# Programming and Software Reference for:

- **MCode**
- **MCode/TCP**

MDrive, MForce and Hybrid Motion Control Products

Schneider Electric

| MCode Programming and Software Reference | | |
|---|---|---|
| **Date** | **Revision** | **Changes** |
| 03/08/2006 | R030806 | Initial Release |
| 02/16/2007 | R021607 | Added TR as a reserved for future use word. |
| 04/20/2007 | R042007 | Revised manual to reflect firmware version MDI3.006. Added D5, analog input filter variable, various corrections and expansions of commands, updated error table. |
| 10/25/2007 | R102507 | Revised manual to reflect firmware version MDI3.007 |
| 02/27/2008 | R022708 | Revised Manual to reflect firmware version MDI3.008. Added QD (Qeued - Page 45) command and PW variable (Page 44 and Appendex G: MForce PWM Current Control). |
| 06/16/08 | R061608 | Updated covers, added disclaimer information. Expanded on Trip definitions. Expanded on Error 92. Minor corrections throughout. |
| 08/14/2008 | R081408 | Added NE (Numeric Enable Command. Relevant to Firmware Version 3.009 |
| 04/29/2009 | R080409 | Added Hybrid commands, reformated to reflect current documentation standards and styling. Updated to reflect firmware release 3.010. |
| 08/04/2009 | R080409 | Added Error 63 - character overrun |
| 07/10/2009 | R071009 | Added Hybrid Flag AF, added errors 87, 88 and 89 for Hybrid |
| 11/23/2009 | R112309 | Appended the HT (Hold Current Delay Time) command to reflect the behavioral difference between Legacy and Plus MDrive units. |
| 02/25/2010 | R022510 | Updates and corrections throughout. |
| 07/02/2010 | R070210 | Updates and corrections throughout. Added warning about enabling clock mode when using limits (pg 5-12). |
| 10/05/2010 | R100510 | Added support for MDrive Ethernet products. Section 6: IMS Terminal renamed to Section 6: Supporting Software with the addition of instructions for the TCP/IP Configuration Utility for MDrive Ethernet and MDrive Hybrid Ethernet. Changed name of MDrive AccuStep to MDrive Hybrid throughout. |
| 02/14/2011 | R021411 | Added procedure for upgrading Ethernet Controller firmware for Ethernet equipped MDrivePlus and MDrive Hybrid models. |
| 04/12/2011 | R041211 | Added support for MDrive Hybrid firmware version 4.004 |
| 08/05/2011 | R080511 | Added second parameter details supporting hybrid calibration. Added correct range for torque speed and system speed. |
| 06/28/2012 | R062812 | Added changes supporting default MDrive Hybrid calibration method. |

The information in Schneider Electric Motion USA product documentation and on this web site has been carefully checked and is believed to be accurate; however, no responsibility is assumed for inaccuracies.

Schneider Electric Motion USA reserves the right to make changes without further notice to any products to improve reliability, function or design. Schneider Electric Motion USA does not assume any liability arising out of the application or use of any product or circuit described; neither does it convey any license under its patent rights of others.

Schneider Electric Motion USA's general policy does not recommend the use of its products in life support or aircraft applications wherein a failure or malfunction of the product may directly threaten life or injury. Per the terms and conditions of sales of Schneider Electric Motion USA, the user of Schneider Electric Motion USA products in life support or aircraft applications assumes all risks of such use and indemnifies Schneider Electric Motion USA against all damages. aircraft applications assumes all risks of such use and indemnifies Schneider Electric Motion USA against all damages.

*MCode Programming and Software Reference Revision R062812*

Copyright © Schneider Electric Motion USA, All Rights Reserved

## Important information

The drive systems described here are products for general use that conform to the state of the art in technology and are designed to prevent any dangers. However, drives and drive controllers that are not specifically designed for safety functions are not approved for applications where the functioning of the drive could endanger persons. The possibility of unexpected or un-braked movements can never be totally excluded without additional safety equipment. For this reason personnel must never be in the danger zone of the drives unless additional suitable safety equipment prevents any personal danger. This applies to operation of the machine during production and also to all service and maintenance work on drives and the machine. The machine design must ensure personal safety. Suitable measures for prevention of property damage are also required.

