DC2M17

USER GUIDE

Motor-Mountable

Step Motor Driver/Controller Module

AccelMotion

TABLE OF CONTENTS

DC2M17 Features:

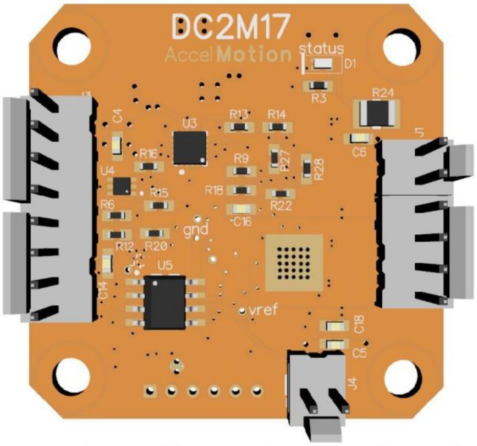
The **DC2M17** is the smallest of the DC-series of Accel**Motion** step motor driver/controllers. It operates a NEMA17- or NEMA23-size stepper motor at up to 2A max current per phase, under control of a host computer or PLC. DC-series controllers can also operate autonomously in Program Mode, when programmed with instructions and saved to non-volatile memory, with no need for an external host.

The **DC2M17** provides**:**

- Capability to drive a Step Motor with phase current up to **2A** max current per phase when equipped with heatsink/case (optional), and **1.5A**/phase when operated without the case
    - Programmable Run and Hold Current settings; control from host or internal program
    - Programmable Hold Current Reduction; use to reduce heat and power
- Single supply voltage, from 9 to 35VDC – including built-in fuse protection
- 38 commands for motion, programming, and configuration, providing powerful control
- Programmable microstepping resolutions: Full, ½ , ¼, ⅛, and $^1/_{16}$ step mode
- Step speed programmable up to 28,000 steps per second, and “divide-by” ratios up to $^1/_{255}$
- **Direct Control Mode** supports host-based control – **Program Mode** allows autonomous operation
    - **Automatic Program Start** allows programs to start at power-on automatically
- Four user-configurable input ports – each can be used for:
    - standard general-purpose user input – supporting IO voltages of 5 to 30VDC
    - standard fixed functions, including: Limits, Home, Go, Stop, or Jog
- Two general-purpose digital output ports, capable of sinking 1A of current per output at up to 30V
- Communications via AccelMotion Comms Bus – recommended adapter: Accel**Motion** **CI-200**
    - Physical: RS485 (RS422 voltage-compliant), supports up to 32 modules on one interface
    - Fast Operation, at 115,200 baud default speed
- Optionally mountable directly to standard NEMA17 stepper motor, for easy local control
- Built-in temperature sensor allows host to query exact temperature
- Status LED provides indication of power status and relative motion step rates
- Optional **DC2M17-CASE:** provides protection and heatsinking (to achieve max 2A drive current)
- Optional **DC2M17-CABLE** provides 30” cable harness and is color-coded for easy wiring

Included in the Box

- DC2M17 Control Module
- Mounting Spacers: Qty 4 (four), #4 nylon, 3/8”



Optional Accessories

Mating Cable Assembly with 30" color-coded leads: order DC2M17-CABLE

The DC2M17-CABLE is a color-coded cable assembly with pre-stripped lead ends and 30" length, providing easy and flexible cable connections to the **DC2M17**.

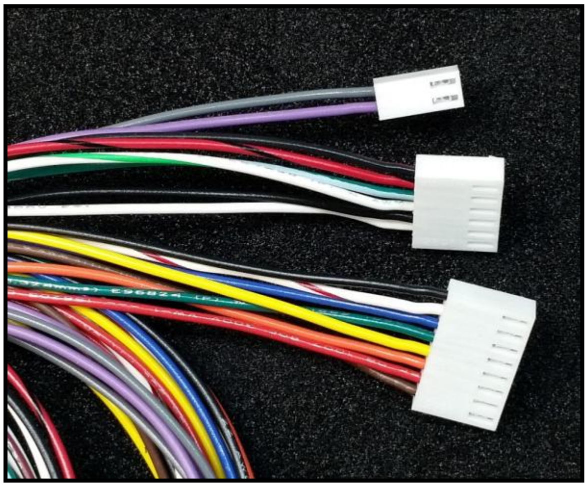


Figure 1 - DC2M17-CABLE

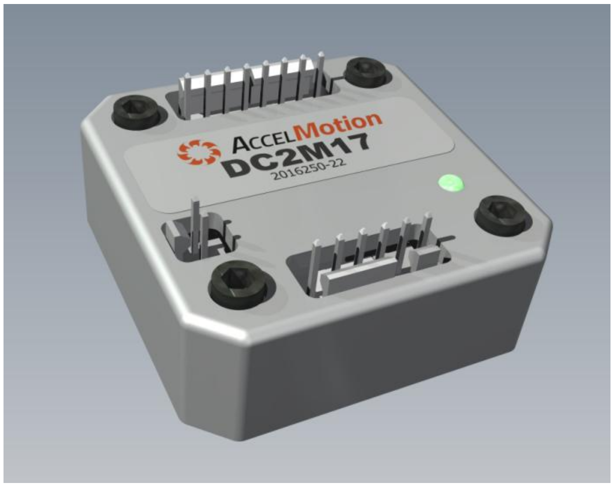


Figure 2 - DC2M17-CASE

Case/Heatsink: order DC2M17-CASE

The **DC2M17-CASE** provides a cover for the **DC2M17** Driver/Controller Module as well as providing additional heatsinking for the on-board stepper drive stage. The case also allows the DC2M17 power to increase up to 2A/phase max.

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 DC2M17, via USB. Drivers for Windows, Mac, and Linux are available and provide computer access to the **DC2M17** Command Line Interface (CLI).

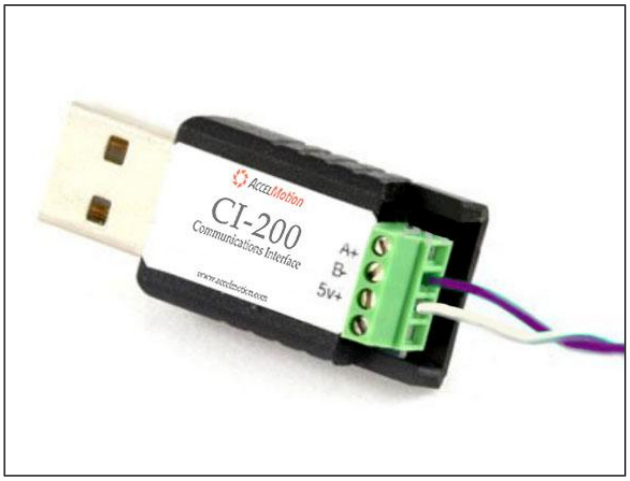


Figure 3 - CI-200 Communications Interface for DC2M17

Connection Diagram

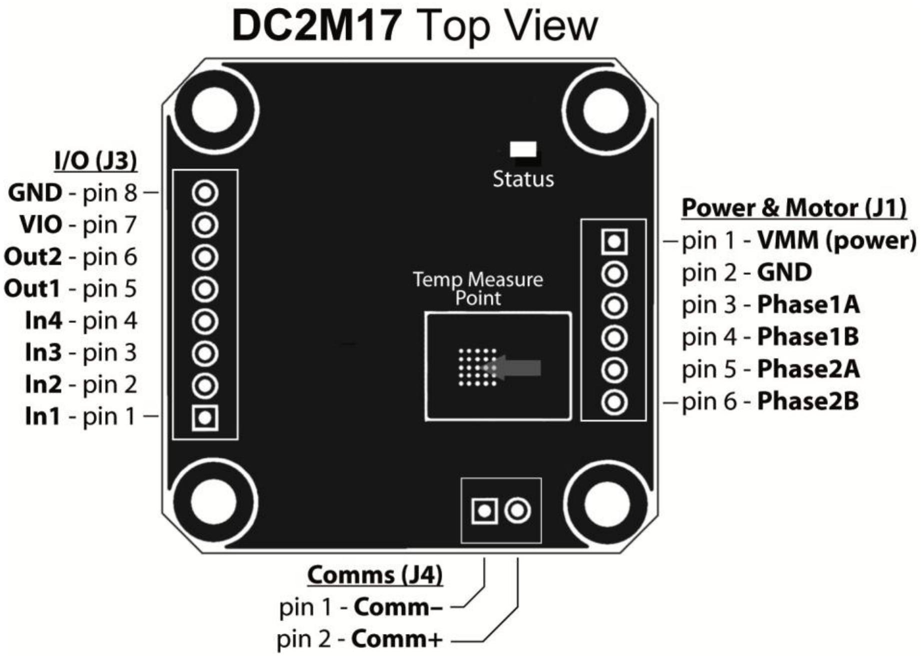


Figure 4 - DC2M17 Connection Diagram

Mating Connectors

Below are the part numbers for construction of DC2M17 mating cable assemblies. These components are readily available through a variety of distributors:

		Crimp Terminals		Housing	
Motor and Power	**J1**	**08-50-0113** x 6	Molex	**22-01-2067** x 1	Molex
Communications	**J4**	**08-50-0113** x 2	Molex	**22-01-2027** x 1	Molex
Input/Output	**J3**	**08-50-0113** x 8	Molex	**22-01-2087** x 1	Molex

Suggested Wiring Color Codes

Signal	Pin	Color	Signal	Pin	Color
Comm-	J4-1	Grey	Input 1	J3-1	Brown
Comm+	J4-2	Violet	Input 2	J3-2	Red
VMM	J1-1	White	Input 3	J3-3	Orange
Ground	J1-2	Black	Input 4	J3-4	Yellow
Phase1A	J1-3	White/Green Stripe	Output 1	J3-5	Green
Phase1B	J1-4	Green/White Stripe	Output 2	J3-6	Blue
Phase2A	J1-5	Red/Black Stripe	VIO	J3-7	White/Red Stripe
Phase2B	J1-6	Black/Red Stripe	Ground	J3-8	Black

Overview

All AccelMotion DC-series controller/drivers provide efficient and powerful control of a single stepper motor. Each unit consists of: a controller section which executes motion commands; and a driver section, which is tailored to the specific drive power and mounting method of that model (see Figure 5 - DC2M17 Functional Block Diagram The Communications Bus connects the DC-series controller to an external computer, which is used for programming and debug, but optionally can be used to control or query the controller directly.

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 draw on those parameters, and programs can be run from that main memory. Programs can be permanently stored by transferring into non-volatile (NV) memory, so that the parameters and programs are always available - see later sections on Programming and Program Mode.

In **Program Mode**, the controller executes commands in line-numbered order, but control can 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.

