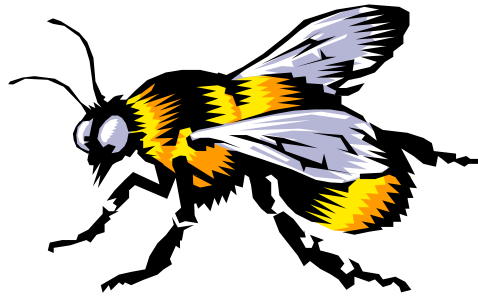


# MINI-BEE

**Automation Adaptor**

**Installation and Users Manual  
(Including Bee-Step14 Software)**

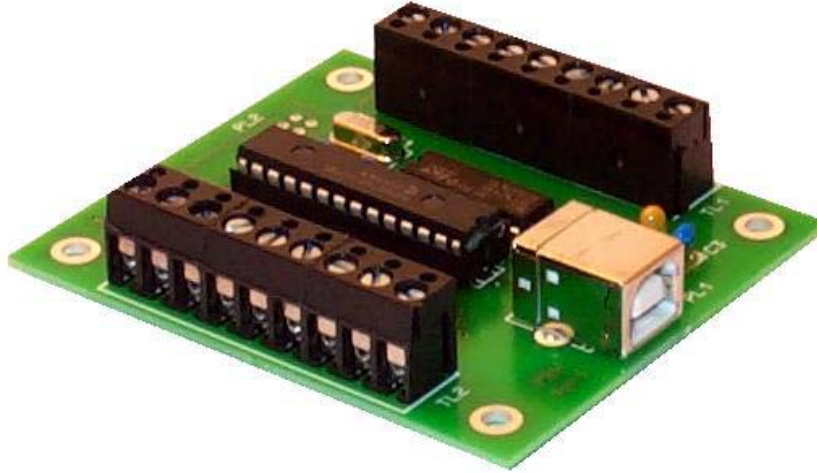


# Contents

- 1 Introduction
- 2 Hardware Installation
- 3 Bee-Step/4 Application Software
  - 3.1 Overview
  - 3.2 Bee-Step/4 Main Screen
    - 3.2.1 Main Sequence Display
    - 3.2.2 Step Editing
    - 3.2.3 Run Control
- 4 Connecting Devices to Mini-Bee
- 5 Pinout of the High Voltage Switching Outputs On Screw Terminals(TL1&TL2)
- 6 Writing Your Own Software to Use Mini-Bee
- 7 Minimum PC System Requirements

## 1. Introduction

The MINI-BEE is a versatile USB adaptor, which allows the PC User to explore the world of real time control and automation. It is a tool, which is attractive to both the novice and experienced user.



For the absolute beginner it can be used straight from the box as a flexible controller for a wide range of projects. The beginner can take advantage of the ease of connectivity of USB, making connection to the PC simple. Screw terminals mean that external devices to be controlled can be attached without any need for soldering. The included *Bee-Step14* application software allows the beginner to quickly create elaborate control sequences without any need for prior programming knowledge or PLC techniques.

For the intermediate user a DLL is provided to allow the programmer to construct their own software applications to take advantage of the MINI-BEE hardware without having to know the details of USB communication protocols etc..

## 2 Hardware Installation

Simply connect the MINI-BEE to any available USB port (*This will require a standard USB cable*). Although it will operate from bus powered hubs it is recommended that you connect it to a primary USB socket or a self powered hub. This allows MINI-BEE to take full advantage of the available 500mA from such a connection. Bus powered hubs are limited to 100mA. Windows operating system will automatically detect and install the appropriate device drivers. The MINI-BEE is regarded by Windows as a standard HID (Human Interface Device) which makes it very easy to install.

### **3. Bee-Step14 Application Software**

#### **3.1 Overview**

When the Bee-Step14 application is started, click on the “Run” menu option to initiate the control dialog box (shown below). The first time Bee-Step14 is run the customary disclaimer agreement will ask for your agreement. Click “I Agree” to never see it again and to start Bee-Step14.

