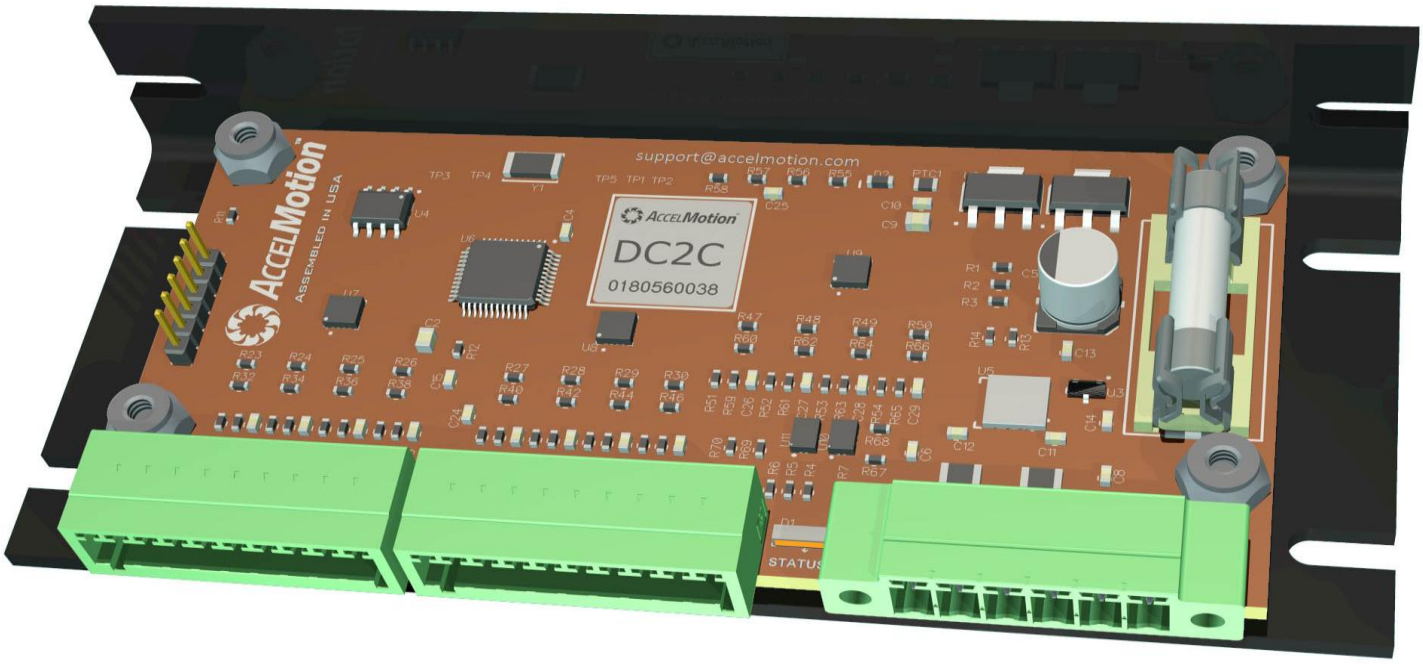
DC2C
USER GUIDE
Step Motor Driver/Controller Module



AccelMotion

DC2C Features

The **DC2C** is a small powerful member of the DC-series of AccelMotion step motor driver/controllers. It can operate a hybrid stepper motor at up to 2A max current per phase, under control of a host computer system or PLC. All DC-series controllers can also operate autonomously, with no need for an external host, when programmed with command instructions saved to non-volatile memory.

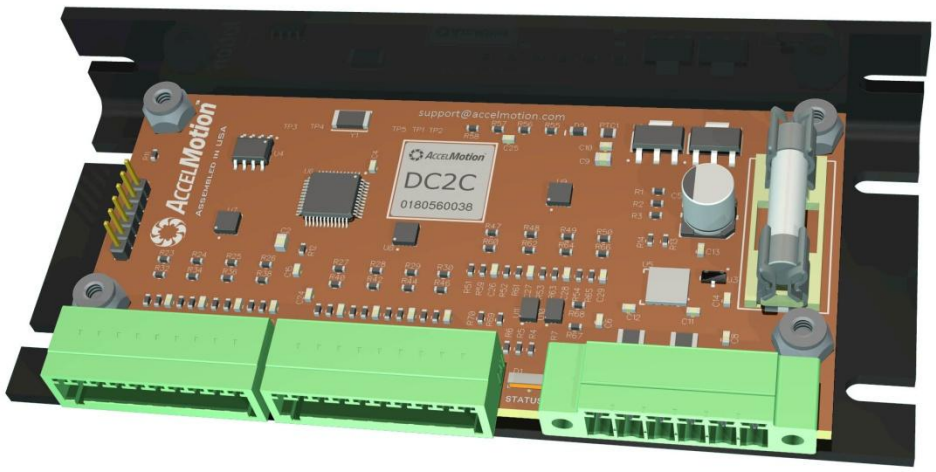
The **DC2C** features**:**

- Step Motor Drive with phase current up to **2A** max current per phase
  - Programmable Run/ Hold Current settings; reduces idle heat and power
  - Overcurrent detection and protection
- **Single supply voltage**, from 10 to 40VDC
  - built-in fuse protection – replaceable
  - built-in VMM and VIO voltage monitoring, via System Status Command
- Temperature sensor, for thermal characterization and real-time reporting
- 40 motion, programming, and configuration commands for powerful control and programming
- **Programmable microstepping** resolutions: Full, ½ , ¼, ⅛, 1/16 and 1/32 step modes
- Step speed programmable from 1 to 32,000 steps per second, and "divide-by" ratios up to 1/255
- **Direct Control Mode** supports host-based control; for systems integrated with external controllers
- **Program Mode** allows autonomous operation; no external host communications required
  - **Automatic Program Mode** allows program to start at power-on automatically
- **Twelve user-configurable input ports** – each can be:
  - a general-purpose programmable user input – at IO voltages of from 5 to 36VDC
  - optionally assignable to standard fixed functions, including:
    - Limits + and - sensors
    - Home sensor
    - Go and Stop switches
    - Jog + and – switches (with selectable jog speeds)
- **Four general-purpose digital output ports** under program control
  - each capable of sinking **1A of current per output at up to 36V**
  - optionally drive standard STEP/DIRECTION external driver controls to two outputs
- Communications via AccelMotion Comms Bus – recommended adapter: AccelMotion **CI-200**
  - Physical: RS485 (RS422 voltage-compliant), supports up to 64 modules on one interface
  - Fast Operation at 115,200 baud default speed
- **Small size** – easily mounted to chassis, equipment box, or other surface via mounting plate
- System Status Command displays current driver temperature, VMM voltage and VIO voltage
- System Status Faults Command tracks and displays failures to host
  - **driver overtemp, VMM overvoltage, VIO overvoltage and driver overcurrent**
- System Settings allow:
  - modification of **polarity** of home and limits
  - create external STEP and DIRECTION outputs to drive high output or duplicated driver
- **Status LED** provides indication of power status, motion, program operation, and faults

Included in the Box

DC2C Driver/Controller

Mating Connectors



Optional Accessories

Communications Interface: order CI-200

The **CI-200** Communications Interface provides an easy and direct connection to an AccelMotion Communications Bus such as used on the DC2C, via USB. Drivers for Windows, Mac, and Linux are available and provide computer access to the **DC2C** Command Line Interface (CLI).

Figure 1 - CI-200 Communications Interface for DC2C

Replacement Mating Connectors

Consult with Sales at AccelMotion for extra sets of mating connectors.

Mating Connector (Plug) part numbers are:

- **J3 and J4 Phoenix 1881406**
- **J1 Phoenix 1847097**

Hardware and Connection Diagram

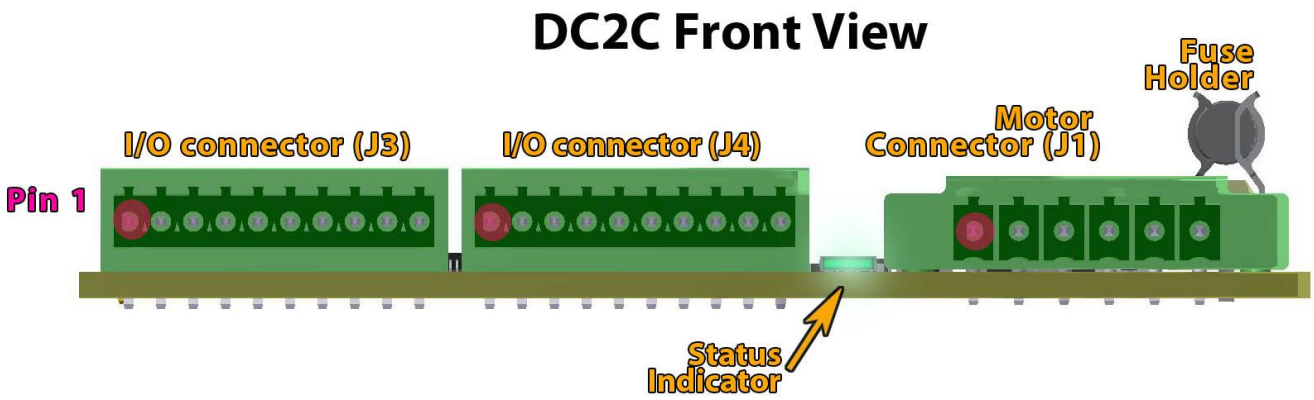


Figure 2 - DC2C Physical Layout

Pin #	Motor Connector (J1)	I/O Connector (J3)	I/O Connector (J4)
1	Motor Phase 1A	Input 1	Input 5
2	Motor Phase 1B	Input 2	Input 6
3	Motor Phase 2A	Input 3	Input 7
4	Motor Phase 2B	Input 4	Input 8
5	Ground	Output 1	Input 9
6	VMM (Main Supply)	Output 2	Input 10
7		Comms Data +	Input 11
8		Comms Data -	Input 12
9		VIO (I/O Supply)	Output 3
10		Ground	Output 4

Figure 5 - DC2C Connection Table

The DC2C makes all connections from the front side, with all connectors and the status display accessible from the front. The unit should be mounted to a chassis via the bottom or back sides.

The DC2C motor connector (J1) mates to a heavy-duty 6-pin plug (screw-type for 22 to 14 gauge wire). I/O Connectors (J3 and J4) utilize a more compact 10-pin connector (spring-type for 28 to 20 gauge).

Supply Fuse

The DC2C is protected by a 1A 5x20mm replaceable fuse, equipped on the left side of the unit.
Replacement part: **Littlefuse 0215001 5x20mm slow-blow**

Important Notes

Do not connect or disconnect the power/motor connector when power is applied

Overview

All AccelMotion DC-series controller/drivers provide efficient and powerful control of a stepper motor. Each unit consists of: a controller section which executes motion commands; a driver section, which is tailored to the specific drive power level of that model; and the Communications Bus which connects the controller to an external computer. The communications interface is used for programming and debug, but can be used to control or query the controller directly from a controlling host.

Each controller contains storage for the operating parameters of the controller and driver, as well as optional control programs. During operation, the parameters and programs are stored in main memory (RAM); all functions use these parameters, and programs can be run from that main program memory. Programs and Parameters can be permanently stored by transferring them into internal non-volatile (NV) memory, so that they are always available – see later sections on Programming & Program Mode.

In **Program Mode**, the controller executes commands in line-number order, but control is be steered by conditional branches based on input port conditions (see **L** Command), such as from switches, sensors, and controller/computer output signals of various voltage levels. In **Immediate Mode** an external host commands the controller, and all conditional and mathematical programming occurs in the host.

The DC2C provides twelve general purpose input ports, but additional functionality can be attributed to any of these ports, including **Go**, **Stop**, **Limits**, **Jog**, and **Home** – see **U** (Set Port) Command. User inputs ports can be configured to external voltage levels by connecting an external supply to the VIO pin.

Four programmable output ports are provided to drive logic levels or external devices (relays, etc).

The driver section provides the power to drive a stepper motor to the step location requested by the controller. The drive power level is programmable: Run current is used when moving, transitioning automatically to Hold current once movement is ended. Stepping parameters, such as microstepping mode, velocity, acceleration, etc. determines steps/revolution, speed, locations, etc. See the *Parameters & Defaults* section for more information on parameters and defaults for command types; and the *Stepping Motor* appendix about how motors and drivers are rated and controlled.

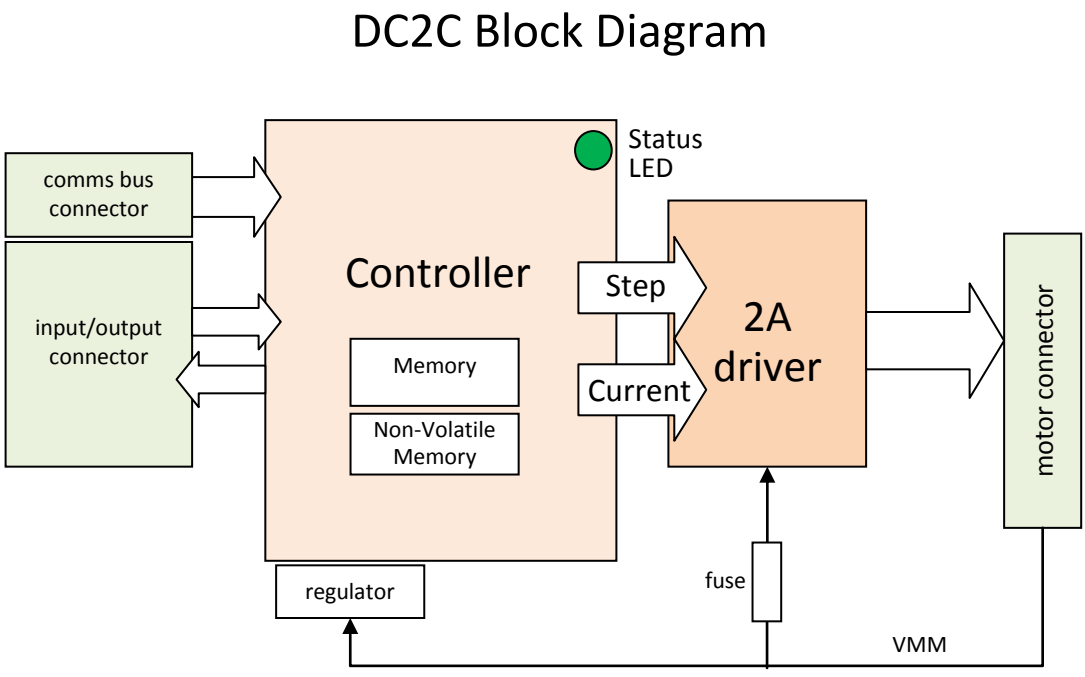


Figure 3 - DC2C Functional Block Diagram

Operational Modes

All AccelMotion DC-series controller/drivers can be operated one of two ways: **Immediate Mode**, where commands are provided by a host (via the communications interface) and are executed in real time; and **Program Mode**, where commands stored internally are executed programmatically, without a host.

Immediate/Direct Mode operates with commands from an external computer, PLC or other controller, and is used with when control from a high-level host is required. This mode can address one or many units (axes), and is needed where unified control of multiple axes is required. Commands and responses come through the Communications Bus and are executed in real time. Some commands are unavailable in Immediate Mode.

Program Mode executes commands directly from internal memory, using programmed command and parameter values and making conditional decisions based on its Input Ports (see "L" Command). **Programs Mode** can be started from **Immediate Mode**, or by fixed-function **GO** input, or automatically started on power-up (see **Automatic Program Mode**, below).

Once program execution has been started, the controller stays in **Program Mode** and executes commands in program order until the program ends or operation is stopped externally, via either the <**ESC**> character or the SOFT-STOP fixed port function. During **Program Mode** the communications port is ignored except for the <**ESC**> character.

Programs are stored internally and operate from main memory, and can be moved permanently into non-volatile (NV) memory via the **S** command – NV program memory is recalled back to main memory at each system reset or power-up, and also by the **C** Command.

Automatic Program Mode executes a stored program at power-up (or **^C** reset); this occurs when a execution branch is placed at location 192 – see **G** (Go/Branch) Command for more data.

Communication Modes

All DC-series controller/drivers are programmed and can be controlled via the AM Communications Bus. Two control/communications methods are available: **Single-Axis Mode** and **Multi-Axis Mode**.

Single-Axis Mode provides an easy way to control a single DC-series controller from a host PC, and has advantages: addressing/naming isn't required, so axis name prefixes aren't required for each command; and a greater range of commands is supported.

Naming (required for Multi-Axis Mode) does require the use of Single-Axis Mode, at least once.

Multi-Axis Mode allows multiple DC-series controller/drivers to be commanded from the same AM Comms Bus, allowing one host to control multiple axes. In this mode each controller must have an axis name; *each command must have an axis name prefixed to the command string to identify the target controller*. Some commands are unavailable in Multi-Axis mode. Multi-Axis mode can be used for single-axis systems, but Single-Axis Mode is easier (see above).

AM Communications Bus

The AM Communications Bus operates in a Serial Multidrop mode, using RS485 signaling and voltages.

The AM Bus operates at a standard speed of **115,200 baud, 8-bit data, 1 stop bit, and no parity**; the communications driver and all terminal or script software must be set to those settings for correct operation with any DC-series unit. Even though DC-series controllers are compatible with Advanced Micro Systems DCB- or MAX-series controllers command and memory structure, they do not use compatible communications bus standards (9600 baud).

Parameters & Defaults

Parameters are characteristics that the controller and driver use to operate the step motor, such as Velocity, Acceleration, Initial Velocity, etc., as well as hardware settings such as Input Port assignments, communications settings, etc. These parameters are located in main memory and can be modified via associated commands like **V**, **K**, **I**, etc.

The Examine command (**X**) displays the current parameter values to the console, in **Single-Axis** mode.

The operating parameter values are retrieved into main memory at every power-up or reset event, from non-volatile (NV) memory; it can be thought of as retrieving "stored settings". Those parameter values stay in main memory unless further modified by a command, or until power-down.

Parameters can be stored into non-volatile memory as defaults using the "S0" command.

Figure 4 - List of Commands with Parameters, showing Factory Defaultsshows the list of parameter command codes and descriptions, and the value the parameter will have from the factory or if it is reset to factory default values. The upper section shows motion parameters and the lower section shows other setup commands for communications and I/O.

