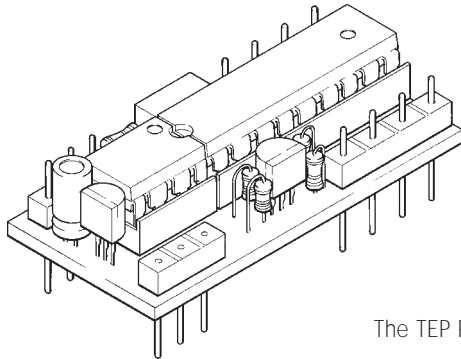


# THE TEP PLC

The TEP PLC system is similar *in principle* to most commercial types. A very general definition of a PLC is:

*'an electronic device that uses a programmable memory to store instructions and to implement specific functions such as logic, sequence, timing, counting and arithmetic to control machines and processes.'*



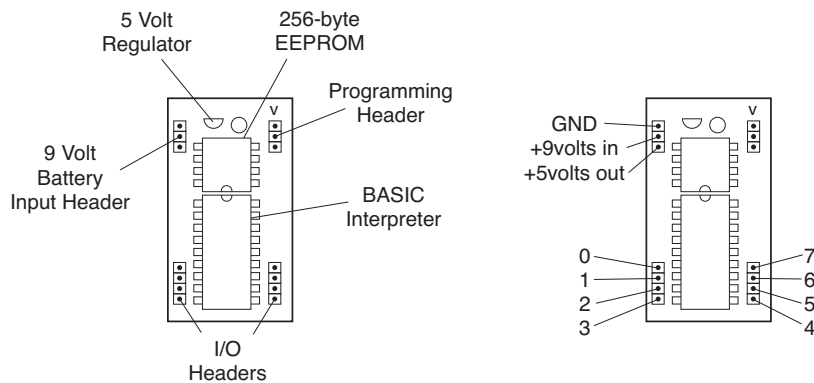
The TEP PLC chip

The PLC chip controller **differs** from most commercial PLCs in only two major respects:

- it requires you to add output stages such as transistors and relays to switch larger currents on and off,
- it is programmed using *Basic* - a common computer programming language using *verbal* commands such as 'if', 'then' and 'goto' rather than the symbols of ladder logic.

You may well already have done some computer control work using a 'buffer box' to link your computer and anything to be turned on and off. Using the PLC chip is very similar. The main difference is that when you have created your program the PLC can then be disconnected from the computer to run it. It has a *non-volatile* memory which means it will store the programme even if its power supply is disconnected.

The diagram shows the components that make up the TEP PLC Chip computer:



**9 Volt Battery input Header:** This is where you connect the power supply to the TEP PLC Chip.

**Do not use any other power source. Connecting other sources of power directly to the circuit board could permanently damage the TEP PLC Chip.**

**5 volt regulator:** This regulates the incoming power supply to 5 volts which is the working voltage of the TEP PLC Chip.

**Programming Header:** This is the connection point for the lead that connects to your PC. The cable must be connected whenever you want to change the TEP PLC Chip's program.

**256 Byte EEPROM:** This is the 'memory' of the TEP PLC Chip. It stores your program instructions. EEPROM stands for - **Electrically Erasable Programmable Read Only Memory.**

**BASIC Interpreter:** This is the 'brain' of the TEP PLC Chip. It reads your instructions that are stored in the EEPROM and carries them out in sequence.

**I/O Header:** This is the connection point for all of the input and output devices.

#### IMPORTANT NOTE:

As you can see the I/O header allows for 8 connection points (numbered 0 to 7). These are similar to the connection points on a standard 'buffer box'. They can be used to switch external devices on or off, or they can be used to input data, e.g. from a switch. When 'High' they will be at 5 volts and when 'Low' at 0 volts.

When used as outputs they can only supply a limited amount of current. Each pin can **sink 25mA** and **source 20mA**. The total sink and source for all 8 pins combined should never exceed **50mA sink** and **40mA source**. This means that you will need to use extra components to interface the TEP PLC Chip to high current loads, e.g. motors.

**Drawing excessive current from the TEP PLC Chip will permanently damage it.**

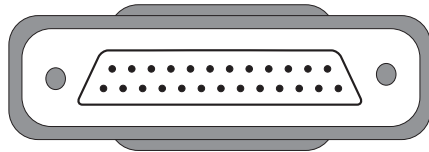
## CONNECTING THE TEP PLC CHIP TO YOUR PC

The TEP PLC Chip can be programmed by any IBM PC or compatible computer with the following specification:

- Disk drive
- Parallel port
- 128K or greater of RAM
- MS-DOS 2.0 or greater

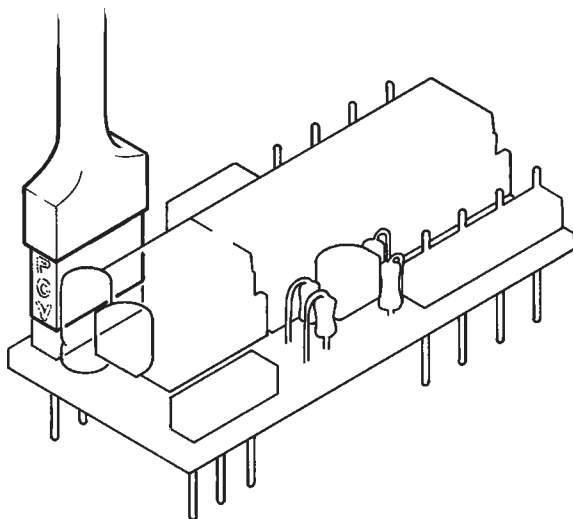
To connect the TEP PLC Chip to the PC, follow these three steps:

1. In the TEP PLC Chip development kit there is a cable to connect it to your PC. It has two ends. Plug the large DB25 connector into the **parallel port** of your PC.



DB25 connector plugs into the parallel port of your PC.

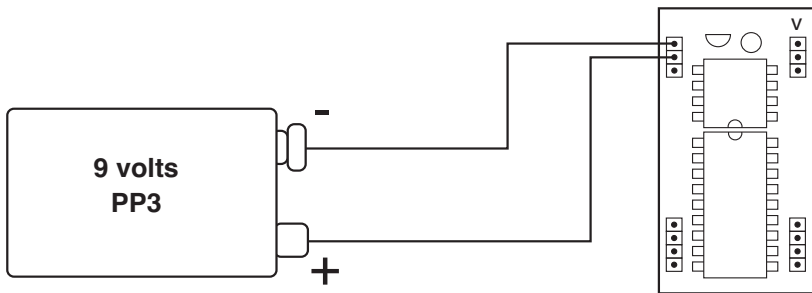
2. Plug the small 3-pin connector into the TEP PLC Chip Programming Header. Note that the 3-pin plug has a small arrow (>) on one side. This should be next to the end of the socket at the edge of the circuit board.



Three pin connector plugs into the Stamp Programming Header.

(Note the 'V' on the plug lines up with the end of the socket at the edge of the circuit board.)

3. Connect a 9 volt PP3 battery to the battery input header.



9 volt PP3 connected to the battery input header

### INSTALLING THE SOFTWARE

The program will run more efficiently if it is installed on the hard disk of your PC. The following steps describe how to install the software onto a hard disk.

Switch on the PC and wait for the C: Prompt

Create a directory called 'Stamp' for the software to be installed into.

Type:

**mkdir stamp** (enter)

Insert the floppy disk into drive A: then copy all of the files on the floppy disk into this new directory.

Type:

**copy a:.\* c:\stamp** (enter)

To run the program you need to change to the Stamp directory and type the name of the program.

Type:

**cd stamp** (enter)

**stamp** (enter)

If your PC does not have a hard disk then it is possible to run the software from the floppy disk provided in the TEP PLC Chip kit. Insert the floppy disk into drive A:

Type:

**stamp** (enter)

Assuming that everything is connected and installed correctly the program will open after a few seconds. The program screen is dark blue with one line across the top that names various disk and downloading functions.