## Qualification of personnel

Only technicians who are familiar with and understand the contents of this manual and the other relevant documentation are authorized to work on and with this drive system. The technicians must be able to detect potential dangers that may be caused by setting parameters, changing parameter values and generally by the operation of mechanical, electrical and electronic equipment.

The technicians must have sufficient technical training, knowledge and experience to recognise and avoid dangers.

The technicians must be familiar with the relevant standards, regulations and safety regulations that must be observed when working on the drive system.

## Intended Use

The drive systems described here are products for general use that conform to the state of the art in technology and are designed to prevent any dangers. However, drives and drive controllers that are not specifically designed for safety functions are not approved for applications where the functioning of the drive could endanger persons. The possibility of unexpected or unbraked movements can never be totally excluded without additional safety equipment.

For this reason personnel must never be in the danger zone of the drives unless additional suitable safety equipment prevents any personal danger. This applies to operation of the machine during production and also to all service and maintenance work on drives and the machine. The machine design must ensure personal safety. Suitable measures for prevention of property damage are also required.

In all cases the applicable safety regulations and the specified operating conditions, such as environmental conditions and specified technical data, must be observed.

The drive system must not be commissioned and operated until completion of installation in accordance with the EMC regulations and the specifications in this manual. To prevent personal injury and damage to property damaged drive systems must not be installed or operated.

Changes and modifications of the drive systems are not permitted and if made all no warranty and liability will be accepted.

The drive system must be operated only with the specified wiring and approved accessories. In general, use only original accessories and spare parts.

The drive systems must not be operated in an environment subject to explosion hazard (ex area).

This page intentionally left blank

# Table of Contents

Important information ..... 3
Qualification of personnel ..... 3
Intended Use ..... 3
**1 Introduction ..... 1-1**
1.1 About this manual ..... 1-1
1.2 MCode device overview ..... 1-1
1.3 Documentation reference ..... 1-1
1.4 Product software ..... 1-1
1.4.1 Communications converter drivers ..... 1-1
1.4.2 ASCII program editor and ANSI terminal emulator. 1-1

**2 Safety ..... 2-1**
2.1 Qualification of personnel ..... 2-1
2.2 Intended Use ..... 2-1
2.3 Hazard Categories ..... 2-2
2.4 General safety instructions ..... 2-2

**3 Introduction to MCode programming ..... 3-1**
3.1 Operational modes ..... 3-1
3.2 Basic components of MCode software ..... 3-1
3.2.1 Instructions ..... 3-1
3.2.2 Variables ..... 3-2
3.2.3 Flags ..... 3-3
3.2.4 Keywords ..... 3-3
3.2.5 Math functions ..... 3-4
3.3 Program structuring ..... 3-4
3.3.1 Programming aids ..... 3-5
3.4 Commonly used variables and instructions ..... 3-6
3.4.1 Variables ..... 3-6
3.4.2 Motion instructions ..... 3-7
3.4.3 I/O instructions ..... 3-9
3.4.4 System instructions ..... 3-10
3.4.5 Program instructions ..... 3-11

**4 Command summary ..... 4-1**
4.1 Setup instructions, variables and flags ..... 4-1
4.2 Misc instructions, variables and flags ..... 4-2
4.3 Motion instructions, variables and flags ..... 4-3
4.4 Program instructions, variables and flags ..... 4-4
4.5 I/O instructions, variables and flags ..... 4-5
4.6 Position related instructions, variables and flags 4-6
4.7 Encoder related instructions, variables and flags ... 4-6
4.8 Hybrid specific instructions, variables and flags 4-7
4.9 Math functions ..... 4-8

**5 MCode command details ..... 5-1**
A (Acceleration) ..... 5-1
AL (Retreive all parameters) ..... 5-1
AF (Hybrid flags) ..... 5-2
AS (Hybrid mode) ..... 5-3
AT (Reservef) ..... 5-3
BD (Communications BAUD rate) ..... 5-4
BP (Break point) ..... 5-5

