

KT-5190 Serial Stepper Motor Controller

Until now it has been difficult to control stepper motors from a computer. The stepper motor kits sold today require you to interface the step and direction inputs to a parallel port and then write the software to switch these signals. If you need to incorporate limit switches and acceleration and deceleration of the stepper motor, what started out as a simple job turns out to be a weekend job and you are in the dog house because you haven't spent time with the wife and kids.

The Serial Stepper Motor Controller (SSMC) overcomes this by providing control for up to four stepper motors from a single controller board attached to the serial port, and does away with the need for software to control the acceleration and deceleration of the motors. Additionally, up to four Serial Stepper Motor Controller boards can be ganged together and individually addressed to provide control for up to sixteen stepper motors, all from the one serial port. You couldn't do that from the quickly disappearing parallel port and with the availability of cheap USB to serial converters the controller could easily be adapted for use on any USB port, hence future-proofing it! Figure 1 shows how the SSMC board is connected

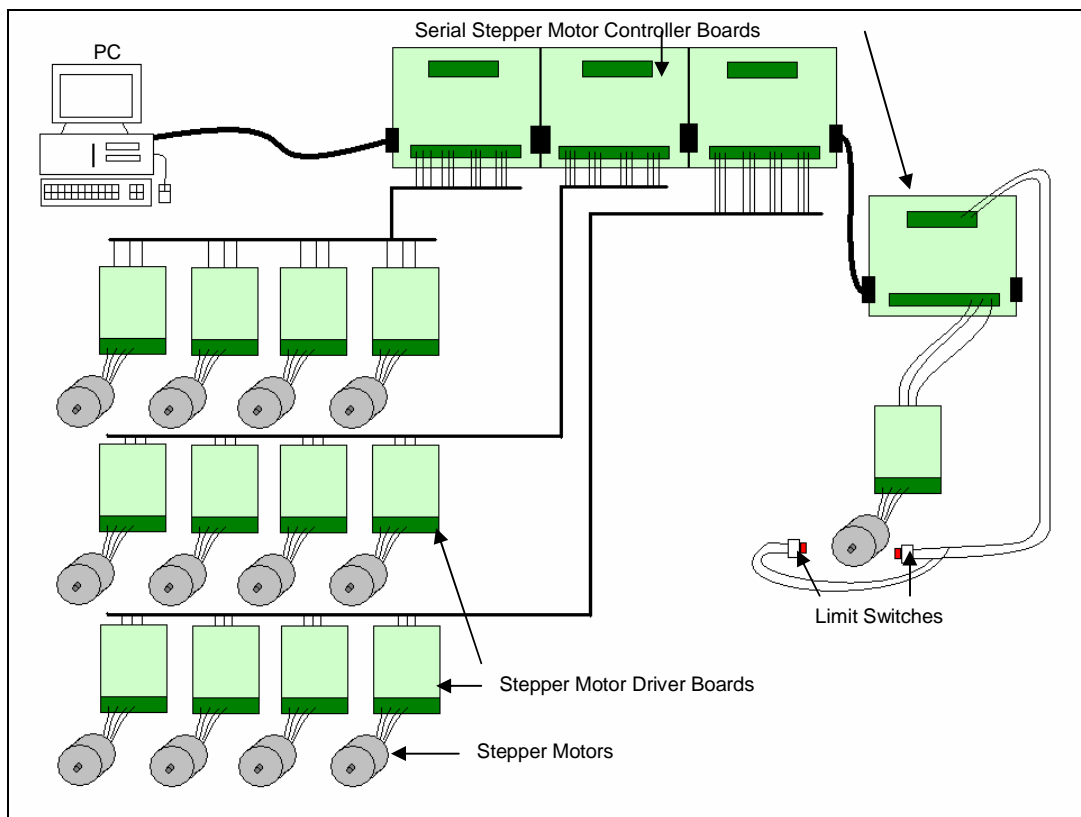


Figure 1 - Multiple Controller Connection Scheme

Stepper Motors:

Lets start with a brief look at stepper motors. Unlike regular DC or AC motors which have commutator brushes to switch the coils in the motor on and off automatically, stepper motors require individual activation of coils, by the use of a stepper motor driver circuit, and have no brushes.

