C4

Controller for MD2 Motor Drivers

PC-Based Automation Made Simple

User's Guide

Revision A

ARRICK ROBOTICS

ArrickRobotics.com

The C4 Motor Controller

Table of Contents

Introduction

C4 is an intelligent controller that can control two MD2 Stepper Motor Drivers - 4 motors total. C4 takes the processing burden off of the PC resulting in smoother motion and new motion features. This also allows the PC to perform other tasks such as networking, sensor data acquisition, and data processing while motors are moving.

C4 receives commands from a serial (RS-232) port and can be controlled by any external processor that can send commands including standard PCs, PLCs, and microcontrollers. A Serial-to-USB converter cable allows C4 to communicate over the USB port as if it was a serial port simplifying programming.

Features and Specifications

MD2 Ports: 40 pin headers. Controls two MD2 drivers (4 motors), cables provided.
Communication Parameters: Serial, 8 data bits, 1 stop bit. 300-115200 baud (Default 9600).
Communication Port: 9-pin female D-Sub connector for straight-through cabling to a PC.
Command Protocol: Simple ASCII commands.
Size: 8.5" wide, 1.5" tall, 3.5" deep.
Weight: 18oz.
Power: 9-15VDC unregulated. 2.5mm DC plug, center positive. 50ma nominal current draw.
A 110V or 220V wall transformer is provided depending on the product ordered.
Driver Outputs: Phase outputs designed for direct connection to MD2 Stepper Motor Driver Systems.
Minimum step speed: 100 steps per second. Unlimited slow speed achieved by host software.
Maximum step speed: 10,000 steps per second. Maximum motor speed depends on motor/load.
Speed accuracy: typically within 3% of specified speed.
Ramping: Linear
Speed Profile: Motor speed can be changed during motion or decelerated to a stop.
Interpolation: Up to 4 axis coordinated linear interpolation. Pseudo-circular via host software.
Position Reporting: Motor position and ramping status can be monitored during motion.
Homing: Built-in home sequence using switch or sensor.
Motion Limiting: Forward and reverse using paralleled home switches, or via host software.
Post-Move Holding: Adjustable time in 1-second increments, full or half current.
General Purpose I/O: The I/O port of MD2 #1 is fully accessible—2 digital outputs, 3 digital inputs.
Internal Firmware: Externally updateable through serial port.

Support and Contact Information

For complete information about Arrick Robotics, contact information, and technical support, see:

Installation/Operation

Power Switch
This switch controls 12VDC power to the C4. Power may be turned ON/OFF at any time without harming motors, but positions, communications and other data may be lost.

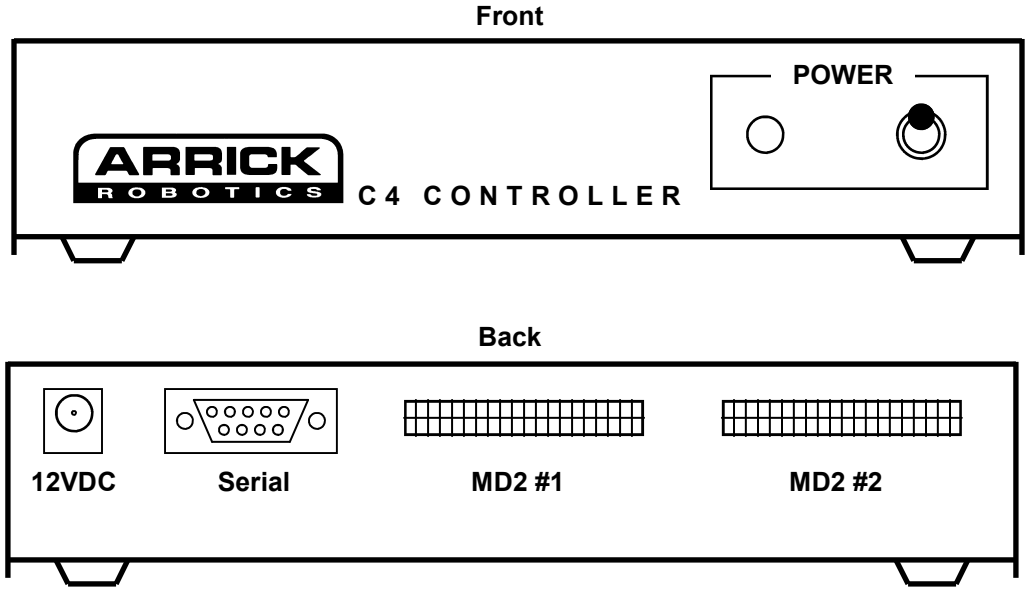
Power Light
This light normally indicates that power is applied to the C4, but it can also be controlled by software and is capable of indicating other conditions by flashing.

12VDC Power Input
2.5mm DC power connector (center positive) can accept any unregulated voltage between 9V and 15V. Current draw is approximately 50ma. Power can be supplied by a wall transformer or batteries. A 110V or 220V wall transformer is provided depending on the product ordered.

Serial Port
Receives commands from the host to control motors. A straight-through cable is needed to connect to a standard personal computer with a 9-pin D-Sub serial port. Do not connect this port to a monitor or any other port as damage may occur. A USB converter cable can be used to connect the serial port to a PC's USB port.

MD2 Ports
These connectors are 40-pin male headers with latching ears. The C4 can control two MD2 Stepper Motor Driver Systems at the same time. A cable with a 40-pin female header (C4 end) and a 36-pin male Centronics-style connector (MD2 end) is provided.

Mounting
C4 is designed to stack on top of one or more MD2 systems. Airflow and heating is not a concern.

