Peter Norberg Consulting, Inc.

Professional Solutions to Professional Problems

P.O. Box 10987 Ferguson, MO 63135-0987 (314) 521-8808

First Use Notes for SimStep and BiStep Controllers

Version 1.02

By

Peter Norberg Consulting, Inc.

Table of Contents

Overview

This document is intended to help first-time users of Peter Norberg Consulting, Inc.'s stepper motor controller products get their boards operating with a minimal amount of difficulty. As such, it provides tricks and hints for initial checkout and testing which can be used to prove that the boards work and motors can be controlled.

This document assumes familiarity with basic electronics, access to standard electronics tools (such as digital volt/ohm meters, jeweler's screw drivers, clip leads, etc.), and access to a computer which can run a terminal emulator. (Most versions of Microsoft Windows™ have this ability.)

The revision history of this manual is as follows:

Version	Date	Description
1.00	July 23, 2003	First reviewed release
1.01	July 28, 2003	Added **Calculating Current Requirements**, some notes on double power use, and more caveats on issues arising from miss-wiring your system
1.02	July 29, 2003	Added more contact information

If you have any problems which this manual does not appear to be able to resolve, please do not hesitate to contact us at our customer support email address or via the telephone.

Our customer support email address is:

(support@stepperboard.com)

Our phone numbers are:

U.S. Toll Free: 877-230-5270

International: 314-521-8808

First Tasks

We suggest that the following tasks be performed the first time any of our boards is used. In each case, ***read the entire section referenced before doing anything***, so that you will have some idea as to what is coming next:

1. Read the "**Product Safety Warnings**" section of the "Universal Stepper" manual relevant to your board.
2. Read the "**Board Connections**" section of the same "Universal Stepper" manual, for information about the various connectors on your board. Note that this manual ("First Use") has a section entitled "**Labeling of Board Signals**", which may also help you understand where various signal lines may be found.
3. Do a "**Simple Serial Test**" of the board, as described in this manual, in order to confirm that the RS232 serial operations of the product are in proper working order. *Do this step even if you intend to use TTL-serial with some other device, such as a Parallax Basic Stamp, as your control system.*
4. Review the "**Calculating Current Requirements**" section of this manual, to reconfirm that the controller and power supplies which you have purchased can really operate your motor. Remember that the "double power mode" of operation is available if the GenStepper firmware is being used, and you only need to operate one motor with the controller.
5. If you are using the GenStepper or NCStepper firmware (these are the normal firmware options), please refer to the "**Wiring Your Motor**" section of this manual. This section will show you how to determine the correct wiring for your motor. As part of that test, you will also prove that the board is correctly operating your motor. You may also wish to review the "**Motor Wiring Examples**" found in the "Universal Stepper" manual.
6. If you are using SerRoute firmware on your board, please refer to the "**Simple SerRoute Test**" section of this manual for some initial information on preliminary tests to show you how to use SerRoute to operate multiple boards via one RS232 interface.
7. If you are going to use TTL-Serial to control your board, and the above tests have succeeded, then disable RS232 serial input at this point and test your TTL-Serial connection. Refer to the "**TTL-Serial Information**" section of this manual for a generic summary of how to disable RS-232 and to wire TTL-Serial. If you are using a Parallax Basic Stamp to operate the board, refer to the "**Getting a Parallax Basic or Javelin Stamp Running**" section in this manual for some tests to perform.

At this point, you should be ready to do your own testing and development. Please remember that if you ever get confused as to how your board is going to respond to a given serial command, you can always go back to using Hyperterminal with RS232-Serial communications to test out your ideas.

Labeling Of Board Signals

There are currently a total of 6 significant versions of the SimStep and BiStep series of boards available. These versions can be roughly grouped into two major releases: ***Release 1***, which has a large (19 pin) SIP header in the middle of the board which contains all of the standard TTL I/O signals, and ***Release 2***, which uses clearly labeled connectors at the edge of the board, containing the same signals.

Release 1 Pinout for J1– SimStepA04, BiStepA04, and BiStepA05

The pinout for the J1 connector on the Release 1 set of boards (the SimStepA04, BiStepA04, and BiStepA05) is as follows, counting from the "top" part of the connector (nearest the DB9 serial connector) and proceeding downward. This connector is the 19 pin SIP header mounted in the middle of the board.

