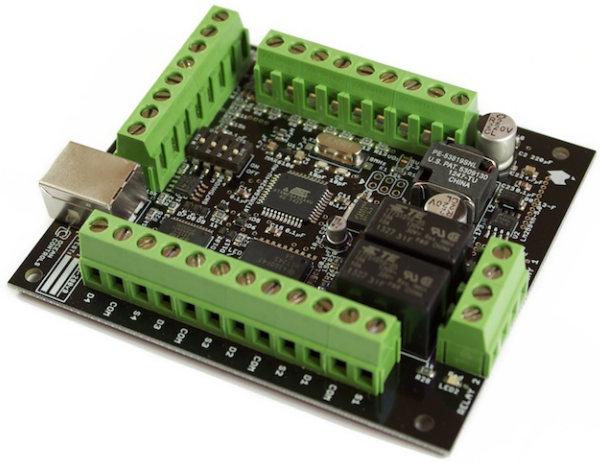

- Controls 4 axes of stepper or servo motors simultaneously
- USB Virtual serial port or RS-485 input
- Easily connect multiple units by RS-485 for up to 16 axes
- Baud rate selectable
- Up to 40 kHz max stepping rate
- Parameters can be stored in non-volatile EEPROM
- Limit switch input for each motor
- 5 VDC step and direction signal levels
- XOR checksum option
- 2 SPST 5 A Relays for control
- 8 to 35 VDC powered

## Introduction

The KTA-290 USB stepper motor controller makes it easy to connect stepper or servo motors to a computer. When connected by USB, the card appears as a serial port on your computer. Simple commands allow configuration of acceleration and frequency settings, relative or absolute moves and control of digital inputs and outputs.

The controller works as a USB to RS-485 converter. Multiple cards can be connected via RS-485, allowing control of up to 16 axes of motors.

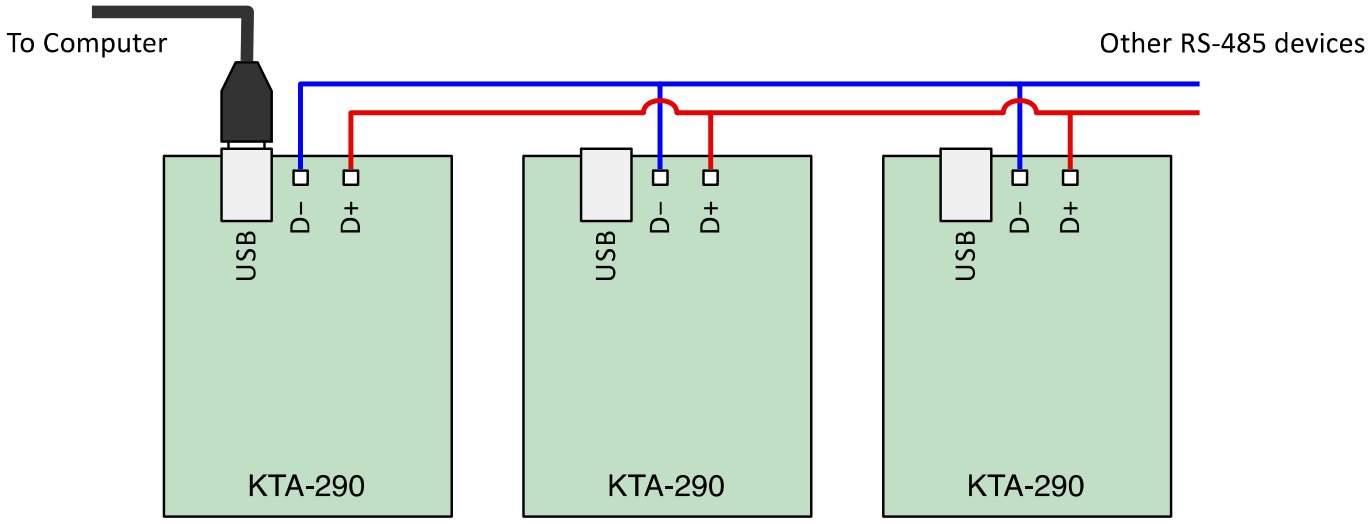


*Figure 1 - Connecting multiple cards to a computer*

## Stepper and Servo Motors

Stepper motors and servo motors require drivers to operate. These commonly take step and direction command signals. The direction signal, high or low, indicates whether to spin clockwise or counter clockwise. Each pulse on the step input of a driver causes the motor shaft to rotate some small fraction of a revolution. A stepper motor controller such as the KTA-290 provides a train of pulses of varying frequency to drive a motor at varying speed.

