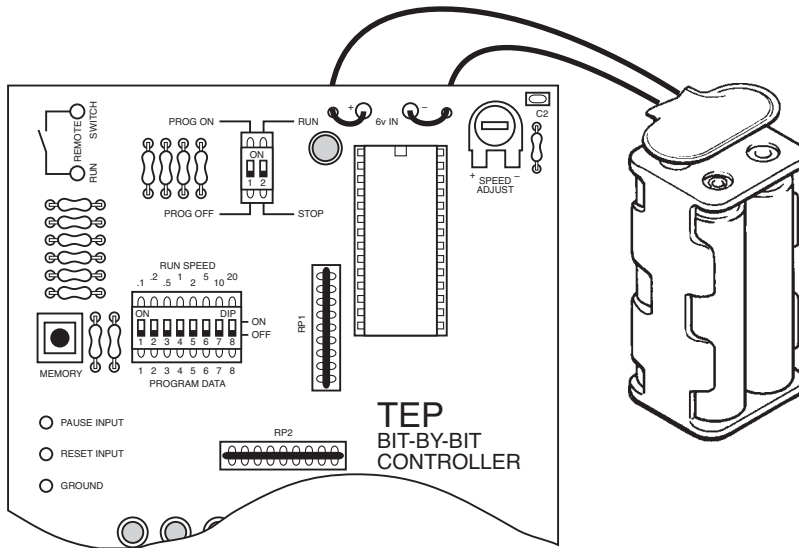
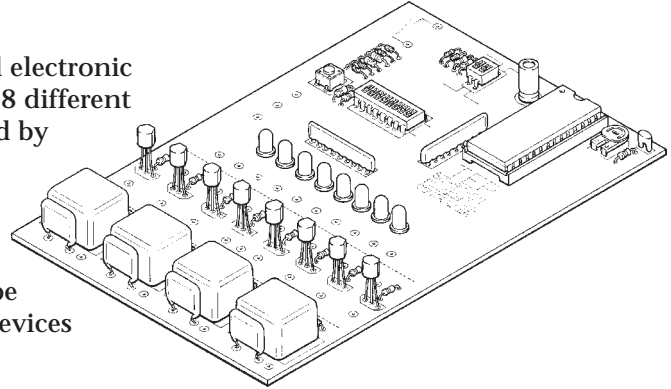


THE TEP BIT BY BIT CONTROLLER

INTRODUCTION

The TEP bit by bit controller is a self-contained electronic controller capable of switching on or off up to 8 different outputs over a period of time. It is programmed by setting each of eight small DIP switches to either 'on' or 'off' and then committing these instructions to memory by pressing a push switch. The memory can hold up to 64 such lines of program. The total program can then be run at different speeds to control a variety of devices such as lamps, buzzers and electric motors.



SETTING UP

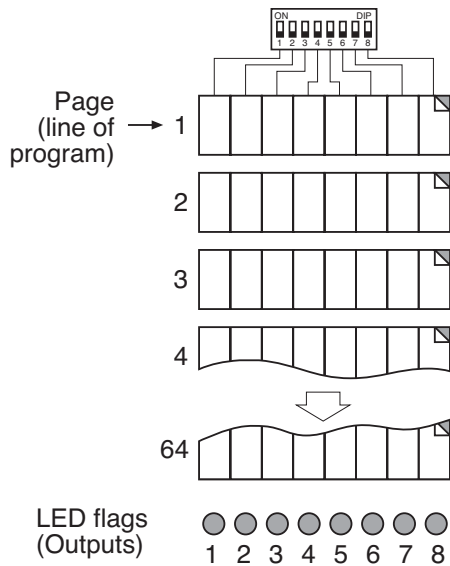
The controller is supplied complete to run and program; output components are added if and as required. The controller requires either a **6V** battery power supply or a supply from a PSU (power supply unit) which is regulated. **THE MAXIMUM SUPPLY VOLTAGE IS 6 VOLTS. IT SHOULD BE NOTED THAT A BATTERY SNAP CAN INADVERTENTLY BE CONNECTED TO A 9V SOURCE.**

A 4 × AA battery box and battery connecting snap is supplied with the board. If used, the snap should be soldered to the points marked + and - at the top of the board - if necessary using the two spare holes as mechanical anchorage for the two leads.

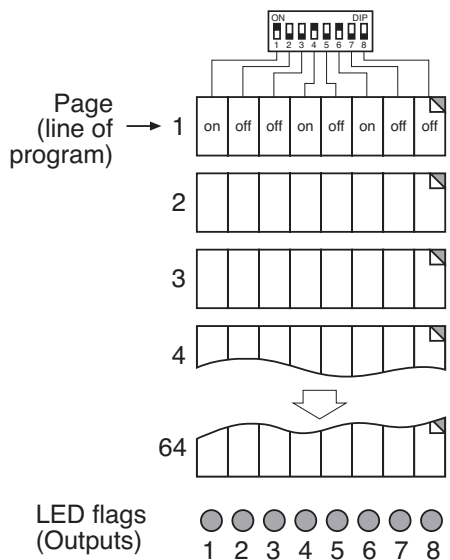
Any program will be lost if the battery is disconnected for more than 20 seconds. Because the standby current consumption of the board is so small, it is preferable to leave the battery connected all the time.

BASIC PRINCIPLES

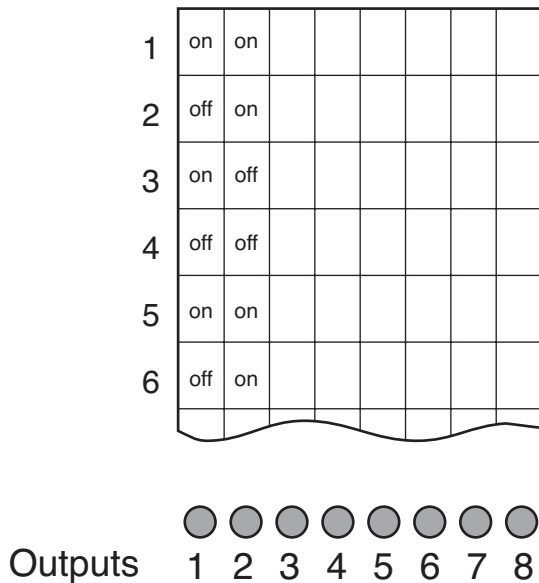
The TEP controller uses a single IC (integrated circuit) containing a memory where information can be stored in electronic form. It is useful to think of this memory as a book having a stack of pages. Every page represents a line of control programming and has 8 blank spaces - each one waiting to be filled with a bit of information. Each *vertical* column of blanks will contain the remembered instructions for a control output. Each control output is connected to an LED 'flag'.