Pin	Board Label	Signal as used in this manual
1	RTC	
2	+5	
3	RST	
4	GND	GND
5	GND	GND
6	A0	LY-
7	A1	LY+
8	A2	LX-
9	A3	LX+
10	B0	Y-
11	B1	Y+
12	B2	X-
13	B3	X+
14	B4	NXT
15	B5/READY	RDY
16	B6/SERIN	SI
17	B7/SEROUT	SO
18	+5	
19	GND	GND

Release 2 – BiStepA06, BiStep2A, and SS0705

The Release 2 boards' serial connectors are fully labeled directly on the artwork. The signal names can be found by looking on the board near each connector. There are 3 input connectors, containing equivalent signals to those from the J1 connector on the earlier release.

- LIM contains RST, LY-, LY+, LX-, and LX+
- SLEW contains Y-, Y+, X-, and X+
- IO contains NXT, RDY, SI, and SO

Simple Serial Test

This section describes a simple RS232-based serial test that you should perform on **any** of our products **before** you connect them to motors or other boards. This test is to verify that serial communications are working in both directions and the board itself is at least superficially operational.

Do a basic power-on test

1. Make certain that absolutely no wires are connected to the board, including those attached to motors, limit switches, other computers, or any other equipment of any form.
2. Make certain that the board is properly configured for RS232 operation.
   - If the board is a SimStepA04, BiStepA04, BiStepA05, or BiStep2A04 (with no JS jumper), make sure that the socketed serial driver chip is installed (the 16 pin chip which is located directly beside the DB9 serial connector, and has a part number of MAX232 or MAX202).
   - If the board is a BiStepA06, BiStep2A05 (with a JS jumper), or SS0705, make certain that the JS jumper is installed. This is the jumper close to the SI/SO pins on the IO connector.
3. Wire, at the very least, the power for the logic system. You may include the power for the motor system, but this is not required for this test. If you only supply power to the logic system, then you need to supply 7.5 to 15 volts to the +Vc/Va pins on the power connector, with the power supply ground connected to one of the GND pins on the same connector. For all boards except the BiStep2A series, 0.3 amps is enough for this supply; the BiStep2A requires 1.0 amps for its logic, as it also runs the fan off this power source. See the Universal Stepper manual relevant to your board for information on the location of this power connector. ***Do not apply power yet***. Before proceeding, ensure that the power supply is correctly connected and not reversed. ***Important: If you wire the board backwards, you will probably damage one or more of the parts. (A possible indicator that this has occurred is blue smoke emerging from one or more components.) This is NOT covered by our warranty!***
4. Temporarily apply power, and observe whether or not the "power" LED mounted on the board illuminates. Regardless of whether or not the test succeeded, turn power back off. If the LED failed to turn on, then your power supply was not delivering power, it does not have enough current capacity, it is wired incorrectly (i.e., backwards), or the board is defective. Please check and adjust as needed; if the problem seems to be our board, please contact us for advice. (Contact can be done through our website at http://www.pdncons.com; further contact information is available in the Overview section of this document.)

Set up a terminal emulator on your computer

If you are using the Windows[tm] operating system, you may use Hyperterminal to test our board. Otherwise, use whatever terminal emulator is available on your system as your interactive test tool. (This document cannot provide direct assistance for other operating systems.)

Setting up Hyperterminal in Windows

Set up Hyperterminal as follows:

a. Load "Hyperterminal". On most installations of Windows, you will find this under "Start", "Programs", "Accessories", "Communications", "Hyperterminal".

b. Usually, under "Hyperterminal", Windows will have several preconfigured options. Select the one labeled "Hyperterminal".

c. When Hyperterminal first loads, it will ask for a new connection name. We suggest that you enter something descriptive, such as "9600 Baud Direct Connection".

d. It will next ask for a "Connect to" request. Select the COM port connected to the SimStep or BiStep unit.

e. It will then ask for the port settings. Select:

   i. 9600 baud

   ii. 8 data bits

   iii. no parity ("none")

   iv. 1 stop bit

   v. flow control off ("none")

f. After you have completed the port settings page, Hyperterminal will go to its emulation page (an empty window waiting for serial input data or for text which you type).

g. Select the "File->Properties" page, and then the "Settings" tab within that page.

   i. Adjust the "Emulation" to be TTY.

   ii. Press the "ASCII Setup" button, and select the "echo type characters locally" option. Press OK twice (for each page which has appeared) to return to the Hyperterminal emulation page.

h. Do the "File->Save" action, to save your settings in the Hyperterminal folder off of the Start menu. This will save your settings so you do not have to repeat this set of actions every time you wish to use your board. (You will need only to load your just-saved file.)

