MD-2

Dual Stepper Motor System

User's Guide

Revision D

P.O. Box 1574
Hurst, Texas 76053 USA
Ph: (817) 571-4528
Fax: (817) 571-2317
E-mail: info@robotics.com
Web: http://www.robotics.com

Limited Liability Statement

Arrick Robotics shall not be held liable for any incidental, consequential, or direct damages or expenses associated with the use or misuse of its products. Arrick Robotics does not guarantee that any of its products are designed for any particular use or purpose. The entire risk of suitability and performance of all products lies with the user. Products manufactured and/or sold by Arrick Robotics are not authorized for use as critical components in devices used in life support and other systems whose failure or performance could result in compromised safety or danger to life or property.

Please see page 1-35 for warranty information.

ARRICK ROBOTICS

Sections

Section 1 - *The MD-2 System*

Installation and operation of the MD-2 dual stepper motor control system. Information on features of the MD-2 hardware and discussion of technical subjects.

Section 2 - *The MD-2 Program*

Installation and use of the MD-2 program which allows the operator to control up to 6 motors (3 MD-2 systems) using the keyboard or joystick. All motor parameters can be edited and experimented with. Motion control programs can be created automatically and executed from the environment, from batch files, or from the command line.

Section 3 - *Level 1 Subroutine Library*

Information about using the level 1 subroutine library to create simple custom motion control programs in BASIC, Q-Basic, Quick-Basic, Visual-Basic, Pascal and C languages.

Section 4 - *Level 2 Subroutine Library*

Information about using the level 2 subroutine library to create complex custom motion control programs in Quick-Basic, Visual-Basic and C languages. features include ramping, linear and circular interpolation, backlash compensation and more.

Section 1

The MD-2 System

Section 1

The MD-2 System

Table of Contents

Introduction

Congratulations for purchasing the MD-2 Dual Stepper Motor System. This package will provide all of the components necessary to operate stepper motors from an IBM style personal computer including electronics, power supply, motors, cables, software and documentation. Even though this guide covers all MD-2 products, the term 'MD-2' will be used to refer to all models. Any differences between the models will be pointed out when necessary. The MD-2 system is very easy to install and operate as you will find out shortly.

If you have the time and are new to motion control, we suggest you read this entire section to learn the various terms and concepts associated with this subject, then go to the section describing the MD-2 program. If you are already familiar with motion control and your time is limited, we suggest you read through this section until you get to the discussion of parallel printer ports, then go directly to the MD-2 program section to begin operating the system. If your application requires a custom program, you can then select the level 1 or level 2 subroutine libraries and read the appropriate section.

System Requirements

The following computer components are required to use the MD-2 system as intended:

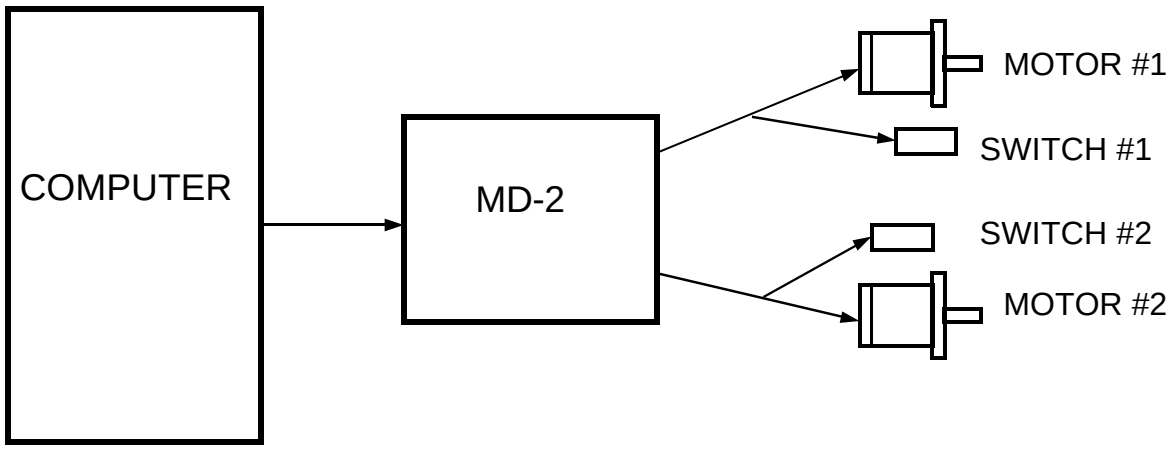
IBM style personal computer (PC, AT, 386 or 486), 286 or higher recommended.

640K of system memory or more.

5.25" (1.2m) or 3.5" (1.44m) high-density floppy disk drive.

Parallel printer port.

DOS 3.0 or higher.



Precautions

The following precautions must be followed carefully. Failure to observe these precautions may result in injury, loss of life and/or, damage to property. Use these precautions as a starting point for safe operation of the MD-2 system. Use common sense whenever using equipment like this to insure safe and productive results. If you have any safety-related questions that aren't addressed here, give us a call.

■ Never attach or remove motor or computer cables while power is applied to the MD-2 unit or the computer.

■ Never apply power to the MD-2 unit while the computer is turned off. Connecting the MD-2 and the computer to the same power strip will eliminate this possibility.

■ Place the MD-2 unit in a well ventilated area to minimize heat buildup. Mount the motors so that heat buildup is minimized.

■ Never use an inappropriate power source.

■ Don't use the Input/Output port incorrectly.

■ Never use the MD-2 with inappropriate equipment or in inappropriate environments.

■ Don't use the MD-2 in situations that could cause danger to life or property

■ Don't replace a blown fuse. This indicates a failure. Return the unit to the factory for repair.

■ Disable any TSR (Terminate and stay resident) programs during MD-2 operation. Not doing this could result in erratic motor speed and lost steps but not damage to the system.

■ When writing custom software, never send an invalid pattern to the MD-2. See the custom software section of this guide for more information.

■ Do not remove the cover of the MD-2 unit or disassemble the motors.

■ Never exceed the specifications of the system.

Failure to observe these precautions could result in injury, loss of life and/or, damage to property.

Hardware Installation

Due to the simplicity of the MD-2 system, installation normally takes less than 20 minutes. The user is required to have average computer experience including a minimal knowledge of DOS commands. It will not be necessary to open your computer.

1 Check your package for the following items,

(1) MD-2 driver box.
(2) Stepper motors with switches.
(2) Stepper motor cables.
(1) 115 VAC power cord.
(1) Parallel printer cable.
(1) Manual.
(1) MD-2 software package.
(1) Warranty registration card.

2 Fill out the warranty registration card and return it! This will insure that you receive information about software updates and new products, and allows us to provide technical support more effectively.

3 Begin by turning off your computer and the power switch on the MD-2 system.

4 Connect the stepper motors to the MD-2 driver box using the motor cables. Never connect or remove the motor cables while power is on.

5 Connect the MD-2 to your parallel printer port using the cable supplied.

6 Plug the MD-2 into a 3-prong, 115 VAC outlet using the AC power cord.

7 You can now turn on your computer, then the MD-2 system. The power light should come on.

8 Insert the MD-2 program diskette into drive A: and type the following at the DOS prompt:

```
A:  (ENTER)
MD2  (ENTER)
```

This is the simplest way to run the MD-2 program, but we suggest that you copy the MD-2 software to your hard disk. See the section covering the MD-2 program for information concerning this. Before moving motors with the MD-2 program, you must select the CALIBRATE item from the OPTIONS menu. You can then enable the MD-2 system, move motors using the keyboard, mouse or joystick and experiment with all motor parameters. The default parameters are normally adequate for initial experimentation, but fine tuning will be required for best results with your particular application. See the MD-2 program section for specific details on operation. On-line help is provided within the program to answer common questions.

Operation

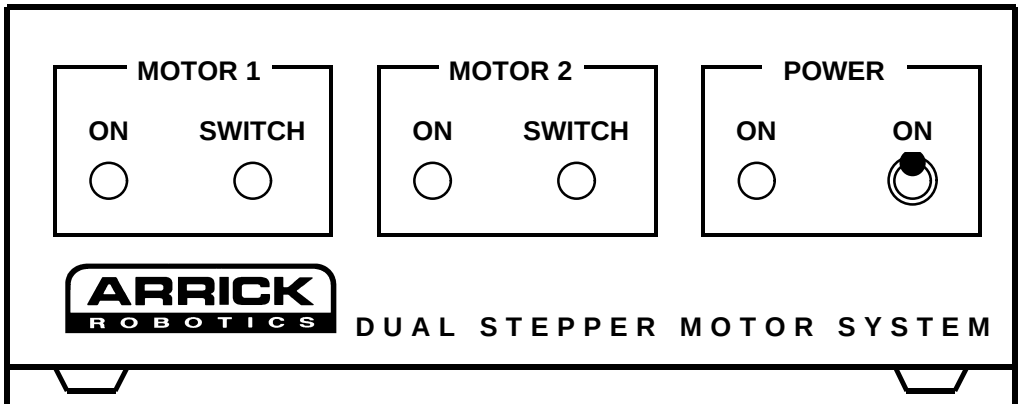
Power Switch
A power switch is provided to control the MD-2 system when it is connected to a 115 VAC outlet. Never apply power to the MD-2 system while the computer is OFF. At any time the power may be turned OFF without harming the motors or electronics and is useful for emergency stops. This switch will not control power to the MD-2 if a battery is being used to provide power.

Power Light
The lights on the front panel of the MD-2 enclosure indicate the current condition of the power supply, motors and switches. The 'POWER' light should be ON whenever 115VAC power is being supplied to the unit and the power switch is in the ON position or when battery power is being applied at the battery connector on the rear panel.

Switch Lights
The 'SWITCH' lights will be ON whenever a switch is pressed. When a switch is pressed, the input signal to the MD-2 is connected to ground (0 volts).

Motor Lights
The 'MOTOR' lights will be ON whenever the motors are energized. These lights do not necessarily indicate motor movement but only that power is applied to them.

See the troubleshooting section if the MD-2 does not operate as expected.



Computer Issues

The software provided with the MD-2 system turns your computer into a powerful motor controller using the parallel printer port. The MD-2 driver box contains the power supply and drive electronics for the motors. The actual motion control functions are performed by the computer and software.

Real-Time Control

To move the motors, the computer must send continuous signals to the MD-2 system. Because of this real-time control, the computer must not be interrupted to perform other tasks while moving the motors. If interruptions occur, the consistency of the signals will be compromised resulting in jerking or rough motor movement and reduced maximum motor speed. The following items can cause problems with these real-time signals -

- Communications port interrupts.
- Mouse interrupts.
- TSR (Terminate and Stay Resident) programs running in the background.
- Hard/floppy disk operation.
- Operating systems that steal processing time such as Windows and OS/2
- Power conservation features.

To minimize these problems, a special parameter has been added to the MD-2 software called 'Leave Interrupts On'. This parameter allows the user to turn off interrupts during motor moves reducing mouse and communications interrupts. Some operating environments such as OS/2 will not let the program disable interrupts which prevents this from working successfully. Check the CONFIG.SYS and AUTOEXEC.BAT files for programs that may operate in the background unexpectedly and remove them when optimum MD-2 performance is required. Some computers have special features that will automatically reduce the computer speed and performance when keyboard or mouse activity is not detected. This will result in a sudden change in motor speeds. Most systems allow these features to be disabled.

Using Windows and OS/2

Operating environments such as Windows and OS/2 are constantly performing operations in the background which can cause problems with a real-time system such as the MD-2. It is possible to greatly enhance the MD-2's performance in these environments by adjusting configuration files, but it is not possible for the system to perform as good as it does in DOS. A PIF (Program Information File) and icon is provided to allow you to operate the MD-2 program in the Windows V3.1 environment. It is necessary to calibrate the MD-2 in this environment if the MD-2 will be used there. Don't use a calibration file that was created in DOS, in another environment such as Windows. Ultimately, experimentation will determine if the MD-2 can be operated at an acceptable level of performance in your environment. Additional information about this subject can be found in your computer manual and operating system manual.

Software

The MD-2 system is provided with software which allows you to perform almost any motion control task imaginable. The software allows the MD-2 system to be controlled interactively with the keyboard and joystick, from the DOS prompt, from batch files, or from custom programs written in BASIC, C and PASCAL.

MD-2 Program

The MD-2 program allows the operator to control the MD-2 dual stepper motor system interactively via the keyboard and joystick. All motor parameters can be edited and experimented with. Motors can be moved and advanced moves such as circles, arcs and grids are also possible. Motion control programs can be loaded, saved, edited and executed. Motion programs as large as 32K can be created automatically using the teach mode which writes code for you. Other features include input bit reading, output bit control, motor speed calibration, standby mode control and port identification. Programs can be run from the DOS command line or from batch files.

Level 1 Subroutine Library

The level 1 subroutine library provides the routines necessary to build simple custom motion control programs in the following languages: Q-Basic, Quick-Basic, GW Basic, Visual-Basic/DOS, Visual-Basic/Windows, C and Pascal. Easily used to create custom programs which integrate motion control with data acquisition equipment. Elaborate features are left out to reduce complexity and program size.

Level 2 Subroutine Library

The level 2 subroutine library provides the routines necessary to create complex motion control subroutine programs in Quick-Basic, Visual-Basic, Visual-Basic and C. Includes complex motion control features such as linear and circular interpolation, ramping, backlash compensation, units conversion and soft limits. A motion program interpreter and parameter editing screen along with other advanced features are also included. The level 2 subroutine library was used as the basis for creating the MD-2 program.

Parallel Printer Ports

The MD-2 system connects to the parallel printer port of your computer. There can be as many as 3 printer ports on your computer. Since each port can be attached to an MD-2, a total of 6 motors can be controlled with a single computer. Each port has its own unique address. The three possible addresses are 3BC, 378 and 278. When adding a new printer port to your computer, make sure that no two ports have the same address. The MD-2 software refers to the motors connected to port 3BC as motors 1 and 2, 378 as motors 3 and 4, port 278 as motors 5 and 6. Your computer may have only one or two ports. Since the motor numbers are determined by which port they are connected to, your system may have motors 3 and 4 but not 1 and 2. you may wish to keep your 3BC port connected to your printer since DOS refers to it as LPT1: or PRN: which is the primary default printer. Parallel printer ports are very inexpensive, easy to install and are normally available at computer stores or by mail-order. There are usually jumpers on the boards that allow you to select the desired port address. If the card has a jumper to set the interrupt request (IRQ), its position is not important for operation with the MD-2 system.

Theory of Operation

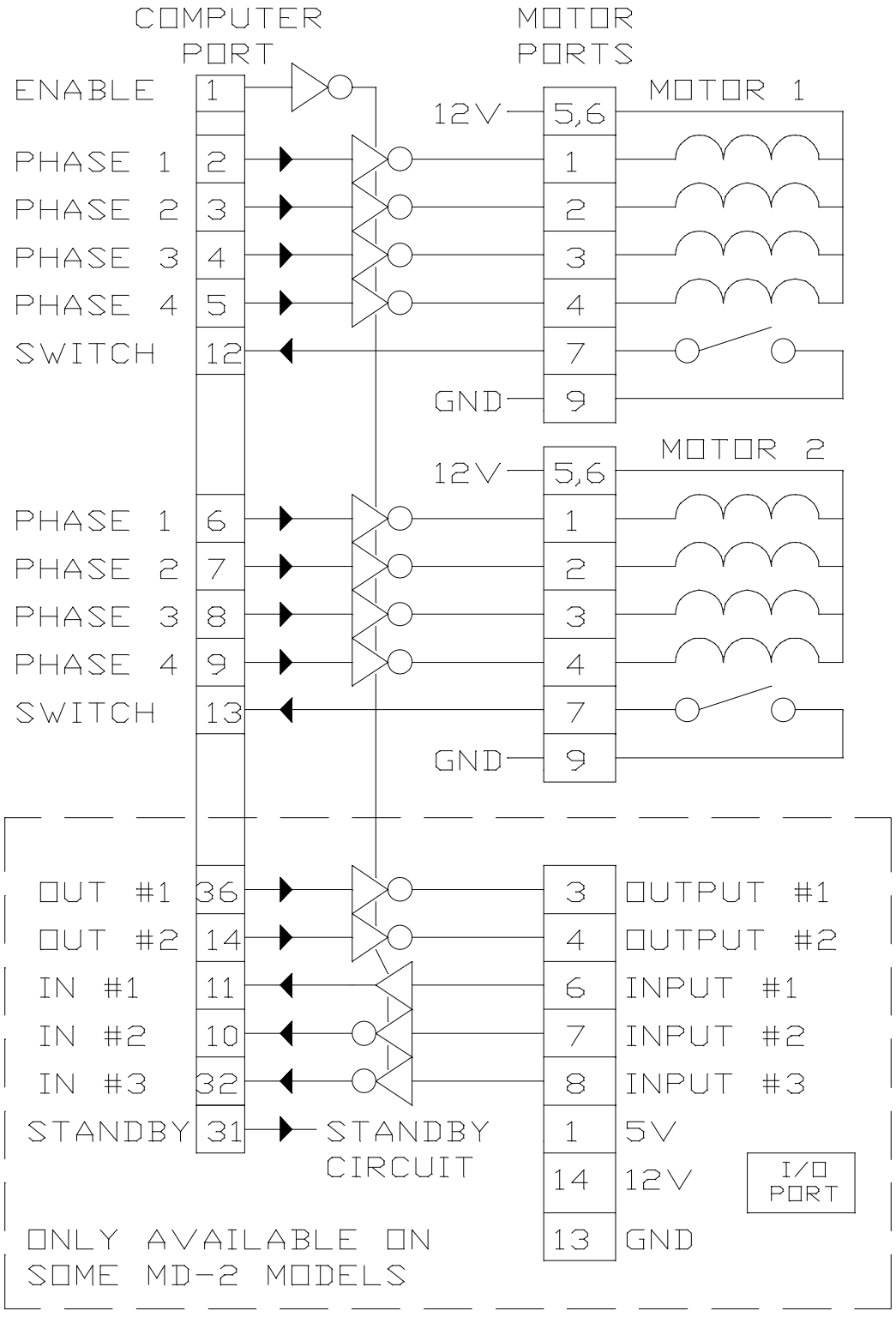
A stepper motor is a special type of motor that lends itself to precise positioning under digital computer control. Unlike servo motors which require encoder feedback, high-speed electronics and expensive amplifiers, a stepper motor can be operated open-loop and without expensive electronics. Although a stepper motor system normally has less performance than a servo system, stepper systems have fewer components, are easier to operate, less expensive and require almost no maintenance.

A stepper motor is unlike other motors in its construction and operation. The motor has four copper coils which create a magnetic field when energized. This field reacts to the permanent magnet connected to the shaft of the motor and causes it to rotate. The sequence in which these coils (or Phases) are energized determines the step angle and direction. The motor will advance one step as each new phase pattern is sent. If the coils are energized but not switching, the motor will remain in the current position and resist rotation. This resistance to motion is referred to as the holding torque. Brushes are not used in stepper motor construction resulting in a long, maintenance-free motor life. Due to their construction, stepper motors can not be harmed or overheated by stalling or binding of the shaft. Motor and driver heating is normal for stepper systems and must be considered during installation and operation. If holding torque is not required, motor heating can be minimized by de-energizing the motor coils when the motion is complete. The running (dynamic) torque of the motor will decrease as the step rate increases. Lost steps will occur if the motor is stepped too fast for a given load, but this will not cause harm to the motor or driver but will cause the computer to loose track of the motor's position. See the torque curves provided in this guide for more information.

A mechanical plunger style switch is attached to each motor cable to provide home position feedback to the computer. When the system is first powered on, a home positioning sequence which uses the switch should be executed under software control to establish the motor position at location zero. All other positions are relative to zero. Optical and other types of switches can also be attached to this input.

The MD-2 system requires a source of control signals to move the motors and to read the switches. Software is provided which will allow a standard IBM style personal computer to perform this function through the parallel printer port. Motion control programs are supplied to control the system along with subroutine libraries for projects requiring custom software. Since as many as three different parallel printer ports can be installed into an IBM style personal computer, 6 motors can be controlled by a single computer.

Functional Diagram



Input / Output Port

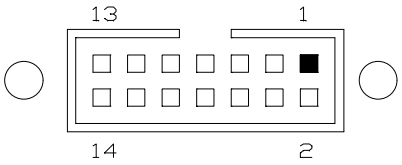
Some MD-2 systems have a miscellaneous input/output (I/O) port which can be used to read digital input signals and to control digital output signals. There are 3 digital inputs which can read signals from sensors and switches. Possible uses for input signals are user-defined push-buttons, over-travel sensing and tool breakage detection. There are 2 digital output signals that can be used to control relays which can, in-turn, control tools such as drills, coolant, lasers and lamps.

All of these signals conform to TTL logic levels meaning that a logic 1 is 5 volts and a logic 0 is 0 volts. Both input and output signals are active low. Connecting a digital input to logic 0 (0 volts) indicates that a switch is activated. This allows the connecting of multiple switches in parallel. Setting a digital output to ON, (0 volts), causes the external device to turn ON.

The I/O port is a 14-pin male, dual-row header connector found on the rear panel of the MD-2 above the computer connector. This type of connector is common and typically connects to standard flat cable.

The I/O port connector also contains several signals that may be useful for external circuits such as +5 volts DC, +12 volts DC, ground and motor switch input signals.

It is important that no more than 25ma of current be drawn from the signals on the I/O port. This should be enough to control solid state relays and other small devices but not enough to drive mechanical relays, solenoids and other larger devices directly. Always protect I/O port signals from high voltages and spikes to prevent damage to the internal MD-2 circuitry.



PIN	DESCRIPTION
1	5 volts DC @ 25ma
3	Output #1, active low
4	Output #2, active low
6	Input #1, active low
7	Input #2, active low
8	Input #3, active low
11	Motor 1 switch, active low
12	Motor 2 switch, active low
13	Ground
14	12 volts DC @ 25ma

Detailed I/O Port Pin Descriptions

The MD-2 system must be enabled before any output signals can be controlled. This prevents the output signals from being turned on when power is initially applied. The MD-2 subroutine libraries provide this initialization code. The following BASIC statements can also be used to enable the MD-2 system. The example program code here uses a base address of 3BC hex, but your port may be at 378 hex or 278 hex.

```
'Turn off outputs and enable the MD-2.
OUT &H3BC+2, &H05

'Disable the MD-2.
OUT &H3BC+2, &H04
```

Pin # 1, 5 VDC @ 25 ma

5 volt DC power supply capable of suppling 25ma of current. This pin can be used to supply small amounts of power to external circuitry. Use this pin in conjunction with output pins to control solid state relays (see below). Drawing more than 25ma of current could cause damage to the MD-2.

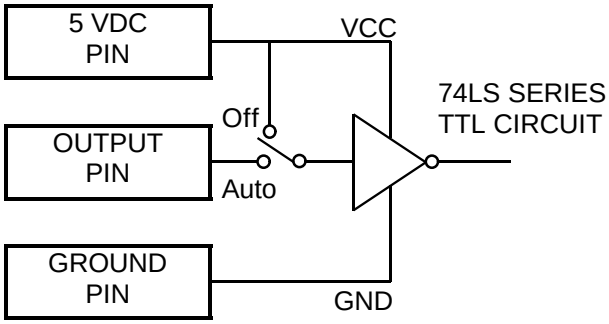
Pin # 3, Output # 1

This TTL level output can be controlled with a simple software 'OUTPUT' command. The drive circuit allows the pin to sink up to 25ma. The pin is active low meaning that low (0 volts) indicates ON. This pin is capable of driving a digital circuit or a solid-state relay which can control other devices. When driving small relays or other inductive loads, connect a flyback diode to protect the internal drive circuit. A manual switch can be placed in the external circuit to allow manual control for safety purposes. Remember to enable the MD-2 system before controlling this and other output signals (See above). The following BASIC statements can be used to control this output signal:

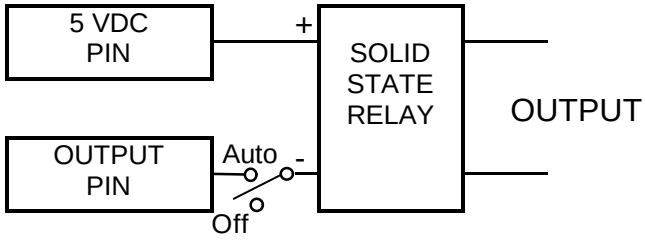
```
'Turn output # 1 pin to low, 0 volts.
OUT &H3BC+2, INP(&H3BC+2) OR &H08

'Turn output # 1 pin to high, 5 volts.
OUT &H3BC+2, INP(&H3BC+2) AND &HF7
```

Typical output circuit diagram

Driving logic circuits



Driving solid-state relays



Use a solid state relay having a 3 volt DC control input such as the CRYDOM D2400 series.

Pin # 4, Output # 2

This output pin is identical to output # 1 in terms of drive capability and usage. The following BASIC statements can be used to control this output signal.

```
'Turn output # 2 pin to low, 0 volts.
OUT &H3BC+2, INP(&H3BC+2) OR &H02

'Turn output # 2 pin to high, 5 volts.
OUT &H3BC+2, INP(&H3BC+2) AND &HFD
```

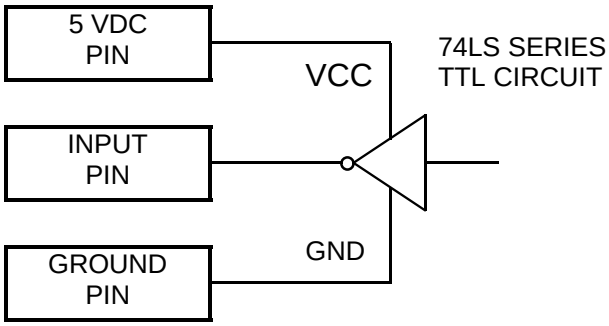
Pin # 6, Input # 1

This TTL level input pin can be used to read switches, sensors and other input devices. Normally this input is simply grounded to detect an input. The following statement can be used to read this input signal.

