand the interface to use seven of them with the Darlington Array. Relays take up rather a lot of space and you will need to use extra pieces of breadboard if you are going to use seven of them. *Verobloc* has lugs to allow boards to be joined together so that the interface remains as a single unit. You may be tempted to try your hand at creating a permanent interface by soldering the components onto stripboard but, unless the interface is going to be dedicated to one task, you will lose the option of changing the components to suit different needs.

The use of a Darlington Array and a relay is really duplicating the switching process but it has two distinct advantages. Firstly the relay contacts are completely remote from the computer so no damage can be caused if the external equipment fails in any way. Secondly, the relays can be seen to move so you have a visual check that they are working - solid state relays are available but it is impossible to see them operating.

Figure 44



The relay used in the projects has operated a single pair of contacts but dual throw relays are available which alter two sets of contacts each time the relay is energised. Figure 44 shows how one single throw and one dual throw relay can be used to control forward and reverse as well as stop and go.

There were two power supplies in the last project - the power supply to drive the chip and the relay, and the batteries to drive the vehicle. It would be possible to use the power supply for the chip and coil to drive the vehicle as well. However, the relay offers extra protection for the computer because the external power for the equipment is routed through the contacts of the relay only and is therefore completely isolated from the computer.

Figure 45 shows two relays connected to a slot car racing set to allow the computer to control two cars. The power for the cars is supplied by the slot car set's transformer and is isolated by the relays from the rest of the interface.