The memory locations are filled with individual *bits* of information - of which there are only two types: logic 1 or logic 0. In the controller's memory these are really instructions which mean either turn ON an output (logic 1) or turn OFF an output (logic 0).



The information is written onto each 'page' of memory by setting the 8 DIP switches to either 'ON' or 'OFF' and then pressing the 'MEMORY' press-button switch . Pressing this button 'writes' the PROGRAM DATA switch settings into memory and automatically turns over to the next 'page'. This procedure can be repeated up to 64 times - the maximum number of pages or locations in the controller's memory.

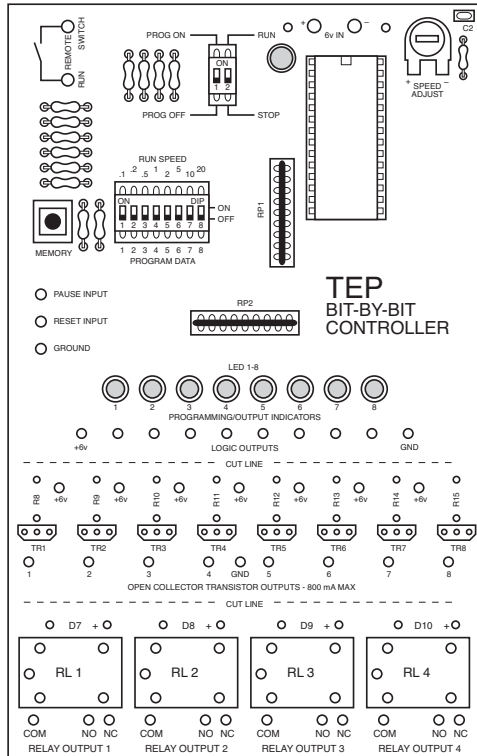


The illustration shows a sample 6-line program for the two left hand outputs. When the controller is instructed to read this program, it turns over the 'pages' at a set speed. An 'ON' bit of information lights up an output LED and an 'OFF' bit turns it off. If, for example, the controller is set to read each page for a second at a time, the LED on the far right hand side will turn on for one second off for the next and so on. LED number 2 will turn on for two seconds and then stay off for two seconds. When the program has been run - i.e., all the 'pages' turned over - the controller will automatically start again at the first line of the program. Unless it is stopped, the program will run over and over again.

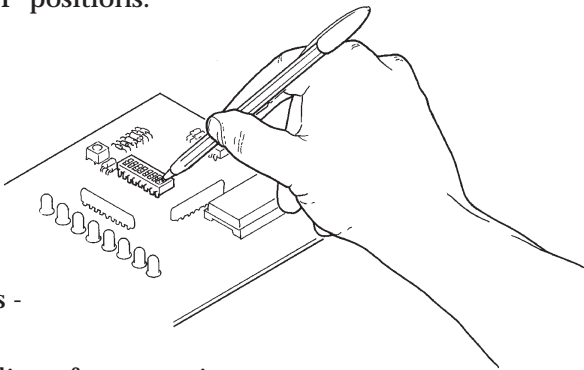
Please note: the remainder of this text will refer to **lines** of program and not pages.

PROGRAMMING THE BIT BY BIT CONTROLLER

Using the whole-board diagram as a guide, you should now be able to follow these instructions for programming the controller.



1. Make sure the RUN and PROGRAM switches at the top of the board are set at the 'PROG OFF' and 'STOP' positions.
2. Connect the battery or power supply.
3. Set the program switch to 'PROG ON'.
4. Write a line of program by setting each 'PROGRAM DATA' switch to either 'ON' or 'OFF'. This will turn the LED outputs on or off. Because the program switches are small, it is more convenient to operate them with a stylus - e.g., the tip of an empty pen.
5. Press the 'MEMORY' switch to write this line of program into memory. When you do so, all of the LEDs will flash on briefly to confirm this has happened.
6. Repeat steps 4 and 5 above up to 64 times - once for each line of memory. If you try to go beyond 64 lines of programming, the extreme left hand LED will flash continuously.

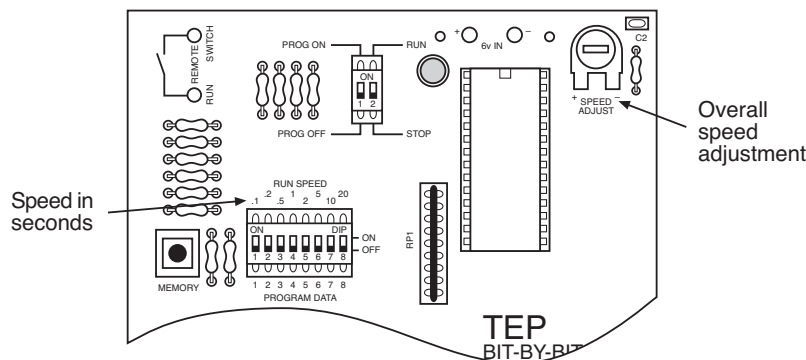


There is no problem if you write a program less than 64 lines. When the program is run, it will loop back to the beginning after the final line.

RUNNING THE PROGRAM

1. Switch the programming switch to 'PROG OFF'.
2. Set all the 'PROGRAM DATA' switches to the 'OFF' position.
3. The 'PROGRAM DATA' switches *will now control the program run speed*. As an example, set the fourth switch from the left to 'ON'.
4. Set the program run switch to 'RUN'. The program will now run at approximately 1 line per second. (Setting one of the other 'PROGRAM DATA' switches will run the program at a different speed - see below.) The LEDs will turn on and off according to the stored program in memory.

The speed of execution of the program depends on which 'PROGRAM DATA' switch is set to the 'ON' position and also on the setting of the 'SPEED ADJUST' resistor at the top of the board. The 'PROGRAM DATA' switches provide speed adjustment in fixed steps or ratios. The 'SPEED ADJUST' resistor provides *overall* continuous adjustment - faster or slower. To calibrate the controller to run at the speeds printed above the 'PROGRAM DATA' switches, create a simple program that turns LED 4 on for one program line, off for the next - and so on (keeping all the other LEDs off). Run this program, and time the result against a watch - altering the 'SPEED ADJUST' so that eventually the LED turns on and off at one second intervals.