Parameter Command Code	Command Parameter Description	Factory Default Value
K (Ramp Slope)	Acceleration/Ramp Slope (ramp up/ramp down)	5/3
I (Initial Velocity)	Initial Velocity (steps/sec)	400/1
B	Jog Speed (steps/sec)	400
V	Slew Velocity (steps/sec)	3000/1
Y	Hold/Run Current % (% of 2A full-power)	10/40
E	Settling Time Delay (in milliseconds x 10)	100 (1 second)
D	Speed Divider	1
H	Resolution Mode	1
U	Input port assignment (for I1 – I12)	1 1 1 1 1 1 1 1 1 1 1 1
T	Echo Mode	1
p	Homing Switch Polarity	0 (normally-open)
N	Axis Name	*blank**

** Programmed only by Ctrl-N naming*

Figure 4 - List of Commands with Parameters, showing Factory Defaults

See the *Commands* Section for more information regarding each parameter as well as how to store defaults into non-volatile memory using the **S** command, and recover defaults with the **C** command

Quick Start

This outlines a few steps required to make a **DC2C** operational:

1. Connect the **main power supply** to connector J1 pins 6 and 5, with input voltage in the range from 12V to 40VDC. Connect the positive to J1 Pin 6 and ground to Pin 5. The power supply must be capable of providing 1.5A of peak current or more. See *Power Supply* section for more details.
2. With power off, connect an appropriate size **stepper motor** to the four motor connections of J1, as shown in the *Hardware and Connection Diagram* section. Review *Stepping Motors and Motor Connection* section to determine your motor wiring, and be sure to insulate all motor leads (and unused leads) to prevent shorts to ground or to other motor leads.
3. Wire the D+ and D- **communications signals** from a RS485 Serial Interface into J3 pins 7 and 8; *we recommend the AccelMotion CI-200 model*. Make a connection between the GND of the DC2C (pin 10) and the GND terminal of the serial interface, to provide a shared ground to all devices (refer to *Grounding* section). If the CI-200 serial interface is used, then plug it into the USB port of the host computer, and ensure the required driver is installed correctly – see CI-200 manual for details.
4. **Start a terminal application** on your computer; it can be any program that allows you to send data through a serial connection; for example TeraTerm, Console, etc. We provide the free **AccelCom** software on the AccelMotion website, which provides an on-screen terminal with specialized buttons for Mode Selection, I/O, etc. Open the configuration settings of your terminal and select the virtual COM port to which the CI-200 is connected, and assure that the baud rate of 115,200 baud is set (AccelCom baudrate is preset to the correct value automatically).
5. **Apply power** to the DC2C. The green LED will flicker slowly to indicate operation.
6. Type <**space**> to connect in **Single-Axis Mode**. The controller will respond with its ID, which indicates that communication is established, and will appear as: `"AccelMotion DC2C Vx"`
7. To verify motion functionality, type a simple **motion command**: for example, **+400** will move the motor 400 steps in positive/plus direction. Command entry will not echo as you type, but at the end of the line the entire command will be echoed before the response (if any).

Type	Echo/Response	Remark
<space>	AccelMotion DC2C V3.01	space starts connection and returns ID
+400 ↵	+400	motor moves 400 steps, in + direction

8. Now, set safe motor **hold current** and **run current** settings, using the **Y command**. Make sure all **Y** command settings are compliant with your motor specs, that the current does not exceed the current rating of the motor, and that overall current will not overheat motor or driver.
9. To save these settings permanently into non-volatile memory (data restored when unit is powered up), type "S0" followed by enter↵. See **S** and **C** command reference sections for more information.

Important Notes

Do not connect or disconnect the motor when power is applied

The power supply voltage, including ripple and line voltage fluctuations, must not exceed 40VDC or be less than 10VDC

Drive Settings must be compatible with the selected motor specifications

Review and Understand the use of the "Y" Current command, to limit power consumption and heat in the driver and motor

Operating the DC2C

Entering and Using Commands

Commands are represented as strings of text, and are input through the command-line interface (CLI) of the controller. Commands can be submitted via the communications port and executed in real-time, or transmitted to, and stored inside, the controller for later use.

Command structure and data fields are shown for each command in the *Commands* section of this guide. First is the command character, followed by its data fields; these fields consist of decimal data values separated by spaces, and some fields are optional. For most commands, if the command is executed without a data field then that command is interpreted as a query of the parameter value. That queried parameter value is returned to the host via the communications channel as a response, for *example: "*V1000*" sets the velocity parameter to 1000 steps-per-second, but the command "*V*" returns the current value (1000) back to the host.*

All command strings terminate with a Carriage-Return (<**CR**>) character, generated real-time on a keyboard with **Enter↵** key, and from a script or application with a **^M** (control-M) or hex #0D character.

Command Highlights:

- **Only one command may be entered per line**
- **Each Command entry is terminated by Enter (from keyboard) or ^M (from script)**
- The command line may be edited using backspace, as characters are typed
- A line entry may be canceled using <**ESC**>
- A space is optional between the command and first parameter value
- A space or comma must be used to separate parameters with two parameter commands

Basic Communications

To establish first communications a host must be connected to the DC-series controller via an RS-485 communications adapter (such as the AccelMotion CI-series adapter). See *Communications* section for more information on RS-485 communications adapters and wiring.

Most users start in **Single-Axis Mode**; this allows Axis Naming, program entry, quick testing, and debugging. Use a 'terminal window' to access the command interface – any terminal software will work as long as it allows for sending and receiving data through a COM port. AccelMotion provides a free terminal called **AccelCom**, but many terminal applications exist including TeraTerm, Console, etc.

Immediate Mode (host mode) start-up

Follow these steps:

1. Power ON, then type <**space**> (causes **Immediate Single-Axis** connection to controller)
2. Type the command: "**R-1000**" and terminate with **Enter↵** (**<CR>**)
   Characters are not echoed as you type; the entire line is echoed after the terminator ↵ is sent
3. The motor should move a small amount (1000 steps, in the negative direction)
4. Type "**Z ↵**". Z command responds with the axis position (-1000)
5. Type "**{ ↵**". Brace command responds with the current driver temp in C

Programming

Program Entry Example

The above examples show commands in **Immediate Mode**. The following are examples of program sequences entered into main memory, to be executed in **Program Mode**. Enter these commands after signing-on with <space>. As commands are entered, the interface will echo the program line number into which the next command (and parameters) will reside – some commands take more space than others, and this space is shown as *Size* in each command's reference section.

Type	Echo/Response		Remark
P0 ↲		P0	**Enter Program Entry Mode** at location 0 (zero)
O0 ↲	0	O0	Set step counter to 0 (← Command O, value zero)
+1000 ↲	5	+1000	Move 1000 steps in "+" direction
W0 ↲	10	W0	Wait until motion is complete
-1000	13	-1000	Move 1000 steps in "-" direction
W0 ↲	18	W0	Wait until motion is complete
Z ↲	21	Z	Respond with current step counter position
G5 ↲	22	G5	Go to line 5
P ↲	25	P	**Exits Program Entry Mode** (P, same as P0)
			does not enter data or code into next line (line 25)

Now, use "Q" command to display the stored program for review - type "Q0" to start the listing at 0. The screen should now display the lines previously entered, as they are stored in main memory. This program stays present in memory until the controller is reset or power is lost. Once programs are debugged and finalized, transfer them to non-volatile (NV) memory with the S Command.

Program Execution Example – entering Program Mode

Type	Remark
G0 ↲	Enter **Program Mode** and start executing the program at specified location 0 (zero).
Notes:	*Program mode can be terminated at any time by hitting <ESC> (escape) key*
	Trace function allows the real-time display of instructions executed (see "G" command)

In Program Mode, all motion commands are automatically stalled (waiting) while any previous motion is still in progress – non-motion commands are not stalled automatically behind a motion command; so, review usage of the W0 Command for setting execution timing – use W0 prior to any command that should wait for completion of a previous motion command

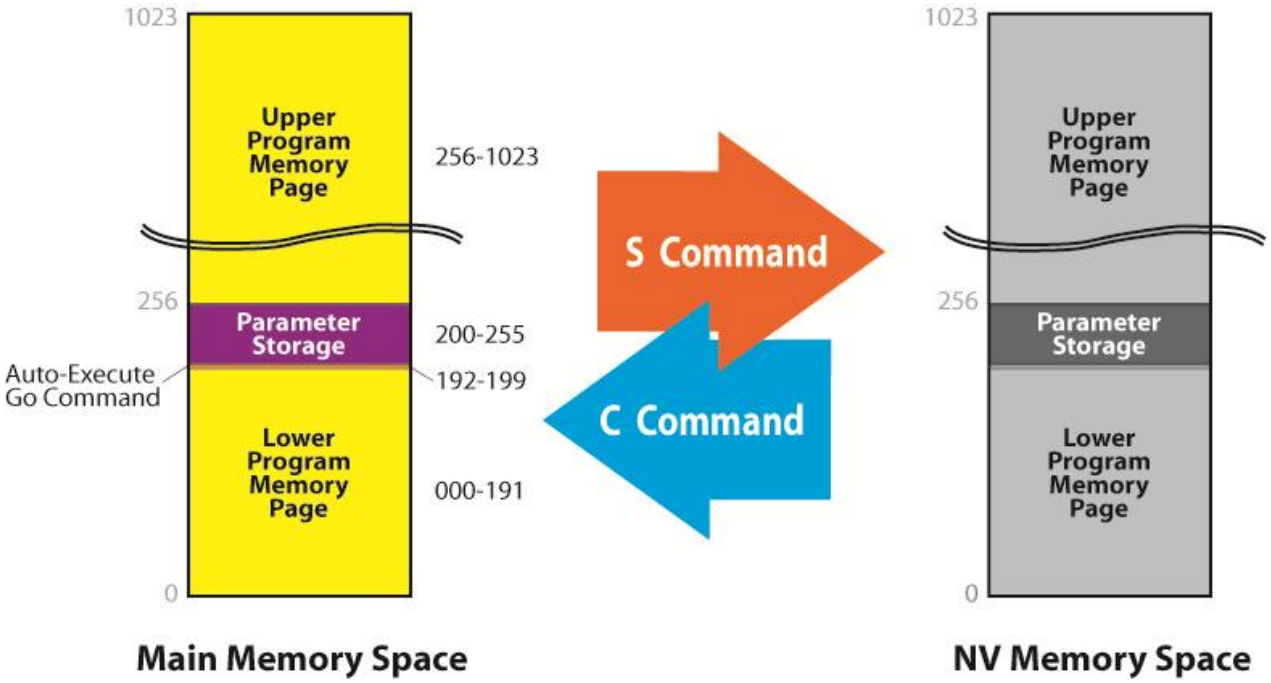
Command Entry Mode Highlights:

- Characters are echoed on a **line-by-line** basis, and only one command is allowed per line
- As commands are entered, the <**backspace**> key will completely reset the current line contents and repeat the program line number – also if illegal command character is used
- Command is terminated by Enter (from keyboard) or ^M (from script)
- A space is optional between the command and first parameter value, but must be used to separate parameters in multiple parameter commands
- Once a program is loaded, it can be saved to non-volatile memory using the "S1" command so that it's retained permanently, even after powering down
- Make sure that command echoes and responses are received completely for each command before issuing further commands

Memory Map

The DC-series controller main memory is organized to be compatible with previous AMS (Advanced Micro Systems) controllers. Programs can start at location 0, and can be located in two distinct pages of memory, shown below. Parameters also live in this main memory space, but are modified only by specific commands that set and retrieve these values.

Memory Space starts with the first program memory page, from locations 0 to 191. Note instructions take up 3 locations on average, so lower space can hold around 55 instructions, and the upper page around 75; more motion commands decrease the instructions per space (as they take 5 locations each). Insure that all programs start and end within the yellow 'program space' areas and do not stretch into Parameter Storage space or beyond the program space limit (1023).



Note that if using **Automatic Program Mode** operation, the Auto-Execute location (192) should contain a **"G" Go/Branch Command**, which steers automatic execution to the desired starting routine location. This space at 192 should only contain the Go command 'vector' to the starting program – note that locations 200 to 255 contain only Parameter Storage, and it is illegal to place programs in that space.

Multi-Axis Considerations

Axis Naming

Multi-Axis Mode allows control of more than one unit (axis) on the same Communications Bus. When using this mode each axis must be assigned a unique address, or "name", so that the target axis for each command can be explicitly identified. Axis naming is not required if *only* Single-Axis mode is used.

To give the unit a name, send a `^N` character (type <Ctrl> and "N" at the same time). When requested, type a single valid axis-name character (Any character A-Z, or a-z), then type "y" (lowercase) to save the new axis name. This naming can only be done in *Single-Axis* and *Immediate Mode*. Utilize the X command to verify the action - it lists the axis name among the other parameters valid for this unit.

If the axis is named while using AccelCom Terminal, the user need only enter the axis name – all other communications are hidden.

Multi-Axis Start-Up

Once all devices have unique address 'names' and are wired as described in the Multi-Axis Wiring section, the system hardware is ready to operate in Multi-Axis Mode.

In Multi-Axis mode every command is addressed by prepending the name of the addressed axis to the command. For example: if the user intends to issue the command `M1000` to axis A, the command would be: `AM1000`. The only exceptions to this are Reset (^C) and <ESC>, which are broadcast commands.

Each controller unit residing on the bus in Multi-Axis Mode acts as a listener, waiting for its specific address (name) character. Once the programmed name is received, the remainder of the command string is received until the terminator (`<CR>` or ^M) is accepted; the unit interprets that command and the command string is echoed to the host. Once the final characters of the echo (`<CR><LF>`) have been received, the host may send another addressed command to the same or any other axis/unit.

Try a few examples in Multi-Axis Mode, (even with only one unit where Multi-Axis isn't needed):

- Power up; start with Host and Communications Adapter, then the controller/driver named "A"
- In the terminal window type **^P** (<**Ctrl**> and "**P**" at the same time - ASCII Code 10 hex)
  All devices should now be in **Multi-Axis Mode** – no additional Enter↵ is needed, and no response is returned
- Type the command: "**R-1000**" and terminate the command with <Enter> Key ↵
  Note, that as you type no characters are echoed until the end, where the entire line is echoed
- The motor should move a small amount (1000 steps in the negative direction)
- Type the **Z** command, and enter↵. This command responds with the axis position (-1000)

Multi-Axis Communication Note

The DC2C incorporates a buffered UART input, but because motion control is of the highest priority, processing of received information may be delayed if commands are sent while stepping at a very fast rate. The host should always monitor echoed data to avoid multiple command UART overrun.

External Host Control Note:

Make sure that command echoes and responses are received completely for each command before issuing further commands.

Hardware

Power Supply Selection and Connection

The DC2C is powered from a single DC power supply (PS). The power supply is connected via the J1 Connector, pins 5 and 6. The input voltage must be in the range of 10VDC to 40VDC.

Pin	Function
J1 - pin 6	VMM (Motor Voltage)
J1 - pin 5	Ground

In general, unregulated DC (or linear-regulated) power supplies are best suited for stepper motor applications. Switching power supplies are cost-efficient, however their ability to provide surge currents can be limited, and may suffer from drop-outs in current during peak operations.

A single power supply can be used for multiple controllers provided that the total maximum peak current is provided for and the PS voltage is within the specified acceptable voltage range for all units. Each unit should use separate power and ground leads that connect directly to the output terminal of the shared power supply. Individual supplies for each unit/axis can be used, as long as the ground connections for all shared devices are common (see Multi-Axis Communication).

For 2A of max phase current, we recommend a supply with output power of 40W (Watts) minimum.

The current capability (in Amps) of any specific power supply will depend on its output voltage – this can be calculated by dividing the total wattage by the output voltage, as shown in these examples:

PS Power	PS Voltage chosen	PS Current Required, at chosen voltage
40W	40V	40W / 40V = **1 Amp @ 40V**
40W	36V	40W / 36V = **1.11 A @ 36V**
40W	24V	40W / 24V = **1.66 A @ 24V**
40W	12V	40W / 12V = **3.33 A @ 12V**

Important Notes

Do not connect or disconnect the motor leads or connector while power is applied

The power supply voltage, including ripple and line voltage fluctuations, must not exceed 40VDC or be less than 10VDC

Wire size between the power source and the driver(s) should be at least 18 gauge AWG. Longer distances between the power supply and driver should use a heavier gauge

When powering multiple units with a common power supply, use individual supply leads, and connect them all together directly at the power supply connection points

Select a common supply that can provide the total max peak supply current for all units

Stepping Motors and Motor Connection

The DC2C is a bipolar chopping-style stepper driver that works with both bipolar and unipolar motors, i.e. 8-, 4- and 6-lead motors. It is also possible to drive half of a 6-lead center tapped motor with the DC2C, however performance may be compromised. To avoid unstable chopping conditions and to provide a higher speed/performance ratio, a motor with a low winding inductance is preferred.

Motor Connection

The J1 connector also provides connections between the DC2C and a Stepper Motor.

Pin #	Description	Function
1, 2	1A, 1B	Phase 1 of the Stepping Motor – connect between Pin 3 and 4 of J1.
3, 4	2A, 2B	Phase 2 of the Stepping Motor – connect between Pin 5 and 6 of J1.

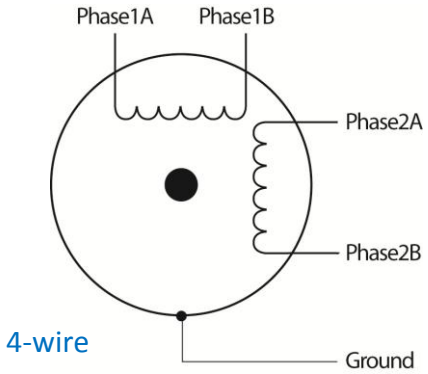
Drive Current

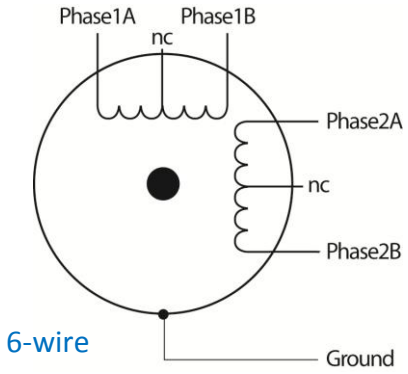
The ideal current for a given motor is based on the specific characteristics of the motor and the requirements of the application. As a result, establishing the correct current is often determined empirically. Insufficient current will result in inadequate torque and under-utilization of the motor. Excessive current can cause high-speed torque ripple, resulting in stalling or pole slippage, overheating of the motor, and general inefficiency of the system. The current can be set using the "Y" Command– see that section for details.

Higher drive currents and higher duty cycles can create higher heat in the driver and motor. The driver temperature cannot exceed 100C; this can be monitored with the **{** Command (System Status).

Step Motor Configurations

These days nearly all stepper motors are of the 4-wire bipolar variety, with two isolated windings driven through a pair of wires; connect each phase of the motor to a phase of the driver, as shown on left.





Some steppers have center-tapped windings (6-leads); We recommend a configuration that ignores the center-taps; connect each outside phase lead to a phase of the driver as shown on right. For 8-lead steppers, please consult the stepper motor white paper on our website, at AccelMotion.com

Direction Setting

The physical direction of the motor with respect to the controller's 'positive' direction will depend on the connection of the motor windings. To reverse the direction of the motor and make it match the controller, simply swap the connections on phase 1, or swap the connection on phase 2, or swap the phase pairs with each other. Example of swapping phase 1: swap 1A and 1B so that 1A on the driver goes to 1B on the motor, and 1B on the driver does to 1A on the motor.