### PROGRAMMING THE TEP PLC CHIP

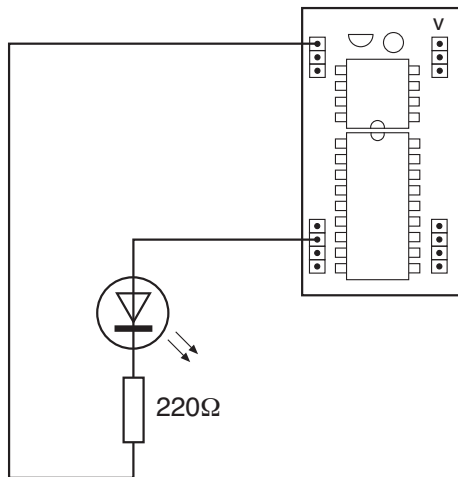
The TEP PLC Chip is programmed using a form of language called BASIC. This is fairly common programming language and is quite simple to learn. If you have done some programming in BASIC before then you will recognise many of the commands and structures that are used for the TEP PLC Chip.

This section will give you an introduction to the form of BASIC that is used to program the TEP PLC Chip. When you have mastered all of the information here you can find out more about programming the TEP PLC Chip by reading the instruction manual supplied by the manufacturers.

### OUTPUTS

We will start by using some simple commands to switch the outputs on or off.

Connect an LED and a resistor to output pin 1 as shown in the diagram. (Make sure that you disconnect the TEP PLC Chip from the PC and the battery while you are doing this.) The PLC Chip can be mounted in a 28 pin DIL IC socket or on a prototyping breadboard.



LED and resistor connected to output pin 1 on the stamp

When the TEP PLC Chip is reconnected to the battery and the PC the LED should not be lit.

The first three commands to use are:

**High** - sets an output pin high (5 volts) The following number sets which pin will be affected from 0 to 7.

**Low** - sets an output pin low (0 volts) The following number sets which pin will be affected from 0 to 7.

**Pause** - pauses the program for 0 to 65536 milliseconds

Now we will write a simple **program** or sequence of instructions for the TEP PLC Chip to follow.

Type:

```
high 1
pause 500
low 1
pause 500
high 1
pause 500
low 1
```

This is a very simple program.

**High 1** — Output pin1 will go high, switching on the LED.

**Pause 500** — It will stay high for 500 milliseconds (0.5 sec).

**Low 1**— Output pin1 will go low, switching off the LED.

**Pause 500** — It will stay low for 500 milliseconds (0.5 sec).

**High 1** — Output pin1 will go high, switching on the LED.

**Pause 500** — It will stay high for 500 milliseconds (0.5 sec).

**Low 1**— Output pin1 will go low, switching off the LED.

When the program is run the LED will flash on and off twice with a 0.5 second gap.

To run the program you will need to **Download** it to the TEP PLC Chip. This process transfers the program from the PC to the TEP PLC Chip. To download a program:

Press and hold down the **Alt** key and press **R** (Alt-R)

When you have done this you will see a box appear on the screen that contains a bar-graph. The bar-graph will fill as the program is downloaded. Notice that the bar-graph is made up from white and red blocks. The white blocks show how much memory is left free in the TEP PLC Chip and the red blocks show how much memory is used by the program. If an error message shows at this stage then check the connection between the TEP PLC Chip and the computer.

When the bar-graph reaches the end, the program will run on the TEP PLC Chip. You should see the LED flash on and off twice with a 0.5 second gap.

If you missed it then you can download the program again by simply pressing Alt-R.

## LABELS AND LOOPS

As you can see a program is a sequence of instructions. The TEP PLC Chip will follow the instructions in order from the first to the last and then the program will finish. It is possible to include a **loop** in the program. This will make the TEP PLC Chip follow the instructions to the end and then loop back to the beginning to follow them again. This type of program will run forever in a closed loop. If you are going to use a loop in the program then you must give the program a **Label** or name. This tells the TEP PLC Chip where the start of the program is.

Try this program. First use the delete key to clear all of the previous commands off the screen then type:

```
Flash: high 1
pause 100
low 1
pause 100
goto flash
```

Download the program by pressing Alt-R and see what happens. You should see the LED flashing on and off continuously with a 0.1 second gap.

This is what the commands do:

**Flash:** — This is the program label. Notice that it has a colon (:) after it

**high 1** — Output pin1 will go high, switching on the LED  
**pause 100** — It will stay high for 100 milliseconds (0.1 sec).  
**low 1**— Output pin1 will go low, switching off the LED.  
**pause 100** — It will stay low for 100 milliseconds (0.1 sec).

**goto flash** — This command tells the TEP PLC Chip to **goto** the start of the program called **flash**.

Now disconnect the TEP PLC Chip from the PC. You will see that the LED keeps flashing. This is because the program has been downloaded from the PC to the TEP PLC Chip. It has all the information it needs to run the program.

Now disconnect the battery from the TEP PLC Chip. The LED will stop flashing because the TEP PLC Chip has no power supply.

Now reconnect the battery. The LED should start flashing again. The TEP PLC Chip can remember its program even if the power supply is disconnected.

Reconnect the TEP PLC Chip to the PC.

To stop the LED from flashing you will have to clear the program from the TEP PLC Chip. To do this you must download a 'program' with no commands in it. Use the delete key to clear all of the commands from the screen and then download this empty 'program' by pressing Alt-R.

When the download is complete the LED should stop flashing.

### FOR...NEXT LOOPS

Another more versatile kind of loop is the **FOR...NEXT LOOP**. This kind of loop is used when you want the TEP PLC Chip to follow a sequence of commands a certain number of times. The number of times that the program runs for is set by a **variable**. A variable is a number that is stored in the memory of the TEP PLC Chip. There are 14 **memory locations** that variables can be stored in. They are numbered from **b0** to **b13**.

Try this program (remember to use the delete key to clear any commands that are on the screen before you begin).

```
Flash:  for b1 = 1 to 20
        high 1
        pause 100
        low 1
        pause 100
        next
```

Download the program by pressing Alt-R and see what happens. You should see the LED flash on and off 20 times with a 0.1 second gap.

This is what the commands do:

**Flash:** — This is the program label. Notice that it has a colon (:) after it.

**for b1 = 1 to 20** — This is the start of the for..next loop. It tells the TEP PLC Chip to store a variable between 1 and 20 in memory location b1. The variable will start at 1 and count up to 20. It will increase by 1 each time the program goes around the loop.

**high 1** — Output pin1 will go high, switching on the LED  
**pause 100** — It will stay high for 100 milliseconds (0.1 sec).  
**low 1** — Output pin1 will go low, switching off the LED.  
**pause 100** — It will stay low for 100 milliseconds (0.1 sec).

**Next** — This is the end of the for...next loop. Each time the program reaches this command it sends it back to the start. Each time the program begins again the variable stored in memory location b1 counts up by 1. When it reaches 20 the program will stop.

When for..next loops are completed a program will continue with any more commands that follow it.

Add these lines to your program.

```
Flash:  for b1 = 1 to 20  
         high 1  
         pause 100  
         low 1  
         pause 100  
         next  
         high 1  
         pause 5000  
         low 1
```

Download the program and see what happens. The LED should flash on and off 20 times with a 0.1 second gap then switch on for 5 seconds and then switch off again.

This is because the program goes around the for...next loop 20 times then comes out of the loop and continues with the program.

#### SAVING AND LOADING PROGRAMS

Once you have written a program you can save it onto the hard disk of your PC so that you can use it again in the future. To do this you press and hold the Alt key then press S (Alt-S). A box will appear on the screen for you to type in your file name. Type in a name and then press enter. The file will be saved on the hard disk of the PC.

**Filenames must use 7 letters or less and have no punctuation.**

To load a program from the hard disk you press and hold the Alt key then press L (Alt-L). A box will appear on the screen for you to type in the file name that you want to load. Type in a name and then press enter. If the file that you want is on the hard disk it will load onto the screen.

#### QUITTING FROM THE PROGRAM

To quit from the program simply press and hold the Alt key then press Q (Alt-Q).

