

## Introduction and Quick Setup

Excitron's **Faster Than Fast® Au Controllercoders** are fully integrated with our stepper motors to make your motion control simple and easy. We pack years of design and plenty of power into ultra small gold plated Au Controllercoders, loaded with innovative features. 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".

They all have standard RS232 (+-10v) for serial port communication to any computer and PLC. USB-RS232 serial port adaptors are available. Various size, power, and control functions give you the best selection for your project.

Our Au Controllercoder /motor configuration and part number system is also simple—**Au57-76M** is the number for the **Au Controllercoder** assembled with our SM57-76M stepper motor.

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

Our experts are happy to assist you in selecting the best Au Controllercoder, stepper motor, and accessories for your project. We also offer linear slides, belt sliders, 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 power supply/RS232 connector, which is a small 2x4 housing, to the **Au Controllercoder**. Note the polarizing post must be inserted into the open slot in the motor cover.
- If you have a USB-RS232 adapter, connect your USB cable between your computer and the **Au Controllercoder** RS232 DB9 connector. PC Software drivers must be installed. 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 **Au Controllercoder** and it displays:

```
Excitron Corp 01/01/11 v4.12 Au42-47M
Controllercodermotor +Vs=12.20
```

```
M>
```

- Type **G** to run the stepper motor.
- Type little **i** to see all **Au Controllercoder** information and the motion profile values.
- Change direction (**C** or **W**), **Vsps**, or number of steps **N**, type **G**, and see the difference.
- Type **?** to display a brief command help list.
- The **Au Controllercoder** 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](mailto:info@excitron.com).

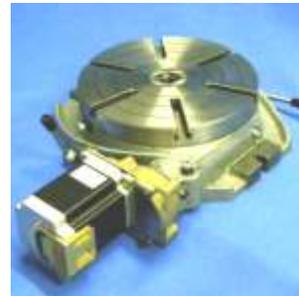
**Au57-40M**



Au57-40M



Mini Z-Lift with camera



10" Rotary Table MRT10-86-118-450



AuG86-118-5



SM60-86



3.3' (1m) / second Wire Slider WS70-012

### 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 **Au Controllercoders**. See the individual motor **Au Controllercoder** data sheets for additional details about connector pins.

The **Au Controllercoder** 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 **Au Controllercoder** 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 **Au Controllercoders**. Our proprietary circuitry was first developed and patented in **1976**. Improvements continue steadily to maximize motor and **Au Controllercoder** efficiency, which maximizes torque.

Three modes of operation are available: Command, Motion Profile, Driver, and **I** Input Profile. Onboard optical Encoders are optional and available in the near future.

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 **Au Controllercoder** 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 +3.3 volt logic input pins. Two input pins accept analog 0-3.3v and converts it to a digital value. **Au Controllercoders** have one TTL/CMOS +3.3v 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 **Au Controllercoder** electronics. With a 24-v power supply, the **Au Controllercoder** only dissipates 0.025 watt, which is insignificant.

All internal voltages are self-generated and thus require only a single voltage supply. Built-in temperature sensing on **Au Controllercoders**, safeguards the electronics and motor from thermal burn out, the limit is 170 degF.

Many aspects and limitations in competitor's Au Controllercoders are obsolete. Excitron **Au Controllercoders** drives any size unipolar stepper motor, even our 110-180—a 25-pound Super-size 42! The power supply voltage range for Au86 and Au110 large motors is 9 to 48V, while all other **Au Controllercoders** operate from 3.3v to 30v. Many options are available. All **Au Controllercoders** 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

Electronic damping on last step taken for precise stepping.

Built-in **Optical Encoder** for position control, see Encoder section.

Driver Mode is integrated with normal Command Mode so you can have access to all commands while accepting step/dir.

Exciting speed control on-the-fly by using the IN3 analog input pin.

Motor torque is increased 10% if half-stepping by powering the half-step at 1.4 times normal. Half-stepping is now superior to full-stepping.

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

Command "H" allows automatically finding home position at power up. Command "h" rotates motor shaft to zero position.

Analog signals on IN2 and IN3 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 '>'. Au86 and Au110 have an optional 2<sup>nd</sup> UART for controlling a slave device.