The major advantage of stepper motors over regular motors is their finite position control, which is virtually impossible with a regular motor.

There are two main types of stepper motor, Unipolar and Bipolar.

Unipolar Stepper Motors:

Unipolar stepper motors have two centre tapped coils inside, and these behave as four individual coils. These motors have either five, six or eight wires. Five wire motors join the two centre taps together, six wire motors bring out the centre taps individually, and eight wire motors have four individual coils, with a wire for each end of each of the coils.

The four coils of a unipolar stepper motor are individually activated and deactivated sequentially and each time this is done the motor advances a step.

Unipolar stepper motors can act as bipolar stepper motors if the centre taps are omitted.

Bipolar Stepper Motors:

Bipolar stepper motors are stepper motors with only two coils and have four wires. Since there is only two coils the driving of bipolar stepper motors is somewhat different to that of unipolar stepper motors, the main difference is that the current flowing through the terminals must be able to be reversed.

Unipolar stepper motors can generally have more steps than bipolar stepper motors of the same size, but bipolar stepper motors can provide more torque.

Using The Controller:

There are a few stepper motor driver kits available which can control stepper motors. Kits KT-5179 (Unipolar), KT-5158 (Bipolar) and KT-5191 (Bipolar Chopper) from Ocean Controls all have Step and Direction inputs which allow the user to control the movement of the motor.

The Ocean Controls Serial Stepper Motor Controller does not provide the driving circuitry for individual stepper motors, but rather connects to the step and direction inputs on the driver cards.

The Circuit:

Refer to the attached circuit diagram for details. Incoming data on the serial port is converted to 0-5V microcontroller friendly levels by the MAX232 chip. The data is then processed by the Atmel ATmega168 microcontroller. The DIP switch SW1 is used to address the board when multiple Serial Stepper Motor Controller boards are used together. The limit switch inputs have pull up resistors and go through R1-R4 for protection. Step and Direction outputs connect to the terminals through the 74HC245 buffer chip, which protects the microcontroller from accidental damage.

K2 is a 9-pin female D-connector, this connects the circuit to the PC (via an RS232 cable). K3 is a 9-pin male D-connector which allows additional controller boards to be connected. The resistor R5 and diode D2 allow multiple devices to be connected to the serial link (See the section on multidropping for more detail).

The terminal connectors provide connection for power (12V DC) at terminal Vs, limit switch inputs (L1-L4), step outputs (S1-S4) and direction outputs (D1-D4).

Using the Controller:

The Serial Stepper Motor Controller is controlled via the serial port of a personal computer, using any serial terminal software or custom software, set at a baud rate of 9600, 8 data bits, 1 start bit, 1 stop bit and no parity (9600,8,N,1).

The commands for the controller are in the form:

@AA CMND XXXXCR

Where AA is the 2 digit number of the motor being addressed, between 01 and 16 (see Table 1), CMND is the 4 letter command, refer to Table 3 for available commands, XXXX is a numeric value associated with the command (refer to Table 2 for detail), and CR is the Carriage return byte (0x0D) .