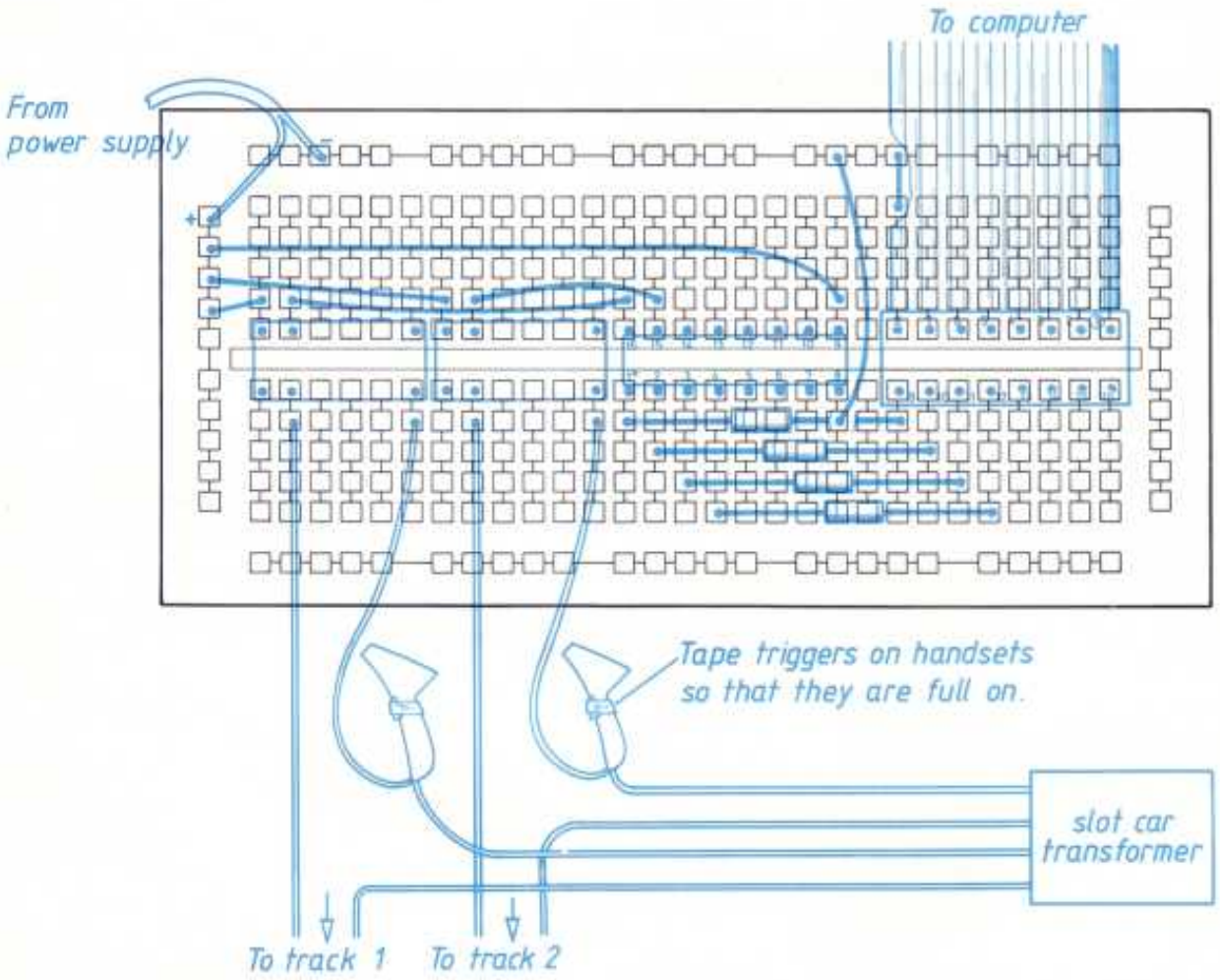


Figure 45

PROGRAMMING TO ALLOW PROGRAMMING

If the system is going to make use of the computer it must be able to store a series of instructions to control the external equipment, otherwise it is only a method of transferring the switches from the equipment to the computer keyboard. If the instructions were incorporated in a *BASIC* program, the program or the data statements would have to be altered each time we wanted to give the cars some new commands.

Program 10 is really a program which creates a new language for the computer. The language has a very limited vocabulary but it allows you to type a series of commands in plain English for the computer to follow and it allows you to delete and add commands and save the resulting sequence on tape or disc.

The computer's new vocabulary is -

1 go
1 stop
2 go
2 stop
wait ? (where ? represents tenths of a second)

Typing *wait 20* tells the computer to wait for 20 tenths of a second (2 seconds) before it acts on the next instruction - so whatever was happening before the wait instruction will continue to happen until two seconds have passed. The computer can remember up to sixty instructions which are held in the array **A$**. More instructions could be remembered if the size of the array was increased but this would produce screen display problems when the commands were being edited.

If the program is used with the slot car equipment (figure 45) the following sequence will make car 1 move, followed by car 2 five seconds later. Both cars will continue to move for another six seconds after which car 2 will stop, and one second later car 1 will stop.

1 go
wait 50
2 go
wait 60
2 stop
wait 10
1 stop

All words must be enterered in lower case letters (the Caps Lock and the Shift Lock are turned off by ***FX202,48** in line 390) and the number must follow the *wait* instruction. However, the order of the *stop* and *go* commands is not important - *stop 1* will produce the same reaction as *1 stop*.

Program 10:

```
10 MODE1
20 VDU23;8202;0;0;0;
30 ?&FE62=255
```

```
40 DIM A$(59)
50 count%=0
60 ON ERROR GOTO 100
70
80
90
100 REPEAT
110 CLS
120 PRINTTAB(0,6);"1 ... run program"''"2 ... edit program"
130 PRINT'"3 ... save instructions"''"4 ... load instructions"
140 PRINT'"5 ... end"''"Push appropriate number key ... ";
150 REPEAT
160 G%=GET
170 G%=G%-48
180 UNTIL G%>0 AND G%<6
190 CLS
200 IF G%=1 PROCact
210 IF G%=2 PROCedit
220 IF G%=3 PROCsave
230 IF G%=4 PROCload
240 UNTIL G%=5
250 CLS:END
260
270 DEF PROClist(P%)
280 PRINTTAB(13*(P% DIV20),(P% MOD20)+2);P%+10;" ";A$(P%)
290 ENDPROC
300
310 DEF PROCedit
320 REPEAT
330 FOR C%=0 TO count%-1
340 PROClist(C%)
350 NEXT
360 PRINTTAB(0,29);"Enter number only to delete"
370 PRINT"Enter number and instruction to insert"'"Push ESCAPE to
finish";
380 PRINTTAB(0,26);"";
390 *FX202,48
400 INPUT"* "S$
410 Z%=VAL(LEFT$(S$,2))-10
420 N$=RIGHT$(S$,LEN(S$)-2)
430 REPEAT
440 IF LEFT$(N$,1)=" " N$=RIGHT$(N$,LEN(N$)-1)
450 UNTIL LEFT$(N$,1)<>" "
460 IF N$="" AND Z%<count% A$(Z%)="":FOR X%=Z% TO
count%:A$(X%)=A$(X%+1):NEXT:count%=count%-1
470 IF N$>"" AND Z%<count% FOR X%=count% TO Z% STEP-
1:A$(X%+1)=A$(X%):NEXT:A$(Z%)=N$:count%=count%+1
480 IF N$>"" AND Z%>=count% A$(count%)=N$:count%=count%+1
490 CLS
500 UNTIL count%>59
```

```
510 ENDPROC
520
530 DEFPROCact
540 ?&FE60=0
550 IF count%=0 ENDPROC
560 FOR C%=0 TO count%-1
570 PROClist(C%)
580 T$=A$(C%)
590 IF INSTR(T$,"wait")>0 T%=VAL(RIGHT$(T$,LEN(T$)-
4)):TIME=0:REPEAT:UNTIL TIME>T%*10
600 IF INSTR(T$,"stop")>0 PROCstop
610 IF INSTR(T$,"go")>0 PROCgo
620 NEXT
630 ENDPROC
640
650 DEF PROCstop
660 IF INSTR(T$,"2")>0 AND ((?&FE60 DIV2) AND1)=1 ?&FE60=?&FE60
EOR 2
670 IF INSTR(T$,"1")>0 AND (?&FE60 AND1)=1 ?&FE60=?&FE60 EOR 1
680 ENDPROC
690
700 DEF PROCgo
710 IF INSTR(T$,"2")>0 AND ((?&FE60 DIV2) AND1)=0 ?&FE60=?&FE60
EOR 2
720 IF INSTR(T$,"1")>0 AND (?&FE60 AND1)=0 ?&FE60=?&FE60 EOR 1
730 ENDPROC
740
750 DEF PROCsave
760 PRINT"":INPUT"Filename",F$
770 X=OPENOUT(F$)
780 PRINT£X,count%
790 FOR C%=0 TO count%-1
800 PRINT£X,A$(C%)
810 NEXT
820 CLOSE£X
830 ENDPROC
840
850 DEF PROCload
860 PRINT"":INPUT"Filename",F$
870 X=OPENIN(F$)
880 INPUT£X,count%
890 FOR C%=0 TO count%-1
900 INPUT£X,A$(C%)
910 NEXT
920 CLOSE£X
930 ENDPROC
```