BR (Program branch) ..... 5-6
BY (Program executing) ..... 5-6
C1 (Counter 1) ..... 5-7
C2 (Counter 2) ..... 5-7
CA (Calibration mode) ..... 5-8
CB (Control bounds) ..... 5-9
CC (Calibration current) ..... 5-9
CE (Software reset enable) ..... 5-10
CF (Clear locked rotor fault) ..... 5-10
CK (Check sum enable) ..... 5-11
CL (Call subroutine) ..... 5-12
CM (Clock mode enable) ..... 5-12
CP (Clear program) ..... 5-13
CR (Clock ratio) ..... 5-13
CT (Calibration time) ..... 5-14
CW (Clock width) ..... 5-14
D (Deceleration) ..... 5-15
D1-D4 (Input switch debounce) ..... 5-15
D5 (Analog input filter) ..... 5-16
D9-D12 (Input switch debounce) ..... 5-16
DB (Encoder deadband) ..... 5-17
DC (Decrement variable) ..... 5-17
DE (Drive enable) ..... 5-18
DG (Disable global response) ..... 5-18
DN (Device name) ..... 5-19
E (End program) ..... 5-19
EE (Encoder enable) ..... 5-20
EF (Error flag) ..... 5-20
EL (Remote encoder line count) ..... 5-21
EM (Echo mode) ..... 5-21
ER (Device name) ..... 5-22
ES (Escape) ..... 5-22
EX (Execute program) ..... 5-23
FC (Filter capture) ..... 5-24
FD (Restore factory defaults) ..... 5-25
FM (Filter motion) ..... 5-25
FT (Internal use only) ..... 5-26
H (Hold program execution) ..... 5-26
HC (Motor holding current) ..... 5-27
HI (Home to index mark) ..... 5-28
HM (Home to home switch) ..... 5-29
HT (Hold current delay time) ..... 5-31
I1, I2, I3 and I4 (Read inputs 1, 2 ,3 or 4) ..... 5-31
I5 (Read analog input) ..... 5-32
I6 (Read encoder index mark) ..... 5-32
I7, I8 (Reserved) ..... 5-32
I9, I10, I11 and I2 (Read inputs 9, 10, 11 or 12) ..... 5-33
I13 (Reserved) ..... 5-33
IC (Increment variable) ..... 5-33
IF (Input variable pending) ..... 5-34
IL (Read inputs 1 - 4 as a group) ..... 5-34
IH (Read inputs 9 - 12 as a group) ..... 5-35
IN (Read inputs 1 - 4 and 9 - 12 as a group) ..... 5-35
IP (Initialize parameters) ..... 5-36
IT (Read internal temperature) ..... 5-36
IV (Input to variable) ..... 5-37
JE (Jog enable) ..... 5-37

L (List program space) ..... 5-38
LB (Label program or subroutine) ..... 5-38
LD (Lead limits) ..... 5-39
LG (Lag limits) ..... 5-39
LL (Position lead/lag) ..... 5-40
LK (Lock user program) ..... 5-40
LM (Limit stop modes) ..... 5-41
LR (Locked rotor) ..... 5-42
LT (Locked rotor timeout) ..... 5-42
MA (Move to an absolute position) ..... 5-43
MD (Motion mode) ..... 5-43
MF (Make up frequency) ..... 5-44
MP (Moving to position) ..... 5-44
MR (Move to a relative position) ..... 5-45
MS (Microstep resolution select) ..... 5-46
MT (Motor settling delay time) ..... 5-47
MU (Make up mode) ..... 5-48
MV (Moving) ..... 5-48
NE (Numeric enable) ..... 5-49
O1, O2, O3 and O4 (Set outputs 1, 2 ,3 or 4) ..... 5-49
O9, O10, O11 and O12 (Set outputs 9, 10 ,11 or 12) ..... 5-50
OE (On error handler) ..... 5-50
OL (Set outputs 1 - 4 as a group) ..... 5-51
OH (Set outputs 9 - 12 as a group) ..... 5-51
OT (Set outputs 1- 4 and 9 - 12 as a group) ..... 5-52
P (Position counter) ..... 5-52
PC (Position capture at trip) ..... 5-53
PG (Program mode)) ..... 5-53
PM (Position maintenance enable) ..... 5-54
PN (Part number) ..... 5-54
PR (Print specified data or text) ..... 5-55
PS (Pause program execution) ..... 5-55
PW (PWM configuration) ..... 5-56
PY (Party mode enable) ..... 5-57
QD (Queued) ..... 5-58
R1, R2, R3 and R4 (User registers) ..... 5-58
RC (Motor run current) ..... 5-59
RS (Resume program execution) ..... 5-59
RT (Return from subroutine) ..... 5-60
S (Save to NVM) ..... 5-60
S1, S2, S3, S4, S9, S10, S11, S12 (Setup I/O 1 - 4 & 9 - 12) ..... 5-61
S5 (Set analog input) ..... 5-63
S7 & S8 (Setup clock I/O points) ..... 5-64
S13 (Setup capture input/trip output) ..... 5-65
SC (Start calibration) ..... 5-65
SF (Stall factor) ..... 5-66
SL (Slew axis) ..... 5-66
SM (Stall detect mode) ..... 5-67
SN (Serial number) ..... 5-67
SS (System speed) ..... 5-68
ST (Stall flag) ..... 5-68
SU (Execute program on power up) ..... 5-69
TA (Trip on Hybrid status) ..... 5-69
TC (Trip capture) ..... 5-70
TD (Torque direction) ..... 5-70
TE (Trip enable) ..... 5-71
TI (Trip on input) ..... 5-71

