Introduction and Quick Setup

Excitron's *Faster Than Fast®* stepper motor **Controllers** make your motion control simple. We pack years of design and plenty of power into very small cases, and fully integrate it with the motor. They contain all the electronics and power for thousands of motorized applications—and they run right out of the box by simply typing "**w**" then a "**G**".

Stepper motors rotate one small step at a time, and are special because both position and speed can be precisely controlled, unlike any other type of motor. Our motors can run continuously for 30 years.

Our part number system indicates the amperage rating (1A, 4A, 10A, or 100A) and the serial communication port. **Controllers** may have standard USB, RS232 (+-10v), or TTL (+5v) for serial port communication to any computer. Other serial port adaptors are available. Various size, power, and control functions give you the best selection for your project:

Controller	Amp max	Serial Port	Main Connector	I/O pins
1A-TTL	1	TTL	DB15 female	6
4A-RS232	4	RS232	10 pin header	3
4A-USB	4	USB	8 pin header	3
4A-TTL	4	TTL	DB15 female	6
10A-RS232	10	RS232	10 pin header	3
10A-USB	10	USB	8 pin header	3
10A-TTL-3SW	10	TTL	DB15 female	6
100A-TTL	100	TTL	DB15 female	6
8E-RS232	8	RS232	1x12 header	7
8E-USB	8	USB	1x8 header	6


1A-TTL-SGM25-25-30

10A-RS232

10A-USB


10A-TTL-3SW-42-47M motor


PS-24-80W Power Supply & USB

Our experts are happy to assist you in selecting the best controller, stepper motor, and accessories for your project. We also offer linear slides, actuators, rotary tables, X-Y tables, XYZ machines, and heavy duty 3-axis milling machines. Visit our website for all documents, prices, and online purchasing.

For a quick start:

- Make sure the AC-DC power supply is off or unplugged.
- Connect the stepper motor connector to the **Controller** (if your controller is integrated, there are no motor wires).
- Connect the power supply/IO connector to the **Controller**.
- If you have a USB **Controller**, connect your USB cable between your computer and the **Controller**. PC Software drivers may need to be installed. Otherwise, your serial port is already connected via the main **Controller** connector.
- Turn on your power supply. The LED should be ON.
- Start Hyperterminal (or any equivalent program) with 57.K baud, 8 bits, and no hardware handshaking.
- Type a little **w** to wake the **Controller** and it displays:

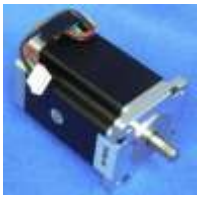
```
Excitron Corp  10/01/09  v3.45  10A-RS232
M>
        (Controller name varies)
```

- Type **G** to run the stepper motor.
- Type little **i** to see all **Controller** information and the motion profile #01 values.
- Change direction (**C** or **W**), **V**sps, or number of steps **N**, type **G**, and see the difference.
- Type **?** for a brief command help list.
- The **Controller** is in Command Mode, and will respond to any serial command. Input Profile, Motion Profile, and Driver Modes create standalone functionality.

Enjoy! Feel free to contact us at info@excitron.com .


Mini Z-Lift


8" Rotary Table MRT8-86-118


SM60-86

Reduction of Hazardous Substances (RoHS) Statement of Compliance

Excitron is a product manufacturer and does not melt or produce any of the raw materials sold to our customers. Excitron only purchases ROHS parts, does not add or expose these materials to any of the hazardous substances (lead, mercury, cadmium, hexavalent chromium, polybrominated biphenyls, polybrominated diphenyl ether) listed within the EU RoHS Directive, except we use tin-lead solder. As cited in statements from our vendors, the materials sold by Excitron do not exceed allowed levels of these substances because they are exempt, and therefore, the materials are compliant with the EU RoHS directive.

Key Features

This document applies to all Excitron **Controllers**. See the individual motor **Controller** data sheets for additional details about connector pins.

The **Controller** performs all calculations and intelligence to automatically step the motor. It also contains powerful motor drivers. No additional electronic device is needed to control and drive a stepper motor.

You have the freedom to enter ***simple*** motion parameters to create ***ingenious*** stepper motor motion profiles. Serial communications and control use ASCII (alphanumeric) characters, thus no programming is required; all you need is a basic serial interface program like Hyperterminal.

Your biggest goal is stepping your motor from point A to point B quickly and accurately. An Excitron **Controller** quickly allows you to attain this goal with truly simple commands for friendly use, yet versatile and powerful enough to control a sophisticated robotic automatic assembly machine.

You can quickly change motion parameters to achieve your goal. You may opt for higher torque and faster movement after finding that the temperature rise is satisfactory. Or you may desire to reduce mechanical noise by varying the acceleration, SPS, or torque. Most step motor documents state that bipolar motors produce more torque than unipolar motors--that is not true with our motors and **Controllers**. Our proprietary circuitry was first developed and patented in **1976.** Improvements continue steadily to maximize motor and **Controller** efficiency, which maximizes torque.

Three modes of operation are available: Command, Motion Profile, Driver, and **I** Input Profile. Encoders may be added.

A full set of parameters for stepping the motor is called a *motion profile*. Twenty-four motion profiles are stored and retained, even after power is removed. A motion profile consists of acceleration, stepping at a constant maximum SPS (steps per second), then deceleration to stop precisely at the number of steps you specified. Deceleration is identical to acceleration. A time delay before or after motion occurs gives you precise timing control.

Some key parameters are acceleration, maximum SPS (steps per second), number of steps, direction, and torque control. The **Controller** uses unique current limiting circuitry for high efficiency and performance. By adjusting the acceleration, top SPS, and other factors, you can minimize the time required for a particular motion. A full range of acceleration is provided for any application: dog slow to extremely fast. SPS can be changed on the fly while the motor is stepping.

Various input devices, such as limit switches and potentiometers, may be connected to the TTL/CMOS +5v logic input pins. One input pin (if included)accepts analog 0-5 V and converts it to a digital value. Most **Controllers** have 1 TTL/CMOS +5v output that you may use to control relays, LEDs, and equipment (proper interfacing is required). A single LED provides visual feedback that power is applied, and it also indicates motor steps or serial communications.

One key unique feature is the extremely low supply power for the **Controller** electronics. With a 24-v power supply, the **Controller** only dissipates 0.25 watt, which is insignificant. All internal voltages are self-generated and thus require only a single external voltage supply. Built-in temperature sensing on **Controllers**, except 1A-TTL, safeguards the **Controller** from thermal burn out, the limit is 170 degF. When mounted on or near the motor, this temperature sensor also protects the motor.