**INPUTS**

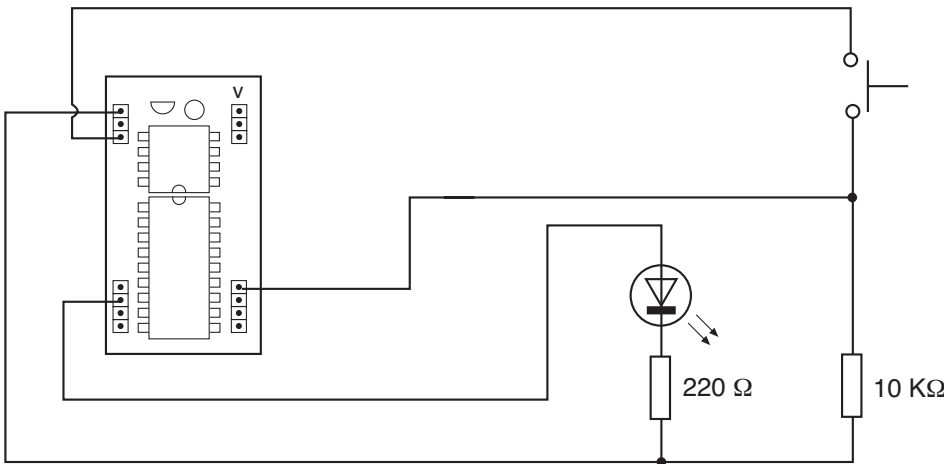
Using the above commands and structures you can now write programs that can set any of the output pins high or low in any sequence, with time delays. To make the TEP PLC Chip more useful as the ‘brain’ in a control system you will need to be able to make it respond to inputs. Any of the pins can be used to read inputs by using the command:

**input** — sets any pin to an input pin. The following number identifies which pin will be affected from 0 to 7. (e.g. Input 6).

You can use a switch to give an input. If you connect 0 volts to an input pin then the TEP PLC Chip will read it as “0”. If you connect +5 volts to an input pin then the TEP PLC Chip will read it as “1”.

A good way to use inputs is with an **If...Then** command.

Make sure that your LED and resistor are still connected as before. Connect a push-to-make switch and resistor to pin 7 as in the diagram.



Push-to-make switch and resistor connected to the PLC chip.  
(LED and Resistor still connected)

Connect the TEP PLC Chip to the PC and the battery. Make sure that the screen is clear of all commands and then try this program.

**input 7**

**switch:** **if pin7 = 1 then flash**  
**goto switch**

**flash:** **for b1 = 1 to 10**  
**high 1**  
**pause 100**  
**low 1**  
**pause 100**  
**next**  
**goto switch**

Download the program by pressing Alt-R and see what happens. The LED should not be lit. When you push the switch the LED will flash 10 times with a 0.1 second gap.

Try pushing the switch again. The program should run again. If you disconnect the TEP PLC Chip from the computer it will remember and be able to run the program. Every time you push the switch the LED will flash 10 times with a 0.1 second gap.

This is what the commands do:

**input 7** — This sets pin7 to an input

**switch:** — This is the label for the first procedure in the program.

**if pin7 = 1 then flash** — This is the if...then command. If the input to pin7 is +5 volts (1) then the program will goto the procedure labelled flash. If the input to pin7 is not +5 volts then the procedure will continue.

**goto switch** — This is the end of the procedure labelled switch. It causes switch to run in an endless closed loop until the switch is pressed. When this happens the **if...then** command will send the TEP PLC Chip into the procedure labelled flash.

**flash:** — This is the label for the second procedure in the program.

**for b1 = 1 to 10** — This is the beginning of a for...next loop that runs ten times.

**high 1** — These four commands flash the LED on  
**pause 100** — and off  
**low 1**  
**pause 100**

**next** — This is the end of the for...next loop. After 10 repeats the procedure will continue to the next line down.

**goto switch** — This sends the TEP PLC Chip back to the start of the procedure labelled switch when the for...next loop is finished.

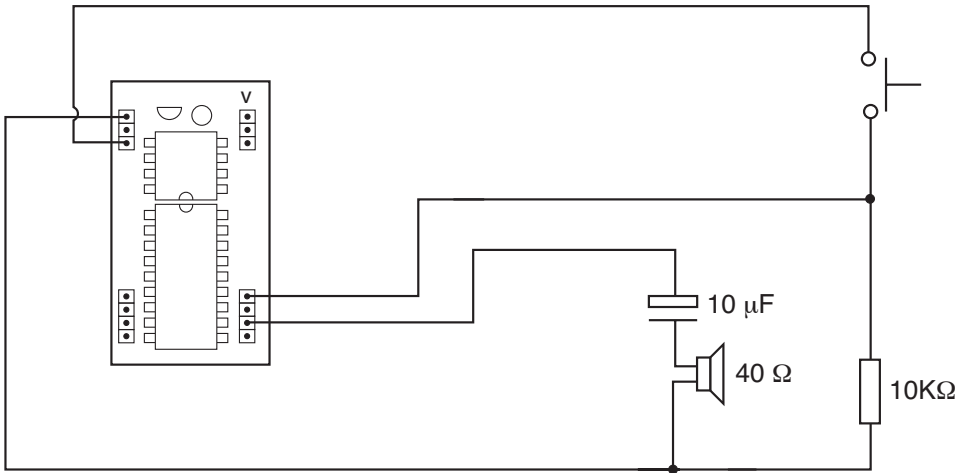
As you can see using the **if...then** command for inputs means that you have to have two labelled procedures in the program. The first procedure is an endless closed loop that keeps checking the state of the input. When the input is in the correct state the TEP PLC Chip is sent into the second procedure to do something. When the second procedure is completed the TEP PLC Chip can be sent back to the start of the first, ready to check the input again.

With this command structure added to the others you can now use the TEP PLC Chip as the 'brain' of some very complex control systems.

There are many more complex functions available from the TEP PLC Chip. One more very useful one to explore is **sound**.

SOUND

The TEP PLC Chip is able to generate sounds. To explore this you will need to connect a speaker and capacitor as shown in the diagram.



Speaker and capacitor connected to the PLC Chip  
(PTM switch and resistor still connected - LED and resistor left off for clarity)

The command structure for sounds is like this:

**sound (pin number), (note, duration)**

**pin number** — sets which pin the sound will be sent to

**note** — sets the pitch of the note. Numbers 1 to 127 are simple notes. The pitch of the note will increase as the number gets higher. Numbers 128 to 255 are “white noise”.

**duration** — sets the length of the note. The larger the number is the longer the note will be.

Connect the TEP PLC Chip to the PC and the battery and then try downloading this single command:

**Sound 5, (100,50)**

You should hear a single note.

Try changing the note and duration numbers in the brackets and downloading the command to the TEP PLC Chip.

You can play a sequence of notes by using the command structure:

**sound (pin number), (note,duration,note,duration..)**

Try downloading this command to the TEP PLC Chip:

**sound 5, (20,50,30,50,40,50,50,50)**

You should hear 4 notes that get higher in pitch each time.

The number that sets the note could be replaced by a **variable**. This variable could be generated using a for...next loop.

Try this program:

```
For b1 = 1 to 127
Sound 5, (b1,5)
next
```

When you download the program you should hear a sequence of notes that run from 1 to 127. This is because the number that sets the pitch of the note is set by the variable that is stored in memory location b1. This variable counts from 1 to 127, counting up 1 every time the program loops.

Try experimenting with the sound command using for...next loops.

You can use the sound command as part of procedures and programs. Try this program it will play a sound whenever the switch is pressed.

**input 7**

**abc:**    **if pin7 = 1 then xyz**  
          **goto abc**

**xyz:**    **for b2 = 1 to 127**  
          **sound 5, (b2,2)**  
          **next**  
          **goto abc**

COMMAND SUMMARY

These are the commands that you have used so far:

**High** — Sets an output pin high, e.g. high 1

**Low** — Sets an output pin low, e.g. low 1

**Pause** — Causes the program to wait for a number of milliseconds between 0 and 65536, e.g. pause 200

**label:** — Labels the start of a procedure in a program

**goto** — Always followed by a label. It will send the TEP PLC Chip to the part of the program where the label is.

**For...next** — Stores a variable in any memory location from b0 to b13 then causes a loop of all the commands between the for and next commands for a set number of times,

e.g.       for b1 = 1 to 20  
          .....  
          .....  
          next

**input** — Sets a pin to an input, e.g. input 5

**if...then** — causes a program to move to another procedure if an input pin is in the correct state,

e.g.       input 5  
  
          switch: **if pin5 = 1 then flash**  
  
          goto switch  
  
          Flash: ...