TP (Trip on position)....5-72
TQ (Set torque)....5-72
TR (Trip on relative position)....5-73
TS (Set torque speed)....5-74
TT (Trip on position)....5-74
UG (Upgrade firmware)....5-75
UV (Read user variables)....5-75
V (Read velocity)....5-75
VA (Create user variable)....5-76
VC (Velocity changing)....5-76
VI (Initial velocity)....5-77
VM (Maximum velocity)....5-77
VR (Firmware version)....5-78
WT (Warning temperature)....5-78

**6 Supporting Software....6-1**
6.1 IMS Terminal....6-1
6.1.2 Features....6-1
6.1.3 Installing IMS Terminal....6-2
6.1.4 Screen and menu overview....6-2
6.4.2 The menu bar structure and operation....6-3
6.4.3 Edit menu operation....6-3
6.1.5 Configuring IMS Terminal....6-5
6.1.6 Troubleshooting communications....6-7
6.1.7 Configuring function keys....6-8
6.1.8 Creating, downloading and uploading programs....6-10
6.1.9 Program troubleshooting using IMS Terminal....6-12
6.1.10 Upgrading firmware....6-14
6.2 TCP/IP Configuration Utility....6-18
6.2.1 Installation....6-18
6.2.2 Screen overview....6-18
6.2.3 Configuration....6-19
6.2.4 SEM Port tab....6-20

**7 Programming and application notes....7-1**
7.1 Party mode communications....7-1
7.1.1 Response to Echo Mode....7-1
7.1.2 Using Check Sum....7-3
7.1.3 Immediate party mode sample codes....7-4
7.2 Programming the I/O....7-5
7.2.1 I/O availability per device type....7-5
7.2.2 Active states defined....7-6
7.2.3 Digital input functions....7-6
7.2.4 Digital output functions....7-9
7.2.5 Programmable I/O usage examples....7-10
7.2.6 Dedicated digital I/O usage....7-13
7.2.7 Analog input usage....7-15
7.3 Factors impacting motion commands....7-16
7.3.1 Motor steps....7-16
7.3.2 Microsteps: (MS)....7-16
7.3.3 Move Command....7-16
7.3.4 Closed loop control with an encoder....7-16
7.3.5 Linear movement....7-17
7.3.6 Calculating rotary movement....7-19
7.3.7 Programming with the optional encoder enabled 7-21
7.4 MForce PWM configuration....7-25
7.4.1 Description:....7-25

7.4.2 PWM Mask <mask> Parameter ..... 7-25
7.4.3 Max PWM duty cycle (%) <period> parameter 7-26
7.4.4 PWM frequency <sfreq> parameter ..... 7-27
7.4.5 PWM checksum <chksum> parameter ..... (boot write only) ..... 7-28
7.4.6 Boot writes remaining <boot_writes_remaining> ... parameter (read only) ..... 7-29
7.4.7 Example PWM settings by motor ..... specifications ..... 7-29

**8 Sample programs ..... 8-1**
8.1 Move on an input ..... 8-1
8.2 Change velocity during a move ..... 8-2
8.3 Binary mask ..... 8-3
8.4 Closed lLoop ..... 8-5
8.5 User input into variables ..... 8-6
8.6 Closed loop with homing ..... 8-7
8.7 Input trip ..... 8-8

**9 Error codes ..... 9-1**
9.1 I/O errors ..... 9-1
9.2 Data errors ..... 9-1
9.3 Program errors ..... 9-2
9.4 Communications errors ..... 9-2
9.5 System errors ..... 9-3
9.6 Motion errors ..... 9-3
9.7 Hybrid errors ..... 9-4