Setting up some other emulator

Set your terminal emulator to:

a. 9600 Baud

b. No parity

c. 8 data bits

d. 1 stop bit

e. No flow control (i.e., no automatic hardware or software handshake is used to control transmission or reception of serial data)

f. Terminal emulation should be TTY (i.e., characters are purely displayed; there are no motion keys)

g. The emulator should be set to echo typed characters locally; otherwise, you will not see what you type.

For information on how to adjust these settings, refer to your terminal emulator's documentation.

Test the board using the emulator

At this point, you should be ready to test the board.

1. Use a "straight-through" (connects pin 2 to pin 2, pin 3 to pin 3, and pin 5 to pin 5) 9-pin serial cable to connect to your computer. (If you are using a USB-to-serial converter cable, you must connect the 9 pin end of the cable to our board and the USB end to an available USB port on your computer)
2. Insure that power is not being supplied to the board at this point.
3. Start your emulator as described in the earlier sections of this manual, if it is not already started.
4. Apply power to the board. If you have connected it correctly, a sign-on copyright message will appear on the terminal emulator, along with the version number for the firmware. Please write down the firmware name and version number, as an added check that you are using the correct product.

If the copyright message does not appear, recheck your work. Assuming that all of the above steps have been properly followed, the most common failures are a defective serial cable, or you are not providing 7.5 to 15 volts to the +Vc/Va pin of the input power connector.

If the copyright message appears, type the following command into the emulation window of your terminal emulator:

-12?

This is the common command across all versions of our firmware to redisplay the copyright message. Accordingly, as soon as you type the "?", another copy of our copyright message should appear. If it does not, then transmission of data from your computer to the board is being blocked. The most common cause of this failure is forgetting to install the JS jumper on our board. Assuming the jumper is installed, you need to contact us for customer service should the message fail to appear; your MAX232/MAX202 driver chip may be defective.

If the copyright message displayed both times, your board has passed the simple serial test.

Calculating Current Requirements

This section describes how to calculate the power requirements for your motors, and for the system as a whole. It does not take into account the higher voltages which are used when operating in ½ power mode, but it does show what currents will be needed.

1. Determine the individual motor winding current requirements.

The first issue is to determine the individual winding current requirements for your stepper motor. Since our system does not monitor current at all (it only estimates current, using a PWM-like technique), the current ratings as seen by our board may not match those specified by a manufacturer who is assuming that current-monitoring based control is being performed.

From the point of view of determining the current requirements for your motor, our system is best modeled using the standard resistor-only based formula (ignoring inductance) of:

V=IR

or, rearranging terms in order to find I,

I=V/R

That is to say, the current (I) as seen by our board equals the voltage (V) from your power supply divided by the resistance (R) of your motor windings. This value can be much greater than that claimed by a given motor manufacturer, since most of them assume that you are using a current-controlled system to run their motors.

For example, if you have a 3 ohm resistance in your windings, then the motor will "draw" 6/3 or 2 amps if 6 volts is driven into it, and it will draw 12/3 or 4 amps (per winding!) if 12 volts is used.

2. Determine current requirement for actually operating the motor(s)

Once you have determined the motor current, then you will need to determine how you intend to run it via our product offerings. We have four modes of operation, which provide for three levels of power per motor. These modes are controlled by the "o" command, which specifies the technique used to drive the windings.

Update Order	*Absolute Current Multiplier*	*Recommended Current Multiplier*
0 (single winding full step)	1.0	1.4
1 (half step: alternate 1, 2 windings)	2.0	2.5
2 (full step: 2 windings at a time)	2.0	2.5
3 (microstep)	1.7	2.0

Note that the “Recommended Current Multiplier” column in the above table includes a "fudge factor"; we always recommend using a power supply which is somewhat larger than the absolute minimum required, in order to avoid overloading issues.

Obviously, if you are going to run multiple motors off of one supply, you will need to add together all of the currents needed in order to determine how large of a supply to use.

For example, if you are going to microstep (mode 3) a motor whose winding current has been calculated to be 0.4 amps, then your power supply needs to be able to supply 2 x 0.4, or 0.8 amps to drive that particular motor.

3. Determine the voltage for your motor power supply

No, this is not "who is buried in Grant's Tomb". With the exception of the SimStepA04 and SS0705 units, all of our drivers lose some of the voltage from the power supply before delivering it to the motor. The amount of loss depends on the controller, the current being supplied as determined by the above I=V/R formula, whether the motor is being operated using our 'double current mode', and whether the motor is being connected to the controller as a Unipolar or Bipolar drive based system.