Bee-Step14 provides the user with a means to edit, save, restore and run a sequence of “patterns” of outputs that will be used to directly control outputs on the attached MINI-Bee adaptor board. This is known as a Directed Sequencer. The sequence is made up of up to 10000 individual “steps”. Each step has three elements

- A duration:
- A pattern of outputs (on or off for each of the outputs)
- The number of the next step to be executed

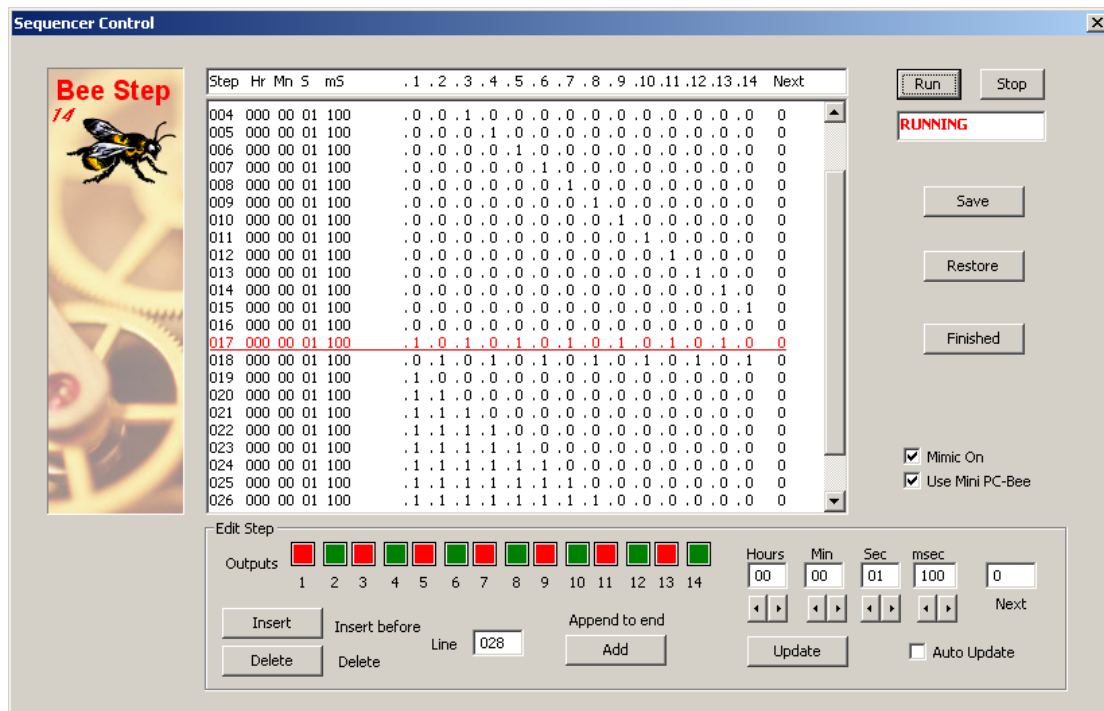
The duration is specified in Hours, Minutes, Seconds and milliseconds. The overall resolution of the time interval is 0.1 seconds. This provides a realistic level of control for devices in the “real world” of automation and robotics.

The pattern of outputs corresponds to the on / off status of the actual control outputs on the MINI-Bee adaptor. All outputs are independent and any desired pattern can be used for each step.

A simple sequence controller sets the outputs for each step in turn finishing when it reaches the end of the list. Bee-Step14 can operate this way if you simply ignore the “next” element of the step. However, to create more elaborate sequences including continuous loops, you can use the “next” element to specify which step to go to when the current step is complete (i.e. duration reached).

#### **3.2 Bee-Step14 Main Screen**

When you first run BeeStep14 you will start from the main menu screen. Click on the menu item “Run” to open the sequence editor. We can now look at each of the elements of the screen in turn....



### 3.2.1 Main Sequence Display Window Sequence Summary List

The sequence summary window shows each individual step in a programmed sequence in summary form. Each line in the window corresponds to a single step in the sequencer. On the left of each line is the line number. This will always be consecutive from 1 to your last entered line. When editing the sequence list using insert or delete (discussed later), the line numbers will be automatically adjusted so that they remain in strict numerical sequence. This line number is used as the reference for the “next” facility (also discussed later) to control sequence loops and jumps. Immediately to the right of the line number is the duration. This is split into Hours, Minutes, Seconds and Milliseconds. The smallest duration is 100msec (0.1sec). The next 14 columns show the on / off status that the MINI-Bee board outputs will use during this step. On is shown as ‘1’ and off as ‘0’. On the right hand end of the line is the “next” selection. This determines which line will be used after this one. A value of 0 in the next position indicates that the next line immediately below the current one will be used. For sequences larger than the sequence summary window can display, the window may be scrolled up and down using the scrollbar on the right

### 3.2.2 Step Editing

The step editing section immediately below the summary window allows individual steps to be added, deleted and edited. Each of the elements of a step is available in this area in editable form. The row of boxes along the top corresponds to the step’s on/off pattern of outputs. Green is off and red is on. Clicking on the box will toggle it between on and off. The duration of the step has individual edit boxes for hours, minutes, seconds and milliseconds. Using the adjacent arrows allows new values to be specified. In a similar way the value of the next step to be run can be edited. Note that the last step in a sequence will always cause the sequence to stop even if the “next” box is set to another location (eg to loop back). To avoid this add a dummy step after the one in which you wish to loop back.