Many aspects and limitations in competitor's controllers are obsolete. Excitron **Controllers** drives any size unipolar stepper motor, even our 110-201—a 26-pound Super-size 42! The power supply voltage range is 10 to 30V, a voltage range of 36-48 is optional, except the newer Controllercoders must run at 24 volts only.

Many options are available. All **Controllers** are upgradeable to new features in the future via firmware updates, thus protecting your investment. Custom firmware applications are also available; just ask the friendly Excitron technical staff.

What's New

Exciting speed control on-the-fly by using the analog input pin.

Motor torque is increased 10% if half-stepping by powering the half-step at 1.5 times normal. Half-stepping is now superior to full-stepping.

New Master Controller mode, where one Excitron Controller sends serial commands to other Excitron Controllers via our Simple Serial Bus. This saves you time, space, money, and complexity by eliminating external PCs, PLCs, and other expensive controllers.

Command "H" allows automatically finding home position at power up. Power-up now delays 1.4 seconds.

Analog signals can trigger 3 Motion Profiles.

Serial Command Mode sends slave address after each command and before the '>', so you can easily see the slave address. Switch or Input Profile activations send the profile number and '>'. Optional 2nd UART for controlling a slave device.

Built-in **Encoder** and/or **Index** commands and interface for absolute position control. View each motor step and encoder pulse via serial port using the little **c>v** c

Total number of motor steps is accumulated and auto-saved (DB15 only) every 10 minutes, useful for maintenance.

Greater versatility and intelligent motion with **Input Profile Mode**. Any input pin can individually select 3 motion profiles to run. This "random" motion selection is available in Command, Motion Profile, and Driver modes.

With the **f**, **F**, and **m** commands, you create a plateau of stepping at a minimum sps at the beginning, end, or both, of your motion.

Mode 17 sends a 2-5 microsecond pulse on the output pin each motor step. Useful for measuring and syncing other Controllers with exact same motion profile.

Warning! Use caution when operating, severe injury can result from the motor rotating. Long wires act like antennas and may cause erratic dangerous motion.

Setup, Configuration, and Power

Excitron motor **Controllers** can operate standalone, or in various modes. The onboard UART serial port is TTL (+5volt logic), RS232, or USB. If your controller is TTL, then do not use RS232 directly--use our RS232-TTL cable adaptor, which contains the electronics to convert RS232 (+-10 volts) to TTL. Damage will result if you plug in any +-10v RS232 directly to any TTL device. The 4A-RS232 and 10A-RS232 have built-in RS232 transmit and receive, thus no other interface device is necessary, just a plain DB9 female connector.

If you order the **Controller** and motor with an RS232-TTL adaptor, it is pre-soldered for you. You will also need a DC switching regulated power supply, from +10V to +30v, with sufficient amperage to drive your step motor. These and other accessories are available on our web site. See your Controller's drawing for interface pins.

A high amperage 12" 18 AWG power cable with a power supply filter is now included with every **Controller**. The power supply wires should be at least 20 AWG stranded, 18 AWG is best, and as short as possible. The amount of current into the motor is NOT the power supply current. A rough guideline is the power supply watts out to the motor equals the motor watts (watts = volts x amperes). You will need to budget at least twice the motor's watts. Power (in watts) is also defined as amperes x amperes x resistance.

Excitron's **Controllers** can operate at 2 to 3 times the rated value for intermittent operation. If the power supply amperage is inadequate, the **Controller** may pull the power supply voltage low enough to cause a reset of the **Controller.** This is harmless to the **Controller**, but stops your motion. If you operate in Motion Profile mode (described later), then the **Controller** will begin anew at the first motion profile selected by the **U** command.

A computer running Hyperterminal or an equivalent serial port communications program is required. The serial port operates with 8 bits, no parity, 1 stop bit, and 57,600 baud (initial). Protocol is none, no hardware handshaking, and just ASCII alphanumerics. The **Controller** will echo all characters received by default, therefore turn off Echo in your communication program. Once setup in standalone mode, the RS232-TTL adaptor is no longer needed, if you like. Each Controller has an assignable address, the default and simplest address is 'M'.

Sending Commands to the Controller

The format of **Controller** commands is simply send the single letter of the command. Important--capital and small letters are different. If user input is expected after the command is typed, the **Controller** responds with '=', the existing value, a colon and a space. This existing value also shows the exact quantity of numbers to send. Then it waits until you send the required quantity of numbers. After all input is received, the **Controller** will send a LF (line feed), CR (carriage return), a variable time delay of 0-9 milliseconds, slave address, and a prompt '>'. The '>' provides feedback to host computers that the command is complete.

Do not type "Enter" key after typing your command, type only the command and any characters as specified by the each command. Note that the ASCII numbers are fixed field format. This means the exact required quantity of alphanumerics must be typed. For example, to enter a step quantity of 188 steps, type:

N (**Controller** sends "=0033000: ") then **0000188**

All 7 digits must be entered. If you enter fewer digits, the command is left waiting for completion.

All input values are validated to be within the range specified for each command. Any input value outside this range results in a '?' being displayed and the default or existing value being used. If you send extraneous alphanumeric characters, a '?' is displayed and no values change. The exception to this rule is that commas, spaces, carriage returns (Enter or CR), and line feeds (LF) are ignored. Serial input characters are *not* buffered.

After waking up with a little **w**, The **Controller** is ready to accept serial commands. The command prompt is 'M**>**', or 'A', for example, if you changed the Controller slave address to A. The information command is **i**, which displays all **Controller** parameters and the selected motion profile. Commands may be typed in any sequence and the **Controller** remembers your last typed values. See the Motion Profile section for how to save your values permanently with the **U** command.

Changing Controller Configuration

The little **c** command accesses commands for changing the controller operation and features. If you change the value, it is saved automatically and immediately:

A=0: Vsps is sent if=1.
b=57.6K: select baud **9**.6K, **1**9.2K, **3**8.4K, **5**7.6K (default), 115,2**0**0, **2**30.4K. Type just *one* digit of the baud (9, 1, 3, 5, 0, or 2 respectively), and immediately the baud is changed.
c change milliseconds time delay after each CR (return).
d driver mode, see Driver Mode section for details.
E=+-0800: Encoder pulses/revolution; +0000 = disable
e=1: echo received serial characters if = 1 or 3. Echo other slave characters to PC if 2 or 3.
h=0: home enable: 0 = disable, 1 = enable
I=0: Index enable: 0 = disable, 1 = enable
M=0400: Motor full-steps/revolution
P=+-00000836: Encoder position
R=+-00000078: Revolutions of Index
r=00004567: change total steps/131K value
s=**M**: slave address **A-Z or a-z**, using Simple Serial Bus
u update total **Steps** to eeprom memory.
v=1: view motor steps, Index, and/or Encoder pulses while the motor is stepping: 0 = off, 1 = on. If on, each motor step CW is an **m**, CCW is an **M**; Index is **i** (CW) or **I** (CCW); Encoder is **e** (CW) or **E** (CCW). Use for external monitoring of position and speed real-time. 57.6k Baud limits top speed to 5,500 sps; 115k baud limits to 10K; 230k baud limits to 20K. Character overrun occurs at higher sps.
q quit and return to main command menu.