**Sound** — sends a sound to an output pin of set pitch and duration, e.g. sound 5, (100,20)

**Save** — Alt-S - Type in the file name to save then press enter.

**Load** — Alt-L - Type in the file name to load then press enter.

**Quit** — Alt-Q

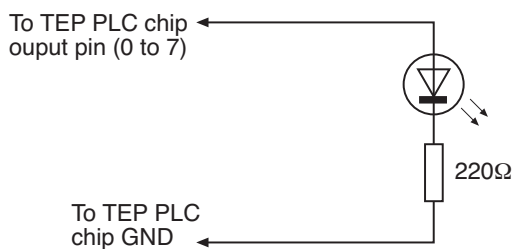
### INTERFACING THE TEP PLC CHIP

When you use the TEP PLC Chip as part of a control system you will want to switch different types of output devices on and off such as lamps, relays, motors, buzzers etc. You will probably want to use sensors as well as switches to provide inputs to the TEP PLC Chip. It is possible to use the TEP PLC Chip to do all of these things but you will need to make the necessary interface circuits.

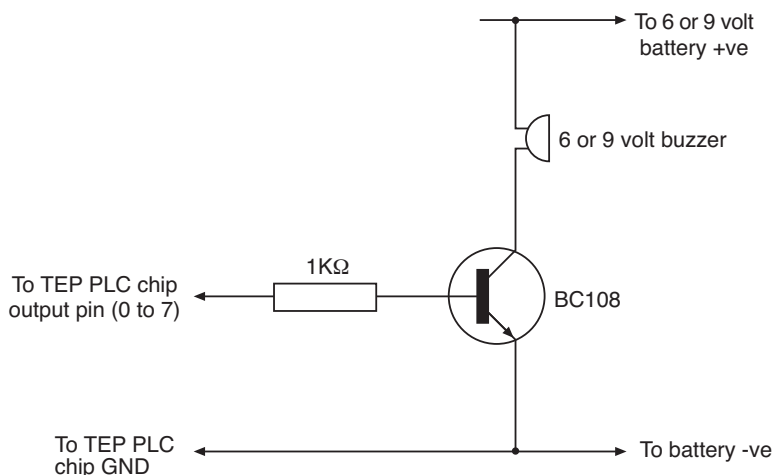
### OUTPUTS

The TEP PLC Chip can supply only a small amount of current to any device connected to an output pin (20mA maximum). Any output device that requires more current than this will need to be interfaced to protect the TEP PLC Chip.

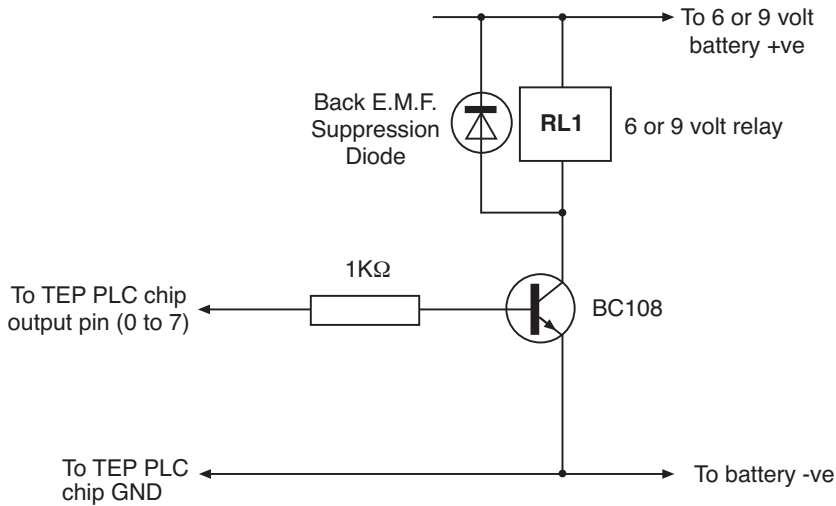
- **LED** — Can be connected directly to an output pin as long as a series resistor is used. A typical resistor value would be 220Ω.



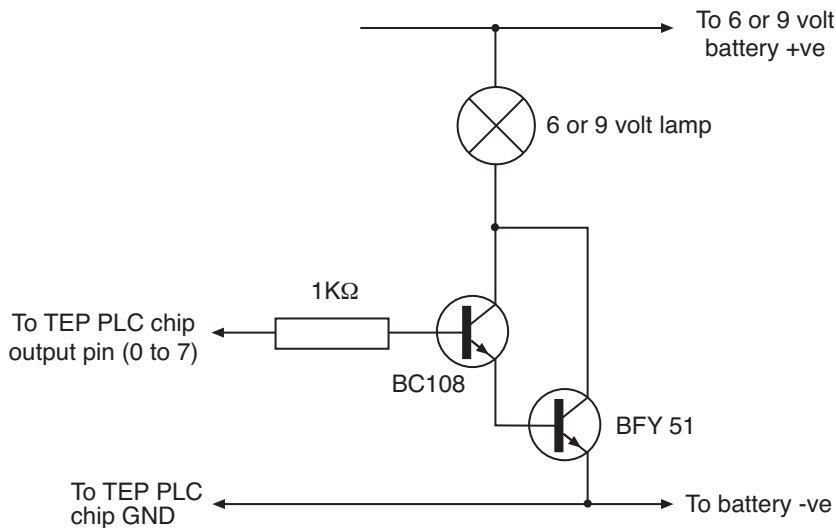
- **Buzzer** — Amplify the output current from the TEP PLC Chip using a single BC108 transistor as shown. Note the addition of an external power supply and that the two 0 volt lines are connected.



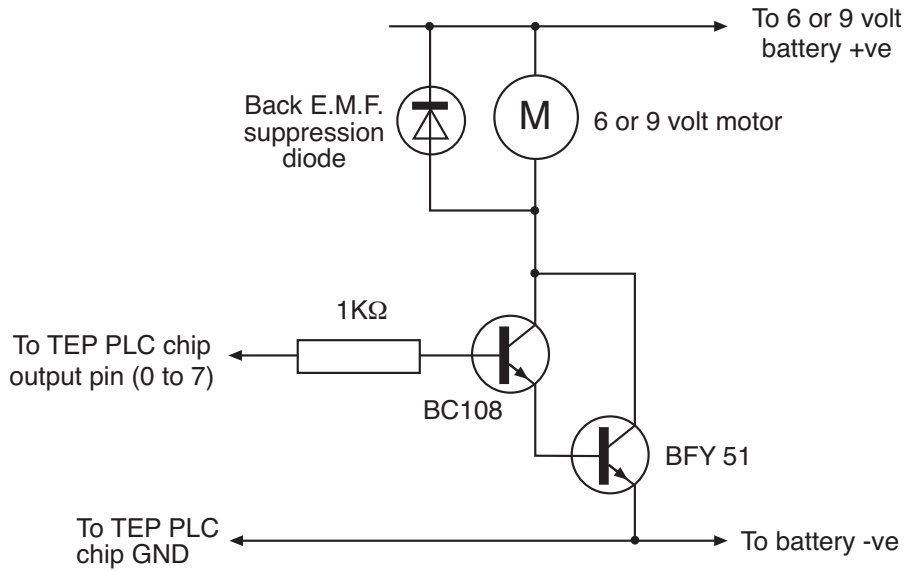
- Relay — Amplify the output current from the TEP PLC Chip using a single BC108 transistor as shown. Note the addition of an external power supply and that the two 0 volt lines are connected. Also note the use of the back e.m.f. suppression diode.



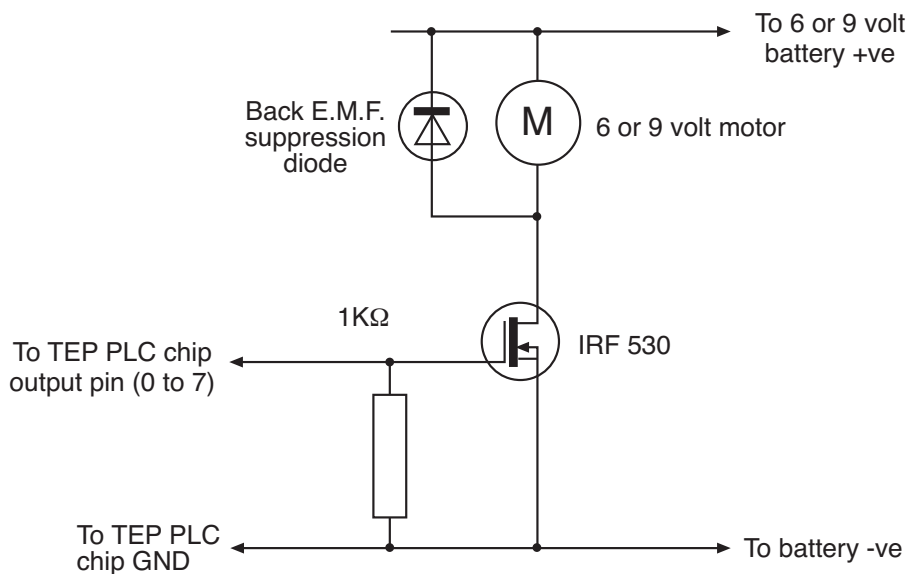
- Small Lamp — Amplify the output current from the TEP PLC Chip using a BC108 transistor and a BFY51 transistor as a Darlington Pair as shown. Note the addition of an external power supply and that the two 0 volt lines are connected.