View each motor step and encoder pulse via serial port using the little **c>v** command. Also, **c>o** command sends a 5 microsecond pulse on the output pin Out4 for each motor step. Useful for external step measuring and for syncing other Au Controllercoders with exact same motion profile.

## What's Coming

Wireless communication, Zigbee, and internet address 6LoPAN.

**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 **Au Controllercoders** can operate standalone, or in various modes. The onboard UART serial port is RS232, which contains all the electronics for RS232 (+6 volts), included with every **Au Controllercoder** is a DB9 female connector.

You will also need a DC switching regulated power supply, with sufficient amperage to drive your step motor. The power supply voltage range for Au86 and Au110 large motors is 9 to 28V, while all others operate from 3.3v to 28v. These and other accessories are available on our web site. See Au\_Controllercoder\_Pins drawing for interface pins.

The power supply wires should be at least 20 AWG stranded, 18 AWG is best, and as short as possible, 4 to 12 inches. 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 **Au Controllercoders** can operate at 2 to 3 times the rated value for intermittent operation. If the power supply amperage is inadequate, the **Au Controllercoder** may pull the power supply voltage low enough to cause a reset of the **Au Controllercoder**. This is harmless to the **Au Controllercoder**, but stops your motion. If you operate in Motion Profile mode (described later), then the **Au Controllercoder** will begin anew at the first motion profile selected by the **U** command.

A computer running Hyperterminal, RealTerm, 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 alphanumeric. The **Au Controllercoder** will echo all characters received by default, therefore turn off Echo in your communication program. Once setup in standalone mode, the RS232 serial port is no longer needed. Each Au Controllercoder has an assignable address, the default and normal address is 'M'.

## Sending Commands to the Au Controllercoder

The format of **Au Controllercoder** 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 **Au Controllercoder** 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 **Au Controllercoder** 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. In other modes, the mode character is also sent.

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 alphanumeric must be typed. For example, to enter a step quantity of 188 steps, type:

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

## Changing Au Controllercoder Configuration

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

- b=57.6K:** select baud: 1=4.8K, 2=9.6K, 3=19.2K, 4=38.4K, 5=57.6K (default), 6=115,200, 7=230.4K. Type just *one* digit of the baud and immediately the baud is changed.
- B=57.6K:** same as b baud but for optional 2<sup>nd</sup> serial port.
- d** driver mode, see Driver Mode section for details.
- E=0060:** Encoder error range; 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, runs at reset/power.
- P=+-00000836:** change Encoder position
- o=0:** output a 5 usec pulse on Out5 for each motor step
- s=M:** slave address **A-Z** or **a-z**, using Simple Serial Bus
- V=1:** View extra Encoder data.
- v=1:** view each motor step pulses while the motor is stepping: 0 = off, 1 = on. If on, each motor step CW is an **m**, CCW is an **M**. 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, so use wisely.
- q** quit and return to main command menu.

**Simple Serial Bus** connects multiple **Au Controllercoders** 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 **Au Controllercoder** 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 **Au Controllercoder**, its Transmit to the next **Au Controllercoder** Receive, and so on, and the last **Au Controllercoder** Transmit is wired to the Receive of the host. See the **Au Controllercoder** Connections sheet.

## 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 **Au Controllercoder** displays the equals sign, the existing value (from 1 to 7 numbers), colon and a space (most commands).

**A=0100: nnnn** Acceleration (1<sup>st</sup> Vsps), 0114 is the minimum and 0120 is typical. If **A** > Vsps, then **A** is set to Vsps.

**a=18: nn** 2<sup>nd</sup> 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 IN3 analog pin: 0=no, 1=enable, IN4 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** repeat quantity, from 001 to 255, if > 100, then  $r = 10 \times r$ . The remaining repeat value is output *after* each motion, except the last.