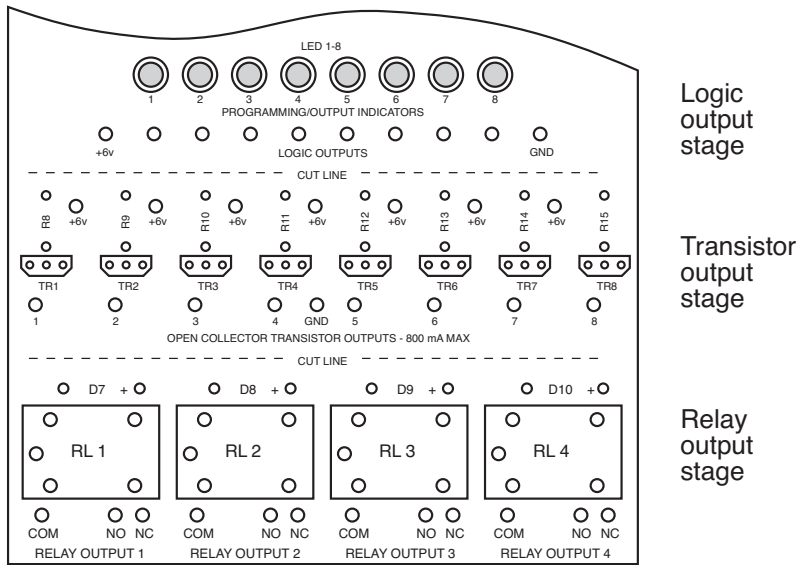
If setting switch 4 provides a run speed of one program line per second, the switch on the far right will give program steps of 20 seconds duration. This adds up to a maximum 64 line program run time of 20 seconds \times 64 lines = 1,280 seconds OR approximately 21 minutes. This run time can be extended further by adjusting the resistor. Remember, though, this also affects timings provided by the other 'PROGRAM DATA' switches.

Important note: The TEP controller has a volatile memory. This means that a program is lost when the power supply is disconnected although the larger capacitor at the top centre of the board will keep it energised for about 20 seconds. However, the Standby Current Consumption of the controller's chip is so low it can be left connected for most practical purposes.

Technical note: capacitor C2, together with the two resistors at the top right hand corner of the board, controls the chip's clock speed. This is 390 pF. If it is replaced with a lower value (no lower than 50 pF) the top run speed can be considerably increased. However, it will also have the effect of flashing the LEDs more rapidly when the memory button is pressed and the standby current consumption will increase slightly.

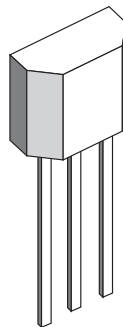
USING THE CONTROLLER'S OUTPUTS

The bit by bit controller has 8 LED *flags* to show the status of each output. This enables you to create programs and run them but not to actually control anything! To switch a *load* such as a motor on or off a *buffer* stage has to be added to each output in use. For convenience, the controller board has additional printed tracks and locations for transistor buffers on all the outputs and transistor-plus-relay buffers on four of them. (Note: the board has only enough room physically for relays on the four left hand outputs.)



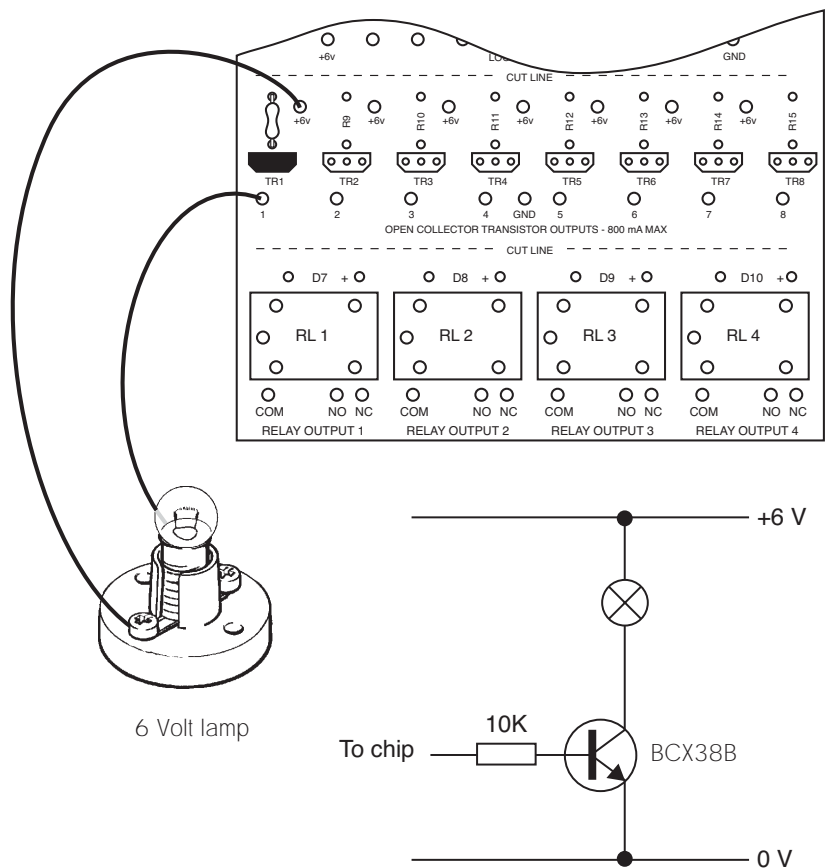
USING THE TRANSISTOR OUTPUTS

The recommended output transistor for which the board has been designed is the inexpensive BCX38B. This is a Darlington pair device and will switch on a load of nearly 1 amp (800 milliamps maximum). This is quite sufficient for most filament bulbs, buzzers and a solar motor.



BCX38B

To add a transistor to any required output, fix and solder in position a 10 K resistor and BCX38B transistor - for example, at the positions marked R8 and TR1. Make sure the transistor is the correct way around by matching the case outline with the outline on the board. Flying leads to the load are soldered to the +6 V point and open collector output for each transistor. The diagram shows a lightbulb connected to output 1. A circuit diagram for this output is also shown.



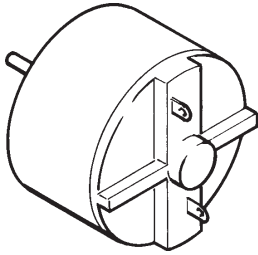
REMEMBER that when several transistors are used, the total load current - which can be quite high if all outputs are used - comes from the battery powering the controller. This could be depleted very quickly. Always work out the total load current (or an average for outputs switching on and off) and think carefully about the type of battery needed.