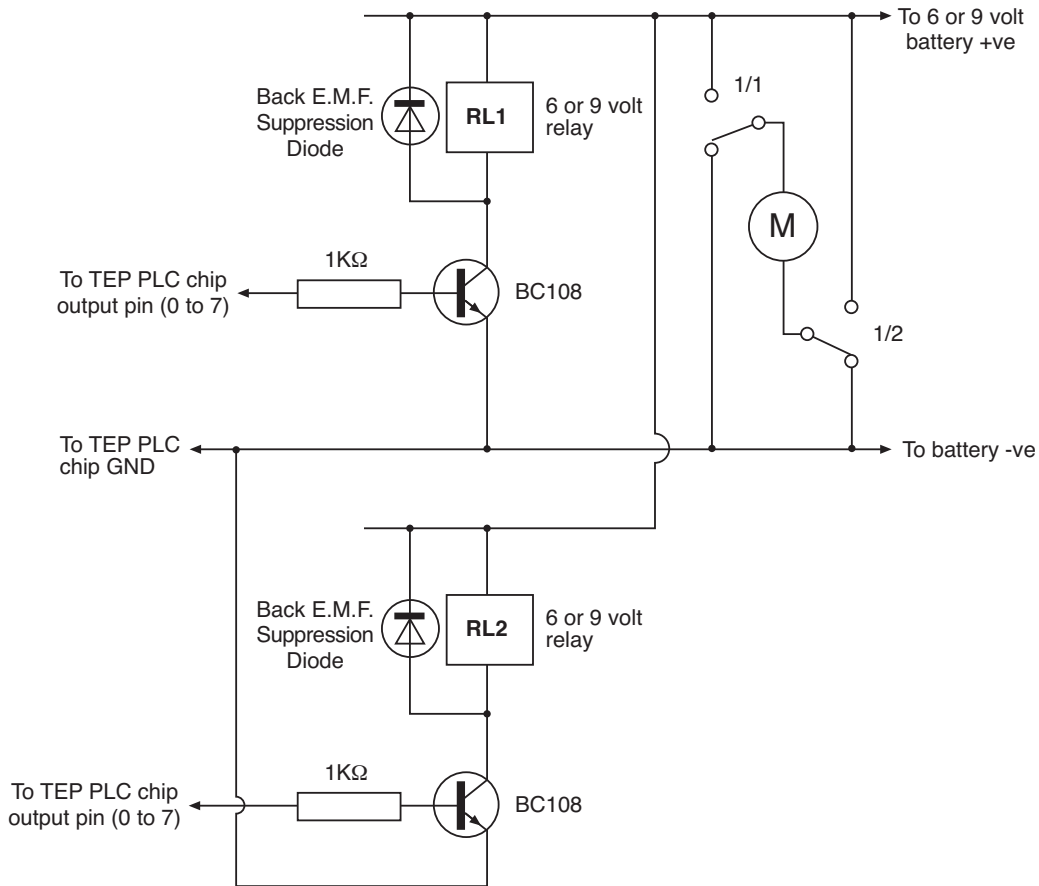
- Motor — Amplify the output current from the TEP PLC Chip using a BC108 transistor and a BFY51 transistor as a Darlington Pair as shown.



Alternatively amplify the current from the TEP PLC Chip using a power MOSFET (e.g. IRF530) as shown. Note the addition of an external power supply and that the two 0 volt lines are connected. Also note the use of the back e.m.f. suppression diode. (Certain motors will generate excessive electrical noise which could affect the operation of the TEP PLC Chip. If this happens then isolate the motor using a relay. If this fails then use a better quality motor.)



- Reversing a Motor — Use two relays as shown. Each relay will need to be driven by a BC108 transistor.



Both outputs low — Both relays de-energised, both motor contacts connected to negative, motor does not turn.

Top output high - Bottom output low — Top relay energised bottom relay de-energised, top motor contact connected to positive and bottom to negative, motor turns.

Top output low - Bottom output high — Top relay de-energised bottom relay energised, top motor contact connected to negative bottom to positive, motor turns in opposite direction.

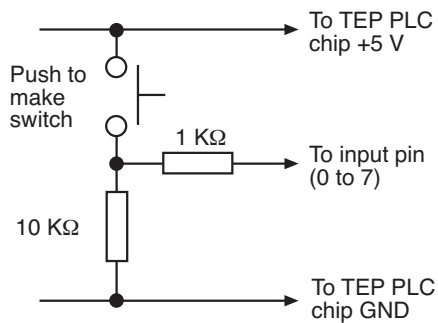
Both outputs high — Both relays energised, both motor contacts connected to positive, motor does not turn.

INPUTS

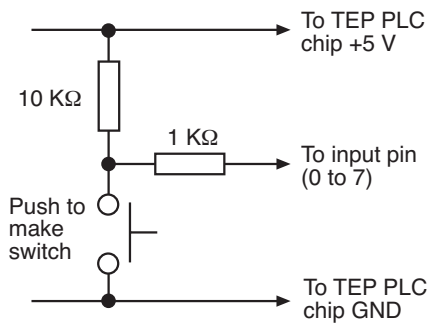
The TEP PLC Chip is a digital computer and will only accept digital inputs, i.e. signals that are either on (0 volts) or off (+5 volts)

**Never apply a voltage that is greater than +5 volts to an input pin as the TEP PLC Chip will be permanently damaged.**

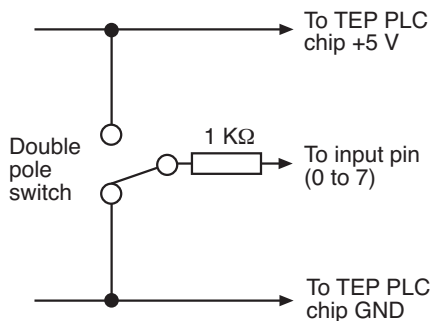
- Switch — different types of switches can be connected as shown to give various inputs.



When switch is pressed, input = 1  
When switch is released, input = 0

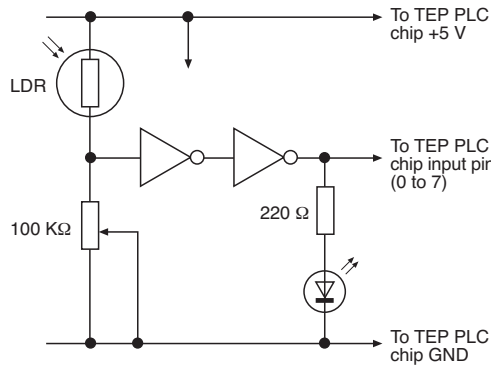


When switch is pressed, input = 0  
When switch is released, input = 1



When switch is in down position, input = 0  
When switch is in up position, input = 1

• **Light Dependent Resistor (LDR)** — A LDR and a resistor connected as a potential divider can give an analogue output voltage. This analogue signal needs to be converted to a simple digital signal. There are many ways of doing this. The method shown uses two inverters from a 4001 or 4011 IC.