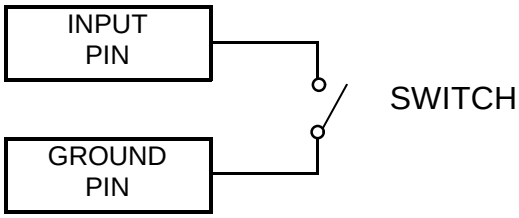
```
PRINT (INP(&H3BC+1) AND &H80)
```

Typical input circuit diagram

Reading logic circuits



Reading switches



Pin # 7, Input # 2

This TTL level input pin has the same capability as other input pins. The following statement can be used to read this input signal.

```
PRINT (INP(&H3BC+1) AND &H40)
```

Pin # 8, Input # 3

This TTL level input pin has the same capability as other input pins. The following statement can be used to read this input signal.

```
PRINT (INP(&H3BC+1) AND &H08)
```

Pin # 11, Motor # 2 switch input

This TTL level input pin has the same capability as other input pins and carries the same input signal that appears on the motor cable. It can be used in addition to, or instead of the motor switch. Normally the input is connected to ground to be detected. The following statement can be used to read this input signal.

```
PRINT (INP(&H3BC+1) AND &H10)
```

Pin # 12, Motor # 1 switch input

This pin is the same as above, but is connected to the motor 1 switch signal. The following statement can be used to read this input signal.

```
PRINT (INP(&H3BC+1) AND &H20)
```

Pin # 13, Ground

This signal should be connected to the external circuitry for correct operation. Input signals can be connected to this ground to be detected.

Pin # 14, 12 VDC @ 25 ma

12 volt DC power supply capable of suppling 25 ma of current. This pin can be used to supply small amounts of external circuitry. Drawing more that 25 ma of current could cause damage.

Holding and Standby Mode

When a stepper motor is energized with a step pattern that is not changing, it produces holding torque which will cause the motor's shaft to resist movement. While the motor is in this holding mode, heat is generated. Some applications require that the motor's shaft resist motion between moves and others do not. If holding torque is not needed, disable the MD-2 which will de-energize both motors and reduce motor heat. If holding torque is needed, keep the MD-2 enabled. The motors and the MD-2 drive electronics are designed to withstand the heat generated.

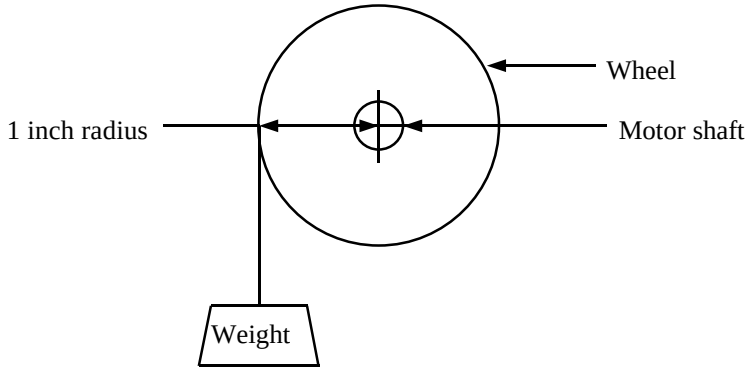
If only partial holding torque is needed, it is possible to place the MD-2 into standby mode. Standby mode is not available on the basic MD-2 system. Standby mode reduces the motor current by 50% which greatly reduces heat buildup while maintaining some holding torque. Trying to move a motor while standby mode is enabled may result in erratic behavior since the torque has been reduced as a result of lower current. Standby mode affects both motors connected to the MD-2 system.

The MD-2 program and the level 2 subroutine library provides control of standby mode. If you are writing a program from scratch, the following BASIC code can be used to control standby mode.

```
'TURN STANDBY MODE OFF.
OUT &H3BC+2, INP(&H3BC+2) OR &H04

'TURN STANDBY MODE ON.
OUT &H3BC+2, INP(&H3BC+2) AND &HFB
```

Motor Torque

Torque is the term used to refer to the rotational strength of a motor. Torque is measured in oz/in, ft/lbs, or any other combination of length and weight. The length indicates the distance from the center of the motor's shaft to the position of the weight.



In this example, the weight is suspended on a string by a wheel that has a radius of 1 inch. The weight is 16 ounces and the length is 1 inch. To move this weight, the motor would have to have at least 16 oz/in of torque. See the specification section for detailed torque information for each MD-2 model.

Holding Torque

Holding torque is the strength of the motor's resistance to rotation while energized but not moving. Holding torque is usually measured with 2 phases energized to provide the maximum rating. If a motor has 50 oz/in of holding torque, it will resist rotation until the weight at the end of a wheel having a 1 inch radius exceeds 50 ounces.

Detent Torque

Detent torque is the motor's residual torque seen when the motor is not energized. This can be experienced by simply turning the shaft of the motor manually while it is disconnected from the driver. A gear reducer will magnify this detent torque and can often cause the mechanical system to resist back-driving even when the motor is off.

Pull-in Torque

The maximum torque that the motor will start and continue to run at without loosing steps. This value is always less than the pull-out torque.

Pull-out Torque

The maximum torque that can be applied to the shaft of the motor and not cause lost steps. This value show the motor's strength after being accelerated.

Torque Curve Chart

A torque curve chart will show how much torque the motor has available at various speeds. The charts will show that the torque decreases as the motor's speed increases. The specification section shows torque curves for all MD-2 models.

Step Types

Step motors operate by energizing their windings (phases) in certain sequences called phase patterns. Changing from one pattern to another causes the motor to move one step. There are 3 different types of phase pattern modes. In half step mode, the motor alternates between one and two motor phases being energized which results in steps that are half the size of full steps. Half steps are usually .9 degrees but differ depending on the motor. Half step mode is the most common since it has twice the resolution of full step modes which reduces vibration. When in half step mode, every other step is stronger than the previous one. There are two types of full step modes. Double-phase full step mode always energizes two motor phases at a time which results in more torque and motor heat. Single-phase full step mode always energizes one motor phase at a time which results in less torque and less motor heat. Both full step modes normally result in steps that are 1.8 degrees. The MD-2 program and the level 2 subroutine library allows you to change the step type used. In most applications half step mode should be used to minimize vibration and increase resolution.

The following charts show the various patterns used to control the 4 motor windings (phases). A zero indicates that a winding is turned ON (energized).

Half Step Phase Pattern

#	4	3	2	1
1	1	1	1	0
2	1	1	0	0
3	1	1	0	1
4	1	0	0	1
5	1	0	1	1
6	0	0	1	1
7	0	1	1	1
8	0	1	1	0

Full-Double Step Phase Pattern

#	4	3	2	1
1	1	1	0	0
2	1	0	0	1
3	0	0	1	1
4	0	1	1	0

Full-Single Step Phase Pattern

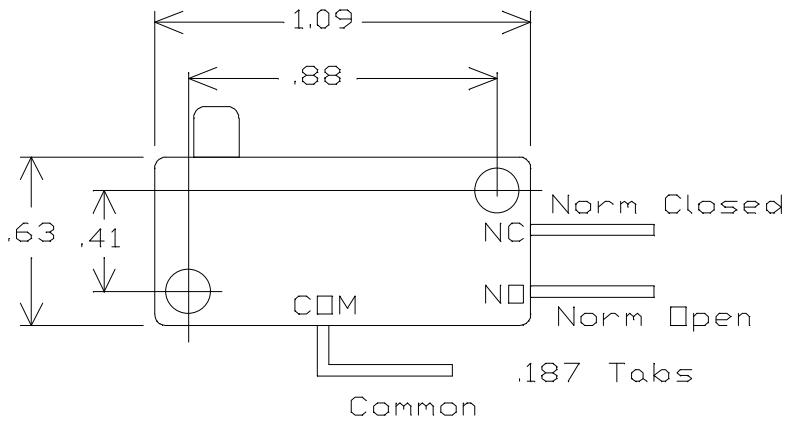
#	4	3	2	1
1	1	1	1	0
2	1	1	0	1
3	1	0	1	1
4	0	1	1	1

Switch Usage

Each motor cable has provisions for one switch input which can be read by the software. The programmer has complete control over the switches and can react in any way upon switch closure or can even choose to ignore it. A mechanical plunger style switch is provided with the system, but other types of switches or sensors can be used.

Normally the switch inputs are used to accomplish 'HOME' positioning when the system is first powered up to establish a reference point for other motion. Software is provided which uses the switches to find the 'HOME' position. The 'HOME' positioning software sequence first moves the motor in reverse until the switch is depressed, then moves the motor forward until the switch is no longer depressed. This sequence has the effect of preloading the mechanical system in the forward direction and can increase the precision of subsequent moves which also approach from this direction.

The switch inputs may also be used as general purpose input which can be read by the software and behave as needed. Optical switches and other sensors may be attached in place of the mechanical switch if needed. Any switch or sensor attached must drive the switch input signal to ground to be detected. See the connector pinout section for detailed information on switch connections.



Battery Usage

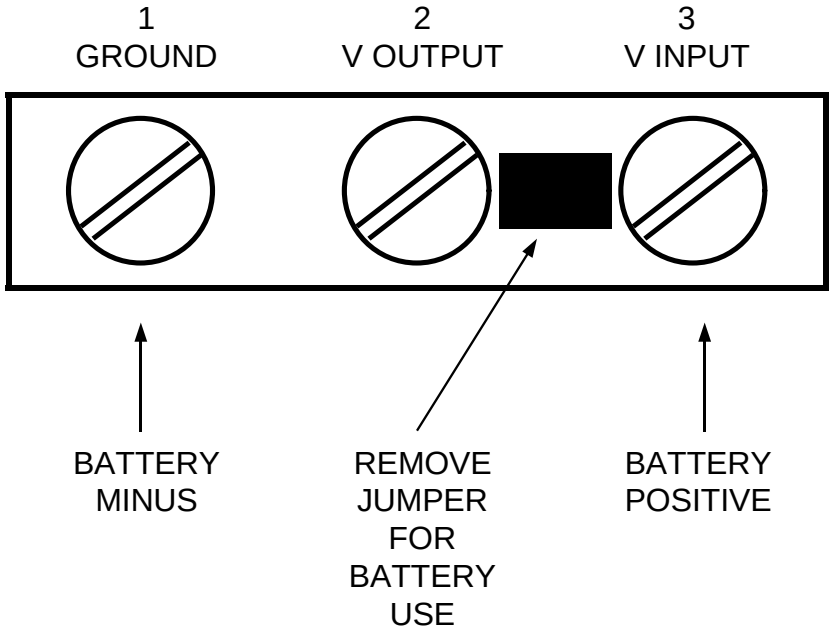
The MD-2 system can be powered by a large battery such as a gel-cel or a car battery when AC power is unavailable. This is useful for remote applications such as telescope positioning. The type and size of battery depends on the MD-2 model and the length of operating time required. The following chart lists the voltages and maximum current requirements. Operation of the MD-2 outside of these voltage ranges may cause damage to the MD-2 system or to the computer.

MD-2	14 +/-2 volts @ 5 amps max.
MD-2a	28 +/-4 volts @ 5 amps max.
MD-2b	28 +/-4 volts @ 13 amps max.
MD-2c	28 +/-4 volts @ 17 amps max.

Most car batteries will produce about 14 volts, two car batteries in series will produce about 28 volts. Several batteries can be placed in parallel to provide power for a longer time.

A 3-screw terminal strip is provided on the rear panel of the MD-2 to attach the external battery. A jumper provides switching between the internal power supply and the external battery. To connect a battery, remove the jumper and attach the ground lead to pin 1 and the positive lead to pin 3 using 14 AWG wire. A fuse should be placed in series with the battery leads for protection against overloads. A switch should also be used since the front panel power switch only operates during AC operation.

1 - Ground
2 - Internally generated power (Output)
3 - Externally applied power (Input)



Connector Pinouts

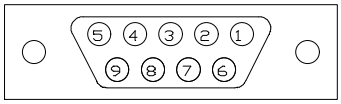
The following connector pinouts are provided for those wishing to attach non-standard equipment to the MD-2 system. Errors in attaching non-standard equipment may result in damage to the computer or the MD-2 system. The battery input connector and the miscellaneous I/O connector pinouts are discussed in sections of this manual dedicated to their function.

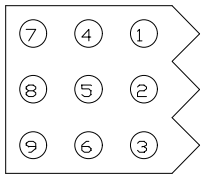
Motor / Switch Connectors

Two motor/switch ports can be found on the MD-2 rear panel. The type of connector used will depend on the current and voltage requirement of the motors supplied with the package. For low power motors, a 9-pin D-Sub connector also known as a DE9 is used. For high power motors, a 9-pin nylon Molex-style connector is used. Most Radio Shack stores will carry both types of connectors along with connector hardware such as hoods and pins.

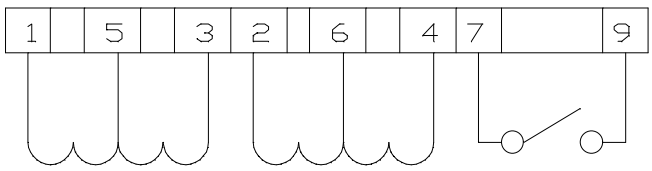
1 - Motor phase 1
2 - Motor phase 2
3 - Motor phase 3
4 - Motor phase 4
5 - Motor phase 1 and 3 common
6 - Motor phase 2 and 4 common
7 - Switch input, active low
8 - +5 volts DC @ 25 ma.
9 - Ground

Connectors as viewed from the rear panel





Typical wiring for the motor and mechanical home switch



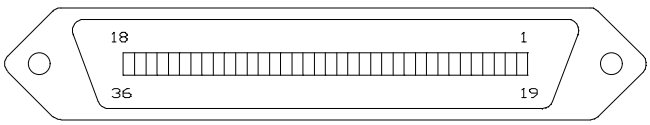
Computer Interface Connector

The computer port on the rear panel of the MD-2 motor driver is designed to connect to an IBM style personal computer's parallel printer port. The computer will have a female 25-pin D-sub style connector. The MD-2 has a 36-pin female centronics style connector like many printers. This arrangement allows easy connection of the MD-2 system using a standard parallel printer cable. The industry standard part number for the mating male 36-pin centronics connector is 57-30360.

CPU 25 PIN	MD-2 36 PIN	NORMAL NAME/USAGE	MD-2 NAME/USAGE
1	1	-Strobe	MD-2 enable, active low
2	2	Data bit 0	Motor 1, phase 1, active low
3	3	Data bit 1	Motor 1, phase 2, active low
4	4	Data bit 2	Motor 1, phase 3, active low
5	5	Data bit 3	Motor 1, phase 4, active low
6	6	Data bit 4	Motor 2, phase 1, active low
7	7	Data bit 5	Motor 2, phase 2, active low
8	8	Data bit 6	Motor 2, phase 3, active low
9	9	Data bit 7	Motor 2, phase 4, active low
10	10	-Acknowledge	* Input # 2
11	11	Busy	* Input # 1
12	12	Paper out	Motor 1 switch, active low
13	13	Select	Motor 2 switch, active low
14	14	-Auto LF	* Output # 2
15	32	-Error	* Input # 3
16	31	-Initialize	* Standby Mode
17	36	-Select In	* Output # 1
18-25	19-25	Ground	Ground

* Used only by models having I/O port and standby feature.

Connector as viewed from the rear panel



Custom Subroutines

This section covers the concepts required to write custom motion control subroutines from scratch. Writing these subroutines will only be necessary for very special applications since most custom software can be written using the level 1 or level 2 subroutine libraries provided. Programmers may wish to read this section to become more familiar with the software even if the subroutine libraries will be used.

Motion control subroutines must manipulate the bits on the parallel printer port that normally go to a printer. Most programming languages provide OUTPUT and INPUT commands to control ports. The software must send the correct sequence of 1's and 0's to control the motor's phases. The software must also control various other bits on the port to provide functions such as MD-2 enable and the reading of the home switches.

There are three possible phase patterns sequences, full step single phase, full step double phase, and half step. See the section on step types for details on the bit patterns used to create these sequences. Sending these patterns to the MD-2 system will cause the motor to step each time the pattern changes. Reversing the sequence will cause the motor to reverse direction. Always continue with the next pattern in the sequence regardless of speed or direction. The following pattern will cause the motor to move in half step mode.

Half Step Phase Pattern

#	4	3	2	1
1	1	1	1	0
2	1	1	0	0
3	1	1	0	1
4	1	0	0	1
5	1	0	1	1
6	0	0	1	1
7	0	1	1	1
8	0	1	1	0

The numbers 4,3,2,1 at the top of the column represent the motor phases. The following chart shows which bit at the port controls which motor phase.

SIGNAL NAME	ADDRESS	BIT	ACTIVE STATE
Motor 1, phase 1	BASE	0	0
Motor 1, phase 2	BASE	1	0
Motor 1, phase 3	BASE	2	0
Motor 1, phase 4	BASE	3	0
Motor 2, phase 1	BASE	4	0
Motor 2, phase 2	BASE	5	0
Motor 2, phase 3	BASE	6	0
Motor 2, phase 4	BASE	7	0
Motor 1, switch	BASE+1	5	0
Motor 2, switch	BASE+1	4	0
MD-2 enable	BASE+2	0	0

It is important not to disturb one motor's bits while controlling another. One way to accomplish this is to read the port, set only the required bits, then write back the entire byte.

Speed control is accomplished by providing a software delay loop between each step. Ramping can be performed by gradually decreasing and increasing the delay as steps are given.

After a motion is complete, the programmer may leave the phases energized with the last step pattern which will result in holding torque. If holding torque is not required, the programmer may read and save the last step pattern issued then de-energize the phases. This will result in a much cooler motor and driver.

An IBM style personal computer can accommodate up to three different parallel printer ports. Their base addresses are 3BC, 378 and 278 hex. The printer port data register which controls the motor's phases resides at this base address. At the base address + 1 resides the status register which is used to read the status of the home switches. At the base address + 2 resides the control register which is used to enable the MD-2 and control output port functions (see the section on the Input/Output port).

MOTORS	BASE ADDRESS-HEX	BASE ADDRESS-DECIMAL
1 & 2	3BC	956
3 & 4	378	888
5 & 6	278	632

The following example program will move motor #1 forward in half step mode continuously.

```
10 A = &H3BC          'SET THE PORT ADDRESS.
20 OUT A, &HFF        'SET ALL PHASES OFF.
30 OUT A + 2, 5       'TURN ON THE MD-2.
40 OUT A, &HFE        'OUTPUT THE STEP PATTERN.
50 GOSUB 1000         'DELAY FOR SPEED CONTROL.
60 OUT A, &HFC
70 GOSUB 1000
80 OUT A, &HFD
90 GOSUB 1000
100 OUT A, &HF9
110 GOSUB 1000
120 OUT A, &HFB
130 GOSUB 1000
140 OUT A, &HF3
150 GOSUB 1000
160 OUT A, &HF7
170 GOSUB 1000
180 OUT A, &HF6
190 GOSUB 1000
200 GOTO 40
1000 FOR I=1 TO 1000 : NEXT I : RETURN
```

It may be necessary to change line 10 to reflect a different port address. To move motor #2, change &HFE to &HEF and so on. To move both motors, change &HFE to &HEE and so on.

It is very important to turn all motor phases OFF before turning the MD-2 system on since the contents of the parallel port data register is unknown and may contain an invalid pattern which could damage the MD-2 system.

The following BASIC statement will print the last pattern issued to the port:

```
PRINT HEX$(INP(&H3BC))
```

The following BASIC statement will turn OFFall phases of both motors:

```
OUT &H3BC, &HFF
```

The switches may be read in software with a few simple statements as this BASIC program example shows:

```
IF (INP(&H3BC+1) AND &H20)=0 THEN
     PRINT "MOTOR 1 SWITCH ON"
ELSE
     PRINT "MOTOR 1 SWITCH OFF"
ENDIF

IF (INP(&H3BC+1) AND &H10)=0 THEN
     PRINT "MOTOR 2 SWITCH ON"
ELSE
     PRINT "MOTOR 2 SWITCH OFF"
ENDIF
```

Other Computers

The MD-2 system was designed for use with an IBM style personal computer but can be used with any computer having 9 bits of digital output and 2 bits of digital input. Use of the input/output port will require another 5 bits. The programmer must have direct and complete control over the bits and signals going to and coming from the port. Most parallel printer ports meet these requirements.

When connecting the MD-2 system to a computer, refer to the connector pinouts in this guide. It is important to use the strobe signal to disable the MD-2 during power-up to prevent an invalid phase pattern from reaching the system. When the strobe signal is low (logic zero) the MD-2 will be enabled.

Use the information in the section describing the writing of custom programs along with the subroutine libraries to create programs for use on other computers.

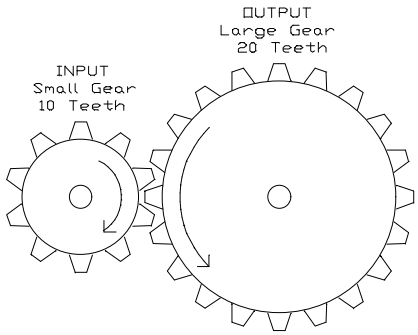
Gear Reduction

Many applications can not be driven directly from the motor's shaft but require some form of reduction to increase torque and resolution at the expense of speed. This reduction may be accomplished with gears, pulleys, friction wheels or some combination of these.

Reduction has several effects on the output of the positioning system.

1. Increase resolution. (The smallest movement possible)
2. Increase torque. (Strength)
3. Decrease speed.

Reduction can be accomplished by connecting a small gear to a large one.

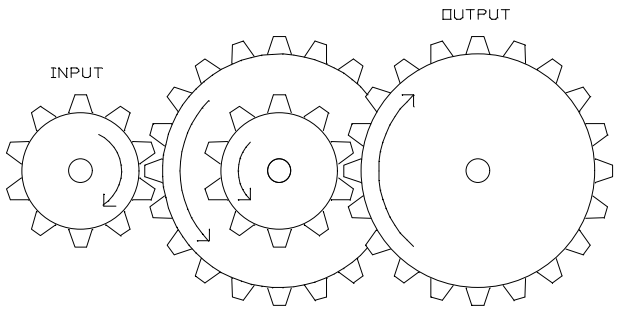


The gear reduction ratio in this example can be found two ways:

1. The ratio of gear diameters.
2. The ratio of gear teeth.

Since there are 10 teeth on the small gear and 20 teeth on the large one, the ratio is 2:1. This arrangement will double the resolution and torque while cutting the speed in half. These same concepts will work with pulleys and friction wheels.

Multiple stages of gearing can accomplish larger reduction ratios as this example shows:



The final gear reduction ratio in this example is: 2:1 X 2:1 = 4:1

Other gear reduction calculations:

Specifications before any reduction:

Resolution:	.9 degrees per step
Torque:	30 oz/inches
Speed:	600 steps per second
	X .9 degrees = 540 degrees per second

Specifications after a reduction ratio of 100 to 1 (100:1)

Resolution:	.009 degrees per step, .54 min, 32.4 sec
Torque:	3000 oz/inches or 15.6 ft/lbs
Speed:	600 steps per second
	@ .009 degrees = 5.4 degrees per second

After a reduction ratio of 1000 to 1 (1000:1) :

Resolution:	.0009 degrees per step, .054 arc min, 3.24 arc sec
Torque:	30000 oz/inches or 156 ft/lbs
Speed:	600 steps per second
	@ .0009 degrees = .54 degrees per second

These calculations make the assumption that the gear box efficiency is 100%, that there is no friction, and that the gear box construction can handle the torque output. The final torque and maximum speed will be somewhat less than the ideal values.

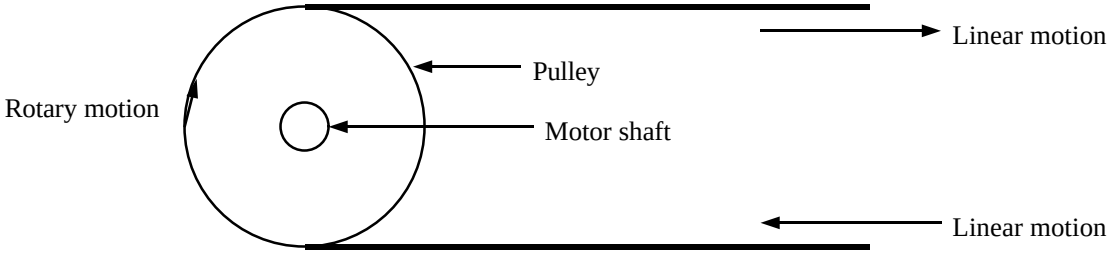
As these examples demonstrate, the effects on resolution, torque and speed are proportional to the gear reduction ratio.

The maximum speed that the motor is capable of moving will vary depending on the friction and inertia of the load. The torque of stepper motors decreases as the speed increases. Lost steps will occur as the motor reaches the maximum speed limit. The user must determine the maximum safe motor speed by experimenting to find the speed at which steps are obviously lost and then reducing it by 30% to 50%. A jerking motion or erratic motor behavior are indicators of lost steps.

Pulley and Belt Drives

Pulley and belt drives can be used to convert rotary motion from a motor into linear motion. The resolution and available torque of the belt-driven application is a function of the diameter of the pulley. The following example shows a pulley with a diameter of .637. The stepper motor has .9 degree steps, then each step of the motor will move the belt .005 inch.

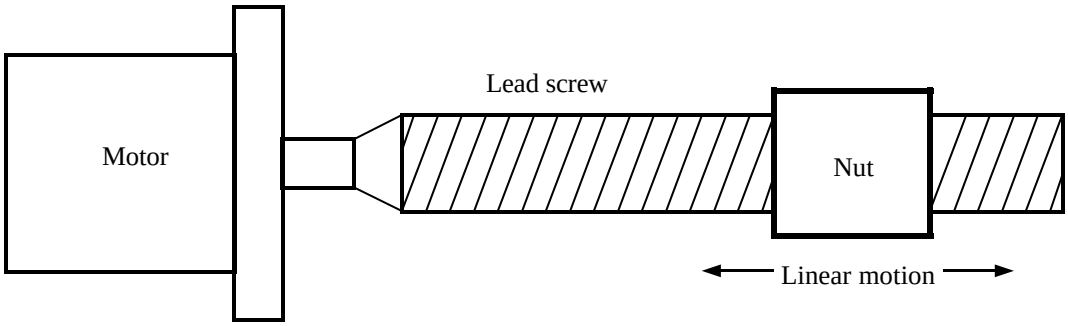
.637dia. X 3.14 = 2 inch circumference / 400 steps per revolution = .005 inch travel per motor step



Lead-Screw Drives

Lead screws are another common way used to convert the rotary motion from a motor to a linear motion. Lead screws normally provide more overall reduction than a pulley drive. The torque is greatly increased along with the resolution at the expense of speed. The efficiency of a lead screw system varies greatly depending on the material used, lubrication and bearing arrangement. This example uses a lead screw with 10 threads per inch.