Simple Serial Bus connects multiple **Controllers** with one host/PC serial port, or no external computer at all. The simple protocol is: **STX-address byte-data bytes-ETX**. STX is ctrl-B and ETX is ctrl-C. A normal **Controller** address is **M** (Master/default); and slaves are **A-Z**. Addresses **a-z** are special masters, which can control many slaves with or without an external PC, email us for information. A slave responds to commands only if a matching address and protocol is received. You wire the host's transmit to the Receive of one **Controller**, it's Transmit to the next **Controller** Receive, and so on, and the last **Controller** transmit is wired to the receive of the host. See the **Controller** Connections sheet.

Driver Mode serially responds only to @, **J**, Switches, and Input Profile activations, and normally just echoes other characters.

Setting Motion Parameters

All motor parameters required for sophisticated motion control of your motor are described here in alphabetical order. Note that you type only the characters shown in **bold**. The **Controller** displays the equals sign, the existing value (from 1 to 7 numbers), colon and a space (most commands).

A=0100: **nnnn** Acceleration (1st Vsps), 0002 is very slow and 1000 is extremely fast. If **A** > Vsps, then **A** is set to Vsps.

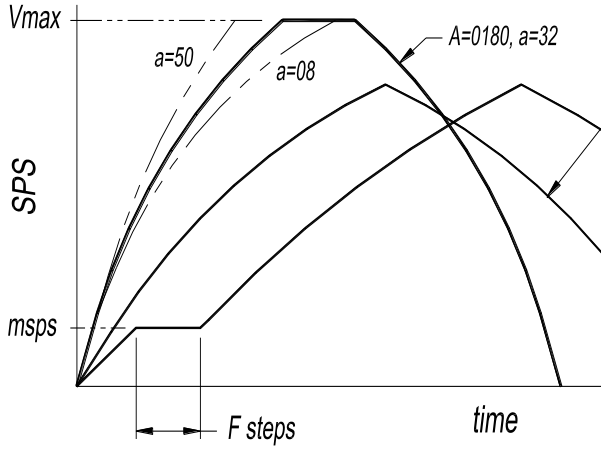
a=18: **nn** 2nd acceleration value to fine tune after 1,000 SPS, 00 is slower and 63 is faster acceleration.

C Clockwise motor direction, as viewed from the motor face.

D=1: **n** Down/up sps on-the-fly using analog pin: 0=no, 1=enable, SPS_dir input pin sets direction—low is CW, high is CCW.

M=0: **n** Micro stepping, full step is 0 and half step is 1.

N=0012000: **nnnnnnn** Number of steps. Use N=0000000 for motion control using inputs or time delay, without any stepping.



r=006: **nn** **r**epeat quantity, from 001 to 255, if > 100, then **r** = 10 x **r**. The remaining repeat value is output *after* each motion, except the last.

t=+-00888: **nnnnn** **t**ime delay either before (**-**) or after (**+**) the motor motion. Input - or +, then 5 numbers representing milliseconds, e.g., **t=+00500** defines a 500 millisecond time delay after the motor motion.

V=04600: **nnnnn** Velocity--maximum SPS. Note that the step quantity and acceleration values may define a motion such that the maximum SPS will not be achieved. Lower **A** values self-limit the maximum SPS.

W Counter-clockwise motor direction.

f,F,m 3 input values that set a flat speed portion at the beginning, rear, or both of the speed profile.

The following graph shows examples and variations of acceleration **A** and little **a**, and using **f**, **F**, and **m** (note that **A** must be less then **V** and **m**). Deceleration is a mirror image of acceleration.

Setting Motor Stepping Torque

Motor torque is directly proportional to motor current, with some non-linearity. Torque is defined as Force x Distance, and is usually measured in oz-in (ounce-inches). The **Controller** uses unique current limiting circuitry for high efficiency and performance for the user. Step motor current (amps) must be limited, or excessive heat and amperage can destroy motors and **Controllers**. Most step motors are rated for 2 to 5 volts; so current limiting allows them to be connected to a 10 to 30 volt (24 volt being most popular) power supply. You control precisely the amount of torque in the motor with one simple command:

T=004: **nnn** sets the internal value for motor torque.

A low value for **T,** such as 001-002, produces low torque. Higher values increase torque. Start at low values for **T** and increase the value until your motion is satisfactory. ***Do not*** increase **T** without adjusting **A**, **a**, and **V** first. And ***do not*** increase **T** to make the motor run faster. At higher speeds the T is not effective, yet it is effective as the motor spins up and during deceleration. See the individual motor data sheets for further guidance. After determining the suitable values, adjust **T** up or down for best performance vs. heat build up. The maximum allowable T is set for each motor size, and is displayed with the ***i*** command. The **Tmax** value will produce 2X the rated amperage at 24 volt, and 1X the rated amperage at 12 volt.

After a certain SPS, the **Controller** determines that current limiting is no longer needed, and the motor is “fully on”. Torque is then limited by the motor’s reluctance. The power supply voltage is the largest factor for torque. Motor current is rated based on the amount of current that causes the motor to reach 85 degC in 10 minutes. If mounted to a relatively large metal plate to dissipate the heat, this current rating will increase significantly. For intermittent operation, which is most stepper motor applications, the motor may be over-driven up to 3 times its rated power, depending on many factors.

Warning—stepper motors require amps of current, and overheating and damage can quickly occur. Start with low **T** values, then gradually increase while monitoring temperature via the ***i*** display. Maximum operating temperature is 75 degrees C (centigrade), and is monitored while the motor is running (not on small **Controllers**). Selecting higher torque than allowed may damage the **Controller**. You are responsible for any damage to the **Controller** or motor.

If the temperature limit is reached, the motor ceases running. Then the LED blinks once per second, and a “s=x Hi temp” message is sent from the serial port, where the “s=x” part of the message is the address of the unit sending this message. No operations occur until an @ reset character is received. Note

that this terminates Motion Profile or Driver Modes, and the **Controller** is then in Command Mode.