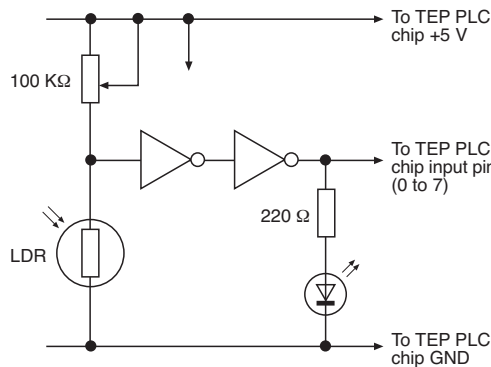
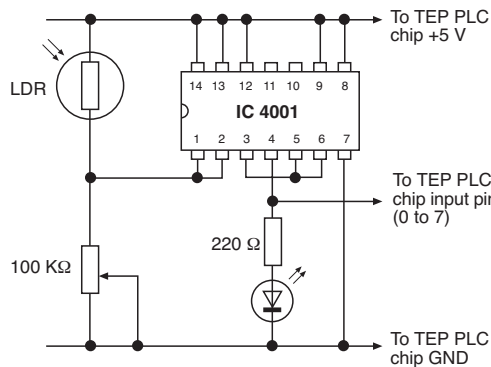


The sensor will give these inputs:

Light = 1  
Dark = 0

Use the 100 KΩ variable resistor to set up the switching point of the sensor.

The LED will be lit when the output from the sensor = 1

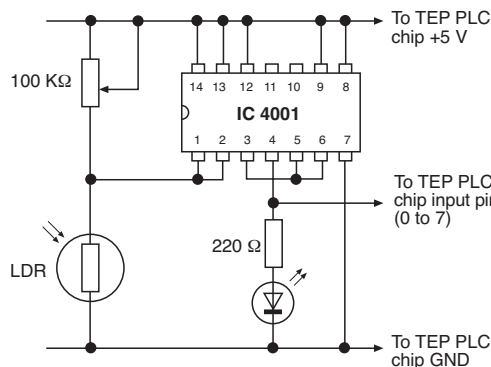


The sensor will give these inputs:

Light = 0  
Dark = 1

Use the 100KΩ variable resistor to set up the switching point of the sensor.

The LED will be lit when the output from the sensor = 1

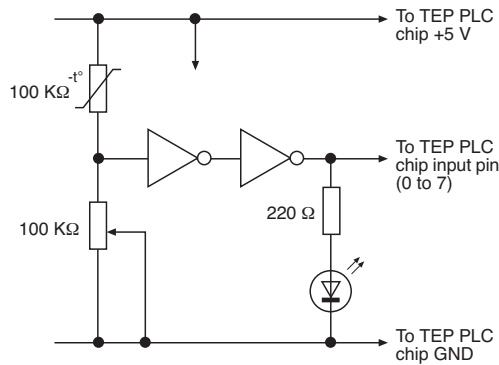


This is a good solution because the 'sensor' can be set up independently of the TEP PLC Chip and the LED gives an indication of the state of the output.

- **Thermistor** — A thermistor and a resistor connected as a potential divider can give an analogue output voltage. This analogue signal needs to be converted to a simple digital signal. There are many ways of doing this. The method shown uses two inverters from a 4001 or 4011 IC.

This is a good solution because the 'sensor' can be set up independently of the TEP PLC Chip and the LED gives an indication of the state of the output.

You may need to experiment with different values of thermistor and variable resistor to cover the temperature range that you need.

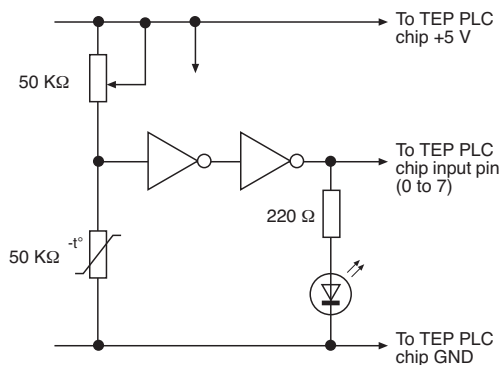
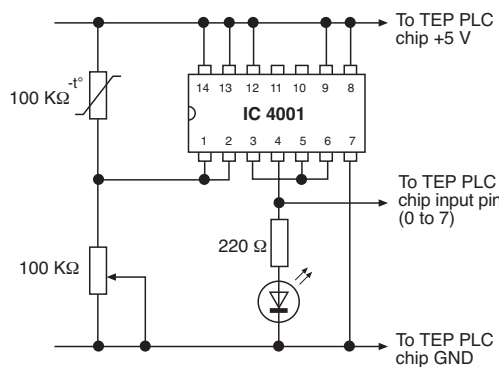


The sensor will give these inputs:

Hot = 1  
Cold = 0

Use the 100 KΩ variable resistor to set up the switching point of the sensor.

The LED will be lit when the output from the sensor = 1

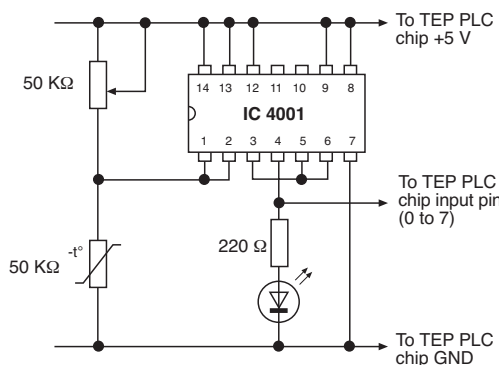


The sensor will give these inputs:

Hot = 0  
Cold = 1

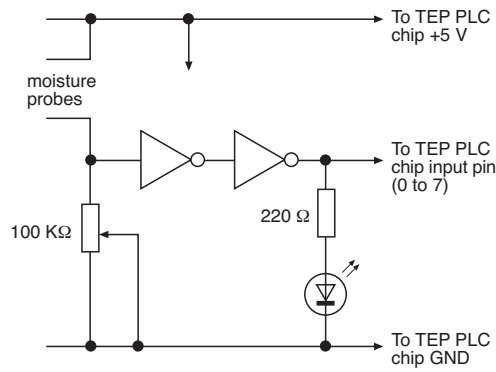
Use the 50 KΩ variable resistor to set up the switching point of the sensor.

The LED will be lit when the output from the sensor = 1



• **Moisture Sensor** — A moisture sensor can be made in a number of ways. The final design will depend upon the situation where it is to be used. The sensor and a resistor connected as a potential divider can give an analogue output voltage. This analogue signal needs to be converted to a simple digital signal. There are many ways of doing this. The method shown uses two inverters from a 4001 or 4011 IC.

This is a good solution because the 'sensor' can be set up independently of the TEP PLC Chip and the LED gives an indication of the state of the output.

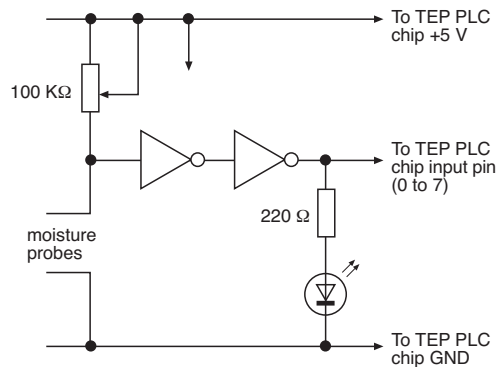
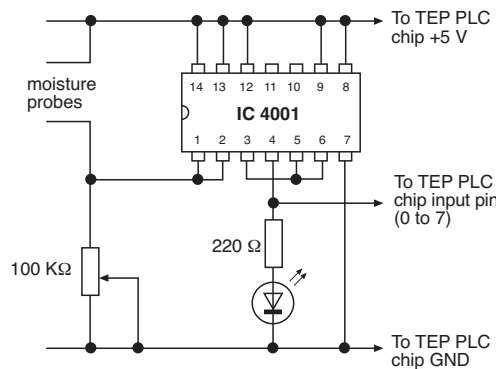


The sensor will give these inputs:

Wet = 1  
Dry = 0

Use the 100 KΩ variable resistor to set up the switching point of the sensor.

The LED will be lit when the output from the sensor = 1

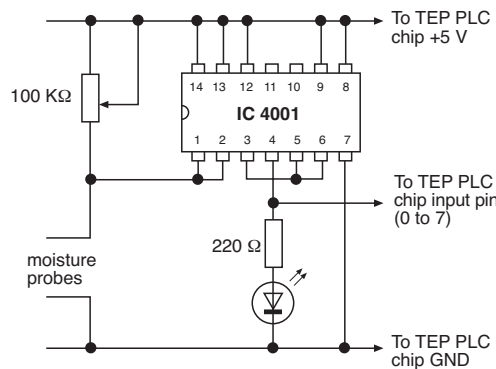


The sensor will give these inputs:

Wet = 0  
Dry = 1

Use the 100 KΩ variable resistor to set up the switching point of the sensor.