Over most of the speed range, the torque required from a motor depends on the rotational inertia of the system and the rate of acceleration. Stepper and servo motors can lose steps or desynchronise when more torque is demanded of them than they can provide. To prevent undesired operation, the acceleration must be limited – motor speed needs to be ramped up and down gradually.

## Connections

The most basic configuration of the KTA-290 requires power, a motor driver connected to one axis and a USB connection to a computer.

### Powering the controller:

Version 8 or earlier of the KTA-290 came in 12V and 24V versions, but version 9 onwards are powered from between 8 and 35 VDC. The version is marked on the PCB, but Version 9 cards are black, where earlier versions were green. All versions output 5 VDC step and direction pulses.

- KTA-290: 8 to 35 VDC (12 V Nominal) ~1.5W + External 5 V drain

Power is connected via the VS (positive) and COM (negative) terminals. When powered, an LED on the board glows.

### Communicating with the Controller:

The controller is connected to a computer via USB or RS-485. Connecting via USB requires a standard A to B cable (not included). Connecting via RS-485 would require a computer with an RS-485 serial port or an RS-485 converter.

Most computers will either have suitable drivers already loaded, or automatically load the correct driver. The card uses a common USB to serial converter IC from FTDI. If the card does not appear as a serial port when powered and plugged into a computer via USB, you will probably need to download the latest driver for your operating system. Virtual COM port (serial port) drivers for Windows, Mac OS and Linux are available from http://www.ftdichip.com/Drivers/VCP.htm

### Step and Direction Outputs:

The controller has four axes of step and direction outputs. The outputs of the KTA-290 provide 5 VDC signals. In most cases these can be wired directly to a motor driver's inputs. Most commonly, a driver will have differential inputs. These will usually be marked in "+" and "–". Figure 2 shows how to wire the controller to a driver with differential inputs.