10 threads X 400 steps per motor revolution = 4000 motor steps per inch of travel.
1 inch / 4000 steps = .00025" of travel per motor step.



Backlash Compensation

Backlash is a term used to describe the amount of mechanical looseness in the positioning system. When gears are used, this looseness is caused by spacing between the teeth of the gears. In systems where belts and pulleys are used, backlash is caused by belt stretch. Bearings, couplings and mounting hardware can also be a source of backlash.

Provisions have been made in the MD-2 program and in the level 2 subroutine library to compensate for backlash. Backlash values are represented in the quantity of steps or units and are used to make corrections whenever the motor's direction changes. The user can determine the backlash by first moving the motor in the forward direction until final application motion is observed, then start moving reverse while counting steps or units and observing motion. The quantity of steps or units required to start the motion of the mechanical system is the backlash value. When this value is entered into the MD-2 software, anytime the direction of the motor changes, this quantity of steps will be added to compensate for backlash. This method can greatly increase a system's bidirectional positioning accuracy.

Suggested Reading

The following list of reading material is provided for those seeking additional information on motion control subjects.

Art and Practice of
Step Motor Control
Albert C. Leenhoust
Intertec Communications Inc.
2472 Eastman Ave. Bldg. 33-34
Ventura, CA 93003
(805) 658-0933
This book contains information about the theory behind step motors and their drive systems. A diskette is included with some BASIC programs for use on an IBM personal computer. Topics include motor construction, drive techniques and microstepping.

Automation Magazine
1100 Superior Ave.
Cleveland, OH 44114-2543
(216) 696-7000
Magazine focused on factory and design automation issues.

Control Engineering Magazine
44 Cook St.
Denver, CO 80206-5191
(303) 388-4511
Monthly publication focusing on industrial control subjects.

DesignFax Magazine
P.O. Box 21640
Eagan, MN 55121-0640
(216) 248-1125
Small magazine containing articles and information about control products intended for design engineers.

Design News
P.O. Box 173377
Denver, CO 80217
(303) 388-4511
Monthly magazine intended for design engineers. Many articles on power transmission and motion control.

Instrumentation & Automation News
P.O. Box 2005
Radnor, PA 19089
(215) 964-4000
This magazine contains various articles and product information for the industrial automation and control field.

Machine Design Magazine
1100 Superior Ave.
Cleveland, OH 44114-2543
(216) 696-7000
Monthly magazine for designers of machines and other products.

Motion Control Magazine
P.O. Box 7907
Wheaton, IL 60188
(708) 858-1888
This magazine contains a large variety of technical information including articles about stepper motors, servos and mechanical systems.

Personal Engineering
and Instrumentation News
P.O. Box 430
Rye, NH 03870
(603) 427-1377
This publication has articles concerning laboratory automation, data acquisition and control.

Scientific Computing
& Automation Magazine
P.O. Box 650
Morris Plains, NJ 07950-0650
(201) 292-5100
Monthly publication discussing control and automation topics.

Component Suppliers

Many users of the MD-2 package are constructing some type of mechanical system which will require gears, bearings, pulleys and various other components. For this reason, the following list of part suppliers has been compiled. Most of the companies listed have catalogs which contain detailed part and technical information and can be obtained at little or no cost.

80/20 Inc.
2570 Commercial Rd.
Fort Wayne, IN 46809
(219) 478-8020
Manufactures aluminum components used to create frames, benches and fixtures.

Allied Devices
2365 Milburn Ave.
Baldwin, NY 11510
(516) 223-9100
Catalog contains over 55,000 items including gears, couplings, speed reducers, dials and shafts.

Bayside Controls
20-02 Utopia Pkwy.
Whitestone, NY 11357
(800) 343-3353
Manufactures precision gear reducers for stepper and servo motors. Catalogs and technical information is available.

Boston Gear
14 Hayard St.
Quincy, MA 02171
(800) 343-3352
A good selection of medium and large gears, pulleys, gear reducers and shaft components. A catalog is available.

Browning Manufacturing
Maysville, KY 41056
(606) 564-2011
Much like Boston Gear, this company carries mostly larger components such as gears and pulleys.

EFD
East Providence, R.I. 02914
(401) 434-1680
Manufactures hand-held dispensing devices used to apply adhesive, solvents and other fluids.

Foredom
Bethel, CT 06801
(203) 792-8622
Manufactures hand-held rotary power tools and accessories.

Igus Inc.
P.O. Box 14349
East Providence, R.I. 02914
(401) 438-2200
Manufactures cable and hose carriers.

Helical Products
901 W. McCoy Lane
Santa Maria, CA 93456
(805) 928-3851
Manufactures precision shaft couplers.

Lovejoy, Inc.
2655 Wisconsin Ave.
Downers Grove, IL 60515
(708) 852-0500
Manufactures a variety of products including shaft couplers.

Martin Sprocket and Gear
P.O. Box 888
Arlington, TX 76004
(817) 465-6377
Stocks medium to large sprockets, gears and other components.

Nordex
50 Newtown Rd.
Danbury, CT 06810-6216
(203) 792-9050
Nordex is an excellent source for small gears, bearings, shafts and various other precision components at a reasonable cost.

PIC Design
P.O. Box 1004
Middlebury, CT 06762
(203) 758-8272
PIC Design is also known as Precision Industrial Components and stocks a wide variety of gears, pulleys, bearings and lead screw assemblies.

Plastock
Three Oak Rd.
Fairfield, NJ 07006
(203) 928-7911
Manufactures plastic gears and pulleys.

SAVA Industries
70 Riverdale Rd.
Riverdale, NJ 07457
(201) 835-0882
Manufactures cables and pulleys.

SECS
520 Homestead Ave.
Mt. Vernon, NY 10550
(914) 667-5600
A good source for gears, handles, dials and timing pulleys.

Seitz
Torrington Industrial Lane
Mt. Vernon, NY 10550
(203) 489-0476
Carries small gears, bearings, pulleys and other precision components.

Small Parts
6891 N.E. 3rd Ave.
P.O. Box 381736
Miami, FL 33238-1736
(305) 751-0856
This company stocks a broad range of precision parts such as screws, tubing, and tools.

Solidur Plastics
200 Industrial Dr.
Delmont, PA 15626
(800) 343-0444
Manufactures gears and other parts from plastic.

Stock Drive Products
2101 Jericho Turnpike
New Hyde Park, NY 11040
(516) 328-0200
This company probably has the broadest line of precision mechanical components available including gears, pulleys, bearings and hardware. Metric sizes are also available. Several catalogs and technical books are available.

Winfred M. Berg
499 Ocean Ave.
East Rockaway, NY 11518
(516) 599-5010
The Berg catalog contains gears, bearings and large assortment of unusual belts and pulley systems.

Troubleshooting

The following list describes the most common problems and their remedies. Repairs should be left to qualified persons.

Problem: Power light does not light.
Remedy: The MD-2 driver is probably not getting power. Check the power cord and fuse. If the fuse is blown, replace with one of the exact same type and value. If the fuse blows a second time, repairs must be made. If a battery is being used, make sure the voltage is correct at the terminal strip during operation. Make sure the power jumper is removed when using a battery and connected when using AC.

Problem: Power light turns on but the motor lights do not and the motors do not move.
Remedy: The motors are not being turned on by the control computer. Check the cable connecting the MD-2 to the computer. If custom software is being used, return to the MD-2 program to verify operation. Check the port address in the software.

Problem: Motor lights turn on but the motors do not move.
Remedy: Check the motor cables for breaks or cuts. Swapping cables and motors using the process of elimination will determine which component is defective. Return to non-custom software to eliminate the possibility of software problems. Also, check the motor speed values since giving step pulses too fast will not allow the motors to move.

Problem: Activating the 'HOME' switches does not turn on the switch lights.
Remedy: Check the wiring of the switches and the motor cables. The switch must drive the input to ground to be detected.

Problem: Activating the switches turns on the lights but the software does not recognize them.
Remedy: Check the cable that connects the MD-2 to the computer. Check for software errors and verify the port address.

Problem: The heat generated by the motors and/or driver must be minimized.
Remedy: Motor and driver heating is normal and occurs mostly during standstill. The motors can be mounted to a metal plate to help dissipate the heat and a fan can be located near the MD-2 driver enclosure to carry away the heat. The software can be configured to de-energize the motor coils during standstill when holding torque is not required. This will greatly reduce motor and driver heating.

Problem: As the load increases the motor sometimes misses steps causing inaccurate positioning.
Remedy: The available torque of a stepper motor decreases as the speed increases. Decrease the speed of the motor to increase the available torque which will reduce the possibility of lost steps. The motor must have enough torque to withstand all momentary requirements.

Problem: The motors move but the speed is not smooth.
Remedy: If another program is consuming the computer while motors are moving, erratic motion will result. Disable any TSR (Terminate and stay resident) programs or other programs that work in the background.

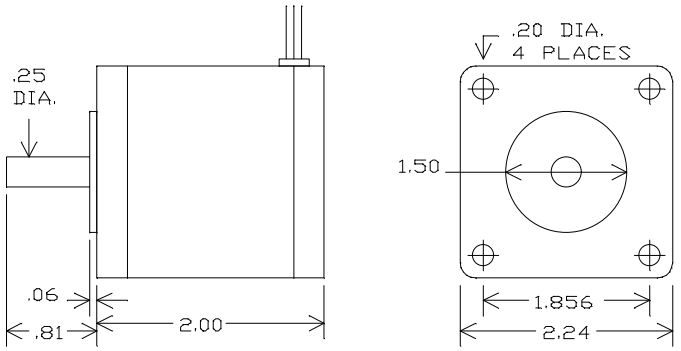
MD-2 Specifications

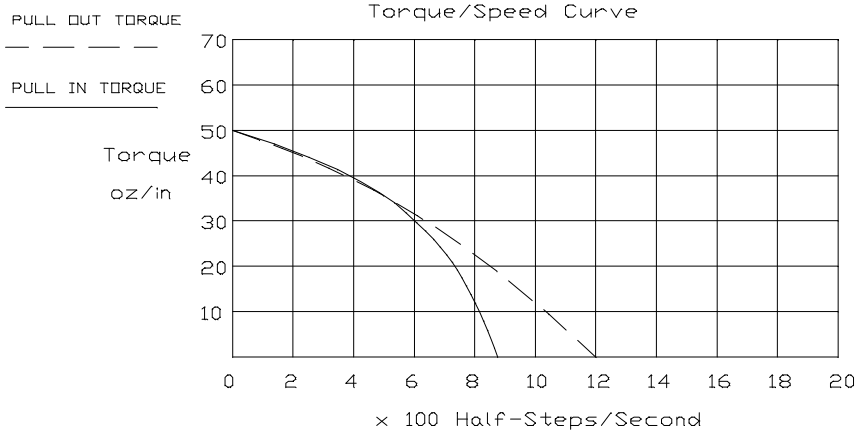
Driver Unit

Size:	8.5" x 3" x 6.5"
Weight:	7 lbs.
Lamps:	Power, motor and switch status.
Power Input:	115 VAC, 50-60 Hz @ 1 Amp or 12 VDC @ 5 amps.
Motor Port:	9 pin D-sub female.
CPU Port:	36 pin centronics female.
Drive Type:	Unipolar L/3R resistance limited.

Motors

Size:	Nema frame #23, 2.25" diameter, 2.25" long,
Shaft:	.25" diameter, .75" long.
Mounting:	4 holes, .2" diameter, 1.856" square pattern.
Weight:	21 ounces.
Windings:	Unipolar, 5.1 volt, 1 amp
Connector:	9 pin male D-sub.
Cable:	9 conductor, 22 AWG.
Full step:	1.8 degrees, +/- 5% accuracy.
Half step:	.9 degrees, +/- 5% accuracy.
Detent Torque:	1 oz/in.
Holding Torque:	50 oz/in.





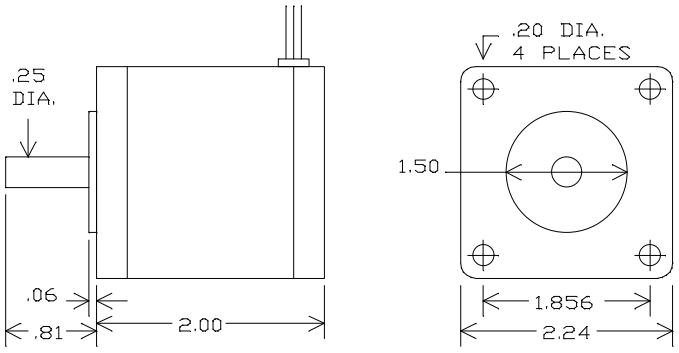
MD-2a Specifications

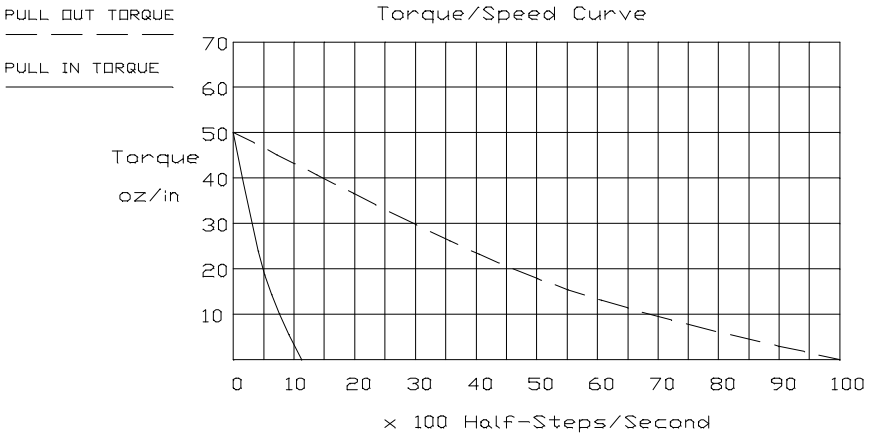
Driver Unit

Size:	8.5" x 3" x 6.5"
Weight:	7 lbs.
Lamps:	Power, motor and switch status.
Power Input:	115 VAC, 50-60 Hz @ 1 Amp or 28 VDC @ 5 amps.
Motor Port:	9 pin D-sub female.
CPU Port:	36 pin centronics female.
Drive Type:	Unipolar chopper.

Motors

Size:	Nema frame #23, 2.25" diameter, 2.25" long,
Shaft:	.25" diameter, .75" long.
Mounting:	4 holes, .2" diameter, 1.856" square pattern.
Weight:	21 ounces.
Windings:	Unipolar, 5.1 volt, 1 amp
Connector:	9 pin male D-sub.
Cable:	9 conductor, 22 AWG.
Full step:	1.8 degrees, +/- 5% accuracy.
Half step:	.9 degrees, +/- 5% accuracy.
Detent Torque:	1 oz/in.
Holding Torque:	50 oz/in.





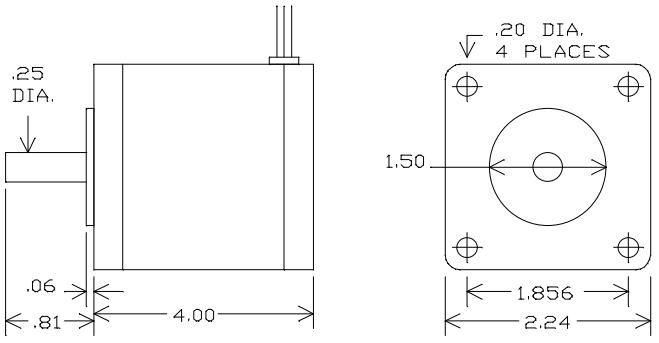
MD-2b Specifications

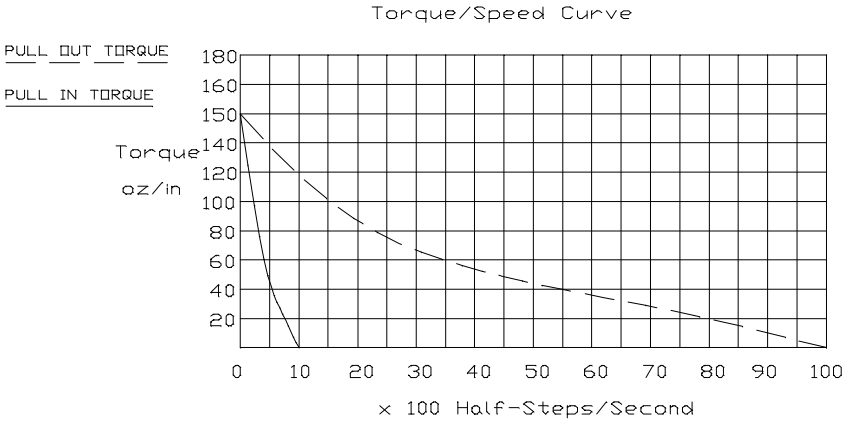
Driver Unit

Size:	8.5" x 3" x 6.5"
Weight:	10 lbs.
Lamps:	Power, motor and switch status.
Power Input:	115 VAC, 50-60 Hz @ 1 Amp or 28 VDC @ 13 amps.
Motor Port:	9 pin Molex female.
CPU Port:	36 pin centronics female.
Drive Type:	Unipolar chopper.

Motors

Size:	Nema frame #23, 2.25" diameter, 4" long,
Shaft:	.25" diameter, .75" long.
Mounting:	4 holes, .2" diameter, 1.856" square pattern.
Weight:	48 ounces.
Windings:	Unipolar, 3.4 volt, 2.9 amp
Connector:	9 pin male Molex.
Cable:	9 conductor, 18 AWG.
Full step:	1.8 degrees, +/- 5% accuracy.
Half step:	.9 degrees, +/- 5% accuracy.
Detent Torque:	1 oz/in.
Holding Torque:	150 oz/in.





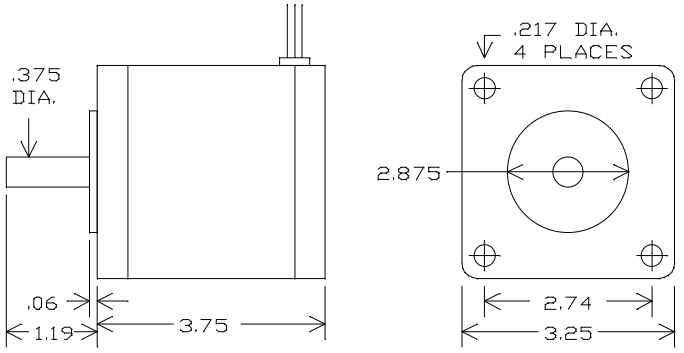
MD-2c Specifications

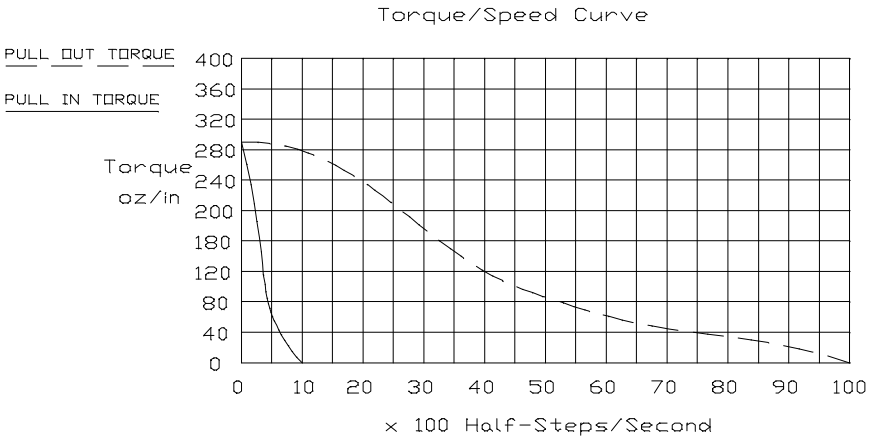
Driver Unit

Size:	8.5" x 3" x 6.5"
Weight:	10 lbs.
Lamps:	Power, motor and switch status.
Power Input:	115 VAC, 50-60 Hz @ 1 Amp or 28 VDC @ 17 amps.
Motor Port:	9 pin Molex female.
CPU Port:	36 pin centronics female.
Drive Type:	Unipolar chopper.

Motors

Size:	Nema frame #34, 3.4" diameter, 3.75" long,
Shaft:	.375" diameter, 1.25" long.
Mounting:	4 holes, .2" diameter, 2.739" square pattern.
Weight:	80 ounces.
Windings:	Unipolar, 3 volt, 4 amp
Connector:	9 pin male Molex.
Cable:	9 conductor, 18 AWG.
Full step:	1.8 degrees, +/- 5% accuracy.
Half step:	.9 degrees, +/- 5% accuracy.
Detent Torque:	1 oz/in.
Holding Torque:	300 oz/in.





Warranty Information

ARRICK ROBOTICS warrantees this product to be in good working order for a period of one (1) year from the date of purchase. Should this product fail to be in good working order at any time during this period, ARRICK ROBOTICS will, at its option, repair or replace the product at no additional charge except as set forth below. This limited warranty does not include service to repair damage to the product resulting from accident, disaster, misuse, abuse, or modification of the product. To obtain warranty service, send the product along with proof of purchase in its original packaging to:

ARRICK ROBOTICS
Attn: Repair Dept.
2107 W. Euless Blvd.
Euless, TX 76040

You agree to prepay shipping charges and to insure the product or assume the risk of loss or damage in transit. All express or implied warranties for this product including the warranties of merchantability and fitness for a particular purpose are limited in duration to a period of one (1) year from the date of purchase, and no warranties, whether expressed or implied, will apply after this period.

If this product is not in good working order as warranted above, your sole remedy shall be repair or replacement as provided above. In no event will ARRICK ROBOTICS be liable to you for damages, including any lost profits, lost savings or other incidental or consequential damages arising out of the use of or inability to use this product.

Some states do not allow limitations on how long an implied warranty lasts, so the above limitations may not apply to you. Some states do not allow the exclusion or limitation of incidental or consequential damages for consumer products, so the above limitations may not apply to you. This warranty gives you specific legal rights and you may also have other rights which may vary from state to state.

Section 2

The MD-2 Program

Section 2

The MD-2 Program

Table of Contents

Introduction

The MD-2 program allows the operator to control the MD-2 dual stepper motor system interactively via the keyboard and joystick. All motor parameters can be edited and experimented with. Motors can be moved and advanced moves such as circles, arcs and grids are also possible. Motion control programs can be loaded, saved, edited and executed. Motion programs as large as 32K can be created automatically using the teach mode which writes code for you. Other features include input bit reading, output bit control, motor speed calibration, standby mode control and port identification.

Many of the items mentioned in this section are covered in detail in the first section.

MD-2 Program Installation

It is possible to run the MD-2 program from a floppy disk but installing it on your hard disk will provide the best performance. Installing the MD-2 program on a hard disk is done by making a subdirectory and coping the files from the floppy disk into the newly created subdirectory. The following example assumes that the floppy disk drive is A: and the hard disk is C:.

```
MD C:\MD2   (ENTER)
COPY A:*.* C:\MD2   (ENTER)
```

The MD-2 program consists of the following files:

MD2.EXE	The MD-2 program executable file.
MD2.HLP	On-line help system file.
MD2.PAR	Default motor parameter file.
MD2.BAS	Default motion program file.
MD2.CAL	Calibration file. (exists only after performing calibration)

Command Line

To invoke the MD-2 program, change to the directory where the program and its files are located and type MD2. The following example assumes the program and files reside in a directory named MD2.

CD \MD2 (ENTER)
MD2 (ENTER)

Automatically Load and Run a Program File

It is possible to invoke the MD-2 program and automatically load and run a motion control program. This powerful feature allows you to perform complex motion control functions with one entry at the DOS command line and allows you to place them along with other programs in batch files. The following example will run a motion control program named DRILL17.BAS from the DOS command line. The MD-2 program will give control back to DOS when the motion control program is finished. The calibration file and any parameter files must be located in the current directory.

MD2 DRILL17.BAS (ENTER)

Automatically Execute Commands

It is also possible to invoke the MD-2 program and automatically execute motion control commands directly from the DOS command line. This allows you to perform simple motion control functions without the need for a program file. The following example shows the use of this feature to move motor 3 home. The default calibration file and the default parameter file (MD2.PAR) is loaded before the command line is executed. The line can include any valid motion control commands and must begin and end with double quotation marks. See the section on motion programs for command descriptions.

MD2 "MD2MOTOR=3 : MD2ON : MD2HOME : MD2OFF" (ENTER)

High Resolution Screen Mode

If you have a color or monochrome VGA style monitor, you can select high-resolution mode before running the MD-2 program. The MD-2 program detects this mode and allows you to view both the main screen and the editor screen or parameter screen simultaneously. Use the following DOS command before invoking the MD-2 program to activate high-resolution mode. This only applies when the MD-2 program is started without command line instructions or a command line file name since the screen is not shown when using those features.

MODE CON: LINES=50 (ENTER)

See your DOS operator's manual for more information on the mode command and screen resolutions. You can return to the standard screen mode by using the following DOS command.

MODE CON: LINES=25 (ENTER)

Using the MD-2 Program

To use the MD-2 program, the MD-2 driver must be connected to your personal computer using a standard parallel printer cable. Motor cables and the AC power cable must also be attached. See the section describing the hardware installation for specific details.

The first thing that must be done before moving the motors is calibration. This process is required for motor speed calculations. You only need to calibrate once since calibration information is stored in a disk file named MD2.CAL and is automatically read by the MD-2 program when started. Select CALIBRATE from the OPTIONS menu by using the mouse or by holding down the ALT key while pressing O then pressing C to select CALIBRATE. Calibration takes about 10 minutes. See the section on calibration for detailed information.