The LED will be lit when the output from the sensor = 1

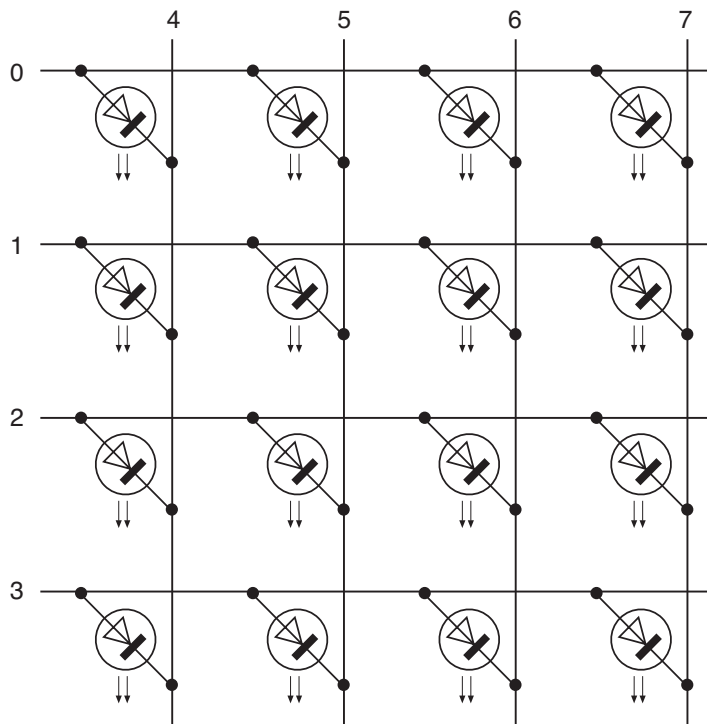


### SOME PROJECT STARTING POINTS

There follow a few ideas for project starting points. Not all of the essential detail is included but enough information is given elsewhere in this book for you to adapt the basic idea to your needs.

#### LED Matrix Output

This idea uses the simple output capabilities of the TEP PLC Chip to drive an LED matrix. The matrix is made up of 16 LEDs arranged in a 4×4 square.



Simple LED Matrix

There are 8 inputs to the matrix, labelled 0 to 7. They are connected to the 8 output pins from the TEP PLC Chip. It is possible to light any single LED or any combination of columns or rows using these 8 inputs.

For all of the LEDs to be off inputs 0 to 3 should be low and inputs 4 to 7 should be high. In this condition every LED is reverse biased and is not lit.

To light an LED you must 'address' it using the inputs like coordinates. For example the top left LED's coordinates are (4,0). To light it you must make input 4 low and input 0 high. Only the top left LED will be forward biased so only it will light. Notice the first number in the coordinate is made low and the second number is made high. You can use this as a general rule to address LEDs.

To light an entire column you would need to address it like this, e.g. To light the right hand column the LED's coordinates are, (7,0) , (7,1) , (7,2) , (7,3). Using the general rule that the first number in the coordinate is made low and the second number in the coordinate is made high, you would need to make input 7 low and inputs 0,1,2,3 all high.

To light an entire row you would use a very similar process, e.g. To light the bottom row the LED's coordinates are, (4,3) , (5,3) , (6,3) , (7,3). Using the general rule again you would need to make inputs 4,5,6,7 low and input 3 high.

You will need to add resistors to the basic matrix to protect the TEP PLC Chip. This could be done in a number of ways. The best solution would be to put a 220Ω resistor in series with each of the LEDs. A less effective but workable solution would be to put a single 1KΩ resistor in series with all of the vertical inputs or alternatively in all of the horizontal inputs.

Experimenting with this matrix of LEDs using the TEP PLC Chip should give you some useful ideas for displaying information.

#### SAMPLE PROGRAMS FOR MATRIX

```
1, abc:  for b1 = 0 to 7
         high b1
         next
         for b2 = 4 to 7
         low b2
         pause 100
         next
         for b3 = 0 to 3
         low b3
         pause 100
         next
         goto abc
```

```
2, xyz:  low 0
         high 1
         high 2
         low 3
         high 4
         low 5
         low 6
         high 7
         pause 500
         goto cde
```

```
cde:    high 0
         high 3
         low 4
         low 7
         pause 500
         goto xyz
```