To edit a step that already exists in the summary window simply click on that line in the window. This will cause a red underline to appear at that line and a copy of the step elements to appear in the edit area. Once editing is complete you can update the step in the sequence list by pressing the “Update” button. Alternatively you can check the “Auto Update” box and see the changes you make updated immediately.

To create a new step to be added to the end of the current sequence make sure the “Auto Update” box is unchecked, construct your step in the edit area and then click on the “Add” button.

To insert a new line somewhere in the middle of the current sequence, again make sure the “Auto Update” box is unchecked, click on the line BELOW the intended point of insertion and, when your editing of the step is complete, click on the “Insert” button.

To delete a line currently in the sequence list, simply click on the line and press the “Delete” button. To preserve the consecutive line numbering, all lines below the deleted line will be automatically moved up one and correspondingly re-numbered. Note that step 0 in the sequence list cannot be deleted or have other steps inserted above it.

Once a control sequence has been created to your specifications it can be saved to hard drive using the “Save” button. Files created in this way can later be restored using the “Restore” button.

### **3.2.3 Run Control**

On the right of the main screen there are two buttons for run control, which are virtually self-explanatory. Clicking “Run” starts execution at the first step (0) in the sequence. It will continue to “run” until either the last step is reached or the “Stop” button is pressed. If the stop button is pressed, the current state of the outputs will be maintained. Care should be taken when arbitrarily stopping the running sequence since it may stop with an output permanently on that shouldn't be (for example it may leave a motor running continuously).

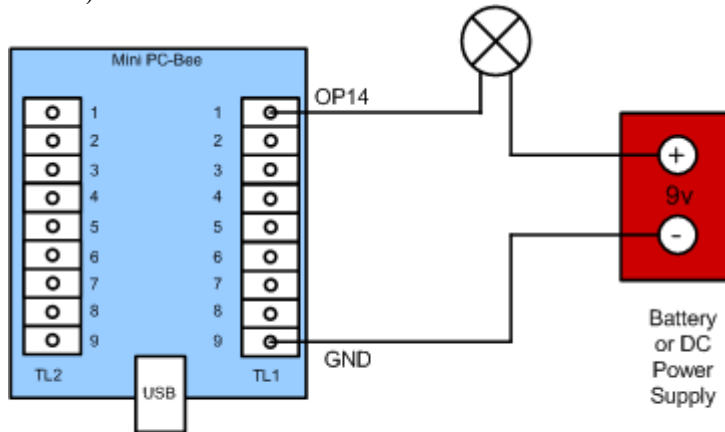
While running it is possible to see the current state of the outputs changing on the screen in real time mimicking the real outputs. This is selected by checking the “Mimic Display On” box. Normally there is no problem in running the mimic concurrently with the real time control sequence, however, if the control sequence involves very short duration steps and/or your PC is relatively slow it is better to turn off the mimic in favour of the real time control outputs.

It is good practice to test your sequence before actually applying it to the MINI-Bee outputs. To do this, simply uncheck the “Use Mini-Bee” box. The mimic will run but no outputs will change on the MINI-Bee board.

#### 4. Connecting Devices to MINI-Bee

The MINI-Bee board is designed to be flexible in its uses and has a number of electrical features that make it easy to use in many applications. The following descriptions are merely suggestions on suitable ways to use the unit.

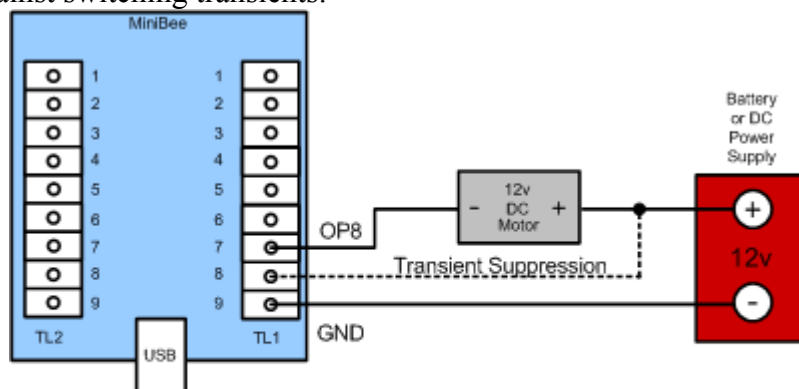
There are 14 *high voltage capable* DC switching outputs available on the screw terminals with each output taking the form of an “open collector” driver. For these outputs to operate correctly it is necessary to link the 0v (or Ground) connection of the MINI-Bee (screw terminals 9 and 18) to the 0v connection of the external power supply which is being used to “drive” the device to be controlled (eg a motor, solenoid, lamp etc...).