Next, you must determine which motor numbers are available on your computer. This depends on the address of the parallel printer port. In the motor name column, "no port" will be listed if that port doesn't exist. See the section describing ports and motor numbers for details.

Before actually moving motors you must enable the MD-2 by selecting the ENABLE check box using the mouse or by pressing ALT 1, 3 or 5, or by pressing simply 1, 3 or 5. Selecting again will disable the MD-2. See the section describing enabling and disabling the MD-2 system for more information.

After the MD-2 system is enabled, the function keys can be used to move the motors. Each time a function key is pressed, the desired motor will move according to the DISTANCE parameter. Click the PARAMETER button with the mouse or type P to see the motor parameter screen (Press the ESCAPE key to return to the main screen). Motor speed and other parameters can be viewed, changed and saved. Default parameters should be adequate to move motors initially for experimentation. You can change the parameters to suit your application then save the parameters to a disk file. The parameter file named MD2.PAR is loaded automatically when the MD-2 program begins. You can save your parameters under this name or choose a different file name. This allows you to maintain a separate parameter file for each application.

When the MD-2 program is started, the default motion program file named MD2.BAS is loaded into the editor. You can enter the program editor by clicking the EDIT button, pressing ALT-E, or simply E (Press ESCAPE to return to the main screen). You can modify this file and save it under the same name or load another. See the section describing motion programs for more information.

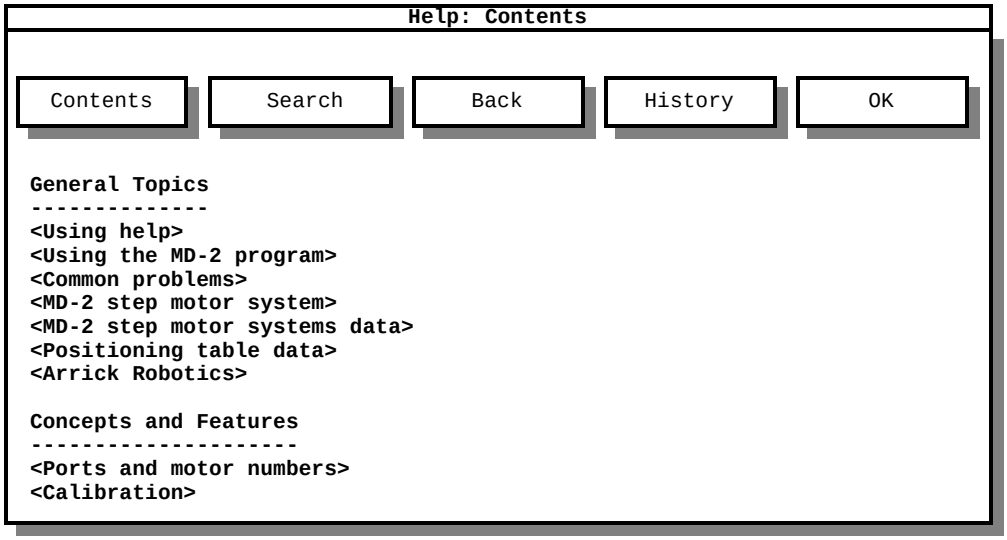
Each feature of the MD-2 program is described in its own section - see the table of contents.

On-Line Help

The MD-2 program contains an on-line help function which can provide you with program information without the need for a manual. The help system can be accessed with the mouse or keyboard and provides a host of features including hyper-text links which allow you to easily move between related topics, a search function that helps you find specific information, and a history screen which lets you to review previously displayed topics.

The MD2.HLP file must be in the current directory to be used by the MD-2 program. If not found, the help system will not be available during program execution.

Most dialog boxes have a help button for direct access to specific information about the selected function. You can also activate the help system by selecting the HELP menu from the main MD-2 screen or the program editor screen. The menu gives you the option of displaying the CONTENTS of the help file, searching for a specific topic, displaying information about how to use the help system, and information about the version of the MD-2 program. When the help screen is displayed, click on the topic using the left mouse button or press the TAB key to select a topic then press ENTER to view that topic. Clicking the BACK button or pressing the B key will display the previous topic. Clicking the HISTORY button or pressing the H key will show all previously displayed topics. Clicking the SEARCH button or pressing the S key will display the search screen. Click the OK button or press the O key to close the help system and return to the program.



Definitions

The following list of definitions should help you become acquainted with the MD-2 system quickly.

Absolute Moves: A type of motor move command where the desired target position is given. The distance and direction the motor must move depends on the current position.

Backlash: Spacing between gear teeth or belt stretch which causes looseness in the mechanical system. Mainly seen when the motor changes direction.

Calibration: Used to perform speed calculations and compensate for various computer speeds. Needed only once.

Circular Moves: A type of motor motion where two motors move together in such a way as to cause a circular pattern to be created on an XY positioning table. Parts of circles can also be requested creating arcs. Various X and Y radii can be given to produce ellipses.

Command Line: Information typed at the DOS prompt to invoke the MD-2 program and give special instructions.

Enable/Disable: The MD-2 system must be enabled before motors can be controlled.

Grid Moves: A type of motor move where two motors are told which column and row of a previously defined grid to move to. Helpful when an application contains columns and rows of items such as a rack of test tubes.

Holding Torque: Motor strength available while the motor is at standstill and energized by the system.

Input/Output Port: Connector on some MD-2 systems that allows external devices to be controlled with the computer and external switches and sensors to be detected.

Limits: The forward and reverse positions at which a motor should not travel beyond.

Line Moves: A type of motor motion where two motors move together in such a way as to cause a straight line to be created on an XY positioning table.

On-Line Help: Documentation built into the program which is accessible with the keyboard or mouse. Often eliminates the need for a manual.

Parameters: Values which control the behavior of the motors such as speed, direction, distance, and limits.

Ports: Connector on a computer or the MD-2 system which is used to communicate information.

Program: Term used when referring to a motion control program which contains commands used to control the MD-2 system. The commands are similar to BASIC language commands. Programs can contain sequences of commands used to perform complex motion control tasks. Programs can be run from the MD-2 program, from the DOS command line, from batch files, or, from Quick-Basic or Visual Basic with minor modifications.

Ramping: Accelerating and decelerating the motor during movement. Used to gradually increase the speed of a motor to the top speed then decreasing the speed before stopping. Causes motion to be smoother and allows larger payloads to be moved at higher speeds.

Relative Moves: A type of motor move where the distance and direction is given relative to the current position.

Single Step Mode: Used to cause the motors to move one step at a time under manual control to position to a specific location.

Size 23, 34 etc: Industry standard motor frame size. Common sizes are 17, 23, 34 and 42. Allows motors from different manufacturers to be interchanged easily.

Standby Mode: Used to reduce the amount of current given to the motors in order to minimize heat buildup while maintaining some holding torque.

Step Motors: A type of motor which moves in small increments known as steps. Steps are normally .9 degrees but can differ depending on motor construction.

Step Types: The type of phase patterns or signals sent to a step motor to cause it to move. Half step, full-double and full-single are possible step types with half-step being the most common.

Teach Mode: A mode in which a motion control program is automatically created while the operator controls the MD-2 system manually. Used to create complex programs without typing commands into the program editor.

Units: Inches, feet, degrees or other measurements. The MD-2 program allows the user to define how many steps are in each unit. This allows the operator to deal with units instead of steps.

The Main Screen

The main screen is shown when the MD-2 program is first started. From this screen, motors can be moved, input bits read, output bits controlled, teach mode can be activated, the joystick can be activated and many other major features controlled.

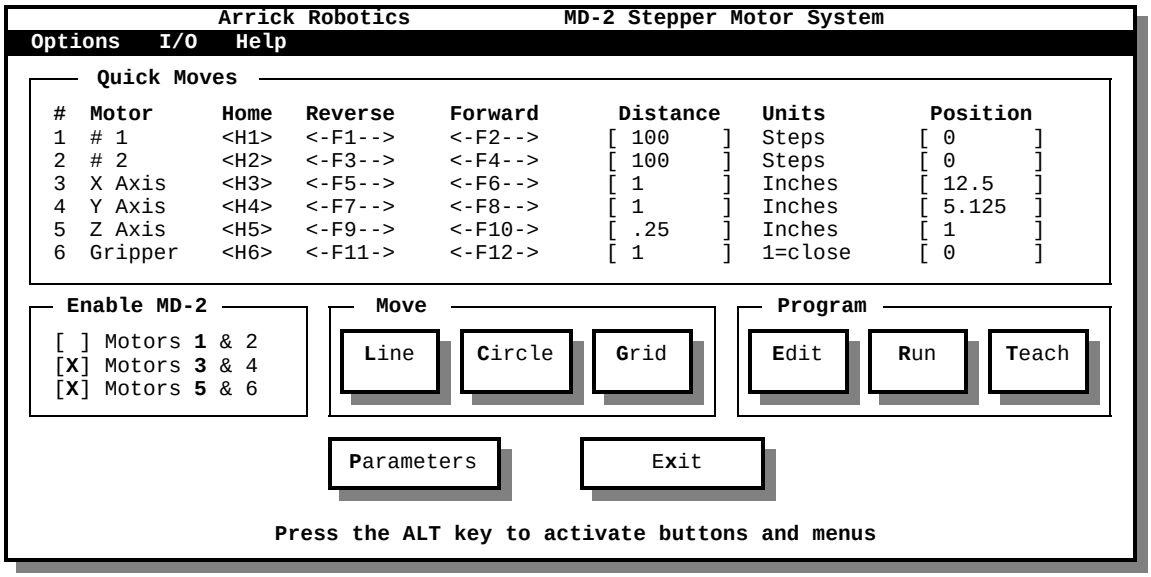
The screen consists of a menu bar across the top, a Quick-move control box directly underneath, and a series of button and check box controls. All features can be activated with the keyboard or the mouse.

Menu items can be activated by clicking them with the left mouse button or by pressing the ALT key along with the highlighted activation letter in the menu name. For example, to activate the OPTIONS menu with the keyboard, press the ALT key and hold it while pressing O. The menu will open showing several selections. To activate a menu selection, press the highlighted access key for that selection or use the up and down arrow keys to highlight the selection and press ENTER.

Control buttons are activated by clicking them with the left mouse button or by pressing the ALT key along with the highlighted access key. For example, to exit the program, press the ALT key while pressing the X key. As a short cut, you may also simply press X. Other controls can also be activated by simply pressing the highlighted access key without using the ALT key.

To activate the ENABLE check boxes, click using the left mouse button or press the ALT key while pressing either 1, 3 or 5 depending on the desired action. Activation will be indicated by an X placed in the check box. Activating the box again will then disable it. As a short cut, you may simply press 1, 3 or 5. The short cut key will not work if the cursor is placed in a distance box or position box since 1, 3 and 5 are valid numeric entries.

To see the motor parameter screen, activate the PARAMETERS button or press P. As a short cut, press the ESCAPE key to go to the parameter screen. Press the ESCAPE key again to return to the main screen. To see the program editor screen, activate the EDIT button or press ALT-E, or simply E. Press the ESCAPE key to return to the main screen.



The Motor Parameter Screen

Activate the parameter screen by clicking the PARAMETERS button, by pressing ALT-P, P, or by pressing the ESCAPE key. The main screen and the parameter screen can be viewed together if in high-resolution mode. Pressing the ESCAPE key while in the parameter screen will return control to the main screen. All motor parameters except circle and grid parameters can be viewed and edited. Parameter files can be loaded from disk and saved to disk. When the MD-2 program is started, the default parameter file named MD2.PAR is loaded. You may continue to use this name for parameter storage or choose another. Using the .PAR extension will make it easy to identify the parameter file although it is not mandatory.

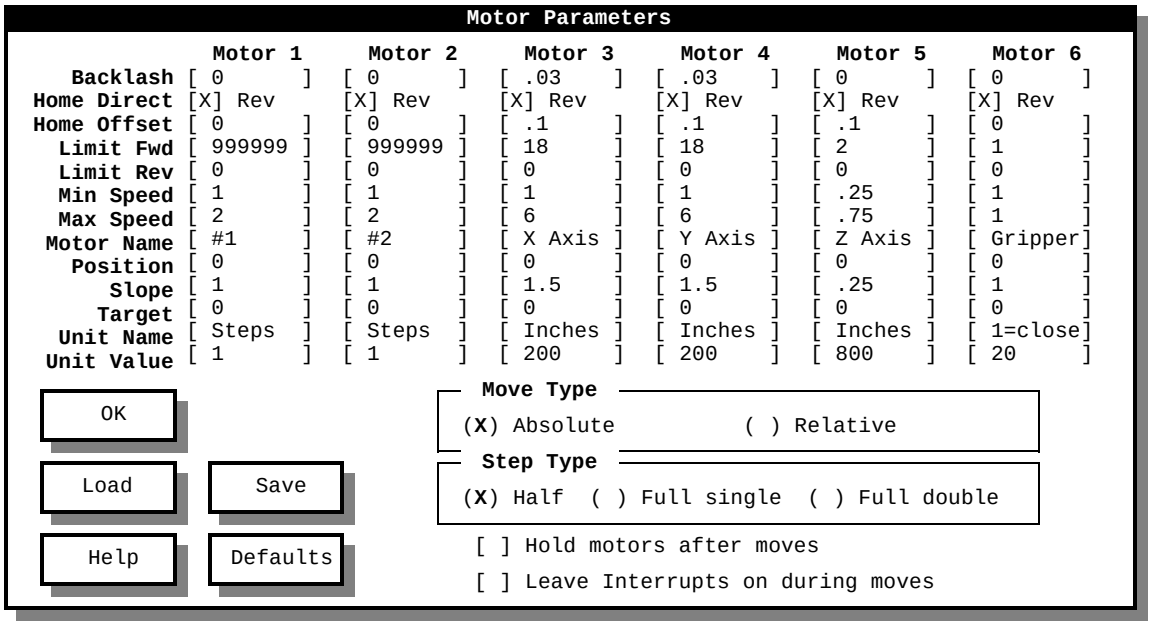
The button controls, the STEP TYPE check boxes, the MOVE TYPE check boxes, the hold check box and the interrupts check boxes can be activated by pressing the ALT key while pressing the highlighted access key for the desired control. For example, to activate the SAVE button, press ALT-S.

To change a parameter, simply click with the mouse on the desired box and type the new value. Use the delete and backspace key when needed. Do not use commas when entering numeric values. You can also use the tab key to select the desired box if a mouse is not available.

When using the SAVE and LOAD buttons, either use the mouse or the tab key to navigate through the controls to select the file, directory and drive.

Pressing the DEFAULT button will set all of the parameters to their default conditions.

Use the HELP button to find information on each parameter.



The Program Editor Screen

Activate the program editor screen by clicking the EDIT button, by pressing ALT-E, or simply E. The main screen and the editor screen can be viewed together if in high-resolution screen mode. Pressing the ESCAPE key will return control back to the main screen.

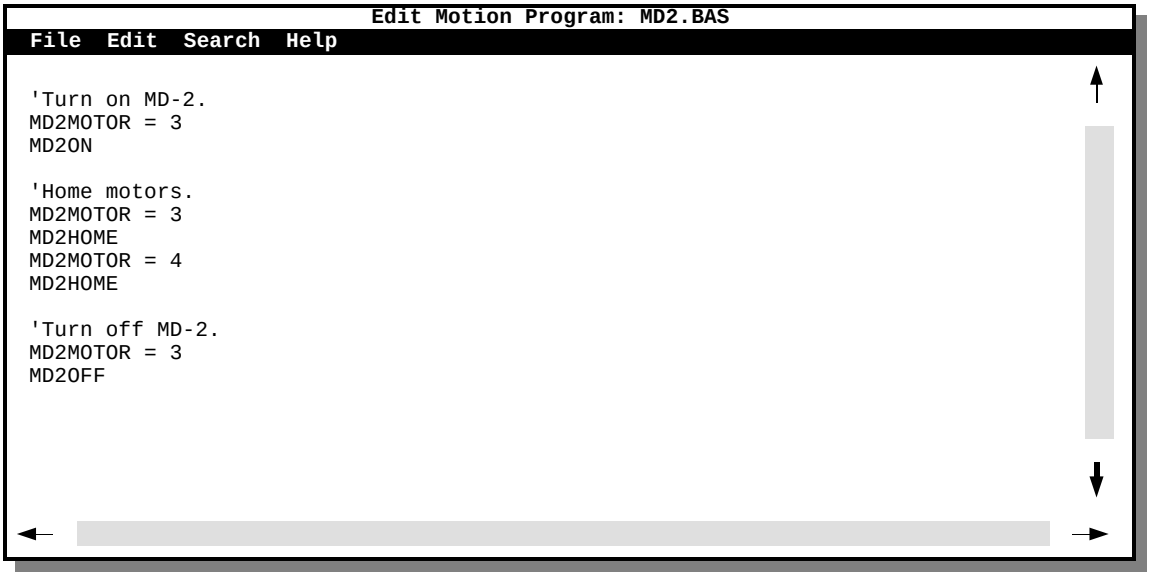
The program editor screen has a menu bar across the top like the main screen. Menu items are selected by pressing ALT along with the highlighted access key or by clicking with the left mouse button. The FILE menu contains common editor commands:

NEW	Erase the current program.
OPEN	Erase the current program and load a program from disk.
SAVE	Save the current program to disk under its existing name.
SAVE AS	Save the current program to disk under a new name.
MERGE	Load a program from disk and place at the end of the current program.
PRINT	Print the current program to a printer.
EXIT	Return to the main screen.

The search menu contains commands to find specific text along with a search-and-replace function. The HELP menu contains the same items as the help menu on the main screen.

Using the editor is just like using a common word processing program. Simply type onto the screen, use the arrow keys and PAGE-UP, PAGE-DOWN keys to navigate. Holding the shift key down while moving the cursor will highlight text allowing you to delete or copy blocks. CTRL-HOME will move the cursor to the beginning of the file and CTRL-END will move to the end. Use the BACKSPACE key to delete text before the cursor and use the DELETE key to delete text after the cursor. The maximum program size is 32,000 characters.

See the section describing motion programs for detailed information on editing programs.



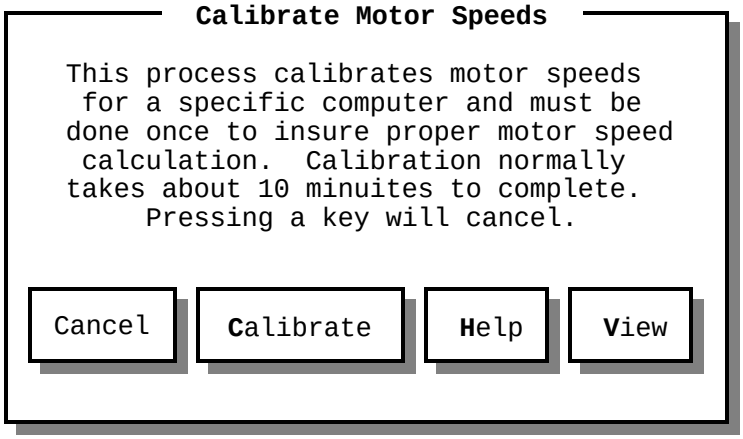
Ports and Motor Numbers

The MD-2 system connects to the parallel printer port of any IBM style personal computer using a standard printer cable. There can be as many as 3 parallel printer ports on a single computer. Since each port can be attached to an MD-2, a total of 6 motors can be controlled with a single computer. Each port has its own unique address. The possible addresses are 3BC, 378 and 278. When adding a new port to your computer, make sure that no two ports have the same address. The MD-2 software refers to the motors connected to port 3BC as motors 1 and 2, port 378 as motors 3 and 4 and port 278 as motors 5 and 6. Your computer may have only one or two ports. Since the motor numbers are determined by which port they are connected to, your system may have motors 3 and 4 or 5 and 6 but not 1 and 2. You may wish to keep your 3BC port connected to your printer so it can be referred to as LPT1. Parallel printer port cards are very inexpensive and are normally available at computer stores or by mail-order. There are usually jumpers on these boards that will select the desired port address. One way to find which port your MD-2 system is connected to is to select INPUTS from the menu and press one of the home switches. The switch status will change from false to true when the switch is pressed indicating which port the MD-2 is connected to.

Calibration

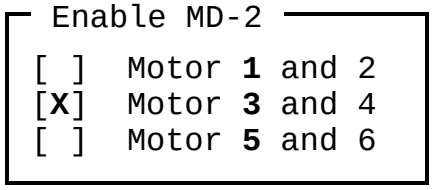
Since there are a wide variety of computers having differing speeds and processing capabilities, it is necessary for the MD-2 program to be calibrated so that motor speeds can be calculated properly. The calibration process does a series of tests and saves the information to a disk file named MD2.CAL. When the MD-2 program is started, this file is loaded automatically.

To calibrate, select CALIBRATION from the OPTIONS menu. This process takes several minutes and is only required once on a computer. Motors can not be moved without a calibration file present. It is important to perform calibration on the computer that will be running the program. Using calibration files created on a different computer may result in inaccurate motor speeds. Many computers have a turbo switch which changes the computer's speed. Perform calibration with the turbo switch in the position that it will be in when the program is being used. Laptop computers sometimes change their speed automatically to conserve power. Disable this feature when running the MD-2 program. also unload any TSR (terminate and stay resident) programs when using the MD-2 program since they steal processing time from the computer.



Enable/Disable MD-2

Before motors can be moved, the MD-2 system must be enabled. Activate the ENABLE check box before moving motors or controlling output signals. While the main screen is active, press ALT-1, ALT-3 or ALT-5 to enable or disable the desired MD-2 system. As a short cut you may press simply 1, 3 or 5 unless the cursor is placed in a DISTANCE or POSITION text box where those numbers are used for input. Disabling an MD-2 will also turn OFF both of the output signals associated with that MD-2.



In a motion control program, enable an MD-2 using the MD2ON command, disable using MD2OFF. At the beginning of a program, enable the MD-2 using an MD2ON command, perform the desired motion, then, at the end of the program, disable the MD-2 using an MD2OFF command. It is necessary to set the motor number before using these commands.

The following example enables (turns ON) the MD-2 for motors 3 and 4.

```
MD2MOTOR = 34 : MD2ON    'Enable MD-2.
```

The following example disables (turns OFF) the MD-2 for motors 5 and 6.

```
MD2MOTOR = 56 : MD2OFF        'Disable the MD-2.
```

While in TEACH mode, this program code is generated automatically when the enable/disable check box is activated.

Holding Motors

When a stepper motor is energized with a step pattern that is not changing, it produces holding torque which will cause the motor's shaft to resist movement. While the motor is in this holding mode, heat is generated. Some applications require that the motor's shaft resist motion between moves and others do not. If holding torque is not needed, disable the MD-2 which will de-energize both motors and reduce motor heat. If holding torque is needed, keep the MD-2 enabled. You may also set the HOLD parameter found on the parameter screen which will determine the holding status of the motors after the next move. The motors and the MD-2 drive electronics are designed to withstand the heat generated.

In a motion program, keep the motors energized by setting the MD2HOLD parameter to TRUE (-1). Set to FALSE to de-energize them. The motors will respond to this setting after the next move.

```
MD2HOLD = -1   'Keep motors energized after next move.
```

Standby Mode

If only partial holding torque is needed, it is possible to place the MD-2 system into standby mode. Standby mode is only available on the MD-2a, ah, b and c models but not the basic MD-2 model. Standby mode reduces the motor current by half which greatly reduces heat buildup while maintaining some holding torque. Trying to move a motor while standby mode is enabled may result in erratic behavior since the torque has been reduced as a result of lower current. Standby mode affects both motors connected to the MD-2 system.

In a motion control program, enable standby mode using the MD2STANDBYON command, disable using MD2STANDBYOFF.

The following example enables (turns ON) standby mode for motors 3 and 4.

```
'Standby mode on.
MD2MOTOR = 34 : MD2STANDBYON
```

The following example disables (turns OFF) standby mode for motors 5 and 6.

```
'Standby mode off.
MD2MOTOR = 56 : MD2STANDBYOFF
```

This program code is generated automatically when the STANDBY MODE item from the OPTIONS menu is activated while in TEACH mode.

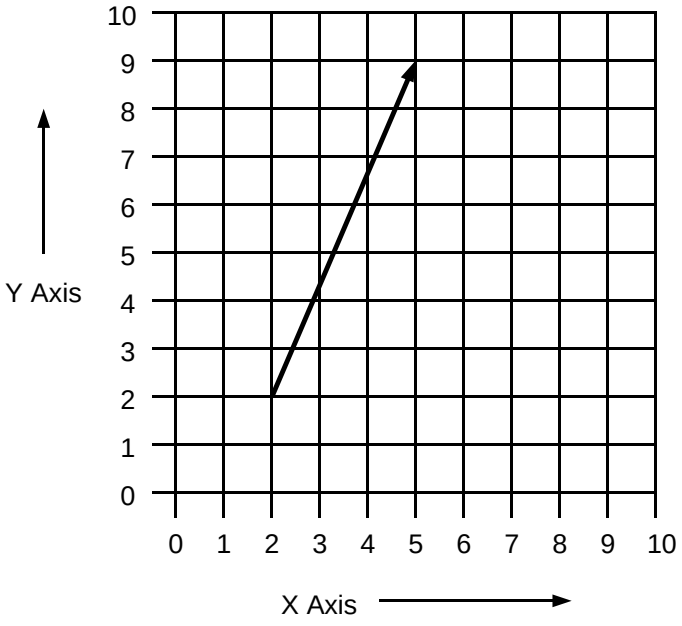
Absolute and Relative Moves

When using line moves or circle moves, the operator can choose between relative or absolute mode. This is set using the MOVE TYPE parameter. In relative moves, the motor will move according to the number of units in the TARGET parameter. Positive numbers are forward and negative numbers are reverse. In absolute moves, the motor will move to the absolute position indicated by the TARGET parameter. Target values are in units with home being the reference. Positive numbers indicate positions forward from home and negative numbers are reverse from home. In absolute moves, the MD-2 will take into account the motor's current position and determines the direction and distance needed to move to the desired position. Both linear and circular moves are affected by the MOVE TYPE parameter. In circular moves, The X and Y center points are either relative to the current position or absolute positions depending on the MOVE TYPE parameter.