# List of Figures

Figure 3.1 Recommended MCode program structure ..... 3-4
Figure 5.1 Homing functions sequence of events ..... 5-30
Figure 5.2 Limit modes of operation ..... 5-41
Figure 6.1 IMS Terminal main screen ..... 6-2
Figure 6.2 Function key setup dialog ..... 6-8
Figure 7.1 Step/direction I/O type & configuration ..... 7-13
Figure 7.2 Quadrature I/O configuration ..... 7-14
Figure 7.3 CW/CCW I/O configuration ..... 7-14
Figure 7.4 Trapezoidal move profile ..... 7-17
Figure 7.5 Rotary examples ..... 7-19
Figure 7.6 Encoder waveform ..... 7-21
Figure 7.7 PWM mask bits ..... 7-26
Figure 7.8 PWM frequency range ..... 7-27

# List of Tables

Table 7.1 Response to echo mode - party and check sum are zero (0) ..... 7-2
Table 7.2 Response to echo mode - party is one (1) and check sum is zero (0) ..... 7-3
Table 7.3 Response to echo mode - party is zero (0) and check sum is one (1) ..... 7-3
Table 7.4: Response to echo mode - party and check sum are one (1) ..... 7-3
Table 7.5 Digital input functions ..... 7-7
Table 7.6 Clock and high speed input functions ..... 7-8
Table 7.7 Digital output functions ..... 7-9
Table 7.8 Clock and high speed output functions ..... 7-10
Table 7.8 PWM Mask Settings ..... 7-26
Table 7.9 Typical PWM Mask Settings ..... 7-26
Table 7.10 Maximum PWM frequency ..... 7-27
Table 7.11 Initial PWM frequency ..... 7-27
Table 7.12 Example PWM settings ..... 7-29

# 1 Introduction

## 1.1 About this manual

This manual is devices utilizing the MCode programming language.

## 1.2 MCode versions

### 1.2.1 MCode

MCode is the programming and control language for products communicating over serial RS-422/485.

### 1.2.2 MCode/TCP

MCode/TCP is the MCode language adapted to communicate over Ethernet TCP/IP networks. The function and usage is identical with the exception that the commands related to RS-422/485 communication are disabled. MCode/TCP will connect to TCP or UDP port 503.

## 1.3 MCode device overview

The following product lines use MCode for configuration and programming:

- MDrive Motion Control products
- MForce Motion Control products
- MDrive Hybrid Motion Control products

Many of the MCode commands are device-specific based on available I/O, Hybrid Motion Technology and communications interface. In the command detail (Section 5), each command is listed with device compatibility. Ensure that the command will function with your device before using.

## 1.4 Documentation reference

The following user's manuals are available for the MCode devices:

- Product hardware manual, describes the technical data and installation of the product.
- Product software manual, describes the configuration and programming of the product.
- Quick Reference, describes the basic wiring, connection and use of this product. The quick reference is shipped in printed form with the product.

This documentation is also available for download from our web site at: http://motion.schneider-electric.com

Revision R062812

### 1.4.1 Related documents

*MODBUS/TCP* The MODBUS/TCP Fieldbus manual describes the implementation of the MODBUS protocol for Ethernet equipped MDrive products. When connected to TCP port 502, the MDrive Ethernet will only respond to MODBUS commands.

*EtherNet/IP* The EtherNet/IP Fieldbus manual describes the implementation of the EtherNet/IP protocol for Ethernet equipped MDrive products.

## 1.5 Product software T

The following drivers and software may be downloaded from the web site at: http://www.schneider-electric-motion.us/downloads/software_interfaces.html

### 1.5.1 Communications converter drivers

If using the MD-CC40x-001 communications converter, drivers are required, these drivers are available for download from our web site at http://www.schneider-electric-motion.us/downloads/cable_drivers.html.

### 1.5.2 ASCII program editor and ANSI terminal emulator

Motion control products may be configured and programmed using any standard ANSI terminal emulator and ASCII text editor.

The recommended tool is the IMS Terminal integrated terminal and program editor. IMS Terminal features color-coded editor, multiple-function keys and is pre-configured to operate using the MDrive default settings

Installation and usages instructions are to be found in MCode software manual.