It is possible, for example, to run the following devices directly from the transistor outputs:

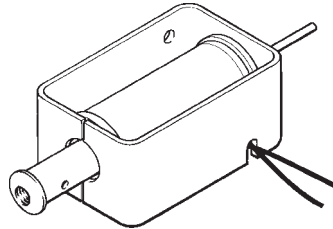
- filament bulb* *buzzer* *solar motor*
- miniature solenoid* *stepper motor (see page 18)*

Any motors other than the more expensive solar motor should be run from a relay (see page 12). This is because they produce a high degree of electrical noise which may interfere with the operation of the chip.

A motor, solenoid or any other device with a coil is an *inductive* load and can produce a high voltage momentarily when switched off (back EMF).

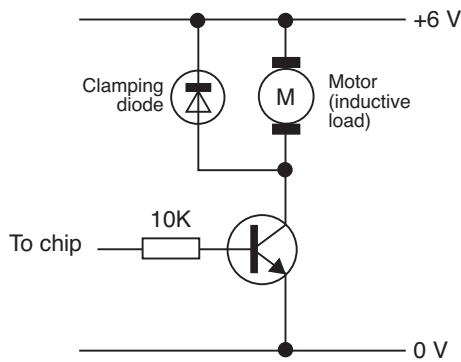


Solar motor

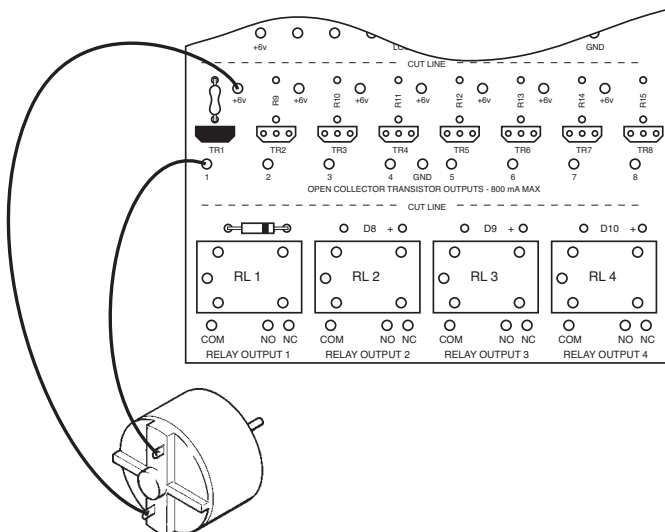


Miniature solenoid

To prevent this damaging the transistor, a clamping diode should be added as shown in the diagram. This can be a general purpose type such as IN4001.

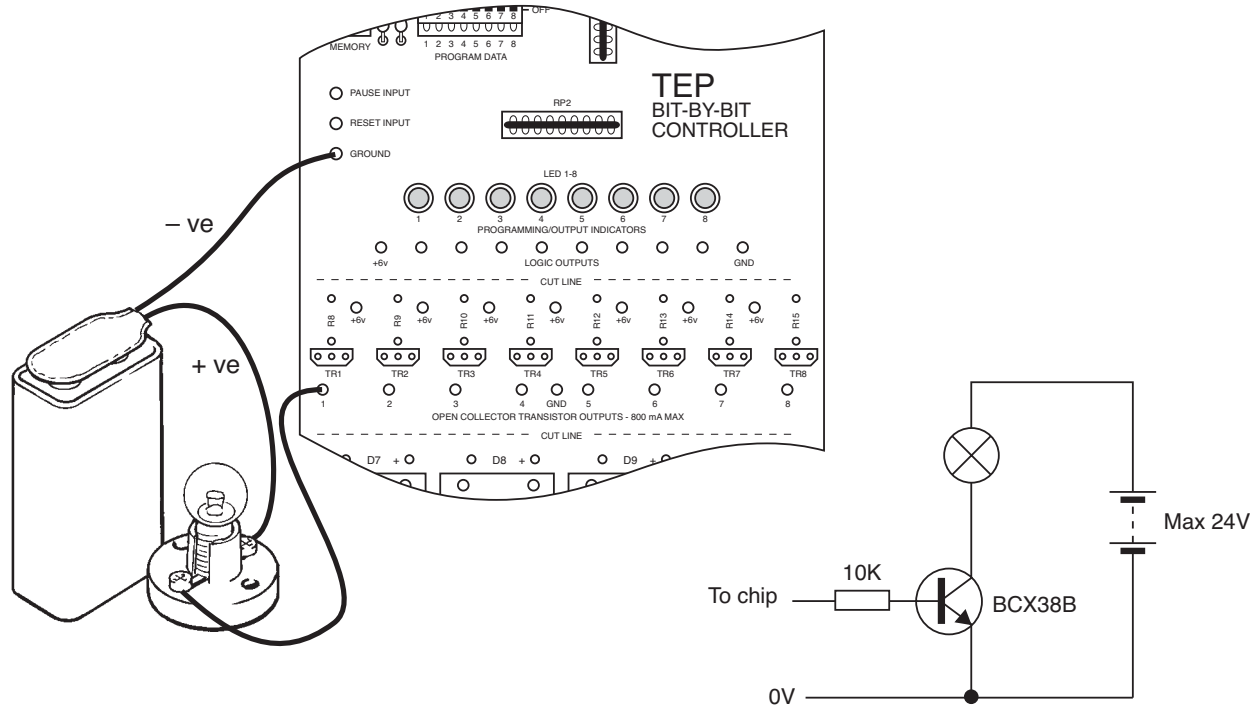


The most convenient way of connecting a clamping diode to an inductive load is to use one of the first four left hand outputs and simply solder in a diode *as if you were using a relay*. It is **IMPORTANT** to ensure that the diode is soldered in the correct way round - with the marked end facing towards the right.



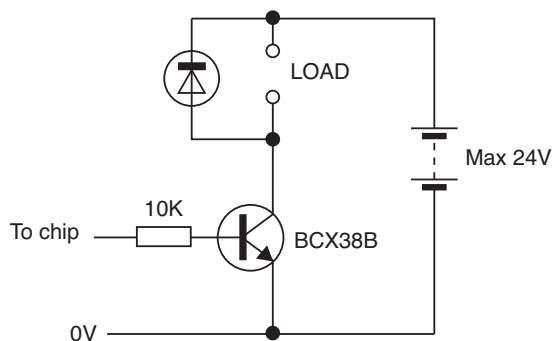
Using a transistor output with an external power supply

To avoid draining the battery powering the board itself, a load can be connected to a separate power supply, ideally a battery, up to 24V. The diagram shows a lightbulb thus connected to output 1. A circuit diagram for this output is also shown.