Because the program uses the **INSTR** function the key words can easily be changed to suit a particular application. If you would prefer to use 'off' and 'on' instead of 'stop' and 'go' the only two lines which need changing are 600 and 610.

In the same way, more lines from the user port can be used by adding to the procedures **PROCstop** and **PROCgo**. If you want to use line three as well add -

```
655 IF INSTR(T$,"3")>0 AND ((?&FE60 DIV4) AND1)=1 ?&FE60=?&FE60
EOR 4
```

and -

```
705 IF INSTR(T$,"3")>0 AND ((?&FE60 DIV4) AND1)=0 ?&FE60=?&FE60
EOR 4
```

Notice that the **INSTR** function is looking for the line's number (3) but the number after **DIV** and after **EOR** must be the line's value - i.e. 4. If line 4 is in use as well the **INSTR** function will look for 4 but the real *value* of the line will be 8.

A large part of the program, **PROCedit**, is concerned with editing the instructions. When *edit* is selected a new instruction can be added at the end of the list by typing the next number in the sequence and then the command i.e. *14 2 stop* or *23 wait 60* (line 480). If an instruction is preceeded by a number which is already in use the new instruction is inserted and the original instruction and the others with higher numbers are shuffled along to make room (line 470). If a number only is entered that instruction is deleted and the rest of the instructions are shuffled back to fill up the gap (line 460). Lines 430 to 460 remove any leading spaces as these will affect the *wait* instruction.

PROClist formats the instructions into three columns on the screen and is used by both **PROCedit** and **PROCact**.

PROCact works through the instructions stopping at line 590 for the required length of time if the **INSTR** fuction finds *wait*. If *stop* or *go* (or any other chosen words) are found **PROCstop** or **PROCgo** are called.

The expressions in **PROCstop** and **PROCgo** which end in **AND1** are used to find out if the particular line is on or off. If *2 stop* was entered when line two was already off it would be turned on by **EOR 2**. By using **IF (?&FE60 DIV2) AND1=?** the computer can make sure that the line really does need switching before using **EOR 2**.

PROCsave and **PROCload** handle the file routines. The program is not fully error trapped and some wrong inputs will cause an error to be produced. Line 60 ensures that the program will re-start from line 100 in the case of a wrong input, so any instructions already entered will not be lost. Pushing **ESCAPE** will always return the program to the menu.

AND ...

Because the program is adaptable it can form the basis of many control experiments. Other procedures can be added for graphics or sound (can the BBC micro produce the sound of a Formula 1 racing car?) without altering the structure of the existing program. It is just a matter of deciding where in the program to call the new procedures.

Keep a safe copy of the original program and use it whenever you are going to set up a new control experiment.