The following examples show the difference between absolute and relative modes. Both examples show dual motor, line moves in the form of an XY grid which simulates an XY positioning table. In the absolute example, the target positions are 5 for the X axis and 9 for the Y axis. The software determines the direction and distance each motor must move in order to reach these target positions. In the relative example, the target for the X axis is -3 which causes that motor to move 3 units reverse, the target for the Y axis is 6 which causes that motor to move 6 units forward.

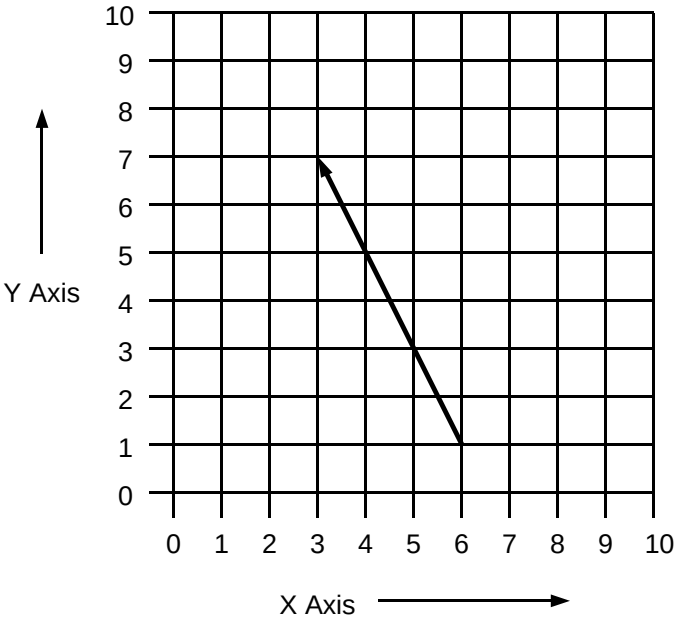
Absolute move example

X axis target = +5
Y axis target = +9



Relative move example

X axis target = -3
Y axis target = +6



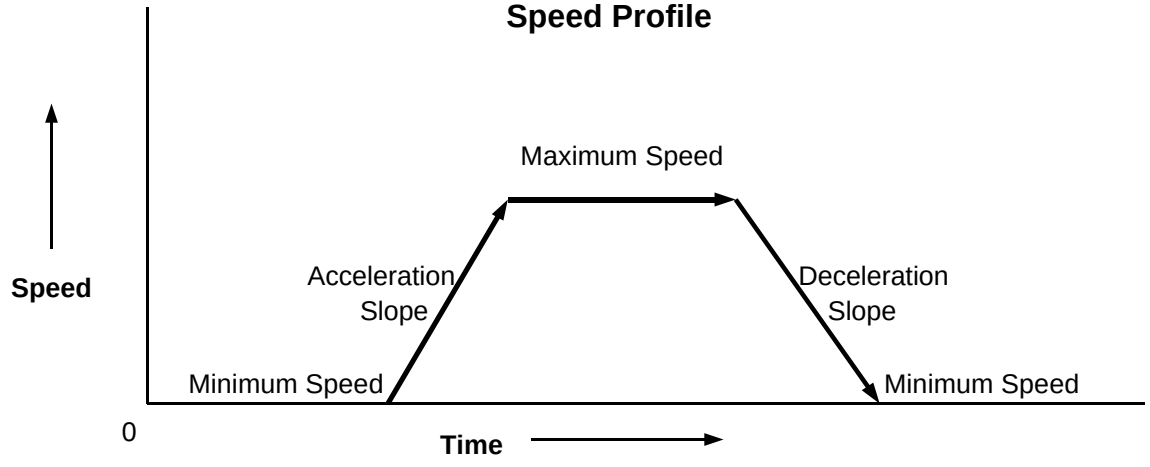
Set the MOVE TYPE parameter in a motion program to an "A" for absolute or an "R" for relative.

```
MD2MOVETYPE = "A"   'Absolute mode.
```

Speeds and Ramping

Motor speeds are determined by the frequency of the pulses sent to the MD-2 system by the computer and are programmable using the MINIMUM SPEED, MAXIMUM SPEED and SLOPE parameters. The minimum speed and maximum speed values are in units per second. Calibration must be done on the computer first to insure proper speed calculation. The minimum and maximum speed ranges are determined by the processing speed of the computer. The VIEW button on the CALIBRATE dialog box displays this information. Fast computers are capable of sending faster step pulses to the motors. If pulses are sent to the motor too fast, the result will be erratic motion or simply vibration. This does not harm the motor or MD-2 system. The torque of a stepper motor decreases as the speed increases which may cause the motor to move a payload at slow speeds but miss steps at faster speeds. Homing the motor can always be used to re-synchronize the system if this occurs. Experimentation will be necessary to determine the optimum speed parameters for a given motor, mechanical system and payload.

Ramping means accelerating the motor from a slow (minimum) speed to a fast (maximum) speed gradually then decelerating back to the slow speed when the motion is done. Ramping greatly increases the speed at which a motor can move a payload and greatly reduces the amount of vibration and jerk in a system. Three parameters are used to accomplish ramping: MINIMUM SPEED, MAXIMUM SPEED and SLOPE. The MINIMUM SPEED is the speed at which the motor begins to move at and ultimately ends at. The MAXIMUM SPEED parameter is the speed at which the motor moves after ramping up (accelerating) and before ramping down (decelerating). SLOPE is the rate at which the speed changes given in number of units. If the SLOPE parameter is set to 100 then it will take 100 units for the motor to go from the MINIMUM SPEED to the MAXIMUM SPEED during acceleration and also 100 units for the motor to go from the MAXIMUM SPEED back down to the MINIMUM SPEED when done. Set the MINIMUM SPEED to a value that the motor can always start and stop at without missing steps or overshooting. Set the SLOPE value to a number that allows the motor to ramp quickly without missing steps. Set the MAXIMUM SPEED to a speed faster than the MINIMUM SPEED parameter but one that does not cause lost steps. Experimentation will provide the best values for these parameters resulting in smooth motion and high speeds. Setting the SLOPE parameter to 0 will cause the motor to move at a constant speed determined by the MAXIMUM SPEED parameter without ramping. Homing the motor only uses MINIMUM SPEED, no ramping is performed.



Backlash Compensation

Most mechanical systems include gears, chains, belts or other devices that contain looseness known as backlash. This results from small spaces between gear teeth, belt stretch, and other imperfections. Backlash is not a problem as long as the motor moves in the same direction. As soon as the motor changes direction, backlash results in a positioning error and the motor must move an additional distance to take up the looseness. This backlash could be eliminated as a source of positioning error by making sure that the desired position is always approached from the same direction. This may require overshooting the desired position and returning to it from the opposite direction. Another way to solve the backlash problem is to compensate for it in software by adding steps whenever the motor changes direction. The BACKLASH parameter is used to do just that. Set this parameter to the number of units of backlash in the mechanical system. Every time the motor changes direction, the backlash will be added to the desired distance. This will result in greatly improved bi-directional repeatability which means that the position will be accurately acquired no matter which direction it is approached from.

As an example, to experience the backlash in a linear positioning table, move the motor in the forward direction for a short distance to take up all of the backlash, then move the motor in the opposite direction one step at a time until the top plate of the positioning table begins to move. The number of steps needed before movement was seen is the backlash. Convert this value to units if needed and enter it as the backlash parameter.

In a motion control program, set the backlash as the following example describes. Place the motor number (1-6) inside of the parenthesis.

```
MD2BACKLASH(3) = .03  'Set backlash for motor 3 to .03 inches.
```

Step Types

Step motors operate by energizing their windings (phases) in certain sequences called phase patterns. Changing from one pattern to another causes the motor to move one step. There are 3 different types of phase pattern modes. In half step mode, the motor alternates between one and two motor phases being energized which results in steps that are half the size of typical full steps. Half steps are usually .9 degrees but differ depending on the motor. Half step mode is the most common since it has twice the resolution of full step modes which reduces vibration. When in half step mode, every other step is stronger than the previous one. This weak/strong pattern may cause some slight positioning error but the error does not accumulate. There are two types of full step modes. Double-phase full step mode always energizes two motor phases at a time which results in more torque and motor heat. Single-phase full step mode always energizes one motor phase at a time which results in less torque and less motor heat. Both full step modes normally result in steps that are 1.8 degrees. The MD-2 program and the level 2 subroutine library allows you to change the step type used. In most applications half step mode should be used to minimize vibration and increase resolution. Step type affects all motors.

The following charts show the various patterns used to control the 4 motor windings (phases). A zero indicates that a winding is turned ON (energized).

Half Step Phase Pattern:

#	4	3	2	1
1	1	1	1	0
2	1	1	0	0
3	1	1	0	1
4	1	0	0	1
5	1	0	1	1
6	0	0	1	1
7	0	1	1	1
8	0	1	1	0

Full-Double Step Phase Pattern:

#	4	3	2	1
1	1	1	0	0
2	1	0	0	1
3	0	0	1	1
4	0	1	1	0

Full-Single Step Phase Pattern:

#	4	3	2	1
1	1	1	1	0
2	1	1	0	1
3	1	0	1	1
4	0	1	1	1

To set the STEP TYPE in a motion control program, set MD2STEPTYPE to "H" for half step, "D" for double-full step, or "S" for single-full step.

```
MD2STEPTYPE = "H"   'Set all motors to half step mode.
```

Units Conversion

Stepper motors move in small steps, usually .9 degrees. Normally the stepper motors are attached to some device which converts these steps to rotational motion or linear motion. In most cases, the user doesn't care how many motor steps must be given to reach a desired location, but is interested only in the distance traveled. This distance is often feet, inches, millimeters, or degrees. The UNITS and UNIT NAME parameters contain user-definable units and are used to convert from the user's units to steps. The UNITS parameter contains the number of steps in each unit. The UNIT NAME parameter contains the name of the units. For example, if a positioning table has 200 steps per inch, the user would set the UNITS parameter to 200 and set the UNIT NAME parameter to "inches". The operator can now deal with motor positions in terms of units (inches) instead of steps. Many other parameters are set in units such as position, backlash, target, limits, speeds etc. Setting the UNITS parameter to 1 and the UNIT NAME parameter to "steps" will allow the operator to deal with motor steps.

Simplifies Mechanical Changes

Units conversion greatly simplifies program modifications needed when the mechanical system is updated or changed. For example, if you create a program to control a positioning table that has 200 steps per inch and then switch to a table that has 1000 steps per inch, simply change the UNITS parameters and run the program without modification.

Use Units Conversion for Scaling

Units conversion can be used to enlarge or reduce motion sequences. For example, if 2 motors are being used to control an XY positioning table that moves a router to cut square shapes out of plastic, changing the UNITS parameter will result in smaller or larger squares without changing the program.

The following example shows how the units conversion parameters are set in a motion program. Place any valid motor number (1-6) between the parenthesis.

```
MD2UNITS(1) = 200              'Motor 1, 200 steps per inch.
MD2UNITNAME(1) = "Inches"      'Motor 1, Inches.
MD2UNITS(2) = 2400             'Motor 2, 2400 steps per foot.
MD2UNITNAME(2) = "Feet"        'Motor 2, feet.
MD2UNITS(3) = 1                'Motor 3, steps.
MD2UNITNAME(3) = "Steps"       'Motor 3, steps.
```

Soft Limits

It is often necessary to limit the travel of the motor to a defined range of motion. This can prevent motions that may cause damage to the mechanics of the system such as tools or bearings. Both forward and reverse limits can be set in the software to prevent the motor from traveling beyond a certain range. The FORWARD LIMIT and REVERSE LIMIT parameters can be used to accomplish this. The values are given in units. If a move is attempted which would move a motor beyond one of these limits, an error message is given.

You can set the limit parameters in a motion program using the following example as a guideline.

```
MD2LIMITR(3) = 0    'Set motor 3 reverse limit to 0 units.
MD2LIMITF(3) = 18   'Set motor 3 forward limit to 18 units.
MD2LIMITR(4) = 0    'Set motor 3 reverse limit to 0 units.
MD2LIMITF(4) = 2    'Set motor 3 forward limit to 2 units.
```

Interrupts

A computer can receive interrupts from various sources such as modems, mice and keyboards. Each action that is generated from these devices causes the computer to quit what it is doing temporarily and process the device's request. If a motor move is in process during one of these interrupts, the motor's speed may change causing undesirable effects such as lost steps and erratic behavior. You can choose to turn interrupts OFF during moves using the INTERRUPTS parameter. Set the INTERRUPTS parameter ON to allow interrupts to occur and be processed during moves, turning it OFF will prevent interrupts from occurring. A check box on the parameters screen is provided for control of this function.

Set the INTERRUPTS parameter in motion programs like the example below shows.

```
MD2INTERRUPTS = 0   'Interrupts OFF during moves.
MD2INTERRUPTS = -1  'Interrupts ON during moves.
```

Input / Output Port

Some MD-2 systems have an input/output port which can be used to read digital inputs such as sensors and switches and used to control digital outputs such as relays, which can in-turn, control lamps, motors and tools. The 14-pin I/O port can be found on the rear panel of the MD-2 system. There are 3 digital inputs and 2 digital outputs along with +5 volts and +12 volts. See the MD-2 manual for detailed hardware information on the I/O port.

Control of the input/output port can be gained through the I/O menu item. A dialog box exists for inputs and one for outputs. On the INPUTS dialog box, each of the 3 input signals along with the status of the home switches for each MD-2 system is displayed in real time. On the OUTPUTS dialog box, a button is associated with the turning ON and turning OFF of each output signal on each MD-2 system. The MD-2 system must be enabled before an output can be activated. Disabling the MD-2 will deactivate both outputs associated with that particular MD-2 system.

When in teach mode, activating outputs with the buttons on the OUTPUTS dialog box will cause program code to be entered into the editor. Viewing inputs does not create code. To read inputs in a motion program, it is necessary to write a custom program using the level 2 subroutine libraries.

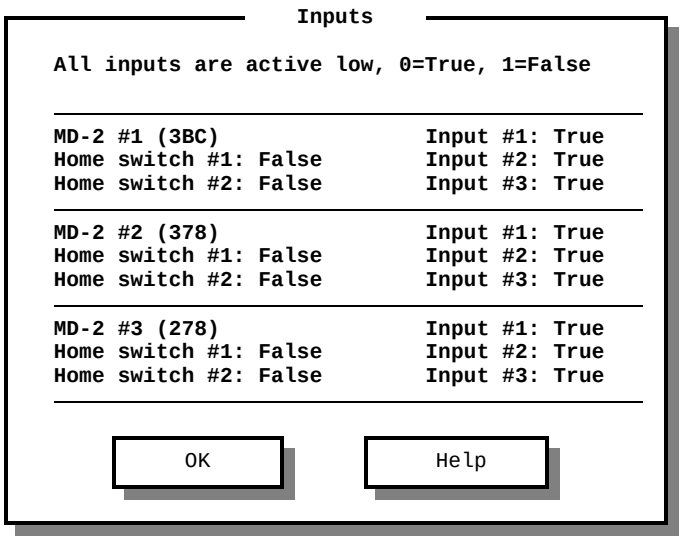
You can control output bits in a motion control program as the following example shows.

```
MD2OUTPUTCODE = 211      'MD-2 #2, Output #1, Signal = ON.
MD2OUTPUTS

MD2OUTPUTCODE = 320      'MD-2 #3, Output #2, Signal = OFF.
MD2OUTPUTS
```

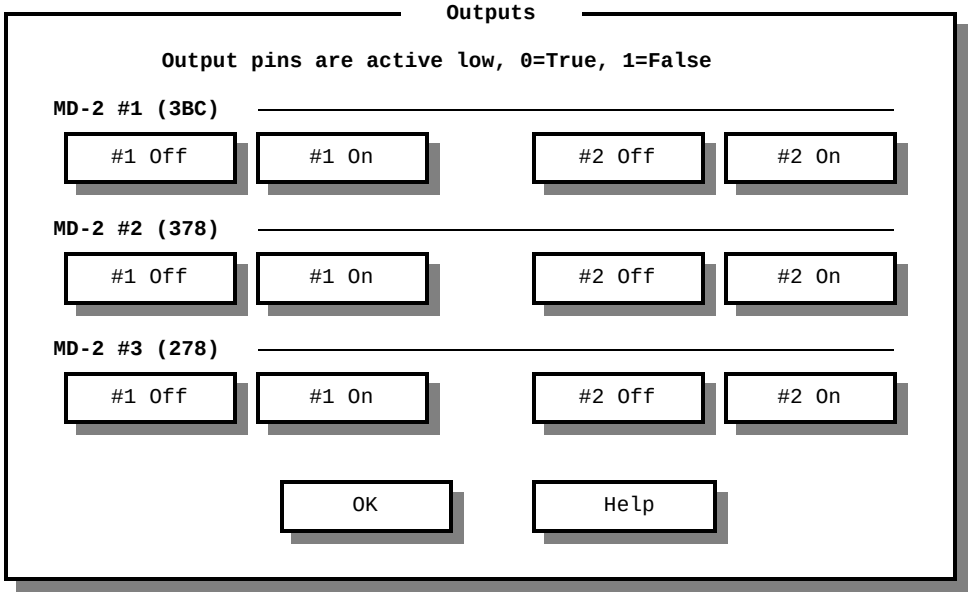
The following chart shows the output codes needed to control output signals in a motion control program. Signals are active low meaning that OFF indicates a signal is raised to 5 volts and ON indicates a signal is lowered to 0 volts (ground).

OUTPUTCODE	MD-2 PORT	OUTPUT	SIGNAL
110	#1 3BC	#1	OFF
111	#1 3BC	#1	ON
120	#1 3BC	#2	OFF
121	#1 3BC	#2	ON
210	#2 378	#1	OFF
211	#2 378	#1	ON
220	#2 378	#2	OFF
221	#2 378	#2	ON
310	#3 278	#1	OFF
311	#3 278	#1	ON
320	#3 278	#2	OFF
321	#3 278	#2	ON

Inputs



Outputs



Quick Moves

Quick moves provide a way to move each motor with a single keystroke or mouse click. The 12 function keys are set up to move each of the 6 motors either forward or reverse. Clicking the button with the mouse in the quick-move box can also be used. The MD-2 must be enabled before any move can take place.

F1	Motor #1, Reverse
F2	Motor #1, Forward
F3	Motor #2, Reverse
F4	Motor #2, Forward
F5	Motor #3, Reverse
F6	Motor #3, Forward
F7	Motor #4, Reverse
F8	Motor #4, Forward
F9	Motor #5, Reverse
F10	Motor #5, Forward
F11	Motor #6, Reverse
F12	Motor #6, Forward

When a function key is pressed, the selected motor will move in the desired direction. The distance that the motor moves is determined by the value placed in the DISTANCE box. This distance is in units. The unit name is displayed next to the DISTANCE box.

The operator can place the system in single-step mode by selecting the item under the OPTIONS menu or by simply pressing D. The values in the DISTANCE boxes will be changed to reflect the distance that a single step will travel. Pressing D again will switch the values back to their original values. This feature allows you move to a position then make minor adjustments. For example, if an XY table has 200 steps per inch, and the DISTANCE value is set to 1, each time a function key is pressed the motor will move 1 inch (200 steps). After activating single-step mode, the DISTANCE value will be changed to .005, and pressing a function key will move the motor .005 inches (1 step).

When teach mode is active, pressing a function key to perform a quick-move will cause a relative move program line to be entered into the program editor.

The POSITION box will be updated to the correct position after each quick-move.

Quick Moves

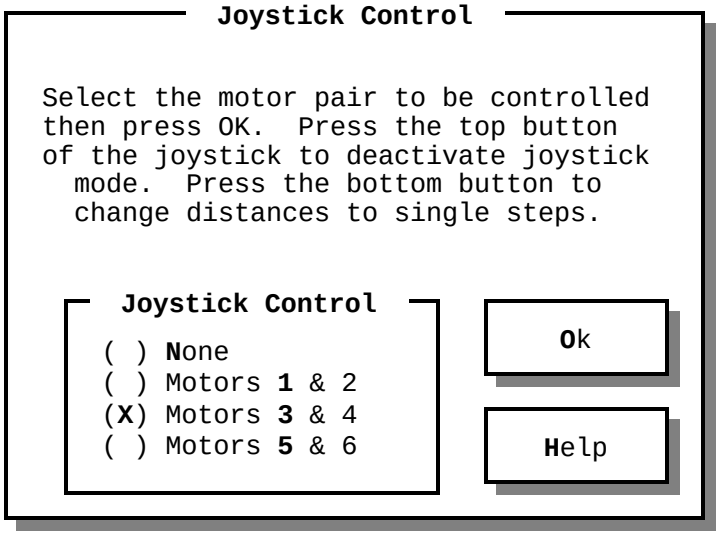
#	Motor	Home	Reverse	Forward	Distance	Units	Position
1	X-Axis	<H1>	<-F1-->	<-F2-->	[1]	Inches	[5]
2	Y-Axis	<H2>	<-F3-->	<-F4-->	[2.5]	Inches	[7.5]
3	Z-Axis	<H3>	<-F5-->	<-F6-->	[.25]	Inches	[.75]
4	Gripper	<H4>	<-F7-->	<-F8-->	[1]	1=close	[0]
5	No Port	<H5>	<-F9-->	<-F10->	[1]	Steps	[0]
6	No Port	<H6>	<-F11->	<-F12->	[1]	Steps	[0]

Joystick Moves

A joystick connected to the game adapter on your computer can be used to simulate quick-moves. Select the JOYSTICK item from the OPTIONS menu or simply press J. A dialog box will be displayed which allows you to select which motor pair to be controlled with the joystick. Pressing J again will turn OFF joystick mode. Each time the joystick is moved to the left, the first motor will move reverse just like the function key has been pressed. Moving the joystick to the right moves the first motor forward. Moving the joystick up moves the second motor forward and moving it down move reverse.

Pressing the top button on the joystick is the same as pressing D which changes the DISTANCE values to single-steps. Press the bottom button on the joystick to turn OFF joystick mode. Buttons on joysticks are not standard and your joystick may respond differently.

When teach mode is active, moving the motors with the joystick will cause a relative move program line to be entered into the program editor just like quick-move that was activated with a function key.

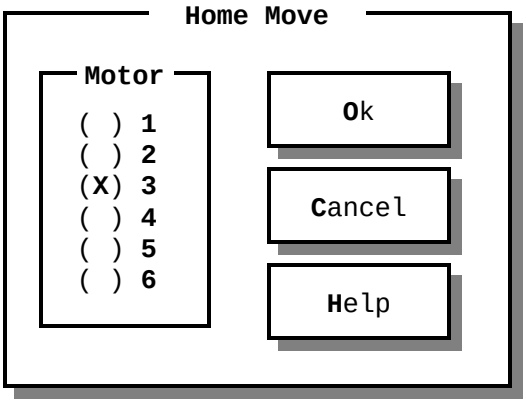


Home Moves

Home moves are used to move the selected motor to the home position using the home switch. Only one motor can be moved home at a time. Each motor on the MD-2 system has a home switch associated with it. At the beginning of a program, the software does not accurately know the positions of the motors. Homing causes the motor to seek the home switch to establish the home position (position zero). All moves are relative to this home position. The programmer or operator can, at any time, home the motors to insure accurate positioning. This is not necessary in most situations where the home function is only used at the beginning of a program once. Homing moves the motor reverse (clockwise as viewed from the front of the motor) until the home switch is activated, then forward until the switch is deactivated. This sequence has the effect of preloading the mechanical system in the forward direction which will increase the accuracy of systems which have backlash or belt stretch. A new home position can be established by homing the motor normally, moving to the desired location, then setting the POSITION parameter to zero. All motions thereafter will be relative to this new home position. This can be used to duplicate motion programs for step and repeat operations without modifications to the fundamental program. The HOMEOFFSET parameter determines how far off of the home switch to move after it is pressed and released. The offset value is in units and can be used to duplicate motion programs at various places on the positioning system. The HOME DIRECTION parameter determines the direction of the home switch which then determines the direction of all other moves. Normally the HOME DIRECTION parameter is set to 0 for reverse. Set it to -1 for forward. This parameter is useful because gear boxes and lead screws often change a motor's direction. During home moves, the motor will move at the MINIMUM SPEED, ramping is not performed.

To activate a home move, click the button in the quick-move box or press H. When H is pressed, a dialog box will allow you to select any of the six motors. As an example, to move motor 3 home, simply press H then 3 at the main screen. This keystroke sequence will become second nature with practice.

While in teach mode, performing a home move will cause the appropriate code to entered into the program editor.



Use the following code to move a motor to the home position in a motion program.

```
MD2MOTOR=3 : MD2HOME      'Move motor 3 home.
```

Line Moves

Motor moves that move one or two motors in a straight line are called line moves. When moving a single motor, the selected motor will begin at the current position and travel according to the TARGET parameter. See the section which discusses absolute and relative moves and the section on speeds and ramping for additional information. When moving two motors simultaneously, the selected motor pair (12, 34 or 56) will begin at their current positions and travel together according to their TARGET parameters. Both motors will begin and end their movement at the same time. This coordinated motion is referred to as linear interpolation since it will create a straight line pattern on an XY positioning table.

To activate a line move, click the LINE button on the main screen, press ALT-L, or simply L. A dialog box will be displayed which allows you to enter the motor or motor pair, move type, and targets. Only a few keystrokes are required to perform a single or dual motor move. For example, to move motor 3 to position 12, press the following keys: L 3 12 (ENTER). To move motors 1 and 2 to positions 6.5 and 2.5 respectively, press: L A 6.5 (TAB) 2.5 (ENTER). In this case, A selected motors 1 and 2, while the TAB key moved from one target box to the next. To select motor pair 3 and 4, press B. To select motor pair 5 and 6, press C. Practice these keystrokes and they will become second nature.

While in teach mode, performing line moves will write code in the program editor screen automatically.