Remember that if a separate battery is used, the total load current should not exceed 800 mA otherwise the transistor will be damaged.

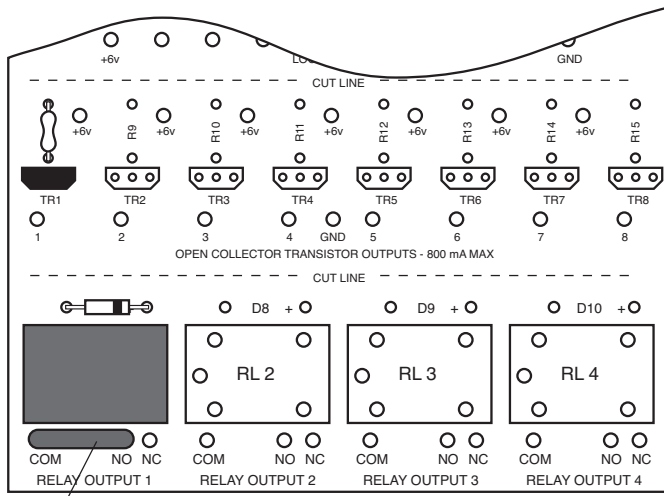
Please remember that if an inductive load, such as a motor, is connected, a clamping diode should be added as shown in the diagram below.



The easiest way to add this might be to connect it directly across the load itself; i.e. across the connecting legs of a motor in parallel with the suppression capacitors.

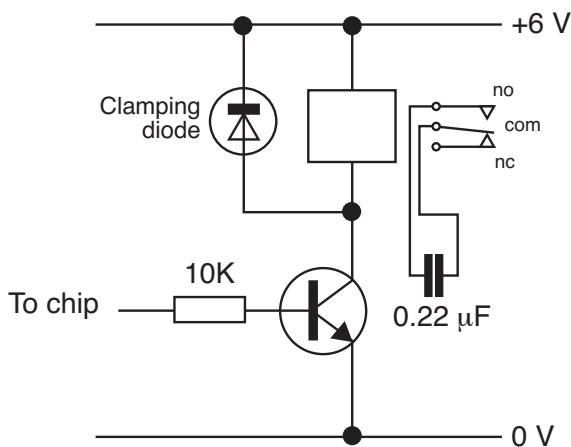
USING A RELAY OUTPUT

The first four left hand outputs are extended at the bottom of the PCB to accommodate relays. To add a relay to any of these outputs, first fix and solder in position a 10 K resistor and BCX38B transistor - for example R8 and TR1. Then fix and solder in position a miniature SPDT relay (e.g. Kam Ling KS1P). Also, solder in a 0.22µF suppression capacitor across 'com' and 'no'. **This is essential.** To test the relay, write in a simple on/off program for this output. When the program is run, the relay will simply click on and off.



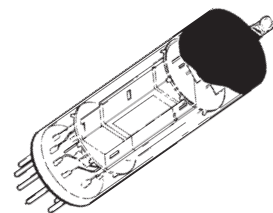
0.22µF capacitor

The load is connected to the relay switch outputs at the bottom of the controller board. 'COM' is the pole of the switch, 'NO' is the normally open contact and 'NC' is the normally closed contact. **The load leads are either soldered to the capacitor legs on the top of the board or the solder points below it.**

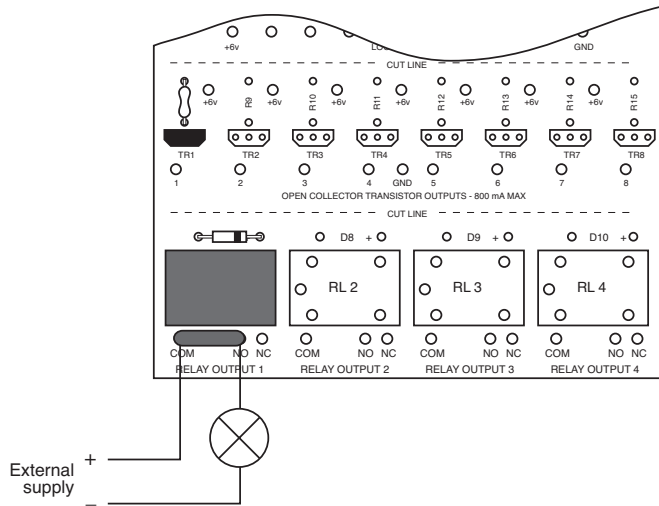


Technical note: a very significant problem in designing industrial control equipment is the suppression of 'electrical noise' or electromagnetic interference. Electromechanical devices invariably produce interference. (Remember that early radio transmitters used an electrical arc to produce radio frequency energy.) The suppression capacitor on the relay is very important to prevent any interference to the chip. European Countries have a convention - EMC - which sets a standard for protection from electromagnetic interference.

Curious fact: the detonation of a nuclear weapon produces a massive electromagnetic 'spike' capable of immobilising chip based equipment. Designers of military communications equipment have actually considered going back to using valves to avoid this problem.

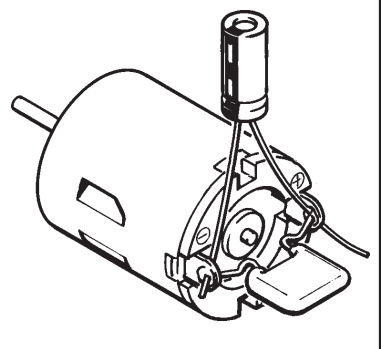


When connecting the load, *treat the relay as an ordinary switch*. Its switches are NOT connected electrically to the controller's power supply and you will need to add an external power supply. You should avoid exceeding the stated values on the relay.