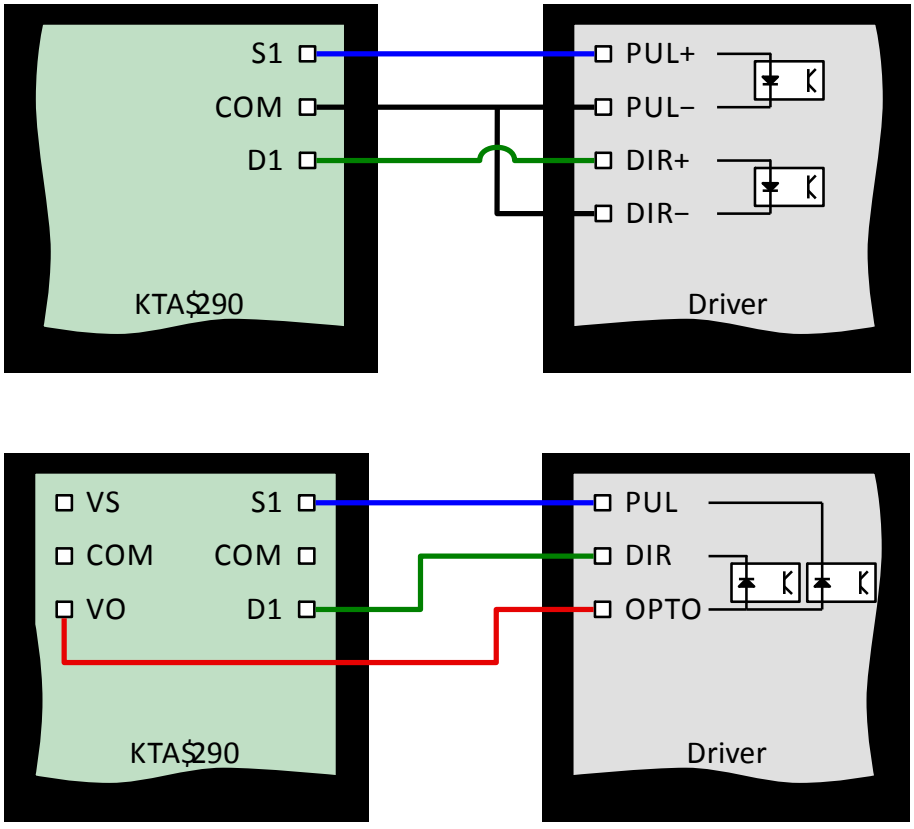


*Figure 3 - Wiring the KTA-290 to a single-ended input driver*

Figure 3 shows the wiring for a single-ended input driver. The VO terminal provides 5 VDC. Each driver wired in single-ended fashion needs its OPTO terminal connected to VO on the controller.

Multiple motors can be driven from a single axis output by wiring the step and direction signals in parallel.

### Relay Outputs:

The relays on the card can be used to switch equipment on and off. Each relay has a pair of screw terminals. When the relay is on, the terminals are shorted together. The relays can switch up to 5 A and 30 VDC. To switch large loads, or mains-voltage loads, a secondary relay is recommended.

### Limit Switch Inputs:

Each axis has a corresponding limit switch input. When the limit switch input is active, the axis is brought to an immediate halt and any move commands issued to the controller only move the axis one step.

The inputs suit dry contact or NPN style sensors. Multiple sensors can be connected to a single input by wiring them in parallel.

### Auxiliary Inputs and Outputs:

The controller has two inputs (labelled AN1 and AN2) and two input-outputs (labelled IO1 and IO2).

AN1 and AN2 can accept 0 to 32 VDC (with no damage up to 50 VDC) and are read using the RDIO and RDAN commands.

IO1 and IO2 can accept 0 to 2.048 VDC (with no damage up to 3.3 VDC), are read using the RDIO and RDAN commands, and can be driven using the WDIO command.

RDIO reads the inputs as digital inputs. When and input is exposed to a voltage higher than approximately 2.0 VDC, it is read as high.

RDIO reads the inputs as analog inputs. The analog voltage in millivolts is returned for all or one input.

WDIO drives the IO1 and IO2 terminals as outputs. They are switched between 0 and 3.3 VDC. The outputs can sink or source up to 10 mA.

## DIP Switches

The controller has a 4-way DIP switch. The switches set the card address and allow the recovery of a card with unknown communication settings. The switches are marked 1 to 4 and can be moved using a small screwdriver, pen or similar item.

Switches 1 and 2 control the card address. Multiple cards can be connected via RS-485. To control specific axes when multiple cards are connected, each axis needs a unique address. Axis addresses start at 1, 5, 9 or 12 depending on the position of switches 1 and 2.

| Switch 1 | Switch 2 | Axis Addresses |
|---|---|---|
| Off | Off | 1, 2, 3 and 4 |
| On | Off | 5, 6, 7 and 8 |
| Off | On | 9, 10, 11 and 12 |
| On | On | 13, 14, 15 and 16 |

*Table 1 - Base address selection*

Switch 3 has no function.

Switch 4, if on, forces the communication baud rate to 57600 bps and the checksum mode off when the controller is powered up. Changing the switch position while power is applied to the controller has no effect.

Using switch 4 to reset the baud rate does not change the baud rate or checksum setting saved in the controller's non-volatile memory. To set the baud rate of a card with unknown communication settings, the full procedure is:

1. Set switch 4 to the on position
2. Apply power to the card
3. Connect to the card with a serial terminal or the KTA-290 Test Utility
4. Issue the BAUD command to set the baud rate to the desired setting
5. If desired, issue the OPTN command to re-enable the checksum
6. Issue the SAVE command to write the new settings to non-volatile memory
7. Set switch 4 to the off position
8. Cycle (remove and reapply) the power to the card or issue the RSET command

## Using the Controller

The controller accepts commands via a virtual serial port on a computer or via RS-485. By default, the controller communicates at 57600 baud, using 8-bit bytes, no parity and one stop bit (also known as 57600, 8N1). The communications settings of a controller can be reset to 57600 8N1 by moving switch 4 to the on position and cycling power to the controller.

