Stepper Bee

Stepper motor control from a PC's USB port

Installation and Users Manual
(Including AutoStep Software)

Available exclusively from
PC Control Ltd.
www.pc-control.co.uk

Contents

1. Introduction

Stepper motors are one of the most useful devices in the world of control, automation and robotics. They form the most convenient and versatile bridge between a set of motion rules in a controller (computer) and the motion itself. They can be made to move slowly, quickly, in reverse, pause, complete revolutions, partial revolutions and even individual steps of less than a degree of rotation. With this flexibility of movement coupled with an abundance of torque for relatively little power applied, the stepper motor finds many suitable applications.

The downside to stepper motors is that they are not as simple to “drive” (electrically speaking) as simple DC motors, which just need the required volts and amps to do their bit. Stepper motors need a precise sequence of pulses delivered to the correct winding at the correct time in order to perform their required task. It would be nice if all you had to do was specify the number of steps to take , in what direction and at what speed and the stepper motor obliged. This is the function of the StepperBee USB adaptor board.

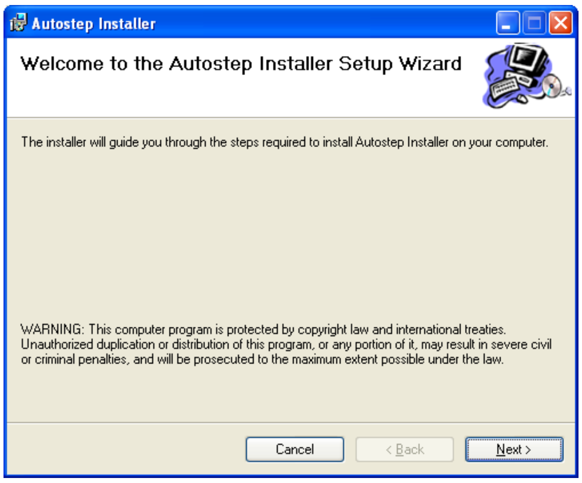
From the PC these requirements can be specified and “sent” to the StepperBee via USB, which then generates the precise sequence of pulses on the appropriate winding to move the stepper motor accordingly. The StepperBee can accommodate two stepper motors operating completely independently using this technique. For example one stepper motor may be sent off to do 1000 steps clockwise at a rate of 1 step every 100ms followed by 100 steps anti-clockwise at 1 step per minute, whilst at the same time motor 2 can be executing it’s own task which might be to do 100 steps clockwise then anticlockwise in a continuous repeating way at a rate of 1 step/sec. The StepperBee sorts out the two motors pulse requirements and makes sure both motors are supplied with the correct “drive” to perform the specified tasks.

The ability to control two motors with independent tasks in this way opens up many possible application areas for the StepperBee which are not easily available with other stepper motor drives. Popular applications that take advantage of this are with x-y co-ordinate drive systems where one motor controls the X-position and one controls the Y. Specifying the appropriate sequence of tasks can produce some very elaborate positional oriented control systems.

Since the StepperBee also has some digital inputs and outputs available for the users program to make use of, it is then possible for these to be used to control a third or 4th axis (albeit in a more simple way) for applications that could, for example, operate a solenoid on the X-Y positioned device. In this way “pick and place” type applications spring to mind. Similarly the StepperBee has a few digital inputs which are typically used to provide some form of feedback from the controlled device. For example a limit switch may be set at some known point and every so often the device could be made to move there until it "makes" the switch establishing to the computer exactly it's position (i.e. datum point).

2. Getting Started

Normally it is necessary to install windows device drivers for new USB devices, which would require some explanation as to the installation procedure. However, in the case of the StepperBee this is unnecessary. Windows XP and Windows2000 already have the drivers installed as part of the operating system (HID device) and as soon as you attach the StepperBee to a USB port, Windows will automatically recognise it and configure it's drivers accordingly.



Welcome to the Autostep Installer Setup Wizard

The installer will guide you through the steps required to install Autostep Installer on your computer.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

StepperBee comes with it's own control program "AutoStep" which allows even the complete beginner to quickly get up and running with stepper motor control. This does need to be installed. Installation is straightforward and simply needs the installation disk inserting into you CD drive. When inserted the installation software will start automatically and provide you with prompts to guide you through the installation process. If this does not happen when you insert the CD then you can manually start it by going to the root directory of the CD (using windows explorer) and double clicking on the "setup.exe" program.

3. Making Connections to StepperBee

The StepperBee has two rows of screw terminals, which are used for making all connections to the external motors and external switching outputs. These are labelled as TL1 and TL2 on the board. The individual terminal connections are numbered from 1 to 9 in sequence with number 1 being labelled on the board for each set of terminals.

StepperBee also has 5 digital inputs available on the PL2 connector. This is a 10 way header type connector which accepts standard 10 way IDC type connectors usually fitted to ribbon cable.

StepperBee is specifically designed for the most common type of stepper motor which is the 4 phase unipolar type. These motors come in a wide variety of specifications differing in power, step resolution, torque, voltage and current requirements. A typical medium torque stepper motor requires 12v at 200mA per phase winding to function correctly. StepperBee can operate up to 500mA per winding at up to 24v, which gives a large amount of flexibility in choosing a StepperMotor for your application. StepperBee can "drive" two stepper motors at the same time with motor 1 connected to TL1 and motor2 to TL2 according to the following connection table…

Pinout of Screw Terminals(TL1)