Table 1 – Addressing

S2	S1	Motor Numbers
OFF	OFF	01-04
OFF	ON	05-08
ON	OFF	09-12
ON	ON	13-16

Table 2 – Commands

Command	Description
POSN	Set the position that motor AA is currently at to be XXXX where XXXX is between -99,999,999 and 99,999,999
PSTT	Returns the position of motor AA
AMOV	Move motor AA to the absolute position XXXX where XXXX is between -99,999,999 and 99,999,999
RMOV	Move motor AA relatively from the current position by XXXX where XXXX is between -99,999,999 and 99,999,999
STOP	Stop motor AA immediately
STAT	Get the status of the motors see "Status Command Detail"
ACCN	Set the maximum stepping rate of motor AA to XXXX where XXXX is between 0 and 9999 see "Acceleration". If the value for ACCN is 0 or less than RATE then no acceleration or deceleration occurs
ACCI	Set the Acceleration interval of motor AA to XXXX where XXXX is between 1 and 9999 see "Acceleration"
RATE	Set the minimum stepping rate of motor AA to XXXX where XXXX is between 1 and 9999 see "Acceleration"
DRON	Turn Direction output (D1-D4) associated with AA on for XXXX * 0.131mS. If XXXX is -1 output will be on until a 'DROF' command is received
DROF	Turn Direction output (D1-D4) associated with AA off immediately from 'DRON' command
DRST	Returns time remaining of 'DRON' command in time intervals, or -1 if it is on permanently
OPTN	Bit 1 of the parameter turns Verbose mode on and off, bit 2 turns Checksum mode on and off. Valid addresses are 01, 05, 09 and 13. See the section on the Option Command
ACCF	When set to 1 the alternate Acceleration Curve is used, when set to 0 the standard Acceleration Curve is used.
SAVE	Saves RATE, ACCI, ACCN, ACCF and OPTN parameters to EEPROM, which are then automatically loaded on the next power up. Valid addresses are 01, 05, 09 and 13.

Note: As of November 2006 the time intervals have been changed from 1ms to allow for a higher speed. See the section titled "**High Speed Version**" on page 6 for details on how to calculate the time interval.

When a valid command is executed by the unit it will respond with the address preceded by a hash symbol, ie. #AA and this will be followed by a value if it is requested.

Status Command Detail:

The status command returns the state of each of the motors attached to a single controller board. Valid stat commands have the address of the first motor on the board, eg:

@01 STAT returns the status of motors 01 to 04

@05 STAT returns the status of motors 05 to 08

@09 STAT returns the status of motors 09 to 12

@13 STAT returns the status of motors 13 to 16

These are the four valid status commands.

The returned value is a 12 bit binary representation of whether the motors are moving, their direction and the state of the limit switches in the following format:

Table 3 – Status Value

msb	11	10	9	8	7	6	5	4	3	2	lsb
L4	L3	L2	L1	D4	D3	D2	D1	M4	M3	M2	M1

Where M, D and L represent the movement, direction and limit switches respectively.

For movement 1 = moving, 0 = stopped.

For direction 1 = forward, 0 = reverse.

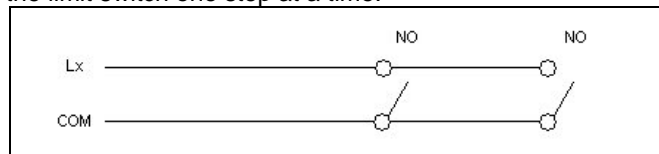
For the limit switches 1 = closed, 0 = open.

Multidropping:

Multidropping is the term used when connecting multiple slaves to one master. This is achieved by including a signal diode on the transmit outputs of the slaves with a resistor to pull the transmit line to ground when it is not being used. The diode prevents voltages produced by transmitted data from the slaves appearing on the transmit pins of the slaves. All the slaves receive the same data from the master and decode it if necessary.

Limit Switches:

The Serial Stepper Motor Controller provides inputs for limit switches for each motor on the board. The inputs have internal pull-up resistors so only require a connection to ground to assert the input. Multiple limit switches can be used on each motor, provided they are wired in parallel (normally open switches must be used). See Figure 4 for wiring detail. When the limiting input is asserted the associated motor stops, and while it is asserted the motor will only perform a single step, for any further command issued. This allows the motor to be backed off the limit switch one step at a time.