### Command format:

`@AA CMND[ X][ Y][ Z][ A]<EOL>[<CS>]`

- Square brackets ('[' and ']') indicate parts of the command that are optional. Commands should not be sent with square bracket characters in them.
- @ is the at symbol. All commands begin with the at symbol.
- AA is the axis address, 1 to 16. One or more space or tab characters must follow the address.
- CMND is the four character command. The command is not case sensitive.
- [ X][ Y][ Z][ A] is a series of up to four parameters. The number of parameters required depends on the command. Parameters must be decimal integer values and cannot contain thousands separators. One or more space or tab characters must be present between the command and the first parameter, and between each parameter.
- <EOL> is one or more a line-ending characters. Acceptable line-ending characters are carriage return ('\r' or 0x0D) or newline ('\n' or 0x0A).
- [<CS>] is the optionally required checksum. If the checksum mode is set (it is not set by default), the controller will only respond to commands if the line end is followed by a single byte checksum value. More detail is provided in the Checksum section below.
- The command must be less than 255 characters long, including end of line and checksum characters.

### Address:

Each command has an address. The address refers to the axis the command is directed at. Each controller has four axes, so will respond to four different addresses.

Some commands apply to parameters of the controller (like the BAUD command) or apply to all axes (like the STOP command). The controller will respond to these commands when the address is within the range of addresses the controller is set up (by the positions of switches 1 and 2) to respond to.

Some commands allow configuration of multiple axes in a single command. These commands take a variable number of parameters – one parameter for each axis. In these cases, the first parameter corresponds to the axis addressed by the command. Any proceeding parameters apply sequentially to the next axis. Examples are given with each command.

### Checksum:

If the checksum option is enabled, the controller will only respond to commands that end with a correct checksum byte.

The checksum is calculated as the exclusive-or of all the bytes in the command message. For example, the command "`@1 STOP\r`" would require a checksum of:

0x40('@')⊻0x30('1')⊻0x20(' ')⊻0x53('S')⊻0x54('T')⊻0x4F('O')⊻0x50 ('P')⊻0x0D('\r') = 0x45('E')

Where ⊻ is the exclusive-or operation. For the controller to accept the command with checksum mode on, the computer program or device sending the command would have to send "@1 ST0P\rE"

The checksum can be disabled by sending the OPTN command or by using DIP switch 4 to reset the communication parameters (see DIP switches.)

**Response:**

The controller will respond to most commands with "#AA\r\n" where AA is the address of the axis the command was directed at. AA is always two digits in responses – commands addressed to axes numbered less than 10 are padded with a leading 0.

Some commands return one or more values. In these cases, the response is of the form "#AA X Y Z A\r\n" where X through to A are the values requested. Values are always returned as decimal integers.

The controller may additionally send a response when an axis finishes its movement depending on the verbose mode and individual response mode settings. By default, the controller has verbose mode set and individual response mode not set.

With the default settings, the controller will respond to a movement command with "#AA\r\n" when the command is received, and then with with "!BB\r\n" when all axes have finished their movements. BB is the address of the axis that finishes last.

Switching verbose mode off (with the OPTN command) suppresses the "!BB\r\n" response. In this case, the controller will only respond by sending "#AA\r\n" when a command is received.

Switching the individual response mode on will cause the controller to send a response of the form "!BB\r\n" when each axis completes its motion. In this mode, a movement command can cause up to five responses from the controller: "#AA\r\n" when the command is first received, followed by one "!BB\r\n" style response for each axis.

## Commands

Table 2 outlines the commands the controller responds to.