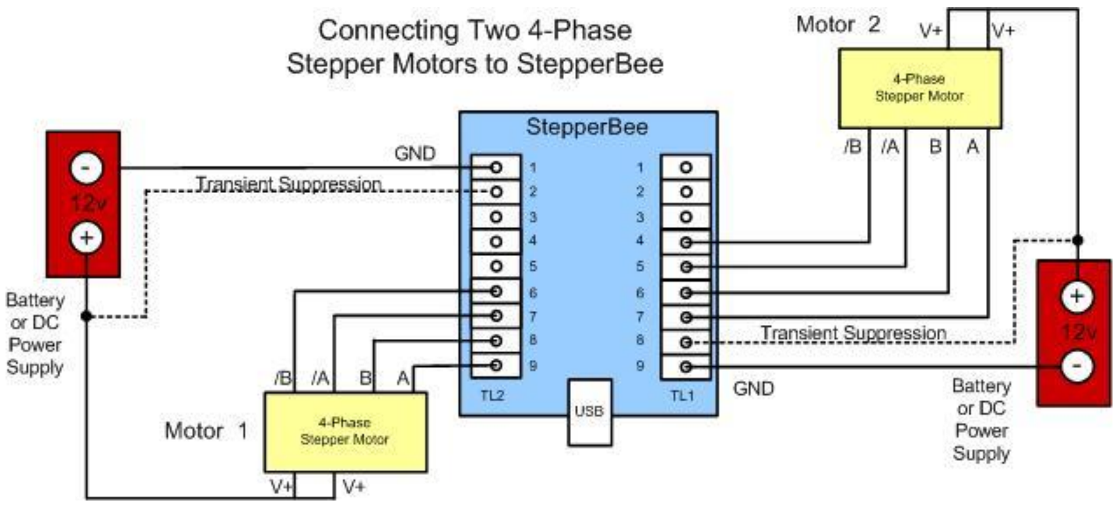
Pin	Signal description
1	Motor 2 Switching Output 3
2	Motor 2 Switching Output 2
3	Motor 2 Switching Output 1
4	Motor 2 Phase /B
5	Motor 2 Phase /A
6	Motor 2 Phase B
7	Motor 2 Phase A
8	Transient Suppression
9	GND

Pinout of Screw Terminals(TL2)

Pin	Signal description
1	GND
2	Transient Suppression
3	Motor 1 Switching Output 3
4	Motor 1 Switching Output 2
5	Motor 1 Switching Output 1
6	Motor 1 Phase /B
7	Motor 1 Phase /A
8	Motor 1 Phase B
9	Motor 1 Phase A

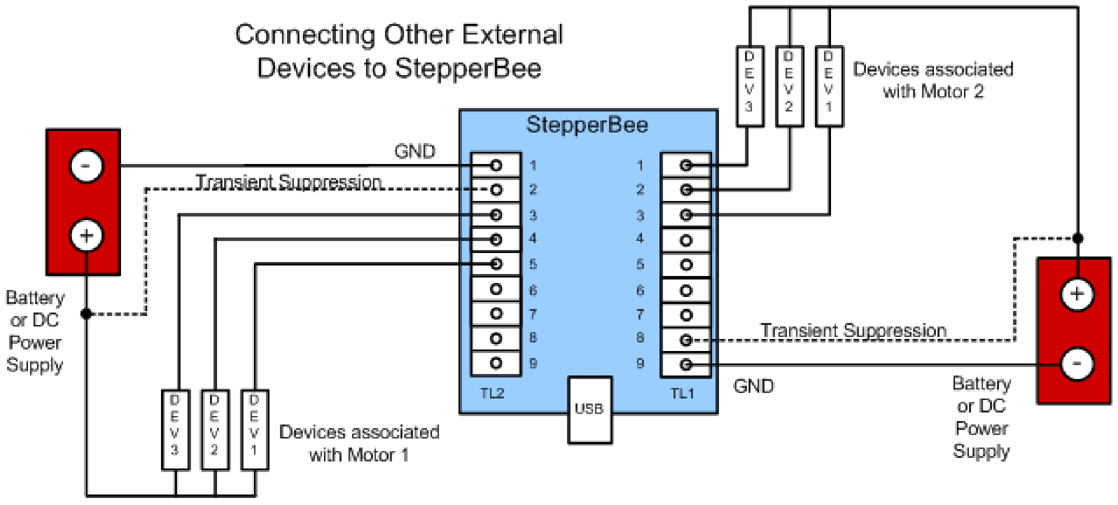
For both sets of terminals, terminal 9 is the one nearest to the terminal number label (i.e. TL1 or TL2) and terminal '1' is marked on the pcb board with a small white number '1'..

To make this clearer the following diagram shows the connection of two 4-phase motors to StepperBee.



Please note that, although the transient suppression connection is shown as a dotted line, this connection is not optional. Although the stepper motors will work correctly without this connection, the stepper bee will be vulnerable to potentially damaging voltage spikes created by the switching on and off of the inductive phase coils of the motor. The transient suppression connection “clamps” these spikes to the positive voltage supply and so protects the board

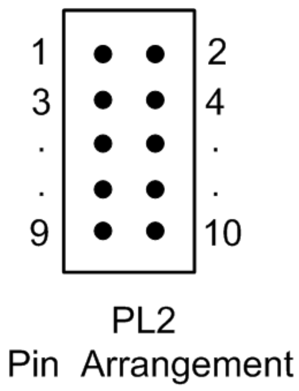
The StepperBee also has 6 high voltage (50v) switching outputs available. Three of these are associated with Stepper Motor 1 and three with Stepper Motor 2. These outputs can be used for a wide variety of purposes including switching on/off lamps, DC motors, solenoids, relays etc… The connection of such devices to these outputs is illustrated below.