The approximate voltage drop for a bipolar motor driven by the BiStep series of products operating in both standard current mode and double current mode can be estimated from the following table: please note that these are only estimates, and will vary somewhat based on actual components, motors, and temperature. A unipolar motor will cause a voltage drop of approximately 1/2 of the amount shown.

Standard Current Mode, Single Winding Current	*Double Current mode, Single Winding Current*	*Approximate Voltage Drop for a BiPolar Motor*
0.5	1.0	Up to 2 volts
1.0	2.0	Up to 3 volts
2.0	4.0	Up to 5 volts

For example, from the above table we would estimate that at most a 15 volts supply should be used to drive a 12 ohm bipolar motor at 1 amp of current when operating in the standard (double motor) mode of operation. Probably, for safety, we should use a supply slightly less than that if any unit other than the BiStep2A is used as the controller, since this is right at the limits of the BiStepA06 system.

Remember, however, that the above numbers are valid for the non-1/2 power mode of operation. If you are operating in ½ power mode, you will have to use an ammeter and a variable power supply to determine the correct voltage setting to deliver the desired current to your motor.

4. Determine the logic supply requirements

The current needed by the logic portion of our product offerings depend upon the product. All of the units except for the BiStep2A need at most 0.4 Amps for their logic (that provided via the Vc or Va connection); the BiStep2A requires 1 amp for this connection.

5. Determine the power supplies you will be using

Your choices are dependent on the desired voltage to the motors, and on the board which you have purchased from us. In all cases, we strongly recommend that linear supplies be used: switching supplies are not very good when used with inductance based loads.

Single Supply.

If your motor power supply voltage is from 7.5 to 15 volts, then you may choose to use a single supply to operate the system (on the BiStepA04, you may only use a single supply; therefore, you must use a supply which is in this voltage range). Obviously, the current capabilities of the supply must exceed the sum of the current requirements of the motor(s) and the logic circuits.

Dual Supply

For all of the other units, you may separate the motor supply from the logic supply. If you do so, we suggest using the lowest voltage in the range of 7.5 to 15 volts on the logic supply which you have available, to reduce generation of waste heat on the board.

The motor supply depends on the product and your needs. The value is normally as calculated under sections 1 through 3, above. If the supply is to drive 2 motors, please remember to double the current needs. The voltage limits allowed on the motor supplies are:

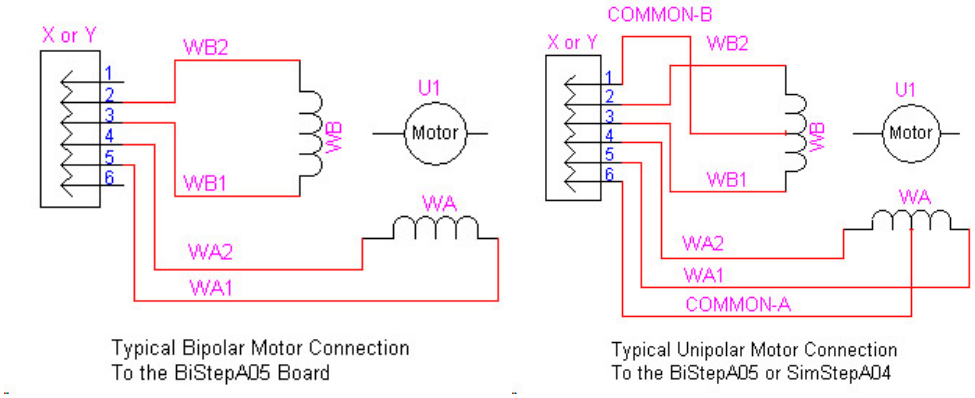
Board	Minimum Motor Supply Voltage	Maximum Supply Voltage
SimStepA04 or SS0705	6.5	30
BiStepA04	7.5	15
BiStepA05	5	36
BiStepA05-1A	5	36
BiStepA06	5	36
BiStep2A	7.5	46

Triple Supply

The BiStep2A has the additional capability of allowing use of three different power supplies. It is quite possible to have one motor operate at a different voltage level than the other, if this turns out to be a requirement. Note, however, that if you operate the BiStep2A in DOUBLE POWER mode, then one supply must be used to operate the power for both motor driving circuits: otherwise, you will cause failure of the board and/or the power supplies! Please note that the BiStep2A supports a motor power supply voltage range of 7.5 to 46 volts.