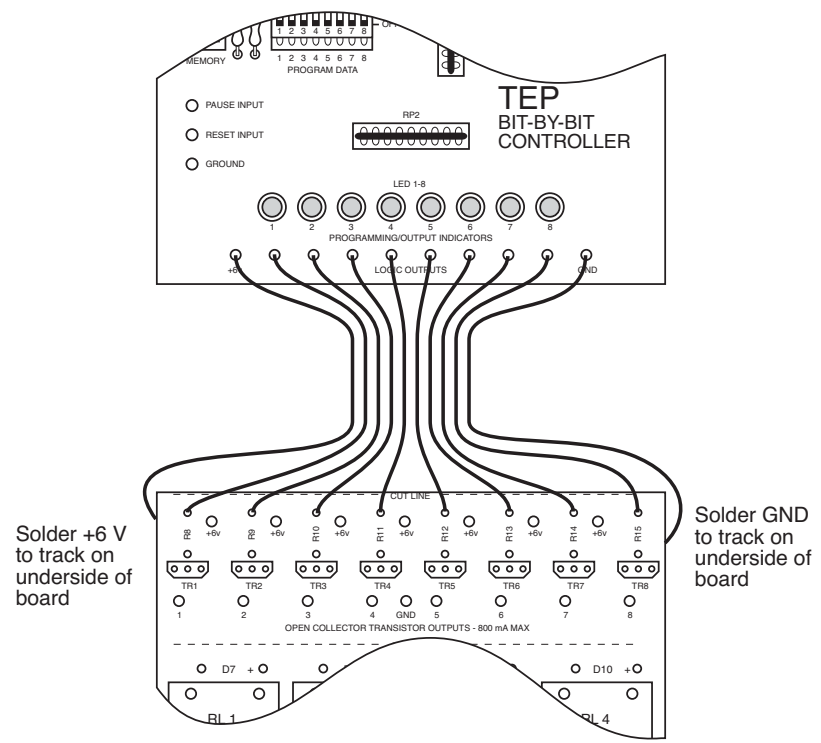


Technical note
To avoid electrical interference **any small electric motor** other than a solar motor **must be suppressed using two capacitors** as shown - 0.22µF ceramic, 10µF electrolytic.

It is important to ensure that the electrolytic capacitor is correctly connected to the power source. The side marked should be connected to -ve.



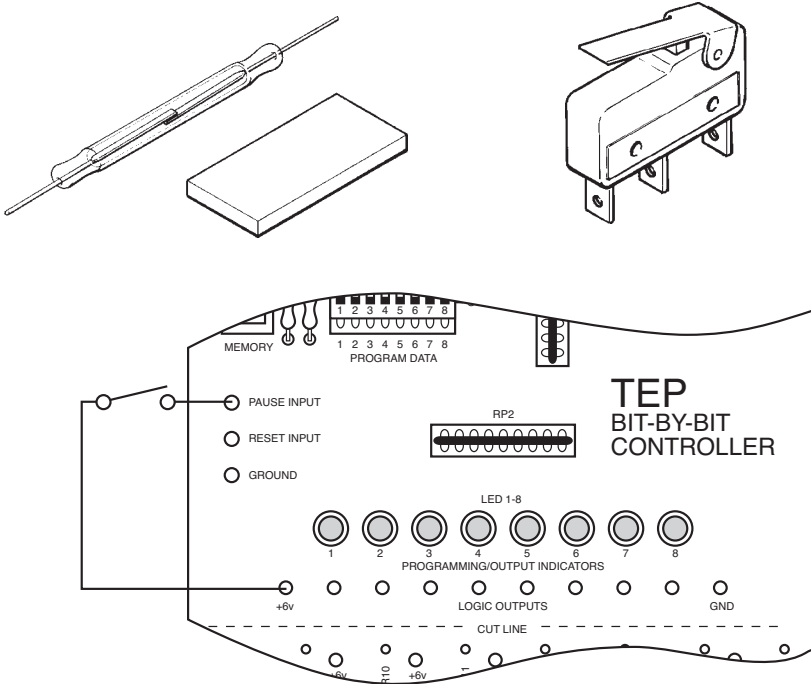
The output stages of the controller board may be cut off as indicated either if they are not wanted or because *the output stages are to be placed elsewhere in use*. For example, the controller might need to be built into a very tight space. If the output part of the board is separated, the +ve and -ve rails must be connected between the two board halves together with a *single wire link* for each output used. Multi-coloured ribbon cable is a useful option for making a number of connections between boards.



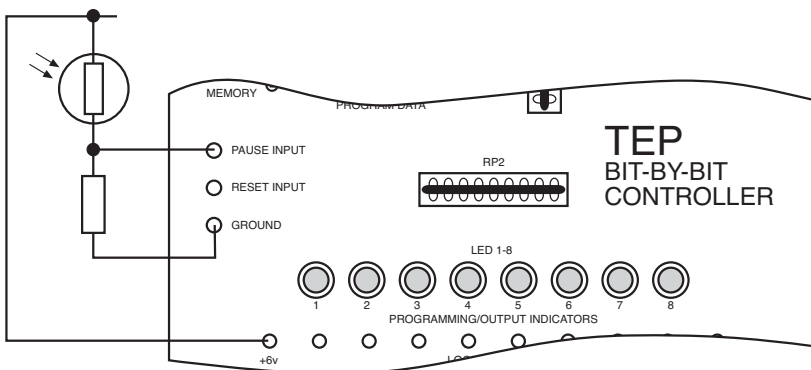
USING THE PAUSE AND RESET FACILITIES

Pause Input

A program can be interrupted by connecting the 'PAUSE INPUT' to any +6v point on the board. At the instant this connection is made, the program stops running. It continues from the point at which it stopped when this connection was broken. Any kind of switch can be used to operate the pause facility - e.g. reed switch, micro-switch etc.



The pause input actually only needs a logic level voltage above +2.5 volts to halt the program. It is therefore possible to use a sensing circuit based on a potential divider. An example light sensing circuit is shown. When no light falls on the LDR, its resistance is high and the voltage at the mid point of the potential divider is below the +2.5 volt threshold. As light falls on the LDR, its resistance drops and the voltage at the mid point rises. A variable resistor in the potential divider enables adjustment for different light thresholds.