Connecting Other External Devices to StepperBee
StepperBee
GND
Transient Suppression
Battery or DC Power Supply
Devices associated with Motor 1
Devices associated with Motor 2
DEV 1
DEV 2
DEV 3
TL2
TL1
USB
Transient Suppression
GND
Battery or DC Power Supply

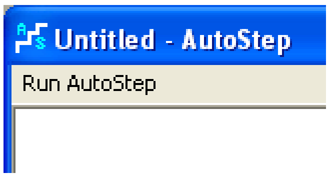
The digital inputs on PL2 provide a convenient way of connecting up to 5 external digital devices such as limit switches or sensors. They only accept standard digital type inputs (i.e. 0v for logic 0 and +5v for logic 1). The pinout of this connector is given below. *Pin1 is nearest to the large chip in the middle of the board.*

Pinout of Digital Input Connector (PL2)

Pin	Signal description
1	Digital Input 1
2	GND
3	Digital Input 2
4	GND
5	Digital Input 3
6	GND
7	Digital Input 4
8	GND
9	Digital Input 5
10	GND

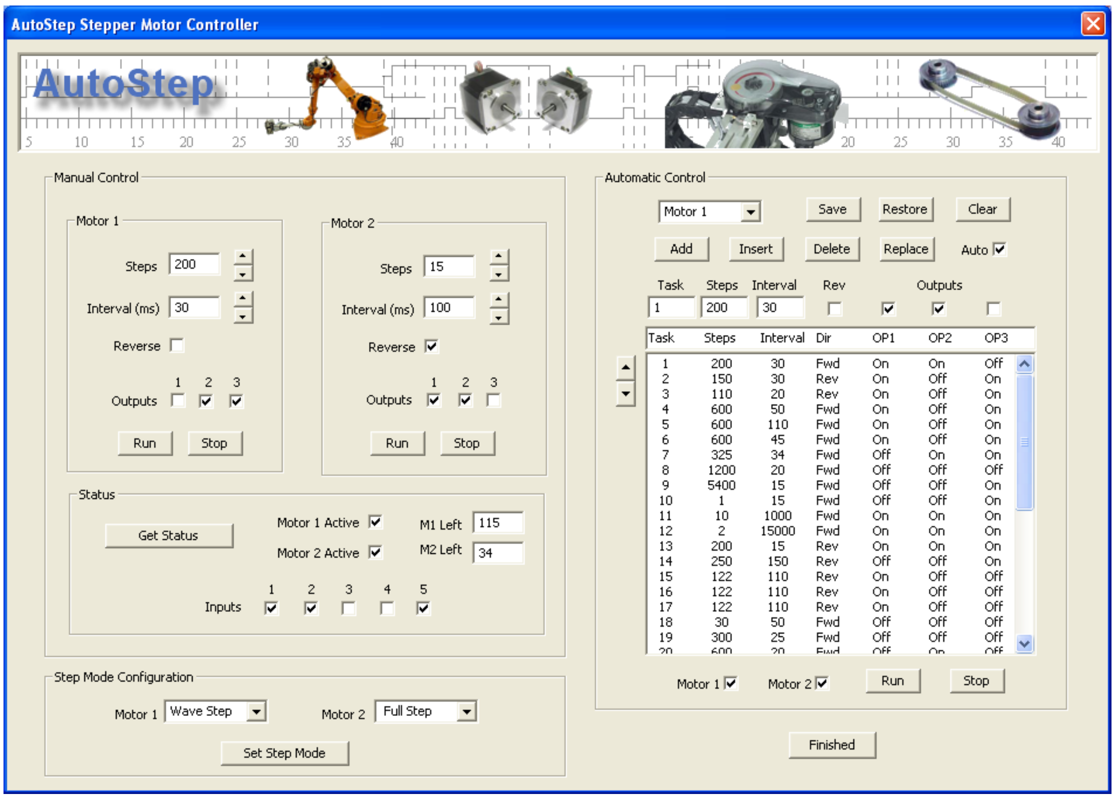


4. AutoStep Software



StepperBee is supplied with AutoStep software which makes it very easy for the beginner to get quickly up and running with stepper motor control. Installation as described above is painless and easy requiring only a Windows 2000 or Windows XP computer with fairly modest specifications. To run the software double click on the desktop icon provided during installation. The AutoStep environment screen will then appear providing the workspace for StepperBee operation. To initialise the StepperBee and start the main controls dialog click on “Run” in the top menu.

AutoStep software is divided into two distinct sections: One for manual operation and one for automatic operation.



Manual Control

Motor 1
Steps 200
Interval (ms) 30
Reverse ☐
Outputs 1 ☐ 2 ✓ 3 ✓
Run Stop

Motor 2
Steps 15
Interval (ms) 100
Reverse ✓
Outputs 1 ✓ 2 ✓ 3 ☐
Run Stop

Status
Get Status
Motor 1 Active ✓
Motor 2 Active ✓
M1 Left 115
M2 Left 34
Inputs 1 ✓ 2 ✓ 3 ☐ 4 ☐ 5 ✓

Step Mode Configuration
Motor 1 Wave Step
Motor 2 Full Step
Set Step Mode

Automatic Control
Motor 1
Save Restore Clear
Add Insert Delete Replace Auto ✓

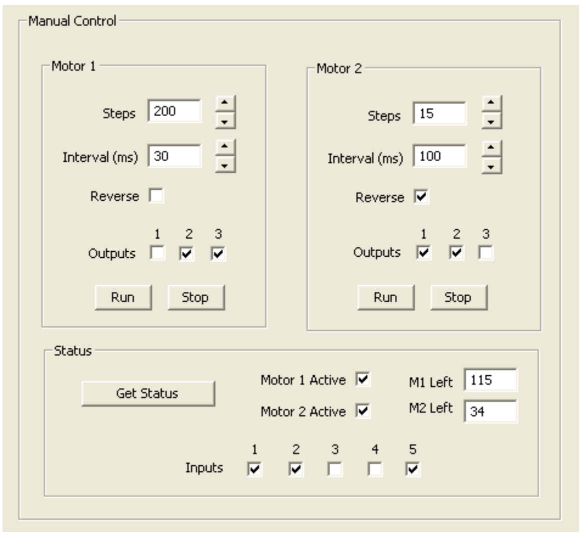
Task	Steps	Interval	Rev	Outputs		
1	200	30	☐	✓	✓	☐