Stepping the Motor

You step the motor with the **G**, **g**, **S** and **s** commands:

G Go step the motor until the specified number of steps are reached.

g Reciprocating motion--same as **G**, and at its completion, the motor reverses direction and performs exactly the same motion profile.

S Step the motor continuously. This command ignores the step number entered. Only a motor stop command **O** or **o** will stop the motor.

s Sinusoidal motion (cyclical)--step continuously only if **f** is 2 or 3. This command ignores the step number entered. Only a motor stop command **O** or **o** will stop the motor.

After you type **G**, **g**, **S** or **s,** you can send **U, D, O, or o:**

U Speed up the motor on-the-fly. The SPS increment is determined by the **V** value.

D Slow down the motor on-the-fly. The SPS decrement is determined by the **V** value.

O Starts deceleration gracefully and the motor slows down according to the *acceleration* factor you specified.

o Motor stepping is stopped immediately. The motor's inertia may cause erratic motion for a few steps.

The motor will stop if sufficient **D**'s occur. You may also change Vsps-on-the-fly using external analog input pin, which is enabled with the **D** command:

D=0: 1 0 disables, 1 enables Vsps-on-the-fly.

This gives wonderful speed control while running, and the adjacent input pin (DB15 IN5, for example) sets the direction.

Your **A** acceleration may be temporarily changed when actual Vsps drops below the original **A** value. Since your N number of steps remains unchanged, best to vary speed slightly so that deceleration occurs properly. Suggest setting **A**=0010-0024 and the 2nd acceleration value **a**=01, these affect the **DU** incremental speed change. Changing Vsps on-the-fly may cause lost steps when stopping because there are not enough steps are left to decelerate properly. The original SPS value remains unchanged.

Braking

When the motor is not stepping, motor current (amps) may be applied to create a braking (holding) torque. WARNING--high temperatures may occur rapidly with high brake values over 110. Values from 1 to 90 produce little braking, and values between 110 and 124 are typical. Always increase braking values slowly and check heat. If you do not need braking torque, specify zero braking current by typing:

B=000: **nnn** Braking (holding) current limiting parameter. Range of values is 000 to 138.

Power saved by minimizing braking results in greater torque available for motor stepping. Higher current results in higher temperatures. **B**=000 also reduces the **Controller** power consumption. Any reset makes **B**=000 for Profile #01 only during Command mode, the saved B value is unchanged for all other modes. **P 01** or **v** restores your **B** value, be careful.

Home Command

You can enable the Controller to auto-home at startup by setting **h** (under the little **c** menu) to 1:

h=0: **1** 0 = disable, 1 = enable

When enabled, Motion Profiles 04 and 05 are run to completion. You have complete control of the values in these 2 profiles, including which direction to rotate and your input sensor pin. Suggest setting Profile 05 for backing up a small amount of steps at a lower speed, to increase homing accuracy. The value of **h** is displayed with the **i** command, on the Encoder line.

You can home at any time by sending **H**, which runs Motion Profiles 04 and 05. Note that auto-home also runs if Motion Profile Mode is enabled.

Command Examples and Notes

Suppose you wish to rotate the motor for 10,244 steps clockwise, half stepping, and a maximum SPS value of 1,800. Send:

```
N=0000080: 0010244
C
M=0: 1
V=04200: 01800
G
```

Note that you do not type '>', '=' nor "enter" with any command, the **Controller** provides it. The '**>**' prompt is not shown here for clarity .The order of the **N, C, M,** and **V** commands is not important, but the **G** command is, since it starts motor motion. The **M, N** and **V** commands must receive their required exact digit quantity. To step the motor again, with the same parameters except CCW (counter clockwise), type:

```
W
G
```

Note that the other commands do not have to be re-entered. The **Controller** temporarily stores the last values of all parameters and commands. Save permanently by using the U command, see Motion Profiles section.

Be aware that the motor and **Controller** will heat up faster while running at slower speeds. At SPS over 3,000, depending on power supply voltage, the motor's indunctance, the motor self-limits the power, and should run cooler. A temperatures over 50 degree C may feel hot to human touch, but that is ok for motor and **Controller** operation (up to the 75 degree C limit).

Acceleration, maximum SPS, torque, and other parameters can be optimized for your application. An acceleration value of 0120 is a good starting point for half-stepping. The **Controller** automatically decreases acceleration as speed increases. This optimally matches the stepper motor's output torque curve, better than a linear speed curve.

Many interesting combinations exist to create the motion you need for sophisticated control of automatic assembly machines or for motorized products. The time command **t,** can be coupled with various input mode commands to produce intelligence and control for almost any application.

All stepper motors have a natural resonant frequency around 200 to 1,200 SPS. If stepped at constant SPS in this speed range, the motor may vibrate and lose steps. This condition is worse at higher torque settings. It is best to accelerate through this range. Lower **T** values help reduce resonance and mis-stepping. Usually running half-step (**M**=1) helps. When changing from half to full stepping, the **Controller** may shift a half step to always be on a full step.

Power-ON Reset

The **Controller** has sophisticated power-on reset and brownout voltage protection circuitry. This protects from power supply glitches, and increases the **Controller's** robustness and integrity. Starting with v3.40, the Controller waits 1.5 seconds for your power supply to stabilize before allowing any motor motion. The RESET pin is normally high. To reset the **Controller**, send a logic low-level control to this pin, and release. After any reset, Brake is 000 for Profile #01 in Command Mode only--the saved ee value is unchanged. This protects against unplanned heat build-up. The first motor step after power-up or reset is always the same step, and 1-3 steps may be needed to sync the **Controller** with the motor's last rotor position. The high value filter capacitors in the **Controller** provide considerable power supply voltage spike attenuation, and a 1-4 second hold-up time. Always wait briefly before unplugging.

Software Reset and @ Commands

Software reset by sending @, which restarts the **Controller** as if a power-on reset occurred, except the response is only 0.01 seconds. Use this command to terminate Motion Profile and Driver Modes, and run serial commands. A little **w** must be sent to wake up the **Controller**. Other special @ commands are:

@@**{** & **STX** (cntrl-B) & an address letter--auto-sets multiple slave's addresses.

@@**{** followed by any character besides STX--send slave address.

@@@ (3 @'s in a row)--clears Input Profile mode **I** to 00.