| | |
|---|---|
| ACCF | Set the maximum frequency for an axis (Hz) |
| ACCI | Set the frequency increment (Hz/step) for an axis |
| ACCS | Set the initial frequency for an axis (Hz) |
| AMOV | Move to a position (steps) |
| BAUD | Set the communications baud rate (bps) |
| DROF | Immediately switch a direction output off |
| DRON | Set a direction output on indefinitely or for a period (tenths of seconds) |
| DRST | Report the current status of a direction output |
| OPTN | Set checksum mode and configure how the controller responds to axis move commands |
| POSN | Set the current position of an axis |
| PSTT | Report the current position of all axes |
| RACC | Report the initial frequency, frequency increment and maximum frequency of an axis |
| RDAN | Report analog voltage at AN1, AN2, IO1, IO2 or VS |
| RDIO | Report the status of AN1, AN2, IO1 or IO2 as a digital input |
| REL1 | Set relay 1 on or off |
| REL2 | Set relay 2 on or off |
| RMOV | Move forward or backwards some number of steps |
| RSET | Reset the controller and load settings from non-volatile memory |
| SAMV | Move an axis to a position with initial frequency, frequency increment and maximum frequency specified in the command |
| SAVE | Save baud rate, checksum mode, axis move response settings, axis frequency settings, current position and target position to non-volatile memory |
| SRMV | Move an axis forward or backward by a number of steps with initial frequency, frequency increment and maximum frequency specified in the command |
| STAT | Report the status of axis movement, direction output and limit switch inputs |
| STOP | Immediately stop all axes |
| WDIO | Set the digital outputs IO1 and IO2 on or off |

*Table 2 - Outline of Controller Commands*

### ACCF, ACCI and ACCS – Frequency and Ramping

An axis movement begins at the frequency set by ACCS, ramps up to the frequency set by ACCF at a rate set by ACCI, continues for the bulk of the movement, then ramps back down to the ACCS frequency before stopping.

During each ramp up step, the ACCI value is added to the axis frequency until the maximum frequency is reached. As the axis approaches the total number of desired steps, the ramp is repeated in reverse to slow the motor to a stop. Higher values of ACCI mean the motor reaches full speed sooner, but also demand more torque from your motor.

The ACCF command sets the maximum frequency in hertz for one or more axes. Valid values are 10 to 50000, defaulting to 1000.

The ACCI command sets the frequency increment and decrement, controlling the acceleration and deceleration of an axis. The increment is specified in hertz per step. Valid values are 1 to 9999, defaulting to 1.

The ACCS command sets the frequency to start and finish at in hertz. Valid values are 10 to 9999, defaulting to 10.

For example, to set the maximum speed of axis 3 to 2.5 kHz, use the command:

```
Send: @3 ACCF 2500\r\n
Receive: #03\r\n
```

The ACCF, ACCI and ACCS commands can also set values for multiple axes (up to 4) at once. The example below sets the maximum frequency for axis 2 to 1 kHz, axis 3 to 2.5 kHz and axis 4 to 6 kHz:

```
Send: @2 ACCF 1000 2500 6000\r\n
Receive: #02\r\n
```

If sent with no parameters, the controller will respond by reporting the current ACCF, ACCI or ACCS setting for the axis:

```
Send: @3 ACCF\r\n
Receive: #03 2500\r\n
```

### AMOV, RMOV Movement Commands:

The AMOV (absolute move) and RMOV (relative move) commands are the simplest way to get your motors turning. The controller internally keeps a record of the position of each axis counted in steps. At power up, these values are read from non-volatile memory.

AMOV drives an axis forwards or reverse to a specific position. The controller calculates the number of steps required to move from the current position to the specified position and performs the move.

RMOV drives an axis forward or reverse a specified distance. The controller keeps track of the position as the motor moves.

AMOV and RMOV can be used to command single axes. The example below tells axis 3 to move to position 10000. The default controller response settings result in two responses – one when the command is first received and one when the move is complete.

```
Send: @3 AMOV 10000\r\n
Receive: #03\r\n
Receive: !03\r\n
```

AMOV and RMOV can also be used to drive up to four axes at once. The example below tells axis 1 to move forward 100 steps, axis 2 to move forward 300 steps and axis 3 to move backward 200 steps. Assuming all three axes have the same frequency and acceleration settings, axis 2 will be the last to finish moving. With the default response settings, the controller responds once when the command is received and then again when axis 2 finishes.

```
Send: @1 RMOV 100 300 -200\r\n
Receive: #01\r\n
Receive: !02\r\n
```

### BAUD – Communications Baud Rate

The BAUD command sets the communications baud rate. Valid values are 10 to 230400 bps, and values 1 to 9 map to common baud rates shown below in Table 3.

| Value | Baud Rate |
|---|---|
| 1 | 2400 bps |
| 2 | 4800 bps |
| 3 | 9600 bps |
| 4 | 14400 bps |
| 5 | 19200 bps |
| 6 | 28800 bps |
| 7 | 38400 bps |
| 8 | 57600 bps |
| 9 | 115200 bps |

*Table 3 - Baud rate shortcuts*

If the BAUD command is issued without a parameter, the controller will respond with the current baud rate setting.