Important Notes

Do not connect or disconnect the motor leads of connector when power is applied

Select the stepper motor for your application first based on the required torque, and then determine the required drive current as a maximum. Most users finalize their current settings empirically, with the final physical setup in-place

When using a 6-lead motor be sure to insulate unused wires.

There is no inherently 'positive' direction of a stepper motor

System Fault Indicators and Status LED

The DC2C validates correct operating conditions continuously while powered. When conditions are incorrect, the system detects and reports any and all faulty condition as the **System Fault**.

The conditions checked are:

- **fuse condition** – error when blown (open)
- **over temp** – error when internal temperature is above the maximum
- **over voltage** (VMM) – error when main supply voltage is above the maximum allowed
- **over voltage** (VIO) – error when VIO supply voltage is above the maximum allowed
- **over current** – error when driver reports overcurrent failure; reasons are:
  - a short from ground to some point on the module main board
  - a short between any of the motor phase leads
  - a short between any of the motor phase leads and the ground or power inputs

Each of these conditions can be queried real-time by using the System Fault command "**}**" (close brace). Each condition is represented in the command response by its code letter or and separated by slashes "/". An example query to the system during a fuse failure would appear as: } F/-/-/-/- . See more information on the System Fault command and responses in the detailed command reference section.

Status LED

The DC2C indicates status real-time via a multicolor LED status indicator.

The status LED flashes green when the unit is powered. This flashing "heartbeat" is visible and appears as a short burst when no motion is occurring, and when the motor is moving the LED flashes at a rate that is based on velocity. This function is meant to visibly indicate general motion in either direction.

If for any reason the indicator is illuminated RED or flashing YELLOW, the unit has sustained a **System Fault**. See the System Fault **}** Command for the fault types, reporting, etc.

During **Program Mode** operation (both manual and automatic start) the status LED will glow blue. Also, in **Immediate Mode** communications from the DC2C will be indicated by blue flashes. See Operational Modes section for more information on these modes.

The table below shows example status LED indicator colors seen and their implied modes:

Status Color	Indication
Black (no light)	Power not connected
Green short flashes	Power on, no motion no faults indicated
Green flashing	Power on, motion (flash rate from velocity) no faults
Blue	Power on, in **Program Mode** no faults
Blue on with green flashing	Power on, in **Program Mode** with motion no faults
Green flashing and Blue flashing	Power on, with communications to/from the host no faults
Red*	***System Fault*** multiple possible conditions
Red with* yellow flashing	***System Fault*** with motion still requested from host

* Critical failure – evaluate via command interface

Connectors

The DC2C requires three connectors for all its functions:

- A 6-pin Power/Motor Connector (J1)
  - connects the main motor voltage (VMM), which is used to drive the motor directly, but also powers all the other on-board functions
  - connects the four motor phase leads for any bipolar stepper motor up to 2A/phase
- A set of two control connectors (J3 and J4), together containing:
  - AM Communications Bus signals DATA+ and DATA-
  - Input/Output signals (twelve inputs and four outputs)
  - VIO input (I/O voltage for input thresholding and output pull-up)
  - Signal Ground

Each connector has physical keying to prevent the incorrect insertion of its mating connector. Each connector with signal numbering is shown in the Hardware and Connection Diagram section on page 6.

Many of the basic functions of the **DC2C** may be accomplished using only J1 and J3, as J3 contains connections for communications, four inputs, two outputs, VIO and ground – the connections were designed to simplify operation where possible.

Input / Output Subsystem

DC-series controllers feature input and output ports – inputs allow external signals to control the DC2C and the outputs allow DC2C to control external devices. These connections are industry-standard **"low-drive"** type, where the ACTIVE state of any input or output is low-voltage (ground), and any ports INACTIVE state is high-voltage (VIO level – either provided externally or automatically uses 5V).

This method allows easy connections to pushbuttons, limit switches, home switches, or proximity switches, which simply connect the input to ground when active. Outputs also can operate equipment such as solenoids, relays or indicators by connecting them between VIO and the chosen output; the output provides a connection to ground (up to the rated current)

Input Ports

VIO

All inputs signals are evaluated based on their voltage compared to the Voltage I/O (VIO) level – if the input voltage is over ½ of VIO then it is PASSIVE or OFF, and if under ½ of VIO then it is ACTIVE or ON.

VIO is also used to "pull-up" all output ports that are OFF or INACTIVE so they are compatible with external devices or PLCs that need a passive voltage provided.

If the user supplies VIO then it is used, but if VIO is not provided externally then a 5V level is used.

Input Port Electrical Configuration

All digital inputs are evaluated against ½ VIO voltage, to determine the actual digital value - above ½ VIO is INACTIVE and below ½ VIO is ACTIVE. This method supports a wide input voltage range for digital IO, yet provides noise immunity. An internal 50k Ohm pull-up resistor to VIO is provided to force inputs normally high (inactive), and must be pulled low by the external circuit to activate.

An input R-C filter is also included and provides about a 2kHz cut-off frequency.

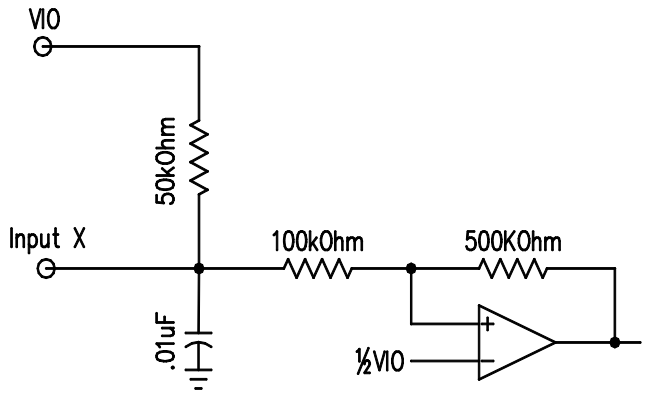


Figure 5 - Input Equivalent Circuit

The typical way to use an Input Port is a simple switch contact to Ground. Switch contacts for fixed functions of **Limits**, **Home**, **Jog**, **Go** and **Stop,** and general-purpose **User IO** inputs from control switches, can be supported in this manner. *Figure 6 - Example of Switch Contact Input* shows how such simple connections are made, for a passive contact to ground. This works because switch connections are low resistance and the input circuit has a 50kohm pull-up resistor; no other external circuit is needed. As long as the input voltage is lower than the threshold (½ VIO) the input is ACTIVE, and when it is above the threshold it is INACTIVE. If the port is configured as a fixed function, then that function is activated; if it's a User I/O, then the data value is set to '1' for all internal status values.

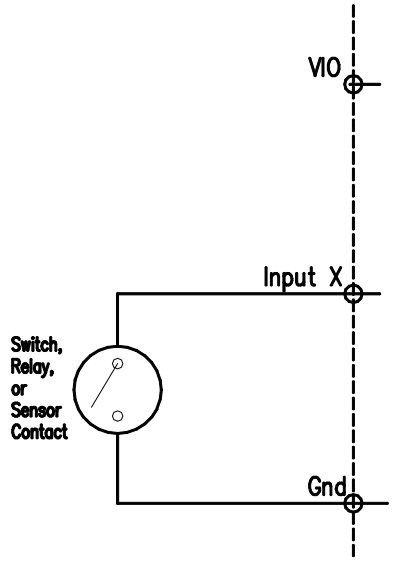


Figure 6 - Example of Switch Contact Input

When a DC2C input port is connected to an output from a computer, a PLC, or an active sensor, then if that device drives the input port to a higher I/O voltage level than 5V (12V, 24V, etc.) the input section will require the external VIO input (the internal default VIO value is 5V). In these cases, **connect the external I/O voltage supply to the VIO input** – the threshold circuit uses the higher of either 5V or the external pin value, and divides that voltage in half for the final VIO threshold voltage.

Figure 7 - Example Active Input using external VIO supply shows an example of an input port X driven from a computer or sensor output terminal, powered by 24V. Connect the external ground to the DC-series controller ground, and the external 24V supply to the VIO pin. This will set the input threshold to 12V (24V ÷ 2), thus will interpret voltages lower than 12V as ACTIVE, and greater to be INACTIVE.

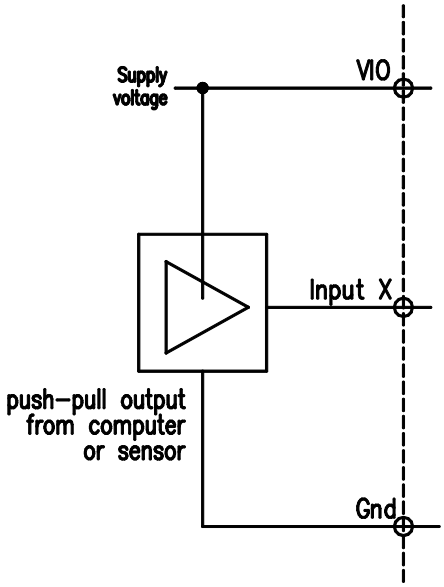


Figure 7 - Example Active Input using external VIO supply

Input Port Configuration – Fixed Function Assignment

Each digital Input Port can be configured for a **fixed function** from the following set; **Limit+ and -, Home, Jog+ and -, Go,** and **Stop**. See the **U** Command Section for detailed reference of each function.

Input ports automatically provide General-purpose Input data irrespective of additional fixed function.

The factory default setting for all Input Ports is **User Input** mode (no fixed function).

Below is the list and description of fixed functions available for each I/O port:

HOME

HOME is used to seek to a zero location within the allowed motion space. It operates via a unique switch closure, activated at the chosen 'home' location – DC-series controllers implement an algorithm to accurately position the step motor output to the same location in an accurate and repeatable way. See the "F" command for more reference data.

GO

The GO input signal is used to start a program located at internal program location 0 (zero), when asserted from INACTIVE to ACTIVE. Once Program Mode is entered the GO input does not have any further effect on a running program. If held will cause continuous program run.

ALL STOP

The ALL STOP input is used to stop all motion and end any program operation, when asserted. ALL STOP input implements an ALL STOP (|) command, so will utilize deceleration functions as it stops. The ALL STOP function is not provided as a primary safety device – always add an external emergency-stop switch to remove main power upon emergency conditions.

JOG+ and JOG-

Jog inputs are used to manually change the motor position, when the input is ACTIVE. Both jog inputs will stop any existing motion command and/or program operating when the input is activated and during activation, before manually moving in the requested direction.

LIMIT+ and LIMIT-

Limit inputs are used to protect physical systems by indicating to the controller when the mechanical system has reached a maximum or minimum physical limit, beyond which damage will occur. Limit inputs stop any pending motion, and will not start any new motion, in that direction when ACTIVE; example: LIMIT– will stop any pending motion in the negative direction and disallow any motion in the negative direction while ACTIVE. Motion is allowed in the opposite direction, so that systems can retract away from the limit switch.

AccelMotion recommends Limit Switches for protection of mechanical systems.

Fixed function assignments are part of the parameters of the system, and as such can be stored into non-volatile memory. All parameters are recovered from non-volatile memory into main memory on each power up or system reset event – therefore stored fixed function assignments are permanent.

Input Port Read

All input ports are readable in **Immediate Mode** by the A (Input Port Read – Decimal) command, which returns the combined value of all input ports, for use by external control scripts running in the host.

The A (Input Port Read – Decimal) **Command reads and returns the combined input port data irrespective of the function assigned to any input port** – even if functions such as HOME, GO, LIMIT, etc. are assigned to a port, the returned value from "A" command contains the raw port status value.

Input Port Conditional Branching

In **Program Mode** all Input Ports can be used for conditional branching via the L command. Examples are shown in the command reference for L Command.

The L (Loop Back) Command controls program branching with raw input port data irrespective of the function assigned to any input port – even if functions such as HOME, GO, LIMIT, etc. are assigned to a port, the L command operates on that raw port status (STOP and JOG cause exit from **Program Mode**, so are exceptions).

Input Port Highlights

- Input ports can be configured to drive various optional fixed functions (U command)
- Input ports can *always* be accessed via **L** Command in Program Mode and **A** and **N** Commands in Immediate/Direct Mode – *raw input data is available irrespective of fixed function assignments*
- Input ports can be used with switch/dry-contact devices without an external VIO supply
- Input ports can be configured to detect higher voltage input levels by connecting an external VIO supply to the VIO input – inputs are evaluated by a threshold of ½ VIO
- Dry-contact devices can operate correctly with external VIO supply connected – therefore, passive and active input ports can be used in the same system
- **All Input Ports must operate at the same VIO threshold, whether configured for optional fixed functions or as general-purpose program user inputs - there is only one shared VIO input pin.**
- **The ALL STOP function is not provided as a primary safety device – always add an external emergency-stop switch to remove main power upon emergency conditions.**

Output Ports

The DC-series controller provides digital output ports capable of driving external devices in active-low open-collector mode, at up to 1A current per output. The DC2C provides four output ports.

Output Port Electrical Configuration

Each output port can operate in one of two general modes: **Logic Mode**, and **Driver Mode**. Both modes are available with no internal configuration required:

Logic Mode – 5V Computer/Logic Voltage: when the controller output is connected to a computer or external logic input (that typically draws less than 10mA), a 10Kohm pull-up resistor pulls the output to VIO voltage in INACTIVE state and when ACTIVE the driver pulls the output to near zero volts. Connect the output port from the DC-series controller directly to the computer input or logic; no VIO connection is needed (VIO is supplied internally at 5V level).

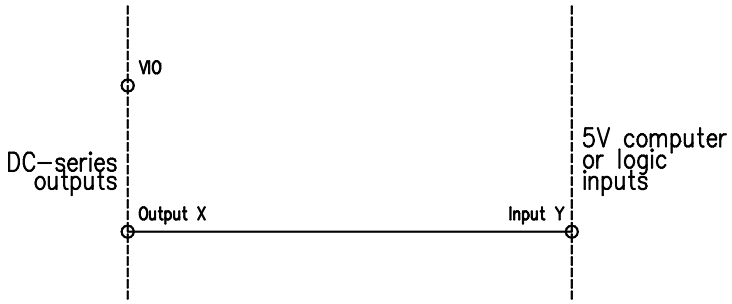


Figure 8 - Output port driving 5V computer/logic input

Logic Mode – Computer/Logic Voltage higher than 5V: if the computer or logic input requires a higher voltage logic signal, *then connect thehigh-level supply voltage to the VIO terminal*, so that the INACTIVE signal level will pull-up to the required higher-voltage level – the external supply voltage will be substituted for the internal 5V VIO, and provide the higher voltage logic signal.

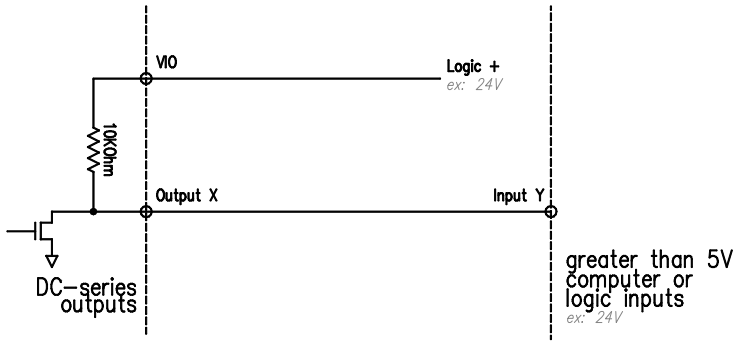


Figure 9 - Output Port driving high-voltage logic inputs (example: 24V)

Driver Mode: the output can be used to directly drive an external device actuated by current rather than voltage, such as a sensor, relay coil, indicator light, etc. In this mode, the output provides a path for sinking current to ground when ACTIVE. Connect the device between the VIO voltage value (but below the abs max allowed) and the output terminal; the output will switch between VIO (at low current) and ground (at up to 1A current). The power supply and the DC-series controller must share a common ground.

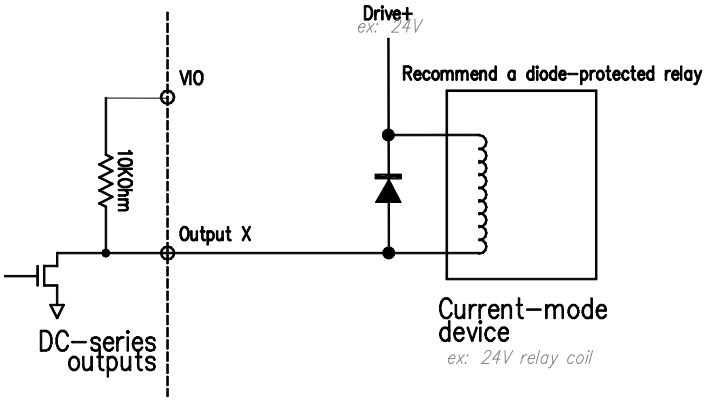


Figure 10 - Driver-Mode Output Port configuration (with diode protection)

Controlling Output Ports

Output ports are actuated via the "w" or "N" Commands, in **Immediate** and **Program Modes**.

The w command controls all outputs with a single data parameter; multiple outputs are updated at one time. The "w" command without a parameter will return the outputs value, in **Immediate Mode** only.

The N Command allows direct control of each output independently from the other outputs – each command contains the port and the value to be set on that port. The N command operates in **Immediate** and **Program Mode**. The "N" command without any parameters returns each ports bit value one at a time – where "w" returns the decimal value, "N" returns individual bit values (binary).

Output Port Highlights

- **Output ports are driven to GROUND when the value is "1", and pulled-up to VIO when "0"**
- Output ports can generate logic outputs at various voltage levels (based on VIO), and can also act as open-collector/open-drain driver for power elements, such as lights, motors, and relays (up to limit)
- Output ports can be controlled by the "N" and "w" commands, and can be read back with them

Important Notes

Because VIO affects both output and input ports, make sure that all input thresholds and all output pull-up levels are equal (or compatible)

Assure that all outputs connected will draw 1A or less from each output terminal

When driving inductive loads (relays, solenoids, motors), be sure to use diode-protected versions of these devices, or add diode protection as shown in Figure 10 - Driver-Mode Output Port configuration (with diode protection)

Communications

The AM Communications Bus provides a command link between host computers, controllers, or PLCs and one or more DC-series Controller/Driver units. The bus utilizes industry-standard RS485 voltages and signaling to connect the host and any single or multiple controller units. Figure 11 - AM Communications Bus Block Diagram shows the basic blocks used in communicating over the Comms Bus.