Additional functionality can be attributed to the four general-purpose input ports, including **Go**, **Stop**, **Limits**, **Jog**, and **Home** - see the `U` (Set Port) Command to configure ports for these functions. User inputs ports can be configured to external voltage levels by connecting an external supply to the VIO pin.

Two output ports are provided to drive external devices (relays, etc.), or data at logic levels.

The driver section provides the power to drive a stepper motor to the step location demanded by the controller. The power level is programmable (Y Command) for motion (Run) and when stopped (Hold); Run current is used when moving, transitioning automatically to Hold current once movement is ended. Stepping parameters, such as microstepping mode, velocity, acceleration, etc. determine the steps/revolution, speed, locations, etc. (as long as the motor output power exceeds the torque requirement). 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.

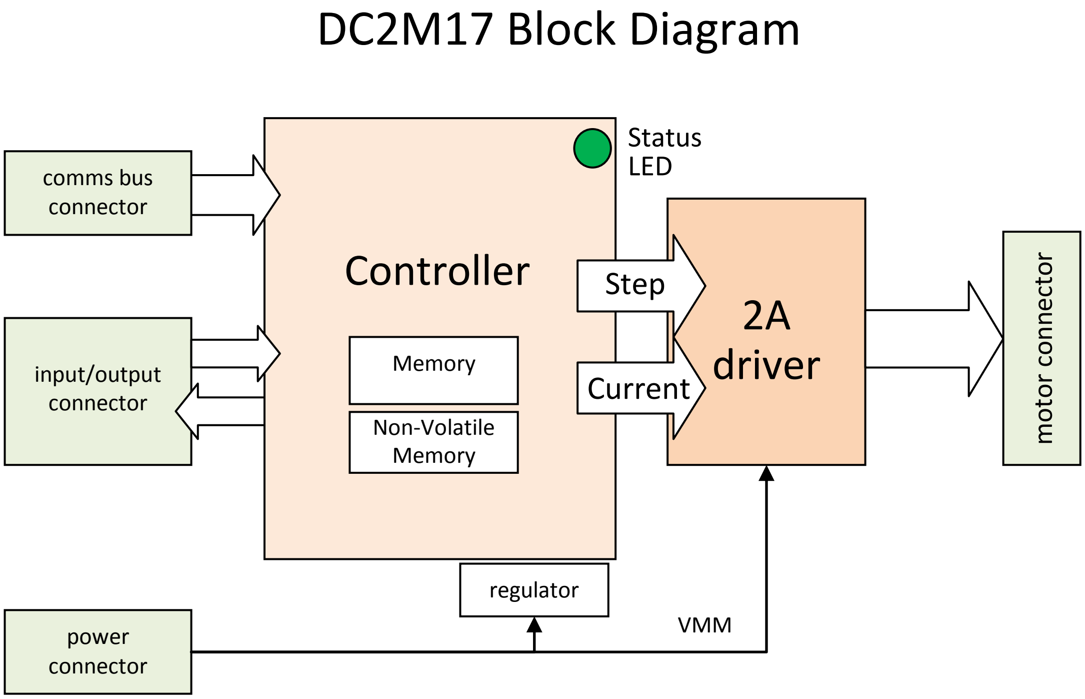


Figure 5 - DC2M17 Functional Block Diagram

Operational Modes

All AccelMotion DC-series controller/drivers can be operated two ways: IMMEDIATE mode, where commands are provided externally via the communications interface and are executed in real time; and as PROGRAM mode, where commands previously stored internally are executed, programmatically.

Immediate or Direct Control Mode accepts commands via an external computer or controller, and is used when requiring integration with control systems or when multiple-axis control from a high-level host is required. Commands and responses come through the Communications Bus and are executed in real time. Manual entry using a keyboard and terminal window, or automatic operation from a host script, are examples of Immediate Mode operation. This mode can address one or many units (axes). Some commands are unavailable in Immediate Mode.

Program Mode executes commands directly from a unit's internal memory, using programmed values and making conditional decisions based on its Input Ports and motion (see "L" Command). Programs can be started from Immediate Mode (see "P (Program Entry Mode - Entry/Exit)" command), or one can be 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 `<ESC>` character, or the STOP input port function). During Program Mode the communications port is ignored except for the `<ESC>` character.

Programs are stored internally in main memory, and can be moved permanently into non-volatile (NV) memory (see "S" command) – NV program memory can be recalled back to Main memory by the "C" Command, or at each system reset (see *^C (Reset) Command* and *Parameters & Defaults* section).

Automatic Program Mode executes a stored program at power-up (or 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 can be programmed and controlled via the AM Communications Bus. Two communications and control topologies are available: SINGLE and MULTI-AXIS.

Single-Axis Mode provides an easy way to control a single DC-series controller from a host PC or in Program Mode. This mode is perfect for single-axis systems, and has advantages: addressing/naming isn't required; axis name prefixes aren't required for each command; and a greater range of commands is supported. Naming does require the use of Single 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 prepended to the command string to identify the target controller*; and some commands are unavailable in Multi-Axis mode. Multi-Axis mode can be used for single unit operation, 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 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. AccelMotion DC-series controllers are *not* compatible with Advanced Micro Systems DCB- or MAX-series controllers communications busses or standards (using 9600 baud).

Parameters & Defaults

The Examine command (X) displays the current parameter values. These parameters are located in Main Memory and are used by the control firmware to set characteristics that the controller and driver will use to operate the step motor, and utilize optional input ports functions (Jog and Home).

The current parameter values are retrieved into Main Memory, at every power-up or reset time, 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 then be stored back into non-volatile memory as new defaults using the "S0" command, if desired.

Figure 6 - *List of Commands with Parameters, showing Factory Defaults* shows the list of parameter command codes and description, and the value the parameter will have from the factory, or if it is reset to factory default values – see command C for more info.

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 1.5A full-power)	25/50
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/I2/I3/I4)	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 6 - List of Commands with Parameters, showing Factory Defaults

See the ***Commands*** Section for more information regarding parameters as well as storing defaults into non-volatile memory using the "S" command, and recovering defaults with the "C" command, or reset.

Quick Start

This outlines the minimum steps required to make a **DC2M17** operational, using Operational Modes mode:

1. Connect the **main power supply** to connector J1 pins 1 and 2, with an output voltage in the range from 9VDC to 35VDC. Connect Pin 1 to the positive voltage (VMM) and Pin 2 to Ground. The power supply must be capable of providing between 1A and 2A of current, depending on the supply voltage and current settings required. See *Power Supply* section for more data.
2. With power off, connect an appropriate size **stepper motor** to the four motor connections of J1. Typical motor connection diagrams are shown in the *Stepping Motor* section. Be sure to insulate all motor leads (and unused leads) to prevent shorts to ground or to other motor leads.
3. Wire the 2-pin **communications connector** from the CI-200 Interface into J4. Make a connection between the GND of the DC2M17 and the GND terminal of the CI-200, to provide a shared ground level between all devices (refer to Grounding). Connect the other end of the CI-200 Interface to the USB port of the host computer. Insure the respective driver is installed correctly – see the CI-200 manual for details.
4. **Start a terminal application** on your computer (any program that allows you to send data through a serial connection; for example TeraTerm, Hyperterminal, Console, etc.). AccelMotion provides the free **AccelCom** software (on the AccelMotion website), which is an on-screen terminal with specialized buttons for Modes, I/O, etc. Open the configuration settings of your terminal and select the respective 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).
5. Apply power to the DC2M17. The green LED will illuminate, and flash slowly.
6. Type `<space>` to connect in Immediate Mode. The controller will respond with its ID, which indicates that communication is established, and will appear as: `AccelMotion DC2M17 V<x>`
7. Set safe motor **hold current** and **run current** settings, using the “Y” **command**. Make sure all "Y" command settings are compliant with the specs of the motor you are using, and that the actual current does not exceed the current rating of the motor.
8. To verify motion functionality, type a simple **motion command**: for example, **+400** will move the motor 400 steps in positive/plus direction. The entry will not echo as you type, but at the end the entire command will be echoed before the response (if any).

Type	Echo/Response	Remark
`<space>`	`AccelMotion DC2M17 Vx`	space starts connection and shows ID
`+400↵`	`+400`	motor moves 400 steps, in positive direction

9. To save these settings into non-volatile memory (data which is restored when unit is powered up or when a reset occurs), type “S0” followed by enter. See “S” and “C” commands 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 35VDC or be less than 9VDC

The selected motor must be compatible with the drive specification

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

Operating the DC2M17

Entering and Using Commands

Commands are represented as strings of text, and interpreted through the command-line interface (CLI) of the controller. Commands can be executed via the communications port in real-time, or interpreted from main program memory inside the controller once a program exists and Program Mode is entered.

Command string structure and data fields are shown for each command in its specific command description, in the Commands section. The data fields consist of decimal data values separated by spaces, and many of the data values are optional. For most commands, if the data field is not provided then the command is interpreted as a query of the current parameter value for that command; that queried value is returned to the host via the communications channel.

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:

- **Characters are echoed to the host on a line-by-line basis** - see Command T (Echo Mode)
- **Only one command may be entered per line**
- **Each Command is terminated by Enter (from keyboard), or `^M` (from script)**
- The command line may be edited using backspace, as characters are typed
- The 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 two parameter commands

Basic Communications

Before establishing first communications, the host must be connected to the DC-series controller via a RS-485 communications adapter (such as a CI-series adapter from Accel**Motion**), and some software method is needed to allow communications to and from the unit. See *Communications* section for more information on RS-485 communications adapters and wiring.

Most customers start in **Immediate 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 examples exist including TeraTerm, Console, etc. All modes other than Automatic Program Mode will require the host, adapter, and communications software.

Immediate Mode start-up

Follow these steps:

1. Power ON, then type `<space>` (causes **Immediate Mode** 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 enter ↵ is sent
3. The motor should move a small amount (1000 steps, in the negative direction)
4. Type the "**`Z`**" command. This command responds with the axis position (-1000)

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 in which the next command (and parameters) will reside before – some take more space than others, and this space is shown as Size in each command's reference data.