### 1.5.3 TCP/IP configuration utility

The TCP/IP configuration utility is a user interface for configuring the Ethernet equipped MDrive models in either EtherNet/IP MCode/TCP or MODBUS/TCP.

This GUI offers basic setup and test functionality and is required to change the MDrive's IP/SubNet address from the default 192.168.33.1/255.255.0.0.

IMS Terminal is then recommended for programming the MDrive in MCode/TCP mode.

# 2 Safety

## 2.1 Qualification of personnel

Only technicians who are familiar with and understand the contents of this manual and the other relevant documentation are authorized to work on and with this drive system. The technicians must be able to detect potential dangers that may be caused by setting parameters, changing parameter values and generally by the operation of mechanical, electrical and electronic equipment.

The technicians must have sufficient technical training, knowledge and experience to recognise and avoid dangers.

The technicians must be familiar with the relevant standards, regulations and safety regulations that must be observed when working on the drive system.

## 2.2 Intended Use

The drive systems described here are products for general use that conform to the state of the art in technology and are designed to prevent any dangers. However, drives and drive controllers that are not specifically designed for safety functions are not approved for applications where the functioning of the drive could endanger persons. The possibility of unexpected or unbraked movements can never be totally excluded without additional safety equipment.

For this reason personnel must never be in the danger zone of the drives unless additional suitable safety equipment prevents any personal danger. This applies to operation of the machine during production and also to all service and maintenance work on drives and the machine. The machine design must ensure personal safety. Suitable measures for prevention of property damage are also required.

In all cases the applicable safety regulations and the specified operating conditions, such as environmental conditions and specified technical data, must be observed.

The drive system must not be commissioned and operated until completion of installation in accordance with the EMC regulations and the specifications in this manual. To prevent personal injury and damage to property damaged drive systems must not be installed or operated.

Changes and modifications of the drive systems are not permitted and if made no warranty and liability will be accepted.

The drive system must be operated only with the specified wiring and approved accessories. In general, use only original accessories and spare parts.

The drive systems must not be operated in an environment subject to explosion hazard (ex area).

## 2.3 Hazard Categories

Safety notes and general information are indicated by hazard messages in the manual. In addition there are symbols and instructions affixed to the product that warn of possible hazards and help to operate the product safely.

Depending on the seriousness of the hazard, the messages are divided into three hazard categories.

DANGER indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.

**⚠WARNING**

WARNING indicates a potentially hazardous situation, which, if not avoided, **can result** in death, serious injury, or equipment damage.

**⚠CAUTION**

CAUTION indicates a potentially hazardous situation, which, if not avoided, **can result** in injury or equipment damage.

**CAUTION**

CAUTION used without the safety alert symbol, is used to address practices not related to personal injury (e.g. **can result** in equipment damage).

## 2.4 General safety instructions

**⚠ DANGER**

**UNINTENDED CONSEQUENCES OF EQUIPMENT OPERATION**
When the system is started, the drives are usually out of the operator's view and cannot be visually monitored.

• Only start the system if there are no persons in the hazardous area

**Failure to follow these instructions will result in death or serious injury.**

**⚠ DANGER**

**EXPOSED SIGNALS**

Hazardous voltage levels may be present if using an open frame power supply to power the product.

**Failure to follow these instructions will result in death or serious injury.**

**⚠WARNING**

**LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop, overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical functions.
- System control paths may include communication links. Consideration must be given to the implication of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines. 1)
- Each implementation of the product must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death or serious injury.**

1) For USA: Additional information, refer to NEMA ICS 1.1 (latest edition), "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control" and to NEMA ICS 7.1 (latest edition), "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems".

**⚠CAUTION**

**MOVEMENT ON POWER APPLICATION**

Hybrid Motion Technology's functionality requires that the rotor and stator of the motor be in precise alignment. To accomplish this, the product will perform an initial calibration move upon power up consisting of a 6 motor full step (10.8°) move in the clockwise direction, followed by a 3 motor full step (5.4°) move inthe counter-clockwise direction.

- Only power the system if there are no persons in the hazardous area.

**Failure to follow these instructions can result in injury or equipment damage.**

**CAUTION**

HOT PLUGGING!

Do not connect or disconnect power, logic, or communications while the device is in a powered state.

Remove DC power by powering down at the AC side of the DC power supply.