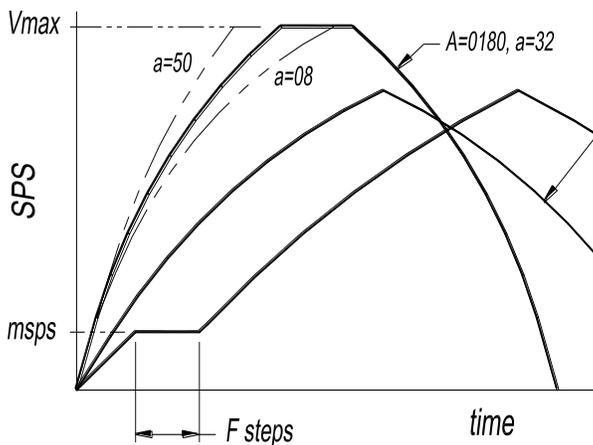
**t=+-00888: nnnnn** time delay either before (-) or after (+) the motor motion. Input - or +, then 5 numbers representing milliseconds, e.g., **t=+00500** defines a 500 millisecond time delay after the motor motion.

**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 than **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 **Au Controllercoder** 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 **Au Controllercoders**. Most step motors are rated for 2 to 5 volts; so Excitron's current limiting allows them to be connected to a higher voltage (12 and 24 volt being most popular) power supply. You control precisely the amount of torque in the motor with one simple command:

**T=050: nnn** sets the internal percentage of full motor torque.

A low value for **T**, such as 020, produces low torque. Typically use **T=040** to **080**. 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--it is only 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 buildup. The **Au Controllercoder** measures power supply voltage at startup, and automatically adjusts the amperage. You do not need to adjust **T** if swapping power supplies.

After a certain SPS, the **Au Controllercoder** 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 2 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 hot to the touch, and may burn you. Selecting higher torque than allowed may damage the **Au Controllercoder**. You are responsible for any damage to the **Au Controllercoder** 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 or power

is cycled. Note that this terminates Motion Profile or Driver Modes, and the **Au Controllercoder** 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.
- ) run one motor step CW, any unsaved changes are discarded.
- ( run one motor step CCW.

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 IN3 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 IN4 sets the direction—low is CW, high is CCW.

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=0114-0200** and the 2<sup>nd</sup> 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.

After running, the motor coils are energized for about half a second to electronic dampen the motor shaft for ultimate step response. You may immediately start another run during this half-second.

## 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 040 to 060 produce little braking, and values between 080 and 100 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 and 040 to 100.

Power saved by minimizing braking results in greater torque available for motor stepping. Higher current results in higher temperatures. **B=000** also reduces the **Au Controllercoder** 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 Au Controllercoder to auto-home at startup by setting **h** (under the little **c** menu) to 1:

**c>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.

Rotate the motor shaft to position zero, if Encoder enabled:

**h** (this is not the **c>h** command)

## 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 **Au Controllercoder** 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 **Au Controllercoder** 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 **Au Controllercoder** will heat up faster while running at slower speeds. At SPS over 3,000, depending on power supply voltage, the motor's inductance, 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 **Au Controllercoder** operation (up to the high temperature 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 **Au Controllercoder** 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 reduce resonance and miss-stepping. You should always run using half-step (**M=1**), which helps a great deal. When changing from half to full stepping, the **Au Controllercoder** may shift a half step to always be on a full step.

## Power-ON Reset

The **Au Controllercoder** has sophisticated power-on reset and brownout voltage protection circuitry. This protects from power supply glitches, and increases the **Au Controllercoder's** robustness and integrity. The Au Controllercoder waits a few seconds for your power supply to stabilize before allowing any motor motion. The RESET pin is normally high. To reset the **Au Controllercoder**, 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 eeprom 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 **Au Controllercoder** with the motor's last rotor position. The high value filter capacitors in the power supply provide considerable power supply voltage spike attenuation, and a 1-4 second hold-up time. When switching off, always wait briefly before unplugging.

## Software Reset and @ Commands

Software reset by sending @, which restarts the **Au Controllercoder** as if a power-on reset occurred. Use this command to terminate Motion Profile and Driver Modes, and run serial commands, then a little **w** must be sent to wake up the **Au Controllercoder**. Other special @ commands are:

@@{ & **STX** (ctrl-B) & an address letter--auto-sets multiple slave's addresses. May have to try this command twice.