Type	(line#)	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 (25)*

Now, use "Q" command to display the stored program for your 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)*

Program Editing example

Example: It is desired to change instruction in location 13 from -1000 steps to -5000 steps. In this example we begin editing at the exact location (13) and we exit Command Entry Mode with <ESC>:

Type	(line#)	Echo/Response	Remark
P13 ↵		`P0`	**Enter Program Entry Mode** at location 13
-5000	`13`	`-5000`	Move 5,000 steps in the minus direction, relative to origin
<ESC>			**Terminates Edit mode** *(remember to Enter before the ESC)*

In Program Mode, all motion commands are automatically stalled (waiting) if a 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 usage for setting execution timing – use W0 prior to any command that should wait for completion of a previous motion

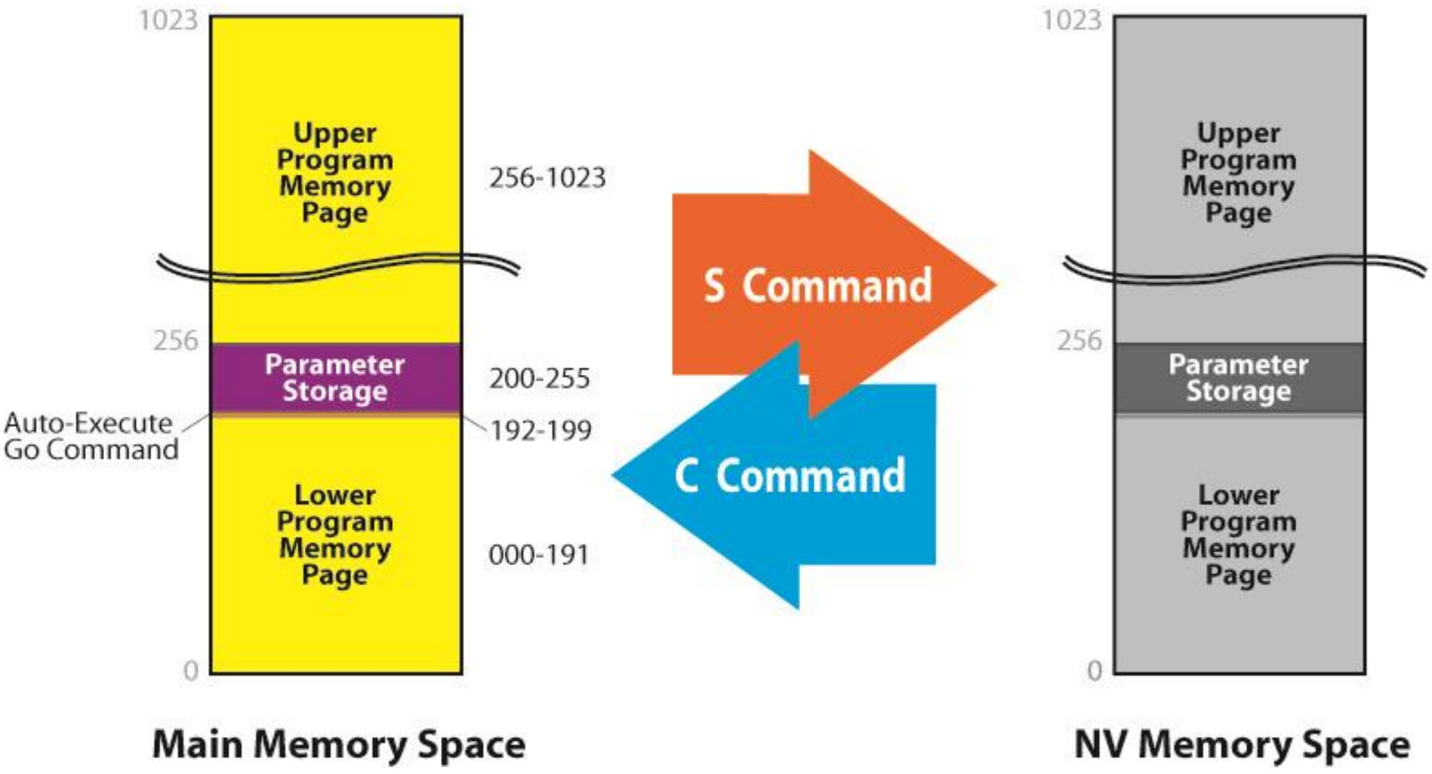
Command Entry Mode Highlights:

- 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 - same reset if illegal command characters 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 and Entry Mode is exited, a program can be saved to non-volatile memory using the "S1" command so that it's retained even after powering down or resetting.
- 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 128; 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. The space available from address 192 to 199 is too small for any reasonable program size – 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.

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. This command responds with the axis position (-1000)

Multi-Axis Communication Note

The DC2M17 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 DC2M17 is powered from a single DC power supply (PS). The power supply is connected via the J1 Connector, pins 1 and 2. The input voltage must be in the range of 9VDC to 35VDC.

Pin	Function
J1 - pin 1	VMM (High drive voltage)
J1 - pin 2	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 power (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 1.5A of max output phase current (DC2M17 max without optional case), we recommend a supply with output power of 30W (Watts) minimum – at 2A, the minimum requirement is 40W.

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
30W	30V	30W / 30V = **1 Amp @ 30V**
30W	24V	30W / 24V = **1.25 A @ 24V**
30W	12V	30W / 12V = **2.50 A @ 12V**
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 while power is applied

***The power supply voltage, including ripple and line voltage fluctuations,
must not exceed 35VDC or be less than 9VDC***

Individual axis units should be fused independently, for fault isolation

***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 DC2M17 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 half a 6-lead center tapped motor with the DC2M17, however the 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 DC2M17 and a Stepper Motor.

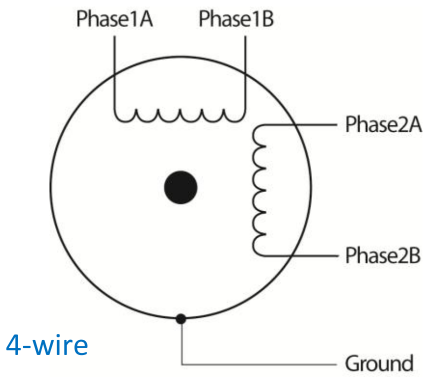
Pin #	Description	Function
1, 2	1A, 1B	Phase 1 of the Stepping Motor – connect between Pin 3 and 4 of J1.
3, 4	2A, 2B	Phase 2 of the Stepping Motor – connect between Pin 5 and 6 of J1.

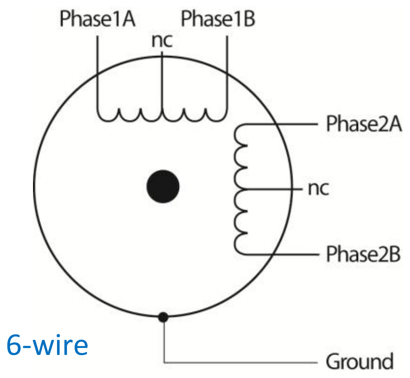
Drive Current

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.

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 Accel**Motion**.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: 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 when power is applied

Select the stepper motor for your application first in reference to 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

Status LED

The DC2M17 indicates its status via a single LED status indicator.

The status display is Green when the unit is powered. A flashing “heartbeat” is visible at all times, from a very slow rate (approximately one flash/sec) when the unit is not moving up to higher flashing rates (over 50 flashes/sec) as the stepper step rate rises. This function is meant to visibly indicate operation of the driver unit.

If for any reason the indicator is illuminated RED or YELLOW, turn off power immediately – this indicates that the on-board fuse has been blown. This failure is due to an electrical short, caused by:

- 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

The main power protection fuse is meant only as a last-resort protection for equipment safety, and is not replaceable or repairable. There are no user-serviceable parts on the DC2M17.
To repair any damage, contact AccelMotion to return the unit for repair, where applicable.

Connectors

The DC2M17 requires three connectors for all its functions:

- A Power/Motor Connector
  - connects the main motor voltage (VMM), which is used to drive the motor 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 Communications Connector for DATA+ and DATA- leads of the AM Communications Bus
- An Input/Output Connector for the four inputs and two outputs (plus VIO and ground)

Each connector has physical keying to prevent the incorrect insertion of each mating connector. Each connector is a different size so as minimize confusion for installation or serving technicians, and the optional cable harness is color-coded for all signals, to ease the installation and assure correct connections.

Input / Output Subsystem

DC-series controllers feature input and output ports. These connections operate in an industry-standard **"low-drive" mode**, where the ACTIVE state of any input or output is low-voltage (ground), and any port's INACTIVE state is high-voltage – either VIO (IO voltage level), or disconnected (and internally pulled-up to VIO).

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

Input Port Electrical Configuration

All digital inputs are evaluated against ½ VIO voltage, to determine the actual digital value - below ½ VIO is INACTIVE and below is ACTIVE. This circuit supports a wide input voltage range for digital IO, yet provides noise immunity at the selected voltage. An internal 50k Ohm pull-up resistor to VIO is provided to force inputs normally high (inactive), and need to be pulled low by the user to activate. An input R-C filter is also included and provides about a 2kHz cut-off frequency.

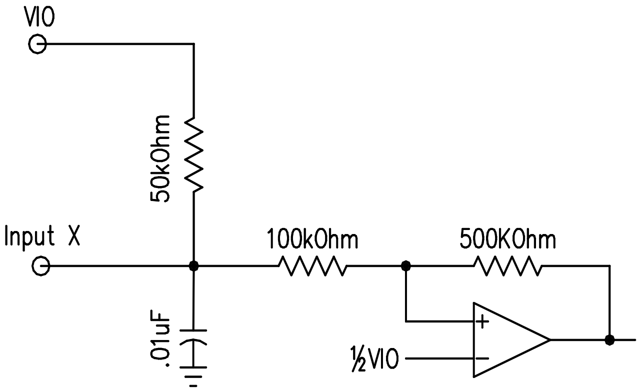


Figure 7 - Input Equivalent Circuit

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 8 - 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 (1/2 VIO) the input is ACTIVE, and when it is above the threshold it is INACTIVE. When the switch is closed, 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.

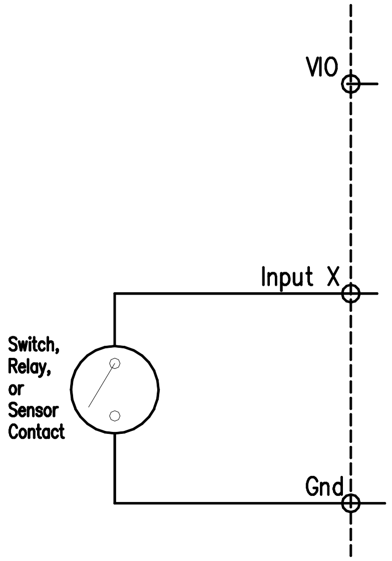


Figure 8 - Example of Switch Contact Input

When an input port is connected to a output from a computer, a PLC, or an active sensor, that device will drive the input at its supply voltage range; at higher I/O voltage levels (12V, 24V, etc.) the input section requires a reference point to create a ½VIO value, because the internal default VIO value is 5V. In these cases, **connect the external I/O voltage to the VIO input** – the threshold circuit uses the higher of either 5V or the external pin value, and divide that voltage by 2 for the threshold voltage.

Figure 9 - 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.