Input/Output Pins and Modes

You can create very interesting and versatile control of stepper motions by using the input pins and their modes. Some control features are: stop and go control, logic only; wait for two inputs to occur; sequence external events, extended time delays; or other ingenious reasons. All pins are TTL/CMOS compatible (0 to +5v), and have Schmidt triggers for noise immunity. You can read your **Controller's** pin values with the **J** command:

DB15	In3	In5	In7	In8	In9	Out6	Sw2
	1018	1	1	1	1	0	1015
4A-RS232, 10A-RS232 and 10A-USB						In3	In4	In5
						1	1	1
1I-TTL and 3I-TTL	In1	In1	Out3	In4	In5	In6	
	1	1	0	1	1016	1015	
3E/8E-RS232 and 3E/8E-USB	In1	In1	In3	Out4	In5	In6	In8
	1	1	1	0	1	1016	1015

The In# matches the connector pin number. Each input pin is normally high (+5v) and can be connected to switches (optical, mechanical, etc.) or to monitor signals from other electronic devices. The correct connection is for one terminal of the switch/control connected to ground, and the other terminal connected to the input pin. Input pins have a 35K (+- 5%) pull-up resistor to +5 volts, except analog pins may have a pull-up resistor of 3.9K (+- 0.5%) on some larger Controllers. Output pins are normally low (0 volts) and can source 1 ma to +5 volts, or sink 10 ma. Do not exceed these current levels, or the **Controller** may reset. Analog input pins measure voltage levels, such as potentiometers, with 10-bit resolution. The **J** display is the digital representation of the analog value, with a full range from 0000 (zero) to 1023 (+5V). If used for digital switch inputs, then any value near 0000 is a digital low while any value close to 1024 is a high. Use a 10K or 25K ohm potentiometer. An internal analog input may exist as Sw2.

You specify the **pin** number *and* the **pin mode** with the **p** command (little **p**). Two inputs are expected, for example:
p=3: **5** mode=01: **13** 1st value is the connector pin number.
2nd value is the mode, ranging from 00 to 31.

A pin value of 0 will ignore any pin and its mode in tht Motion Profile. Valid Pin Modes are:

00 – wait until pin is low, then step
01 – wait until pin is high, then step
09 – at power-up, count pin CTR, using Profile 01 repeat value
12 – step until pin goes high, then stop smoothly
13 – step until pin goes low, then stop smoothly
14 -- step until analog pin > ADC value, then stop smoothly
15 -- step until analog pin < ADC value, then stop smoothly
17 – output a 2-5 microsecond pulse on the output pin each motor step. This pulse starts at +5v and goes low.
28 – output pin is high at start, low after motion profile
29 – output pin is low at start, high after motion profile

Driver and Input Profile Modes restrict some pins as general input. For **repeat** and/or reciprocal motions, an input triggers only 1 motion, not multiple motions.

I (Input) Profile Mode

The **I** command enables the input pins to randomly run a set of 3 motion profiles. This gives you great standalone operation, and can be used in other modes. For example, 5 push buttons can select any of 5 positions for .10, .50, 1.00, 2.00, and 5.00 inches of travel. Then you can move a precise distance by any combination of button pushing. The **I** value also enables the input pins according to a priority. Example:

I=00: **12** enables the 3rd, 4th, and 5th input pins to trigger their 3 profiles. Your new value is auto-saved.

I=06 runs Motion Profiles 06, 07, & 08
I=09 runs Profiles 09, 10, & 11, enables 2nd thru 5th pins
I=12 runs Profiles 12, 13, & 14, enables 3rd, 4th & 5th pins
I=15 runs Profiles 15, 16, & 17, enables 4th & 5th pins
I=18 runs Profiles 18, 19, & 20, enables only 5th pin

These new analog **I** Profile Mode will trigger the 3 profiles when the analog pin value is lower than the **k** ADC value you specify:

I =76 runs Profiles 18, 19, & 20, enables all pins (DB15 only)
I =85 runs Profiles 15, 16, & 17, enables all pins (not DB15)
I =88 runs Profiles 18, 19, & 20, enables all pins (not DB15)

Example: **I** =88 and **k** value = 0910. Profiles 18, 19, & 20 run when the analog value on the 5th pin is lower than 0910, approx. 4.5 volts.

Input pins are active ***low***. Setting **I**= 06 allows all 5 input pins to trigger each associated set of 3 motion profiles. Setting **I**=15 makes motion profiles 15-17 run when the 3rd input pin is *low**, and*** enables the higher range of profiles 18-20, which will run if 5th input is low, but disables the 1st, 2nd, or 3rd pins. Once triggered, the 3 profiles run to completion, even if the input pin is raised high, so use caution.

To run only 1 profile when the input goes low, just set the remaining 2 profiles with **N**=0000000 and **t**=00000 so that they are ignored. You may also use a lower numbered input pin with any higher numbered pin mode, but some combinations are illogical. Driver Mode or encoder/index disables the corresponding Input Profile Modes. Sending 3 @'s in a row (@@@) clears **I** to 00.

Motion Profiles

The **Controller** stores all motor stepping parameters for each of the 24 motion profiles. You can view and edit these profiles individually, and then set them to automatically execute from memory upon power-up. The motion profiles are stored in EE memory, and thus retain their values when power is off.

The major purpose of motion profiles is to allow a standard programmed **Controller** to operate as a standalone unit, not needing a PC or external **Controller**. Once set up, only a power supply is needed. With the **Q** command, you specify any of the 24 profiles as the starting profile, and a subsequent profile as the last one, for example, profiles #05 through #12. The **Controller** will sequentially run these motion profiles, and automatically repeat after the last one is completed.

You can easily modify existing motion profiles to fit your own application by entering values for acceleration, SPS, etc. The **Controller** provides for a broad range of applications. All you do is change the profiles.

You can change motion profiles for designing, trouble-shooting, demonstrating, or marketing purposes. For example, most mechanisms operate at blinding fast speed—the motion profiles can be altered to slow down an interesting movement so that fine details can be observed. Since most applications only use less than 5 profiles, you can create an entire second range of motion profiles that operate at slow speed. You can even offer this feature to your customers.

The commands for retrieving, displaying, and saving motion profiles are: **P, Q, U,** and **v** (little v). For each of these commands, simply type the letter of the command. The **Controller** responds with a '=', then you type the required numbers. If you issue a **P, Q,** or **v** command, changes to current parameters are lost, unless you save them first by using the **U** command.

P=**nn** Display the motion profile specified by **nn**.

Q=**nn yy** Quit command mode and start motion profile (standalone) mode. Thereafter, at power up or a reset, the **Controller** will automatically run the motion profiles you specify with the **nn** and **yy** numbers.

U=08**:** **nn** Update and save the current motor parameters to the motion profile specified by **nn**. Example: **U**=08**:03** will store the current values in **08** to motion profile **03**.

v View all 24 motion profiles. The **Controller** displays a header line, followed by 24 data lines.