@@{ followed by any character (besides STX)--sends slave address.

@@@ (3 @'s in a row)--clears Home & 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 pinmodes. 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 +3.3v) and are also +5 volt tolerant. They have Schmidt triggers for noise immunity and +-30kV static protection. You can read your **Au Controllercoder** pin values and the Input Profile Modes with the **J** command:

```
IPM: +86 -09 -12 +15 +18 +21
Pin: In2 In3 In4 In6 In7 In8 Out5
      1018 1015 1 1 1 1 0
```

The In# is the 2x5 I/O connector pin number, and the J command also shows the matching **Input Profile Mode** number, useful for the next section. Each input pin is normally high (+3.3v) 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 In4-In8 are digital only, and have an 8.2K (+- 5%) pull-up resistor to +3.3 volts. Input pins In2 and In3 are digital or analog, depending on the pinmodes, and have a pull-up resistor of 35K (+- 10%). The output pin Out5 is normally low (0 volts) and can source 1 ma to +3.3 volts, or sink 10 ma. The +3.3 volt pin 10 can provide 3ma for external circuitry. Do not exceed these current levels, or the **Au Controllercoder** 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 (+3.3V). 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** 1<sup>st</sup> value is the connector pin number. 2<sup>nd</sup> value is the mode, ranging from 00 to 31.

A pin value of 0 will ignore any pin and its mode in that 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
- 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 all other modes. For example, 6 push buttons can select any of 6 positions for .01, .10, .50, 1.00, 2.00, and 5.00 inches of travel. The **I** values are set independently of each other, and these are the allowable values:

```
I=+06 or +86 runs Profiles 06, 07, & 08 (In2)
I=+09 or +89 runs Profiles 09, 10, & 11 (In3)
I=+12 runs Profiles 12, 13, & 14 (In4)
I=+15 runs Profiles 15, 16, & 17 (In6)
I=+18 runs Profiles 18, 19, & 20 (In7)
I=+21 runs Profiles 21, 22, & 23 (In8)
```

When you type **I**, the Controller sends all 6 current values, and waits for you to type + (enable) or - (disable) followed by one of the valid numbers, and your new value is auto-saved. The new analog **I** Profile Mode for In2 and In3 will trigger their 3 profiles when the analog pin value is **lower** than the **k** ADC value you

specify, in the Motion Profile, using the **k** command. Example: **I** =+86 and **k** value = 0910. Profiles 06, 07, & 08 run when the analog value on In2 pin is lower than 0910 (2.9 volts). Input pins are active **low**. 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, set the remaining 2 profiles with **N=000000** and **t=00000** so that they are ignored. You may also use an input pin and mode inside the Motion Profile. Driver Mode or external encoder disables the corresponding Input Profile Modes. Sending 3 @'s in a row (@@@) clears **I** to 00. See page 10 for more details.

## Motion Profiles

The **Au 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 **Au Controller** to operate as a standalone unit, not needing a PC or external **Au 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 **Au 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 **Au 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 **Au 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 **Au 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 **Au 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 Au 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 **Au Controller** from executing motion profiles by typing the reset command **@**. Then send a little **w** to wake it up. The **Au 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 **Au Controller** to finish writing its internal values. The **Au 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 **EEPROM** writes require 10ms per character to save data in the **Au Controller**. When running motion profiles, the **Au 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=1**, and it is saved immediately, and the prompt changes to **M d>**. Setup your favorite CNC program, which send steps and direction pulses from a PC LPT1 parallel port, or other device. The **Au Controller** steps the motor using 2 external input signals Step and Direction:

```
STEP  negative or positive going pulse, > 5 usec, this is IN7
DIR    0 is CW, logic 1 is CCW, this is IN8
```

Driver Mode is now integrated with Command serial mode so that you can also run all serial commands, such as **i** or **G**. The only precaution is if the motor is running using step/dir, do not run **G** or any other run command until Driver steps are complete.

The **Au Controller** is CNC-ready, and runs with almost any CNC PC or PLC program and +3.3 or +5 volt logic. An enable signal is not needed because the **Au Controller** will automatically stop when the STEP pulses cease. The **Au Controller** *serial port* is **not** used for STEP and DIR. See the CNC Diagram for more details.