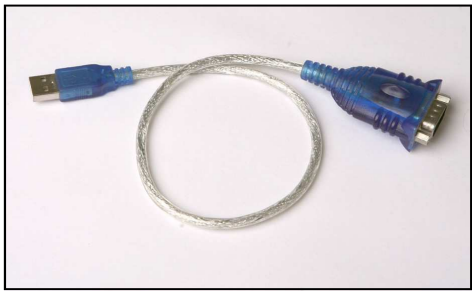


Installation/Operation —continued

Serial Port Connection

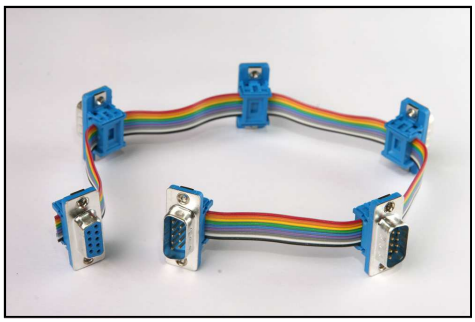
The serial port is a standard RS-232 port and can be connected directly to a Personal Computer with a 9-pin D-Sub connector. The C4 has a female connector and the PC will have a male connector. A male-to-female cable is provided for this connection.

USB Port Connection



A simple USB-to-Serial converter is used to allow the PC's USB port to control C4. The converter comes with software drivers that must be installed before use. The drivers allow the USB port to look like a standard serial port to software running on the PC. This makes programming very simple within any programming language—Visual Basic, C, etc. See the converter package for details about installing the drivers.

Daisy Chaining



The serial port on the C4 system can be daisy-chained to allow up to three C4's to run from the same port, thereby controlling up to 12 motors (6 MD2 systems) via a single port.

Daisy chaining is accomplished by simply connecting all of the C4 serial ports in parallel using a custom cable. Each connector is a 9-pin D-Sub style with males going to the C4's and a female going to the PC end.

Commands from the PC are sent to every C4 in the chain but only the ones matching the ID character will respond. Each command begins with a '!' character followed by an ID character which is usually '1', '2', '3', etc. The factory default is '1'. Each C4 must be pre-programmed with its own unique ID prior to connecting it to the chain. The ID is saved inside the C4 during power-off. See the 'wi' command for more details.

Programming

Overview
C4 can be commanded by any device that has a standard Serial (RS-232) port, or USB using a USB-to-Serial converter. This could be a desktop or laptop personal computer, an industrial PLC, or even a small microcontroller such as a Basic Stamp or PIC processor.

Programming
Commands are sent as simple text to make programming easy. These commands can be sent manually using a dumb terminal program such as Hyperterm in Windows, or from a programming language such as Visual Basic or C.

Commands
Each command line begins with a '!' character followed by an ID character to identify the C4 controller (useful when daisy-chaining). The ID character is set to '1' as a factory default and can be changed. The ID character is saved in non-volatile memory inside of the C4 controller and not destroyed when power is removed. It is possible to configure C4 so that the '!' and ID prefix are not required by setting the ID to '0'. It is also possible to force all daisy-chained C4 controllers to respond to a command using the '!!' prefix. See the 'wi' command for details.

Commands consist of a single character such as 'm' for move or 'h' for home, and many are then followed by parameters needed by the command such as distance, speed, etc. Commands are NOT case sensitive.

Commands are terminated with a Carriage Return (CR) (0x0D) -OR- a Line Feed (LF) (0x0A). When the termination is received, C4 will execute the command. The command buffer will respond properly to a backspace. Escape (ESC 0x1B) clears the command buffer.

Command Examples

`!1h1`	*Home motor #1*
`!1m4f2040`	*Move motor 4 forward 2040 steps*

Command Results, Acknowledgment and Error Responses
Most commands will respond with a result, a progress code, 'a' indicating acknowledgement, or 'b' indicating a bad parameter.

Power-Up Conditions
At power-up, both MD2s are enabled, motors are de-energized and output pins are turned off. The ID character is retained from the previous session - factory default is '1'. The communication baud rate is also retained from the previous session - factory default is 9600.

Command Summary and Quick Reference

The most common commands are listed below. Examples shown use an ID of 1 and must be terminated with a CR. Commands and values are not case sensitive.

A Abort motion. !1a

B Command buffer control. Buffered commands can be used for future execution.
!1bf CR then XXXX = fill XXXX from beginning, terminate with CR.
!1baXXXX = Add XXXX to end of buffer.
!1be = execute buffer. !!be = Cause all daisy-chained C4s to execute.
!1bd = display buffer. !1bs = Display buffer size. !1bu = buffer usage.

H Home motors. !1h1 = Home motor #1, !1h2 = Home motor #2, etc.
Returns progress codes 1,2,3. 1 = seeking home switch, 2=moving off switch, 3=moving offset.

I Read inputs. !1i = Returns 7 digits of 0 or 1 referring to home switch 1, 2, 3, 4, and inputs #1, #2, #3.

M Move motor(s).
2nd character specifies motor #, 3rd specifies direction (f=forward, r=reverse), following digits specify # of steps.
Multiple moves can be put on the same line to create linear interpolation of multiple motors. With multiple motor moves, all motors will start and stop and the same time. Speed parameters of the longest traveling motor are used.
Single-motor move example: !1m1f200 = Move motor #1, forward, 200 steps.
Dual-motor example: !1m1f200m2r5500 = Move #1 forward 200 steps, #2 reverse 5500 steps.
Append 'n' to move command to return 'o' indicating the move is complete.

N Notify when input, motor state, motor position, or time delay is complete.
!1ns0 = Notify when move state is 0. Returns 'o'. Many other options.

O Output control. 2nd character specifies the output #, 3rd character specifies On/Off (1 or 0).
!1o20 = Output #2, off. !1011 = Output #1, on.