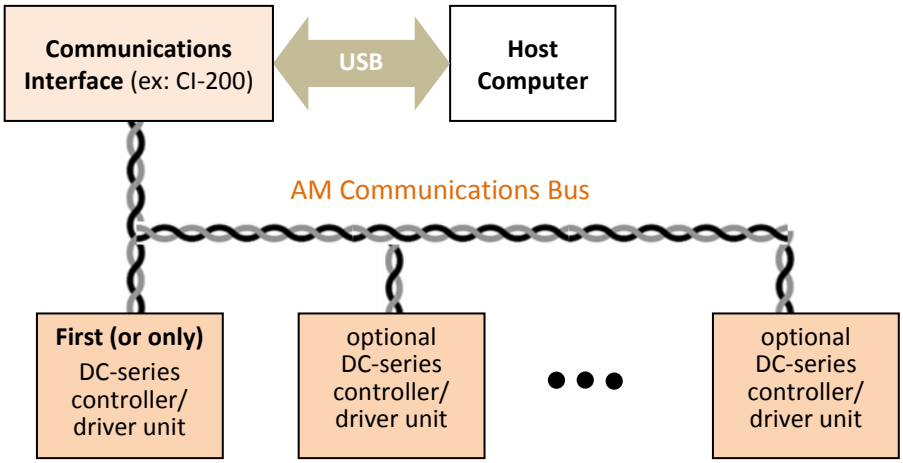


Figure 11 - AM Communications Bus Block Diagram

AccelMotion makes available the **CI-200 Communications Interface,** which interfaces standard computer hosts (via USB) and the AM Communications Bus. Other RS-485 adapters may be used as long as they convert to two-wire RS485, expose a ground reference pin, and include signal termination. Note: your product warranty may be at risk if a non-AM adapter causes electrical damage.

The AM Comms Bus operates serially in a standard multidrop mode (even with one unit), using RS485 signaling and voltages – see available web resources on communications for more info. Serial communication standards for the AM Comms Bus are: **115,200 baud, 8-bit data, 1 stop bit, and no parity**; terminal or scripting software and the communications interface must use those settings for correct operation with any DC-series unit. AccelMotion DC-series controllers are *not* compatible with Advanced Micro Systems DCB- or MAX-series controllers using 9600 baud.

Even when connected in single-axis mode, the host (master) must only transmit when the units (slaves) are not providing an echo or response message, therefore simple care must be taken with control scripts: make sure that echoes and responses are received completely before issuing subsequent commands. This rule is true even when communicating with a single unit, in Multi- or Single-Axis mode.

Communications for a single DC-Series Unit in Single-Axis Mode

Many applications can be accomplished with a single DC-series driver/controller, with no need for the complications of Multi-Axis Mode operation. These implementations require only a single triplet of wires between the RS485 communications interface and the single controller unit – these are the Data+ and Data– leads, and a shared Ground. Refer to Hardware section for the connection pin locations.

The next step is to connect to the unit in **Single-Axis Mode**; this utilizes the special signaling command <SPACE>. Once in Single-Axis Mode, all commands are addressed inherently to that single controller unit, with no need to add an address. See *Entering and Using Commands* section for start-up examples.

Using Multi-Axis (Party Line) Mode to Command Multiple Controller/Driver Units

Some applications may require multiple DC-series controllers to be commanded at one time from a single control host (computer/PLC). In these cases, **Multi-Axis Mode** can be used, via the AccelMotion Communications Bus, to address up to 64 controller/drivers individually via the same script/software.

First, to command multiple units, the system must necessarily utilize **Multi-Axis Mode**; each unit must be uniquely named in this situation. So, review the section called *Multi-Axis Considerations,* showing naming and other Multi-Axis concerns. *The naming process requires at least one session of Single-Axis operation, so that the unique name (address) can be entered into each DC-series unit, one at a time.*

The bus operates at a standard speed of **115,200 baud, 8-bit data, 1 stop bit, and no parity**; all terminal or script software must be set to those settings for correct operation with any DC-series units.

Communications Wiring

After naming, connect all units as shown below: connect all RS485+ nodes together and all RS485- nodes together, and connect all grounds from all units and the comms interface together. The wiring method can be simplified depending upon the distance between units and the electrical/magnetic environment. Two example wiring models are shown, below:

Short Bus Length - Simplified Wiring Model

When the location of units places them within a wire length of one meter or less, and the electrical and/or magnetic environment is not hostile, then the wiring model can be simplified. Examples are closed systems little to no high-power electrical equipment nearby (motors, solenoids, relays, etc.). In these cases, direct wiring with simple unshielded conductors, can work acceptably. *But, it is **strongly recommended that all systems utilize the twisted-pair shielded wiring model,** to assure the highest quality connection and lowest exposure to induced electrical currents from equipment or other noise sources.*

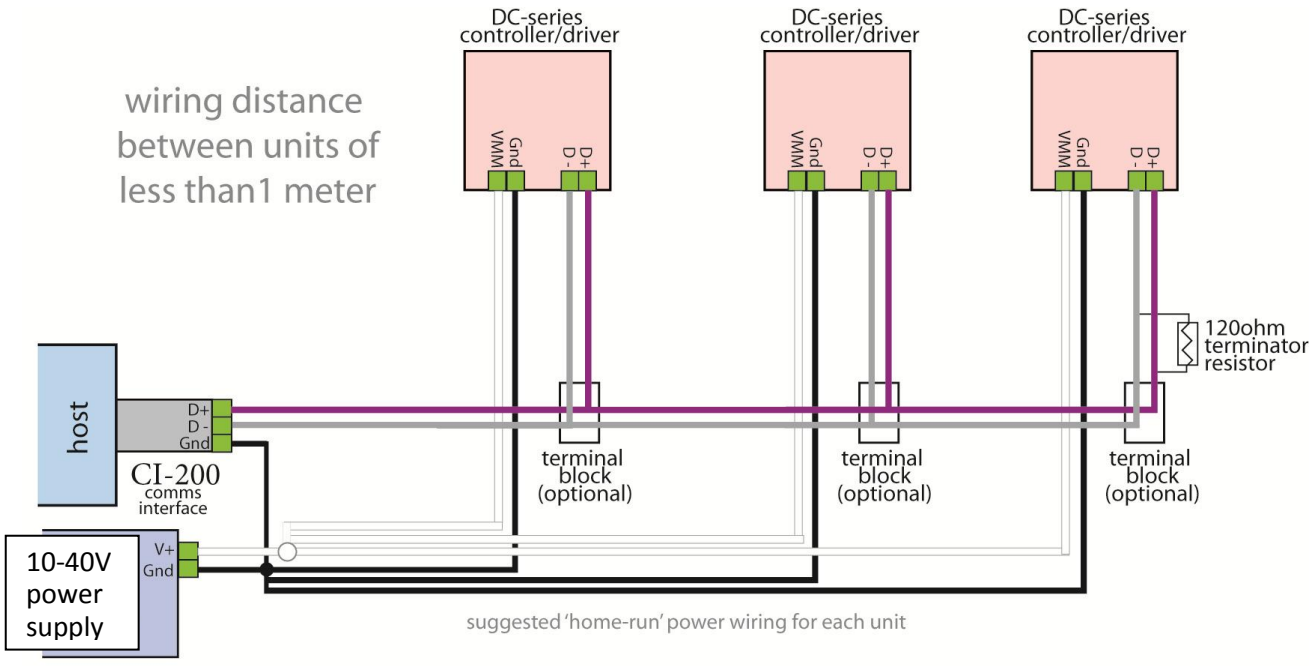


Figure 12 - Simplified short bus wiring method (not recommended)

High-Reliability Method - Twisted-Pair and Shielded Wiring

When units are more than one meter from each other or the host interface, it is recommended to make connections with twisted-pair wiring, in a shielded cable bundle. This method is similar to Ethernet wiring and can use the same wire type (four twisted pairs), or a single-pair cable.

The communication interface and each controller unit should be connected to the main twisted-pair wiring bus with connections as short as possible, with all D+ connections bussed to one conductor and D- connections to the other conductor; this method minimizes electrostatic and electromagnetic interference and noise. Connect wiring shield to ground on the *host end only*.

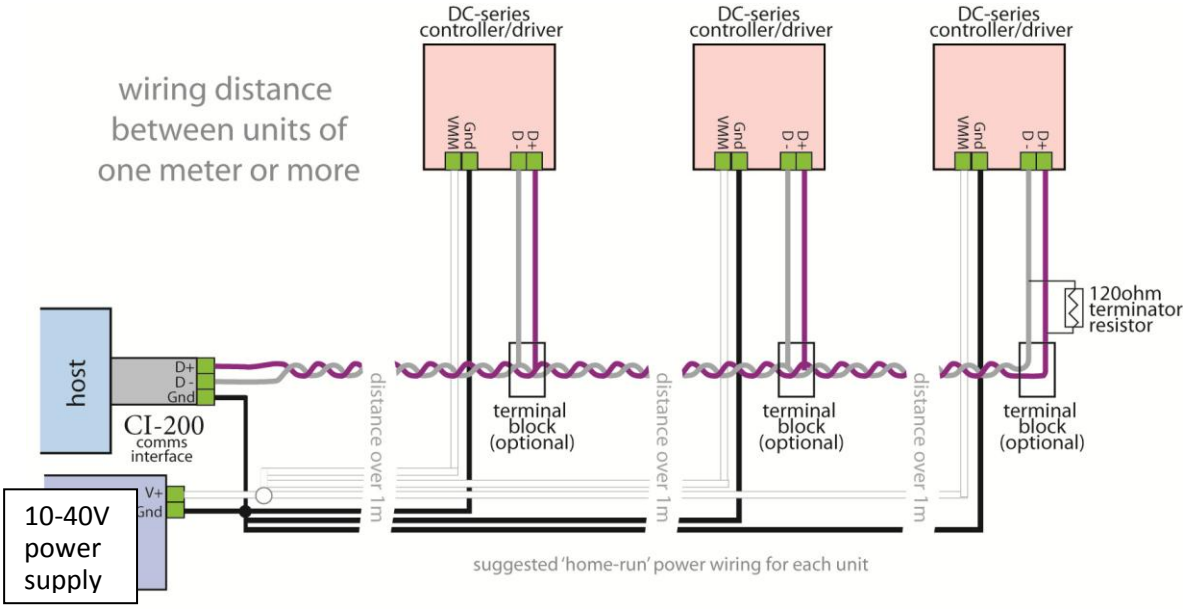


Figure 13 - Twisted-pair and Shielded Wiring example (recommended)

Multidrop busses like the AM Comms Bus are organized in a 'party-line' arrangement, where devices connect to a main cable trunk via short network stub connections. Typically these busses are resistively 'terminated' at each end to control reflections that would reduce the reliability of the bus (high-speed bus architectures and termination theory are beyond the scope of this document). It is important to keep the wiring from each device to the main bus as short as possible, to minimize deterioration of the signal waveforms. Ideally, these segments should also use twisted-pair leads.

For both methods, add a 120 ohm termination resistor to the bus (as illustrated in the diagrams, and provided with the CI-200) across the conductors, at the end of the bus farthest from the master communications source. Near-side termination is implemented inside the CI-200.

Grounding

Grounding is another important aspect to communications – a common ground keeps all devices communications signals at or near the same voltage level. Wire all DC-series unit Ground connections together as well as to the communications adapter ground (such as CI-200) for reliable communications, or risk creation of out-of-spec voltages, and thus damage to communications hardware.

All shared ground and power leads are recommended to be connected in a 'home-run' arrangement from each unit to the single connection point at the main shared powers supply, so that voltage drops from one unit are not experienced by another unit, possibly causing drive or communications errors.

Commands

Command Description Template

Command	***Function***	***Type***		***Size***
char	*Command Name and Description*	*Type*		*bytes*
	Example	***Data 1***	***Data 2***	***Result***
	*(axis) Command *(data1) *(data2) (↲)	*(Range)	*(Range)	*return*

Where:

Command:	Keystroke or character mnemonic for the command
Function:	Functional description of command
Type:	Immediate Mode = Direct execution when provided by host Program Mode = Executable in stored program Global = Targets all controllers present in Multi-Axis Mode
Size:	Storage requirements in program memory steps (and NV memory steps)
Example:	Usage, shown as: Single character prefix used in multi-controller protocol, appended by parameters where valid; prefixed by address (axis name) when used in Multi-Axis mode
*Axis:	Axis name, needed if in Multi-Axis Mode
*Data 1:	First Parameter (if required)
*Data 2:	Second Parameter (if required)
↲	Optional Enter (^M) termination character
*Range:	Valid numerical range of parameter
Result:	Information returned to host as a result of command execution

** Items marked with parenthesis are optional, depending upon mode*

Command Notes:

- All Commands can be used in **Single-Axis Mode** or **Multi-Axis Mode**, with exceptions: ^N, ^P, and X
- Multi-Axis commands are only applicable when sent in **Immediate (Direct Control) Mode**, and aren't valid in **Program Mode** — *programs are local and controllers cannot communicate with each other*
- In **Multi-Axis Mode** commands (non-global ones) are always prefaced by the required address (name), such as: `X+1000` , where X is the addressed axis name, and "`+1000`" is the command
- "P" command enters **Program Entry Mode**; sent again in that mode it ends **Program Entry Mode**
- Global commands affect all connected units at the same time
- Global commands are single-character, and don't need or support termination character (Enter↲)
- Characters are echoed on a line-by-line basis (see Command **T** (Echo Mode))
- Only one command may be entered per immediate entry line, or program line
- Non-Global **Immediate Mode** Commands are terminated by Enter↲ (keyboard), or ^M (with script)
- The command line may be edited using backspace, as characters are typed
- An **Immediate Mode** line entry may be canceled using <ESC>
- A space is optional to separate the command and first parameter value
- A space or comma must be used to separate parameters, where two are present

^N (Name Axis)

Command	***Function***	***Type***	***Size***
^N	Name Controller	Immediate	N/A
	Example	***Data 1***	**Data 2**	***Result***
	^N	none	none	none

The single-character command **^N** ("Ctrl N," or 0E hex) is used to assign each DC-series controller an address, called an 'axis name'. This name is used to address commands to individual controllers, when more than one unit is on the same Communications Bus. See *Multi-Axis Considerations* for more data.

Each DC-series controller in a Multi-Axis network must be uniquely named — this allows multiple axis commands and operations over a single communications bus.

The axis naming operation must be performed in **Single-Axis mode** during which only one controller is connected to the communications bus (via the host communications adapter). Naming cannot be performed in Multi-Axis mode. See *Axis Naming* section for more information.

Procedure: Upon sending **^N** the controller responds with "NAME?" Enter the desired name character, and the controller will respond "Save (Y)?" Type "y" (lower or upper case) to approve. "OK" is echoed back and the name is immediately stored in NV memory. The controller is then reset, similar to ^C (see
^C (Reset) command). Any other answer to the "Save (Y)?" question cancels the name change.

Legal name characters are limited to "a" through "z" and "A" through "Z". All other names will be rejected — you will be asked to re-enter a new axis name until a legal one is entered.

^P (Enter Multi-Axis Mode)

Command	***Function***	***Type***	***Size***
^P	Enter Multi-Axis Mode	Global	N/A
	Example	***Data 1***	***Data 2***	***Result***
	^P	none	none	none

The single character **^P** ("Ctrl P", or 10 hex) immediately places all connected controllers into **Multi-Axis Mode.** It does not require or support a line-termination character, and the mode change occurs immediately. There is no echo of this character.

From 1 to 64 controllers may be connected in a Multi-Axis network, and all will react to the ^P command together.

When multiple DC-series controllers (for multiple axes) are connected in one communications bus, the host must place the system into **Multi-Axis Mode** immediately after power up — signing-in to Single-Axis mode with the <space bar> with multiple units connected is illegal.

In **Multi-Axis Mode** no command echo is sent (to avoid potential bus conflicts).

SPACE (Enter Single-Axis Mode)

Command	***Function***	***Type***	***Size***
SPACE	Enter Single-Axis Mode	Immediate	N/A

Example	***Data 1***	***Data 2***	***Result***
(Name) <SPACE>	none	none	Model ID

The single-character command **<SPACE>** (Space Key, or 20 hex) places the controller into **Immediate Mode**, allowing a direct exclusive connection from the host to a single DC-series controller. The result returned is the Model ID number and revision of the DC-series controller/driver unit. There is no additional command echo. The result appears as: `AccelMotion <model> <version>`

Benefits of **Immediate Mode** include more valid commands available and allows commands to be sent without addressing. Entry into this mode is needed at least once for Axis Naming (see *Multi-Axis Considerations* Section and ^N command reference).

This command does not require or support a line-termination character, and the effect is immediate.

^C (Reset)

Command	***Function***	***Type***	***Size***
^C	Reset Controller(s)	Immediate, Global	N/A

Example	***Data 1***	***Data 2***	***Result***
^C	none	none	none

The single-character command **^C** (“Ctrl C“ or 03 hex) resets all connected controllers to power-up conditions. It is analogous to “Ctrl-Alt-Delete” on a classic computer. It does not require or support a line-termination character, and the reset action occurs immediately.

All connected units will be reset as follows:

- all outputs are set to inactive (high)
- default parameters and programs are reloaded from non-volatile (NV) memory
- position is set to zero
- If the controller has an automatic program start GO in location 192, it will execute
- controller will wait for `<SPACE>` character, to signal Immediate Single-Axis connection or ^P to enter Multi-Axis mode

This command may not be placed in program memory.

This command will end Program Mode on all units – all programs will stop.

If in Multi-Axis Mode, this command will take all units out of that mode, and back to waiting. To continue working in **Multi-Axis Mode**, the “**^P**” needs to be issued again after the “**^C**”.

No echo is sent for this command.

ESC (Global Abort)

Command	***Function***		***Type***	***Size***
ESC	Terminate Operations		Immediate, Global	N/A
	Example	***Data 1***	***Data 2***	***Result***
	<ESC>	none	none	none

The single-character command **<ESC>** (Escape Key, or 1B hex) aborts active operations. It does not require or support a line-termination character, and the effect is immediate.

<ESC> terminates any active motion, and stops any program in process, for all connected units. In **Immediate Mode** all units enter idle state awaiting more commands.

- Stepping will cease immediately without deceleration – the lack of deceleration can cause mechanical overshoot, so use caution with this command when in motion
- All current programs stop, and the unit(s) exits **Program Mode**
  - If sent in **Multi-Axis Mode**, all programs stop on all connected units at once
- If sent during **Program Entry Mode**, the controller will immediately exit that mode
- Output port settings and output port states are not affected

This command may not be used within program memory - if sent during **Program Entry Mode**, it will not create a command, but instead will exit Program Entry Mode.

If <ESC> is sent in **Multi-Axis Mode**, unit(s) will stay in **Multi-Axis Mode**, awaiting more commands – No echo character is sent

@ (Soft Stop)

Command	***Function***		***Type***	***Size***
@	Soft Stop		Immediate, Program	1
	Example	***Data 1***	***Data 2***	***Result***
	(Name) **@** ↲	none	none	none

If axis is moving, **@** command decelerates the motor to a stop, using "K" *Decel* ramp parameters.