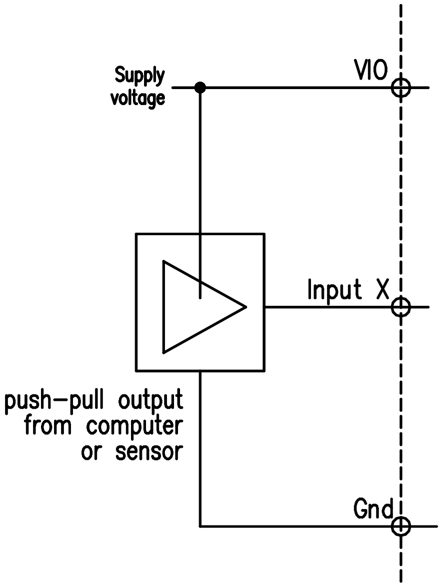


Figure 9 - Example Active Input using external VIO supply

Input Ports configured for Fixed Function

Each digital Input Port (IN1 - IN4) can be configured for a fixed function from the following set; **Limits, Home, Jog, Go,** and **Stop** provide fixed functions, whereas **User Input** selects general-purpose status accessed by conditional branching in a program. The fixed functions are detailed in the "U" Command.

The factory default setting for all Input Ports is **User Input** mode.

HOME

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

GO

The GO input 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, whether held or reasserted.

ALL STOP

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

JOG+ and JOG-

Jog inputs are used to manually change the motor position, when the input is ACTIVE. Both jog inputs stop any existing motion command 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 motion, in the direction indicated 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 the mechanical endpoint from the limit switch, either manually or automatically.

AccelMotion recommends Limit Switches for protection of mechanical systems.

Input Port Read

All input ports are readable in **Immediate Mode** by the A (Input Port Read) 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) 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 correct raw port status.

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.

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 are configurable to drive various optional fixed functions (U command)
- Input ports are accessed via L (Loop) in Program Mode and A (Read Inputs) command in Immediate/Direct Mode, and access raw input data 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 a higher VIO voltage and threshold by connecting an external VIO supply to the VIO input
- Dry-contact devices can operate correctly even with external VIO supply connected - so, 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.**

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 DC2M17 provides two 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 output is connected to a computer or logic input (and 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 with 5V).

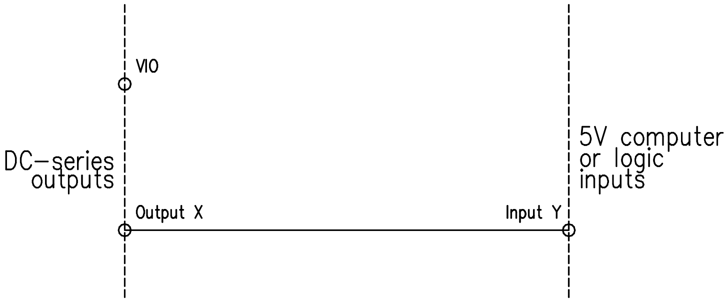


Figure 10 - Output port driving 5V computer/logic input

Logic Mode – Computer/Logic Voltage higher than 5V: if the computer or logic input operates on a higher positive voltage rail, *then connect that high-level supply voltage to the VIO terminal*, so that the INACTIVE signal level will be correct – the external positive voltage will be substituted for the internal 5V VIO, thus maximizing the voltage swing of the logic signal.

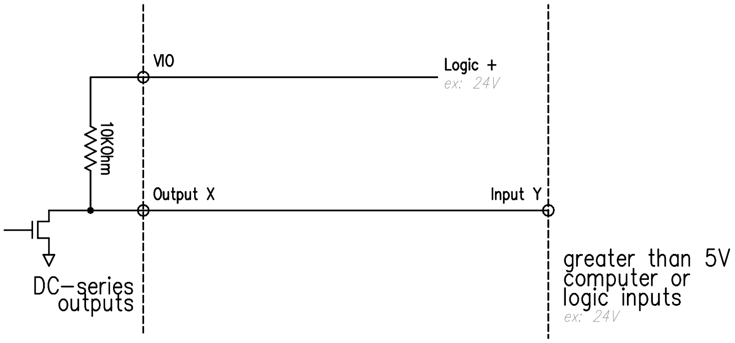


Figure 11 - Output Port driving high-voltage logic inputs (example: 24V)

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 during ACTIVE period. Connect the device between the rated positive 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).

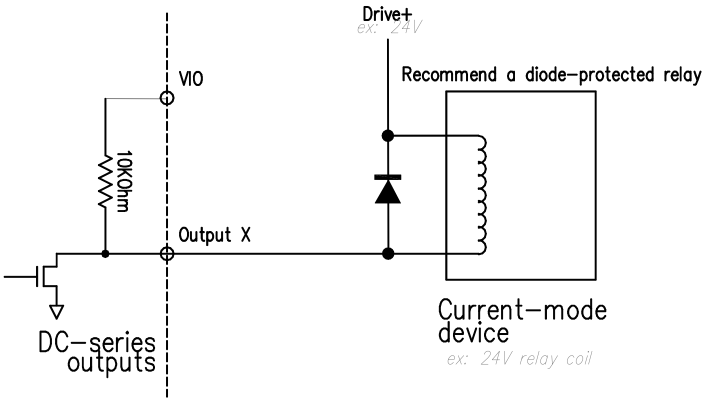


Figure 12 - Driver-Mode Output Port configuration (with diode protection)

Controlling Output Ports

Output ports are actuated via the “w” Command, **Immediate** and **Program Modes**. All output port control bits are referred by a single “w” parameter; multiple outputs can be updated at one time, or just one at a time, as needed.

The “w” command issued without a parameter will read back the status of the outputs, in **Immediate Mode** only.

Output 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 are controlled by the “w” command, and can be read back with same command

Important Notes

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

Assure that all outputs 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 12 - 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 13 - AM Communications Bus Block Diagram shows the basic blocks used in communicating over the Comms Bus.

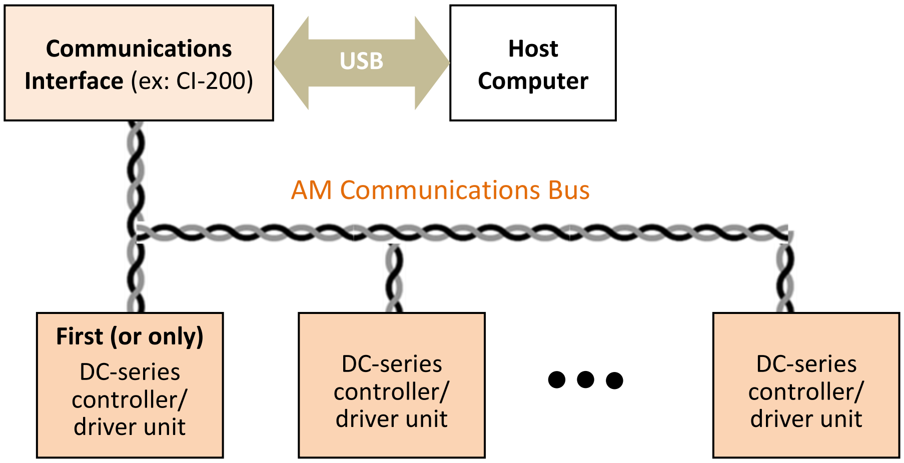

Communications Interface (ex: CI-200)
USB
Host Computer
AM Communications Bus
First (or only) DC-series controller/ driver unit
DC-series controller/ driver unit
DC-series controller/ driver unit

Figure 13 - AM Communications Bus Block Diagram

Accel**Motion** makes available the **CI-200 Communications Interface,** which interfaces standard computer hosts (via USB) and the AM Communications Bus. Other communications adapters may be used as long as they convert to two-wire RS485, expose a ground reference pin, and include signal termination; but 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**; all terminal or scripting software and the communications interface must be set to those settings for correct operation with any DC-series unit. Note also that Accel**Motion** 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.

AccelMotion has anOptional Accessoriesavailable, which can make all wiring easier.

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 the *Connection Diagram* for the connection pins needed.

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 available to that single controller unit, with no need for addressing. 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 32 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 one DC-series unit 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 with multiple axes in close proximity, and with 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, Accel**Motion** **strongly recommends that all systems utilize the High-Reliability Twisted-Pair and Shielded Wiring Model,** to assure the highest quality connection and lowest exposure to induced electrical currents from equipment or other noise sources.*

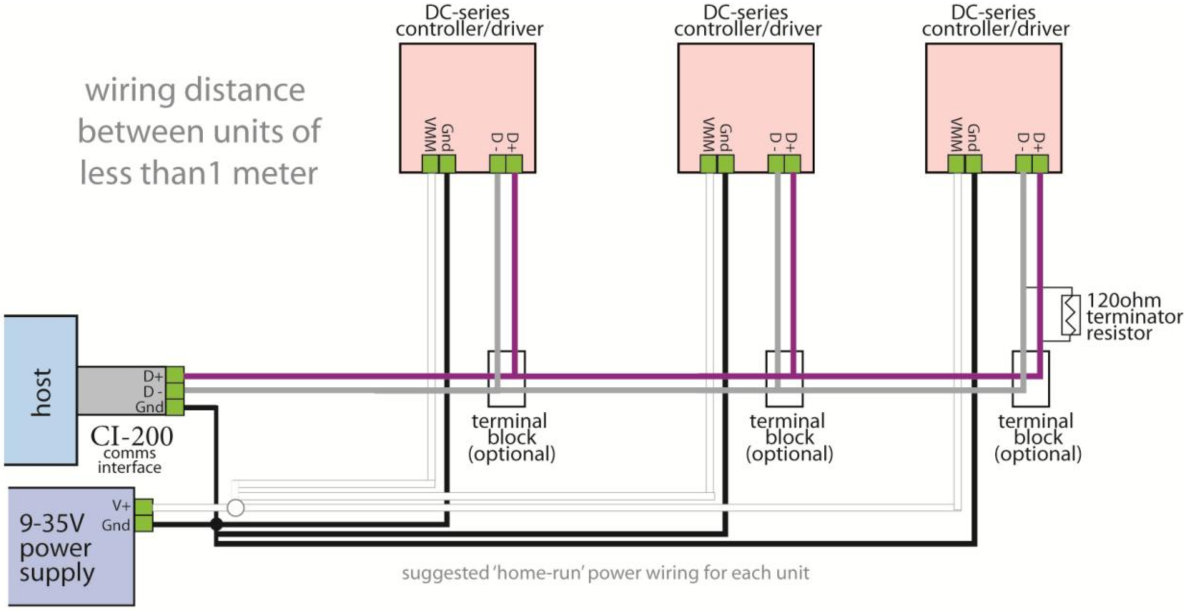

DC-series controller/driver
DC-series controller/driver
DC-series controller/driver
wiring distance between units of less than1 meter
VMM
Gnd
D-
D+
120ohm terminator resistor
host
D+
D -
Gnd
CI-200
comms interface
terminal block (optional)
terminal block (optional)
terminal block (optional)
9-35V power supply
V+
Gnd
suggested 'home-run' power wiring for each unit

Figure 14 - 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*.