R Read parameters. 2nd character indicates what to read -same characters as **W** (write) command.
!1rp1 = read position of motor 1 (1-4). Position is steps moved on last move.
!1rs = read move state. Returns 0=off, 1=accelerating, 2=constant speed, 3=decelerating. No CR.

W Write parameters. 2nd character indicates what to write - same characters as **R** (read) command.
!1wv1,1000,2500,2000 = write velocity parameters for motor 1 (1-4).
2nd value is minimum speed in steps/sec, 3rd value is maximum speed, 4th is ramp slope.
!1we11 = Energize motor 1. 1st value is motor (1-4), 2nd is state (0=de-energize, 1=energize).
!1wh1,r,100 = Write home parameters. First value is motor (1-4),
2nd value is direction (r=reverse, f=forward), 3rd value is home offset in steps.
!1wk2,10 = Write hold parameters. First value is mode: 0=off, 1=full current, 2=half current,
2nd value is time in seconds after motion is stopped.

C4 Command Reference

(c) - Copyright Notice

Returns copyright notice string terminated by a CR.

To retrieve copyright notice: !1(c)
Returns: Copyright string terminated by a CR.

A - Abort Motion

Stops any motion in progress.

Motor positions (steps taken) can be read after aborting using the 'rp' command.

If motors were running fast, they may not be able to stop instantly and their reported positions may not be accurate. In this case, home the motor(s) with the 'h' command to insure accurate positioning.

To abort current motion: !1a
Returns: 'a' as acknowledgement

B - Command Buffer Control

The 'B' command is used to store and execute commands. This allows sequences of motion to perform elaborate profiles such as circles via short line segments, and allows buffering of commands in multiple C4 controllers for simultaneous execution.

A - Add a single command to the end of the buffer: !1baxxxx where xxxx is a standard command
Returns: 'a' as acknowledgement

C - Clear command buffer: !1bc
Returns: 'a' as acknowledgement

D - Display the command buffer: !1bd
Returns: The buffer contents.

E - Execute the command buffer: !1be
Returns: Commands as they are executed and results from each.
To execute the command buffer on all daisy-chained C4 controllers simultaneously: !!be
Returns: Ignore because all C4s will be speaking at once.

F - Fill the buffer with multiple commands from the beginning: !1bf CR, then commands.
Terminate the commands with 2 CRs to complete filling.
Returns: 'a' as acknowledgement

L - Load buffer from EEPROM: !1bl
Loads the command buffer with a sequence previously saved in EEPROM memory.
Returns: 'a' as acknowledgement

P - Power up autoexecute: !1bp
Causes the sequence stored in EEPROM memory to load and run upon power-up.
Disable autoexecute with a 'wa' command.
Returns: 'a' as acknowledgement

S - Save buffer into EEPROM: !1bs
Saves the command buffer into non-volatile EEPROM memory.
Returns: 'a' as acknowledgement

U - Usage of command buffer: !1bu
Returns: The buffer usage in bytes (characters) including the end terminator.

Z - Size of command buffer: !1bz
Returns: The buffer size in bytes (characters). 1 byte is reserved for the end terminator.

Daisy-chained controller execution example:

`!1bf` **`CR`** `m1f200` **`CR CR`**	Stores a move motor #1 command in controller #1 (!1)
`!2bf` **`CR`** `m1f400` **`CR CR`**	Stores a move motor #1 command in controller #2 (!2)
`!!be` **`CR`**	Execute both commands together.

F - Firmware Information and Update

The 'F' command reports information about C4's internal firmware and allows for updates to add features, improve performance or to fix bugs.

To retrieve the product name: !1fp
Returns: 'C4' followed by a CR.

To retrieve the product description: !1fd
Returns: Full product description string followed by a CR.

To retrieve the current firmware version: !1fv
Returns: '1.2.3' followed by a CR. Version shown is an example.

To begin upload new firmware: !1fu
Begins firmware update. The host must then read a hex file and upload it to C4. The MD2xp program can perform this function. See ArrickRobotics.com/c4 for the latest firmware files.
Returns: nothing.

H - Home Motor(s)

The 'h' command moves motor(s) to their home position using the home switch. This sets the reference (zero) location for a motor and is normally used at startup. If multiple motors are specified, each will be moved home separately in the sequence specified in the command.

The following homing sequence is used. This pre-loads the mechanics in the forward direction (away from home) and improves repeatability for subsequent moves also made in the forward direction.

#1 - Move the motor in the direction set by the Home Direction parameter, without ramping, until the home switch is activated.
#2 - Move in the opposite direction without ramping until the home switch is de-activated.
#3 - Move a distance set by the Home Offset parameter, with ramping.

The Home Offset parameter can allow a program to run at various places in the work envelope.

The Home command responds with characters indicating progress. '1' = moving home, '2' = moving away from home, '3' = moving to offset, 'a' = completed. Sending an 'a' (without a CR or prefix) during homing will abort the command and return an 'a'.

Home motor 1: !1h1
Returns: '123o' as the homing sequence progresses.

Home motor 4 then motor 1: !1h41
Returns: '123o' as the homing sequence progresses for each motor.

I - Read Inputs

The 'I' command returns a string of digits indicating the status of home switches and input bits. The order of the returned input signals is: home switch #1, home switch #2, home switch #3, home switch #4, input #1, input #2, input #3 - all from MD2 #1 I/O port

To read inputs: !1i
Return example: 0000101 **CR** In this example, inputs #1 and #3 are ON (1), all others are OFF (0)

M - Move Motor(s)

The 'M' command starts one or more motors moving. All 4 motors (2 MD2 systems) can move together in a coordinated fashion - all motors will start and stop at the same time regardless of their travel distances. This results in linear interpolation on a Cartesian (XY, XYZ) positioning system.