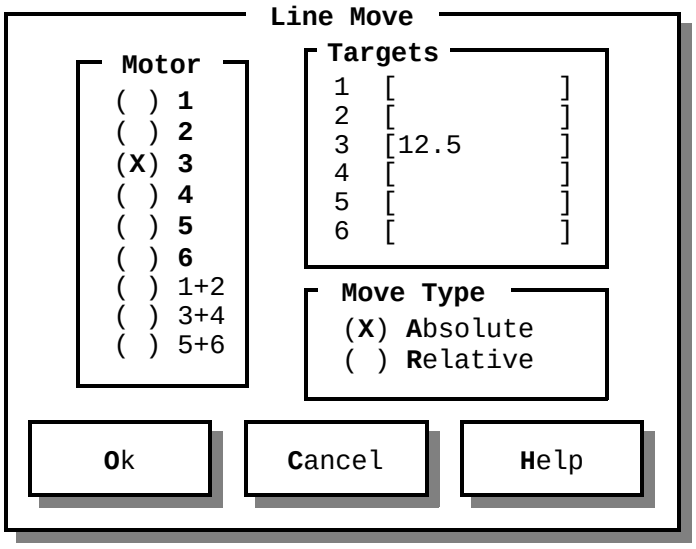
To perform a single motor line move in a motion control program, use the following example.

```
'Move motor 3 to 14.5.
MD2MOTOR=3 : MD2TARGET(3)=14.5 : MD2MOVE
```

To perform a dual motor line move use the following example.

```
'MOVE MOTOR 3 TO 14.5 AND MOTOR 4 TO 12.
MD2MOTOR=34 : MD2TARGET(3)=14.5 : MD2TARGET(4)=12 : MD2MOVE
```

These examples assume that all other motor parameters are set as desired.



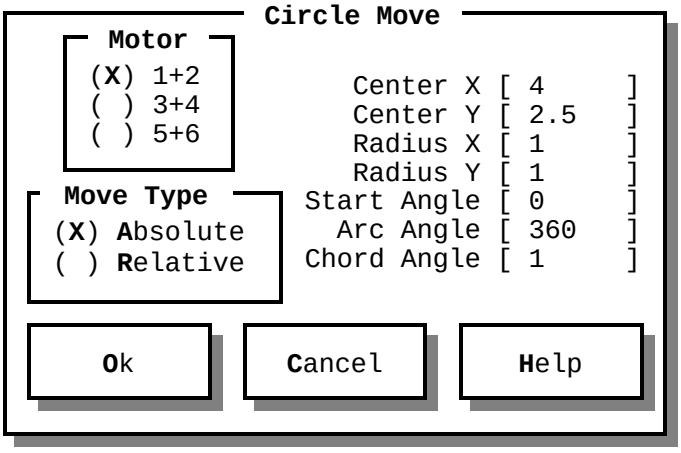
Circle Moves

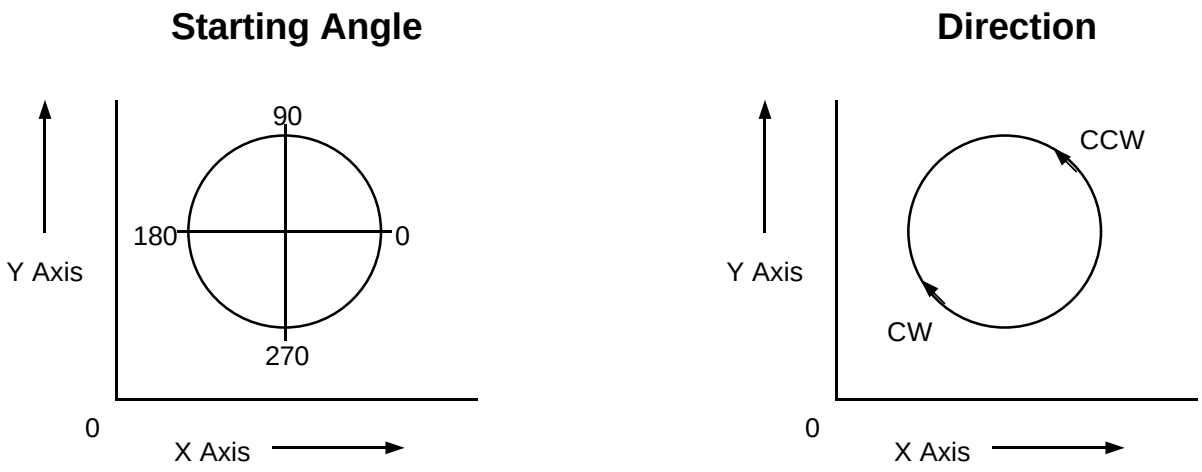
Circle and arc moves provide a way to move two motors which are attached to an XY positioning table in a circular pattern. The two motors must be a valid pair such as 1 & 2, 3 & 4 or 5 & 6. By changing the parameters associated with circular moves, you can perform arcs (partial circles), ellipses, partial ellipses, and polygons. Circle moves can be either relative or absolute. In relative mode, the X and Y center parameters are relative to the current position. In absolute mode, the X and Y parameters are absolute positions. The circle parameters are:

X & Y motor numbers.
Absolute or relative move type.
X-axis radius in units, Y-axis radius in units.
X-axis center position or distance in units, Y-axis center position or distance in units.
Starting angle in degrees.
Arc angle in degrees.
Chord angle in degrees.
Speed and other motor parameters.

Setting both X and Y radii parameters to the same value will result in circles and setting them to different values will produce ellipses. Each circular motion profile is created by moving in straight lines between points on the circle. The resolution of these points is set using the chord angle parameter. Setting the chord angle to 1 will result in a line every one degree. Setting the chord angle to 10 degrees will result in a 36 sided circle. Setting the chord angle to 45 degrees will produce an octagon, 60 degrees will produce a hexagon. The arc parameter determines how complete the circle is. Set arc to 360 for a complete circle or 90 for a quarter circle. A positive arc angle will produce counter-clockwise motion and negative angles will produce clockwise motion. If an arc angle can not be evenly divided by the selected chord angle, the last line segment will be shortened so that the desired arc angle results.

Ramping is not done with circle moves. The speed of the motors is determined by their minimum speed parameters. Best performance is achieved by setting the minimum speed parameters for both motors to the same value.





While in teach mode, performing a circle move will cause program code to be added to the program editor. All circle parameters are included in the newly generated code.

Use the following examples as guides to incorporate circle moves in motion programs.

This example moves motors 3 and 4 in a complete circle with a radius of 2 units and a center position of 10.5 and 4.1.

```
'Move circle.
MD2CIRCLERADIUSX=2
MD2CIRCLERADIUSY=2
MD2CIRCLECENTERX=10.5
MD2CIRCLECENTERY=4.1
MD2CIRCLESTART=0
MD2CIRCLEARC=360
MD2CIRCLECHORD=1
MD2MOVETYPE="A"
MD2MOTOR=34
MD2CIRCLE
```

This example moves motors 1 and 2 in a 90 degree arc with a radius of 4 units and a center position which is 1 unit forward from the current position in the X axis and 1 unit forward from the current position in the Y axis.

```
'Move circle.
MD2CIRCLERADIUSX=4
MD2CIRCLERADIUSY=4
MD2CIRCLECENTERX=1
MD2CIRCLECENTERY=1
MD2CIRCLESTART=0
MD2CIRCLEARC=90
MD2CIRCLECHORD=1
MD2MOVETYPE="R"
MD2MOTOR=12
MD2CIRCLE
```

Grid Moves

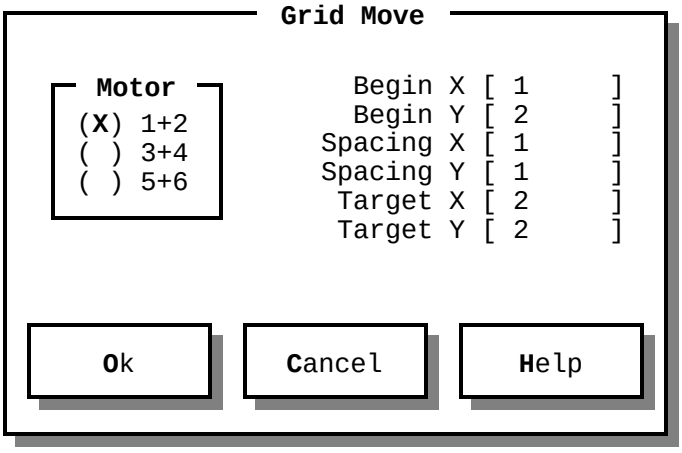
Many motion control applications use XY positioning tables and need to position to a matrix (or grid) of workpieces such as a rack of test tubes. A grid consists of a series of positions on the X axis (columns) and a series of positions on the Y axis (rows). Instead of positioning to a certain cell of the grid by using distances and directions, you can simply use X (column) and Y (row) targets with a grid move. The move type parameter does not apply since only absolute grid coordinates are used. Grid moves require the following parameters:

X & Y motor numbers.
X-axis beginning position of the grid in units.
Y-axis beginning position of the grid in units.
X-axis spacing between cells in units.
Y-axis spacing between cells in units.
Target X-axis grid cell (column).
Target Y-axis grid cell (row).

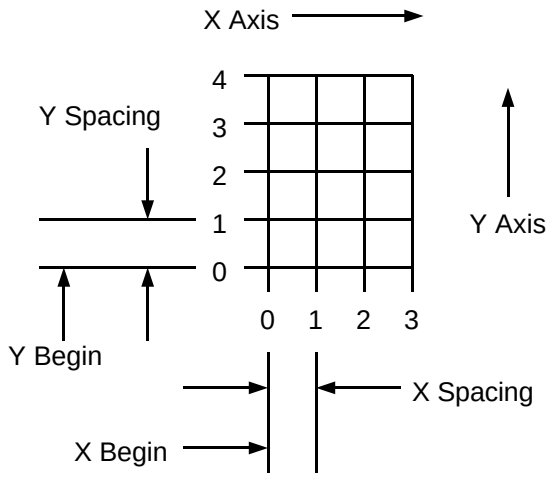
Activate grid moves by clicking the grid button or simply pressing G. Set the parameters and select OK. When in teach mode, performing a grid move will cause code to be added to the program editor.

The following code will perform a grid move in a motion program.

```
'Grid move to column 1, row 2.
MD2MOTOR=34
MD2GRIDBEGINX=0
MD2GRIDBEGINY=0
MD2GRIDSPACEX=2.5
MD2GRIDSPACEY=2.5
MD2GRIDTARGETX=1
MD2GRIDTARGETY=2
MD2GRID
```



Sample Grid



Parameters

Each motor has several parameters which are used to control their behavior during moves. Some parameters apply to individual motors while others apply to all motors. When setting a parameter that applies to a particular motor, the motor number (1 through 6) is placed in parenthesis following the parameter name. This list describes each parameter, what their valid values are, and how to set them in a motion program.

Parameter: Backlash Compensation
Description: Used to increase positioning accuracy on mechanical systems that have gear backlash or belt stretch. The backlash values is added to the motion when the motor's direction changes.
Values: Any valid unit value.
Default: 0.
Example: MD2BACKLASH(1) = .01 'Set backlash for motor 1 to .01 units.

Parameter: Home Direction
Description: Controls direction which the motor must move to reach the home switch. Has the effect of swapping all directions for motor moves .
Values: 0 = reverse, -1 = forward.
Default: 0.
Example: MD2HOMEDIR(3) = -1 'Set home direction to forward for motor 3.

Parameter: Home Offset
Description: Causes the motor to move away from the home switch in the forward direction when moving home. This position is referred to as position 0, home.
Values: Any valid unit value.
Default: 0.
Example: MD2HOMEOFFSET(6) = .1 'Set home offset to .1 units for motor 6.

Parameter: Hold motors after moves.
Description: Controls holding mode of all motors. When set to true, motors are left energized after moves which causes them to resist shaft rotation. When set to false, motors are de-energized after moves which resists heat buildup.
Values: 0=false (off), -1=true (on)
Default: 0.
Example: MD2HOLD = -1 'Turn hold on for all motors.

Parameter: Interrupts left on during moves.
Description: Allows interrupts to be turned off during motor moves to maximize smoothness. Controls interrupts received from com ports, modems and mice.
Values: 0=false (off), -1=true (on)
Default: 0.
Example: MD2INTERRUPTS = 0 'Turn interrupts off.

Parameter: Limit - Forward
Description: Limits motor motion beyond a preset limit in the forward direction. Any move that would exceed this limit will not be performed.
Values: Any valid unit value.
Default: 999999.
Example: MD2LIMITF(2) = 18 'Set forward limit for motor 2 to 18 units.

Parameter: Limit - Reverse
Description: Limits motor motion beyond a preset limit in the reverse direction. Any move that would exceed this limit will not be performed.
Values: Any valid unit value.
Default: -999999.
Example: MD2LIMITR(2) = 0 'Set forward limit for motor 2 to 0 units.

Parameter: Selected motor(s)
Description: Sets the motor (1-6) or motor pair (12, 34 or 56) that are to be acted upon. Line moves can be single or dual motor moves. Circle and grid moves must be dual motor moves. Also used to select MD-2 system for functions such as MD2ON, MD2OFF, MD2STANDBYON/OFF, etc.
Values: 1,2,3,4,5,6,12,34,56.
Default: 1.
Example: MD2MOTOR = 3 'Set motor number to 3.

Parameter: Motor name
Description: Used simply as a description of the motors function and is only used for display.
Values: Any 8-character combination of numbers and letters.
Default: "Motor #1", "Motor #2", etc.
Example: MD2MOTORNAME(4) = "X axis" 'Set motor name for motor 4.

Parameter: Maximum Speed.
Description: Determines the maximum speed that a motor will reach after accelerating.
Values: Given in units per second.
Default: 500.
Example: MD2MAXSPEED(1) = 3.2 'Motor 1 maximum speed = 3.2 units per second.

Parameter: Minimum Speed.
Description: Determines the minimum speed that a motor will begin and end with.
Values: Given in units per second.
Default: 100.
Example: MD2MINSPEED(1) = .5 'Motor 1 minimum speed = .5 units per second.

Parameter: Move Type.
Description: Selects relative or absolute moves. Affects all motors.
Values: "A" = absolute, "R" = relative.
Default: "R".
Example: MD2MOVETYPE = "A" 'Absolute mode.

Parameter: Output Code
Description: Selects the desired output bit to be controlled before using the MD2OUTPUTS function. See section on I/O port for complete description of codes.
Values: 110, 111, 120, 121, 210, 211, 220, 221, 310, 311, 320, 321.
Default: 110.
Example: MD2OUTPUTCODE = 211 'MD-2 #2, Output #1, ON.

Parameter: Position
Description: The current motor position in units. Updated automatically by move functions. Can be set manually to force the system to think it's at a different position.
Values: Any valid unit value.
Default: 0.
Example: MD2POSITION(2) = 12 'Set current position to 12 for motor 2

Parameter: Slope
Description: Determines the rate of acceleration and deceleration. The value sets the distance of motion needed to accelerate from the minimum speed to the maximum speed and to decelerate from the maximum speed to the minimum speed.
Values: Any valid unit value. 0 will cause constant speed using maximum speed value.
Default: 100.
Example: MD2SLOPE(5) = 1.5 'Set slope for motor 5 to 1.5 units.

Parameter: Step Type
Description: Sets the type of step patterns given to the motors. Affects all motors.
Values: "H" = half step, "D" = double-full step, "S" = single-full step.
Default: "H".
Example: MD2STEPTYPE = "H" 'Set step type to half.

Parameter: Target
Description: Determines the distance and direction a motor will move in relative mode and determines the desired target position in absolute moves.
Values: Any valid unit value.
Default: 400.
Example: MD2TARGET(3) = 4.1 'Motor 3 target = 4.1 units.

Parameter: Units value.
Description: Sets the number of motor steps for each user defined unit. Inches, mm and feet are common units.
Values: Positive whole numbers.
Default: 1.
Example: MD2UNITS(3) = 200 'Motor 3, 200 steps per inch.

Parameter: Unit Name.
Description: Describes units. Only used for display purposes.
Values: Any 8-character combination of numbers and letters.
Default: "Steps"
Example: MD2UNITNAME(4) = "Inches" 'Motor 4 = Inches.

Circle Parameters

Parameter: X-Axis Radius
Description: Sets the radius of the X axis motor. Setting different than the Y-axis radius will result in an elliptical pattern.
Values: Any valid unit value.
Default: 1.
Example: MD2CIRCLERADIUSX = 2.2 'Set X-axis radius.

Parameter: Y-Axis Radius
Description: Sets the radius of the Y axis motor. Setting different than the X-axis radius will result in an elliptical pattern.
Values: Any valid unit value.
Default: 1.
Example: MD2CIRCLERADIUSY = 2.2 'Set Y-axis radius.

Parameter: X-Axis Center Position
Description: Sets the center position of the circle on the X axis motor.
Values: Any valid unit value.
Default: 1.
Example: MD2CIRCLECENTERX = 4.5 'Set X-axis center.

Parameter: Y-Axis Center Position
Description: Sets the center position of the circle on the Y axis motor.
Values: Any valid unit value.
Default: 1.
Example: MD2CIRCLECENTERY = 4.5 'Set Y-axis center.

Parameter: Start Angle
Description: Sets the starting angle in degrees.
Values: Integer 0 - 360.
Default: 0.
Example: MD2CIRCLESTART = 180 'Set starting angle to 180 degrees.

Parameter: Arc Angle
Description: Sets the arc angle in degrees. 360 will result in a circle, 180 in 1/2 circle etc.
Values: Integer 0 - 360.
Default: 360.
Example: MD2CIRCLEARC = 90 'Set arc angle to 90 degrees.

Parameter: Chord Angle
Description: Sets the chord angle in degrees. This is the distance between points on the circle that are connected together to create the arc or circle. The maximum resolution is 1 which causes a point to be placed every 1 degree. 45 will create an octagon, 60 will create a hexagon.
Values: Integer 1 - 90.
Default: 1.
Example: MD2CIRCLECHORD = 1 'Set chord angle to 1 degree.

Grid Parameters

Parameter: X-Axis Beginning Position.
Description: Sets the beginning of the grid on the X axis motor.
Values: Any valid unit value.
Default: 1.
Example: MD2GRIDBEGINX = 0 'Set X-axis beginning position.

Parameter: Y-Axis Beginning Position.
Description: Sets the beginning of the grid on the Y axis motor.
Values: Any valid unit value.
Default: 1.
Example: MD2GRIDBEGINY = 0 'Set Y-axis beginning position.

Parameter: X-Axis Spacing.
Description: Sets the spacing between grid points on the X axis motor.
Values: Any valid unit value.
Default: 1.
Example: MD2GRIDSPACEX = 1 'Set X-axis grid spacing.

Parameter: Y-Axis Spacing.
Description: Sets the spacing between grid points on the Y axis motor.
Values: Any valid unit value.
Default: 1.
Example: MD2GRIDSPACEY = 1 'Set Y-axis grid spacing.

Parameter: X-Axis Target.
Description: Sets the desired target grid coordinate for the X axis motor.
Values: Integers.
Default: 1.
Example: MD2GRIDTARGETX = 0 'Set X-axis target position.

Parameter: Y-Axis Target.
Description: Sets the desired target grid coordinate for the Y axis motor.
Values: Integers.
Default: 1.
Example: MD2GRIDTARGETY = 0 'Set Y-axis target position.

Parameter Files

Parameter files are stored on disk and contain all motor parameters such as backlash, speeds, motor positions, circle and grid parameters. Parameter files are normally used to store the beginning parameter values before moves are made or programs are run. Parameter files are text files that can be edited using a text editor that works with unformatted documents. Typographical errors will produce FILE ERRORS when the parameter file is read. It's best to edit and save parameters in the program to insure accurate results.

When the MD-2 program starts, the default parameter file named MD2.PAR located in the current directory is loaded. You can simply modify these parameters and save under the same name or create and save your own under any valid DOS file name. Using the file extension .PAR can help you locate the file but is not mandatory.

While in teach mode, loading a parameter file from the parameter screen creates code in the program editor.

To load a parameter file in a motion control program, use the following example.

```
MD2PARFILE="PROJECT.PAR" : MD2PARLOAD     'Load parameters.
```

Motion Programs

The MD-2 program allows the operator to create and run motion programs that can be used to perform motion control functions for custom applications. Motion programs are similar to programs written in the BASIC language but without features such as FOR/NEXT loops, subroutines, etc. This provides a convenient way to write motion control programs without using the MD-2 subroutine libraries and a language compiler. Most motion programs can be loaded into Quick-Basic or Visual-Basic/DOS along with the MD-2 level 2 subroutine library and run like a normal BASIC program. A program can contain commands to load parameter files, set motor parameters, move motors using home, line, circle or grid moves and commands to turn on and off output signals to control tools. The MD-2 program includes an editor which can be used to create and change programs. Any common DOS or Windows text editor that will create files in plain ASCII text can also be used.

The MD-2 program can handle programs as large as 32,000 characters. If larger programs are needed, use the level 2 subroutine libraries. The program size is then limited by the compiler used.

Several example programs can be found on the distribution disk. Locate them and study them for detailed information on the creation and modification of program files.

Here is an example of a motion program:

```
MD2MOTOR=34                    'TURN ON MD-2 FOR 3 AND 4.
MD2ON
MD2PARFILE="MOTOR.PAR"         'LOAD MOTOR PARAMETERS.
MD2PARLOAD
MD2MOTOR=3                     'MOVE MOTOR 3 HOME.
MD2HOME                        'MOVE MOTOR 3 HOME.
MD2MOTOR=4                     'MOVE MOTOR 4 HOME.
MD2HOME
SLEEP 0                        'WAIT FOR A KEYPRESS.
MD2TARGET(3)=3                 'MOVE MOTOR 3, 3 UNITS.
MD2MOTOR=3
MD2MOVE
MD2OUTPUTCODE=211              'TURN ON OUTPUT # 1.
MD2OUTPUTS
SLEEP 10                       'WAIT 10 SECONDS.
MD2TARGET(4)=1                 'MOVE MOTOR 4, 1 UNIT.
MD2MOTOR=4
MD2MOVE
MD2OUTPUTCODE=210              'TURN OFF OUTPUT # 1.
MD2OUTPUTS
BEEP                           'BEEP.
MD2MOTOR=34                    'TURN OFF MD2.
MD2OFF
END
```

Programs consists of both commands and parameter settings.

Commands

MD2SETUP	Used at the beginning of every program. Not needed when running a program with the MD-2 program.
MD2ON	Turn on (Enables) an MD-2.
MD2OFF	Turn off (Disables) an MD-2.
MD2PARLOAD	Load a parameter file.
MD2HOME	Move a motor home.
MD2MOVE	Move motor(s) in a line.
MD2CIRCLE	Move motors in a circle.
MD2GRID	Move motors to grid coordinates.
MD2OUTPUTS	Turn on or off an output signal.
MD2STANDBYON	Turn on standby mode.
MD2STANDBYOFF	Turn off standby mode.
BEEP	Make a beep sound.
SLEEP	Wait for a number of seconds or wait for a keypress.
END	End of a program.

Parameters

(M) indicates a motor number 1,2,3,4,5 or 6.

MD2BACKLASH(M)	Backlash in units.
MD2HOLD	-1=Hold motors, 0=don't. Applies to all motors.
MD2HOMEOFFSET(M)	Offset from home switch.
MD2HOMEDIR(M)	0=rev,-1=fwd, direction of home switch
MD2INTERRUPTS	-1=leave on, 0=turn off during moves. Applies to all motors.
MD2LIMITF(M)	Forward limit in units.
MD2LIMITR(M)	Reverse limit in units.
MD2MOTOR	Motor number 1-6 or 12, 34, 56 for dual.
MD2MOTORNAME(M)	Name of motor.
MD2MAXSPEED(M)	Maximum accelerated speed.
MD2MINSPEED(M)	Start/stop speed.
MD2MOVETYPE	A=Absolute, R=relative. Applies to all motors.
MD2OUTPUTCODE	Code to control output port signals.
MD2POSITION(M)	Current motor position in units.
MD2SLOPE(M)	Acceleration/deceleration slope in units.
MD2STEPTYPE	H=Half, D=double full, S=Single full. Applies to all motors.
MD2TARGET(M)	Target position or distance and direction.
MD2UNITS(M)	# of steps per unit.
MD2UNITNAME(M)	Name of unit.

Circle Parameters

See the section on circle moves and the section on parameters for detailed information on circle parameters.

MD2CIRCLERADIUSX	X-axis radius in units.
MD2CIRCLERADIUSY	Y-axis radius in units.
MD2CIRCLECENTERX	X-axis center in units.
MD2CIRCLECENTERY	Y-axis center in units.
MD2CIRCLESTART	Starting angle in degrees.
MD2CIRCLEARC	Arc angle in degrees.
MD2CIRCLECHORD	Chord angle in degrees.

Grid Parameters

See the section on grid moves and the section on parameters for detailed information on grid parameters.

MD2GRIDBEGINX	X-axis beginning position in units.
MD2GRIDBEGINY	Y-axis beginning position in units.
MD2GRIDSPACEX	X-axis spacing in units.
MD2GRIDSPACEY	Y-axis spacing in units.
MD2GRIDTARGETX	X-axis target in grid coordinates.
MD2GRIDTARGETY	Y-axis target in grid coordinates.

Editing Programs

The MD-2 program includes a program editor which allows the operator to create, edit, save and load motion control programs. The editor can handle programs as large as 32K. To activate the editor, click the EDIT button with the mouse or simply press E from the main MD-2 screen. To return to the main MD-2 program screen, select EXIT from the FILE menu or simply press ESCAPE.