AND NOW TRY

DESIGNING A SYSTEM

To control any equipment you must first decide what needs to be switched on and off to produce the required effect. That may seem obvious, but often the sequence and the number of relays required are not so obvious.

Imagine an electric train set with two sets of motorized points, two sets of lights and an engine. At first glance it may seem that only five relays are needed. But the engine must be able to go forwards and backwards and the points also have two directions in which they can travel. By the time the system is built, eight relays will have been used.

The design for the interface will be -

relay 1:Switch power to the tracks on and off
relay 2:Reverse power for backwards movement
relay 3:Switch lights 1 - red or green
relay 4:Switch lights 2 - red or green
relay 5:Set direction for point 1 - open or closed
relay 6:Switch on and off power to send pulse to point moto- 1
relay 7:Set direction for point 2 - open or closed
relay 8:Switch on and off power to send pulse to point motor 2

Relay 2 will need to be a double throw type to control the direction of the engine. The Darlington Array used for the projects had only seven stages but in this design eight stages are needed. There are octal (eight stage) darlington arrays available - alternatively both sets of lights can be switched by one relay so that when one set of lights is green the other is red and vice versa.

The points need two relays each because the point motors only require a short pulse to make them work. If the power is left on they will burn themselves out. As well as the relay to provide the burst of power another relay must select whether the points are opened or closed.

The relays' actual functions will be -

Relay	Open	Closed
1	no power to track	power to track
2	track polarity normal	track polarity reversed
3	power to red light 1	power to green light 1
4	power to red light 2	power to green light 2
5	connect relay 6 to open points	connect relay 6 to close points
6	no power to point motor 1	power to point motor 1
7	connect relay 8 to open points	connect relay 8 to close points
8	no power to point motor 2	power to point motor 2