In **Program Mode** when this command is encountered: any motion already in-progress stops – use a W0 command before @ to allow previous motion to complete. Program continues.

| (All Stop/End)

Command	***Function***		***Type***	***Size***
| (pipe)	All Stop/End Program		Immediate, Program	1
	Example	***Data 1***	***Data 2***	***Result***
	(Name) **|**	none	none	none

The single-character command **|** (pipe, or 7C hex) aborts active operations. It does not require or support a line-termination character, and the effect is immediate.

| command operates on the addressed unit – in **Single-Axis mode** it's the connected unit and in **Multi-Axis mode** it is the addressed unit via name prefix – as follows:

- terminates active motion on addressed unit immediately without deceleration – the lack of deceleration can cause mechanical overshoot, so use caution with this command
- Program on addressed unit will stop
  - unit exits **Program Mode** and returns to idle awaiting commands in **Immediate Mode**
- If sent in **Multi-Axis Mode**, all units stay in Multi-Axis mode after stopping motion/programs
- Output port settings and output port states are not affected
- Effect is immediate – No echo character is sent

This command may not be used within program memory - if sent during **Program Entry Mode**, it will not create a command, but instead will exit Program Entry Mode.

***|** causes motion in-progress to stop – use W0 command before | to allow previous motion to complete*

Stop Command Highlights – <ESC>, ^C, @ and |

- **^C** restarts the sign-on process for all units, requiring either **^P** to continue in multi-axis mode or **<space>** to continue in single-axis direct mode
- **^C** and **Esc** are Global and affect all units; **|** (pipe) and **@** stop only the addressed unit
- **|** stops motion immediately with no decel, but **@** stops with decel
- **|** is a single-character stop and valid at all times, but **@** is only valid as a separate command line, with CR/LF

^ (Read Moving Status)

Command	Function	Type	Size
^	Read Moving Status	Immediate	1
	Example	**Data 1**	**Data 2**	**Result**
	(Name) ^ ↲	none	none	Status

The **^** command is used to determine the motion status of a controller, in **Immediate Mode**; response "1" indicates axis is still moving, and "0" means no motion on that axis. ^ is not valid in Program Mode.

It can be used in **Single-Axis Mode,** or **Multi-Axis Mode** by addressing the axis controller, by name.

When **^** is used to poll the completion of a motion command, the command will:

- **return the motion status result *immediately***
  motion commands execute in 0 time, so the next command including ^ happen immediately (see Command W reference page for more information on command ordering)
- **return status irrespective of the** "Run to Hold timer". See the **E** (Delay to Hold Time) Command for more info

A similar function under **Program Mode** is the L command, allowing looping based on motion status.

{ (System Status Read)

Command	Function	Type	Size
{	System Status Read	Immediate	N/A
open brace	**Example**	**Data 1**	**Data 2**	**Result**
	(Name) { ↲	none	none	System Status

The **{** command (Open brace, or 7B hex) responds with a result string of the current System Status.

This command returns system status values about over-temperature, overvoltage, and undervoltage.

This status consists of the following values:

1. current driver temperature, in decimal degrees Celsius
2. current VMM supply voltage, in tenths of volts (24.5V = 245)
3. current VIO supply voltage, in tenths of volts (12.0V = 120)

The format of the response is:

{TT/VVV/vvv

The values are: the temp T; the VMM voltage V; and the VIO voltage v – the values are separated by the character "/". An example response would be:

{52/304/124 <-- this example shows a temp of 52C, 30.4V VMM, and 12.4V VIO

This data can be combined with the } **System Fault** status command to determine the system status.

Review the Thermal Considerations section for recommendations on controlling heat with settings and environmental adjustments.

} (System Fault Status Read)

Command	Function	Type	Size
} close brace	System Fault Status Read	Immediate	N/A

Example	Data 1	Data 2	Result
(Name) } ↲	None	None	Faults
(Name) } 0 ↲	*0 (zero) – clear all*	None	none

The } command (close brace, or 7D hex) responds with a result string of current **System Fault Status**.

System faults occur by continuously comparing temperature and voltages to specification maximum and minimum values. Once a fault is reported, the fault condition is latched so that the operator will know that the fault has occurred and can review the situation and repair the fault. System faults clear on power-up or full reset event.

The **}** command responds with status of the system faults, identified by letter values for easy parsing:

1. **F – Fuse Fault**, where fuse is missing or has been blown due to overcurrent
2. **T – Over-temperature Fault**, where driver temperature has exceeded maximum limit
3. **V – Over-voltage Fault VMM**, where VMM main supply voltage exceeded maximum limit
4. **I – Over-voltage Fault VIO**, where VIO supply voltage exceeded its maximum limit
5. **D – Driver Fault**, where the driver reported a driver overcurrent (due to short or arcing)

Each fault status type is reported in the response string in order, as either the letter of the fault (such as F, V, etc.) or a – (dash character) indicating no fault. This response assures that the non-fault status message arrives actively.

Example response with no faults reported : **}-/-/-/-/-**

Example response with all faults reported : **}F/T/V/I/D** <- this is an unlikely situation

The system faults are maintained (latched) until power-down or full reset (^C).

This data can be combined with the { **System Status** command to determine the full system status.

The System Fault **}** command with data 0 (zero) will clear all currently logged faults. Note, the System Fault status bits are explicitly cleared, but if these faults continue or re-occur then the status will continue to be reported as the fault condition remains. A Fuse fault will clear (and red light will disappear) as soon as the fuse is replaced.

+ (Index in Plus Direction)

Command	***Function***		***Type***	***Size***
+	Index (step) relative to current position in Plus Direction for n counts		Immediate, Program	5 (7 with v)
	Example	***Data 1***	***Data 2***	***Result***
	(Name) + n ↲	n steps (0 to 2,147,483,647)	v (new velocity)	none

Command **+** causes stepping in the positive direction for the specified 'n' step count, relative to the axis current position. [Refer to Direction Setting section for information on positive and negative directions]

The full range of offset n is: 0 to 2,147,483,647 steps from the *current position*. If the position counter reaches count 2,147,483,647, it will overflow to -2,147,483,648 and continue to increment.

The motion sequence is:

1. Wait until any motion already in progress is complete
2. If optional Data2 velocity value (v) is present, then the **V** Velocity value is updated as if the **V** Command had been issued just before the +/- command (see **V** command for more data)
3. Energize the driver, using the "Y" command RUN Current parameter
4. Start stepping in the positive direction at the rate of initial velocity ("I" command parameter)
    - If I parameter is larger or equal to the existing V parameter, then skip to step 5
5. Accelerate using slope set by the K (Ramp Slope) *Accel* parameter, to V (slew velocity)
6. Continue stepping at the V Speed, until the calculated deceleration point is reached
7. Decelerate, at a rate determined by the K *Decel* parameter, to a stop at the final index value
8. Once a motion is complete, if another motion is not commanded within the settling delay period (see **E** Command) then output power changes back to the HOLD Current setting – if another motion does occur inside the "E" delay, then driver power stays at RUN throughout.

In Program Mode, all motion commands (M, +/-, R+/R-) automatically stall (waiting) as long as previous motion is still in progress – non-motion commands are not stalled automatically behind a motion command, so review usage of the W0 Command usage for setting execution timing

Use W0 prior to any command that should wait for previous motion to complete

– (Index in Minus Direction)

Command	***Function***		***Type***	***Size***
–	Index (step) relative to current position in Minus Direction for n counts		Immediate, Program	5 (7 with v)
	Example	***Data 1***	***Data 2***	***Result***
	(Name) – n ↲	n Steps (0 to -2,147,483,648)	v (new velocity)	none

Same as **+** command, but in the opposite direction and causing opposite overflow. Parameters I, V, and K Accel/Decel are used in the negative direction, but always translated from absolute-value format.

See **+** Command reference for complete description and information.

A (Input Port Read – Decimal)

Command	***Function***	***Type***	***Size***
A	Input Port Read	Immediate	1

Example	***Data 1***	***Data 2***	***Result***
(Name) **A** ↲	none	none	Input State in decimal

The **A** Command queries the input port status, without a parameter; the value of all Input Ports are returned combined into a single decimal value, representing data as a binary value; if **A** command returns "A9" (decimal 9 = binary 1001) then I4 and I1 are ACTIVE, and all others are INACTIVE.

All inputs signals are evaluated based on their voltage compared to the Voltage I/O (VIO) level – if the input voltage is over ½ of VIO then it is PASSIVE or OFF, and if under ½ of VIO then it is ACTIVE or ON. See *Input Port Electrical Configuration* for more details.

If the user supplies VIO then it is used, but if VIO is not provided externally then a 5V level is used

All inputs signals are evaluated based on their voltage compared to the Voltage I/O (VIO) level – if the input voltage is over ½ of VIO then it is PASSIVE or OFF, and if under ½ of VIO then it is ACTIVE or ON.

Input Port encoding into Input State Value												
Port Number	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1
Decimal value	2048	1024	512	256	128	64	32	16	8	4	2	1

All ports are readable by "A" Command in **Immediate Mode**, and all can control conditional branching via the "L" Command in **Program Mode,** irrespective of any fixed function assigned to each port.

B (Jog Speed)

Command	**Function**	**Type**	**Bytes**
B	Jog Speed/Velocity (default = 400)	Immediate, Program	3

Example	**Data 1**	**Data 2**	**Result**
(Name) B ↲	none	none	B value
(Name) B (n) ↲	SPS (0 – 32,000)	none	none

The **B** command modifies the Jog Speed value – the speed which the motor will move when an input is assigned as JOG (**U** command) and the input is triggered. The parameter n is in steps-per-second (SPS). It is updated into main parameter memory; retain it permanently in NV memory with the S Command.

If the command is issued without data, it will return the current setting of the **B** parameter (useful in **Multi-Axis Mode** where the X command is not available). If the command is used with data, it will change the Jog Velocity value immediately to the SPS value.

JOG+ and JOG- activation *will stop any running program, and stop any motion already in progress*.

C (Clear and Restore main memory)

Command	**Function**		**Type**	**Bytes**
C	Clear and Restore main memory		Immediate	N/A
	Example	**Data 1**	**Data 2**	**Result**
	(Name) C ↲	None (implied n=0)	none	none
	(Name) C n ↲	n = 0-2	none	none

Command **C** erases or restores the contents of Main (RAM) Memory depending upon the specific command parameter 'n' used. This command does not modify NV (non-volatile) Memory in any way, with any parameter. main memory is defined as the space where current configuration is stored.

The command is controlled using data1 parameter 'n', as follows:

"C0" restores all Parameters from NV (non-volatile) Memory into main memory
'C0' does not affect main memory Program Space, or controller axis name

"C1" restores all Program Memory from NV (non-volatile) Memory into main memory
'C1' does not affect main memory Parameters, or controller axis name

"C2" resets all main memory Parameters to Factory Default Values
'C2' does not affect main memory Program Space or controller name.

*To save these Factory Defaults to NV storage, issue "**C2**" then "**S0**" command*

"C3" erases main memory Program space, and thus deletes all programs from main memory.
'C3' does not affect current Parameters or controller axis name

*To clear NV Program Memory as well, issue "**C3**" then "**S1**" command*

Issuing **C** command without parameter is equivalent to "C0", and would restore parameters from NV.

Use the **S** command to store new default parameters to NV, or to clear NV Program Memory.

Memory Space organization and address space info can be found in the *Memory Map* section.

D (Speed Divider)

Command	Function	Type	Bytes
D	Divide Speed Value (default= 4)	Immediate, Program	2

Example	Data 1	Data 2	Result
(Name) D ↲	none	none	D value
(Name) D n ↲	Divider n (1-255)	none	none

The "D" command modifies the Speed Divider value. All step speeds during ramping and slewing are divided by this divider value (n) in a range between 1 and 255. Speeds as low as three revolutions/day may be obtained by increasing resolution (H) with high D value. The D command updates the setting into main memory; retain it permanently in NV memory with the S Command.

As the divider "n" is increased, parameters for all motion speeds must be adjusted if it is desirable to maintain the same output step speed.

The D command can achieve very slow speeds, or can achieve smoother Accel and Decel (Command K) ramps. When D is larger, more discrete steps are used when ramping from one motor speed to another.

The D value should never be changed while moving.

If the command is issued without data in Immediate Mode, it will return the current divider value.

If the command is used with data, it will change the Speed Divider value immediately.

E (Delay to Hold Time)

Command	Function	Type	Bytes
E	Hold Settling Time (default=100)	Immediate, Program	2

Example	Data 1	Data 2	Result
(Name) E ↲	none	none	E value
(Name) E t ↲	time **t** x 10msec (5-255)	none	none

The "E" command modifies the Hold Settling Delay value; this delay is the time between the moment the last step of a motion command is *completed* until the driver switches from RUN current to HOLD drive current (if no other command is executed during that period). It provides an easy way to achieve power reduction, if the HOLD value is set lower than the RUN value, and its use is recommended.

The **E** Command updates main memory; retain the delay permanently in NV memory with **S** Command.

Delay value t is in 10mS (0.01 second) increments – the maximum delay is ~2.5 seconds. The factory default value is 100, or 1.0 seconds. Note that even if very slow stepping rates are programmed, the Hold Settling Delay will not be engaged between slow pulses because the *actual movement command is not yet complete.* The countdown begins on the *last step pulse* of a motion command; if another command causes a step before the timeout has ended then the new motion starts at RUN current and the timer starts after that command – continuous commands never HOLD. To prevent the transition from RUN to HOLD entirely either set E to 255, or set RUN and HOLD to the same value (**Y** command).

Entering values of E less than 5, or more than 255, are invalid; E will be set to the default value of 100.

If the E command is issued without data in Immediate Mode, it will return the current value of delay.

If E command is used with data, it will change Hold Time Delay to that new value immediately.

F (Find Home)

Command	Function		Type	Bytes
F	Find Home		Immediate, Program	4
	Example	**Data 1**	**Data 2**	**Result**
	(Name) F n (d) ↵	n SPS (1-32,000)	d Direction (0,1)	none

The **F** command implements the Find Home function, which moves the axis to a physical location determined by a HOME switch, such as a microswitch or other detector. DC-series controllers contain a special Home algorithm which eliminates mechanical hysteresis caused by system mechanical backlash.

First, fix the location of the HOME switch – most installations place the HOME location to one extreme edge of the available mechanical motion space, so that homing can always occur in the same direction from anywhere in that axis. Set the home switch mechanical positioning so that the switch is activated once the axis hardware moves to the outside of the home location – a microswitch can work in this case. Be sure that it stays actuated for some overshoot distance beyond the HOME location. The process steps below assume the homing direction d is chosen 'toward' the HOME location.

- The process starts with the F command; the first parameter is the 1st stage velocity toward HOME, and the optional second parameter is the requested starting direction toward the Home switch; 0 is positive and 1 is negative (if d is not provided, then positive is assumed) – a higher speed is usually selected for the homing speed n, so that the axis reaches home quickly.
- The axis begins motion in direction d, accelerating from **I** velocity with K accel characteristics up to speed n; this home speed is usually set to a higher rate, for quick homing. The axis motion continues at speed n until the HOME switch is activated.
- When the HOME switch becomes ACTIVE, the axis reduces speed at K decel rate until stopped, with some overshoot in direction d, and then automatically reverses direction and begins moving at rate **I** (Initial Velocity), slowly searching for deactivation of the HOME switch.
- At the moment the HOME switch goes INACTIVE, the axis motion stops at its HOME location – this allows both quick homing and accurate repeatable placement of the physical end-effector.

Using Normally-Open Home Switch

The default operation of the **F** command assumes a normally-open HOME switch. Such a switch or detector would be wired from the chosen Input Port (see Command **U**) to ground. The DC-series controller automatically inverts the input, similar to how general-purpose inputs are inverted (see A command). So, when the switch is activated physically it connects the input to ground, and the input circuit converts that to an ACTIVE status. The process shown above works in this default way.

Using Normally-Closed Home Switch

If a normally-closed HOME switch or sensor is desired (where the ACTIVE condition occurs when the switch is NOT operated, and the signal is INACTIVE when the switch is operated), use the **p** Polarity Inversion Command. Once polarity setting for HOME is inverted and a normally-closed switch is used, the F homing function operates identically to the normally-open switch without polarity.

This command may be implemented within a program. Following is an example:

```
P 0          Enter program mode
F 1000 1     Find the home switch in the "1" direction at a step rate of 1000 SPS.
               <add other commands, motion now based upon HOME location>
P            Exit program mode.
```

G (Go/Branch)

Command	Function		Type	Size
G	Execute Program/Branch to location		Immediate, Program	4
	Example	**Data 1**	**Data 2**	**Result**
	(Name) G a (t) ↲	a address (0-1023 excluding 200-255)	t (0-1)	none

The **G** command is used to enter **Program Mode** and begin execution of a user-programmed sequence, starting at memory location "a". Programs can start at "0", however, any starting address may be used. In **Immediate Mode** "G" starts execution of a program manually, and is used within a program to branch to a new location. For conditional branches, refer to **L** (Loop Waiting on Condition) Command.

The available address range is 0 to 1023. In addition, address locations between 200 and 255 are reserved for parameter storage and may not be used in programs – this is a historical attribute of equipment from Advanced Micro Systems.

If optional parameter t is 1 (one), then **Trace Mode** is enabled; in **Immediate Mode** only Trace displays the instruction step being executed as the program is running, as a tool for debugging. The list format is the same as that of the "Q" command. Trace Mode will be in effect until the program terminates or until another "Go" is executed without the Trace attribute.

The controller is factory set with a test program similar to the following example:

```
        P0          Enters Program Entry Mode, from Immediate Mode
0       +801        Move 801 steps in the plus direction
5       W100        Wait 100 milliseconds
8       -800        Move 800 steps in the minus direction
13      W100        Wait 100 milliseconds
16      Z0          Display step position
17      G0 0        Go (Jump) to location 0 and continue executing
21      P           Exits Program Entry Mode
```

Type G0 (or just G) to execute this program at zero

A running program can be aborted from the console with the ESC command, returning control from **Program Mode** to **Immediate Mode** – In **Program Mode** only the commands "ESC","^C", "|" and "@" are permitted to be sent via the Communications Interface. In **Program Mode**, the | command is used to end a program, and exit Program Mode.

Upon completion of a program the controller will return a <CR><LF> message to the host, indicating the completion of the program and the exiting of **Program Mode** (unless in Multi-Axis mode).

Auto Program Execution

During power-up or after any reset procedure (see ^C command) the controller will automatically run any program starting at location 192. This feature is known as **Automatic Program Mode**, and allows the designer to place programs, such as initialization sequences or a homing routine, or to start a complete program sequence upon power-up. All Automatic Programs must begin with a "G" command, to quickly redirect program flow to the main program location; programs cannot be placed in locations 200 – 255.

H (Microstepping Resolution)

Command	Function	Type	Size
H	Resolution (default 1)	Immediate, Program	2
	Example (Name) H ↲ (Name) H n ↲	**Data 1** none n 0-5,8,9	**Data 2** none none	**Result** H value none

The **H** command sets the Stepping/Microstepping Resolution operating parameter, as shown below. The setting is updated into main RAM memory; retain it permanently in NV memory with the S Command.