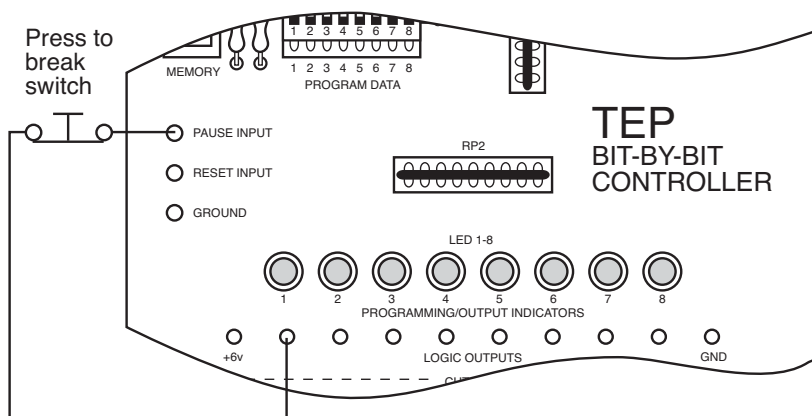
(Please note that this kind of input circuit will not necessarily provide the controller with a rapid or clean transition voltage. This problem may be overcome by adding a *schmidt trigger* to the potential divider circuit.)

Any additional input circuits, such as the one described, will probably use the controller's power supply. For example, a logic circuit might be connected which causes the program to be interrupted if one or more switches connected to an AND gate are closed. An *externally* powered circuit can be connected to the 'PAUSE INPUT' **providing that the external voltage does not exceed 20 volts**, and that the 0V rail of the external circuit is connected to the 0V rail of the controller.

Reset Input

This input instantly resets the program to the beginning when 'RESET INPUT' is connected to any +6v point on the board. To use the reset function, simply follow the rules described above for 'PAUSE INPUT'.

NOTE: Both the 'PAUSE INPUT' and 'RESET INPUT' facilities can be activated by connecting them to any one of the 'LOGIC OUTPUT' points. This means, for example, that output 1 could be programmed with one or more 'ON' instructions to interrupt an entire program at different points until - say - the connection between the 'LOGIC OUTPUT' and 'PAUSE INPUT' is broken by momentarily pressing a switch. In many consumer products - e.g. washing machines - a control program interrupts until the operator manually instructs it to continue.



Output controls the pause input

Remote Run Switch

The Remote Run Switch connections parallel the on-board program Run/Stop switch. A switch can be connected using flying leads for control at a distance from the board.

PROGRAM PLANNER

This planner allows you to write a program by marking each of the bit boxes. Block in boxes for 'on' and leave them blank for 'off'.

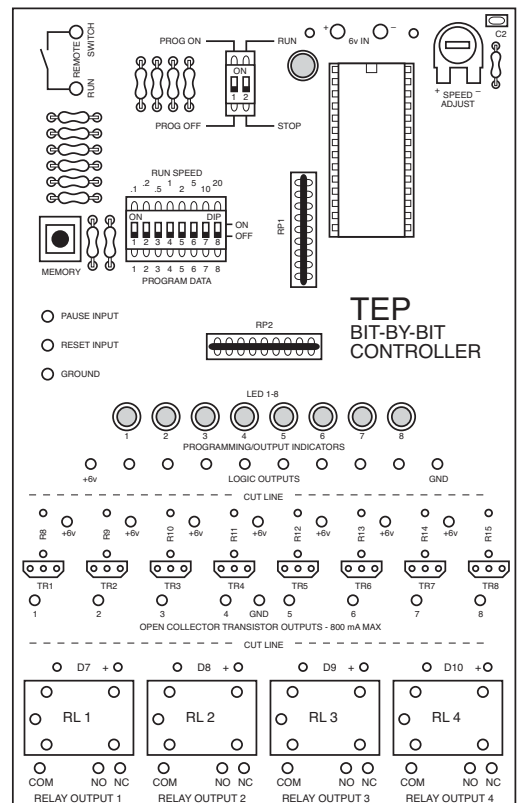
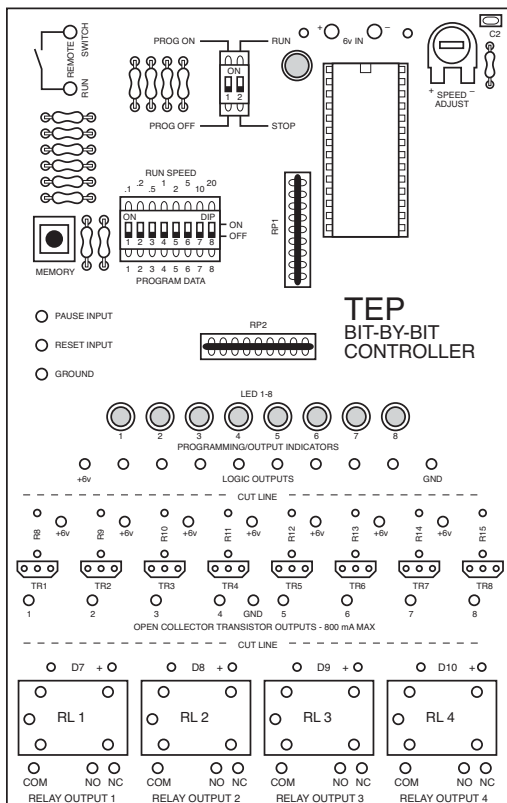
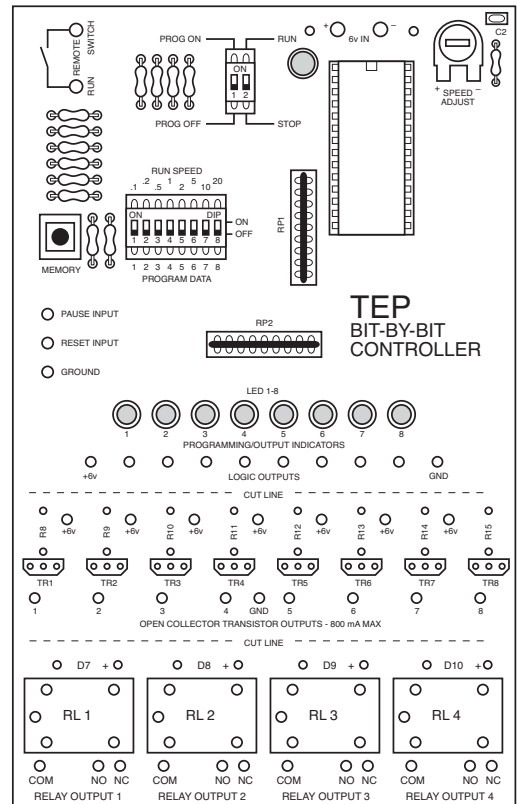
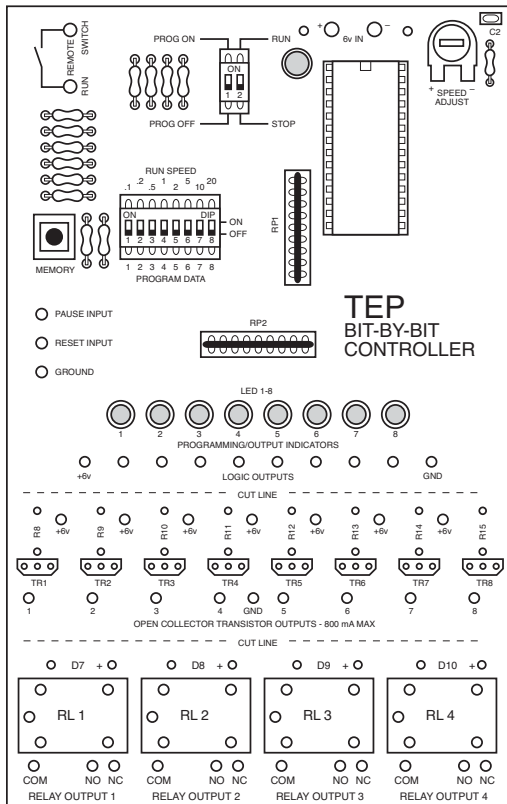
The EVENT column gives you space to make a note of what is going on during any line of programming.