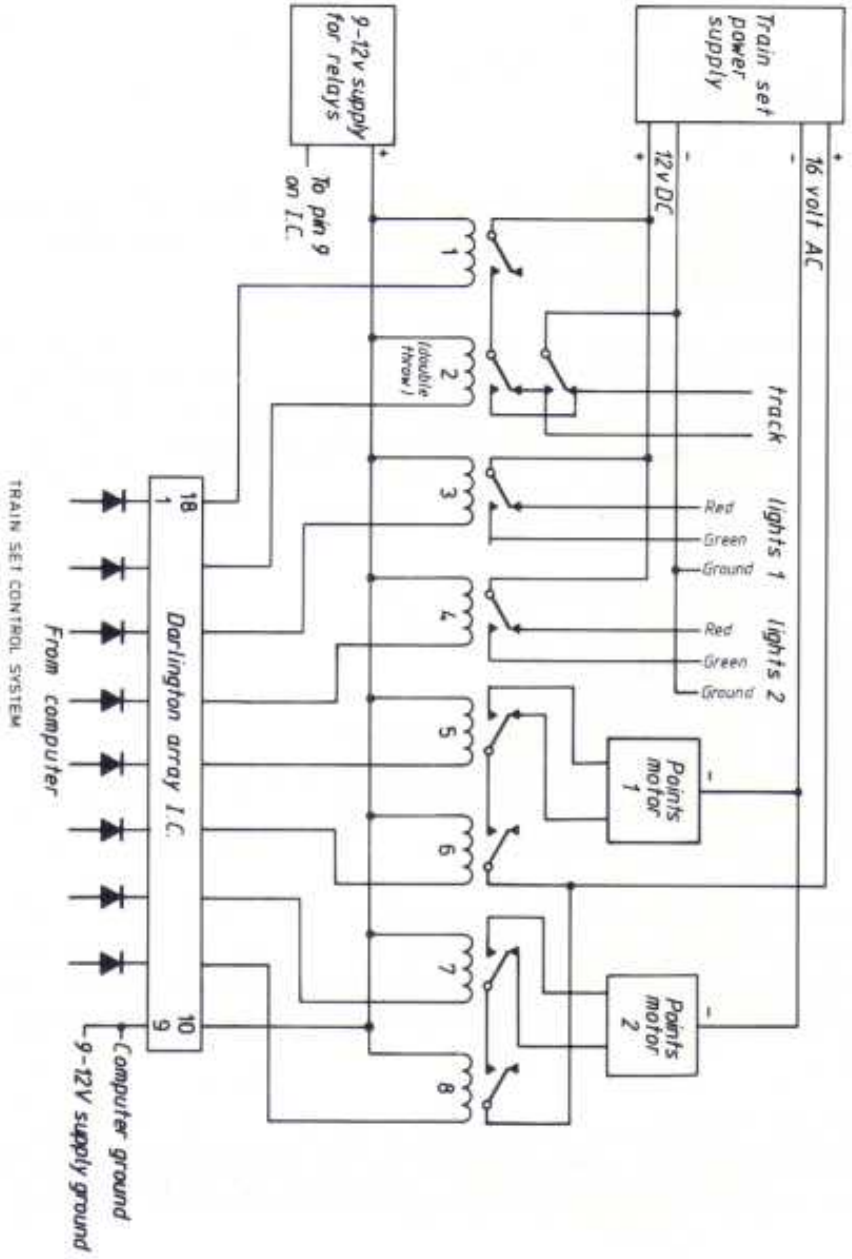


Figure 46

The power supply can also be chosen to suit each task. Most train set transformers have a 16 volt AC outlet to power point motors. So relays 6 and 8 can switch this power through relays 5 and 7 while the other relays switch the 12 volt DC supply to the track and the lights.

Figure 46 shows the system in diagramatic form. The program to control relays 1 to 4 is quite straight forward - each will be either on or off according to the current instruction. The relays which control the points will work slightly differently - if point 1 is to be moved the program must first set relay 5 in the right direction, then it must close relay 6 to send power to motor 1, wait for about a tenth of a second, and then open relay 6 to switch off the power. In the same way the second set of points will be controlled by relays 7 and 8.

BUT WHERE IS THE TRAIN?

Computer's are not *clever* - they just work so much faster than human beings that they seem to be clever. We can use the speed of the computer to set it extra tasks which it can carry out, while it is controlling the train, without the delay being noticeable.

The second A to D project used a LDR to find out the speed of a vehicle. We do not want to measure the speed of the train but it would be useful to know where the train was on the track if, for example, it needs to stop at a station or to avoid wrongly set points.

If a LDR is positioned at the end of the station platform and connected to the A to D converter the **ADVAL** reading from it will change when the train passes. By checking the signal from the LDR the train can be made to stop in the right place every time.

A single line in the program may be enough - '**IF ADVAL(?)<>normal THEN turn off relay 1**'. Provided this line is evaluated regularly the computer can check when the train reaches the station.

AND THEN CAME THE ROBOT ...

As soon as the computer appears to be controlling equipment intelligently then *control* becomes *robotics*. When the computer stops the train because it has reached the station or because it is about to derail itself on the points then it appears to be acting with intelligence. Of course the intelligence is all contained in your instructions to the computer.

The ultimate train set control system will always change the points before the train reaches them or stop the train if a light is red. But, for the non-serious application, a stage is eventually reached when the computer is doing all the work and all you can do is to sit back and watch.

At that point it is probably better to move on and to try something else -

* build a lift or elevator system using motors and pullies (motors scrounged from old cassette players are recommended as they are of reasonable quality and they often have a pully wheel attached).

* buy some *LEGO* and some *LEGO* motors and see if you can build a *real robot* with LDRs for eyes and a tilt switch to aid balance. 'Legs' are very difficult to make so use wheels or tracks instead.

* try to build your own *turtle*. A real *turtle* uses stepper motors but a reasonable vehicle can be made using a DC motor for each wheel and LDRs for eyes. Can you used sensors to count the rotations of the wheels so that the computer can keep track of the vehicle's position?

It is beyond the scope of this book to mention all the facilities available from the user port. In particular, the handshake lines can be very useful when controlling external equipment and no mention has been made of the user port's input facility. That is left to the relevant sections in the *Advanced User Guide* and some of the more advanced projects published in the computer magazines (see Appendix 3 for further reading).

However, many of the more specialised tasks for the user port demand that a dedicated program is written to control the equipment and often the program needs to be written in assembly language to ensure that the computer can work at the speed demanded by the system.

For example, a high speed A to D converter can be connected which feeds signals into the port. The A to D converter and the computer can work very fast - fast enough to measure voice patterns or to decode weather satellite signals - but a program written in BASIC will not work quickly enough to display the signals so machine code is the only possible alternative.

There is a great deal of scope for experimenting with control and sensing without getting involved in complicated electronics or assembly language programming. If you can think of something which moves you can probably design a simple system to use the computer to control it.

Appendix 1

Buying the components

If you have no experience of buying electronic components it is strongly recommended that you purchase the kits available from **Greenweld Electronics Ltd** of **443 Millbrook Road, Southampton SO1 0HX.** Two kits are available, one contains all the parts needed for the A to D projects and the other contains all the parts for the User Port section - except the toy and the slot car track! There is nothing worse than completing a project only to find that it doesn't work because one of the items is unsuitable.