Wiring Your Motor

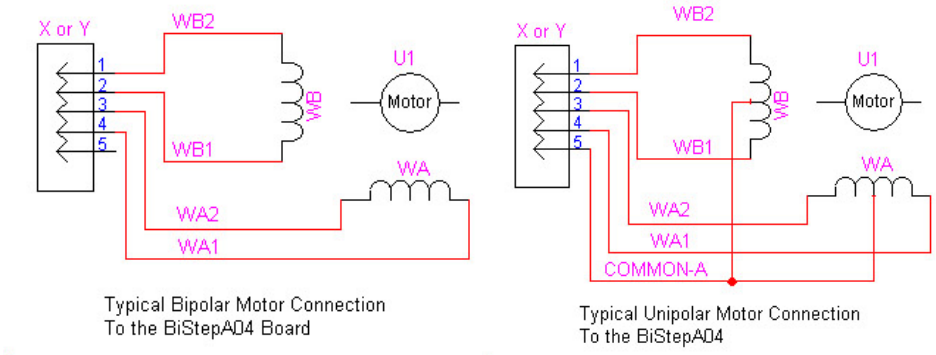
There are two identical connectors used to operate the X and Y motors. The connectors are labeled with respect to which motor they operate. (This designation affects only which commands are to be used to control the motors.) They are wired as follows for the SimStepA04, SS0705, BiStepA05, BiStepA06 and the BiStep2A series of controllers (pins counting from top to bottom):



Pin	Name	Description
1	GND or +V	If BiStep board, GND; if SimStep or SS0705 board, +V
2	WB2	Winding B, pin 2
3	WB1	Winding B, pin 1
4	WA2	Winding A, pin 2
5	WA1	Winding A, pin 1
6	GND or +V	If BiStep board, GND; if SimStep or SS0705 board, +V

This pinout was selected to allow simple reversing of the connector (i.e., take it out and turn it around) to reverse the direction of the motor if a non-polarized connector is used.

The BiStep A04 board does ***not*** support the above reversing functionality. The "top" ground pin is removed (the connectors are 5 pins instead of 6), generating the pinout:



Pin	Name	Description
1	WB2	Winding B, pin 2
2	WB1	Winding B, pin 1
3	WA2	Winding A, pin 2
4	WA1	Winding A, pin 1
5	GND	Ground return

Stepping sequence, testing your connection

The current is run through these connectors to generate a clockwise sequence as follows:

Step	*WB2*	*WB1*	*WA2*	*WA1*
0	0	0	0	**1**
1	0	**1**	0	**1**
2	0	**1**	0	0
3	0	**1**	**1**	0
4	0	0	**1**	0
5	**1**	0	**1**	0
6	**1**	0	0	0
7	**1**	0	0	**1**

The actual wiring configuration to connect to a given stepper motor depends on the motor type. For most unipolar motors, each winding has three leads. The center-tap (shown in the above schematics as "COMMON-A" or "COMMON-B") is connected to the GND signal in the BiStep series controllers, or to +Vm on the SimStep/SS0705 series of controllers. The other two leads are connected to pins WA-1 and WA-2 or WB-1 and WB-2, as shown in the above schematics. For bipolar motors, the windings match the labels – that is to say, pins 2-3 are for winding B, and 4-5 are for winding A. Note that the unipolar motors will also match the labels, but it may be more difficult to identify the windings.

For a unipolar motor, the common line may be identified as distinct from the direct winding lines by its having ½ of the maximum resistance seen between pairs. A 6-wire 4-phase unipolar motor will have two "common" wires. You will normally connect one of the wires to pin 1, and the other to pin 6. Note that you can usually operate a 6-wire unipolar motor as if it were a 4-wire bipolar motor (when using the BiStep series of controllers) by insulating the common leads and leaving them disconnected. This usually provides more torque for the motor, but it requires double the voltage (at the same level of current) from the power supply. You cannot operate with this pair of wires disconnected if they are connected together inside the stepper motor; if the resistance between the common leads is very low (less than 10 ohms), such a connection exists and you must therefore operate using the unipolar wiring scheme.

For a bipolar motor, if a wiring diagram is not available, an ohm-meter can be used to determine the windings. The low-resistance pairs are the opposite ends of matching windings; high-resistance pairs are different windings.

Always double check all of your power and motor connections before you apply power to the system. If you have reversed any power leads, you will both blow out our board and you may blow out your power supply! If you are operating a unipolar motor and you short a common lead to a winding pin (WA or WB), then you will blow out our drivers! Similarly, any winding which is shorted with any other winding may burn out our board. If you are setting up to use double-power mode (connecting one motor to both the X and Y motor connectors in order to drive a larger motor), failure to connect the LY- and LY+ limit inputs both to ground will also cause the board to fail. None of these issues are warranted failures!