OUTPUTS CONNECTED TO:

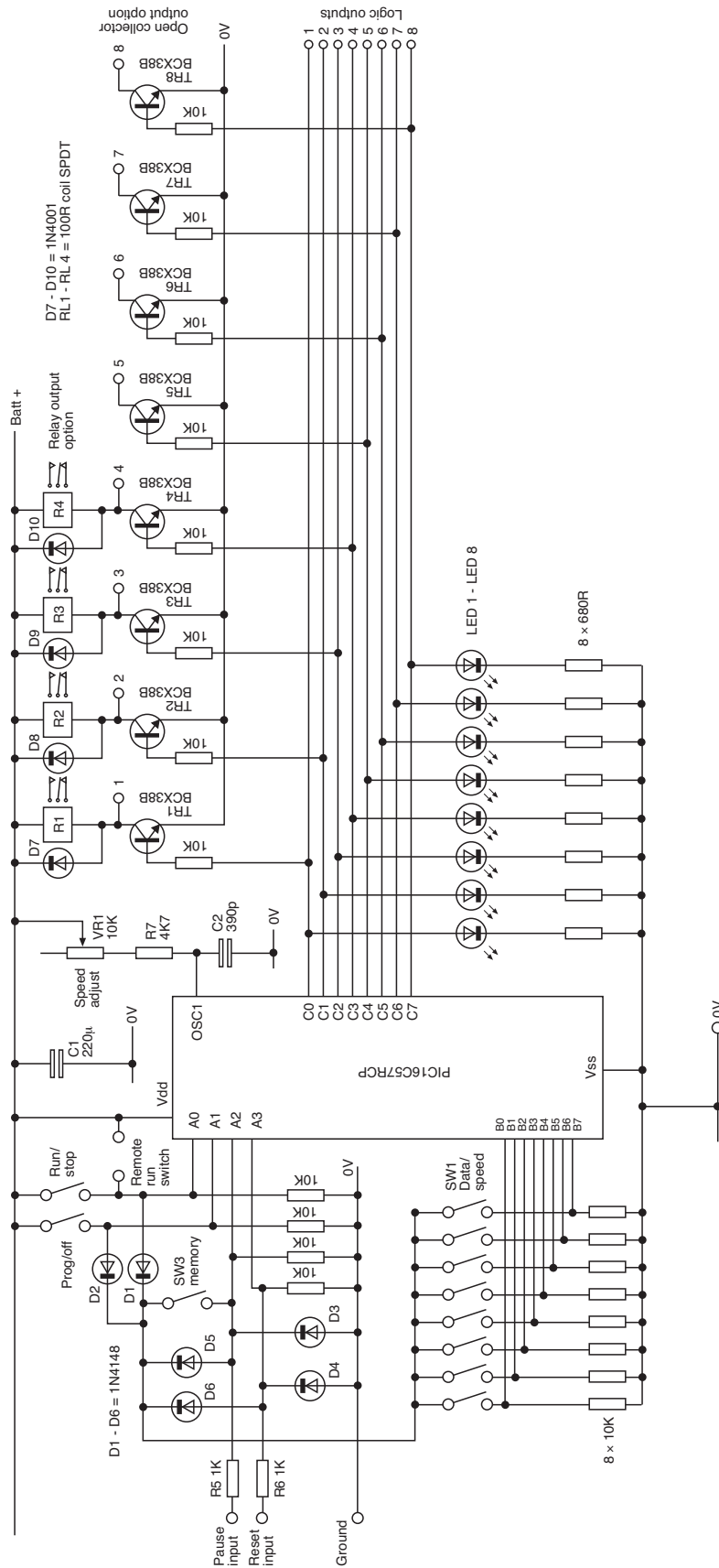
1. _____
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____

		OUTPUTS								EVENT
		1	2	3	4	5	6	7	8	
1										-----
2										-----
3										-----
4										-----
5										-----
6										-----
7										-----
8										-----
9										-----
10										-----
11										-----
12										-----
13										-----
14										-----
15										-----
16										-----
17										-----
18										-----
19										-----
20										-----
21										-----
22										-----
23										-----
24										-----
25										-----
26										-----
27										-----
28										-----
29										-----
30										-----
31										-----
32										-----
33										-----
34										-----
35										-----
36										-----
37										-----
38										-----
39										-----
40										-----
41										-----
42										-----
43										-----
44										-----
45										-----
46										-----
47										-----
48										-----
49										-----
50										-----
51										-----
52										-----
53										-----
54										-----
55										-----
56										-----
57										-----
58										-----
59										-----
60										-----
61										-----
62										-----
63										-----
64										-----

BOARD LAYOUT



TEP BIT BY BIT CONTROLLER CIRCUIT DIAGRAM



MIDDLESEX UNIVERSITY TEACHING RESOURCES

Components and Stock Numbers for TEP Controller and Accessories

EW2 011	Solar motor
ET1 004	BCX38B transistor
EW2 017	SM42 stepper motor
EC4 007	0.22 μ F ceramic capacitor
EC4 012	10 μ F electrolytic capacitor
ED1 001	1N4001 diode
ER2 10K	10K resistor (Pack of 100)
EW2 003	Kam Ling SPDT relay