In Driver Mode, internal Torque (amperage), temperature, Full/half step, Brake, and Encoder controls are maintained automatically by the **Au Controller**, using the current Motion Profile. Use caution if you change Profiles.

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. The 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 RS232 serial characters, do not send CNC step pulses while the motor is already stepping, and vice versa, or they will be lost.

## Help, Hints and Tips

? Displays the simple command help.

The Excitron motors and **Au Controllercoders** 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 you're 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 **Au Controllercoder** 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 6 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=+21**, and Profile #22 has **p=8 mode=00**. Pushing a switch attached to **In8** triggers Profiles #21-23, but motion stops at Profile #22 because **p=8 mode=00**. Pushing your switch again triggers Profile #22 then #23.

## Optical Encoder Details

Excitron **Au Controllercoders** contain an internal Optical Encoder. This feature gives you valuable motion feedback by measuring the motor shaft position directly, and then you know exactly where the motor rotated. Successful motion or jamming is indicated with serial port characters.

The Au Controllercoder constantly measures the encoder to determine that the motor is indeed physically stepping, and validates the Position

step counter, up to +-99999999 steps. The absolute Shaft value is also constantly updated real time, stored in volatile memory.

The Optical Encoder contains two. Unfortunately an incremental encoder has no idea where it is—it can only count pulses, which forces you to always find "home" or find some absolute placement by using an external extra sensor. Excitron is a technological market leader in motion control and dynamics.

You enable and set the Encoder (**Encdr**) value with a command on the **c** submenu, and any changes are immediately saved in eeprom memory. If enabled, the encoder position and shaft position is also sent at the end of each motion, example:

**G** P=+00000912 023 ok

"ok" is sent if no errors were detected, otherwise " ? " is sent.

The **main menu** commands associated with the Absolute Optical Encoder are:

- X** Measure and calibrate the encoder by rotating the motor exactly one revolution and storing the results in flash memory. Flash memory is limited to 10,000 writes, and this command is intended to only be run once, or perhaps weekly if values have changed significantly. Excitron ships all motors with this X command executed.
- x** Send the stored flash table of encoder values, this includes the heading. This command has no typical user purpose, and is included for advanced data processing.
- R** Recalibrate the motor shaft position by measuring the encoder values. You can view the current shaft position via the little **i** command.

**e=1**: 0 disables stop and 1 enables it. If enabled, the Au Controllercoder will cease stepping immediately (without deceleration) when the encoder detects physical rotation has stopped. **Encdr** value affects sensitivity to stoppage.

The **little "c" submenu** commands are:

- E** Encoder value for setting the range of allowed encoder errors. A value of zero disables the encoder entirely. Lower values of 10-30 tend to stop the motor even when no motor stepping fault has occurred. Typically use values of 032-080, which help ensure correct continuing motor stepping, yet will stop quickly if a true motor jam occurs.

If the Encoder is enabled, and you do not want the Au Controllercoder to automatically stop the motor if an encoder error occurred, then use the main menu command **e** (NOT the 'c' submenu **e** echo command):

If you do not have an encoder, then set **e=0** for each Motion Profile. You 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. All Index features are temporarily disabled.

## Additional Command Details

The **i** command provides important information:

Excitron Corp 01/01/11 v4.12 Au86-118 Controllercodermotor +Vs=24.08 ← firmware date/revision, Au number, volts

```
s baud echo In TQnum degC +Vs Steps/131K v Baud1
M 57.6K 1 00 028 021 24.08 - 0 19.2K ← Au Controllercoder values
```

```
Revs Encdr Motor Position Shaft v Play home out
- 0800 00200 +00000000 012 1 000 0 0 ← Encoder values
```

```
P# D M G Acc a2 Brk Number Vsps Trq% msec rep pinmode 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 2<sup>nd</sup> & 3<sup>rd</sup> 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 = 1<sup>st</sup> motion profile set of 3 for Input Profile Mode, **TQnum** is the motor's electrical torque value. All **Au Controllercoders** measure temperature and voltage: **degC**= degreeC of the **Au Controllercodermotor**, **Vs** = power supply voltage, Not enabled yet--**Steps/131K** shows approximate number of motor steps / 131K. **v** sends extra Encoder data. Baud1 is the Master Serial Bus baud, useable only if **m** is the controller address.

Next 2 lines show: **Revs** = temporarily disabled, **Encdr** = Encoder pulses/rev., if installed—make 0000 if you do not have an encoder, or your motor will not run properly. **Motor** = motor full steps/rev and is fixed for each motor. **Position** = number of Encoder pulses/rev, + (CW) or - (CCW), 99999999 max. **Shaft** = physical half-step position of the motor shaft, relative to your zero location. **v** = view motor steps real-time on serial port. **Play** is backlash measured by the **Au Controllercoder** if Encoder enabled, and is updated once each time you reverse direction and Encoder pulses occur (temporarily disabled). See **Encoder** section.

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 <b>U</b> 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— <b>G</b> (single), <b>g</b> (reciprocal), <b>S</b> (continuous), <b>s</b> (pseudo-sinusoidal)
Acc	A	Acceleration, 0114 to 1000, this is also the 1 <sup>st</sup> step/sec value. If Acc > Vsps, ACC becomes = SPS
a2	a	2 <sup>nd</sup> acceleration value, 00 (slowest) to 63 (fastest), value should be 02 for larger motors
Brk	B	Brake, 000 to 134—use care with braking values—132 is full on!
Number	N	Number of steps, 0000000 to 9999999 (for larger values, use <b>S</b> command for continuous stepping)
Vsps	V	Velocity maximum steps per second, 00114 to 22,000, limited by <b>A</b> value
Trq%	T	Torque percent of full rating, 020 to 200—use carefully, changing T changes the amperage!
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.
pinmode	p	Input pin to control this motion (0 means to ignore), then enter a mode number, 00-31.
f	f	run at a flat sps using <b>msps</b> , 0 = ignore, 1 = at start, 2 = at end, 3 = both. (for end, Acc must be < msps)
Fnum	F	number of steps to run at <b>msps</b> , minimum is 0002, according to “f”. Add to <b>N</b> to get total steps.
msps	m	minimum sps for running at start or end, minimum is 0114, 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.
ADC	k	Analog input value 0 to 1023, for pin/modes and for <b>I</b> Profile Modes 76, 85, or 88.

### Options

- 2<sup>nd</sup> RS232 serial— built-in on Au86 and Au110 series for connecting to PCs, PLCs, etc. using +-10 volt RS232 serial lines.
- USB-RS232 external adapters (1-port, 2-port, 4-port) for USB serial communications.
- Firmware modifications--to reduce or eliminate your external electronics, such as PLCs or computers.

**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 because motors move and rotate.**

## 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/Au Controllercoder Au86-118 by sending the i and v commands, then copy/paste here:

```
M>i
Excitron Corp 01/11/11 v4.12 Au86-118 Controllercodermotor +Vs=24.14

s baud echo In TQnum degC +Vs Steps/131K v Baud1
M 57.6K 1 00 028 021 24.14 - 0 19.2K

Revs Encdr Motor Position Shaft v Play home out
- 0060 00200 +00000000 012 0 000 0 0

P# D M G Acc a2 Brk Number Vsps Trq% msec rep pinmode f Fnum msps DU e ADC
01 W 1 G 0080 32 000 0000800 02000 060 -00000 001 0 00 0 0100 0100 1 0 0255
M>v
P# D M G Acc a2 Brk Number Vsps Trq msec rep pinmode f Fnum msps DU e ADC
01 C 1 G 0140 32 000 0000400 06000 060 -00200 001 0 00 0 0100 0100 1 0 0255 ← Driver Mode values
02 C 1 G 0114 32 000 0000400 02000 080 -00000 001 0 00 0 0100 0100 0 0 0255
03 C 1 G 0114 32 000 0001600 03000 090 +00200 001 0 00 0 0002 0250 0 0 0255
04 W 1 G 0114 32 000 0003000 01000 080 -00030 001 0 00 0 0003 0045 0 0 0255 ← Home values
05 C 1 G 0114 32 000 0000020 00600 080 -00200 001 0 00 1 0002 0100 0 0 0255 ← Home values