If you do buy the parts individually you may find that some suppliers do not stock the Darlington array as this is a slightly specialised chip. The relays need to be of suitable dimensions to fit in the space available on the 'Verobloc' and it is possible that the pin connections will be different.

DC motors

If you are building equipment to control through the user port it will not be long before you find that you need a motor to drive a wheel or a pulley. The motors used in battery driven toys are not suitable and it may be difficult to get hold of the type of motors used in cassette players.

One possibility is the motors made by **FISCHERTECHNIK** and **LEGO** which are quite robust and can be fixed to the equipment quite easily.

Another alternative is to be on the lookout for companies selling specialised motors which are redundant stock or possibly not up to the required standard (a sub-standard motor often works perfectly but runs at a speed which is unsuitable for its intended application).

For example, Greenweld Electronics have a supply of motorized gearboxes with individual drive to each wheel and a magnetic clutch. These would make an ideal drive unit for a 'turtle' type vehicle.

If you are on the lookout for suitable bits and pieces it is suprising what can be obtained. An old car cassette player which is about to be thrown away can provide a motor, pullies, levers and all sorts of bits and pieces which turn out to be just what's wanted for a particular project.

Appendix 2

Four A to D channels

Figure 47 shows the extra connections needed to convert your A to D interface so that all four channels and both 'Fire' buttons can be used.

If the existing interface is mounted on a larger piece of wood another terminal block can be mounted alongside the original one so that the two parts of the interface are twins.

Remember that *FX16,n selects how many channels are being converted. If you are adapting some of the original programs you will have to change *FX16,2 to *FX16,4 to use all the channels.