Task	Steps	Interval	Dir	OP1	OP2	OP3
1	200	30	Fwd	On	On	Off
2	150	30	Rev	On	Off	On
3	110	20	Rev	On	Off	On
4	600	50	Fwd	On	Off	On
5	600	110	Fwd	On	Off	On
6	600	45	Fwd	On	Off	On
7	325	34	Fwd	Off	On	On
8	1200	20	Fwd	Off	Off	Off
9	5400	15	Fwd	Off	Off	On
10	1	15	Fwd	Off	Off	On
11	10	1000	Fwd	On	On	On
12	2	15000	Fwd	On	On	On
13	200	15	Rev	On	On	On
14	250	150	Rev	Off	Off	On
15	122	110	Rev	On	Off	Off
16	122	110	Rev	On	Off	Off
17	122	110	Rev	On	Off	Off
18	30	50	Fwd	Off	Off	Off
19	300	25	Fwd	Off	Off	Off
20	600	20	Fwd	Off	On	Off

Motor 1 ✓ Motor 2 ✓ Run Stop

Finished

4.1. Manual Control



Motor 1
Steps 200
Interval (ms) 30
Reverse ☐
Outputs 1 ☐ 2 ✓ 3 ✓

Motor 2
Steps 15
Interval (ms) 100
Reverse ✓
Outputs 1 ✓ 2 ✓ 3 ☐

Status
Motor 1 Active ✓
M1 Left 115
Motor 2 Active ✓
M2 Left 34
Inputs 1 ✓ 2 ✓ 3 ☐ 4 ☐ 5 ✓

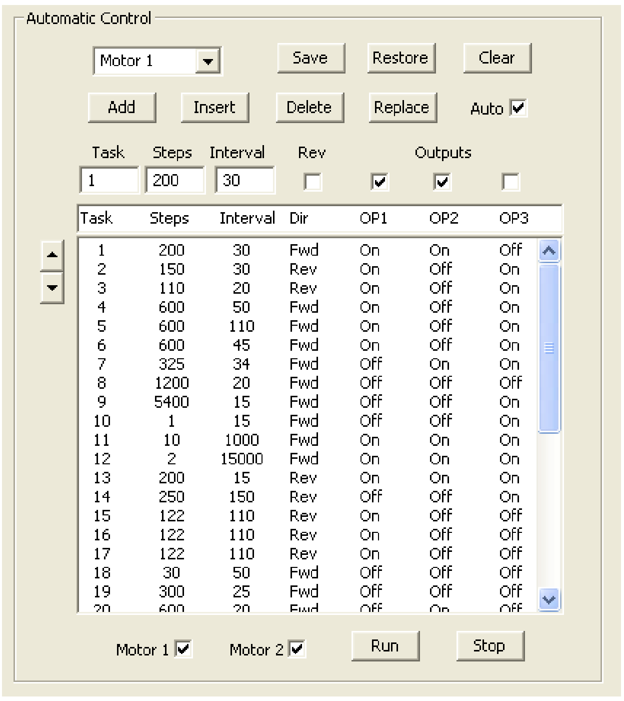
Using the manual controls, each motor can be given an individual "task" to perform, which will be executed immediately. The task is pre-specified in terms of number of steps, time interval between steps and direction (forward or reverse) and then "sent" to the StepperBee to "Run". Lets look at the manual controls section of the main screen…

There is a separate section for each motor. Looking at Motor 1 on the left the number of steps required can be entered into the "Steps" box. This number must be in the range 1 – 16000. The time interval that each step will take should be entered into the "Interval" box. This should also be a number in the range 1 – 16000 and corresponds to the time in milliseconds (ms) for each step. Note that these are approximate figures for guidance only. Absolute accuracy of step interval timings is not guaranteed.

The direction is implied to be forward unless the "Reverse" tick box is ticked. In addition to specifying the step movement of the motor as just described, it is also possible to specify what the digital outputs should do during that step by ticking the appropriate boxes 1-3. When the box is ticked the corresponding output will be on (i.e. logic '1' or "+5v"). This output will hold this state for the duration of the specified step. Once the specifications of the task have been entered it can be executed by the StepperBee by clicking on the "Run" button. The task will then run to completion as specified and then stop automatically. If you need to abort a task before it has been completed then simply press the stop button. This will send an abort command to the StepperBee and terminate the task immediately. Note that the outputs will hold the state they were in at the time the stop command was issued.

Within the status section of manual controls, the current status of both motors and the digital inputs can be examined by clicking on the "Get Status" button. This will send a request to StepperBee which will return the current status. If, for example, motor 1 is still active the "Motor 1 Active" tick box will be ticked and the number of steps remaining to be completed will be shown in the box labelled "M1 Left". Similarly for motor 2 using "Motor 2 Active" and "M2 Left".

4.2. Automatic Control



Task	Steps	Interval	Rev	Outputs		
1	200	30	☐	✓	✓	☐

Task	Steps	Interval	Dir	OP1	OP2	OP3
1	200	30	Fwd	On	On	Off
2	150	30	Rev	On	Off	On
3	110	20	Rev	On	Off	On
4	600	50	Fwd	On	Off	On
5	600	110	Fwd	On	Off	On
6	600	45	Fwd	On	Off	On
7	325	34	Fwd	Off	On	On
8	1200	20	Fwd	Off	Off	Off
9	5400	15	Fwd	Off	Off	On
10	1	15	Fwd	Off	Off	On
11	10	1000	Fwd	On	On	On
12	2	15000	Fwd	On	On	On
13	200	15	Rev	On	On	On
14	250	150	Rev	Off	Off	On
15	122	110	Rev	On	Off	Off
16	122	110	Rev	On	Off	Off
17	122	110	Rev	On	Off	Off
18	30	50	Fwd	Off	Off	Off
19	300	25	Fwd	Off	Off	Off
20	600	20	Fwd	Off	On	Off

In manual control the tasks to be performed by the Stepper Motor(s) are executed as soon as you press the “run” button and are limited to just one task at a time per motor. In automatic control you can pre-specify a sequence of tasks to be performed by one or both motors before starting them.