All moves are relative to their current position. Each command must specify the motor #, direction (f = forward, r = reverse) and the distance in steps. Valid step count values are 1 - 2,000,000,000 (2 billion). Absolute moves (to a specific absolute position regardless of the current position) must be handled by the host by maintaining a current position value and calculating the relative move needed.

Speed parameters (minimum speed, maximum speed, and acceleration/deceleration) that were previously set using the 'wv' command are used.

The definition of forward and reverse is determined by the Home Direction parameter. See the 'rh' and 'wh' command to read and write home parameters.

For multi-axis moves, simply concatenate the motion commands, then terminate with a CR.

The 'M' command starts motion and immediately returns an 'a' to acknowledge receipt of the command. While motor(s) are moving, the programmer may request their position to monitor progress using the 'rp' command, and monitor the current Move State using the 'rs' command.

Appending an 'n' to the end of a move command causes it to notify the host with an 'o' when the motion is complete ok, or 'l' indicating a limit switch abort.

Move motor #3 forward 2200 steps: !1m3f2200
Move motor #1 forward 1400 steps, and #2 reverse 24 steps: !1m1f1400m2r24
Returns: 'a' to acknowledge receipt of the command.

Move motor #2 forward 100 steps and notify when done.: !1m2f100n
Returns: 'a' to acknowledge receipt of the command, and an 'o' when complete.

N - Notify of Event

The 'N' command notifies the host when an event occurs without constant polling by the host using 'r' commands. Event options include condition of a home switch, input signal, motor position, move state, or time delay. When the event occurs, an 'o' is returned. Notify commands are aborted (and an 'o' returned) if any character is received on the serial port. Other commands can not be given while awaiting an event.

The second character of the command determines which type of event is monitored:

i = input signal or home switch
s = move state
p = motor position
d = delay

The final characters determine which event and/or values depending on the command.

Notify when home switch #1 goes low(0): !1ni10 Home switches 1-4 are #1-4. Last # is value (1/0).
Notify when input #1 goes high (1): !1ni51 5 = input #1, 6 = #2, 7 = #3.
Notify when input #3 goes low (0): !1ni70
Notify when motor #1 gets to position 300 or beyond: !1np1,300
Notify when move state goes to deceleration: !1ns3 0=off, 1=accel, 2=constant, 3=decel.
Notify when motors stop (move state = 0): !1ns0
Notify when 10 seconds has passed: !1nd10
Returns: 'o' when the condition is meet or any character is received which aborts the command.

O - Outputs

C4 gives the programmer access to the general purpose I/O port on MD2 #1 which provides 2 digital output pins. Output pins are active Low meaning that ON = 0 volts, OFF = +5 volts. See the MD2 user guide for circuit ideas.

The 'O' command is followed by a number specifying the output pin, a comma, then a 1 or a 0 specifying the state where 1=ON=0 volts, 0=OFF=+5 volts.

Turn ON output #1: !1o1,1 ON = 0 volts at output pin.
Turn OFF output #1: !1o1,0 OFF = +5 volts at output pin.
Turn ON output #2: !1o2,1
Turn OFF output #2: !1o2,0
Returns: 'a' as acknowledgement

R - Read Parameters / W - Write Parameters

The 'R' command reads parameters. The second letter of the command indicates what parameter. For most 'R' commands, there is a corresponding 'W' command to write the parameter's value.

Most 'R' (read) commands will return value(s) followed by a **CR**, and most 'W' (write) commands will return an 'a' as acknowledgement.

RA, WA - Automatically Execute Buffer at Power-on Flag

The auto-execute flag causes C4 to execute the commands previously saved in the EEPROM upon power up. See also the 'BP' command and other buffer-related commands.

Read auto-execute flag: !1ra
Returns: '1' if ON, '0' if OFF, followed by a CR
Write auto-execute flag to ON: !1wa1
Write auto-execute flag to OFF: !1wa0
Writes return: 'a' as acknowledgement

RB, WB - Baud Rate (speed) for Host Serial Port

This value sets the baud rate for the host communications. The default is 9600 which will work for most applications. The selected baud rate is stored in non-volatile memory and is retained after loss of power. The bootloader (See 'F' command) uses only 9600 baud for updating the firmware.

Baud rate codes are: 1=300, 2=600, 3=1200, 4=2400, 5=4800, 6=9600, 7=14400, 8=19200, 9=28800, 10=38400, 11=57600, 12=76800, 13=115200.

Read baud rate: !1rb
Returns: code followed by a **CR**
Write baud rate to 1200 baud: !1wb3
Writes return: 'a' as acknowledgement

RD, WD - Deceleration Enable Flag

This value determines if deceleration is used or not when a motor stops. The default is ON and it normally doesn't require changing.

Read deceleration enable flag: !1rd
Returns: '1' if ON, '0' if OFF, followed by a CR
Write deceleration enable flag to ON: !1wd1
Write deceleration enable flag to OFF: !1wd0
Writes return: 'a' as acknowledgement

Read / Write parameter commands continued...

RE, WE - Energize Motor(s)

The energize motor flag determines if a motor has power applied. At standstill, an energized motor will retain it's position with holding torque. This generates heat and consumes power. Under most circumstances, it is not necessary to read or write this flag since energizing occurs automatically with move commands and holding parameters determine the condition after moves are completed. See the 'rk' command for holding parameters.

Read energized flag for motor 1: !1re1 Specify motor 1-4.
Returns: '1' if ON, '0' if OFF, followed by a CR
Write energize motor #1: !1we11
Write de-energize motor #2: !1we20
Writes return: 'a' as acknowledgement

RH, WH - Homing Parameters

Homing parameters control the direction and the home offset used during motor homing. Direction is r=reverse, f=forward. Offset is in steps. Normally, offset is a small number such as 25 steps so the motor does not rest too close to the switch, but it can also be used to perform a sequence of moves at various locations within a work envelope. See the 'H' command for details about homing motors.