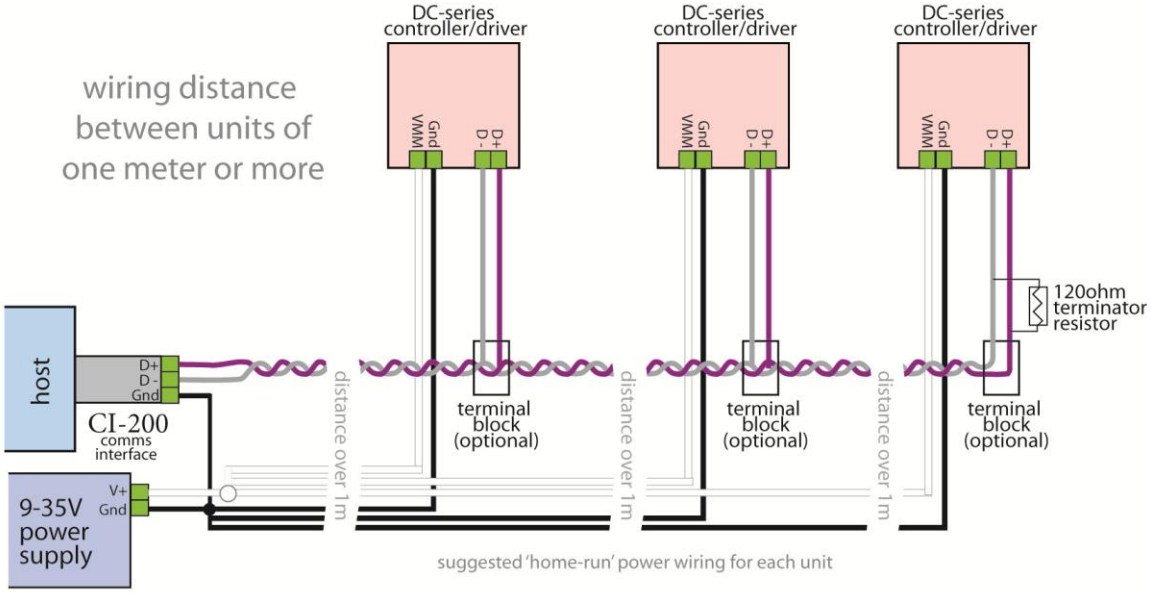

DC-series controller/driver
DC-series controller/driver
DC-series controller/driver
wiring distance between units of one meter or more
VMM
Gnd
D-
D+
120ohm terminator resistor
host
D+
D -
Gnd
CI-200
comms interface
terminal block (optional)
terminal block (optional)
terminal block (optional)
distance over 1m
distance over 1m
distance over 1m
9-35V power supply
V+
Gnd
suggested 'home-run' power wiring for each unit

Figure 15 - Twisted-pair and Shielded Wiring example (recommended)

Multidrop busses like the AM Comms Bus are organized in a 'daisy-chain' or '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 which would not be ideally protected by the terminators and connection topology. 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 CI-200. Near-side termination is already 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***
	Example *(axis) Command *(data1) *(data2) (↵)	***Data 1*** *(Range)	***Data 2*** *(Range)	***Result***

Where:

Command:	Keystroke or character mnemonic for the command
Function:	Functional description of command
Type:	Immediate Mode = Direct execution 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 as a result of command execution or examination

** 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*
- Multi-Axis 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 ends **Program Entry Mode**
- Global Broadcast commands affect all connected units at the same time, once issued
- Global Broadcast commands are single-character, and *don't need termination character* (Enter↵)
- Characters entered via communications port are echoed on a line-by-line basis (see Command T (Echo Mode))
- Only one command may be entered per immediate entry or program line
- Non-Global **Immediate Mode** Commands are terminated by Enter↵ (keyboard), or ^M (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 two parameters, where two are present

^N (Name Axis)

<table>
<tr><td rowspan="2"><b><i>Command</i></b>
^N</td><td colspan="2"><b><i>Function</i></b>
Name Controller</td><td colspan="2"><b><i>Type</i></b>
Immediate</td><td><b><i>Size</i></b>
N/A</td></tr>
<tr><td><b><i>Example</i></b>
^N</td><td colspan="2"><b><i>Data 1</i></b>
none</td><td><b>Data 2</b>
none</td><td><b><i>Result</i></b>
none</td></tr>
</table>

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.

Each DC-series controller that is intended to become part of a Multi-Axis network must be uniquely named — if more than one axis had the same name then commands would be duplicated, and responses and echoes would overlap and cause serious communications bus conflicts.

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.

Upon reception of **^N** the controller responds with "NAME?" Type 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)

<table>
<tr><td rowspan="2"><b><i>Command</i></b>
^P</td><td colspan="2"><b><i>Function</i></b>
Enter Multi-Axis Mode</td><td colspan="2"><b><i>Type</i></b>
Global Broadcast</td><td><b><i>Size</i></b>
N/A</td></tr>
<tr><td><b><i>Example</i></b>
^P</td><td colspan="2"><b><i>Data 1</i></b>
none</td><td><b><i>Data 2</i></b>
none</td><td><b><i>Result</i></b>
none</td></tr>
</table>

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.

From one to 32 controllers may be connected in a Multi-Axis network, and all will react to the ^P command together.

When multiple DC-series controllers (axes) are connected, the host must place the system into Multi-Axis mode immediately after power up or ^C reset — signing-in to Single-Axis mode with the <space bar> in this case would result in erroneous operation.

If in Multi-Axis mode multiple axes may be connected and since all are affected no echo is sent, which would otherwise result in 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 allowing 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).

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

^C (Reset)

Command	**Function**		**Type**	**Size**
^C	Reset Controller(s)		Immediate, Global Broadcast	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
- 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)

<table>
<tr><td rowspan="4"><b><i>Command</i></b>
<b>ESC</b></td><td colspan="2"><b><i>Function</i></b></td><td colspan="2"><b><i>Type</i></b></td><td><b><i>Size</i></b></td></tr>
<tr><td colspan="2">Terminate Operations</td><td colspan="2">Immediate, Global Broadcast</td><td>N/A</td></tr>
<tr><td><b><i>Example</i></b></td><td><b><i>Data 1</i></b></td><td><b><i>Data 2</i></b></td><td></td><td><b><i>Result</i></b></td></tr>
<tr><td>&lt;ESC&gt;</td><td>none</td><td>none</td><td></td><td>none</td></tr>
</table>

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, ends Program Mode, and causes the controller to revert to the idle state in Immediate Mode, waiting for a new command.

- 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 exits **Program Mode**
- If sent in **Multi-Axis Mode**, all programs stop on all connected units and exit **Program Mode**
- If sent during **Program Entry Mode**, the controller will immediately exit Program Entry Mode
- Output port settings and output port states are not affected
- The controller will echo a "#" character.

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

If in Multi-Axis mode, the command will terminate motion on all axes instantly – it is not possible to send this command to one axis alone.

If in Multi-Axis mode when the command is sent, unit will stay in Multi-Axis mode, awaiting a command

No echo character is sent, which in some cases could result in potential bus conflicts.

@ (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.

| (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, and in
  - 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)

<table>
<tr><td rowspan="2"><b><i>Command</i></b>
^</td><td colspan="2"><b><i>Function</i></b>
Read Moving Status</td><td colspan="2"><b><i>Type</i></b>
Immediate</td><td><b><i>Size</i></b>
1</td></tr>
<tr><td><b><i>Example</i></b>
(Name) ^ ↲</td><td colspan="2"><b><i>Data 1</i></b>
none</td><td><b><i>Data 2</i></b>
none</td><td><b><i>Result</i></b>
Status</td></tr>
</table>

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 - Temperature)

<table>
<tr><td rowspan="2"><b><i>Command</i></b>
{
<i>(open brace)</i></td><td colspan="2"><b><i>Function</i></b>
System Status Read (Temperature)</td><td colspan="2"><b><i>Type</i></b>
Immediate</td><td><b><i>Size</i></b>
1</td></tr>
<tr><td><b><i>Example</i></b>
(Name) <b>{</b> ↲</td><td colspan="2"><b><i>Data 1</i></b>
none</td><td><b><i>Data 2</i></b>
none</td><td><b><i>Result</i></b>
Status</td></tr>
</table>

The **{** command (Open brace, or 125 hex) responds with a result string of the current System Status, in **Immediate Mode**. The current temperature of the unit is returned in degrees centigrade. The temperature is measured near the hottest location (the motor driver).

This value should never go above the maximum allowed value shown in the specifications subsection *Thermal and Mechanical Specification*. More information can be found in the

Thermal Considerations **section.**

This request can be made in Single-Axis Mode, or Multi-Axis Mode by addressing the request to a particular axis controller, by name.

Note: in Program Mode the motion commands allow program execution to occur during motion, or hold commands until after motion has completed with the W0 command. See the reference pages for these commands for details.

+ (Index in Plus Direction)

Command	***Function***		***Type***	***Size***
+	Index (step) relative to current position in Plus Direction		Immediate, Program	5
	Example	***Data 1***	***Data 2***	***Result***
	(Name) + n ↵	n steps (0 to 2,147,483,647)	none	none

Step (index) in the positive direction for the specified step count, relative to the axis current position. [Refer to the stepper motor section for information on the nature of 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 in-process motion is complete
2. Energize the motor winding, using the "Y" command RUN current parameter
3. 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
4. Accelerate using slope set by the K (Ramp Slope) *Accel* parameter, to V (slew velocity)
5. Continue stepping at the V Slew Speed, until the calculated deceleration point is reached
6. Decelerate, at a rate determined by the K *Decel* parameter, to a stop at the final index value
7. Once a motion is complete, if another motion is not commanded within the settling delay period (see "E" Command) then output power changes to the HOLD current setting after the period ends – if another motion does occur: repeat step 8 and again wait for idle period

In Program Mode, all motion commands are automatically stalled (waiting) if a 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 usage for setting execution timing – use W0 prior to any command that should wait for completion of a previous motion

– (Index in Minus Direction)

Command	***Function***		***Type***	***Size***
–	Index (step) relative to current position in Minus Direction		Immediate, Program	5
	Example	***Data 1***	***Data 2***	***Result***
	(Name) – n ↵	n Steps (0 to -2,147,483,648)	none	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 entered as absolute-value.

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

A (Input Port Read)

Command	***Function***		***Type***		***Size***
A	Input Port Read		Immediate		1
	Example	***Data 1***		***Data 2***	***Result***
	(Name) A ↵	none		none	Input State

The "A" Port command displays the input ports, represented in one Input State value (**Operational Modes** only). It is used without a parameter; the value of the Input Ports are returned as "input state" value, representing port data as a binary value, where the low bit corresponds to I1; if an "A" command returns "`A9`" (decimal 9 = binary `1001`) then I4 and I1 are ACTIVE, and I3 and I2 are 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) on the input. See *Input Port Electrical Configuration* for more details.