Using the various editing facilities here you can construct a separate sequence for each motor. The first step is to select the motor. This is done using the drop down menu box which will show either “Motor 1” or “Motor 2”. Once you have made this selection , all further operations in constructing a sequence of tasks will be for that motor. You can change this to the other motor at any time to do the same thing for the other motor without losing what you have specified for the first one. Let’s look at the edit boxes.

The task number box indicates the number of the task to be edited. i.e. each task you plan to send to the StepperBee is sequentially numbered starting from 1. The “Steps” and “Interval” boxes show the number of steps to be taken and the interval between them in milliseconds respectively. The “Rev” tickbox indicates the direction of movement during this task (reverse when ticked) and the 3 output boxes show the state of the switching outputs during the task. Once you have the first task for the selected motor entered into these boxes simply press the “Add” button to add this task to the list of tasks to be performed. The list of tasks currently specified is shown in the display area immediately below the task edit boxes. This list can hold up to 10,000 tasks. When the display area becomes full, a scroll bar will appear on the right hand side to allow scrolling through the entire list. When scrolling through the list (using the up and down arrow buttons) the task at the top of the list will be copied into the edit boxes. This allows any entry in the task list to be changed. To change such an entry , bring it to the top of the list using the up/down arrows and then enter the new data into the edit boxes. When complete press the “Replace” button to replace the task in the list with the one just edited.

The “Insert” and “Delete” buttons give added flexibility for editing the task list. The “Insert” button will insert the task specified in the edit boxes above the task currently at the top of the displayed list. The “Delete” button will delete the task currently at the top of the list. Pressing the “Clear” button will empty the entire task list. You will be prompted “Are you sure ?” when pressing this button to avoid accidental erasure of your task list.

When the “Auto” tick box is ticked any editing of the boxes will be reflected immediately in changes to the task at the top of the list. i.e. without pressing the “Replace” button.

Once you have a fully specified task list for a particular motor you can save this to hard disk by pressing the “Save” button and supplying a name for the file.

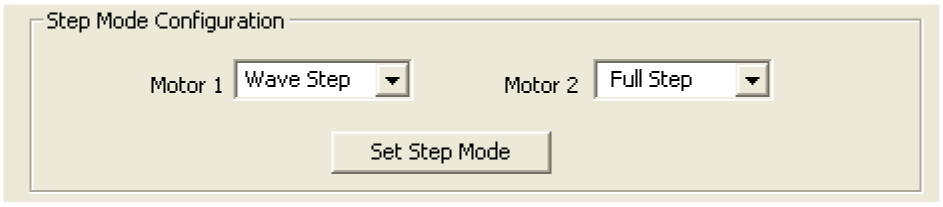
Correspondingly you can retrieve a task list by pressing the “Restore” button and specifying the previously saved file to restore. Note that the save and restore facilities operate on the task lists for the individual motor indicated. To save both task lists you need to save the task list for motor 1 and motor 2 separately. This also means that you can save a task list specified for motor 1 and restore it to motor 2 if desired. This can save a large amount of time when producing two very similar task lists.

Once you have a specified your complete task list for one or both motors you can start actual motor operation by pressing the “Run” button. There are two tick boxes to the left of the “Run” button labelled as “Motor 1” and “Motor 2”. When ticked, the corresponding motor will be run according to it’s specified task list. i.e. having specified two completely independent task lists (one for each motor) both task list can be executed concurrently by ticking both of these boxes. It should be noted that the task list does not need to be displayed in the edit windows etc.. for it to be used for running the motors. In fact only one motor at a time can have it’s task list displayed in this way, but both motors can run their respective task list which is stored in internal memory. It should also be noted that the separate tasks specified for each motor do not need to be synchronised in any way. i.e. one task within a task list does not need to finish before a task in the other task list for the other motor can be started or stopped. This independence and concurrency gives the StepperBee a very powerful and flexible approach to multiple stepper motor control, separating it from other more restrictive controllers.

Pressing the “Stop” button will abort the tasks being executed by both motors at the end of their current task. To stop a motor immediately click on the “Stop” button of the automatic control followed by the corresponding “Stop” button in the manual control area. The “Stop” button in the manual control area can be used at any time , even during automatic operation, to stop the current task. Note, that if there is a task list being run in automatic mode then pressing the manual stop button on it’s own merely stops the current task in that list. i.e. the automatic sequence control will then move onto the next task in the list. In this way the manual stop may be used as a kind of “skip” facility to skip over the current task being run..

5. Stepper Step Mode

When using stepper motors there are a number of options for the pattern of pulses required to move the motor in the specified direction at the specified speed for the given number of steps. The most common two are “Full Step” and “Wave Step”. Without going into the precise details of the timing differences of these options we can summarise by saying that Wave Step uses less power for energising the motor windings but delivers less torque to the load, whereas Full Step uses more power but provides the motor with significantly more torque.


Step Mode Configuration

Motor 1: Wave Step

Motor 2: Full Step

AutoStep provides you with the choice of step method using the selection boxes on the bottom of the screen. The step method can be different for each motor. To set the step mode, select the option from the drop down box for each motor and then press the “Set Step Mode” button. The step mode will then be set for any subsequent operations for both manual and automatic control.

In addition to Step Patterns of pulses the step mode configuration allows the motor to be completely turned off by choosing the “Power Off” mode from the drop down list. In this mode all phase outputs are turned off. This can be useful when the motor spends a lot of time stationery. When stationery the motor windings will consume more power and can make the motor overheat. The downside to turning off the power is that the motor will no longer be supplying torque to “hold its position”. In many applications, especially where a gearbox is used, this is not a problem.

6. Writing your own software for StepperBee

To use StepperBee straight from the box does not require any programming other than entering your task list details into AutoStep. However, if you prefer to design you own software then the following information will be of use.