Read home parameters: !1rh1 Specify motor 1-4.
Returns: r,100 followed by a CR This example: r=reverse, 100 steps offset.
Write home parameters for motor #1: !1wh1,r,100 This example: r=reverse, 100 steps offset.
Writes return: 'a' as acknowledgement

RI, WI - C4 ID

This sets or reads the ID character used for C4 commands. The default is '1', but it can be changed to allow several C4's to share a single communication line. The ID is stored in non-volatile memory and is retained after loss of power. If you don't know the ID and C4 will not accept any commands, use an ID of '!' with a 'WI' command to set it again. The ID '!' forces C4 to accept the command. You can also issue a '!!ri' command and force a C4 to return its ID regardless of what it is. Its also possible to turn OFF the command prefix requirement by setting the ID to '0'.

Read ID: !1ri
Returns: ID character followed by a CR Use ID values from 1 to 9
Change the ID from 1 to 2: !1wi2 Subsequent commands must now use "!2"
Writes return: 'a' as acknowledgement

Read / Write parameter commands continued...

RK, WK - Hold Parameters

Hold parameters determine how a motor acts when a motion is complete. Holding torque can be retained for a certain period of time at full current or half current, or the motor can be de-energized immediately after the move. Hold parameters apply to all motors.

The first value is the hold mode. This determines the condition of the motor current after the time period is up. Codes are 0=off (de-energized), 1=full current, 2=half current (AKA standby mode). The second value is the time period in seconds (0-250). Mode 1 keeps the motors energized forever and the time period is meaningless.

Read hold parameters for motor 1: !1rk
Returns: 2,10 followed by a CR This example shows hold mode 2, 10 seconds.
Set motors to de-energize 15 seconds after moves: !1wk0,15
Set motors at half current 5 seconds after moves: !1wk2,5
Set motors at full current forever after moves: !1wk1,0 The time period is meaningless.
Writes return: 'a' as acknowledgement

RL, WL - Limit Switch Stop Flag

The limit switch stop flag determines if motors will stop when a home switch is activated during a move. A forward travel limit switch can be wired in parallel with the home switch, and both switches together will provide auto stop limits during moves. When a forward limit switch stops a motion, the limit switch stop flag can be turned OFF in order to reverse the motor away from the limit, then turned back ON. The default for this flag is ON.

Read limit switch stop flag: !1rl
Returns: '1' if ON, '0' if OFF, followed by a CR
Set limit switch stop flag to ON: !1wl1
Set limit switch stop flag to OFF: !1wl0
Writes return: 'a' as acknowledgement

RN, WN - MD2 Enable Flag

The Enable flag gives manual control of MD2 driver enabling. It's not normally necessary to manually control this because MD2s are automatically enabled at power-up and disabling is of minimal use.

Read MD2 enable flag: !1rn
Returns: '1' if ON, '0' if OFF, followed by a CR
Set MD2 enable flag to ON: !1wn1
Set MD2 enable flag to OFF: !1wn0
Writes return: 'a' as acknowledgement

Read / Write parameter commands continued...

RO, WO - Echo Commands

Normally, command characters are not echoed back to the host as they are received, but this flag allows control of it. The flag is not retained after power-off.

Read Echo flag: !1ro
Returns: '1' if ON, '0' if OFF, followed by a CR
Set Echo flag to ON: !1wo1
Set Echo flag to OFF: !1wo0
Writes return: 'a' as acknowledgement

RP, WP - Motor Positions

Motor positions are the number of steps given during the current or last move. These positions can be read during moves to monitor progress and after moves to see how many steps were given.

The number of steps taken is the same as the number of steps given unless motion was stopped with an abort command ('a'), or a limit switch. Both of these conditions can result in uncertain motor positions and motors should be homed afterwards. Motor positions can also be incorrect if they are moved too fast or the load exceeds their torque capacity.

Motor positions can be written but is of minimal use.

Read Position of Motor 1: !1rp1
Returns: An integer position followed by a CR
Read Position of Motor all 4 motors: !1rp
Returns: An 4 integer positions separated by commas and followed by a CR
Writes return: 'a' as acknowledgement

RR, WR - Motor Ramping Enable Flag

Motors are normally ramped (accelerated and decelerated) during moves but it can be disabled with this command. The default is ON.

Read Ramping flag: !1rr
Returns: '1' if ON, '0' if OFF, followed by a CR
Set Ramping flag to ON: !1wr1
Set Ramping flag to OFF: !1wr0
Writes return: 'a' as acknowledgement

Read / Write parameter commands continued...

RS, WS - Move State

The Move State indicates the current state of a motion. Motion typically starts by accelerating, then runs at a constant speed, then decelerates to a stop. The current Move State can read to monitor the progress of a move, or written to change how a move behaves. Move state codes are:

0 = Not moving
1 = Accelerating
2 = Constant speed
3 = Decelerating

Read the move state: !1rs
Returns: Move state code: 0-3, NOT followed by a CR for speed.
Set move state to cancel motion: !1ws0
Set move state to begin deceleration: !1ws3
Writes return: 'a' as acknowledgement

RT, WT - Step Type

Step type determines the pattern used to energize coils as a motor is moved. Under most circumstances, it's best to stay with the default Half step mode which produces 400 steps per revolution and reduces vibration. You can also select Full single-coil or Full double-coil modes which produces 200 steps per revolution.