Special note for the **U** command: it stores the ***last*** **G**, **g**, **S** or **s** value executed, into the motion profile you specify. This is the only command to store the **G**, **g**, **S** or **s** value.

You can enable the Controller to auto-home at startup by setting **h** (under the little **c** menu) to 1, see Home Command section.

With a few simple command settings, you can create very interesting motion profiles. For instance, if P #01 is CCW and set for mode #13, P #02 is CW and set for mode #12, and you have a switch connected to an input pin, then after you start motion profile mode with the **Q** command, the switch position determines direction of rotation!

You stop the **Controller** from executing motion profiles by typing the reset command **@**. Then send a little **w** to wake it up. The **Controller** will execute motion profiles when you again issue the **Q** command. In the normal command mode, you can change any motion profile and then save it, overwriting any existing motion profile. The motor stepping parameters may also be loaded by sending an ASCII text file. The format of this file may be any of 3 formats: csv (comma separated values), space separated values, or nothing between values. An example of a csv format, easily created from MS Excel or Notepad, is:

A0200,V04400,B000,N0080000,U12

Note that the order is not important, except the U12 command is last, which saves the input values to Profile number 12. You may also have CR and/or LF after each input command. If you send multiple lines, you must insert a 300-millisecond delay after each line to allow the **Controller** to finish writing its internal values. The **Controller** does not buffer input commands. Windows Notepad or similar text editors can be used to capture information via cut and paste, or to set parameters. Note that **ee** writes require 10ms per character to save data in the **Controller**. When running motion profiles, the **Controller** outputs the letter P, the profile number, and a space, followed by the repeat number(s) (counting down) and a space, if echo is on. The life expectancy of the EE memory is 100,000 write commands (**Q** or **U**). Do not exceed this number.

Driver Mode

You set Driver Mode for CNC operation by typing little **c** then little **d**., and is saved immediately. Setup your CNC program, which send steps and direction pulses from a PC LPT1 parallel port, or other device. The **Controller** steps the motor using 2 external input signals Step and Direction:

DIR	0 is CW, logic 1 is CCW
STEP	negative going pulse, > 5 usec

The **Controller** is CNC-ready, and runs with almost any CNC PC or PLC program and +5volt logic. An enable signal is not needed because the **Controller** will automatically stop when the STEP pulses cease. The **Controller** *serial port* is ***not*** used for STEP and DIR. See the CNC Diagram for more details. The 10A-RS232 and 10A-USB Controllers now run Driver Mode.

In Driver Mode,iInternal Torque (amperage), temperature, Full/half step, Brake, and Encoder controls are maintained automatically by the **Controller**, using the values of Motion Profile #01. Only a reset @ serial character terminates Driver Mode and enables Command mode. Driver Mode serially responds only to @, and normally just echoes other characters. Switches, and Input Profile activations give you easy jogging or

other semi-automatic positioning. Since Driver Mode uses two input pins for step/direction, the corresponding Input Profile Modes are disabled. v3.45 improved Driver Mode (step/dir), automatically processes active high or low pulses, and with no time restrictions.

Since it is possible to run the motor using an input pin or the manual switches, do not send CNC step pulses while the motor is already stepping, or they will be lost.

Onboard Manual Switches

Your **Controller** may have onboard button switches for manual rotation**. Sw1** and **Sw2** run 2 Motion Profiles each: #21-22 and #23-24 respectively. A 3rd manual switch **Sw3**, if present, runs Motion Profile #03 to completion once triggered. You can set the number of steps, direction, time delay, and other parameters for each switch. When pushed, **Pxx Sw1**, **Pxx Sw 2**, or **Pxx Sw 3** is sent to the serial port followed by the normal prompt. They are very useful, especially for robotics or CNC machines. For example, you can easily set **Sw1** for exactly 1 revolution CCW and **Sw2** for 1 revolution CW.

Pushing **Sw1** or **Sw2** continuously results in repeating its 2 Profiles. When you release the switch, the motor decelerates and stops. If you release the switch before it completes its number of steps, then the **t** time delay is ignored. Any manual switch is ignored if the motor is already running. The repeat value **r** should be 001. Pushing both switches simultaneously causes an emergency stop. Pushing both again for a second resets the **Controller**. Custom firmware is possible for these switches. Unless you saved your changes, pushing any switch erases your temporary unsaved changes.

Help, Hints and Tips

? Displays the simple command help.

The Excitron motors and **Controllers** definitely have the torque range specified, and generally, torque decreases as speed increases. Small burrs, chips, non-flatness, or dents on your moving parts can result in the motor stalling. The faster you step, the worse the effect of the small burr. Also be aware that even small machine screws can exert a thousand pounds of force, bend heavy steel parts, and thus bind up moving parts. Tightening a small timing belt can bend steel motor shafts easily, due to the leverage action.

Stepper motors run faster and stronger with direct shaft attachment. Best is to slide our motor shaft into a precision bored hole in your part, and clamp using a C collar clamp. Note that if your bored hole is off even by .003", your top speed will be limited. Non-rigid "rubber" couplers introduce vibration and phase lags, which can cause motor stalling. Best is to shock mount the motor, not the drive parts.

If your motor just hums without stepping, check your **A**, **V**, and **T** values. Use low **A** (acceleration) so that your motor with load, has time to speed up, and to verify correct stepping at lower rates. DO NOT increase **T** while trying to get the motor to spin faster—see other notes about **T**. If your **Controller** has manual push buttons, push one to see correct rotation. Once you verify correct running, then increase **A** and **V** to suit your purpose.

Use multiple Motion Profiles for added control. For instance, with **r** = 250 (repeat = 2,500) set in three consecutive Motion Profiles, then when you run all three profiles, the repeat quantity is 7,500. For longer time delays, use multiple Motion Profiles with **t** set to 65000 msec each.

If 1, 2, or 3 Motion Profiles suit your operation without auto-repeating, then use **I Input Profile Mode**, which trigger ***once*** each time that particular input pin goes low. And this works well with up to 5 inputs. If you need 4 or more profiles, and automatically repeating forever is desired, then use **Motion Profile Q Mode**. In either mode, once started, an input pin can control motion. An example is **I**=09, and Profile #10 has **p**=5 mode=00. Pushing a switch attached to **In5** triggers Profiles #09-12, but motion stops at Profile #10 because **p**=5 mode=00. Pushing your switch again triggers Profile #10 then #11.

Optical Encoder Details

Excitron **Controllers** can connect directly to an incremental optical encoder using A/B quadrature signals, and also to an Index (once per revolution) signal. Excitron optical encoders are best suited for mounting on the motor shaft..