Input Port encoding into Input State Value				
Port	I4	I3	I2	I1
Decimal value	8	4	2	1

Each of the available input port signals (I1, I2, I3, I4) can either be assigned to a specific fixed function using the U (Set Port) Command, or be used as a general-purpose input – the factory default for all input ports is *general-purpose user input*. General-purpose inputs are readable by "A" Command in **Immediate Mode**, and control conditional branching via the "L" Command in **Program Mode**.

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 – 28,000)		none	none

The "B" command modifies the Jog Speed value – the speed which the motor will jog 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 RAM memory; retain it permanently in NV memory with the S Command.

If the command is issued without data in Immediate Mode, 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 in Program Mode, it will change the Jog Velocity value immediately.

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

C (Clear and Restore Main Memory)

Command	**Function** Clear and Restore Main Memory		**Type** Immediate	**Bytes** N/A
C	**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 the "C" command without a parameter is equivalent to "C0", and would restore NV parameters.

Use the "S" command to move 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)

<table>
<tr><td rowspan="4"><b>Command</b>
D</td><td colspan="2"><b>Function</b>
Divide Speed Value (default= 4)</td><td colspan="2"><b>Type</b>
Immediate, Program</td><td><b>Bytes</b>
2</td></tr>
<tr><td><b>Example</b></td><td colspan="2"><b>Data 1</b></td><td><b>Data 2</b></td><td><b>Result</b></td></tr>
<tr><td>(Name) D ↲</td><td colspan="2">none</td><td>none</td><td>D value</td></tr>
<tr><td>(Name) D n ↲</td><td colspan="2">Divider (1-255)</td><td>none</td><td>none</td></tr>
</table>

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

As the divider “n” is increased, parameters for all internal speeds must be adjusted to maintain the desired output step speed.

The D command can be used to achieve very low speeds, or to achieve smoother Accel and Decel ramps. When D is larger, more discrete steps are used when ramping from one specific motor speed to another.

This command is usually implemented only once when assigning the default parameter set, however, it may be changed within a program as needed. *The D value should never be changed while moving*.

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

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

E (Delay to Hold Time)

<table>
<tr><td rowspan="4"><b>Command</b>
E</td><td colspan="2"><b>Function</b>
Hold Settling Time (default=100)</td><td colspan="2"><b>Type</b>
Immediate, Program</td><td><b>Bytes</b>
2</td></tr>
<tr><td><b>Example</b></td><td colspan="2"><b>Data 1</b></td><td><b>Data 2</b></td><td><b>Result</b></td></tr>
<tr><td>(Name) E ↲</td><td colspan="2">none</td><td>none</td><td>E value</td></tr>
<tr><td>(Name) E t ↲</td><td colspan="2">5-255 (time x 10mSec)</td><td>none</td><td>none</td></tr>
</table>

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 setting is updated into main RAM memory; retain it permanently in NV memory with the 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, and if another command causes a step before the timeout has ended then the timer is reset. 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 and 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 E.

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**
	Find Home		Immediate, Program	4
F	**Example**	**Data 1**	**Data 2**	**Result**
	(Name) F n (d) ↵	n SPS (1-28,000)	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 spring-type 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 by issuing the F command; the first parameter is the desired 1st stage velocity toward HOME, and the optional second parameter is the requested starting direction toward the Home switch (if d is not provided, then 0 or positive direction 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 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, usually overshooting motion in direction d, and then automatically reverses direction and begins moving at rate I (Initial Velocity), more 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, producing a '1' from a low-voltage input and a '0' from an open switch (or a push-pull driven active position sensor) 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 'F' process shown above works in this 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.
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 0-1023 (excluding 200-255)	t (0-1)	none

The Go command is used to enter **Program Mode** and begin execution of a user-programmed sequence, starting at memory location “a”. Most programs will start at “0”, however, any starting address may be used.“G” can start execution from Immediate Mode to begin 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.

If optional parameter t is a 1 (one), **Trace Mode** is enabled; 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 an another “Go” is executed without the Trace attribute – so change all G branches in any debug code to Trace Mode.

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
```

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 redirect program flow to other locations, as programs cannot be placed in locations 200 – 255.

H (Microstepping Resolution)

Command	**Function**		**Type**	**Size**
	Resolution (default 1)		Immediate, Program	2
H	**Example**	**Data 1**	**Data 2**	**Result**
	(Name) H ↲	none	none	H value
	(Name) H n ↲	n 0-3	none	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
0	Full-Step
1	Half-Step (factory default)
2	¼ Step
3	⅛ Step
4	1⁄16 Step

This command can be issued in an interactive session or in a program. It can be even issued while motion is taking place and becomes effective immediately.

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
	Initial Velocity (default 2000)		Immediate, Program	3
I	Example	Data 1	Data 2	Result
	(Name) I ↲	none	none	I value
	(Name) I n ↲	n speed (0 to 28,000 SPS)	none	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 whatever direction the motion command (F, M, +, –, R+, and R–) demands. All velocity parameters, including the Initial Velocity is also 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
- Accelerate start point and decelerate end point with M constant velocity command
- Starting velocity for Home (F command), if Home Speed is greater than Initial Velocity

If the I command is issued without data in Immediate Mode, it will return the current setting of I (such as `I400/1`). This is especially useful in Multi-Axis mode where the X command is not available.

J (Jump to Address Multiple Times)

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

<b>J</b></td><td colspan="2"><b>Function</b>
Jump to Address Multiple Times</td><td><b>Type</b>
Program</td><td><b>Size</b>
4</td></tr>
<tr><td><b>Example</b>

(Name) J a n ↲</td><td><b>Data 1</b>

a address (0-1023)</td><td><b>Data 2</b>

n additional loops 0-255</td><td><b>Result</b>

none</td></tr>
</table>

The "J" Command is used (in Program Mode) to execute a jump to a program location *repeatedly* – 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 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:

	P 0	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.

K (Ramp Slope)

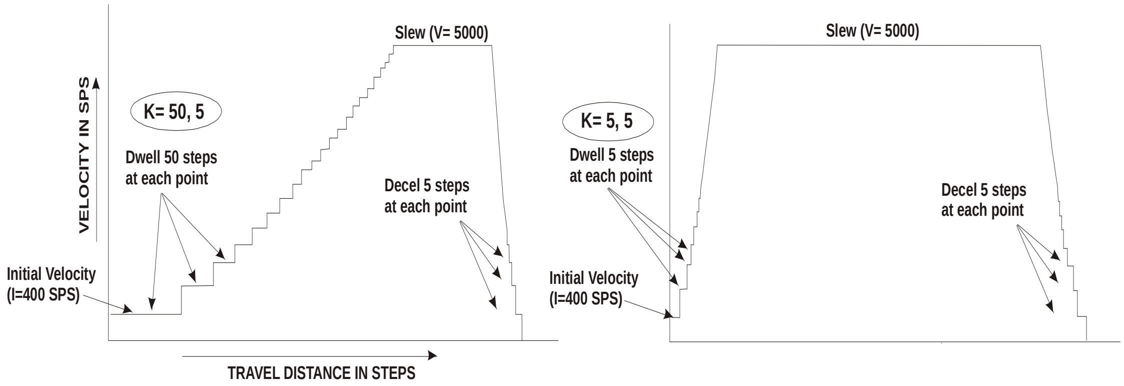
<table>
<tr><td rowspan="3"><b>Command</b>
<b>K</b></td><td colspan="2"><b>Function</b>
Ramp Slopes (defaults 5/3)</td><td colspan="2"><b>Type</b>
Immediate, Program</td><td><b>Size</b>
3</td></tr>
<tr><td><b>Example</b></td><td colspan="2"><b>Data</b> 1</td><td><b>Data 2</b></td><td><b>Result</b></td></tr>
<tr><td>(Name) K ↵
(Name) K (Accel) (Decel) ↵</td><td colspan="2">none
Accel (1-255)</td><td>none
Decel (1-255)</td><td>K values
none</td></tr>
</table>

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 values of initial (beginning) and slew (target) velocities, 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. A K of 0 will eliminate ramping, but is generally not recommended.

In practical applications, it is typically easier to decelerate a mechanical system than to accelerate a system. The separate decelerate parameter feature is a valuable time-saver when compared to controllers with fixed acceleration/deceleration times.

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 (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 (input port or motion-detect) and polarity are selected with one value, c, as follows:

Source	**Fall-through Condition c**	
	When source INACTIVE (port at VIO voltage)	**When source ACTIVE (port at 0V)**
I1	0	1
I2	2	3
I3	4	5
I4	6	7
"moving"	64 (fall through when not moving)	65 (fall through when moving)

The "L" command is valid only in **Program Mode**.

Following is an example which waits until *input port 3* is activate, 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 (also can use 2048) Fall through to next instruction when port 3 is active
5	+ 1000	Move 1000 steps in the plus direction.
10	P	Exit program mode.

Condition 64 can be used during motion to implement other tasks, such as blinking an output port indicator, and only continuing execution when the motion is complete. 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
	Move at Constant Velocity		Immediate, Program	3
M	**Example**	**Data 1**	**Data 2**	**Result**
	(Name) M (+/-) s ↲	s speed (±0 - 28,000 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 current velocity or the I Command Initial Velocity, whichever is closer to the target velocity; the motor will ramp up or down from that starting velocity using K command Accel, to the target constant 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 command or condition occurs. Ramp parameters (K Accel or Decel) may be modified prior to each velocity command, allowing different acceleration and deceleration ramp slopes. The "M" Command has the capability of decelerating from full speed in one direction, then accelerating to full speed in the opposite direction if desired.

Move Command motion is terminated by:

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

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

- (H) Microstepping
- (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 the M command used in a program; when executed, it starts an indexing motion at 500 steps/sec, then uses L (Loop Waiting on Condition) to detect when the axis mechanicals activate a detector (switch, sensor) on Input Port 1, and then immediately decelerate and stop 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 1        Loop Back to address location 3 (itself) until port 1 is ACTIVE (low)
7       @           Decelerate and stop program execution
8       P           Exit Program Entry Mode
```

O (Set Origin)

<table>
<tr><th rowspan="5">Command

O</th><th colspan="2">Function</th><th colspan="2">Type</th><th>Size</th></tr>
<tr><td colspan="2">Set Origin (position counter reset)</td><td colspan="2">Immediate, Program</td><td>5</td></tr>
<tr><th>Example</th><th colspan="2">Data 1</th><th>Data 2</th><th>Result</th></tr>
<tr><td>(Name) O ↲</td><td colspan="2">None (0 implied – clears to zero)</td><td>none</td><td>none</td></tr>
<tr><td>(Name) O a ↲</td><td colspan="2">a position (-2,147,483,648 to +2,147,483,647)</td><td>none</td><td>none</td></tr>
</table>

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 Position Counter may be modified by the O command without affecting the distance of a previous in-progress motion command, but the counter is updated immediately, perhaps leading to incorrect result, so we recommend updating the counter only after a W0 (wait for motion end).