The BAUD command can be addressed to any of the axes of the card.

Due to limitations in system frequency and dividers, the controller cannot set its baud rate to every value in the range 10 to 230400 bps exactly. When the BAUD command is issued with a value, the controller chooses the closest baud rate it can attain. This is the value that is repeated back when the BAUD command is issued without a parameter. It is normal for the requested baud rate and closest attainable baud rate to differ by a few per cent and this difference will not affect the operation of the controller.

Changing the BAUD value does not immediately update the communications baud rate. The baud rate must be saved to non-volatile memory with the SAVE command and power to the board must be cycled or the RSET command issued.

Examples:

```
Send: @2 BAUD 5\r\n
Receive: #02\r\n
Send: @3 BAUD\r\n
Receive: #03 19202\r\n

Send: @1 BAUD 57600\r\n
Receive: #01\r\n
Send: @1 SAVE\r\n
Receive: #01\r\n
Send: @1 RSET\r\n
Receive: #01\r\n
```

### DRON, DROF and DRST – Direction Output Timers

When not being used to control motors, the direction outputs can be used as general-purpose digital outputs.

DRON sets an output on. If the parameter given is -1, the output stays on until switched off with a DROF command. The output can also be turned on for a fixed length of time. The DRON command, when issued with a parameter that is a positive value, causes the output to switch on for 100 ms times the value given. For example, to turn D4 on for 5 seconds, issue the command "`@4 DRON 50\r\n`".

The DROF command turns a direction output off, cancelling any timed operation for the output.

The DRST command returns the number of tenths of seconds left on the output timer for that axis.

Examples:

```
Send: @2 DRON -1\r\n
Receive: #02\r\n
Send: @2 DRST\r\n
Receive: #02 -1\r\n

Send: @3 DRON 100\r\n
Receive: #03\r\n
Send: @3 DRST\r\n
Receive: #03 98\r\n
Send: @3 DRST\r\n
Receive: #03 96\r\n
Send: @3 DROF\r\n
Receive: #03\r\n
Send: @3 DRST\r\n
Receive: #03 0\r\n
```

The DRON command can be used to set multiple axes at once. The example below sets D1 and D2 on for 10 seconds and sets D3 and D4 on for 20 seconds.

```
Send: @1 DRON 100 100 200 200\r\n
```

```
Receive: #01\r\n
```

The DROF and DRST commands can also be used on multiple axes at once. In this case the number of parameters determines how many axes are affected. The values of the parameters are ignored. For example, to cancel timers on D2, D3 and D4 in a single command, issue:

```
Send: @2 DROF 0 0 0\r\n
Receive: #02\r\n
```

Similarly, the DRST command can request the status of more than one axis. The example below indicates D2 is on indefinitely, D3 is counting down and will be on for 9 more seconds and D4 is off.

```
Send: @2 DRST 0 0 0\r\n
Receive: #02 -1 90 0\r\n
```

### OPTN – Checksum and Movement Response Options

The OPTN command sets or returns the status of three controller configuration parameters: checksum mode, verbose mode and individual response mode. These modes are explained in the Checksum and Response sections above.

The OPTN command can be addressed to any axis of the controller. If the command is addressed without a parameter, the controller will respond with the current options value. When used to set the options, the command takes a single parameter. The values correspond to the modes shown in the table below.

| Value | Verbose Mode | Checksum Mode | Individual Response |
|---|---|---|---|
| 0 | Off | Off | Off |
| 1 | On | Off | Off |
| 2 | Off | On | Off |
| 3 | On | On | Off |
| 4 | Off | Off | On |
| 5 | On | Off | On |
| 6 | Off | On | On |
| 7 | On | On | On |

*Table 4 - OPTN Values*

By default, the controller has an OPTN value of 1 (verbose mode on, no checksum, no individual response.) Examples:

```
Send: @1 OPTN 1\r\n
Receive: #01\r\n

Send: @1 OPTN 5\r\n
Receive: #01\r\n
Send: @3 OPTN\r\n
Receive: #03 5\r\n
```

Options take affect as soon as the command is received. Options are not saved to non-volatile memory until a SAVE command is issued. If the controller is reset before options are saved, the checksum mode, verbose mode and individual response mode settings will revert to those last saved in non-volatile memory.