Many motor control situations benefit by connecting the Excitron optical encoder directly to your external controller, and not to the Excitron Controller. The reason is your external controller can read values while running, and it can exercise total control over the Excitron motor.

You enable and set the motor's full steps/revolution, Index and/or Encoder values with commands on the **c** (controller) submenu, refer to that section. At the end of each motion, the index and/or encoder position is sent:

G R=+000000036 P=+00000912 ok

"ok" is sent if no errors were detected, otherwise " ?" is sent. If you have an encoder, the Index is not checked for errors.

If the Index or Encoder is enabled, and you do not want the controller to automatically stop the motor, then use command:

e=1: 0 disables auto-stop and 1 enables it. If enabled, the controller will cease stepping immediately (without deceleration) when physical rotation of the encoder or index stops.

You can store the **e** value for each Motion Profile by using the **U** command. If a physical problem causes a motor stoppage, your mechanism may be left in an unknown state. Action is needed to remedy the stoppage. Subsequent motion may be harmful to people or your mechanism, so use care in all situations.

Note that if you have an encoder or Index installed, the pin/modes and the Input Profile Modes associated with these input pins are disabled. For example with the 100A-TTL-4 Controller, if you have **I=06** and an Encoder and Index installed, then Input Profile Modes 09, 12, and 15 will not trigger. Sending @@@ will make **I**= 00. If you have some Input pins already set, and want to add an Encoder or Index, set p=0 before attaching an Encoder or Index.

After each movement that reverses direction, the **Controller** measures backlash approximately, if any. This value can be seen with the little **i** command.

Additional Command Details

The **i** command provides important information:

```
Excitron Corp  10/01/09 3.45 10A-TTL-3SW                    ← firmware date and revision, Controller name

s  baud echo In Tmax degC    Vs Steps/131K V Baud1          ← Controller values
M 57.6K    1 00  028  021 24.14   00000318 0 19.2K

Index Rotations Encdr Motor Ratio  Position v Play home     ← Encoder values
    0 +00000000 +0800 00200 00002 +00000000 1  007    0

P# D M G  Acc a2 Brk  Number  Vsps Trq   msec rep pin/mode f Fnum msps DU e  ADC
01 C 0 g 0200 50 000 0004800 03600 012 -01000 004   3    01 0 0100 0125  1 0 0344     ← profile values
```

The 2nd & 3rd lines show hardware values: **s** is slave number (M, a-z for master, slaves are A-Z); **baud** is primary serial port bits/sec; **echo** =0 (off) or 1 (on); In = 1st motion profile set of 3 for Input Profile Mode, **Tmax** is the maximum T for command entry—NOTE: the safe *ALLOWABLE* T value is lower than Tmax--depends on voltage, use caution. Some **Controllers** may measure temperature, voltage, or have an encoder: **deg**C= degreeC of the **Controller** (and motor if assembled together), **Vs =** power supply voltage, Steps**/131K** shows approximate number of motor steps / 131K; you can change this value with **c>r** command, and save immediately with the **c>u** command. DB15 type Controllers auto-save these steps every 10-15 minutes. **A** shows the global Analog Vsps change value. Baud1 is Simple Serial Bus baud, displayed only if Master. The CR (return) delay value is not displayed, to view or change, type **c** then **c**.

Next 2 lines show: **Index** = once/rev Index pulse: 0 = no, 1 = yes. **Rotations** = number of Index revolutions. **Encdr** = Encoder pulses/rev. (9900 max), if installed—make 0000 if you do not have an encoder, or your motor will not run; +- is for encoder positive direction relative to the motor. **Motor** = motor (x gear ratio) full steps/rev. **Position** = number of Encoder pulses/rev, + (CW) or – (CCW), 99999999 max. **v** is view motor steps, Index, and/or Encoder real-time on serial port. **Play** is backlash measured by the **Controller** if an Encoder exists, and is updated once each time you reverse direction and Encoder pulses occur.

The last two lines, which are also displayed after **i, P** or **v** commands, are described below. Note that this display also defines the quantity of input numbers for each parameter. The following table describes each of the column headings, and defines the minimum, maximum, or allowable values for the parameters.

Heading	Command	Description, these values apply to the motion profile, and are all saved when power is off with the **U** command
P#	P	Profile number, 01 to 24
D	C or W	direction of rotation, C (clockwise) or W (counter clockwise)
M	M	Micro stepping, 0 (full) or 1 (half)
G	G,g,S,or s	Go—run this motion—**G** (single), **g** (reciprocal), **S** (continuous), **s** (pseudo-sinusoidal)
Acc	A	Acceleration, 0002 to 1000, this is also the 1st step/sec value. If Acc > Vsps, ACC becomes = SPS
a2	a	2nd acceleration value , 00 (slowest) to 63 (fastest)
Brk	B	Brake, 000 to 138—use care with braking values—134 is full on!
Number	N	Number of steps, 0000000 to 9999999 (for larger values, use **S** command for continuous stepping)
Vsps	V	Velocity maximum steps per second, 00002 to 26,110, limited by **A** value
Trq	T	Torque value (non-dimensional), 001 toTmax—use carefully! Motors have different Tmax (maximum).
Msec	t	time delay in milliseconds, before (-) or after (+) the motion profile, 00000 to 65000
rep	r	repeat quantity, 01 to 255, if >100, qty = 10 x **r**. While running, the remaining number is output.
pin/mode	p	Input pins to control this motion (0 means to ignore), then enter a mode number, 00-31.
f	f	run at a flat sps using **msps**, 0 = ignore, 1 = at start, 2 = at end, 3 = both. (for end, Acc must be < msps)
Fnum	F	number of steps to run at **msps**, minimum is 0002, according to "**f**". Add to **N** to get total steps.
msps	m	minimum sps for running at start or end, minimum is 0002, msps must be < Vsps, according to "**f**".
DU	D	DU (down/up) change speed on-the-fly using ADC analog input pin, 0 = ignore, 1 = enable, SPS_dir sets direction.
e	e	enable (=1) or disable (=0) auto-stopping the motor if detected by the Encoder or Index.
ADC	k	Analog input value 0 to 1023, for pin/modes and for **I** Profile Modes 76, 85, or 88.

Options

- RS232-TTL adaptor— built-in or add-on adaptor for connecting to PCs, PLCs, etc. using +-10 volt RS232 serial lines.
- USB –built-in or add-on adaptors (1-port, 2-port, 4-port) for USB serial communications.
- RS485-TTL adaptor—for connecting to PCs, PLCs, and other equipment using 5 volt differential serial lines.
- Firmware modifications--to reduce or eliminate external electronics, such as PLCs or computers.
- Rear shaft, knob, and optical encoders from 1 to 2,000 positions per revolution.