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.

Hardware reset (^C) and power-on reset clears the Position Counter to “0”

The Z command is used to read the current value of the position counter.

P (Program Entry Mode - Entry/Exit)

<table>
<tr><th rowspan="5">Command

P</th><th colspan="2">Function</th><th colspan="2">Type</th><th>Size</th></tr>
<tr><td colspan="2">Program Entry Mode - Entry/Exit</td><td colspan="2">Immediate</td><td>N/A</td></tr>
<tr><th>Example</th><th colspan="2">Data 1</th><th>Data 2</th><th>Result</th></tr>
<tr><td>(Name) P ↲</td><td colspan="2">Implies address 0 (zero)</td><td>none</td><td>none</td></tr>
<tr><td>(Name) P a ↲</td><td colspan="2">a address (0-1023)</td><td>none</td><td>none, #</td></tr>
</table>

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 in Program Entry Mode transitions back to Immediate Mode. Program Entry Mode may also be terminated with the ESC (Global Abort) (escape) character. When in Program Entry Mode, commands and data entered are placed into Main Memory Program space for future execution, one command at a time.

It’s 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 the built-in editor. It is recommended that programs be written or documented on the host computer and then downloaded to each respective controller, due to the long-term application storage space as well as easier editing and viewing of the program on a computer. From a documentation perspective, this method is safest.

(continued)

When using the AccelCom editor, 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 in the editor:

```
P               enters Program Entry Mode
+1000           |
W250            |
-1000           | -------- THIS IS THE PROGRAM
W250            |
L0 100          exits Program Entry Mode
P               re-enters Program Entry at 100
P100            |
+100000         |
@               |
P               exits Program Entry Mode
```

If a mistake is made while entering code, the user may use the Backspace key, which will erase all characters of the current line, giving the user the opportunity to reenter the line. In single axis mode the line number will again be displayed after clearing with 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.

There are special address ranges shown below:

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

During program entry, addresses between 200 and 255 or beyond 1023 will be ignored.

See Programming section for a more program entry examples.

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, but always at power-on.

When entering or modifying programs with the P command, those program commands are entered into the Main Memory level. These programs can be saved permanently to NV memory using the “S1” command. All programs entered or changed into Main Memory are lost when powering down the controller; on power-up Main Memory is restored from NV memory.

p (Polarity Inversion for Homing Switch and Limit Switches)

Command	Function		Type	Size
	Polarity Inversion of Home and Limit Switches		Immediate, Program	2
p (small "p")	**Example**	**Data 1**	**Data 2**	**Result**
	(Name) p ↲	none	none	s values
	(Name) p s ↲	s switch/type (0-3)	none	none

The "p" command is used to set the polarity of either the optional Home switch (used in the F (Find Home)), or the optional Limit switches (both set by the U (Set Port)).

Polarity is adjusted so that the ACTIVE level of the HOME and LIMIT+/LIMIT– can be matched to the type of switch used for these mechanical functions. Most Home and Limit switches are the simple normally-open variety, but some sensors or logic functions may close the circuit to ground (active) when the mechanical actuation is not present, and 'open' the connection when it is present. A benefit of normally-closed switch use is increased protection – if a normally-open limit switch fails by electrical means (wiring, connections, etc.), the failure may be masked because the circuit is *normally open*, but a normally-closed switch circuit would fail early if a wiring or connector failure occurs; thus it would presumably be ready when the Limit switch must actuate under some future 'runaway' condition.

Home Switch	Limit Switches	Setting 's'
Normally open	Normally open	0
Normally open	Normally closed	1
Normally closed	Normally open	2
Normally closed	Normally closed	3

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.

Q (List Program)

Command	Function		Type	Size
	List Program		Immediate	N/A
Q	**Example**	**Data 1**	**Data 2**	**Result**
	(Name) Q ↲	None implies address start is zero	none	Listing
	(Name) Q a (m) ↲	a address (0-2560)	m mode 0-1	Listing

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

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

The values will be displayed only if applicable to the particular instruction type. In Single-Axis mode twenty instructions are displayed at a time. Use the Enter↲ key to list up to 20 more commands without pause. 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 (m) is not provided or is"0", the command will display code segments until a non-command is encountered. If m is "1" the command will list starting with the address specified by Data 1, and continue to list all programs found between the starting address and the end of memory (1023).

R (Index Relative to Origin)

Command	Function		Type	Size
	Index Relative to Origin		Immediate, Program	5
R	**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" command is:

1. Wait until any previous motion is completed
2. Read *Position Counter* value, then subtract from the target position to yield offset steps required
3. Energize the motor winding at MOVE current value and start the settling timer
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 index
9. If another index is not commanded within the settling period, move to HOLD current

The R command may be implemented within a program. It is very useful when used in conjunction with the Origin command. Following, is an example that illustrates the benefit of R: When conditional steps can move the axis to an unpredictable location, the Origin provides the anchor mechanism that always allows the programmer to find a location (in reference to the current 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 this motion will allow the next command to start immediately! ------
L0 3           Loop Back to address location 3 (itself) until port 1 is ACTIVE (low)
@              Soft Stop will Decelerate and stop motion
R30000         Note that R 30000 moves the axis to that 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 counter value.

In Program Mode, all motion commands are automatically stalled (waiting) if a 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 usage for setting execution timing – use W0 prior to any command that should wait for completion of a previous motion

S (Save)

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

The "S" Command is used to save Parameters and/or Programs into NV (non-volatile) memory.

The controller's current operational parameters and programs are kept in main RAM memory, but this memory is lost at every power-down. To retain the parameters and programs permanently, they can 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, or can be recalled with the C Command at any time. Refer to the *C command*, and the *Memory Map* sections for more info.

Frequent or programmed use of S command should be avoided, as memory longevity may be affected.

This command should never be issued while a motion command is being executed, as it may interrupt motion for a short period of time during the save operation.

S0 - The parameter area (locations 200 to 255) contents are saved into the NV Memory, which are recalled as defaults during subsequent power-on resets. All parameters are saved from the working registers in the controllers Main Memory to the NV Memory.

S1 - The program area (locations 0 to 199, and 256 through 1023) contents are saved in the NV Memory, to be recalled as Main RAM Program Memory during subsequent power-on resets.

T (Echo Mode)

Command	Function		Type	Size
	Echo Mode		Immediate, Program	2
T	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 “`S0`” 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, I2, I3, I4). Data 1 defines the port (1=I1, 2=I2, 3=I3, 4=I4) and Data 2 defines the function, according to the table below:

Data 2	**Function & Command**	**Description of Function when ACTIVE****
1	**User Input Port**	Port value read using A command 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. **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, the controller 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
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 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 I1, I2, I3, I4 as user inputs (value 1).

The X command will show the current U setting for the input ports (such as U=1189).

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

If the U command is issued without data in Immediate mode, it will return the current setting of U in the order I1, I2, I3, I4. 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 28,000 SPS)	none	none

The "V" Command replaces the Slew Velocity V, with a positive non-zero value n, in steps-per-second (SPS). This is the maximum speed reached after acceleration from the initial velocity and carried on throughout the slewing motion. The maximum speed on your hardware will be limited by the motor characteristics and your 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 modified to negative or positive speeds to match whichever direction the motion command (+, –, R+, and R–) requires. Example: V is programmed to 3000 SPS, but becomes -3000SPS when a -10000 negative step command is issued.

Attempts to enter values above the maximum speed are limited to the maximum velocity (28000 SPS).

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

The slew velocity:

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

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.

This command is generally implemented during the initial parameter assignment; however, it may be used and changed within a program as needed.

In-motion adjustments of Slew Velocity

The "V" command (when used in **Operational Modes**) also allows changes to the speed of a motion *in-progress,* after it has already begun – in some limited cases a commands +, –, 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 operational register in real-time and modify the speed at which the motion continues, at that moment. Changes to Slew Velocity values do not change the target location of the motion – only the step rate, and thus the time it takes to reach that target.

Because non-motion commands can run at processor speeds *between motion commands,* the command interpreter will continue to accept and process commands that do not use the same resource (motion commands stall other motion commands until the 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 a motion, and some can continue during the motion. One of those actions 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)

Command	***Function***		***Type***	***Size***
w small w	Write Outputs/Read Back Outputs		Immediate, Program	2
	Example	***Data 1***	***Data 2***	***Result***
	(Name) w ↲ (Name) w (d) ↲	None d data 0-255 (only bits 0 & 1 valid)	none none	Output State none

The "w" command modifies output port levels in **Immediate** and **Program Mode**, and reads back those values in **Immediate Mode** only. When issued with a value d, the outputs are modified based on that value – when issued without a value, outputs are unchanged and the controller returns the existing output state as a result.

The default value of the output ports at power-on/reset is 0 (INACTIVE), pulled-up to VIO voltage. 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 *Input Port Electrical Configuration* section for more details.

General-Purpose Output State Values (binary encoding)	
Output	**Output**
O2	O1
2	1

Example: all outputs start inactive, then activate O2, disable O2, activate O1 and O2, then deactivate all:

command		**O2**	**O1**
w2 ↲	← O2 is worth 2 – send 2	ACTIVE	INACTIVE
w0 ↲	← O2 is worth 2 – send 0	INACTIVE	INACTIVE
w3↲	← O2 = 2 and O1 = 1 –> send 3	ACTIVE	ACTIVE
w0 ↲	← all off – send 0	INACTIVE	INACTIVE

The 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.

W (Wait)

<table>
<tr><td rowspan="2">Command
W</td><td colspan="2"><b>Function</b>
Wait / Wait for motion end</td><td colspan="2"><b>Type</b>
Program</td><td><b>Size</b>
3</td></tr>
<tr><td><b>Example</b>
(Name) W 0 ↲
(Name) W m ↲</td><td colspan="2"><b>Data 1</b>
none, implies 0 (wait for motion end)
m wait value x 10 mSec. (0-65535)</td><td><b>Data 2</b>
none
none</td><td><b>Result</b>
none
none</td></tr>
</table>

The “W” Wait command is used in **Program Mode**, and is used to create delay. “W0” is a reserved command that causes any motion command to complete before allowing following commands to execute – “Wm” command, where m is any non-zero value between 1 and 65535, will cause a wait of that number of milliseconds x 10 (minimum delay is 1mS and maximum delay is 655.3 seconds).

When used *after* fixed-step-count motion commands, such as a +, –, R+, and R–, “W0” command will cause the next command to wait for the previous motion to complete before execution. If W0 is used subsequent to 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.