### POSN – Set Current Position

The POSN command sets the current position for one or more axes. The command can only be used to set an axis position when the axis is idle (not moving). Valid position values are signed 32 bit integers. This allows a range of over ±2 billion.

When issued with no parameters, the POSN command request the current position of the address axis.

Examples:

```
Send: @1 POSN 0 100 200 300\r\n
Receive: #01\r\n
Send: @3 POSN\r\n
Receive: #03 200\r\n
```

### PSTT – Report Axis Positions

The PSTT command takes no parameters. It can be addressed to any axis of the card. When issued, the controller responds with the positions of the four axes controlled by the card.

Examples:

```
Send: @1 POSN 0 100 200 300\r\n
Receive: #01\r\n
Send: @3 PSTT\r\n
Receive: #03 0 100 200 300\r\n
```

### RACC – Report Axis Frequencies

The RACC command takes no parameters. When issued, the controller responds with the initial frequency, frequency increment and maximum frequency for the addressed axis.

Examples:

```
Send: @2 ACCS 10\r\n
Receive: #02\r\n
Send: @2 ACCI 1\r\n
Receive: #02\r\n
Send: @2 ACCF 3000\r\n
Receive: #02\r\n
Send: @2 RACC\r\n
Receive: #02 10 1 3000\r\n
```

### RDAN – Read Analog Inputs

The RDAN command reads the voltage at AN1, AN2, IO1, IO2 or the supply voltage to the controller.

With no parameter, the command returns all 5 voltages, in millivolts.

Examples:

```
Send: @1 RDAN\r\n
Receive: #01 0 12000 500 250 23500\r\n
```

In this example, the voltage at AN1 is 0 mV, at AN2 is 12.0 V, at IO1 is 500 mV, at IO2 is 250 mV and the board is being supplied by 23.5 VDC.

Please note that the supply voltage is measured after a protection diode, and so will be approximately 0.7 VDC lower than the voltage applied at the VS terminal of the board.

With one parameter, the command returns the voltage in millivolts and one of AN1, AN2, IO1, IO2 or the controller supply voltage.

| Command | Function |
|---|---|
| @01 RDAN 0 | Return voltage in mV at AN1 |
| @01 RDAN 1 | Return voltage in mV at AN2 |
| @01 RDAN 2 | Return voltage in mV at IO1 |
| @01 RDAN 3 | Return voltage in mV at IO1 |
| @01 RDAN 4 | Return controller supply voltage in mV |

*Table 5 - RDAN single parameter functions*

### RDIO – Read Digital Inputs

The RDIO command takes one parameter or no parameters. With no parameter, it returns a value representing the state of IO1, IO2, AN1 and AN2 digital inputs. The table below decodes the meaning of the values returned.

| Value | IO1 | IO2 | AN1 | AN2 |
|---|---|---|---|---|
| 0 | Off | Off | Off | Off |
| 1 | On | Off | Off | Off |
| 2 | Off | On | Off | Off |
| 3 | On | On | Off | Off |
| 4 | Off | Off | On | Off |
| 5 | On | Off | On | Off |
| 6 | Off | On | On | Off |
| 7 | On | On | On | Off |
| 8 | Off | Off | Off | On |
| 9 | On | Off | Off | On |
| 10 | Off | On | Off | On |
| 11 | On | On | Off | On |
| 12 | Off | Off | On | On |
| 13 | On | Off | On | On |
| 14 | Off | On | On | On |
| 15 | On | On | On | On |

*Table 6 - RDIO values*

With one parameter, the RDIO command returns the state of one digital input. The value returned is 0 if the input is less than 2.0 VDC, and 1 if the input is above 2.0 VDC.

| Command | Function |
|---|---|
| @01 RDIO 0 | Return status of IO1 |
| @01 RDIO 1 | Return status of IO2 |
| @01 RDIO 2 | Return status of AN1 |
| @01 RDIO 3 | Return status of AN2 |

*Table 7 - RDIO single parameter functions*

### REL1 and REL2 – Set Relays

The REL1 and REL2 commands set or report the status of the two relays on the controller. They can be addressed to any axis of a controller. When issued with no parameters, the controller will respond with 0 to indicate the relay is off, or 1 to indicate the relay is on.