Disclaimer—all values and statements contained herein are subject to change without written notice. No products manufactured by Excitron may be used for life support equipment. **Use extreme caution when operating--severe injury or death can occur.**

Examples of using I Profile Modes with pin/modes

You can achieve remarkable and interesting motion by combining **I** Profile Modes with a pin/modes. The following example is how we setup the Motion Profiles for use with our Switchbox. It explains the simple setup, and to offer further help about the differences between **I Profile Modes** and **Input pin/modes**. Please read those sections. The following view is obtained from Motor/Controller 100A-TTL-86-118 by sending the **i** and **v** commands, then copy/paste here:

```
M>i
Excitron Corp  07/21/09 v3.44  100A-TTL-4

s  baud echo In Tmax degC    Vs Steps/131K V Baud1
M 57.6K    1 12  020  029 24.17   00000004 1 19.2K      ← notice that In=12

Index Rotations Encdr Motor Ratio  Position v Play home
    0 +00000000 -0000 00200 00000 +00000000 0  000    0

P# D M G  Acc a2 Brk  Number  Vsps Trq   msec rep pin/mode f Fnum msps DU e  ADC
01 W 1 G 0080 32 000 0000800 02000 008 -00000 001   0   00 0 0100 0100  1 0 0255
M>v
P# D M G  Acc a2 Brk  Number  Vsps Trq   msec rep pin/mode f Fnum msps DU e  ADC
01 C 1 G 0080 32 000 0000400 06000 006 -00200 001   0   00 0 0100 0100  1 0 0255 ← Driver Mode
02 C 1 G 0080 32 000 0000400 02000 008 -00000 001   0   00 0 0100 0100  0 0 0255
03 C 1 G 0080 32 000 0001600 03000 009 +00200 001   0   00 0 0002 0250  0 0 0255 ← $w3
04 W 1 G 0060 32 000 0003000 01800 008 -00030 001   0   00 0 0003 0045  0 0 0255 ← home
05 C 1 G 0060 32 000 0000020 00800 008 -00200 001   0   00 1 0002 0100  0 0 0255 ← home

06 C 1 G 0100 32 000 9999999 06000 008 -00200 001   3   12 0 0002 0100  0 0 0255 ← I=06
07 C 1 G 0080 32 000 0000000 03000 008 -00000 001   0   00 0 0002 0100  1 0 0255
08 C 1 G 0080 32 000 0000000 03000 008 -00000 001   0   00 0 0002 0100  1 0 0255

09 W 1 G 0080 32 000 9999999 01800 008 -00000 001   5   12 0 0002 0100  0 0 0255 ← I=09
10 W 1 G 0080 32 000 0000000 03000 008 -00000 001   0   00 0 0004 0050  1 0 0255
11 W 1 G 0080 32 000 0000000 03000 008 -00000 001   0   00 0 0004 0050  1 0 0255

12 C 1 G 0080 32 000 9999999 02000 008 +00200 001   7   12 0 0002 0100  1 0 0255 ← I=12
13 W 1 G 0080 32 000 0000000 03000 008 -00000 001   0   00 0 0004 0050  1 0 0255
14 W 1 G 0080 32 000 0000000 03000 008 -00000 001   0   00 0 0004 0050  1 0 0255

15 W 1 G 0080 32 000 9999999 04000 008 -00050 001   8   12 0 0002 0100  1 0 0255 ← I=15
16 W 1 G 0080 32 000 0000000 03000 008 -00000 001   0   00 0 0004 0050  1 0 0255
17 W 1 G 0080 32 000 0000000 03000 008 -00000 001   0   00 0 0004 0050  1 0 0255

18 C 1 G 0080 32 000 9999999 06000 008 -00400 001   9   12 0 0002 0100  1 0 0255 ← I=18
19 W 1 G 0080 32 000 0000000 03000 008 -00000 001   0   00 0 0004 0050  1 0 0255
20 W 1 G 0080 32 000 0000000 03000 008 -00000 001   0   00 0 0004 0050  1 0 0255

21 W 1 G 0080 32 000 0000800 02400 010 +00700 001   0   00 0 0002 0100  0 0 0255 ← $w1
22 W 1 G 0100 32 000 0000000 02400 010 -00000 001   0   00 0 0002 0100  0 0 0255 ← $w1

23 C 1 G 0080 32 000 0000800 02400 010 +00700 001   0   00 0 0002 0100  0 0 0255 ← $w1
24 C 1 G 0100 32 000 0000000 02400 010 -00000 001   0   00 0 0002 0100  0 0 0255 ← $w1
```

Extra line spacing is shown here for clarity. These Motion Profiles and pin numbers are based on the 100A-TTL-86-118, other Controllers have different pin numbers and torque. The far right column shows the special Motion Profile functions. The importance for the **I** Input Profiles Mode is the order of the input pins. Because the **I** Input Profiles value **= 12**, then IN3 or IN5 do not trigger any Motion Profiles, and IN7, IN8, and IN9 will trigger running their set of 3 Motion Profiles. Example: profiles 12, 13, and 14 will run when the Controller ***3rd*** input pin goes low. For the 100A-TTL-4, the 3rd input pin is IN7. Since the Switchbox has a pot for analog input IN3, and by setting D=1, then IN5 is used for the direction. This a global setting, and works on any Motion Profile that has D=1.

Notice that in Profile 12, the same input pin IN7 is set with mode 12, which is "run until this pin goes high") and this is used to stop motion. Why does this work? First, a switch is wired to IN7. Flipping the IN7 switch low (ON) triggers the 3 motion profiles, then when you flip the switch to high (OFF), the motor gracefully stops because of pin 7 mode 12. Same pin, but two different functions. Similarly for pin 8 mode 12 in Profile 18.

Errata

3.45 Improved Driver Mode (step/dir), automatically processes active high or low pulses, and with no time restrictions.
3.44 Speed on-the-fly using the analog input pin, and Command **V**, if set to 1, on the little **c** menu will send Vsps 16X per second.
3.42 Change speed on-the-fly using the analog input pin, and command **D** enables this change for each motion profile. Command **A** on the little **c** menu is deactivated. Removed pin/modes 04 to 08, since **D** enables speed change.
3.40 Added 1.5 second time delay and brief braking at power-up to stabilize external power supplies and to sync the motor. For Input Pins: added Mode 17; removed Mode 16. Motor torque is increased 10% if half-stepping by powering the half-step 1.5 times normal.
3.36 Driver Mode works for all Controllers. New Mode 76 (DB15) & 85 allows analog triggering of 3 Motion Profiles.