Connecting a Lamp to Mini PC-Bee

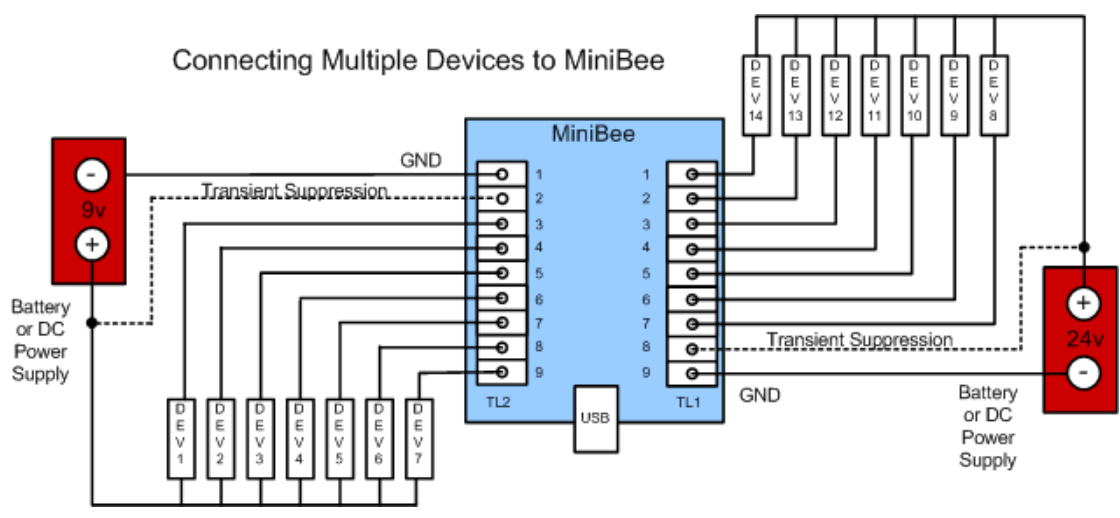
The device being controlled is then connected between one of the MINI-Bee control outputs (e.g. OP1 on Screw Terminal 1) and the external positive end of the supply (e.g. +12v).

When the MINI-Bee output is turned on (by the control step in BeeStep14 or by your own software) the terminal becomes a low impedance path to ground and current flows in the external circuit through the attached device. For low power non-inductive devices such as lamps, that is all that is necessary. If you are directly attaching an inductive load (such as a motor or relay) it is advisable to take precautions against switching transients.



Connecting a motor to MiniBee  
(making use of transient suppression)

Switching transients are spikes in the voltage that occur when an inductive load is turned off and can be high enough to cause damage to attached circuitry. MINI-Bee has built in facilities to suppress these transients by using suppressor diodes connected to two of the screw terminals (8 and 17). To make use of these simply connect either of these terminals to the external positive supply being used to “drive” your inductive load. Terminal 8 is intended for inductive devices connected to terminals 1 to 7 and terminal 17 for those connected to terminals 10 to 16. Care should be exercised if using different supply voltages for inductive loads. For example to operate both a 12v and a 24v dc motor using the MINI-Bee and using the transient suppression technique just described, you must ensure that you don’t have two different supplies on the same group of 7 outputs. i.e. in this case you could have the 12v motor on terminal 1, the protection terminal 8 connected to the +12v supply, and the 24v motor on terminal 10 with it’s protection terminal 17 connected to the +24v supply.



In summary , when using transient protection ensure you do not have more than one supply voltage on a given group of 7 outputs.