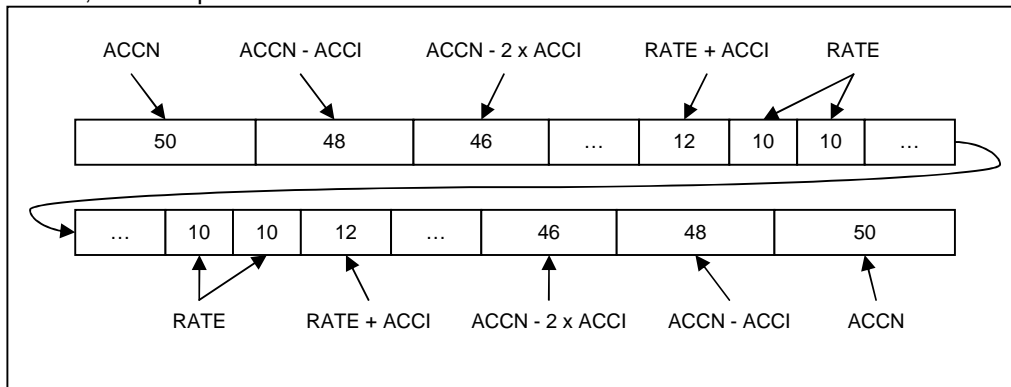


Limit switch wiring

Acceleration Detail:

Each time the command to move a stepper motor is issued the Serial Stepper Motor Controller calculates the stepping times to give a gradual acceleration and deceleration. The acceleration and final speed is determined by the ACCN, ACCI and RATE parameters. These default to 50, 2 and 10 respectively and are changeable by issuing the appropriate command. See the section titled "High Speed Version" on page 6 to see how these Parameters relate to speed.

The motor will start stepping at one step every ACCN interval, decrease this by ACCI intervals every step until the interval is RATE. When the motor is approaching the final position the step interval will increase by ACCI intervals from RATE until the final position is reached, at which point the interval will be back at ACCN.



Step Timing

New Acceleration: Nov 06

An alternate acceleration curve can be activated by using the ACCF command with parameter 1. In this mode the motor starts one step at ACCN, then two steps at ACCN+ACCI, four at ACCN+2xACCI, eight at ACCN+3xACCI etc until RATE is reached. This can be used to reduce the effects of the non-linearity of the normal acceleration. This mode of acceleration will take $2^{((ACCN-RATE)/ACCI)}$ steps to reach top speed. Deceleration is done in the same manner.

Assembly & Testing:

Start by soldering the resistors and diodes and then the capacitors and crystal. Next add the IC sockets, voltage regulator and DIP switch. Finally add the terminals and 9 pin D connectors, noting the placement of the male and female connectors.

If you are going to gang two or more controller boards together you will need to unscrew the screw connectors on the right hand side of all the D9 connectors. This will allow the connectors to mate when pushed together.

To test the controller, connect power to the relevant terminals and measure the voltage across pins 10 and 20 of the socket for the 74HC245, ensuring that it is 5V. If this is ok remove the power and insert the microcontroller, max232 and 74HC245 IC's. Reconnect power and connect the board to the computer using an RS232 cable. Set the DIP switches so they address from 01 to 04 (both off) and then run a terminal program at 9600 baud and type **@01 STAT**. This should return **#01 0** if the kit is working properly and nothing else is connected. If you have an oscilloscope you can give a move command such as **@01 RMOV 1000** and view the pulses at terminal S1 on the oscilloscope, if you don't, connect a driver board to the S1, D1 and COM terminals and connect a stepper motor to the driver board. Giving a move command should now move the motor!

New Commands: Jan 06

Three new commands have been added as of January 2006, these are the DRON, DROF and DRST commands. The DRON command will turn on the direction output for a set time, or turn it on until the DROF command is received. This can be used to turn a relay on for a set time or as a digital output if the connection is not being used for a motor. The DROF command turns the output off. The DRST command returns the time remaining that the output to be on for, or -1 if the output is on permanently.

New Commands: Nov 06

The Option, Acceleration Factor and Save commands have been added.

Turning the Acceleration Factor on with ACCF and parameter 1 will make the controller use the alternate acceleration curve which is described in "**New Acceleration**" on page 4.