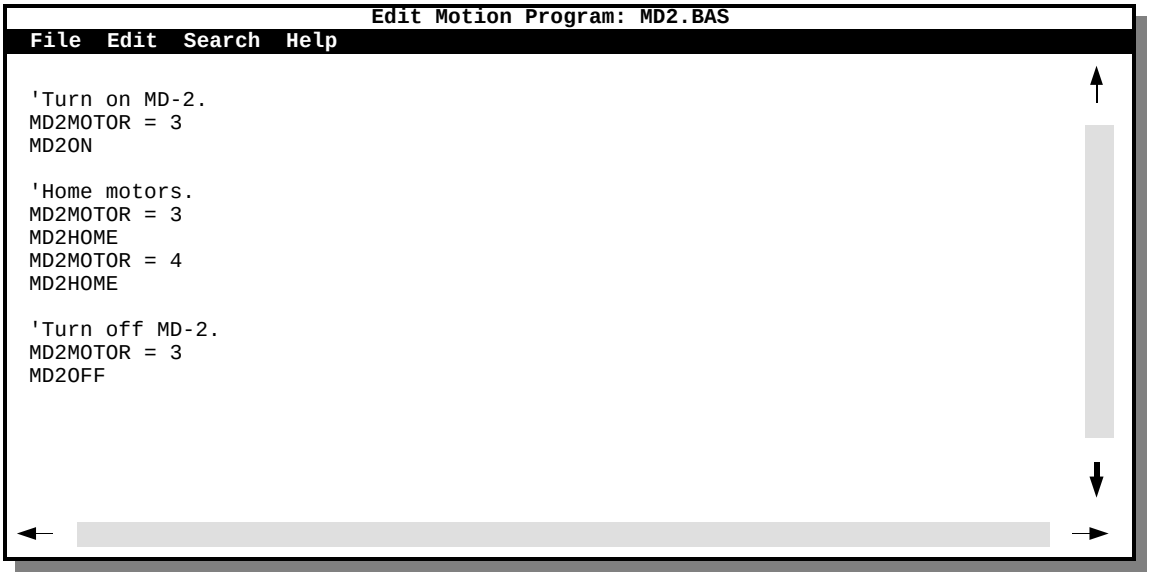
When the MD-2 program is started, the default program named MD2.BAS is loaded from the disk's current directory. You can name your primary project by this name or choose OPEN from the FILE menu on the menu bar and select a different one. The OPEN dialog box only shows files which have a . BAS extension making it easy to find your file. This .BAS extension is not mandatory.

The editor responds to the common editing keys such as arrow keys, PAGE-UP, PAGE-DOWN, BACKSPACE, DELETE, HOME, and END.

A MERGE function is provided in the FILE menu which will open a disk file and append it to the end of the existing program in the editor. This is helpful when creating programs that use pieces of other programs.

You can print the program to a printer by selecting the PRINT item in the FILE menu.

COPY and PASTE commands are available from the EDIT menu and can be used to duplicate or move sections of your program. COPY and other commands such as DELETE operate on selected text. Selecting text is done by placing the cursor at the beginning of the text, then hold the shift key down while using the arrow keys to move down or over. The text will become highlighted to indicate that it is selected. You can then choose COPY or CUT which will act on the selected text. Pressing the DELETE key while text is selected will delete it all. Selecting text can also be done by placing the mouse pointer at the beginning of the text, hold the left mouse button down while dragging to the end of the text.



Teaching Programs

One of the most powerful features of the MD-2 program is the ability to create custom motion programs by teaching the computer. Teaching is accomplished by activating the TEACH button on the main screen, by pressing ALT-T, or simply T. A message on the main screen will indicate that teach mode is active.

While in teach mode the following commands are placed in the program editor automatically:

Enable/Disable MD-2	Grid moves
Standby on/off	Beep
Quick moves (function keys)	Sleep 0 (wait for keypress)
Joystick moves	Sleep # (wait for # of seconds)
Line moves	Output control
Home moves	Load parameters
Circle moves	Comments '

These commands will result in program code being added to the end of the motion program in memory. If an error occurs, the code is entered anyway. You can use teach mode to create complete programs or just parts of programs. The program editor allows you to move the code that is placed at the end of the program to somewhere else in the program. Make sure not to exceed the 32K program length limit.

Manual comments can be added while in teach mode by pressing the ' key and typing a line of text.

A beep sound can be placed in a program by pressing B.

A delay can be created in a program by pressing S, followed by the number of seconds to delay. If this entry is set to 0 then the program will wait for a key press before continuing.

To try teach mode, erase any motion program from memory and follow these instructions:

Press the TEACH button to turn ON teach mode.
Enable the MD-2 system by pressing 1, 3 or 5.
Move the desired motor home by pressing H then the motor #.
Press B to create a beep sound.
Press S then enter 10 to wait for 10 seconds.
Move the motor to a position by pressing one of the function keys.
Press B again to create another beep.
Disable the MD-2 system by pressing 1, 3 or 5.
Press the TEACH button to turn OFF teach mode.

Your custom program is complete. You can see the program created by activating the EDIT button. You can run the program by pressing the RUN button and selecting OK. Your custom motion control program will execute just like you entered it.

Running Programs

To run a motion program, click the RUN command button on the main screen, press ALT-R, or simply R. You will be asked to type the name of a motion program file. If you leave this blank and simply press ENTER, the motion program that is in memory will be run. The program that is in memory can be viewed with the program editor. If you enter a program file name, it must be located in the current directory. The program file commands will be executed in order starting at the beginning of the program. when you give a file name, the program that is in memory is not disturbed.

If the RUN REPEATEDLY check box is selected, the program will automatically restart when finished.

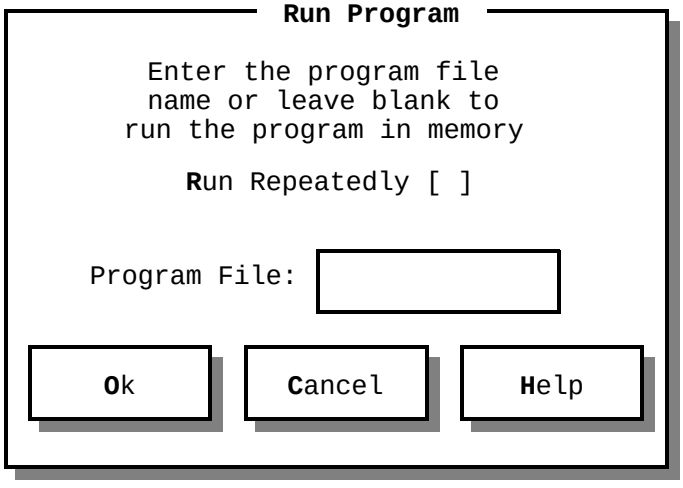
Pressing the space bar will stop the program when the current command is complete. You will be given the option to continue or cancel the program at that time.

While running a motion program, use the power switch on the MD-2 system as an emergency stop switch. This is preferred over pressing the space bar since it responds faster. The MD-2 system(s) and any tooling for your application such as drills or lasers can be attached to a power strip so that one switch can be used to control everything.

You can use the RUN command to perform functions quickly by using very short names or even single letters. For example, you could create a program named GO that turns ON all motors and moves each one to home. To use this program, simply type R GO (ENTER).

If you RUN a program while TEACH mode is active, the entire program will be added to the end of the existing program in the editor. This powerful feature allows the user to create several sub-programs and incorporate them into any program whenever needed.

See the command line section of this manual for information about running programs from the DOS command line and from DOS batch files.



Section 3

Level 1 Subroutines

Section 3

Level 1 Subroutines

Table of Contents

Overview

Different levels of motion control subroutines are available to provide different levels of sophistication, features and performance. The level 1 subroutines offer the most basic features needed to move the motors of the MD-2 system. There are not any fancy features such as ramping, linear interpolation, multi-motor moves, or units conversion. The library includes all the subroutines necessary to control up to 6 motors. All motor parameters are global variables. The programmer simply sets the desired parameters such as speed, direction and distance and calls the subroutine. The library is very easy to use and can be easily connected to other subroutine libraries such as those provided with data acquisition systems. To select a level 1 subroutine library, choose a language from the list below such as QB for Quick-Basic, then print out the documentation file (MD2QB1S.TXT in this case) and review it. You can also print example programs. For example, if you were going to write a program in Visual Basic for DOS, you can get a list of all associated files by typing the following command at the DOS prompt:

```
DIR MD2VD1*.*
```

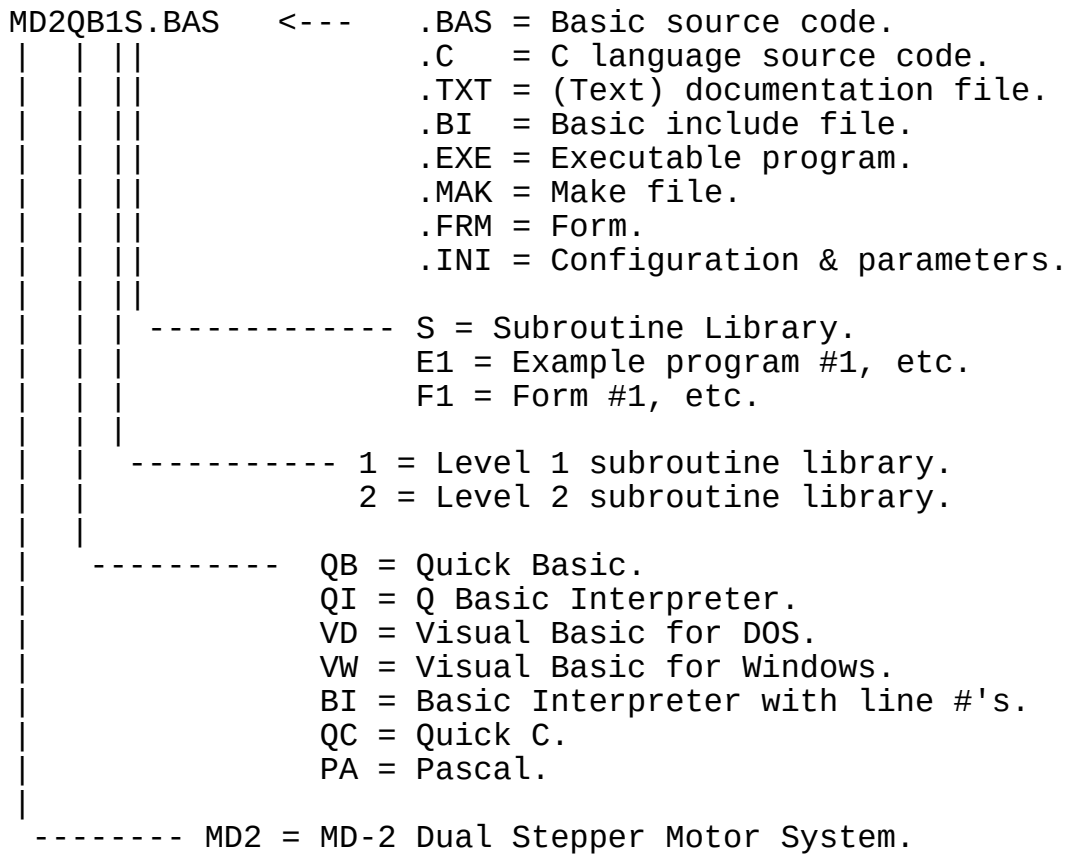
For other languages, simply replace the VD with the 2 letter code for that language. Consult the documentation (.txt) file for the specific language for more detailed information. Consult the MD-2 user's manual for more detailed information on hardware and software.

Features

Complete control of all motion parameters - speed, direction, etc.
Control up to 6 motors on one computer.
Single motor moves.
Control of holding torque.
Half step mode.
Relative moves (step count & direction).
Motor homing using switches

File Names

File names are coded to describe their contents.



Files

Q-Basic Interpreter (Not Quick-Basic)

MD2QI1S.TXT	Documentation file.
MD2QI1S.BAS	Subroutine library.
MD2QI1E1.BAS	Example program #1.
MD2QI1E2.BAS	Example program #2.
MD2QI1E3.BAS	Example program #3.

Basic Interpreter with line numbers (GW Basic)

MD2BI1S.TXT	Documentation file.
MD2BI1S.BAS	Subroutine library.
MD2BI1E1.BAS	Example program #1.
MD2BI1E2.BAS	Example program #2.
MD2BI1E3.BAS	Example program #3.

Quick-Basic (Not Q-Basic)

MD2QB1S.TXT	Documentation file.
MD2QB1S.BAS	Subroutine library.
MD2QB1E1.BAS	Example program #1.
MD2QB1E2.BAS	Example program #2.
MD2QB1E3.BAS	Example program #3.

Visual-Basic for DOS

MD2VD1S.TXT	Documentation file.
MD2VD1S.BAS	Subroutine library.
MD2VD1S.BI	Include file.
MD2VD1E1.BAS	Example program #1.
MD2VD1E1.MAK	Make file for example program #1.
MD2VD1E2.BAS	Example program #2.
MD2VD1E2.MAK	Make file for example program #2.
MD2VD1E3.BAS	Example program #3.
MD2VD1E3.MAK	Make file for example program #3.
MD2VD1E4.FRM	Form for example program #4.
MD2VD1E4.MAK	Make file for example program #4.

Visual-Basic for Windows

MD2VW1S.TXT	Documentation file.
MD2VW1S.BAS	Subroutine library.
MD2VW1E1.FRM	Form for example program #1.
MD2VW1E1.FRX	Graphics for example program #1.
MD2VW1E1.MAK	Make file for example program #1.
MD2VW1E2.FRM	Form for example program #2.
MD2VW1E2.FRX	Graphics for example program #2.
MD2VW1E2.MAK	Make file for example program #2.
MHELP.VBX	DLL for I/O from MicroHelp.

Quick-C (and other C languages)

MD2QC1S.TXT	Documentation file.
MD2QC1S.C	Subroutine library with example main().

Pascal

MD2PA1S.TXT	Documentation file.
MD2PA1S.PAS	Subroutine library.
MD2PA1E1.PAS	Example program #1.

Note: This list may vary as the level 1 subroutine library is updated.

Concepts

The MD-2 System

The MD-2 dual stepper motor system is a complete dual axis motion control system that connects to any IBM style personal computer. The system comes complete with 2 motors, cables, drive electronics, software and documentation. Simply connect the MD-2 to a printer port and load the software. These motors can be used to control a wide range of items including robotic arms, telescopes, cameras and lasers. The operator and programmer have complete control over the speed, direction and other motor parameters. Sophisticated motion control programs can be created to accomplish almost any job.

Parallel Printer Ports

The MD-2 dual stepper motor system connects to the parallel printer port of any IBM style personal computer using a standard printer cable. There can be as many as 3 parallel printer ports on a single computer. Since each port can be attached to an MD-2, a total of 6 motors can be controlled with a single computer. Each port has its own address. The three possible addresses are 3BC, 378 and 278. When adding a new printer port, make sure that no two ports have the same address. The MD-2 software refers to the motors connected to the MD-2 which is connected to port 3BC as 1 & 2, port 378 as 3 & 4, and port 278 as 5 & 6. Your system may only have one or two ports. Since the motor numbers are determined by which port they are connected to, your system may have motors 3 & 4 or 5 & 6 but not 1 & 2. You may wish to keep your 3BC port connected to your printer so it can be referred to as LPT1: or PRN: in which case your MD-2 motors would be 3 & 4 or 5 & 6. Parallel printer port cards are very inexpensive and are normally available at local computer stores or by mail-order.

Homing Motors

Each motor on the MD-2 system has a home switch associated with it. At the beginning of a program, the software does not accurately know the positions of the motors. Homing causes the motor to seek the home switch to establish the home position (position zero). All moves are relative to this home position. The programmer or operator can, at any time, home the motors to insure accurate positioning. This is not necessary in most situations where the home function is only used at the beginning of a program once. The MD2HOME subroutine moves the motor reverse (clockwise as viewed from the front of the motor) until the home switch is activated, then forward until the switch is deactivated. This sequence has the effect of preloading the mechanical system in the forward direction which will increase the accuracy of systems which have backlash or belt stretch. A new home position can be established by homing the motor normally, moving to the desired location, then setting the MD2POSITION(M) parameter to zero. All positions thereafter will be relative to this new home position. This can be used to duplicate motion sequences for step and repeat operations without modifications to the fundamental program. The home switches may also be used for general purpose inputs by reading them with input statements.

Moving Motors

A motor is moved by setting the motor parameters and calling the MD2MOVE subroutine by name. The MD2MOTOR parameter determines which motor is to be moved (1,2,3,4,5 or 6). The MD2TARGET (M) parameter determines how many steps the motor will move and the direction. Positive numbers will move forward and negative numbers will move reverse. The motor will move at a constant speed determined by the MD2SPEED(M) parameter. See the section on motor speeds for more information. The MD2POSITION(M) parameter will be set to the motor's position relative to home in steps. A motor move can be stopped by pressing any key on the keyboard. The home switches are ignored. The motor may be left energized or de-energized depending on the value of the MD2HOLD parameter. If MD2HOLD=0 then the motor will be de-energized, if -1 then power will be left on the motor which results in holding torque and motor heat.

Motor Speeds

Motor speeds are given in delay counter values. This number is the delay time between steps which determines the motor's speed. A small number will move the motor fast, a large number will move the motor slower. Speed values depend greatly on the computer's speed. Fast computers will require higher speed values in order to achieve the same speed as slower computers with smaller numbers. This means that a motion control program may act differently on different computers. The operator needs to have the option of changing the speed values for optimum operation on a particular computer. Finding the correct speed values requires experimentation. The torque of stepper motors decreases as the speed increases. If the speed value is to small (fast), the motor may just vibrate without moving. This condition does not harm the motor or driver. Increase the speed value (slower) until the motor begins to move without missing steps or vibrating, then increase the value another 30% or so to give a margin of error. If a motor ever misses steps, then the MD2POSITION(M) variable will not accurately represent its current position. Select motor speeds that are slow enough to prevent missed steps which causes this condition. Homing the motors can be used to resyncronize the system at any time throughout a program.

Subroutines

MD2SETUP Used at the beginning of a program to initialize motor parameters to their default values. Must be used before any other subroutines.

MD2ON Turns ON an MD-2 driver. Use before attempting to move or home a motor. To use, set MD2MOTOR then GOSUB MD2ON.

MD2OFF Turns OFF an MD-2 driver. Normally used at the end of a program. To use, set MD2MOTOR then GOSUB MD2OFF.

MD2HOME Moves a motor to the home position by watching the home switch. Current position is set to zero. Normally used once before move motors to establish the home position. To use, set MD2MOTOR, MD2SPEED(M) and MD2HOLD, then GOSUB MD2HOME.

MD2MOVE Moves a motor according to motor parameters. To use, set MD2MOTOR, MD2SPEED(M), MD2HOLD and MD2TARGET(M), then GOSUB MD2MOVE.

Parameters

MD2HOLD Integer. -1 (True) = Leaves the motor energized after a move to cause the motor to hold its position. 0 (False) causes the motor to turn OFF after each move which conserves power and reduces heat.

MD2MOTOR Integer. The selected motor. 1, 2, 3, 4, 5, or 6.

MD2SPEED(M) Integer. Delay count between steps. 0=fast, 32766=slow. Actual motor speed depends on computer speed. M=motor #.

MD2POSITION(M) Long Integer. Current motor position for each motor (M) relative to home in steps. Negative values are reverse from home, positive are forward from home.

MD2STATUS String. Completion status. O=motion completed ok, K=keypress stopped motion, B=bad parameter.

MD2TARGET(M) Long Integer. Number of steps to move and direction. Positive numbers are forward and negative numbers are reverse. M=motor #.

Example Program

The following example program shows the use of the subroutine library for Q-Basic (not Quick-Basic). Its use is similar to other languages. Consult the documentation file for each language for specific details on its use. The DIM statements in the subroutine library must be moved to the beginning of the program as required by Q-Basic. It may be necessary to change the MD2SPEED(M), and MD2MOTOR parameters for this program to operate on your computer.

```
'MOVE DIM STATEMENTS FROM SUBROUTINE LIBRARY TO HERE...
DIM . . . .

'SET DEFAULTS AND FINDS PORTS.
GOSUB MD2SETUP

'TURN ON MD-2 FOR MOTORS 3 & 4.
MD2MOTOR = 3                        'MOTOR 4 WOULD ALSO WORK HERE.
GOSUB MD2ON                         'TURN ON MD-2.

'SET MOTOR PARAMETERS THAT ARE DIFFERENT THAN DEFAULTS.
MD2SPEED(3) = 2500                  'SET SPEED FOR MOTOR 3.
MD2SPEED(4) = 1000                  'SET SPEED FOR MOTOR 4.

'MOVE MOTORS 3 & 4 HOME.
MD2MOTOR = 3                        'MOVE 3 HOME.
GOSUB MD2HOME
MD2MOTOR = 4                        'MOVE 4 HOME.
GOSUB MD2HOME

'MOVE MOTOR 3 275 STEPS FORWARD.
MD2MOTOR = 3                        'MOTOR 3.
MD2TARGET(3) = 275                  '275 STEPS FORWARD.
GOSUB MD2MOVE                       'MOVE THE MOTOR.
PRINT "STATUS IS ";MD2STATUS        'DISPLAY STATUS.

'MOVE MOTOR 4 400 STEPS FORWARD.
MD2MOTOR = 4                        'MOTOR 4.
MD2TARGET(4) = 400                  '400 STEPS FORWARD.
GOSUB MD2MOVE
PRINT "STATUS IS ";MD2STATUS        'DISPLAY STATUS.

'DISPLAY MOTOR POSITIONS.
PRINT "MOTOR 3 POSITION IS: ";MD2POSITION(3)
PRINT "MOTOR 4 POSITION IS: ";MD2POSITION(4)

'TURN OFF MD-2 SYSTEM.
MD2MOTOR = 3                        'SELECT MOTOR AT THAT PORT.
GOSUB MD2OFF                        'TURN OFF MD-2.

END

'MD-2 SUBROUTINES START HERE . . .
```

Section 4

Level 2 Subroutines

Section 4

Level 2 Subroutines

Table of Contents

Overview

Different levels of motion control subroutines are available to provide different levels of sophistication, features and performance. The level 2 subroutine library provides a set of functions that the programmer can use to create complex motion control programs. Sophisticated features such as programmable ramping, backlash compensation, linear and circular interpolation, parameter saving and loading, I/O port control and soft limits are included just to name a few. The library includes all the subroutines necessary to control up to 6 motors. All motor parameters are global variables. The programmer simply sets the desired parameters such as speed, direction and distance and calls the subroutine. The subroutines are very easy to use and can be easily connected to other subroutine libraries such as those provided with data acquisition systems. To select a level 2 subroutine library, choose a language from the list below such as QB for Quick-Basic, then print out the documentation file (MD2QB2S.TXT in this case) and review it. You may also print example programs. For example, if you were going to write a program in Visual Basic for DOS, you can get a list of all associated files by typing the following command at the DOS prompt:

```
DIR MD2VD2*.*
```

For other languages, simply replace the VD with the 2 letter code for that language. Consult the documentation (.txt) file for the specific language for more detailed information. Consult the MD-2 user's manual for more detailed information on hardware and software.

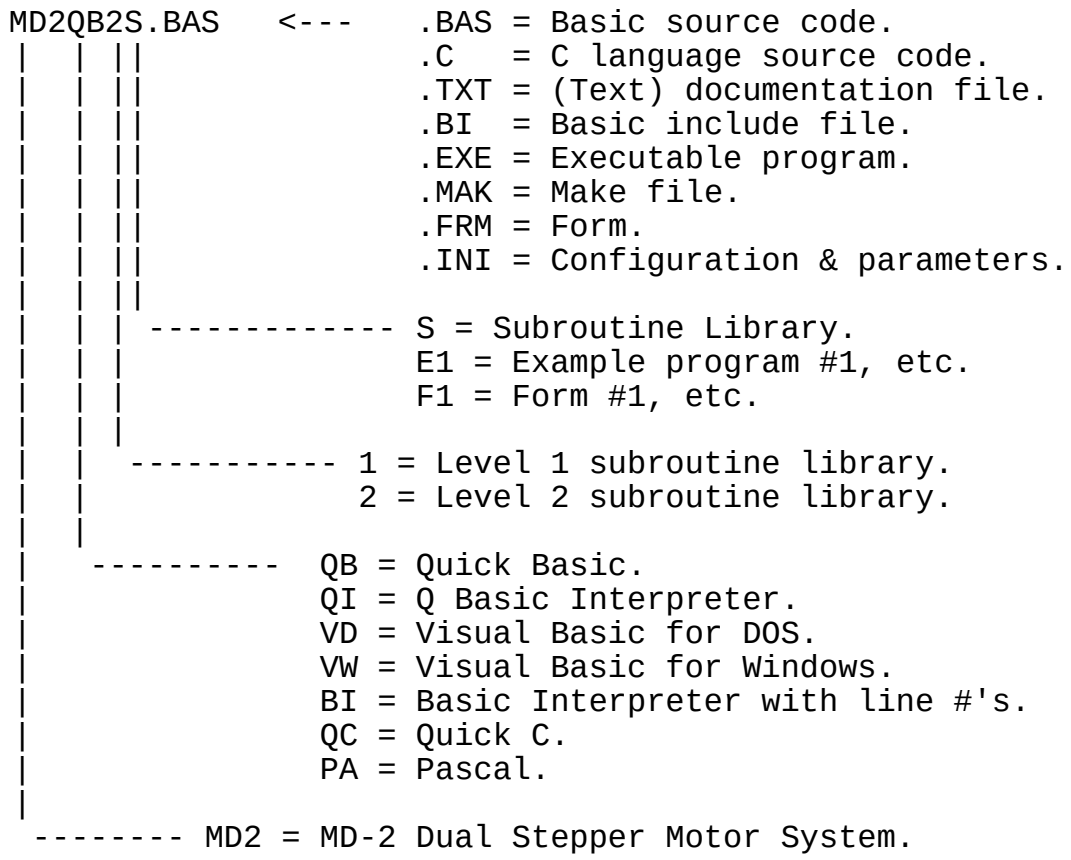
This manual contains example code written in BASIC. If you are writing a custom program in another language such as C, simply change the syntax to match the requirements of that language. Parameter names, subroutine names and variable names are the same for all language libraries

Features

Complete control of all motion parameters - speed, direction, etc.
Control up to 6 motors on one computer
Single and dual motor moves
Linear interpolation
Circular interpolation
Arc and ellipse moves
Grid coordinate moves
Soft limits
Backlash compensation
Adjustable acceleration/deceleration
Relative/absolute modes
Motion control sequence interpreter
Input/Output port control
Standby mode control.
Motor speeds independent of computer speed
Parameter saving and loading
User definable units conversion

File Names

File names are coded to describe their contents.