The output switching components used by MINI-Bee are DS2003 High Current / High Voltage Darlington Drivers. These devices are capable of being switched up to 50v and 350mA. However, although capable of these operating limits, the recommended application of the MINI-Bee is for voltages up to 24v. The current capability on each output is 350mA. As the DS2003 data sheet suggests, this can be extended by connecting outputs in parallel. If doing this, it is obviously essential that your control sequence ensures that all outputs that are paralleled are always in the same state (on/off) at the same time or you run the risk of one output taking all of the current and exceeding maximum limits. You must also take account of the overall power handling capability of the DS2003 when using multiple outputs each with high current. Refer to the graph of “peak collector current vs duty cycle and number of outputs” in the DS2003 data sheet included on the installation CD.

## 5 Connector Pinouts

Pinout of the High Voltage Switching Outputs On Screw Terminals(TL1)

Pin	Signal description
1	Switching Output 14
2	Switching Output 13
3	Switching Output 12
4	Switching Output 11
5	Switching Output 10
6	Switching Output 9
7	Switching Output 8
8	Transient Suppression
9	GND

Pinout of the High Voltage Switching Outputs On Screw Terminals(TL2)

Pin	Signal description
1	GND
2	Transient Suppression
3	Switching Output 1
4	Switching Output 2
5	Switching Output 3
6	Switching Output 4
7	Switching Output 5
8	Switching Output 6
9	Switching Output 7

For both terminals, terminal 9 is the one nearest to the terminal number label (i.e. TL1 or TL2)

## 6. Writing your own software for MINI-Bee

To use MINI-Bee straight from the box does not require any programming other than entering the step details into BeeStep14. However, if you prefer to design you own software then the following information will be of use.

Provided with MINI-Bee is a DLL (dynamic link library) called “mb.dll”. This encapsulates the functions used by BeeStep14 in communicating with MINI-Bee across the USB interface into a few simple functions easily understood and used in custom software. Although the DLL was written in ‘C’ it can be used (called) by programs written in a number of popular languages including BASIC (visual BASIC etc..).

## 6.1 Writing Visual Basic Programs for MINI-BEE

The DLL provides a general purpose interface that greatly simplifies the task of writing programs for a USB device. It can be tricky manipulating the USB comms into sending and receiving messages to and from a device which can easily be plugged and unplugged at any time. The DLL eliminates all of these headaches by simplifying the task into two library functions.

[BeeInit\(\)](#) and [SetOutputs\(outputs\)](#)

[BeeInit\(\)](#) is called somewhere near the start of your program and takes care of all of the USB comms initialisation and prepares the Mini-Bee for receiving messages.

[SetOutputs\(outputs\)](#) can then be called at any time during your program to set the output pattern of on's and off's. The parameter Outputs is simply a 32 bit integer value where bit0 corresponds to output 1, bit 1 to output2, etc... Where a logic value of 1 turns the output on and a value of 0 turns it off. For example the statement below would turn on the first three outputs...

[SetOutputs\(7\)](#)

The only other thing that a VB program must do is to declare the functions that it is going to use within the DLL and the name of the DLL itself. This must be done at the start of your program or at least before any references to the two functions are made. The following is an program excerpt showing how this is done...

```
Declare Function BeeInit Lib "mb.dll" () As Boolean
```

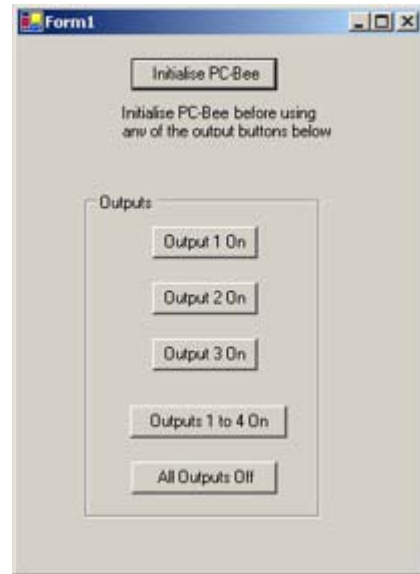
```
Declare Function SetOutputs Lib "mb.dll" (ByVal Outputs As Integer) As Boolean
```

The first declaration states that the function [BeeInit](#) has no parameters, is found in [mb.dll](#) and returns a boolean value. The second states that [SetOutputs](#) has one integer parameter passed by value rather than reference, is found in [mb.dll](#) and also returns a boolean value. It should be noted that the `....Lib "mb.dll"....` part lets the program know where to find the [mb.dll](#) file. When written like this it assumes that since there is no path information that the [bee.dll](#) file can be found in the windows system directory `c:\windows\system32`. If you like you can copy the file [mb.dll](#) on the installation disk to the system32 directory and the above statement will work perfectly. Alternatively you can copy the file to some other location and give that location in the declaration as below...

```
Declare Function BeeInit Lib "c:\library\mb.dll" () As Boolean
```

To speed up your development of software for the Mini-Bee, a complete working example is available in the directory [VBmini](#). It creates a very simple form based program that has individual buttons for various functions such as initialising the Mini-Bee and setting various patterns on the outputs. The main (and only) screen is shown on the right. This has been written using [Microsoft Visual Studio .net](#) and the directory contains the full workspace (solution) details to allow you to immediately open and start editing or running.