Read the step type: !1rt
Returns: Step type: h=half, s=single, d=double, followed by a CR
Set step type to half: !1wth
Writes return: 'a' as acknowledgement

RU, WU - Current Motor Speed

Motor speed can be read or changed while the motor is moving. The speed value is in steps per second. Speed changes are instant and do not ramp. Changes to the speed beyond the start-stop speed of the motor (about 1000 sps) may result in stalling and lost positions. If multiple motors are moving simultaneously, the speed of the fastest motor is changed and others will follow in the correct ratios. If no motor is moving when the speed is changed, nothing will happen. If no motor is moving when the speed is read, the return value will be 0.

Read the step type: !1ru
Returns: value of current speed in steps per second followed by a CR
Change the current speed to 800 steps per second: !1wu800
Writes return: 'a' as acknowledgement

Read / Write parameter commands continued...

RV, WV - Velocity Parameters

A typical motor move will begin at the minimum speed, accelerate to the maximum speed, run at a constant speed, then decelerate back down to the minimum speed before coming to a complete stop. C4 controls the speed while counting steps so when the final deceleration is done, the desired step count is also complete.

When multiple motors are moving together, velocity parameters of the motor moving the most steps is used. The other motors will be moving at the same speed or a slower than this speed depending on their step counts. All motors will start and stop at the same time even though their step counts are different. The result is linear interpolation on XY or XYZ positioning systems.

Each motor maintains 3 velocity parameters. In most applications, each motor will have the same values. The first value is the minimum speed (steps per second) that the motor starts at before accelerating, and ends at after deceleration. The second value is the maximum speed (steps per second) which is the top speed the motor accelerates to. The third value is the acceleration/deceleration slope given in steps per second squared. Think of the slope as the value of increase in velocity each second. A slope of 2000 means that the motor will increase speed by 2000 steps per second, every second.

It's important to select speed and slope values that are within the motor's operating range. A stepper motor's torque diminishes as its speed increases. Values which are too high will result in lost steps, but will not harm the motor. Find the motor's limits for your application payload then back off about 50% to allow a margin of error.

Read the velocity parameters of motor #1: !1rv1 Specify motor 1-4
Returns: 3 values separated by commas, followed by a CR Min speed, max speed, slope
Set velocity parameters of motor #3: !1wv3,1000,3300,2000 Min speed, max speed, slope
Writes return: 'a' as acknowledgement

RY, WY - Standby Mode (Half Current Mode)

Standby is a special mode that delivers only half of the normal current to each motor. This results in reduced heat dissipation but also lower torque. Standby mode is useful when motors must retain their position while stopped for long periods of time. Normally, it's not necessary to manually control standby mode because the Hold parameters control what happens after motion is complete - see the 'wk' command. Standby mode affects all motors together.

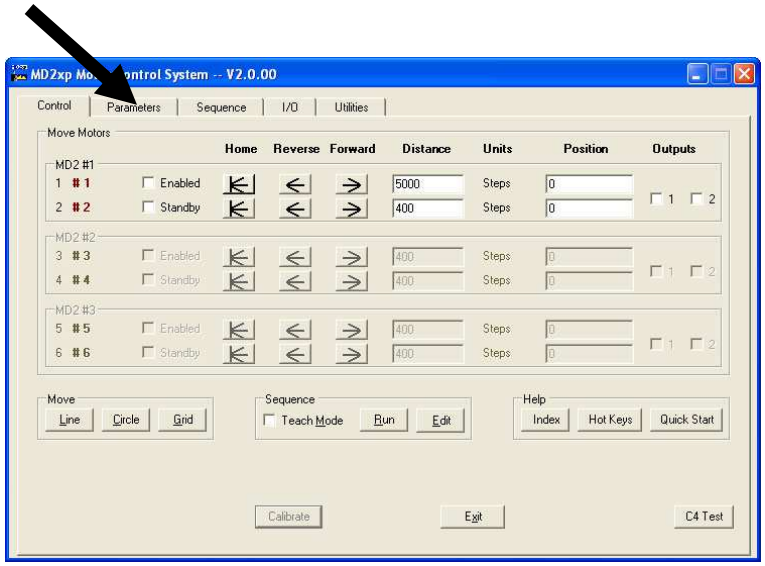
Read the step type: !1ry
Returns: '1' if ON, '0' if OFF, followed by a CR
Set standby mode OFF: !1wy0
Set standby mode ON: !1wy1
Writes return: 'a' as acknowledgement

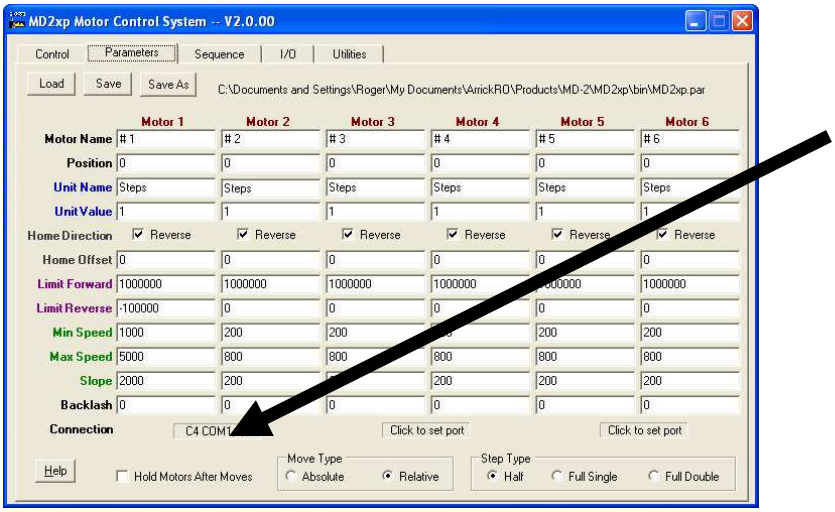
MD2xp Program

Version 2.0 and newer can operate MD2 motor drivers from either the parallel printer port, or via a C4 controller through a serial port. (or USB with a converter).