N value	Resolution Mode	Steps/rev (1.8° motor)
0	Full-Step (max torque)	200
1	Half-Step (max torque) **← default microstep setting**	400
2	¼ Step	800
3	⅛ Step	1600
4	1/16 Step	3200
5	1/32 Step	6400
8	Full Step Modified (sum of phase current always equal)	200
9	Half Step Modified (sum of phase current always equal)	400

The full-step and half-step modes 0 and 1 provide maximum torque by using 100% instantaneous current value during overlapping steps. Modes 8 and 9 provide smoother operation with good torque by limiting current to only 71% of max current setting. All current values scale by the **Y** command.

H command changes effective speed and steps-per-revolution for all motion (M, F, +/-, R, V, I, K, etc).

If the **H** command is issued without data in **Immediate Mode**, it will return the current setting of H. This is especially useful in **Multi-Axis Mode** where the **X** command is not available.

I (Initial Velocity)

Command	Function	Type	Size
I	Initial Velocity (default 2000)	Immediate, Program	3
	Example (Name) I ↲ (Name) I n ↲	Data 1 none n speed (0 to 32,000 SPS)	Data 2 none none	Result I value none

The **I** Command modifies the Initial Velocity parameter, n, in steps per second. I is the first speed used at the beginning of acceleration (if I is lower than the target speed). It must be slow enough that the motor can start without missing steps (stalling) at the existing K accel setting. The setting is updated into main RAM memory; retain it permanently in NV memory with the S Command.

I and V Command velocity parameters are entered as *positive absolute values*, but are used in the direction the motion command (F, M, +, –, R+, and R–) demands. All velocity parameters, including Initial Velocity, are divided by the Divide Factor (Command D); use the examine (X) command to display velocities with the division factor included.

The initial velocity applies to: All index (step) commands: +, –, R+, and R–, during acceleration and deceleration, start point and end points with M constant velocity command, and the starting velocity for Home (F command).

If the I command is issued without data in **Immediate Mode**, it will return the current setting of I (such as `I400`). This is especially useful in **Multi-Axis Mode** where the **X** command is not available.

J (Jump to Address Multiple Times)

Command	**Function**		**Type**	**Size**
J	Jump to Address Multiple Times		Program	4
	Example	**Data 1**	**Data 2**	**Result**
	(Name) J a n ↲	a address (0-1023)	n additional loops 0-254	none

The **J** Command is used (in Program Mode) to execute a jump to a program location *repeatedly n times* – this command operates like a FOR loop, allowing a set of commands to be executed up to 255 times. A notable difference to the J command is that it is usually placed after the command or sequence, not before, to repeat them in a loop.

The address 'a' must be a valid command target address, and J command loops may not be nested.

The n parameter specifies the ***additional number of loops*** of commands (or sequences) that will be executed, beyond the first one. The final number of sequences will be n + 1.

This command is only valid in **Program Mode**. Following, is an example:

```
     P0        Enters Program Entry Mode, at location zero
0    +1000     Move in the plus direction 1000 steps
5    W250      Wait 250mS after completion of the previous move
8    -1000     Move in the minus direction 1000 steps
13   W250      Wait 250mS after completion of the previous move
16   J0 3      Jump Multiple, to location 0, to run 3 additional times
20   @         Stop, ends Program Mode when executed
21   P         Ends Program Entry Mode
```

In the above example the commands between locations 0 and 16 will be executed 4 total times: once when the program sequence is executed plus three more times resulting from the "J0 3" command. A more powerful example would include more complicated motion and even conditional operations within the loop core, but this example illustrates the method.

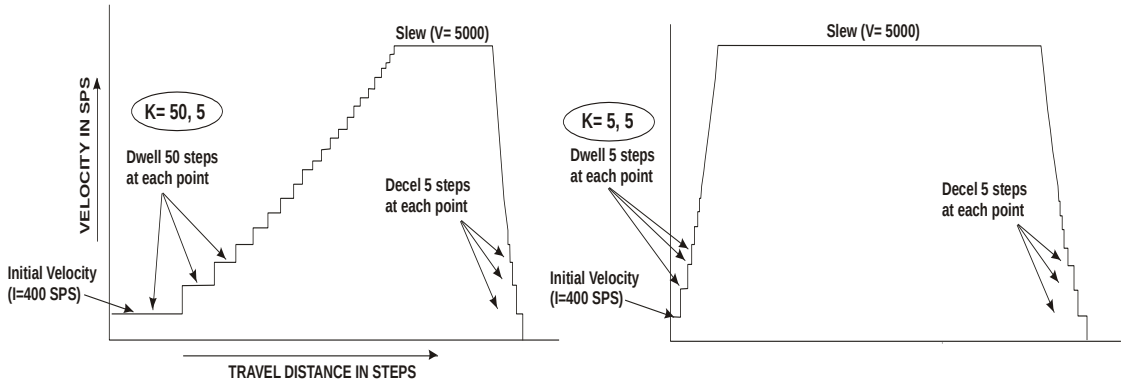
K (Ramp Slope)

Command	Function	Type	Size
K	Ramp Slopes (defaults 5/3)	Immediate, Program	3
	Example	**Data** 1	**Data 2**	**Result**
	(Name) K ↲	none	none	K values
	(Name) K (Accel) (Decel) ↲	Accel (1-255)	Decel (1-255)	none

The **K** command is used to adjust the acceleration and deceleration ramp slope, under changing motor motion. The Accel and Decel values are entered as positive multiplier 'step counts', as shown below. The setting is updated into main RAM memory; retain it permanently in NV memory with the S Command.

An internal lookup table defines the basic profile of the acceleration/deceleration curve, and the K multipliers adjust the curve. A number of discrete velocities are utilized between the value of **I** initial (beginning) and **V** slew (target) velocity, and a smooth curve of acceleration and deceleration of the motor speed is approximated by moving through these discrete velocities. During ramping, the K value determines how many *motor steps* are output at each velocity point on the acceleration (or deceleration) curve, before moving to the next velocity point. Higher K values will increase the dwell time at each discrete point on the acceleration ramp, while lower values of K will increase the acceleration rate. Very low K values will reduce ramping, but is generally not recommended.

The following two examples are of ramped indexes, each 2000 steps with I=400, V=5000, but different K parameter values: K50 5, and K5 5



The ramp is also influenced by the choice of D. Increasing D while adjusting the target speed correspondingly to result in the overall same system slew speed, will result in more speeds along the ramp curve and therefore finer speed steps and a slower ramp. Thus when D, K, and V are adjusted together they can be used to smooth the ramp where desired.

K command can be used in **Immediate** or **Program Mode**.

Changing **K** values during motion is disallowed.

If the **K** command is issued without data in Immediate mode, it will return the current settings of K (such as `K20/20`). This is especially useful in **Multi-Axis Mode** where the **X** command is not available.

L (Loop Waiting on Condition)

Command	Function		Type	Size
L	Loop Waiting (conditional jump)		Program	4
	Example	**Data 1**	**Data 2**	**Result**
	(Name) L a c ↲	a address (0-1023)	c source/polarity (see table)	none

Command **L** (Loop Waiting on Condition) is used to implement conditional execution flow, by testing a specified source and polarity (value c) based on input port values or other system conditions.

The command starts by acquiring the specified condition. **If the condition c has occurred, execution "falls through" to the next instruction – if condition has not yet occurred, execution jumps back to location a.**

Self-address mode: To make waiting loops easier, a special mode exists causing loop-to-self while waiting, without requiring a self-address calculation – implement by using target address **2048**. If the loop command targets address **2048**, then comparison failure returns to the same command location.

Address location "a" can be used to modify program flow as follows:

- if address a is the address of the same L command (or **address 2048**) then program flow will loop on the L instruction, until condition c is satisfied – ***L Command is used most easily this way***
- if address a is an address before the L command, then the *command sequence* between address 'a' and the L command location will execute repeatedly, waiting until condition c is satisfied – this could move the axis conditionally based on a result from a sensor (position, force, etc.)
- if address a is an arbitrary address then the L command simply operates as a single one-time conditional jump, going to the *next instruction if condition is true* and to *address a if false*

Source and polarity are selected with one value, c, as follows:

Source for Conditional branch		Fall-through Condition c	
		Fall through when source INACTIVE (port at VIO voltage)	Fall through when source ACTIVE (port at 0V)
inputs	**I1**	0	1
	I2	2	3
	I3	4	5
	I4	6	7
	I5	8	9
	I6	10	11
	I7	12	13
	I8	14	15
	I9	16	17
	I10	18	19
	I11	20	21
	I12	22	23
outputs	**O1**	32	33
	O2	34	35
	O3	36	37
	O4	38	39
	motor moving*	64 (fall through when not moving)	65 (fall through when moving)
	System Fault	92 (fall through when no fault)	93 (fall through when fault)

* looping with moving conditional is similar to waiting for stop with w0 but additional actions can occur during motion

Looping on output port could operate as a user-defined variable, whether port is connected or not.

Following is an example which waits until *input port 3* is activated, and then moves forward 1000 steps:

```
     P 0      Enter Program Mode
0    L0 5     Loop Back to location 0 (same location) while port 3 is inactive (see table)
              Fall through to next instruction when port 3 is active
5    + 1000   Move 1000 steps in the plus direction.
              <more program here>
```

Condition 64/65 during motion can implement other tasks, like setting an output port during motion. In most cases the use of W0, or not, after a motion can accomplish similar functions.

M (Move at a Constant Velocity)

Command	**Function**		**Type**	**Size**
M	Move at Constant Velocity		Immediate, Program	3
	Example	**Data 1**	**Data 2**	**Result**
	(Name) M (+/-) s ↲	s speed (± 0 - 32,000 SPS)	none	none

The M command causes the axis to move at a constant velocity (after completing its acceleration ramp). The direction of motion is specified by the sign preceding the velocity; "+" or "-" sign are valid, and no sign preceding the velocity is interpreted as positive. The velocity is in SPS steps per second.

Motion will start at the **I** Command Initial Velocity; ramping up using **K** command Accel value, to the target Move velocity specified by speed 's', in steps per second. Motion will continue at speed s until a new **M** Move command is entered or a Soft-Stop (@ Command) or other resetting condition occurs.

Move Command motion is terminated by:

- The "M 0" command; stops with K deceleration profile
- Soft-Stop (@) command; stops with K deceleration profile
- Soft-Stop fixed function input active; stops with K deceleration profile
- Abort (ESC); stops without K deceleration profile

The following commands modify the 'M' parameter's *effective speed*:

- (H) Microstepping value – visible on the X examine response
- (D) Divide – visible as a divided value, on the X examine response
- (K) Ramp factor – Accel and Decel parameters modify the acceleration/deceleration curves

Below is an example of M command; when executed, it starts a motion at 500 steps/sec, then uses L (Loop Waiting on Condition) to detect when the axis mechanicals activate a detector (switch or sensor) on Input Port 5, and then immediately decelerates and stops via the Soft-Stop command.

```
     P 0      Enter Program Entry Mode
0    M500     Move at constant step rate (accelerate from I, at rate K, to 500 SPS)
3    L3 9     Loop Back to address location 3 (itself) until port 5 is ACTIVE (low)
7    @        Decelerate and stop program execution
8    P        Exit Program Entry Mode
```

N (Output Port Read/Write - Binary)

Command	***Function***		***Type***	***Size***
N	Write Outputs/Read Back (Binary)		Immediate, Program	2
	Example (Name) N ↲ (Name) N i (d) ↲	***Data 1*** None o port# (1-12)	***Data 2*** none d data value (0 or 1)	***Result*** Outputs (decimal) none

The **N** command modifies output port values in Immediate and Program Mode, and can read back those values in Immediate Mode only. When used with a port number and data, only the individual output o is modified based on that d value – if data d is a 1, then the port pointed to by o is made ACTIVE (1), and if data is 0 then the o port is made INACTIVE (0). *All other ports are unaffected*.

If **N** is issued alone, the controller returns the state of all outputs in binary as a result. An example result when O3 and O1 are active would be: "`N0101`". All ports are unaffected by this read command.

The default value of all output ports at power-on/reset is 0 (INACTIVE) – *note: INACTIVE state is a high voltage level (VIO), and 1 or ACTIVE state corresponds to a low level (ground).*

> *All inputs signals* are evaluated based on their voltage compared to the Voltage I/O (VIO) level – if the input voltage is over ½ of VIO then it is PASSIVE or OFF, and if under ½ of VIO then it is ACTIVE or ON.
>
> VIO is also used to "pull-up" all output ports that are OFF or INACTIVE so they are compatible with external devices or PLCs that need a passive voltage provided.
>
> If the user supplies VIO then it is used, but if VIO is not provided then internal 5V level is used.

The '**N**' Command allows control of each output port value individually. The **w** method is easier when all output levels required are known at the same time, or some outputs must change at the same time.

Example shown below, steps are: all outputs start inactive, then activate O2, disable O2, activate O1 and O2, then deactivate all at once:

command	**event**	**O2**	**O1**	
		INACTIVE	**INACTIVE**	
N2 1 ↲	O2 on	ACTIVE	INACTIVE	**N** changes ports individually
N2 0 ↲	O2 off	INACTIVE	INACTIVE	
N2 1↲	O2 on	ACTIVE	INACTIVE	
N1 1↲	O1 on	ACTIVE	ACTIVE	
w0 ↲	all off	INACTIVE	INACTIVE	**w** changes all ports at once

Output ports can drive logic-level outputs but can also drive medium-power devices such as indicator lights, relays, sounders, small solenoids, etc. at up to 30V and 1A on each channel. See the *Input / Output Subsystem* chapter to determine proper output circuits, connections, and possible limitations for your application.

Reading output port conditions in **Program Mode** with the **L** command can detect changes in outputs from different parts of your program, or it can allow an unused output port to work as a 'variable' to communicate between various parts of your program.

O (Set Origin)

Command	Function	Type	Size
O	Set Origin (position counter reset)	Immediate, Program	5

Example	Data 1	Data 2	Result
(Name) O ↲	None (0 implied – clears to zero)	none	none
(Name) O a ↲	a position (-2,147,483,648 to +2,147,483,647)	none	none

The **O** command sets the internal 32-bit signed *Position Counter (Origin Counter)* to a specified value; this counter is used as the reference in origin-relative motion commands (such as R+, R–), and is incremented or decremented by all motion command movements.

The maximum value the counter can reach is +2,147,483,647 and the minimum value is –2,147,483,648. If maximum or minimum value is encountered, the counter will 'roll over' to the opposite polarity value and continue updating.

It is recommended to update the counter only at idle or after a W0 (wait for motion end).

Hardware reset (**^C**) and power-on reset clears the Position Counter to "0" similar to **O**0 (o zero).

Utilize the **Z** command to manually read out the position counter in Immediate Mode, and **O** command to reset or preset the origin counter value. Use **O** in combination with the Home (**F**) command to set the virtual origin to the mechanical origin.

P (Program Mode - Entry/Exit)

Command	Function	Type	Size
P	Program Mode - Entry/Exit	Immediate	N/A

Example	Data 1	Data 2	Result
(Name) P ↲	Implies address 0 (zero)	none	none
(Name) P a ↲	a address (0-1023)	none	none, #

The **P** command forces the controller into and out of **Program Entry Mode**; the first use of "**P**" transitions the controller from Immediate Mode to Program Entry Mode starting at location 'a', and "**P**" entered while in Program Entry Mode transitions back to **Immediate Mode**. Program Entry Mode may also be terminated with the (escape) character. When in Program Entry Mode, commands and data entered are placed into main memory Program space, one command at a time.
Use S1 (save) command to save main memory program to NV memory, and make it permanent.

It is strongly recommended that program development & debugging be done in Single-Axis/Direct Mode. When entering programs in Single-Axis/Direct Mode, the controller automatically provides the target program address location to the console, before accepting the command for that location; in **Multi-Axis Mode** the convenient program location numbers are not provided. Also, *entering commands in Multi-Axis Mode requires every line entered from the host to be prepended with the targeted axis name*, just as any regular command in Multi-Axis mode would. Program Entry in Multi-Axis mode will only echo the axis name and the command entered. For more, see *Operational Modes* section.

Internal Programs are an optional feature of the DC-series controller – the user can certainly control all units (axes) via a host-based script or application (see Programming section); but, when designing internal programs development can be done directly at the console input, or (for most terminals) into a built-in text editor. It is recommended that programs be written or documented in an editor on the host computer and then downloaded to each controller, due to easier editing and viewing of the program on a computer. From a documentation perspective, this method is safest as it allows host-based storage of the programs – it's then easy to download the program to a new or replacement unit as needed.

(continued)

When using an editor (such as AccelCom), it's useful to add the commands needed to enter Program Entry mode, then the program contents, then the exit; here's a simple program created in the editor:

```
P              enters Program Entry Mode
+1000           |
W250            |
-1000           | -------- THIS IS THE MAIN PROGRAM
W250            |
L0 100    <- loops here waiting for condition
P              exit Program Entry Mode
P100           re-enter Program Entry at 100
+100000         |
@               |
P              exits Program Entry Mode
```

When entering code directly, if a mistake is made the user may use the Backspace key, which will erase all characters of the current line giving the user the opportunity to re-enter the line. In single-axis mode the line number will be displayed after the backspace key. When the line entry is cleared with Backspace in Multi-Axis Mode, the axis name *will be retained*, but there will be no location number presented.

If an invalid command character is entered for Program Mode, it will be rejected. In Single-Axis mode the user will see the line number repeated and the user can then enter a valid command. In Multi-Axis Mode if the user enters an invalid command, he may follow it by a valid command and the invalid command character will be ignored. When entering an invalid command followed by <Enter>, it will be ignored and the user will be required to start the new line by re-entering the axis name.

More than one program may be entered, separated by different starting addresses. These programs can then be executed via the "G (address)" command for each desired function.

Program and parameter memory exist in 2 layers: A volatile main memory (RAM) area which contains the current operational parameters and program, and a non-volatile NV Memory layer that is stored permanently, and can be restored to main memory at any time, and always at power-on. Programs can be saved permanently to NV memory using the "S1" command.

All programs entered into or modified in main memory during any session are lost when powering down the controller, if not saved to NV; on power-up main memory is restored from NV memory.

There are special address ranges shown below (refer to *Memory Map* section):

Address	Function
0 - 191	User program memory section 1
192 - 199	program vectors for automatic start at power-up
200 - 255	**Do not use**
256 - 1023	User program memory section 2

During program entry, any code entered beyond address 1023, or between 200 -255, will be ignored.

See *Programming* section for a more program entry examples.

p (System Settings, includes polarity and external step/dir output)

Command	Function		Type	Size
p (small "p")	System Settings		Immediate, Program	2
	Example (Name) p ↲ (Name) p s ↲	**Data 1** none s switch/type (0-3)	**Data 2** none none	**Result** s values none

The System Settings **p** command is used to modify and read a set of standard controls for DC-series controllers. The data value s is used to determine the setting and setting value, as shown below. If the **p** command is issued without data in **Immediate Mode**, it will return the current settings of p (such as "p0"). This is especially useful in **Multi-Axis Mode** where the X command is not available.