W command with a wait value will cause execution to wait from that moment until m x 10msec have passed, **with no consideration for previous motion commands progress** (this should be fully understood before utilizing programming waits of any kind). Ex: “W250” yields a wait of 2.5 seconds.

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
P0	*Exit program mode*

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

P 0	*Enter program mode*
F2000	First resets the axis to the HOME location
+100000	Move 100000 steps plus (all moves wait for HOME to complete before going)
W200	Waits 2.5 seconds **from the start of the previous move**
w1	Turn output port 0 ACTIVE
W50	Wait 0.5 seconds
w0	Turn output port 0 INACTIVE
P0	*Exit program mode*

“W” command issued without the m parameter (m is assumed zero) creates a W0 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

This command returns the current value of all operational parameters. It also includes product name and code revision.

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

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
T=1
l=0
N=A
```

Where:

K	=	Ramp up (accel) / ramp down (decel)
I	=	Initial velocity / Divider
B	=	Jog Speed / Divider
V	=	Slew Velocity/ Divider
Y	=	Hold Current / Run Current
E	=	Settle time (delay between)
D	=	Divider
H	=	Resolution
U	=	Defines function assigned to input Pins (I4/I3/I2/I1). See U command
T	=	Echo Mode. See T command
l	=	Homing switch polarity. See l command
N	=	Controller name

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, **<u>but without data</u>**; 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**
	Program Hold and Run Current		Immediate, Program	3
Y	**Example**	**Data 1**	**Data 2**	**Result**
	(Name) Y ↲	none	none	Y values
	(Name) Y h r ↲	h Hold (0-100) in %	r Run (0-100) in % (133 opt*.)	none

The “Y” command specifies the Hold and Run values of motor current (per phase) in 1% increments. The value 100% represents a maximum of 1.5 Amps per phase. **The use of the optional heatsink/case allows specification of the Run current to 2 Amps, represented by a value of 133 (133%).*

The switch between using Hold and Run values for the output driver is automatic and immediate whenever a motion function is begun. Current reduction back to the Hold value is also automatic and occurs a fixed period of time after the axis is stationary; the programmable "settling time” (see "E" command) must be completed without another motion command before the reduction to Hold occurs.

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

The following procedure is used to set the programmable "Hold" and "Run" current feature:

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

   Entering “Y10 80” yields a 10% Hold current (150mA) and an 80% Run current (1.44A)

2. Optionally issue an "S0" (Save) command to store parameters into non-volatile memory

Once Y values are programmed, an index or other 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.

To implement **Current Disable** feature, set the Hold current to zero – this will maximize efficiency and minimum motor and driver heating (where applicable)

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)

<table>
<tr><td rowspan="2">Command
<b>Z</b></td><td colspan="2"><b>Function</b>
Read/Display Current Position Counter</td><td><b>Type</b>
Immediate, Program</td><td><b>Size</b>
2</td></tr>
<tr><td><b>Example</b>
(Name) Z ↲</td><td><b>Data 1</b>
none</td><td><b>Data 2</b>
none</td><td><b>Result</b>
Position</td></tr>
</table>

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 direction of travel. The initial value of the counter is set or modified by the O (Set Origin) 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, that when in **Multi-Axis Mode** the Z command response will be suppressed if issued from within a program. The reason for this is the potential for bus conflicts if the controller sends data that is not synchronized with the host.

The Position Counter may be modified by the O command without affecting the distance of a previous in-progress motion command, but the counter is updated immediately, perhaps leading to incorrect result, so we recommend updating the counter only after a W0 (wait for motion end).

Specifications

Absolute Maximum and Minimum Ratings

Characteristic	Symbol	Min	Max	Unit
Supply Voltage	V_{MM}	**10**	**35**	V (DC)
Positive Voltage Reference for Input/Output ~4.8V (unloaded) generated internally, if not externally provided	V_{IO}	**5**	**35**	V (DC)
Input and output signal voltage range (J3 pins 1-6)	V_{input} V_{output}	**-0.5V**	***VIO***	V (DC)
RS485 input voltage	V_{data}	**-12**	**12**	V (DC)
Continuous Operating Temperature Measured with System Status Read **}** Command	T_{maxpcb}	**0**	**85**	°C

Electrical Operating Characteristics

Characteristic	Symbol	Min	Max	Unit
Supply Voltage, recommended maximum value +/- 5%	V_{MM}	**9**	**33**	V (DC)
Input/Output positive voltage reference allowed input ~4.8V (unloaded) generated internally, if not externally provided	V_{IO}	**5**	**33**	V (DC)
Output Current/Phase (PEAK), without case/heatsink (set by Y command - maximum value **100**)	i_{phase}	**0**	**1.5**	A
Output Current/Phase (PEAK), with case/heatsink (set by Y command - maximum value **133**)	i_{phase}	**0**	**2**	A

Thermal and Mechanical Specification

Peak Operating Temperature........0 to +100°C abs max (driver stuttering if temp exceeds 85°C)
Storage Temperature .... -40 to +125°C
Size .... 1.55” x 1.55” x 1.1”, including connectors
Weight .... 20g, including connectors

Programming Specifications

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

Design Tips

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)

Thermal Considerations

The DC2M17 has been optimized to minimize power usage and heat generation, but it is the duty of the user/integrator to assure that motor currents and duty cycle used will not overheat the unit at working ambient temperatures.

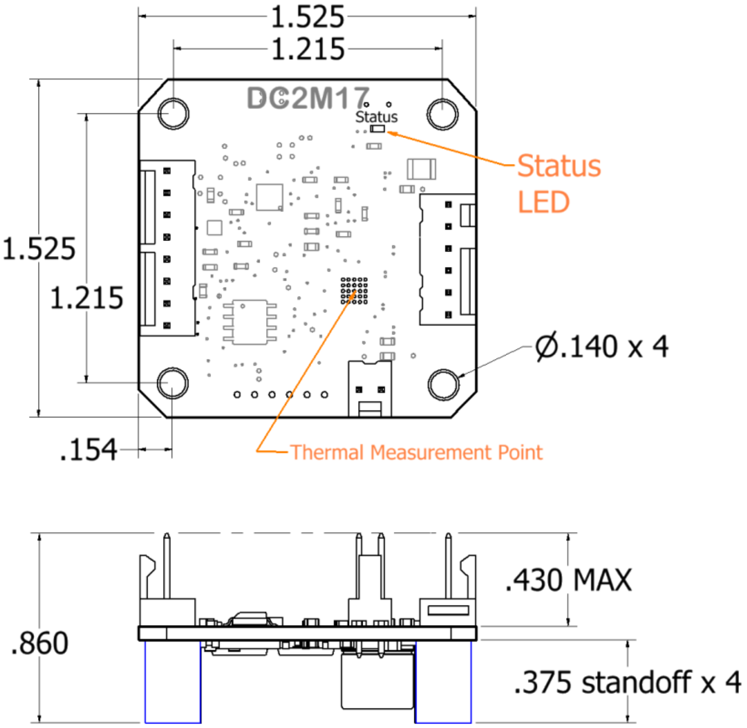
Adjust airflow so that the temperature inside the drive stage does not exceed 85°C -- utilize the internal thermal sensor for this data (Read Machine Status command ‘**{** ‘).

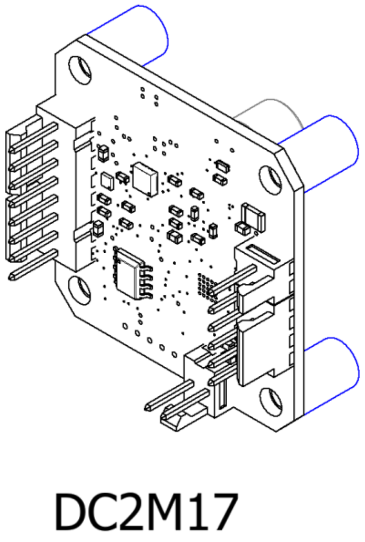
The driver will thermally-throttle at over 85°C, and may cause intermittent stopped motion (stuttering).

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.

Dimensions & Mounting





Drawing shows 3/8” #4 spacers (nylon), for mounting. Mounting holes match NEMA17 dimensions only

Motor Mounting

When mounting the DC2M17 onto a motor replace the four machine screws that hold the motor together with longer screws which both attach the DC2M17 and hold the motor together. See the table below for suggested screws and lengths with AccelMotion or AMS motors; for different brand motor refer to that manufacturers information to determine the replacement screw type.

AMS/AccelMotion Motor	Machine Screw Required	
	Screw type*	Length (mm)
AME17-32-S	M3-0.5	40
AME17-60-S		45
AME17-75-S		55

* When selecting the type of machine screw, it is critical that the bottom side of the head of the screw is flat and not tapered
When removing screws from any stepper motor do not disassemble the motor; it can damage the motor and void the warranty

Set-up for Phase Current Measurement

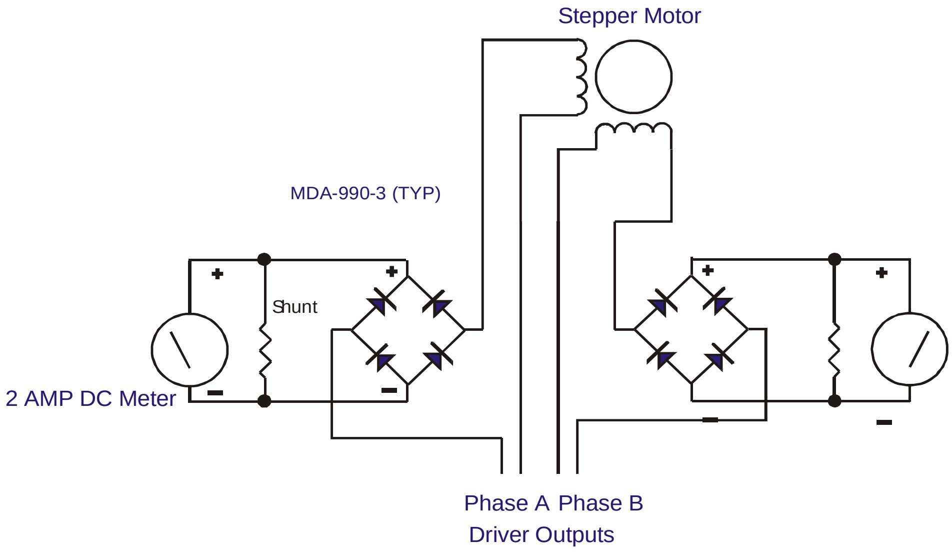
The following is the basic setup diagram for 2-phase average current measurement:

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 DC2M17
3. Set the DC2M17 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 read about 750mA – 50% of the 1.5A maximum current.

WARNING:

CONNECTING or DISCONNECTING MOTORS WHILE POWER IS APPLIED WILL CAUSE DAMAGE THAT IS NOT COVERED BY AccelMotion 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

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