The commands can take a single parameter. If the parameter is 0, the relay is switched off. Any other parameter will switch the relay on.

Examples:

```
Send: @1 REL2 1\r\n
Receive: #01\r\n
Send: @4 REL2\r\n
Receive: #01 1\r\n
```

### RMOV – Relative Move

See AMOV above.

### RSET and SAVE – Reset Controller and SAVE Parameters

The SAVE command records the controller's configuration settings to non-volatile memory so they aren't lost when power is removed. The RSET command causes the controller to reset, simulating a removal and reapplication of power. The commands can be addressed to any axis and take no parameters.

On reset or power up the controller reads settings from non-volatile memory for:

- Axis initial frequency, maximum frequency and frequency increment
- Axis current position
- Checksum mode
- Verbose mode
- Individual response mode
- Baud rate

The controller then applies the baud rate to the serial port and sends out a power-up message.

Cycling the power causes the USB circuitry to reset, which generally causes terminal software to close the serial connection. Using the RSET command doesn't cause the serial port to disconnect, allowing the user to see the controller power-up message. The power-up message includes the firmware version and the current card address.

### SAMV and SRMV – Single-line Absolute and Relative Move

The SAMV and SRMV commands issue move commands to a single axis, specifying target position or distance and frequency settings. The limits for each parameter match the limits specified in the AMOV, RMOV, ACCS, ACCF and ACCI sections of this document.

The order of parameters is:

- Target position (SAMV) or distance (SRMV)
- Initial Frequency
- Maximum Frequency
- Frequency Increment

The response issued by the controller will depend on the verbose and individual response settings. See the Response section for more detail.

Example:

```
Send: @12 SAMV -20000 10 5000 1\r\n
Receive: #12\r\n
Receive: !12\r\n
```

### SAVE – Save Parameters to Non-Volatile Memory

See RSET and SAVE.

### STAT – Axis Movement, Direction and Limit Switch Status

The STAT command returns a single value that indicates the status of each axis (moving or not), the direction (forwards or reverse) and the status of each limit switch input. It can be addressed to any axis of the controller with the same result. It takes no parameters.

The value returned is calculated as the sum of powers of two, each term dependent on a particular status. As a result, when the value returned is expressed as a binary number, each bit corresponds to one of the statuses being reported.

The bit positions are:

- Bit 0: Axis 1 moving (1: moving, 0: idle)
- Bit 1: Axis 2 moving

- Bit 2: Axis 3 moving
- Bit 3: Axis 4 moving
- Bit 4: Axis 1 direction output status (1: forward, 0: reverse)
- Bit 5: Axis 2 direction output status
- Bit 6: Axis 3 direction output status
- Bit 7: Axis 4 direction output status
- Bit 8: Limit switch 1 status (1: active, 0: open circuit)
- Bit 9: Limit switch 2 status
- Bit 10: Limit switch 3 status
- Bit 11: Limit switch 4 status

For example, if the controller returns the value 2374 the status can be determined by first converting the value to binary (2374 = 0b 1001 0100 0110) then reading off each bit. In this case bits 11, 8, 6, 2 and 1 are set. This indicates axes 2 and 3 are moving, axis 2 is moving forward and axis 3 is moving backward and limit switches 1 and 4 are active.

### STOP – Halt all motion

The STOP command stops all axes immediately, without the gradual deceleration defined by the axis frequency settings. It can be addressed to any axis and will stop all axes. It takes no parameters.

The controller responds to the STOP command with "#AA\r\n" where AA is the axis addressed. Depending on the verbose and individual response modes, the controller may also respond with "!BB\r\n" for axes that were in motion.

### WDIO – Set Digital Outputs

The WDIO command takes a single parameter and sets the IO1 and IO2 digital outputs. The command can be addressed to any axis of the controller. The table below shows the effect of each value.

| Value | IO1 | IO2 |
|---|---|---|
| 0 | Off | Off |
| 1 | On | Off |
| 2 | Off | On |
| 3 | On | On |

*Table 8 - WDIO values*