Provided with Stepper-Bee is a DLL (dynamic link library) called “stp.dll”. This encapsulates the functions used by AutoStep in communicating with StepperBee across the USB interface into three 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, the most popular of which is BASIC (Visual BASIC). Described below are the techniques to use the DLL in Visual Basic and C++. If you program in another language please refer to your compiler manual on the details of calling a C++ library function, which will be very similar to the techniques described below.

6.1 Using StepperBee with Visual Basic.

Using Stepper-Bee with your own programs written in visual basic is very simple. At the head of your program, before using any of the Stepper-Bee functions, you must make the following declarations…

```
Declare Function InitStp Lib "stp.dll" () As Integer

Declare Function RunMotor1 Lib "stp.dll" (      ByVal steps As Integer,
                                                ByVal interval As Integer,
                                                ByVal direction As Integer,
                                                ByVal outputs As Integer,
                                           ) As Boolean

Declare Function RunMotor2 Lib "stp.dll" (      ByVal steps As Integer,
                                                ByVal interval As Integer,
                                                ByVal direction As Integer,
                                                ByVal outputs As Integer,
                                           ) As Boolean

Declare Function StopMotor1 Lib "stp.dll" ( ByVal outputs As Integer,  ) As Boolean

Declare Function StopMotor2 Lib "stp.dll" ( ByVal outputs As Integer,  ) As Boolean

Declare Function SetStepMode Lib "stp.dll" (    ByVal M1Mode As Integer,
                                                ByVal M2Mode As Integer,
                                             ) As Boolean

Declare Function GetStatus Lib "stp.dll" (      ByRef M1Active As Integer,
                                                ByRef M2Active As Integer,
                                                ByRef M1Steps As Integer,
                                                ByRef M2Steps As Integer,
                                                ByRef Inputs As Integer,
                                          ) As Boolean
```

Normally each of the parameters of the functions would be on the same line in your program. They are shown here on separate lines for clarity. These provide sufficient information for your compiler to determine the correct way to use the functions contained within the DLL. You should also ensure that you copy the stp.dll file from the installation disk to your c:\windows\system32 directory so that your compiler can find it.

Using the InitStp() function

Before using any of the motor control functions you must first initialise the StepperBee using the initialise function as follows…

```
InitStp()
```

There are no parameters required for this function call. Once initialised you can then use the functions, listed below, anywhere in your program.

Using the RunMotor1() Function

The following applies to both RunMotor1() and RunMotor2() functions.
The RunMotor1() function has 4 parameters

All four parameters are type integer and would be declared somewhere in your program as follows;

```
Dim steps, interval, direction, outputs  As Integer
```

The corresponding function call would be

```
RunMotor1(steps, interval, direction, outputs)
```

steps - integer in the range 1 to 16000 corresponding to the number of steps to execute

interval – integer in the range1 to 16000 corresponding to the time interval in milli-seconds between each step.

direction – integer in the range 0 to 1. Zero corresponds to forward and 1 to reverse.

outputs – integer in the range 0 to 7 corresponding to the bit pattern for on/off of the additional switching outputs associated with motor 1. e.g. a value of 5 (which is 00000101 in binary) would result in outputs 1 and 3 being on.

Example

To run Motor1 forward for 200 steps with 50ms between steps and all additional switching outputs off would use the following function call…

```
RunMotor1(200, 50, 0, 0)
```

Using the StopMotor1() Function

The following applies to both StopMotor1() and StopMotor2() functions.
The StopMotor1() function has one integer parameter and simply terminates the current task being performed by motor1 immediately while updating the current state of the other switching outputs.

The parameter is of type integer and would be declared somewhere in your program as follows:

```
Dim outputs  As Integer
```

The corresponding function call would be

```
StopMotor1(outputs)
```

outputs – integer in the range 0 to 7 corresponding to the bit pattern of on/ff of the additional switching outputs associated with motor1

Example

To stop motor 1 and set the switching outputs 1 and 2 to ON, use the following function call

```
StopMotor1(3)
```

TIP: This is also a convenient way of manipulating the switching outputs without any need to run the motor

Using the SetStepMode() function

The StepMode can be set to one of three possible options… Full Step, Wave Step and Power Off. The SetStepMode() function has therefore three possible values for its parameters M1Mode and M2Mode. A '0' sets Wavestep mode, a '1' sets Full step mode and a 2 sets Power Off mode. These mode settings are independent for motors 1 and 2 and can be changed at any time, even during a step interval. The parameters would be declared in your program somewhere as

```
Dim M1Mode, M2Mode As Integer
```

The corresponding function call would be

```
SetStepMode(M1Mode, M2Mode)
```

By way of example.. setting Motor1 to wavestep mode and Motor2 to FullStep mode would require….

```
.
.
M1Mode = 0
M2Mode = 1
SetStepMode(M1Mode, M2Mode)
.
.
```

Using the GetCurrentStatus() function

This function may be called at any time to determine the running status of both motors and the status of the digital inputs. This function has 5 integer parameters passed by reference. The parameters would be declared in your program somewhere as..

```
Dim M1Active, M2Active, M1Steps, M2Steps, Inputs  As Integer
```

The corresponding function call would be

GetCurrentStatus(M1Active, M2Active, M1Steps, M2Steps, Inputs)

After calling this function the values returned in the parameters correspond to the following…

M1Active – set to 1 for motor1 active, 0 for motor1 stopped.

M2Active – set to 1 for motor2 active, 0 for motor2 stopped.

M1Steps – integer in the range 0 – 16000 corresponding to the number of steps motor1 has left to complete.

M2Steps – integer in the range 0 – 16000 corresponding to the number of steps motor2 has left to complete.