Even if you don't have Visual Studio, the source code is virtually self explanatory with the most relevant sections being in "[Form1.vb](#)" which can even be opened in a simple text editor such as notepad.



## 6.2 Writing Visual C++ programs for Mini-Bee

Ignoring some of the formalities in the construction of a Visual C++ program for the windows environment the techniques in using “mb.dll” consists of four main tasks....

### Loading the DLL into memory

Before any functions within the DLL can be used it is necessary to instruct windows to load it into memory. This is done by calling the LoadLibrary() function. i.e.

```
.....  
HINSTANCE BeeHandle; // declaration of variable to hold the handle to the dll  
....  
BeeHandle = LoadLibrary(“mb.dll”); // load the dll into memory and return handle
```

The declaration of the variable BeeHandle used to store the handle to a DLL , uses a built in type definition which is called HINSTANCE in this particular ‘C’ compiler, but you should use the appropriate one defined in your own compiler for this purpose.

The LoadLibrary() function return a handle to the DLL if the load is successful otherwise NULL. Ideally your own program should check for a NULL returned and give an error message. Make sure the function parameter is the full pathlist to where you copied the bee.dll file from the installation CD.

### Get the addresses of the functions within the DLL

Using the DLL handle returned above you can now obtain pointers to the functions within the DLL. Using the following

```
TypeInitMbee      InitMbee;  
TypeSetOutputs    SetOutputs;  
  
.....  
InitMbee = (Type InitMbee)GetProcAddress( BeeHandle, " InitMbee ");  
SetOutputs=(TypeSetOutputs)GetProcAddress(BeeHandle, "SetOutputs");  
.....
```

The TypeInitMbee and TypeSetOutputs type definitions are contained in the header file “mbdll.h” and defines the correct type of function pointer to reference the DLL function. mbdll.h should be included in your own source file eg.

```
.....  
#include “mbdll.h”  
.....
```

The call to GetProcAddress() returns a pointer to this function if found within the DLL otherwise NULL. Once the functions pointers have been obtained in this way the internal functions within the DLL are simply accessed like ordinary function calls e.g.

```
.....  
InitMbee ();  
SetOutputs(0x1234);  
.....
```

### Initialising The DLL

Once the addresses of the DLL functions are obtained as above the remaining functions required to use them are very simple. The first step is to initialise the DLL using....

```
int status;  
.....  
status = InitMbee();
```

Your program should check to see if a value of zero has been returned by `InitMbee()`. Any other value indicates an error. E.g. MINI-Bee not connected etc...

### Using the SetOutputs() Function

The majority of your programming will use the `SetOutputs()` function. This simply applies the pattern 1's and 0's of the first fourteen bits of the 32 bit parameter directly to the outputs. For example: to create a pattern of alternate on and off over all outputs use..

```
SetOutputs(0x00001555);           // hexadecimal number
```

Or to turn on just output 1 only use ...

```
SetOutputs(0x00000001);          // hexadecimal number
```

Etc.....

More generally.....

```
unsigned long bits;  
.....  
bits = 0x00001234;               // hexadecimal number  
SetOutputs(bits);  
.....
```

This example will turn on outputs 13, 10, 6, 5 and 3.

Although this only gives a glimpse of the possibilities of writing your own programs, it should be apparent that the use of the DLL functions greatly simplifies this process. It frees the programmer from the task of getting to know the fine details of programming USB interface communications and lets him concentrate on the main function of controlling switching outputs.

## 7. Minimum PC System Requirements

Mini-Bee and BeeStep14 software do not require a high spec PC for correct operation, but the following system is suggested as a sensible minimum

Processor	500MHz Pentium
Memory	64MB
HDD	10MB free space required
Screen Resolution	1024x768 (256 colours)
Interface	One free USB socket (1.0 or 2.0)
Operating System	Windows 2000, XP or Vista

**WARNING:** The Mini-Bee adaptor board is intended for DC voltages less than 50v. It should not be connected directly to the mains under any circumstances.