Before moving motors, you must first select the C4 port and ID. This is done on the Parameter tab, in the Connection field.

Parameters Tab



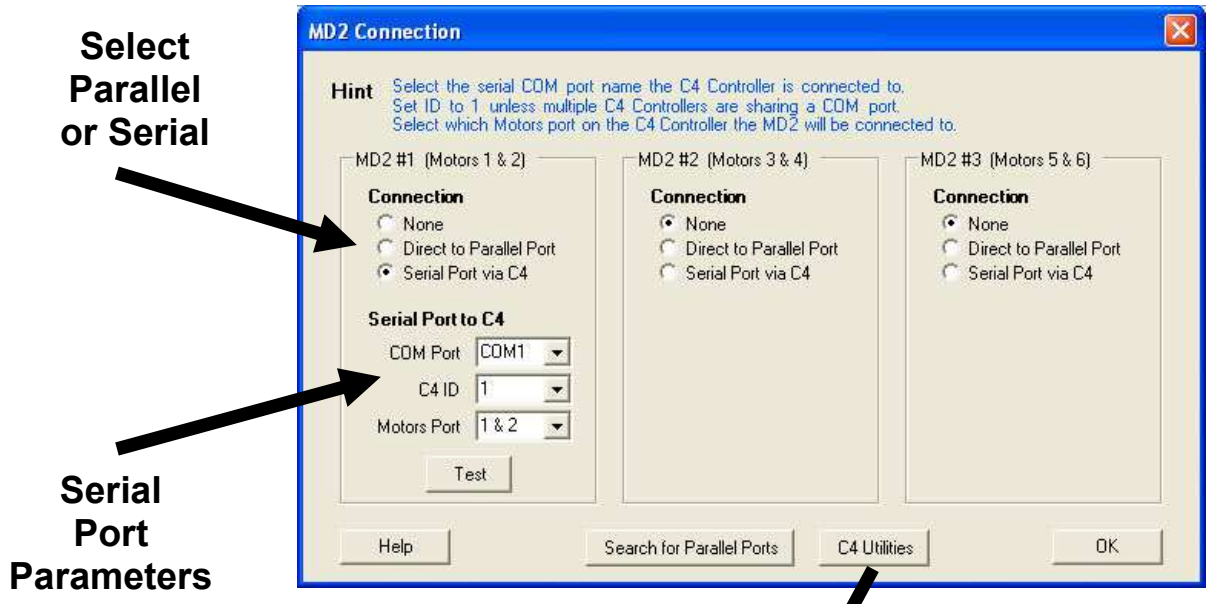


Connection field Click to show the Connection Form

MD2xp Program continued...

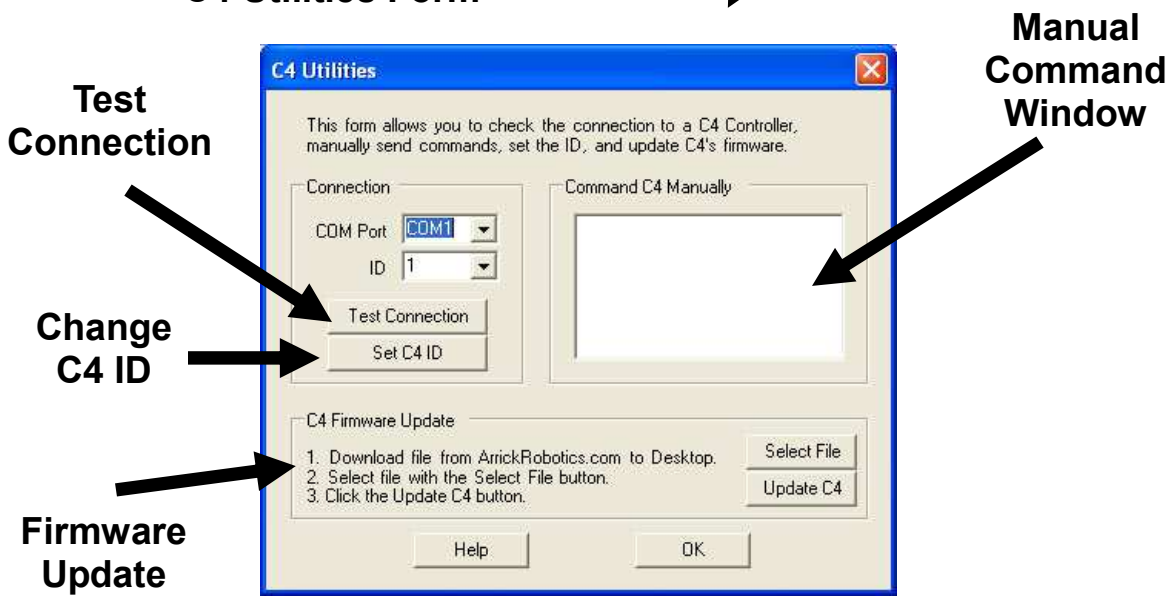
The MD2 connection form lets you select a direct parallel port connection to the MD2, or a connection via the C4 controller. For a C4 connection , select a serial port (varies depending on your computer), an ID (Usually 1), and the Motors Port (usually 1&2). A test button is provided so you can try various serial (COM) ports.

MD2 Connection Form



The Utilities form lets you manually control a C4, change C4's internal ID character, and upgrade the firmware.

C4 Utilities Form



Visual Basic Programming

Visual Basic Express is freely available from the Microsoft website. Controlling C4 is as simple as opening a serial port and sending and receiving short text commands. Below is an example. Create a form, add a button control, and copy this code into the button's click event.