Inputs – integer in the range 0 – 31 corresponding to the bit pattern of the current digital inputs. (bit 0 corresponds to input 1, bit 1 to input 2, etc…)

Example

If motor 1 is running with 123 steps left to complete, motor 2 is stopped and only digital input 4 is on then the following call…

GetCurrentStatus(M1Active, M2Active, M1Steps, M2Steps, Inputs)

… will results in the following values returned …

M1Active = 1,   M2Active = 0,   M1Steps = 123,   M2Steps = 0,   Inputs = 8

As a guide, the source code to a fully working Visual Basic program is contained on the installation CD. This is in the VBStepper directory. It was written using Microsoft Visual Studio and contains the complete workspace for this environment. This allows you to immediately get started editing and running a working visual basic program for the StepperBee. If you don’t have “visual studio” you can still get most of the required information by cutting and pasting code from the main program in the file Form1.vb which can even be opened in a simple text editor such as notepad.

6.2 Using StepperBee with Visual C++

Ignoring some of the formalities in the construction of a Visual C++ program for the windows environment the techniques in using “dgb.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 HStpDll;               // declaration of variable to hold the handle to the dll
….
HStpDll = LoadLibrary(“stp.dll”);   // load the dll into memory and return handle
```

The declaration of the variable StpHandle 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 returns 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 stp.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

```
Type_InitStp              InitStp;
Type_RunMotor1            RunMotor1;
Type_StopMotor1           StopMotor1;
Type_RunMotor2            RunMotor2;
Type_StopMotor2           StopMotor2;
Type_SetStepMode          SetStepMode;
Type_GetCurrentStatus     GetCurrentStatus;
```

```
……..
InitStp = (Type_InitStp)GetProcAddress( HStpDll, "InitStp");
RunMotor1 = (Type_RunMotor1)GetProcAddress( HStpDll, "RunMotor1");
RunMotor2 = (Type_RunMotor2)GetProcAddress( HStpDll, "RunMotor2");
StopMotor1 = (Type_StopMotor1)GetProcAddress( HStpDll, "StopMotor1");
StopMotor2 = (Type_StopMotor2)GetProcAddress( HStpDll, "StopMotor2");
SetStepMode = (Type_SetStepMode)GetProcAddress( HStpDll, "SetStepMode");
GetCurrentStatus = (Type_GetCurrentStatus)GetProcAddress(HStpDll,GetCurrentStatus");
……….
```

The type definitions shown above are contained in the header file “st.h” and defines the correct type of function pointer to reference the DLL function. This file is included in the DLL directory on the installation disk.
st.h should be included in your own source file eg.

```
………..
#include “st.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.

```
………..
InitStp ();
RunMotor1(200, 50, 0, 0);
RunMotor2(1000, 20, 1, 0);
…………
```

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 = InitStp();
```

Your program should check to see if an error value of zero has been returned by InitStp (). A value of ‘1’ indicates success. The number returned is actually the number of stepper bee boards currently connected to the computer. However, if it is more than 1 you would need to use our free “Beehive” program and DLL for multi-board control (see www.pc-control.co.uk/beehive.htm for more details). The Beehive DLL is almost identical to the normal one but allows for board numbering.

Using the RunMotor1() Function

The following applies to both RunMotor1() and RunMotor2() functions.
The RunMotor1() function has 4 parameters

```
RunMotor1(int steps, int interval, int direction, int outputs);
```

All four parameters are type integer and would be declared somewhere in your program as follows;

```
int steps, interval, direction, outputs;
```

Steps - integer in the range 1 to 16000 corresponding to the number of steps to execute

Interval – integer in the range1 to 16000 corresponding to the time interval in milli-seconds between each step.

Direction – integer in the range 0 to 1. Zero corresponds to forward and 1 to reverse.

Outputs – integer in the range 0 to 7 corresponding to the bit pattern for on/off of the additional switching outputs associated with motor 1. e.g. a value of 5 (which is 00000101 in binary) would result in outputs 1 and 3 being on.

Example

To run Motor1 forward for 200 steps with 50ms between steps and all additional switching outputs off would use the following function call…

```
RunMotor1(200, 50, 0, 0);
```

Using the StopMotor1() Function

The following applies to both StopMotor1() and StopMotor2() functions.
The StopMotor1() function has one integer parameter..

```
StopMotor1(int outputs)
```

This simply terminates the current task being performed by motor1 immediately while updating the current state of the other switching outputs.

The parameter is of type integer and would be declared somewhere in your program as follows:

```
int outputs;
```

Outputs – integer in the range 0 to 7 corresponding to the bit pattern of on/ff of the additional switching outputs associated with motor1

Example

To stop motor 1 and set the switching outputs 1 and 2 to ON, use the following function call

```
StopMotor1(3);
```

TIP: This is also a convenient way of manipulating the switching outputs without any need to run the motor

Using the SetStepMode() function

The StepMode can be set to one of three possible options… Wave Step, Full Step and Power Off. The SetStepMode() function has therefore three possible values for its parameters M1Mode and M2Mode. A '0' sets Wavestep mode, a '1' sets Full step mode and a '2' sets power-off mode. These mode settings are independent for motors 1 and 2 and can be changed at any time, even during a step interval. It should be noted that in power off mode all phases are off. This would mean that there is no applied torque to hold the current motor position. The parameters would be declared in your program somewhere as

```
int M1Mode, M2Mode;
```

The corresponding function call would be

```
SetStepMode(M1Mode, M2Mode);
```

By way of example.. setting Motor1 to wavestep mode and Motor2 to FullStep mode would require….

```
.
.
M1Mode = 0;
M2Mode = 1;
SetStepMode(M1Mode, M2Mode);
.
.
```

Using the GetCurrentStatus() function

This function may be called at any time to determine the running status of both motors and the status of the digital inputs. This function has 5 parameters passed as pointers to integers.

```
GetCurrentStatus(int *M1Active, int *M2Active, int *M1Steps, int *M2Steps, int *Inputs);
```

The parameters would be declared in your program somewhere as..

```
int M1Active, M2Active, M1Steps, M2Steps, Inputs;
```

and passed to the function as their address. e.g…..