Once winding lines have been determined, identifying a running sequence can always be done via testing the lines using following sequence, connecting to the X motor connector. Use of clip leads can simplify the process. ***Turn off power*** to the board in between each test, so that power is not on while you change the wiring.

For wires A, B, C, and D (where A, B, C, and D are initially connected to the WA1, WA2, WB1, and WB2 lines) try these orders:

	WA1	***WA2***	***WB1***	***WB2***
1.	A	B	C	D
2.	A	B	D	C
3.	A	D	B	C
4.	A	D	C	B
5.	A	C	D	B
6.	A	C	B	D

For each pattern, request a motor motion in each direction using the appropriate one of the following techniques:

- GenStepper Firmware, using a terminal emulator:
  - Issue the command “x1000gi0gi”, which should cause the motor to spin to logical location 1000, then back to 0.
- GenStepper Firmware, using switch closure to ground of the slew lines:
  - Short the +X to ground (observe the motor spinning), then release and short the –X to ground, then release. These signals are found on the “slew” connector, or on the 19 pin header in the middle of the board (depending on your board).
- NCStepper Firmware, using a terminal emulator:
  - Issue the command “1000xg0xgi”, which should cause the motor to spin to logical location 1000, then back to 0.

Only when the motor is wired correctly will you get smooth motion first in one direction and then the other.

Once a possible pattern has been determined, you may find that the direction of rotation is reversed from that desired. To reverse the rotation direction, you can either turn the connector around (this may be the easiest method, if a SIP style connector is used), or you can swap both the WA (swap pin 2 with pin 3) and WB pins (swap pin 4 with pin 5). For example, to reverse

A B C D, rewire as

B A D C.

Motor Wiring Examples

Please see the UniversalStepper manual pertaining to your board hardware for descriptions of how some motors we tested are wired into the system.

Simple SerRoute Test

Our SerRoute firmware changes the personality of any of our stepper motor controller boards into that of a serial router. The RS232 input port of the SerRoute board is normally connected to your computer, while the various I/O lines get connected to other of our products as described in the SerRoute manual.

As a simple initial test to get you familiar with use of the product in this configuration, set up a 3 board system:

- The SerRoute board should be connected via RS232 to your computer, using your terminal emulator (as described earlier).
- The other two boards (for the sake of argument, let us assume that they are both GenStepper units, connected to motors so you can see the results of your commands) get connected to the SerRoute board logical serial lines 0 and 1 as documented in the SerRoute manual.
- Make sure that all of the boards share a good common ground (GND) for their power supplies; otherwise, serial communications will be unreliable. For this test, it is often easiest to run all of the boards off of one power supply, if possible.

When a SerRoute-based board is connected to one of its "child" boards, the connection is normally done with the "TTL-Serial" technique. You therefore need to refer to the "TTL-Serial Information" section of this manual to see how to disable the RS232 serial mode of the GenStepper boards, and to identify the locations of the "SI" and "SO" signals on those boards to use for the TTL-serial connections. The connections to those boards become:

Serout Signal	Target Board	Target Board Signal
Y-	0	SO
Y+	0	SI
X-	1	SO
X+	1	SI

Make absolutely sure to disable the RS232 serial on target boards 0 and 1. Otherwise, you will be overloading the drivers associated with the Y+ and X+ lines on the SerRoute board. (This could potentially damage the board!)

When you power on the system (assuming that you have your board connected to your terminal emulator running on your computer), you should see the "copyright" message associated with the "parent" SerRoute board appear on your screen. Issue the command:

{0}-12?

You should see the copyright message associated with your board identified as board "0" in the above table (the one connected to the SerRoute Y-/Y+ signals). If you now issue a motor motion command, such as:

1000g *(1000xyg, if your board is NCStepper based)*

you should see the motors associated with that controller move.

Issue the command:

{1}-12?

You should see the copyright message associated with the second board. Since you cannot tell the difference between GenStepper copyrights, issue the command

-1000g *(-1000xyg if your board is NCStepper based)*

You should see the other set of motors spin.

Finally, you may issue the command:

{}-12?

This will show you the copyright message for the SerRoute based boards. At this point, all commands are directed to SerRoute, thus allowing you to control its relays and monitor its TTL input lines.

Please note that you only need to send the "{}", "{0}", and "{1}" commands to switch boards targeted by any commands which follow: the "-12?" was merely an example to demonstrate some response.

Serial Operation Details

The RS-232 and TTL-Serial communications features allow for full access to all internal capabilities of the system. It operates at 2400 or 9600 baud, no parity, 1 stop bit, and no flow control. Note that you must wait about ¼ second after power-on or reset to send new commands to the controller; the system does some initialization processing which can cause it to miss input during this "wake up" period.