Create the 's' data field from the sum of the data fields of each section below: Add SP and SE to create the s data field value. Add SP + SE = s Example: SP = 1 and SE = 4 – 's' value is 5.

Polarity for Home and Limit Switches

The p command is used to set the polarity of either the optional Home switch (used in the F command), or the optional Limit switches (always valid when configured by the U (Set Port)Command).

Polarity adjustment sets the ACTIVE level of the HOME and LIMIT+/LIMIT- matched to the type of switch selected for these functions. Most Home and Limit switches are the normally-open variety, but some sensors or logic functions may require normally-closed versions ('opens' the connection when actuated).

A benefit of normally-closed switch use is increased protection – if a normally-open limit switch fails it may be masked because the circuit is normally open; but a normally-closed switch circuit would fail early and be detected if a wiring or connector failure occurs, and thus be detected early before it is needed.

Home Switch Type	Limit Switch Type	Setting SP for bits 1 and 0
Normally open	Normally open	0
Normally open	Normally closed	1
Normally closed	Normally open	2
Normally closed	Normally closed	3

External Step and Direction Outputs

This system setting outputs standard step/direction outputs for use with external drive systems (such as AccelMotion D2C, D5C, etc. or AMS CDR4MPS driver systems). When active, output 1 to drives STEP output and output 2 drives DIRECTION output.

Step/Direction	Setting SE for bits 1 and 0
Output 1 and Output 2 used for outputs	0
Output1 -> STEP / Output2 -> DIRECTION	4

Q (List Program)

<table>
<tr><td rowspan="2">Command

Q</td><td colspan="2">Function
List Program</td><td>Type
Immediate</td><td>Size
N/A</td></tr>
<tr><td>Example
(Name) Q ↲
(Name) Q a (m) ↲</td><td>Data 1
None implies address start is zero
a address (0-2560)</td><td>Data 2
none
m mode 0-1</td><td>Result
Listing
Listing</td></tr>
</table>

The **Q** Command is used to list programs stored in main memory, using the output format:

```
Location Address <Space> Command Char Value 1 <Space> Value 2
```

Twenty instructions are displayed at a time. The values will be displayed only if applicable to the particular instruction type. Use the Enter↲ key to list up to 20 more commands. ESC ends the listing, and any other key causes the listing to 'single-step' display each line.

This command is available in **Immediate Single-Axis** and **Multi-Axis** Modes, but is not valid within a program. In Multi-Axis mode every line response is preceded by the axis name.

If Data2 (mode) is not provided or is "0", the command will display instructions until a non-valid instruction is encountered. If mode is "1" the command will list addresses and instructions starting with the address specified by Data 1, and continue to list all valid program address contents found between the starting address and the end of memory (1023).

R (Motion Relative to Origin)

Command	Function	Type	Size
R	Motion Relative to Origin	Immediate, Program	5

Example	Data 1	Data 2	Result
(Name) R ↲	None implies move to zero (0)	none	none
(Name) R n ↲	n position (-2,147,483,648 to 2,147,483,647)	none	none

Command **R** will cause motion similar to **+** or **–** commands, but relative to the internal 32-bit signed *Position Counter (Origin Counter)*. The target position has a range of ±2,147,483,647 steps from the 'Z' origin (set by "O" command). The n position can be prefaced by + or –, but no preface implies positive.

The motion sequence for **R+** or **R-** command is:

1. Wait until any previous motion is completed
2. Read *Position Counter* value, then subtract from 'n' target position to yield offset steps required
3. Energize the motor winding at MOVE current value
4. Start stepping in the calculated direction at initial velocity **I** (assuming it is equal or less than V)
5. Accelerate using the profile defined by the **K** command Accel value
6. Acceleration continues until the Slew Speed is attained, as specified by the **V** command
7. Motion continues at the Slew Speed, until the deceleration point is reached
8. Decelerate (determined by the Decel **K** value) to a stop completing the motion (index)
9. If another motion is not commanded within the **E** settling period, move to HOLD current

R offsets from the origin O, where +/- offset from any current position. Following, is an example that illustrates the benefit of R: when conditional branches move the axis to unpredictable locations, the origin provides the anchor that always allows the programmer to find the fixed location (the Origin 0):

```
O          Set position counter to 0
+50000     Move toward  location 500000 (accelerate from I, at rate K, and slew at V)

    ------ NOTE that non-motion commands shown below start immediately! ------

L3 1       Loop Back to address location 3 (itself) until port 1 is ACTIVE (low)
@          Soft Stop will stop motion when port 1 active irrespective of actual location

...              <other program activity>

R30000     R 30000 moves the axis to offset O + 30000 location, irrespective of how far it
           came in the previous portion of the program
...              <continuing program steps>
```

R can be used in Immediate and Program Modes. Utilize the **Z** command to manually read out the position counter in Immediate Mode, and **O** command to reset or preset the origin counter value. Use **O** in combination with the Home (**F**) command to set the virtual origin to the mechanical origin.

In Program Mode, all motion commands (M, +/-, R+/R-) automatically stall (waiting) while previous motion is still in progress – non-motion commands <u>are not stalled</u> behind motion commands; review W0 Command usage

Use W0 prior to any command that should wait for previous motion to complete

S (Save)

Command	Function		Type	Size
S	Save to NV Memory		Immediate, Program	N/A
	Example	**Data 1**	**Data 2**	**Result**
	(Name) S t ↲	t type (0,1)	none	none

The **S** Command saves Parameters and/or Programs into NV (non-volatile) permanent memory.

A controller's operational parameters and programs are kept in main RAM memory, but this memory is lost at every power-down (and **^C** reset). To retain the parameters and programs permanently, they must be saved to *non-volatile (NV) memory*, individually. Once saved, the parameters and programs in NV are recalled during each power-up sequence as the default starting condition. Also, they can be recalled with the **C** Command at any time. Refer to ***C** command* and *Memory Map* sections for more info.

Programmed use of **S** command is not recommended, as NV memory longevity may be affected.

This command should never be issued while a motion command is being executed, as it may interrupt motion during the save operation. Review the *Memory Map* section for more information.

S0 - All parameter contents (locations 200 - 255) are saved into NV Memory – these are recalled as default parameters during subsequent power-on events or reset (**^C**).

S1 - All program area contents (locations 0 - 199, and 256 - 1023) are saved in NV Memory – the entire program space is recalled into Main Program Memory during subsequent power-on reset (**^C**).

Sending **S** command with no data value t is illegal, and not equivalent to S0 or S1.

T (Echo Mode)

Command	Function	Type	Size
T	Echo Mode	Immediate, Program	2

Example	Data 1	Data 2	Result
(Name) T ↲	none	none	T value
(Name) T m ↲	M Mode (1-3)	none	none

The **T** command selects from three different command-entry echoing modes, explained below. If the **T** command is issued without data, it will return the current echo mode parameter.

Various echo modes are useful depending upon the requirement for return data and the need to track correct reception of commands at the host. Line-by-line mode is most convenient for most situations, but the other modes are available for more demanding designs.

The echoing modes are shown, with formats and responses, as follows:

Data 1 = 1: Line-by-line echo mode

Characters sent by the host since the last <enter> are echoed, once <enter> is received:

Characters shown in this view are in time order, with the responses below

```
Host:              M  1  0  ↲ (enter key)
Controller:                      M  1  0  <CR><LF>
                   → time axis
```

(continued)

Under Echo Mode 1, command requests data from the controller (such as a Z command), then that data will be added to the response prior to the end of the line:

```
Host:          Z  ↲
Controller:          Z  5  3  2  6  <CR><LF>
               → time axis
```

Data 1 = 2: Checksum Echo Mode

Characters sent by the host since the last <enter> are echoed, once an <enter> is received. In this mode, instead of echoing the message sent, only a 1-byte checksum is returned.

```
Host:               M  1  0  ↲
Controller:                <ASCII(174)>  <CR><LF>
               → time axis
```

The algorithm for the checksum is as follows:

Checksum byte = ((sum of all bytes sent by host) modulo 256) bitwise OR (binary 1000 0000)

In the case of Multi-Axis mode, the axis name is NOT included in the "sum of all bytes sent by host". Example in Multi-Axis mode:

```
Host:               A  M  1  0  ↲
Controller:                A  <ASCII(174)>  <CR><LF>
               → time axis
```

Calculation is as follows: "M" carries the ASCII code 77 (in decimal), "1" is 49 and "0" is 48; the arithmetic sum is 174. The total is below 256 so the modulo operation yields the same result. In this case the highest bit is already one, so the bitwise OR will not change the result value.

Commands that requests data from the unit (such as the Z command) will not include any response data in the checksum calculation. It will be inserted between the checksum byte and the <CR> <LF>.

Data 1 = 3: No echo mode

In mode 3 only the <CR> is echoed; no other characters, not even the axis name in case of Multi-Axis mode. If a command requests data it will be inserted prior to the <CR>. Examples in Multi-Axis mode:

```
Host:          B + 1  0  0  0  ↲        C  Z  ↲
Controller:                  <CR><LF>        5  3  2  6  <CR><LF>
               → time axis
```

Note that there are a few exception commands that differ from these described echo rules. These are:

- There are a few commands for which the controller will not echo at all. These are: **^C**, **^P**, and **ESC**. These commands affect all connected axes and an echo could result in comms bus contention.
- The ^N (name) command is an interactive dialogue for name setting and provides replies real-time.

The echo mode parameter can be saved in NV memory, using the "SO" command.

U (Set Port)

Command	Function	Type	Size
U	Set Ports	Immediate, Program	3

Example	Data 1	Data 2	Result
(Name) U ↲ (Name) U p f ↲	none p port 1-4	none f function 0-9	U values none

The **U** command enables the assignment of fixed functions to each input port (I1 – I12). Data 1 defines the port (1=I1, 2=I2, … 12=I12) and Data 2 defines the function, according to the table below:

Data 2	Function & Command	Description of Function when ACTIVE**
1	**User Input Port**	**Normal User Input Port Function – no fixed function** Use A or N command to read input port values in **Immediate Mode**, and L command allows branching on input port status in **Program Mode***.
2	**Home** **F**	HOME input polarity for ACTIVE state can be inverted via **p** command. **Note: This function allowed only on Input 1**
3	**Go** **G**	When Go is driven from INACTIVE to ACTIVE, a program at location 0 begins executing. If Go is held ACTIVE, will start execution at zero repeatedly, even if program ends or stopped by <ESC> or "@"
4	**Soft Stop** **@**	When Soft-Stop input is transitioned from INACTIVE to ACTIVE** the current motion is decelerated and stopped, and program is stopped – if held active will block motion or program start.
5	**Jog+** **B**	While Jog+ is driven ACTIVE, the motor will jog in positive (+) direction at speed B. Current motion or programs will stop
6	**Jog–** **B**	While Jog– is driven ACTIVE, the motor will jog in minus (–) direction at speed B. Current motion or programs will stop
7	**Reserved**	Reserved – Do not use
8	**Limit+**	Limit+ inhibits all motion in the positive (+) direction if activated** LIMIT input polarity for ACTIVE state can be inverted via **p** command
9	**Limit–**	Limit– inhibits all motion in the negative (–) direction if activated** LIMIT input polarity for ACTIVE state can be inverted via **p** command

* A, N and L commands read the value of inputs regardless of their function – User Input function is provided for convenience

** INACTIVE is defined as a voltage between the high-rail voltage on VIO and ½ VIO – ACTVE is defined as a voltage between ½ VIO and ground – activate inputs by pulling the signal to ground, with low resistance – see *Input Ports* section for more info

As an example: "U2 3" will assign the GO input to port I2. ***HOME function is valid on I1 input only.***

The factory default sets all inputs to function 1 as general-purpose user inputs.

The current status of the input ports can be read using the **A** or **N** command.

The **X** command will show the current U setting for the input ports (such as U=111111341189).

If **U** command is issued without data in Immediate mode, it returns the current setting of U (such as U=111111341189). This is especially helpful in Multi-Axis mode where the "X" command is not usable.

V (Slew Velocity)

Command	Function	Type	Size
V	Slew (final) Velocity (default= 10,000)	Immediate, Program	3

Example	Data 1	Data 2	Result
(Name) V ↲	none	none	V value
(Name) V (n) ↲	n speed (1 to 32,000 SPS)	none	none

The **V** Command sets the Slew Velocity V to n steps-per-second (SPS) which must be a positive non-zero value. It is the maximum speed reached after acceleration from the initial velocity **I** and is carried on throughout the slewing motion. The maximum speed on users hardware will be limited by mechanical and motor characteristics and power setting – for more see Command **Y** (Hold and Run Current). The setting is updated into main RAM memory; it can be retained permanently in NV memory with the **S** Command.

I and **V** Command velocity parameters are entered as *positive absolute values*, but are utilized as negative or positive speeds matching whichever direction the motion command (+, –, R+, and R–) requires. Example: V is programmed to 3000 SPS, but becomes -3000 SPS when a -10000 negative step command is issued.

Attempts to enter values above the maximum speed are limited to the maximum velocity.

As with all velocity parameters, the actual slew velocity is divided by the **H** Microstepping and **D** (Speed Divider). Using the **X** examine command displays all velocity parameters with **D** division factor included.

The slew velocity V:

- applies to all index (step) commands: +, –, R+, and R–, after acceleration from I
- **does not apply** to **M** Move constant velocity command (has its own velocity value)
- **does not apply** to **F** Home Command (has its own velocity value)

If **V** command is issued without data in Immediate Mode, it will return the current setting of V. This is especially useful in **Multi-Axis Mode** where the X command is not available.

In-motion adjustments of Slew Velocity

The **V** command can allow changes to the speed of a motion *in-progress,* after it has already begun, when +, –, or R motion may be required to change speed *during the move*. The **V** command, when executed during motion will alter the contents of the V parameter in real-time and modify the speed at which the motion continues, from that moment on. Changes to V Velocity values do not change the target location of the motion – only the step rate changes and thus the time it takes to reach the target.

The DC-series is designed to allow the command interpreter to accept and process commands that do not use the same resource, so non-motion commands are processed even while motion is occurring (but motion commands stall other motion commands until the earliest motion is complete). This is why the W0 (wait for motion completion) command is needed – some non-motion actions must be synchronized with the end of motion, and some can continue during motion. One action available during motion is updating the V slew velocity. See the +, –, or R motion commands for more info on command stalling.

w (Output Port Read/Write - Decimal)

Command	***Function***	***Type***	***Size***
W small w	Write Outputs/Read Back (Decimal)	Immediate, Program	2

Example	***Data 1***	***Data 2***	***Result***
(Name) w ↲	None	none	outputs (decimal)
(Name) w (d) ↲	d data 0-15	none	none

The **w** command modifies output port levels in Immediate and Program Mode, and can read back those values in Immediate Mode only. When issued with a value d, all outputs are modified based on that d value, at once.

When command **w** is issued without a value, outputs are unchanged and the controller returns the existing output state in decimal, as a result.

The default value of the output ports at power-on/reset is 0 (INACTIVE. Note that 0 (zero) or INACTIVE state is a high voltage level (VIO), and 1 or ACTIVE state corresponds to a low level (ground). See *VIO*

All inputs signals are evaluated based on their voltage compared to the Voltage I/O (VIO) level – if the input voltage is over ½ of VIO then it is PASSIVE or OFF, and if under ½ of VIO then it is ACTIVE or ON.

VIO is also used to "pull-up" all output ports that are OFF or INACTIVE so they are compatible with external devices or PLCs that need a passive voltage provided.

If the user supplies VIO then it is used, but if VIO is not provided externally then a 5V level is used.

Input Port Electrical Configuration section for more details.

The **w** method is easier when all requested output levels are known at the same time and when more than one value must change at once. In contrast, the **N** Command allows control of each output port value individually.

Output States from decimal d value

d value	**Output**			
	O4	**O3**	**O2**	**O1**
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Command **w** Example:

command	**event**	**O4**	**O3**	**O2**	**O1**
		INACTIVE	**INACTIVE**	**INACTIVE**	**INACTIVE**
w2 ↲	set 0, 0, 1, 0	INACTIVE	INACTIVE	ACTIVE	INACTIVE
w0 ↲	set 0, 0, 0, 0	INACTIVE	INACTIVE	INACTIVE	INACTIVE
w10↲	set 1, 0, 1, 0	ACTIVE	INACTIVE	ACTIVE	INACTIVE
w3↲	set 0, 1, 0, 1	INACTIVE	ACTIVE	INACTIVE	ACTIVE
w0 ↲	all off	INACTIVE	INACTIVE	INACTIVE	INACTIVE

Output ports can drive logic-level outputs but can also drive medium-power devices such as indicator lights, relays, sounders, small solenoids, etc. at up to 30V and 1A each. See the *Input / Output Subsystem* chapter to determine proper output circuits, connections, and possible limitations for your application.

W (Wait)

Command	**Function**	**Type**	**Size**
W	Wait fixed / Wait for motion end	Program	3

Example	**Data 1**	**Data 2**	**Result**
(Name) W 0 ↲	none, implies 0 (wait for motion end)	none	none
(Name) W m ↲	m wait value x 10 mSec. (0-65000)	none	none

The **W** Wait command is used in **Program Mode** to create predictable delay. "W0" is a reserved command that waits for previous motion command to complete before allowing following command to execute – "Wm" command, where m is any non-zero value between 1 and 65000, instead will cause a wait of m milliseconds x 10 (minimum delay is 1mS and maximum delay is 650 seconds) before continuing execution.

When used *after* (fixed-step-count) motion commands, such as a +, –, R+, and R–, "W0" command will cause the next command to **wait for that previous motion to complete**. A **W** Command with no data value is equivalent to a "W0" command.

W command with a numeric wait value will wait until m x 10msec have passed, **with no consideration for previous motion command progress**. Ex: "W250" waits 2.5 seconds from the command, whether a motion is occurring or completed. Since motion command execution takes no time to execute and has no inherent delay, a "W250" directly after a +1000 will count essentially from the motion start time.

The following example program makes a move, waits for motion to complete, then turns on an output port for 5 seconds. Some uses for this could be illuminating a LED, signaling a sequence is complete, or operating a valve.

P 0	*Enter program mode*
+1000	Index 1000 steps in the plus direction
W0	**Wait for motion to complete before continuing**
w1	Turn output port 0 ACTIVE
W500	Wait 5 seconds
w0	Turn output port 0 INACTIVE
...	< other program instructions >
P0	*Exit program mode*

The following example operates differently; by using program steps including delays after a motion command, other functions can be accomplished in parallel with the motion. It shows a large movement starting and then an output is enabled **exactly ½ second after the long motion command starts** – this could be used to actuate a valve, motor or indicator during the movement.