```
GetCurrentStatus(&M1Active, &M2Active, &M1Steps, &M2Steps, &Inputs);
```

After calling this function the values returned in the parameters correspond to the following…

M1Active – set to 1 for motor1 active, 0 for motor1 stopped.

M2Active – set to 1 for motor2 active, 0 for motor2 stopped.

M1Steps – integer in the range 0 – 16000 corresponding to the number of steps motor1 has left to complete.

M2Steps – integer in the range 0 – 16000 corresponding to the number of steps motor2 has left to complete.

Inputs – integer in the range 0 – 31 corresponding to the bit pattern of the current digital inputs. (bit 0 corresponds to input 1, bit 1 to input 2, etc…)

Example

If moor 1 is running with 123 steps left to complete, motor 2 is stopped and only digital input 4 is on then the following call…

```
GetCurrentStatus(&M1Active, &M2Active, &M1Steps, &M2Steps, &Inputs);
```

… will results in the following values returned …

```
M1Active = 1,   M2Active = 0,   M1Steps = 123,   M2Steps = 0,   Inputs = 8
```

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 reading inputs and setting outputs.

7 Minimum PC System Requirements

StepperBee and AutoStep 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 XP (or later)

8. Troubleshooting

If , after connecting a suitable stepper motor to the stepper bee and running Autostep still leaves you with a non functioning system then use the following sequence of instructions, **following them exactly**, to help you get round most of the common mistakes.

a. Connect your motor to Motor 1 connections (TL2) and make sure your wiring is completely correct.

b. Run Autostep (All of the following instructions apply to the "Manual Control" area (Do not use the automatic controls yet)

c. Click "Get Status" and see if any motor is active

d. If any motor is active click the "Stop" button for that motor

e. Click "Get Status" until the all motors show as not active

f. Set motor 1 to Steps = 200, Interval=20ms, Reverse (off)

g. Click on "Run" for motor 1

h. Motor 1 should now do 200 steps (which should take about 4 seconds) and then stop. (the first time you do this be prepared to wait up to 2 minutes before any movement is observed. This can be due to a noisy boot which leaves a random long delay in the very first step)

WARNING: The StepperBee adaptor board is intended for low voltage operation (less than 30 volts). It should not be connected directly to mains voltages under any circumstances.

StepperBee
Regulatory Compliance and Safety Information

Product Name: StepperBee
Part No. BRD005

IMPORTANT PLEASE RETAIN THIS INFORMATION FOR FUTURE REFERENCE

Warnings

- This product should not be connected to mains voltages under any circumstances.
- This product should be placed on a stable, flat, non-conductive surface in use and should not be contacted by conductive items.
- The connection of non CE compliant devices may affect overall compliance or result in damage to the unit and invalidate the warranty.

Instructions for safe use

- To avoid malfunction or damage to your board please observe the following:
- Do not expose it to water, moisture or place on a conductive surface whilst in operation.
- Do not expose it to heat from any source; the MotorBee is designed for reliable operation at normal ambient room temperatures.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Avoid handling the board while it is powered. Only handle by the edges to minimize the risk of electrostatic discharge damage.
- All peripherals used with the board should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met.

Compliance Information

- The board complies with the relevant provisions of the RoHS Directive for the European Union.

WEEE Directive Statement for the European Union

- In common with all Electronic and Electrical products the board should not be disposed of in household waste. Alternative arrangements may apply in other jurisdictions.

EMC Compliance Statements

European Union (EU) Electromagnetic Compatibility Directive Compliance Statement

This product is in conformity with the protection requirements of EU Council Directive 2004/108/EC on the approximation of the laws of the Member States relating to electromagnetic compatibility.

Warning: This is equivalent to an EN 55022 Class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

PC Control Ltd. 18 Beech Close, Desborough, Kettering, Northants NN14 2XQ, UK
www.pc-control.co.uk

Terms of Use for all Goods Supplied

Definitions

‘Supplier’ shall mean PC Control Ltd.
‘Buyer’ shall mean the person, company or any other body that purchases or agrees to purchase Goods.
‘Goods’ shall mean all goods and services which the Buyer agrees to buy from the Supplier including replacements for defective Goods, hardware, documentation and software products licensed for use by the Buyer.
Use of the Goods in any way by the Buyer constitutes acceptance of these terms and conditions.

Terms and Conditions

1. The Goods are intended to be part of the buyer’s own design of apparatus and not a finished product in their own right.
2. The Goods supplied are not to be used in any design where there is a risk, however small, either directly or indirectly, of death or personal injury.
3. The Buyer will be responsible for ensuring the fitness for purpose of the Goods for the Buyer’s application.
4. To the extent permitted by law, the Supplier accepts no liability whatsoever or howsoever arising in respect of loss, damage or expense arising from errors in information or advice provided whether or not due to the Supplier’s negligence or that of its employees, agents or sub-contractors save for any loss or damage arising from death or personal injury.
5. To the extent permitted by law, the Supplier shall not be liable to the Buyer by reason of any representation (unless fraudulent), or any implied warranty, condition or other term, or any duty at common law, or under the express terms of any Contract with the Buyer, for any indirect, special or unforeseen loss or damage (whether for loss of profit or otherwise), costs, expenses or other claims for compensation whatsoever (whether caused by the negligence of the Supplier, its employees or agents or otherwise) which arise out of or in connection with the supply of the Goods or their use or resale by the Buyer.
6. The entire liability of the Supplier under or in connection with the Contract with the Buyer shall not exceed the price of the Goods except as expressly provided in these terms and conditions.
7. These terms are an important part of the full terms and conditions of business as published on the website at www.pc-control.co.uk/general-terms.htm which also apply.

If you cannot agree to the terms and conditions of use of the StepperBee then you should return the StepperBee to the supplier within 7 days of receipt to receive a refund. Your use of the board or the associated software in any way whatsoever will be regarded as an acceptance of these terms and conditions.