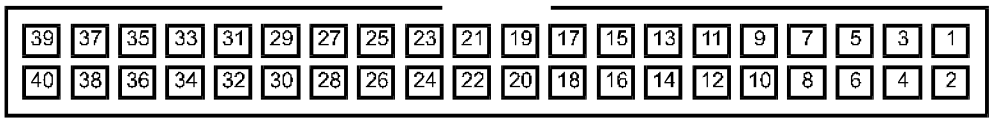
```
'C4 Controller Example Program for Visual Basic

Dim S As String                    'String to hold responses.
Dim C4Port As New IO.Ports.SerialPort 'Object to hold port.

C4Port.PortName = "COM1"           'Change this for your COM port.
C4Port.BaudRate = 9600             'Set port parameters.
C4Port.Parity = IO.Ports.Parity.None
C4Port.DataBits = 8
C4Port.StopBits = 1
C4Port.Open()                      'Open the port.

'Clear any bad data in port buffers.
C4Port.Write(Chr(&H1B))            'Send Escape.
Threading.Thread.Sleep(100)        'Short wait.
C4Port.DiscardInBuffer()           'Discard anything received

'Display firmware version.
C4Port.Write("!1fv" & vbCr)        'Firmware Version command.
S = "C4 Firmware: V" & C4Port.ReadTo(vbCr) & vbCrLf
MessageBox.Show(S, "C4 Version") 'Show message.

'Move motor 1 home.
C4Port.Write("!1h1" & vbCr)        'Home command.
Do                                 'Wait for o indicating finished.
   S = Chr(C4Port.ReadByte)
Loop Until S = "o"

'Move motor 1 forward 400 steps, n=notify when done.
C4Port.Write("!1m1f400n" & vbCr) 'Move command.
S = Chr(C4Port.ReadByte)           'Wait for a indicating started.
S = Chr(C4Port.ReadByte)           'Wait for done status character.

MessageBox.Show("Done")            'Done message.
```

The MD2xp program source code is also provided on the CD which offers a more extensive example using our subroutine library for Visual Basic .NET -- MD2VN2.vb.

Connector Pinouts

MD2 Ports

MD2 ports are designed to be wired to 36-pin male Centronics connectors with flat cable.
Connector: 40 pin .1 IDC male Header

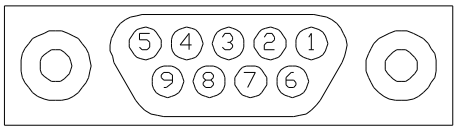


1 – MD2 Enable-
2 – Ground
3 – Motor 1, phase 1-
4 – Ground
5 – Motor 1, phase 2-
6 – Ground
7 – Motor 1, phase 3-
8 – Ground
9 – Motor 1, phase 4-
10 – Ground
11 – Motor 2, phase 1-
12 – Ground
13 – Motor 2, phase 2-
14 – Ground
15 – Motor 2, phase 3-
16 – Ground
17 – Motor 2, phase 4-
18 – Ground
19 – Input #2 *
20 –
21 – Input #1 *
22 –
23 – Home #1-
24 –
25 – Home #2-
26 – Standby
27 – Output #2- *
28 –
29 –
30 –
31 –
32 –
33 –
34 –
35 –
36 – Output #1- *
37 – Reserved
38 – Reserved
39 – Ground
40 – +5 V

* Motors 1&2 port only

Host Serial Port

Connects C4 to the host RS-232 serial port.
Connector: 9 pin female D-sub connector (DE9).
Cable: Use a 9 pin straight through M-F cable for connection to a PC's serial port.



Pinout:
1 – Jumpered to 4 and 6
2 – Transmit data from C4
3 – Receive data to C4
4 – Jumpered to 1 and 6
5 – Ground
6 – Jumpered to 1 and 4
7 – (RTS) to C4
8 – (CTS) from C4
9 –

Troubleshooting

C4 Won't Respond to Commands
There are many possible reasons for this including improper communication parameters (use 9600-N-8-1), a failed firmware update (retry), improper ID character, etc. If the Power LED is off, then check the power supply.

Reset Default Settings
Internal settings include communication parameters, ID character, and the auto-execute flag. To reset these parameters, put a jumper across pins 1 and 3 of MD2 #2 port. These 2 pins will be on the upper right of the connector - see the connector pinouts page for details. Use a small jumper block commonly used on computer motherboards or a small screw driver. When C4 is powered on and these pins are jumpered, defaults will be set including: baud rate = 9600, ID character = '1', Auto-execute flag = off. Additional parameters may also be reset in newer versions of the software.

Unknown ID Character
You've accidentally set the ID to an unknown character or forgot it. Use the Reset default settings procedure, or execute this command '!!wi1' which will set any connected C4 ID to '1'.

Communication Link Problems
Before using and programming C4, insure that your serial (COM) port is identified and working. If you have a USB-Serial converter, install its software and test its operation.

One way to test a serial connection is to use a terminal program like HyperTerm and jumper the connector to loop back the data. Any characters you type should return to the screen (Make sure automatic Echo in the terminal program is turned Off). On the 9-pin D-Sub serial connector, jumper pins 2 and 3 together (data) and pins 7 and 8 (RTS/CTS). This can be done with alligator clips or by soldering jumper wires on a mating a connector.

Once the loop-back test is positive, remove the loop-back, set the communication parameters to 9600 baud, 8 data bits, no parity and 1 stop bit. Connect the serial cable to C4 and use the terminal program to send commands manually. Try '!1FV' - it should return the firmware version string. Commands must be terminated with a carriage return (Enter).

Once this works, then you can begin sending commands from a program using Visual Basic, C or any other programming language that provides access to the serial port.

Motion Problems
If experiencing lost steps or other motor-motion-related problems, see the MD2 User Guide for details. Typically these are caused by moving a motor too fast or exceeding its payload capacity.

ArrickRobotics.com