Serial input either defines a new current value or executes a command. The current value remains unchanged between most commands; therefore, the same value may often be sent to multiple commands by specifying the value, then the list of commands. For example,

1000XY

would mean "Set the next locations to X=1000, Y=1000", when issued to a board with the NCStepper firmware installed.

The firmware only recognizes and responds to each new command at about ¼ of the way through the stop bit of the received character. This means that the command starts being processed at ¾ bit-intervals before completion of the character bit stream. In most designs, this will not be a problem; however, since all commands issue an '*' upon completion, and they can also (by default) issue a <CR><LF> pair before starting, it is quite possible to start receiving data pertaining to the command before the command has been fully sent! In microprocessor, non-buffering designs (such as with the Parallax, Inc.[tm] Basic Stamp [tm] series of boards), this can be a significant issue. The firmware handles this via a configurable option in the 'V' command. If enabled, the code will "send" a byte of no-data upon receipt of a new command character. This has the result that the first data bit of a response to a command will not occur until at least 7-8 bit-intervals after completion of transmission of the stop bit of that command (about 750 uSeconds at 9600 baud); for the Basic Stamp[tm], this is sufficient for it to switch from send mode to receive mode.

Selecting Baud Rate

By default, the system is fixed at 9600 baud, no parity, and 1 stop bit. Operation at 2400 baud may be selected during the initialization process by adding a resistor between the RDY pin and ground. During reset (from power on, hard reset, or processing of the "!" command), RDY is temporarily used as an input to determine the baud rate. If it is tied to ground via a 1K resister, then the baud rate is set to 2400 baud. If it is left to float (the default), then the baud rate is set to 9600 baud.

TTL-Serial Information

In order to operate in "TTL-Serial" mode (to operate our products without use of RS232 levels of signals), you must disable use of the MAX232 (or the equivalent MAX202) chip on our board.

If you have a SimStepA04, BiStepA04, BiStepA05, or the BiStep2A04 (the version **without** a JS jumper), then you must remove the MAX232/MAX202 chip from its socket. This 16-pin chip is located directly beside the DB9 connector, and is in a socket that allows it to be easily removed and reinserted. Be sure to remove it carefully, so it can be later reused in case you need to re-enable RS232 operations. It is a static sensitive part, so you will need to save it in a static-safe container.

If you have a BiStepA06, SS0705, or BiStep2A05 (the version **with** the JS jumper), then you need only to remove the JS jumper.

Once you have disabled the RS232 serial, you then connect your TTL-serial from your controller (such as the Parallax Basic Stamp^{tm}) to the SI and SO pins of the board.

Connect the TTL-serial **output (**sent from your controller to the board), to the "SI" (Serial Input) signal of the board. Similarly, you connect the TTL-serial **input** (sent from the board to your controller) to the "SO" (Serial Ouput) signal of the board. Note that on the "Release 1" boards (SimStepA04, BiStepA04 and BiStepA05) the SO and SI signals are found on the long 19 pin header in the middle of the board. See the section "Labeling of Board Signals" near the start of this manual for more information.

Please remember to re-enable the RS232 operation of the board (and to disconnect your controller) if you need to use Hyperterminal (or some similar tool) on your computer to do diagnostics on the board; if you do not, the board will be unable to respond to RS232 commands.

Getting a Parallax Basic or Javelin Stamp Running

Most customers using our stepper motor controllers have no problems operating them with Parallax Basic Stamp or Javelin Stamp products. A few, however, have had some issues; this section summarizes the steps which you should go through, if your first attempt does not appear to work.

Before you connect the Parallax Basic Stamp (or Javelin Stamp) to our products, you should first go through the following standard procedures (also listed at the beginning of this manual).

1. Read the "**Product Safety Warnings**" section of the "Universal Stepper" manual pertaining to the board you have purchased.
2. Read the "**Board Connections**" section of the same manual, in order to understand the various connectors on your board. Note that this manual ("First Use") has a section entitled "**Labeling of Board Signals**", which may help you understand where various signal lines may be found.
3. Do a "**Simple Serial Test**" of the board, as described earlier in this manual, to confirm RS232 serial operations of the product. *Do this step even if you intend to use TTL-serial with some other device, such as a Parallax Basic Stamp, as your control system.*
4. If you are using the GenStepper or NCStepper firmware (these are the normal firmware options), please refer to the "**Wiring Your Motor**" section in this manual. This section will allow you to prove that the board is correctly operating your motor and will show you how to determine the correct wiring. You may also wish to review the "**Motor Wiring Examples**" found in the "Universal Stepper" manual.
5. If you are using a SerRoute firmware based board to control one or more of our other boards, then please refer to the "**Simple SerRoute Test**" section of this manual for an initial guide to preliminary tests to demonstrate how to use SerRoute to operate multiple boards via one RS232 interface.