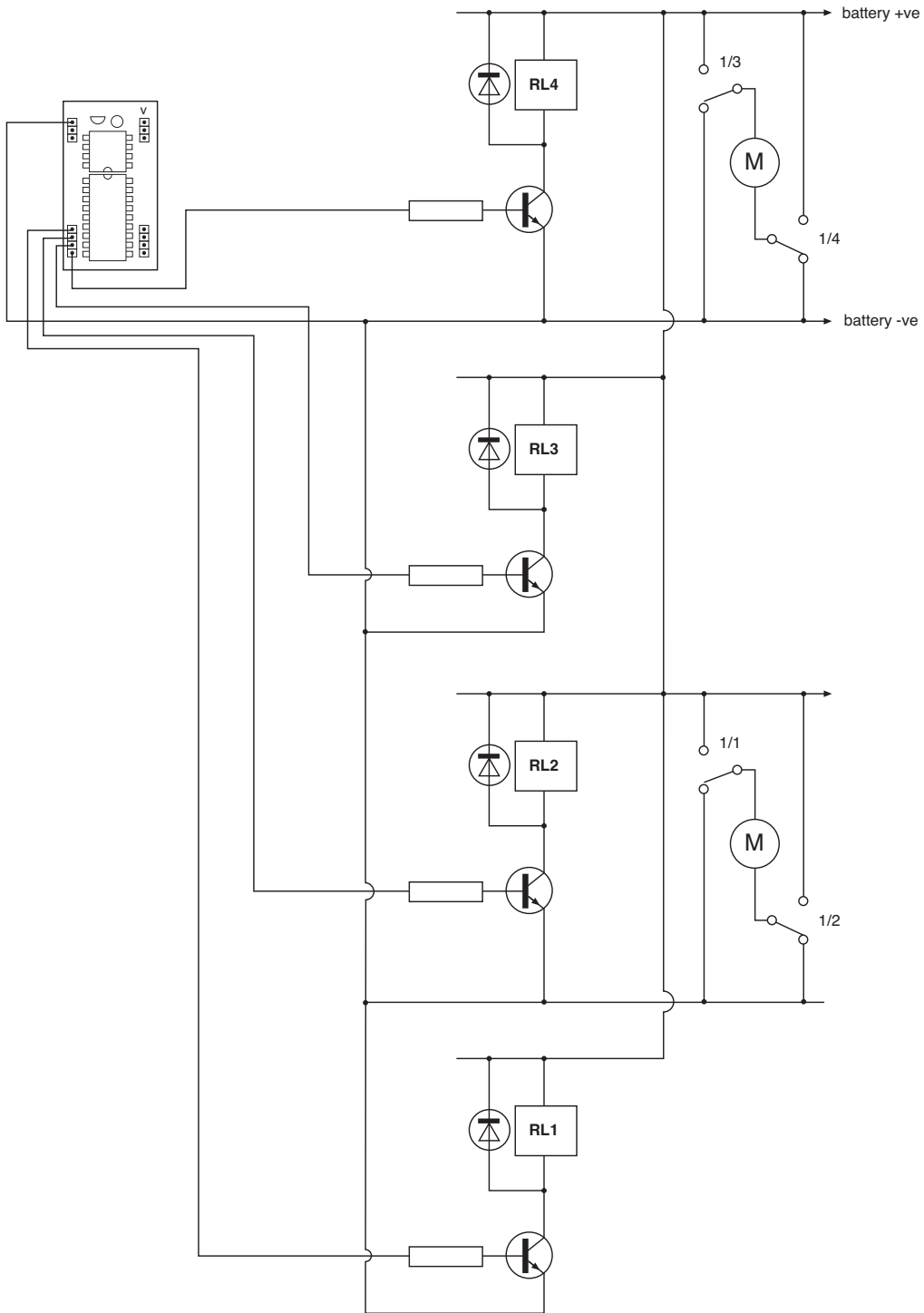
**Please note:**

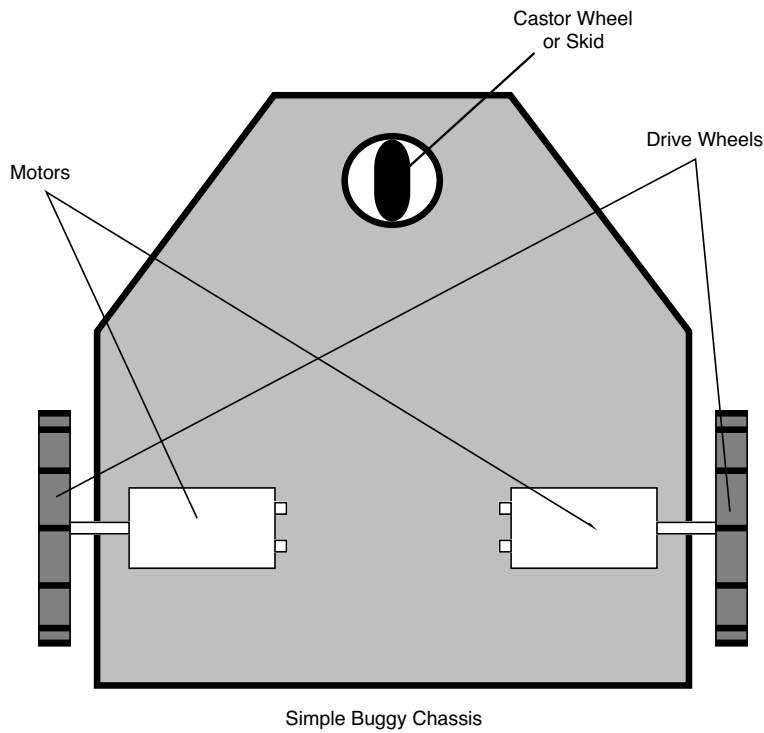
**If all the LEDs are lit at once then the maximum current that the TEP PLC Chip can supply will be exceeded.**

### SEQUENCE CONTROLLER

The TEP PLC Chip could be used for many different control systems that require a sequence of operations to be completed, from a simple 'programmable buggy' to a robot arm.

A simple buggy that has two reversible drive motors is shown.





The TEP PLC Chip outputs to manoeuvre the buggy are:

**Forward**    low 0  
                 high 1  
                 low 2  
                 high 3

**Reverse**    high 0  
                 low 1  
                 high 2  
                 low 3

**Left Turn**   low 0  
                 high 1  
                 low 2  
                 low 3

**Right Turn** low 0  
                 low 1  
                 low 2  
                 high 3

Other combinations of low and highs at outputs 0 to 4 would give different manoeuvres.

The four remaining TEP PLC Chip pins could be used to give outputs to LEDs, lamps, buzzers or speakers or they could be used as inputs. A switch mounted on the front of the buggy could cause it to stop, reverse, turn and continue when it bumps into an obstruction.

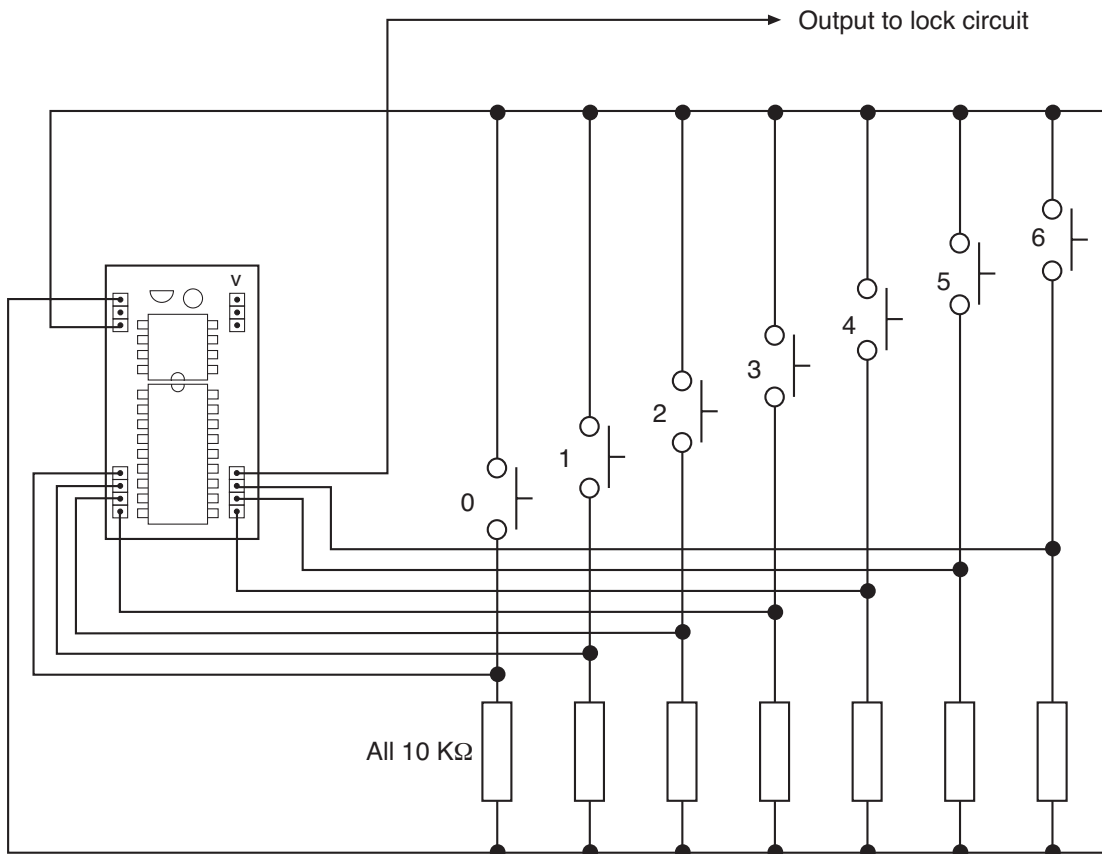
LDR sensors could be used to make the buggy seek out a light source or follow a white line.

Four switches could give a 'remote control' function to the buggy allowing a different manoeuvre to be carried out when each one is pressed.

### A CODE LOCK

The TEP PLC Chip could be used to produce a code lock where the combination could be changed simply by changing the program.

Seven of the inputs could be used leaving the eighth as an output to activate a solenoid in the lock mechanism.



## SAMPLE PROGRAM

```

lock:   low 7
        input 0
        input 1
        input 2
        input 3
        input 4
        input 5
        input 6
        goto code

code:   if pin4 = 1 then goto code2
        goto code

code2:  if pin0 = 1 then goto code3
        goto code2

code3:  if pin6 = 1 then goto code4
        goto code3

code4:  if pin3 = 1 then goto open
        goto code4

open:   high 7
        pause 5000
        goto lock

```

This is how this simple code lock program works.

```

lock:   Sets the output pin7 to low which locks the door. It
        then sets pins 0 to 6 to inputs so that they are ready
        to receive the code.

code:   Waits for input pin4 to be high before moving to
        code2.

code2:  Waits for input pin0 to be high before moving to
        code3.

code3:  Waits for input pin6 to be high before moving to
        code4.

code4:  Waits for input pin4 to be high before moving to
        open.

open:   Sets output pin 7 high for 5 seconds to open the
        door then returns to the beginning of the program at
        lock.

```

The code is set by the code procedures in the program. The code for this lock would be **4063**. You could change the code by changing the pin numbers in each of the code procedures. You could add more numbers into the code simply by adding more code procedures into the program.

This is a very simple lock program. Its biggest problem is that it does not do anything if the wrong switch is pressed. If someone continually pressed all the switches in any sequence then the lock would eventually open. To make the lock program more effective you need to add procedures that stop this from happening. This could be done by sending the program back to the start if the wrong switch is pressed. You could do this by adding a line similar to these to each of the code: procedures,

```
code:  if pin4 = 1 then goto code2
       if pin0 = 1 then goto lock
       if pin1 = 1 then goto lock
       if pin2 = 1 then goto lock
       if pin3 = 1 then goto lock
       if pin5 = 1 then goto lock
       if pin6 = 1 then goto lock
       goto code
```

Any of these basic ideas can be improved upon, adapted or combined. The TEP PLC Chip will offer you many opportunities to produce very sophisticated control systems using quite basic electronic circuits. The secret of the system lies with the programming of the TEP PLC Chip. Perhaps the only limit to what you will be able to achieve is your imagination! Be creative, experiment and enjoy...