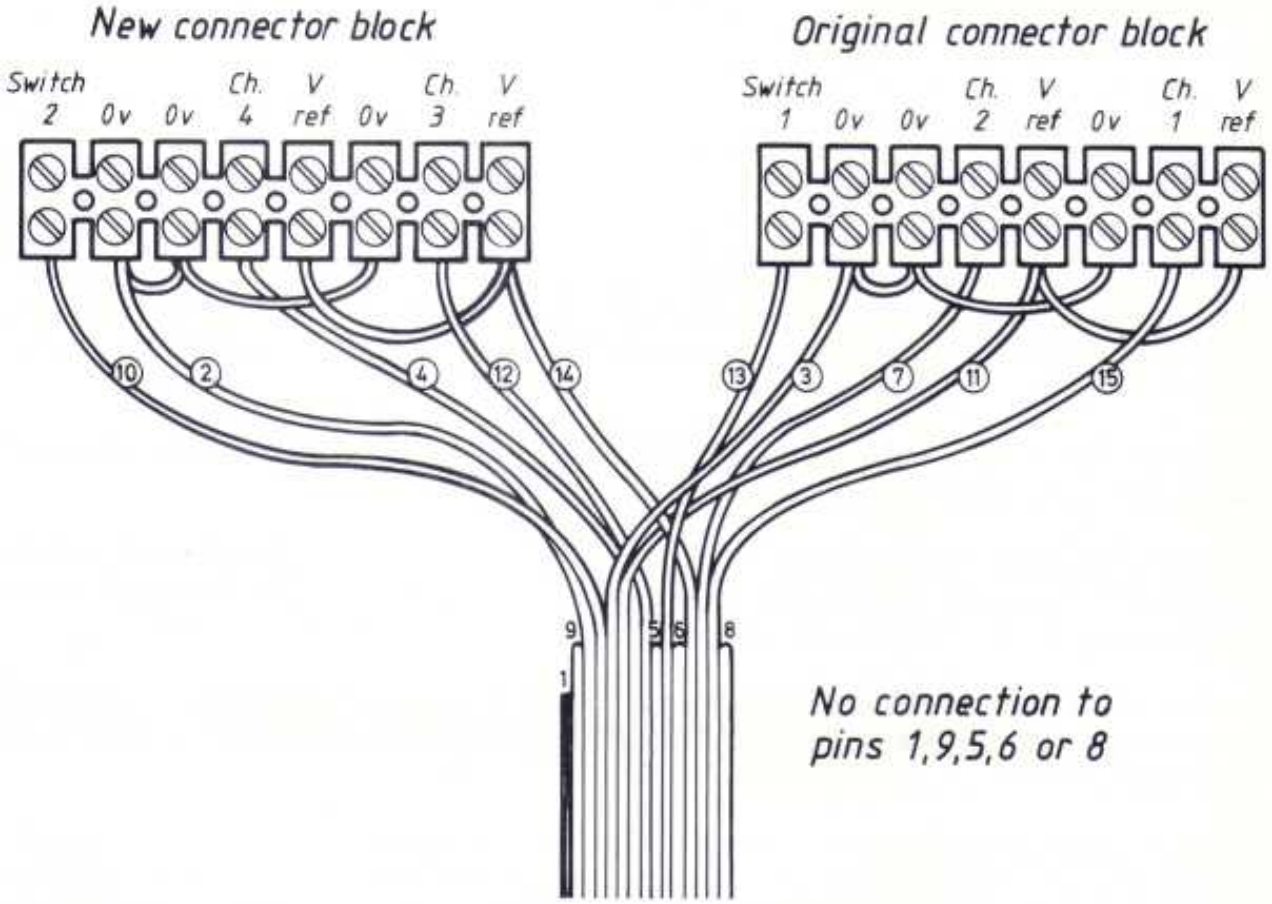


Figure 47

Appendix 3

- technical terms -

- programming terms and commands -

- further reading -

Technical Terms

Analogue to Digital Converter - a device which produces a digital value dependent upon the voltage of the analogue input signal.

Breadboard - used to describe perforated board with interconnected holes used to make prototypes of electrical circuits. Usually refers to the type of board which allows components to be inserted without soldering.

Cable - insulated wire.

Chip - see Integrated Circuit.

Component - a single device for use as part of an electrical circuit which will affect the action of the electrical signals in the circuit.

Connector block - see Terminal Block.

Current - the flow of electricity.

D.i.l. - dual-in-line, a particular arrangement of the pins which protrude from a chip or a connector.

Darlington Driver Array - a chip containing a number of Darlington transistor stages. The Darlington arrangement of open collector transistors can be used as switches.

Diode - a component which allows electricity to flow through it in one direction only.

Ground - also called Earth (0 volts).

IDC - insulation displacement connector. A type of connector which pierces the cable's insulation when it is closed making contact with the wires inside.

Integrated circuit (I.C.) - a number of components forming all or part of an electrical circuit etched on a small piece of silicon and contained in a plastic package. Also called a micro-chip or chip.

Interface - a method of joining two electrical systems.

LED - a diode which produces light when a current flows through it.

Light Dependent Resistor (LDR) - a device which produces a resistance dependent upon the amount of light falling on it.

Plug - the half of a connecting device which holds the pins.

Push-to-make switch - type of switch in which the contacts are only joined when the switch is pushed.

Port - a plug or a socket which can be used to feed signals into or out of a computer.

Potentiometer - a component which produces a variable resistance to an electrical current e.g. a volume control.

Power supply - any source of electrical power. Usually refers to a transformer.

Reed switch - a switch which is operated by a magnetic force.

Reference voltage - a fixed voltage used as a comparison for other variable votages.

Relay - a switch which is operated by an electrical signal. Electro-mechanical relays contain coils which physically move two contacts together using electro-magnetic force.

Resistor - a component which resists the flow of electricity.

Ribbon cable - multi-way cable moulded as a flat strip with the individual cables side by side.

Sensor - a device which detects change in the environment (light, temperature, sound etc.).

Socket - the half of a connecting device into which the pins are inserted.

Terminal block - a block, usually made of plastic, holding integral contacts used to join two or more wires. Also called Connector Block.

Thermistor - a device which produces a resistance dependent upon its temperature.

Tilt switch - a switch which is on or off according to its attitude e.g. a glass bead containing mercury.

Tin - to thinly cover with solder.

Transformer - a device which produces an electric current (either AC or DC) from another of a different voltage.

Voltage - a measurement of electromotive force.

Wire - strictly speaking the metal conductor inside a cable, but often used to mean the conductor and its external insulation.

Programming Terms

(Page numbers refer to the relevant entries in the User Guide)

ADVAL - short for Analogue to Digital Value. Produces a reading from a selected channel of the the A to D converter e.g. ADVAL(3) produces the last known value from channel 3. ADVAL(0) AND n tests a selected 'fire' button (pages 202, 426 and 467).

DIV - operator which produces the whole number result of a division without any remainder or fraction e.g. 39/7=5 rem 4, 39 DIV 7=5 (page 238).

EOR - exclusive-or, compares corresponding bits in two numbers. The resultant bit will be 0 if both are the same, 1 if they are different (page 250).

MOD - operator which produces the remainder from a division (using whole numbers only) e.g. 39/7=5 rem 4, 39 MOD 7=4.