The SAVE command has been added so that the RATE, ACCI, ACCN, ACCF and OPTN parameters are saved to the controllers EEPROM. The parameters will be loaded automatically the next time the unit is powered up. "**@01 SAVE**" saves the parameters for motors 1-4.

Option Command: Nov 06

The option command is used to turn the Verbose and Checksum modes on.

"@01 OPTN 1" will turn Verbose mode on and Checksum mode off.

"@01 OPTN 2" will turn Checksum mode on Verbose mode off.

"@01 OPTN 3" will turn both Verbose and Checksum modes on.

Turning the Verbose mode on will cause the SSMC to transmit !AA (where AA is the address) when the motor has reached its target position through an RMOV or AMOV command, or hit the limit switch during a move.

With Checksum mode on the controller will ignore commands unless a XOR checksum byte follows the command immediately after the Carriage Return. The XOR is performed on all the bytes of the command, including the @ and CR.

Eg. The checksum for the command **"@01 RMOV 100"** will be $(64 \text{ xor } 48 \text{ xor } 49 \text{ xor } 32 \text{ xor } 82 \text{ xor } 77 \text{ xor } 79 \text{ xor } 86 \text{ xor } 32 \text{ xor } 49 \text{ xor } 48 \text{ xor } 48 \text{ xor } 13) = 123$ which is the "{" character.

This is generally only useful if you have high level language controlling the SSMC, but not if you are typing the commands as you go. Note also, there must not be a Line Feed character between the Carriage Return character and Checksum character.

It may be handy to know that the checksum for the **"@01 OPTN 0"** command is the "O" character. In case you accidentally turn it on.

High Speed Version: Nov 06

As of November 2006 the Serial Stepper Motor Controller has been upgraded to allow for a higher stepping rate. Kits sold before this date had a top speed of 1mS/step or 1kHz. These kits used an 8MHz crystal. The new version uses a 20MHz crystal and allows for a maximum speed of 0.159mS/step or 6.3 KHz. Time parameters are no longer in 1mS intervals but can be approximately calculated using the following:

$$t_n = 0.159 + [(n - 1) \times 0.1058] \text{ mS}$$

And the frequency can be calculated from:

$$f_n = \frac{1}{t_n} \text{ kHz}$$

Where n= the parameter value for RATE, ACCI, ACCN etc.

The graph on page 7 can be used to quickly choose the rate for a desired operating frequency.

Example program:

An example visual basic program with source code for controlling four motors is available on upon request and can be [easily expanded for control of 16 motors](#) and virtually any application.

Parts List:

- 1 28-pin DIP socket (or 2 14-pin DIP sockets) (U1A, U1B)
- 1 16-pin DIP socket (U3)
- 1 20-pin DIP socket (U2)
- 1 2-way DIP switch (SW1)
- 1 D9 Female right angle connector (K2)
- 1 D9 Male right angle connector (K3)
- 6 3-way terminal blocks (T1, T3-T4)
- 1 2-way terminal block (T2)
- 1 20MHz crystal (X1)

Semiconductors:

- 1 Atmel ATMEGA168-20PU Programmed Microcontroller (U1)
- 1 74HC245 Octal Buffer (U2)
- 1 MAX232 RS232 to TTL Level Shifter (U3)
- 1 7805 5V Voltage Regulator (VR1)
- 1 1N4004 silicon diode (D1)
- 1 1N4148 silicon diode (D2)

Resistors:

- 1 10K SIL pull up network (S1)
- 4 10K (R1-R4)
- 1 18K (R5)

Capacitors:

- 2 22pF ceramic (C1, C2)
- 4 0.1uF monolythic (C3-C6)
- 4 1uF electrolytic (C7-C10)

Notes:

- V1.01 PCB - DIP Switch "ON" goes toward edge of board
- V1.02 PCB - DIP Switch "ON" goes toward terminal block

Resistor R5 may be supplied as 20K or similar

.

Frequency Vs Rate Graph:

This graph can be used to quickly estimate the RATE parameters required for a desired stepping frequency.