Once the board has passed the above set of tests, you can connect it to your Stamp. The Stamp products are usually connected to our board via the "TTL-Serial" technique, described earlier in this manual. In summary:

1. Use a linear power supply, not a switching power supply.
2. Remove the JS jumper from our board (Jumper Serial) if you have the BiStepA06, BiStep2A05 or SS0705 unit. If you have the BiStepA04, BiStepA05, BiStep2A04 or SimStepA04, remove (and save) the 16-pin Max232 serial controller chip located right by the 9 pin serial connector.
3. Make certain both the Stamp and the board's logic are run off of the same power supply.
4. Make sure your power supply is large enough to meet the demands of your application.
5. Keep your leads between the two units short . TTL signals are only "clean" (that is to say, little "bounce" or induced voltages) at up to 3 feet. Note, however, that we have successfully operated via TTL serial at distances of up to 26 feet.
6. We suggest running communications at 9600 baud.
7. Make certain that you have adjusted our sample code to match your stamp product. Different stamps require different settings for the "PortStepperBaud" parameter used in our samples.

Discussion:

1. Use of a linear power supply (as opposed to a switching power supply) is a very strong recommendation. Switchers generally do not have the short-term power reserves needed to handle the extreme variances in power demanded on motor startups, and they are usually much "noisier" from an electrical point of view. We have had no reported power-supply related problems when our customers use linear supplies of adequate size; we have had frequent issues when switchers are used.
2. Removal of the JS jumper or the Max232 chip is needed to avoid connecting the outputs of two chips to each other. This generates what is called a "wired or", and can cause eventual failure of the chips involved.
3. Using a common power supply for the boards avoids ground referencing problems. *This is the single most common cause of issues when using the stamp*. If separate supplies must be used, then you must be certain that you connect their grounds together with a reasonable size wire, in order to make certain that both boards "float" at the same potential. Otherwise, TTL signals will not be correctly interpreted, and serial communications will fail.
4. We have frequently had customers complain that their motor stalls or does not appear to have enough power. In most cases, this has been traced to the customer having selected a power supply which can only deliver about 1/2 of the current required by the motor. Remember that a stepper motor can have 2 windings on at a given time; this means that it can draw twice as much current as you would expect, referring only to the current rating of the motor. Please refer to our "Universal Stepper" manual section "Calculating Motor Currents" for a more

complete summary of how to calculate the current requirements for your application.

5. TTL signals are quite susceptible to induced voltages from antenna effects, impedance matching issues, and from neighboring signals. In general, when the signals are kept to under 3 feet, the amount of induced signals are too small to cause a logic circuit to trigger. When operating using TTL-Serial mode (as opposed to RS232 serial, which uses much larger voltages) on a 26 foot cable, we have found that there can be about a 0.1 microsecond wide 5 volt spike induced onto one signal line when a neighboring signal line changes state (i.e., when a serial bit is sent to our board, a narrow spike can be induced on the serial output line from our board to your stamp). Due to the nature of the serial timing between the products, this spike is normally ignored under serial I/O (our firmware rejects it via a digital filter, and the timing of serial I/O to and from the stamp is such that the stamp would not see it). Our suggestion of 3 feet or less avoids this potential issue.
6. The most recent versions of the firmware (GenStepper versions 1.65 and above) have enough idle time built into it when operating in the suggested modes for Stamp communications that you should have no problems. Operate at 2400 baud only if you cannot get reliable operation at 9600 baud (and please contact us for further assistance in this case).
7. When you edit the sample code for your system, you need to look for the line which contains the "{$STAMP" command, and adjust it to match your stamp (such as BS2, BS2sx, etc.). You then need to edit the line which defines the "PortStepperBaud" parameter to contain the correct value for your stamp. Please refer to your stamp manual for the values. For some of the stamps, the values are:

Stamp Unit	***9600 Baud***	***2400 Baud***
BS2/BS2e	84	396
BS2sx/BS2p	240	1021

For example, to generate 9600 baud on a BS2sx, you need to have the PortStepperBaud line contain:

```
PortStepperBaud con 240 ' Baud rate to generate 9600 baud on BS2sx
```