06 C 1 G 0114 32 000 9999999 06000 080 -00200 001 2 12 0 0002 0100 0 0 0255 ← I=06 IN2
07 C 1 G 0114 32 000 0000000 03000 080 -00000 001 0 00 0 0002 0100 1 0 0255
08 C 1 G 0114 32 000 0000000 03000 080 -00000 001 0 00 0 0002 0100 1 0 0255
09 W 1 G 0114 32 000 9999999 01800 080 -00000 001 3 12 0 0002 0100 0 0 0255 ← I=09 IN3
10 W 1 G 0114 32 000 0000000 03000 080 -00000 001 0 00 0 0004 0050 1 0 0255
11 W 1 G 0114 32 000 0000000 03000 080 -00000 001 0 00 0 0004 0050 1 0 0255
12 C 1 G 0114 32 000 9999999 02000 060 +00200 001 4 12 0 0002 0100 1 0 0255 ← I=12 IN4
13 W 1 G 0114 32 000 0000000 03000 050 -00000 001 0 00 0 0004 0114 0 0 0300
14 W 1 G 0114 32 000 0000000 03000 052 -00000 001 0 00 0 0004 0114 0 0 0560
15 W 1 G 0114 32 000 9999999 04000 080 -00050 001 6 12 0 0002 0100 1 0 0255 ← I=15 IN6
16 W 1 G 0114 32 000 0000000 03000 080 -00000 001 0 00 0 0004 0050 1 0 0255
17 W 1 G 0114 32 000 0000000 03000 080 -00000 001 0 00 0 0004 0050 1 0 0255
18 C 1 G 0114 32 000 9999999 06000 008 -00400 001 7 12 0 0002 0100 1 0 0255 ← I=18 IN7
19 W 1 G 0114 32 000 0000000 03000 008 -00000 001 0 00 0 0004 0050 1 0 0255
20 W 1 G 0114 32 000 0000000 03000 008 -00000 001 0 00 0 0004 0050 1 0 0255
21 W 1 G 0114 32 000 0000800 02400 070 +00700 001 8 12 0 0002 0100 0 0 0255 ← I=21 IN8
22 W 1 G 0114 32 000 0000000 02400 075 -00000 001 0 00 0 0002 0100 0 0 0255
23 C 1 G 0114 32 000 0000800 02400 060 +00700 001 0 00 0 0002 0100 0 0 0255
24 C 1 G 0114 32 000 0000000 02400 082 -00000 001 0 00 0 0002 0100 0 0 0255
```

Extra line spacing is shown here for clarity. These Motion Profiles and pin numbers are the same for all Au Controllercoders. The far right column shows the special Motion Profile functions. Since the Switchbox has a pot for analog input IN2 and one for IN3, and by setting D=1, then IN4 is used for direction. This speed change on-the-fly works on any Motion Profile that has D=1.

Notice that in Profile 12, the same input pin IN4 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 IN4. Flipping the IN4 switch low (ON) triggers the 3 motion profiles, then when you flip the switch to high (OFF), the motor gracefully stops because of pin 4 mode 12. Same pin, but two different functions. Similarly for the other input pins. If you want to run a fixed distance, then make N = your step quantity and pinmode = 0 00, then when you flip the switch up and down, the motor runs the number of steps you specified.

## Errata and notes

- V4.12 Added notes for Input pins triggering Input Profile Mode on page 6 & 10.
- v4.12 Fixed baud display and added all bauds to command line. Corrected torque values on pg10.
- v4.11 Fixed issue in v4.06-4.09 where saving g or S value (via U command) falsely returned s. G had no issue.
- v4.10 Fixed internal TQnum value--resulting in low torque for larger motors, now this value is precise.
- v4.09 Enhanced Encoder, new commands ( , ), h. Fixed an issue that the Output OUT5 was cleared if changing to a different Motion Profile, workaround is to set c>o=1 then do not change to another Motion Profile. Energy E value =017 as default.
- v4.05: fixed some internal firmware issues only.
- 1<sup>st</sup> release, no errata