P 0	*Enter program mode*	*delay*
+100000	Move 100,000 steps (positive direction at speed V)	*none*
W200	Waits 2.5 seconds **from the start of the previous move**	**2.5s**
w1	Turn output port 0 ACTIVE	none
W50	Wait 0.5 seconds	0.5s
w0	Turn output port 0 INACTIVE	none
...	< other program instructions >	
P0	*Exit program mode*	

If W0 is used after an M (constant velocity) command, the W0 will allow execution of the next command once it has ramped to the target speed of the M command.

X (Examine Parameters)

Command	Function	Type	Size
X	Examine Settings	Immediate	N/A
	Example	**Data 1**	**Data 2**	**Result**
	X ↵	none	none	All Settings

X command returns the current value of all operational parameters in **Single-Axis Immediate Mode**.

Please see the section *Parameters & Defaults* for the factory default values of the various parameters.

Here is an example response:

```
K=5/3
I=400/1
B=400/1
V=3000/1
Y=25/50
E=100
D=1
H=1
U=1 1 1 1 1 1 1 1 1 1 1 1
T=1
l=0
N=A
```

Where values displayed are:

K	=	Ramp up (accel) / ramp down (decel)
I	=	Initial velocity / Divider value
B	=	Jog Speed / Divider value
V	=	Slew Velocity/ Divider value
Y	=	Hold Current / Run Current
E	=	Settle time (delay between)
D	=	Divider value
H	=	Resolution
U	=	List of functions assigned to all input port (I12 -> I1)
T	=	Echo Mode value
l	=	Homing switch polarity value
N	=	Controller name character

In **Multi-Axis Mode** the **X** command is not available. To read parameters back in Multi-Axis Immediate Mode, execute the respective command for each parameter, **but without data**; the controller will respond with the echo of the command letter, then the value of the parameter. Please see the reference data for each command for more details.

Y (Hold and Run Current)

Command	Function	Type	Size
Y	Set Hold and Run Current	Immediate, Program	3

Example	Data 1	Data 2	Result
(Name) Y ↲ (Name) Y h r ↲	none h Hold in %max (0-100)	none r Run in %max (0-100)	Y values none

The **Y** command specifies the Hold and Run values of motor drive current (per phase), in 1% increments. The value 100% represents a maximum of 2Amps per phase.

Run current is defined as the drive current used during any motion (index), and Hold current refers to an optional lower value of drive current when the motor is not in motion.

The purpose of this feature is to allow mechanical systems with less torque requirement to hold position at rest using less power and generating less heat, yet use more power when accelerating, moving, and decelerating. Most systems benefit from the reduction of heat and power loss in the driver during rest (hold). To implement complete **Hold Current Disable**, set the Hold current to zero – this will maximize efficiency and minimize motor and driver heating. Example: *"Y0 80"* sets hold current off and move current to 80%

The switch from Hold to Run value is automatic and immediate whenever a motion function starts. Current reduction from Run back to the Hold value is also automatic and occurs a fixed period of time after the motion has completely ended; this "settling time" (see **E** command) must be completed without another motion command before the reduction to Hold current occurs.

If **Y** is issued without data in **Immediate Mode**, the controller will return the current setting of **Y**, in the format "Y Hold/Run". This is useful in **Multi-Axis mode** where the **X** command is not available.

The following is an example of Hold/Run current feature:

1. Issue the **Y** command to program the desired current values

   Entering "Y10 80" yields a 10% Hold current (200mA) and an 80% Run current (1.6A)

2. Drive current is modified immediately to 10% (200mA) because the system is at rest
3. Optionally issue an "S0" (Save) command to store all parameters into non-volatile memory
4. Any subsequent motion command causes the driver circuits to be set to the 80% current value, instantly. On completion of the motion (and after E settling time delay), the current is automatically reduced back to the 10% Hold current level.

Entering out of range values for either the hold or the run current will cause factory defaults to be selected for the respective value.

Note: Refer to section "Stepping Motors" for more detail on setting the proper motor current.

Z (Read Position)

Command	**Function**		**Type**	**Size**
Z	Read/Display Current Position Counter		Immediate, Program	2
	Example	**Data 1**	**Data 2**	**Result**
	(Name) Z ↲	none	none	Position

The **Z** command is used to read the 32-bit signed *Position Counter (Origin Counter)*. During a move command, this value will update depending on steps moved and direction of travel. The counter signifies steps moved at the current **H** microstepping value and **D** divider value. The initial value of the counter is 0 at power up, and can be modified by the **O** Command.

The maximum value the counter can reach is +2,147,483,647 and the minimum value is –2,147,483,648. If maximum or minimum value is encountered, the counter will 'roll over' to the opposite polarity value and continue updating.

The **Z** command can be issued from a host in **Immediate Mode,** or be used in **Program Mode**. Note, however, Z is invalid from **Program Mode** when also in **Multi-Axis Mode**. All commands that return data (such as Z) are suppressed in **Program Mode** and **Multi-axis Mode** together to assure predictable communications traffic that is synchronized with the host.

We recommend updating the position counter with **O** command only after a W0 (wait for motion end).

Specifications

Absolute Maximum and Minimum Ratings

Characteristic	Value	Min	Max	Unit
Supply Voltage	V_{MM}	**10**	**40**	V (DC)
Positive Voltage Reference for Input/Output ~4.8V (unloaded) generated internally, if not externally provided	V_{IO}	**0**	**36**	V
Input and output signal voltage range (J3 pins 1-6)	V_{input} V_{output}	**0**	**VIO**	V
Output Drive Current/Phase (at V_{MM} voltage) set by Y command - maximum value **100**	i_{phase}	**-**	**2**	A
VIO supplied current	I_{IO}	**-**	**250**	mA
RS485 input voltage	V_{data}	**-12**	**12**	V
Continuous Operating Temperature Measured at thermal sensor via command { (open brace)	T_{maxdrv}	**0**	**85**	°C

Thermal and Mechanical Specification

Peak Operating Temperature* ..... 0 to +100°C abs max (> 85°C will cause driver thermal cut off)
Storage Temperature ..... -40 to +125°C
Size ..... 4.5” x 2.2” x 0.85”
Weight ..... ~100g, including mating connectors

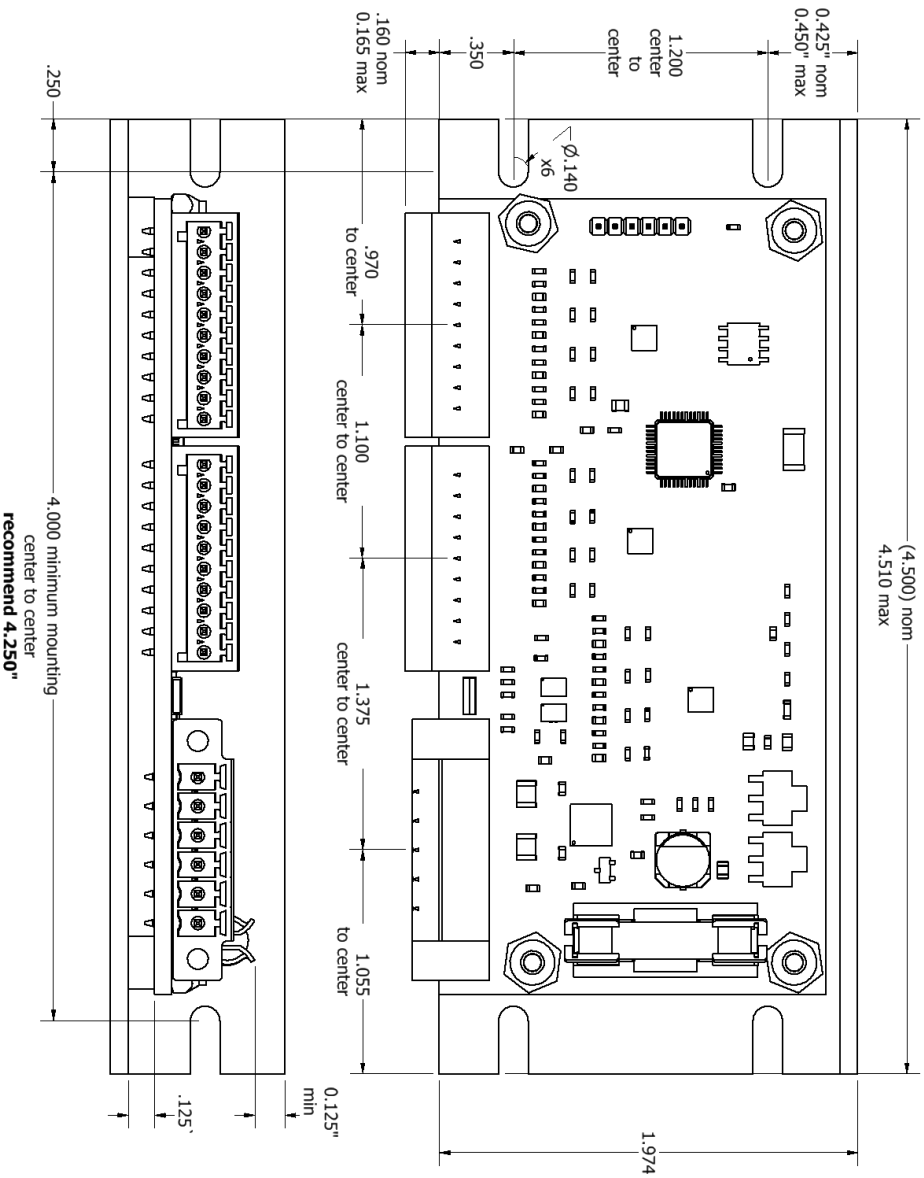
Programming Specifications

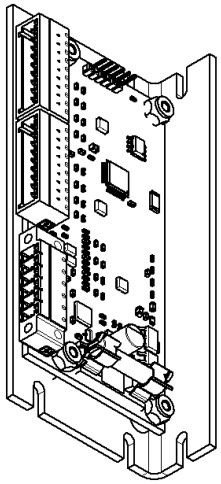
Microsteps Per Full Step..... 1 (full-step), 2, 4, 8, 16, or 32 (programmable)
Non-Volatile Memory..... 1024 Program Locations (see each command for requirements)
Position Counter..... -2,147,483,648 to +2,147,483,647 steps
Baud Rate ..... 115200 baud

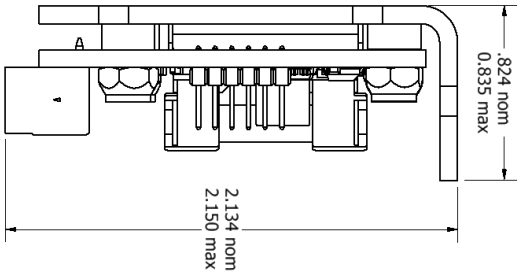
Dimensions & Mounting

Mounting is recommended against the bottom plate or back plate surfaces using provided mounting holes (4.250” centers wide and 1.20” row to row on bottom).

Assure that mounting provides appropriate thermal heatsinking or proper airflow (see Thermal Considerations).







Design Tips

Thermal Considerations

The DC2C has been optimized to minimize power usage and heat generation. Under most ambient temperatures, motor currents, duty cycles and supply voltages the unit will not overheat. But, ambient air temperature requirement must be determined by the user, based on usage settings of drive current and duty cycle – the mounting and airflow characteristics are then derived that would keep the internal temperature of the unit under the operating max of 85°C.

To measure temperature limits, utilize the System Status Read Command **{** .

System Status Read command will return the actual temperature near the driver in degrees Celsius. See the **{** command reference for details. **Adjust the mounting, airflow, cooling, duty cycle or output drive current to achieve a maximum temperature under the operating max of 85°C.**

If the temperature inside the drive stage exceeds a higher critical limit, the drive stage will begin shutting off drive current until the temperature is below that critical threshold – this may appear as a stuttering or missing of steps, so look to thermal issues if stuttering or dropout occurs.

All motor drivers must be thermally engineered for proper operation

Keep the DC2C under 85°C by adjusting motor max current and duty cycle to reduce heat, and increase air flow to reduce temperature.

EMI

EMI (electromagnetic interference or electrical noise) can be a major source of problems when integrating power drivers with microprocessor-based devices. EMI is typically generated through ground loops and AC power line disturbances. External devices such as relays, coils, solenoids, arc-welders, motors are all sources of EMI.

The following design tips will help to prevent EMI from interfering with the system operation:

- Shield the device and wiring by mounting it in its own metal enclosure where possible, as far away from noise sources as possible.
- Use shielded and twisted pair cables for the motor, I/O, and communications wiring. Ground motor wiring shield leads only at the driver end.
- Make sure that all power wiring (motor, AC, etc.) is routed far away from the I/O signal wiring and communications lines.
- Mechanical and electrical grounds should all be tied to Earth at a single point. Chassis and motor grounds should be tied to the frame at a single point, and the frame should be tied to Earth.
- Use solid-state relays or opto-isolators whenever possible to isolate remote input signals.
- Suppress all mechanical relays with capacitors or MOV's.
- Add protection diodes to all relays (see *Output Ports* section)

Set-up for Phase Current Measurement

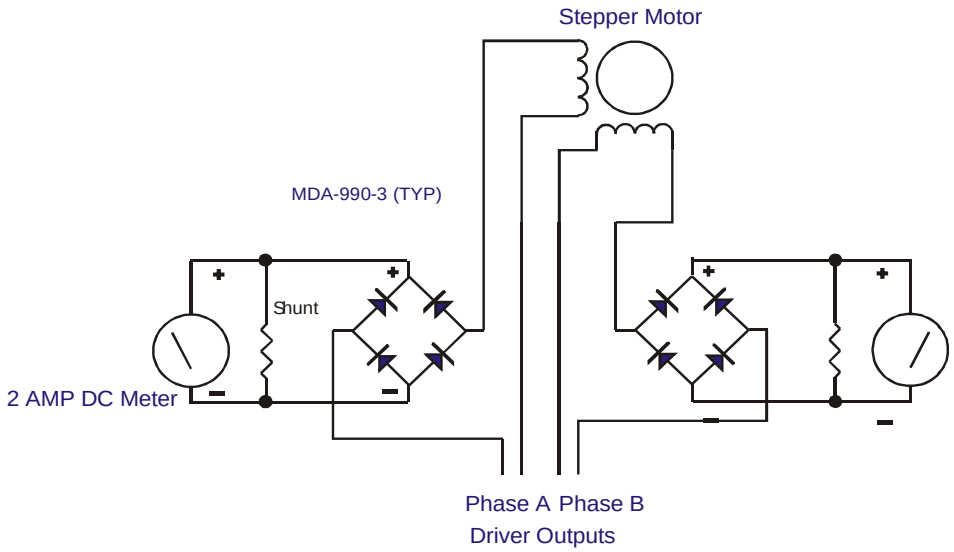
The following is the basic setup diagram for 2-phase average current measurement:

A. The ammeter can be digital or analog

B. The bridge rectifier should be rated above the maximum expected voltage and current

C. A small capacitor (filter) may be needed across the meter

D. Additional meter protection circuitry may be desired (not shown)



General Procedure

Before beginning, make sure the power is off and let any residual power supply capacitors discharge whenever motor circuits are connected or disconnected.

1. Ensure set-up is wired as shown in the above diagram
2. Apply power to the DC2C
3. Set the DC2C to half-step mode using the H command (H1)
4. Set the hold and run current to a safe reference measurement value of 50%, using "`Y50 50`"
5. Issue multiple individual steps ("+1" or "-1" command), until one channel has reached maximum current and the other is zero. As you step, you will see the current increase/decrease with every step. For both readings of the maximum phase current, alternate stepping to a point where the current is at its peak value for each phase.
6. The meter should now read about 1000mA – 50% of the 2A maximum current. Check both phases.

WARNING:
CONNECTING or DISCONNECTING MOTORS
WHILE POWER IS APPLIED WILL CAUSE DAMAGE
THAT IS NOT COVERED BY AcclMotion WARRANTY

ASCII Character Code

Ctrl	Char	Dec	Hex	Code		Dec	Hex	Char		Dec	Hex	Char		Dec	Hex	Char
^@		00	00	NUL		32	20			64	40	@		96	60	`
^A	☺	01	01	SOH		33	21	!		65	41	A		97	61	a
^B	☻	02	02	STX		34	22	“		66	42	B		98	62	b
^C	♥	03	03	ETX		35	23	#		67	43	C		99	63	c
^D	♦	04	04	EOT		36	24	$		68	44	D		100	64	d
^E	♣	05	05	ENQ		37	25	%		69	45	E		101	65	e
^F	♠	06	06	ACK		38	26	&		70	46	F		102	66	f
^G	•	07	07	BEL		39	27	‘		71	47	G		103	67	g
^H	◘	08	08	BS		40	28	(		72	48	H		104	68	h
^I	○	09	09	HT		41	29	)		73	49	I		105	69	i
^J	◙	10	0A	LF		42	2A	*		74	4A	J		106	6A	j
^K	♂	11	0B	VT		43	2B	+		75	4B	K		107	6B	k
^L	♀	12	0C	FF		44	2C	,		76	4C	L		108	6C	l
^M	♪	13	0D	CR*		45	2D	-		77	4D	M		109	6D	m
^N	♫	14	0E	SO		46	2E	.		78	4E	N		110	6E	n
^O	☼	15	0F	SI		47	2F	/		79	4F	O		111	6F	o
^P	►	16	10	DLE		48	30	0		80	50	P		112	70	p
^Q	◄	17	11	DC1		49	31	1		81	51	Q		113	71	q
^R	↕	18	12	DC2		50	32	2		82	52	R		114	72	r
^S	‼	19	13	DC3		51	33	3		83	53	S		115	73	s
^T	¶	20	14	EC4		52	34	4		84	54	T		116	74	t
^U	§	21	15	NAK		53	35	5		85	55	U		117	75	u
^V	▬	22	16	SYN		54	36	6		86	56	V		118	76	v
^W	↨	23	17	ETB		55	37	7		87	57	W		119	77	w
^X	↑	24	18	CAN		56	38	8		88	58	X		120	78	x
^Y	↓	25	19	EM		57	39	9		89	59	Y		121	79	y
^Z	→	26	1A	SUB		58	3A	:		90	5A	Z		122	7A	z
^[	←	27	1B	ESC		59	3B	;		91	5B	[		123	7B	{
^\	∟	28	1C	FS		60	3C	<		92	5C	\		124	7C	|
^]	↔	29	1D	GS		61	3D	=		93	5D	]		125	7D	}
^^	▲	30	1E	RS		62	3E	>		94	5E	^		126	7E	~
^_	▼	31	1F	US		63	3F	?		95	5F	_		127	7F	

* On recent keyboards generated by Enter↵ key

Revision Log

July 2018	Release version 1.00 released
October 2018	Added L2048 self-addressing loop function and documentation

Contact AccelMotion

Email: support@accelmotion.com
Phone: 512-212-7300

AccelMotion Systems 3051 N Hwy 183, Unit 4 Liberty Hill, TX 78642 www.accelmotion.com