Quick-Basic (Not Q-Basic)

MD2QB2S.TXT	Documentation file.
MD2QB2S.BAS	Subroutine library.
MD2QB2S.BI	Include file.
MD2QB2E1.BAS	Example program #1.
MD2QB2E1.MAK	Make file for example program #1.

Visual-Basic for DOS

MD2VD2S.TXT	Documentation file.
MD2VD2S.BAS	Subroutine library.
MD2VD2S.BI	Include file.
MD2VD2E1.FRM	Form for example program #1.
MD2VD2E1.MAK	Make file for example program #1.

Visual-Basic for Windows

MD2VW2S.TXT	Documentation file.
MD2VW2S.BAS	Subroutine library.
MD2VW2E1.FRM	Form for example program #1.
MD2VW2E1.FRX	Graphics for example program #1.
MD2VW2E1.MAK	Make file for example program #1.
MHELP.VBX	DLL for I/O from MicroHelp.

Quick-C (and other C languages)

MD2QC2S.TXT	Documentation file.
MD2QC2S.C	Subroutine library.
MD2QC2S.H	Include file.
MD2QC2E1.C	Example program #1.
MD2QC2E1.MAK	Make file for example program #1.

Note: This list may vary as the level 2 subroutine library is updated.

Concepts

The level 2 subroutine library was used to create the MD-2 program which was written in Visual Basic for DOS. Because of this, concepts and features of the MD-2 program are the same as the level 2 subroutine library. The manual for the MD-2 program can be used as a source of information to prevent duplication here. Look through the manual and read the sections concerning these subjects:

Ports and Motor Numbers	Soft Limits
Calibration	Interrupts
Enable / Disable MD-2	Input / Output Port
Holding Motors	Home Moves
Standby Mode	Line Moves
Absolute / Relative Moves	Circle and Arc Moves
Speeds and Ramping	Grid Moves
Backlash Compensation	Motor Parameters
Step Types	Parameter Files
Units Conversion	Motion Programs

Motion Sequences (Programs)

In the MD-2 program, the operator can edit, load, save and run motion control programs. In the level 2 subroutine library, these motion control programs are referred to as sequences. Sequences are programs much like Quick Basic programs but without complex features such as IF-THEN's, DO's, or calculations. Sequences also have a limit of 32,000 characters in length. In a sequence program, the programmer can set any motor parameter and call most MD-2 subroutines in the level 2 subroutine library. Commands such as BEEP, SLEEP, END and comments are also permitted. Subroutines are provided which allow the operator to load, save and run sequences.

Calibration Files

The level 2 subroutine libraries use a calibration file to control the speeds of the motors. The calibration file compensates for the computer speed. A subroutine is provided which creates this calibration file. It is important to run the calibration subroutine when the computer is in the exact environment that the program will ultimatly be used in. Calibrating when the program is running in debug or development mode will result in incorrect motor speeds when run as a stand alone program. Sharing calibration files between different languages or with the MD-2 program may result in inaccurate motor speeds or program errors. Place a calibration option in each custom program you write to allow it to create a new file.

Subroutines

Subroutines are provided to perform almost any motion control function imaginable. Each subroutine is described here along with an example of its usage.

MD2SETUP

Used at the beginning of a program to initialize motor parameters to their default values, checks for printer ports and loads a calibration file if present. If MD2CALFILE is not set to a file name, it will be set to MD2.CAL. Use this subroutine before using any others in this library.

```
MD2SETUP        'Initialize MD-2 system.
```

MD2ON

Used to turn ON (enable) an MD-2 system. This must be used before a motor can be moved or the I/O port controlled. The affected MD-2 system is determined by the setting of the MD2MOTOR parameter. Setting MD2MOTOR to 1, 2 or 12 will select the MD-2 for motors 1 and 2, setting to 3, 4 or 34 will select the MD-2 for motors 3 and 4, setting to 5, 6 or 56 will select the MD-2 for motors 5 and 6.

```
MD2MOTOR = 3 : MD2ON            'Turn on MD-2 for motors 3 and 4.
```

MD2OFF

Used to turn OFF (disable) an MD-2 system. This can be used at the end of a program to turn everything OFF or it can be used to simply de-energize the motors. MD2OFF also turns off all output signals on the I/O port. The affected MD-2 system is determined by the setting of the MD2MOTOR parameter. Setting MD2MOTOR to 1, 2 or 12 will select the MD-2 for motors 1 and 2, setting to 3, 4 or 34 will select the MD-2 for motors 3 and 4, setting to 5, 6 or 56 will select the MD-2 for motors 5 and 6.

```
MD2MOTOR = 56 : MD2OFF          'Turn off MD-2 for motors 5 and 6.
```

MD2STANDBYON

Puts the selected MD-2 system in standby mode which reduces current to the motors by 50%. This results in lower heat dissipation during standstill while maintaining some holding torque. Use this when the motor must be standing still for a long period of time and some, but not all, holding torque is needed. Standby mode affects both motors on the MD-2 system. The setting of MD2MOTOR determines which MD-2 system is affected.

```
MD2MOTOR = 3 : MD2STANDBYON    'Standby mode ON for motors 3 & 4.
```

MD2STANDBYOFF

Turns OFF standby mode which returns full current to both motors on the system. The setting of MD2MOTOR determines which MD-2 system is affected.

```
MD2MOTOR = 3 : MD2STANDBYON    'Standby mode ON for motors 3 & 4.
```

MD2CALIBRATE

Enables accurate motor speed calculations on a particular computer. Use this subroutine only once on a computer. Calibration information is saved to a disk file who's name is stored in MD2CALFILE and is read by MD2SETUP at the beginning of a motion control program. Motors can not be moved unless a calibration file exists. Calibration normally takes 10 minutes. Disable TSR's and don't interrupt the computer speed to insure reliable results.

```
'Calibrate and save to disk.
PRINT "Calibrating, please wait . . . . ";
MD2CALIBRATE
BEEP
```

MD2PARLOAD

Loads motor parameters from a disk file in the current directory. The parameter file name is stored in MD2PARFILE. All motor, circle and grid parameters are loaded. Eliminates the need to set all motor parameters manually in code. Normally used near the beginning of a program to load the initial values of parameters. The file is a text file which can be edited with a standard text editor but it's safer to create the file by using the MD2PARSAVE subroutine.

```
MD2PARFILE = "PROJECT1.PAR" : MD2PARLOAD     'Load parameters.
```

MD2PARSAVE

Saves motor, circle and grid parameters to a disk file in the current directory. The parameter file name is stored in MD2PARFILE.

```
MD2PARFILE = "MD2.PAR" : MD2PARSAVE     'Save parameters.
```

MD2PAREDIT

Allows the user to edit all motor, circle and grid parameters using the keyboard. Save, load and print functions are also available. Does not work in some screen modes, in VB/Windows, or while forms are showing in VB/DOS.

```
MD2PAREDIT          'Allow user to edit parameters.
```

MD2SEQLOAD

Loads a motion control program (sequence) into MD2SEQUENCE. The sequence file name is stored in MD2SEQFILE.

```
'Load a motion program sequence.
MD2SEQFILE = "MD2PGM2.SEQ"
MD2SEQLOAD
```

MD2SEQSAVE

Saves the variable MD2SEQUENCE to the file name stored in MD2SEQFILE.

```
'Save current motion sequence program.
MD2SEQFILE = "PGM144.SEQ"
MD2SEQSAVE
```

MD2SEQRUN

Executes the motion sequence program found in MD2SEQUENCE.

```
'Load and run a sequence.
MD2SEQFILE = "MD2PGM44.SEQ"
MD2SEQLOAD
MD2POINTER = 1
MD2SEQRUN
```

MD2INPUTS

Reads the condition of the input bits on the MD-2 I/O port and the condition of the home switches for all three MD-2 systems and sets MD2INPUT(X) accordingly. Where X indicates the input bit. Input signals are active low meaning that connecting a signal to ground (0 volts) indicates TRUE and raising a signal to +5 volts indicates FALSE. The following chart shows which element of MD2INPUT(X) represents which signal. If a signal is TRUE (0 volts), then the variable MD2INPUT(X) will be set to TRUE (-1), otherwise it will be set to FALSE (0).

VARIABLE	PORT	SIGNAL
MD2INPUT(11)	3BC	Home switch #1
MD2INPUT(12)	3BC	Home switch #2
MD2INPUT(13)	3BC	Input signal #1
MD2INPUT(14)	3BC	Input signal #2
MD2INPUT(15)	3BC	Input signal #3
MD2INPUT(21)	378	Home switch #1
MD2INPUT(22)	378	Home switch #2
MD2INPUT(23)	378	Input signal #1
MD2INPUT(24)	378	Input signal #2
MD2INPUT(25)	378	Input signal #3
MD2INPUT(31)	278	Home switch #1
MD2INPUT(32)	278	Home switch #2
MD2INPUT(33)	278	Input signal #1
MD2INPUT(34)	278	Input signal #2
MD2INPUT(35)	278	Input signal #3

Example use of the MD2INPUTS subroutine.

```
'Display status of home switch #2.
MD2INPUTS
IF MD2INPUT(22) = -1 THEN
     PRINT "TRUE"
ELSE
     PRINT "FALSE"
ENDIF
```

MD2OUTPUTS

The MD2OUTPUTS subroutine allows the programmer to control the 2 output signals on the I/O port of each MD-2 system. The setting of the MD2OUTPUTCODE parameter determines the MD-2 system, the I/O signal, and the action (On or Off).

OUTPUT CODE	PORT	SIGNAL	ACTION
110	3BC	#1	OFF
111	3BC	#1	ON
120	3BC	#2	OFF
121	3BC	#2	ON
210	378	#1	OFF
211	378	#1	ON
220	378	#2	OFF
221	378	#2	ON
310	278	#1	OFF
311	278	#1	ON
320	278	#2	OFF
321	278	#2	ON

Example use of the MD2OUTPUTS subroutine.

```
'Turn on output #2.
MD2OUTPUTCODE = 121
MD2OUTPUTS
```

MD2HOME

Moves the motor specified by the MD2MOTOR parameter to the home position using the home switch. Ramping is not performed, the MD2MINSPEED(M) parameter determines the motor's speed. The direction of the home switch is determined by the MD2HOMEDIR(M) parameter. After reaching the home switch, the motor will move forward by the distance stored in the MD2HOMEOFFSET(M) parameter. Only one motor can be moved home at a time.

```
'Home motors 3 and 4.
MD2MOTOR = 3
MD2HOME
MD2MOTOR = 4
MD2HOME
```

MD2MOVE

Used to move one or two motors. The MD2MOTOR parameter determines which motor or motor pair will be moved (1, 2, 3, 4, 5, 6, 12, 34, or 56). All motor parameters apply such as backlash compensation, soft limits, units conversion, move type, step type, etc. Ramping will be performed. When moving two motors, both will start and stop at the same time (linear interpolation).

```
'Move motor 1 and 2.
MD2MOTOR = 12
MD2TARGET(1) = 12          'Motor 1, 12 units.
MD2TARGET(2) = 3.3         'Motor 2, 3.3 units.
MD2MOVETYPE = "A"          'Absolute move.
MD2MOVE

'Move motor 3.
MD2MOTOR = 3
MD2TARGET(3) = -144.2      'Reverse 144.2 units.
MD2MOVETYPE = "R"          'Relative move.
MD2MOVE
```

MD2CIRCLE

Used to move two motors connected to an XY positioning table in a circular pattern. Partial circle (arcs), ellipses, partial ellipses, and polygons are also possible by change the circle parameters. Changing the MD2CIRCLERADIUS parameters will create an ellipse, changing MD2CIRCLECHORD will create polygons. Ramping is not performed, the MD2MINSPEED(M) parameter determines the motor's speed.

This example moves motors 3 and 4 in a complete circle with a radius of 2 units and a center position of 10.5 and 4.1.

```
'Move circle.
MD2CIRCLERADIUSX=2
MD2CIRCLERADIUSY=2
MD2CIRCLECENTERX=10.5
MD2CIRCLECENTERY=4.1
MD2CIRCLESTART=0
MD2CIRCLEARC=360
MD2CIRCLECHORD=1
MD2MOVETYPE="A"
MD2MOTOR=34
MD2CIRCLE
```

This example moves motors 1 and 2 in a 90 degree arc with a radius of 4 units and a center position which is 1 unit forward from the current position in the X axis and 1 unit forward from the current position in the Y axis.

```
'Move circle.
MD2CIRCLERADIUSX=4
MD2CIRCLERADIUSY=4
MD2CIRCLECENTERX=1
MD2CIRCLECENTERY=1
MD2CIRCLESTART=0
MD2CIRCLEARC=90
MD2CIRCLECHORD=1
MD2MOVETYPE="R"
MD2MOTOR=12
MD2CIRCLE
```

MD2GRID

Moves two motors using grid coordinates. Simplifies tasks that require positioning to rows and columns of a grid.

The following code will perform a grid move.

```
'Grid move to column 1, row 2.
MD2MOTOR = 34
MD2GRIDBEGINX = 0
MD2GRIDBEGINY = 0
MD2GRIDSPACEX = 2.5
MD2GRIDSPACEY = 2.5
MD2GRIDTARGETX = 1
MD2GRIDTARGETY = 2
MD2GRID
```

Status Codes

The following status codes are returned in the MD2STATUS variable when a subroutine is complete. It tells the program if an operation was successful or if it failed.

CODE	MEANING
O	OK, operation successful.
B	Failed, bad parameter.
L	Failed, soft limit exceeded.
K	Failed, keypress interrupted move.
F	Failed, file error.
P	Failed, port not available.
S	Failed, setup (MD2SETUP) not performed yet.
E	Failed, MD-2 not enabled.
C	Failed, calibration needed.

Parameters

Each motor has several parameters which are used to control their behavior during moves. Some parameters apply to individual motors while others apply to all motors. When setting a parameter that applies to a particular motor, the motor number (1 through 6) is placed in parenthesis following the parameter name. This list describes each parameter, what their valid values are, and how to set them in a motion program.

Parameter: Backlash Compensation
Description: Used to increase positioning accuracy on mechanical systems that have gear backlash or belt stretch. The backlash values is added to the motion when the motor's direction changes.
Values: Any valid unit value.
Default: 0.
Example: MD2BACKLASH(1) = .01 'Set backlash for motor 1 to .01 units.

Parameter: Home Direction
Description: Controls direction which the motor must move to reach the home switch. Has the effect of swapping all directions for motor moves .
Values: 0 = reverse, -1 = forward.
Default: 0.
Example: MD2HOMEDIR(3) = -1 'Set home direction to forward for motor 3.

Parameter: Home Offset
Description: Causes the motor to move away from the home switch in the forward direction when moving home. This position is referred to as position 0, home.
Values: Any valid unit value.
Default: 0.
Example: MD2HOMEOFFSET(6) = .1 'Set home offset to .1 units for motor 6.

Parameter: Hold motors after moves.
Description: Controls holding mode of all motors. When set to true, motors are left energized after moves which causes them to resist shaft rotation. When set to false, motors are de-energized after moves which resists heat buildup.
Values: 0=false (off), -1=true (on)
Default: 0.
Example: MD2HOLD = -1 'Turn hold on for all motors.

Parameter: Interrupts left on during moves.
Description: Allows interrupts to be turned off during motor moves to maximize smoothness. Controls interrupts received from com ports, modems and mice.
Values: 0=false (off), -1=true (on)
Default: 0.
Example: MD2INTERRUPTS = 0 'Turn interrupts off.

Parameter: Limit - Forward
Description: Limits motor motion beyond a preset limit in the forward direction. Any move that would exceed this limit will not be performed.
Values: Any valid unit value.
Default: 999999.
Example: MD2LIMITF(2) = 18 'Set forward limit for motor 2 to 18 units.

Parameter: Limit - Reverse
Description: Limits motor motion beyond a preset limit in the reverse direction. Any move that would exceed this limit will not be performed.
Values: Any valid unit value.
Default: -999999.
Example: MD2LIMITR(2) = 0 'Set forward limit for motor 2 to 0 units.

Parameter: Selected motor(s)
Description: Sets the motor (1-6) or motor pair (12, 34 or 56) that are to be acted upon. Line moves can be single or dual motor moves. Circle and grid moves must be dual motor moves. Also used to select MD-2 system for functions such as MD2ON, MD2OFF, MD2STANDBYON/OFF, etc.
Values: 1,2,3,4,5,6,12,34,56.
Default: 1.
Example: MD2MOTOR = 3 'Set motor number to 3.

Parameter: Motor name
Description: Used simply as a description of the motors function and is only used for display.
Values: Any 8-character combination of numbers and letters.
Default: "Motor #1", "Motor #2", etc.
Example: MD2MOTORNAME(4) = "X axis" 'Set motor name for motor 4.

Parameter: Maximum Speed.
Description: Determines the maximum speed that a motor will reach after accelerating.
Values: Given in units per second.
Default: 500.
Example: MD2MAXSPEED(1) = 3.2 'Motor 1 maximum speed = 3.2 units per second.

Parameter: Minimum Speed.
Description: Determines the minimum speed that a motor will begin and end with.
Values: Given in units per second.
Default: 100.
Example: MD2MINSPEED(1) = .5 'Motor 1 minimum speed = .5 units per second.

Parameter: Move Type.
Description: Selects relative or absolute moves. Affects all motors.
Values: "A" = absolute, "R" = relative.
Default: "R".
Example: MD2MOVETYPE = "A" 'Absolute mode.

Parameter: Output Code
Description: Selects the desired output bit to be controlled before using the MD2OUTPUTS function. See section on I/O port for complete description of codes.
Values: 110, 111, 120, 121, 210, 211, 220, 221, 310, 311, 320, 321.
Default: 110.
Example: MD2OUTPUTCODE = 211 'MD-2 #2, Output #1, ON.

Parameter: Position
Description: The current motor position in units. Updated automatically by move functions. Can be set manually to force the system to think it's at a different position.
Values: Any valid unit value.
Default: 0.
Example: MD2POSITION(2) = 12 'Set current position to 12 for motor 2

Parameter: Slope
Description: Determines the rate of acceleration and deceleration. The value sets the distance of motion needed to accelerate from the minimum speed to the maximum speed and to decelerate from the maximum speed to the minimum speed.
Values: Any valid unit value. 0 will cause constant speed using maximum speed value.
Default: 100.
Example: MD2SLOPE(5) = 1.5 'Set slope for motor 5 to 1.5 units.

Parameter: Step Type
Description: Sets the type of step patterns given to the motors. Affects all motors.
Values: "H" = half step, "D" = double-full step, "S" = single-full step.
Default: "H".
Example: MD2STEPTYPE = "H" 'Set step type to half.

Parameter: Target
Description: Determines the distance and direction a motor will move in relative mode and determines the desired target position in absolute moves.
Values: Any valid unit value.
Default: 400.
Example: MD2TARGET(3) = 4.1 'Motor 3 target = 4.1 units.

Parameter: Units value.
Description: Sets the number of motor steps for each user defined unit. Inches, mm and feet are common units.
Values: Positive whole numbers.
Default: 1.
Example: MD2UNITS(3) = 200 'Motor 3, 200 steps per inch.

Parameter: Unit Name.
Description: Describes units. Only used for display purposes.
Values: Any 8-character combination of numbers and letters.
Default: "Steps"
Example: MD2UNITNAME(4) = "Inches" 'Motor 4 = Inches.

Circle Parameters

Parameter: X-Axis Radius
Description: Sets the radius of the X axis motor. Setting different than the Y-axis radius will result in an elliptical pattern.
Values: Any valid unit value.
Default: 1.
Example: MD2CIRCLERADIUSX = 2.2 'Set X-axis radius.

Parameter: Y-Axis Radius
Description: Sets the radius of the Y axis motor. Setting different than the X-axis radius will result in an elliptical pattern.
Values: Any valid unit value.
Default: 1.
Example: MD2CIRCLERADIUSY = 2.2 'Set Y-axis radius.

Parameter: X-Axis Center Position
Description: Sets the center position of the circle on the X axis motor.
Values: Any valid unit value.
Default: 1.
Example: MD2CIRCLECENTERX = 4.5 'Set X-axis center.

Parameter: Y-Axis Center Position
Description: Sets the center position of the circle on the Y axis motor.
Values: Any valid unit value.
Default: 1.
Example: MD2CIRCLECENTERY = 4.5 'Set Y-axis center.

Parameter: Start Angle
Description: Sets the starting angle in degrees.
Values: Integer 0 - 360.
Default: 0.
Example: MD2CIRCLESTART = 180 'Set starting angle to 180 degrees.

Parameter: Arc Angle
Description: Sets the arc angle in degrees. 360 will result in a circle, 180 in 1/2 circle etc.
Values: Integer 0 - 360.
Default: 360.
Example: MD2CIRCLEARC = 90 'Set arc angle to 90 degrees.

Parameter: Chord Angle
Description: Sets the chord angle in degrees. This is the distance between points on the circle that are connected together to create the arc or circle. The maximum resolution is 1 which causes a point to be placed every 1 degree. 45 will create an octagon, 60 will create a hexagon.
Values: Integer 1 - 90.
Default: 1.
Example: MD2CIRCLECHORD = 1 'Set chord angle to 1 degree.

Grid Parameters

Parameter: X-Axis Beginning Position.
Description: Sets the beginning of the grid on the X axis motor.
Values: Any valid unit value.
Default: 1.
Example: MD2GRIDBEGINX = 0 'Set X-axis beginning position.

Parameter: Y-Axis Beginning Position.
Description: Sets the beginning of the grid on the Y axis motor.
Values: Any valid unit value.
Default: 1.
Example: MD2GRIDBEGINY = 0 'Set Y-axis beginning position.

Parameter: X-Axis Spacing.
Description: Sets the spacing between grid points on the X axis motor.
Values: Any valid unit value.
Default: 1.
Example: MD2GRIDSPACEX = 1 'Set X-axis grid spacing.

Parameter: Y-Axis Spacing.
Description: Sets the spacing between grid points on the Y axis motor.
Values: Any valid unit value.
Default: 1.
Example: MD2GRIDSPACEY = 1 'Set Y-axis grid spacing.

Parameter: X-Axis Target.
Description: Sets the desired target grid coordinate for the X axis motor.
Values: Integers.
Default: 1.
Example: MD2GRIDTARGETX = 0 'Set X-axis target position.

Parameter: Y-Axis Target.
Description: Sets the desired target grid coordinate for the Y axis motor.
Values: Integers.
Default: 1.
Example: MD2GRIDTARGETY = 0 'Set Y-axis target position.

Example Programs

The following example program shows the use of the subroutine library for Quick-Basic. Its use is similar to other languages. Consult the documentation file for each language for specific details on its use. It may be necessary to change some parameters for this program to operate on your computer.

```
'INCLUDE FILE FOR SUBROUTINE LIBRARY.
'$INCLUDE: 'MD2QB2S.BI'

'SET DEFAULTS AND FINDS PORTS, LOAD CALIBRATION FILE.
MD2SETUP
IF MD2STATUS <> "O" THEN BEEP : END

'LOAD PARAMETER FILE.
MD2PARFILE = "MD2.PAR"              'THIS FILE MUST EXIST.
MD2PARLOAD
IF MD2STATUS <> "O" THEN BEEP : END

'TURN ON MD-2 FOR MOTORS 3 & 4.
MD2MOTOR = 3                        'MOTOR 4 WOULD ALSO WORK HERE.
MD2ON                               'TURN ON MD-2.

'MOVE MOTORS 3 & 4 HOME.
MD2MOTOR = 3                        'MOVE 3 HOME.
MD2HOME
MD2MOTOR = 4                        'MOVE 4 HOME.
MD2HOME

'MOVE MOTOR 3 20 UNITS FORWARD.
MD2MOTOR = 3                        'MOTOR 3.
MD2TARGET(3) = 20                   '275 UNITS FORWARD.
MD2MOVETYPE = "R"                   'RELATIVE.
MD2MOVE                             'MOVE THE MOTOR.

'MOVE MOTORS 3 & 4 TO POSITIONS 12 AND 4.
MD2MOTOR = 34                       'MOTORS 3 & 4.
MD2TARGET(3) = 12                   'TO POSITION 12 UNITS.
MD2TARGET(4) = 4                    'TO POSITION 4 UNITS.
MD2MOVETYPE = "A"                   'ABSOLUTE MOVE.
MD2MOVE

'DISPLAY MOTOR POSITIONS.
PRINT "MOTOR 3 POSITION IS: ";MD2POSITION(3)
PRINT "MOTOR 4 POSITION IS: ";MD2POSITION(4)

'TURN OFF MD-2 SYSTEM.
MD2MOTOR = 3                        'SELECT MOTOR AT THAT PORT.
MD2OFF                              'TURN OFF MD-2.

END
```

This Quick-Basic example program allows the user to edit motor parameters and run a motion sequence program.

```
     'INCLUDE FILE FOR SUBROUTINE LIBRARY.
     '$INCLUDE: 'MD2QB2S.BI'

     'SET DEFAULTS AND FINDS PORTS, LOAD CALIBRATION FILE.
     MD2SETUP

     'LOAD PARAMETER FILE.
     MD2PARFILE = "MD2.PAR" : MD2PARLOAD
     MD2MOVE

MENULOOP:

     'DISPLAY MENU.
     CLS
     PRINT "1 - EDIT PARAMETERS."
     PRINT "2 - RUN SEQUENCE."
     PRINT "3 - QUIT"
     PRINT : INPUT "ENTER SELECTION: ",ENTRY$

     'EDIT PARAMETERS?
     IF ENTRY$ = "1" THEN MD2PAREDIT

     'RUN SEQUENCE?
     IF ENTRY$ = "2" THEN
          INPUT "ENTER SEQUENCE FILE NAME: ",MD2SEQFILE
          MD2SEQLOAD
          IF MD2STATUS = "0" THEN
               MD2POINTER = 1
               MD2SEQRUN
          ELSE
               INPUT "ERROR LOADING FILE, PRESS ENTER ",ENTRY$
          ENDIF
     ENDIF

     'QUIT?
     IF ENTRY$ = "3" THEN END

     GOTO MENULOOP
```

P.O. Box 1574
Hurst, Texas 76053 USA
Ph: (817) 571-4528
Fax: (817) 571-2317
E-mail: info@robotics.com
Web: http://www.robotics.com