? - the indirection operator which reads or writes one byte (eight bits). When used before the equals sign it reads the memory location, when used after the sign it writes to the memory location e.g. X=?&70 - X takes the value in memory location &70; ?&70=X - memory location &70 will contain the value of X (page 409). *This method of addressing the 6522 VIA is strictly speaking 'wrong' though it is far easier to use than the 'correct' method. It will not work if a second processor is connected to the BBC micro. The 'correct' method uses the OSBYTE call with A=&96 to read and A=&97 to write to the VIA registers (page 436).*

***FX16,n** - selects which A to D channels will be converted in each cycle (page 426 and *FX17 which follows).

State - the value of any particular bit or line. If an electrical signal is present the line will be high (5 volts) and is represented by a binary 1. If no signal is present the line will be low (0 volts) and is represented by binary 0.

Register - a particular memory location used for a specific purpose. Note that the bits in a particular eight bit location are correctly described as bits 0 to 7 not bits 1 to 8. Therefore the first line from the user port is controlled by bit 0. For simplicity the lines have been described as 1 to 8 in this book.

Channel - a path between different parts of a microcomputer along which signals can pass.

Data Direction Register - the register in the 6522 VIA which sets the lines from the User Port to be inputs or outputs (address &FE62).

Further Reading

THE USER GUIDE *(British Broadcasting Corporation)* - in particular the relevant entries in the BASIC keywords section and chapter 44.

THE ADVANCED USER GUIDE *by Bray, Dickens and Holmes (The Cambridge Microcomputer Centre)* - chapters 22, 24 and 26. Detailed descriptions of the relevant hardware. Useful for those who want to access the A to D Converter or the User Port from assembly language programs.

THE BBC MICRO - AN EXPERT GUIDE *by Mike James (Granada Publishing Limited)* - an excellent introduction to some of the more advanced features of the BBC micro. Chapter 6, 'Interfacing', gives a full description of the facilities available from the User Port.

THE MICRO USER *(monthly)* - the regular 'Body Building' features in the magazine offer some interesting ideas. Most of the projects expect some skill with a soldering iron.

ACORN USER *(monthly)* - no regular interfacing series but often contains relevant articles.

ELECTRONICS AND COMPUTING *(monthly)* - not specifically BBC orientated. Many projects each month but more technical than the MICRO USER.

SOFTWARE

All the programs included in this book are available on Disc or Cassette Tape from:

Primary Programs Ltd.
Claypits
Debden Road
Saffron Walden
Essex CB11 3JS

In addition to the programs in the book a 'friendly' User Port control program is included. Features include loops, variable delays, help page etc.

Cassette - £5.95, Disc - £7.95 (prices applicable until June 1985)

KITS OF PARTS FOR THE PROJECTS

Two kits are available which contain all the parts needed to complete the projects described in the book.

Kit 1 - Analogue to Digital Converter Projects

Contents: 15 way D plug IDC type
1 metre 16 way ribbon cable
12 way terminal block
2 metres of 2 core cable
Push-to-make switch
Two Light Dependent Resistors
Tilt switch (resin dipped)
15K Thermistor
Two 2K Linear pots
Two 1K resistors
Two 2.2K resistors
One 15K resistor
Roll PVC tape
Block of wood 10cm x 7cm x 2 cm

Kit 2 - User Port Projects

Contents: 2 metres 1/.06 solid core equipment wire
IDC 20 way pin socket
1 metre 16 way ribbon cable
16 way IDC d.i.l. connector
One piece 'Verobloc'
7 way Darlington Driver Array I.C. with R-in
Two sub-minature 12 volt relays
Four 1N4001 diodes
Four 470ohm resistors
Four L.E.D.s
Power Supply Unit

Both kits are available from:

Greenweld Electronics Ltd.
443 Millbrook Road
Southampton SO1 0HX.
Telephone (0703) 772501/783740.

Please telephone for current prices before ordering.

The designers of the BBC microcomputer included a number of sockets to allow the computer to receive signals from and send signals to the outside world. Yet the majority of owners of the BBC micro have only used the cassette and TV/RGB sockets at the back of the computer and the printer and disc drive plugs underneath.

This book introduces ways of using the *ANALOGUE IN* socket and the *USER PORT* to interface the computer with its environment. The projects are fully explained in non-technical language at every stage and ideas are given for additional experiments.

No experience of electronics is needed – not even a soldering iron. But that does not mean the projects are trivial – they have been carefully designed around components which can be fitted together without soldering.

If you have never used the computer as a link to the outside world then you will not have realised how versatile the BBC micro can be. This book will show you the way to many exciting hours of experimenting.

ISBN 0-946705-03-8

£5.95 UK ONLY

443 Millbrook Road
Southampton SO1 0HX

Micro books