**Failure to follow these instructions can result in equipment damage.**

# 3 Introduction to MCode programming

This section will acquaint the user with basics of MCode programming and the simple 1 and 2 character mnemonics which make up the MCode programming language.

- Operational modes.
- Basic components of the MCode programming language.

## 3.1 Operational modes

There are two operational modes for the MCode compatible products: Immediate and Program.

**1)** **Immediate**: Commands are issued and executed directly to the controller by user input into the terminal window.

**2)** **Program:** Program Mode is used to input user programs into the motion controller.

## 3.2 Basic components of MCode software

There are five basic components of the MCode Programming Language, they are:

1) Instructions
2) Variables
3) Flags
4) Keywords
5) Math functions

### 3.2.1 Instructions

An instruction results in an action. There are four types of instructions:

*Motion* Motion instructions are those that result in the movement of a motor. The syntax for these commands is as follows: type the command followed by a space, and then the velocity or position data. For example:MA 2000 will move the motor to an absolute position of 2000.

*I/O* An I/O instruction results in the change of parameters or the state of an input or output. The syntax for these commands are as follows: type the command then an equal sign, then the data. Example: O2=0 will set output 2 to 0.

*Program* A program instruction allows program manipulation. The syntax of these vary due to the nature of the command. Some command examples would be: PG 100, which toggles the system into program mode starting at address 100; BR LP, I1=1, which will branch to a program labeled LP if input 1 is true.

*System* A system instruction is an instruction that can only be used in immediate mode to perform a system operation such as program execution (EX) or listing the contents of program memory (L). For example: EX 100 will execute a program located at address 100 of program memory space, or EX K1 will execute a program labeled K1.

### 3.2.2 Variables

A Variable is identified by a mnemonic and allows the user to define or manipulate data. These can also be used with the math functions to manipulate data. There are two classes of variables: factory-defined and user-defined. There are 192 user program labels and variables available. The syntax for each variable may differ.

*Factory defined variables* These variables are predefined at the factory. They cannot be deleted. When an FD (Factory Default) instruction is given, these variables will be reset to their factory default values. There are two types of factory defined variables:

- **Read/Writable:** These factory defined variables can have their value altered by the user to affect events inside or outside of a program. For example A (Acceleration variable) can be used to set the Acceleration, or P (Position variable) can be used to set the position counter.
- **Read Only:** These factory defined variables cannot be changed by the user. They contain data that can be viewed or used to affect events inside a program. For example, V (Velocity variable) registers the current velocity of the motor in steps per second.

*User defined variables* The VA instruction allows the user to create a 2 character name to a user defined variable (32 bit value).

The restrictions for this command are:

1) A variable cannot be named after an MCode Instruction, Variable or Flag.
2) The first character must be alpha, the second character may be alphanumeric.
3) A variable is limited to two characters.

With these the user can define a variable to store and retrieve data and perform math functions. When the FD (Factory Defaults) instruction is given, these variables will be deleted! There are two types of user defined variables:

- **Global variables:** global variables are variables that are defined outside of a program. The benefit to using a global variable is that no user program memory is required. For example, the user can define a variable called SP for speed by entering VA SP into the terminal. The user can then set that variable equal to the value of a read only variable V (velocity) by entering SP = V into the terminal.
- **Local variables:** this type of user defined variable is defined within a program and can only affect events within that program. It is stored in RAM. Note a local variable is not static, but is erased and declared again each time a program is executed.

### 3.2.3 Flags

Flags show the status of an event or condition. A flag will only have one of two possible states: either 1 or 0. Unlike variables, there are only factory defined flags.

*Factory defined flags*

Factory defined flags are predefined at the factory and cannot be deleted. When a FD (Factory Defaults) instruction is given, these flags will be returned to their factory default state. There are two types of factory defined flags:

- **Read/Writable:** This type of flag is user alterable. They are typically used to set a condition or mode of operation for device. For example EE = 1 would enable encoder operation, or EE = 0 would disable the encoder functions.
- **Read Only:** Read Only flags cannot be changed by the user. They only give the status of an event or condition. Typically this type of flag would be used in a program in conjunction with the BR (Branch Instruction) to generate an if/then event based upon a condition. For example the following line of code in a program BR SP, MV = 0 would cause a program to branch to a subroutine named “SP” when the MV, the read only moving flag, is false.

### 3.2.4 Keywords

Keywords are used in conjunction with the PR and IP instructions to indicate or control variables and flags. For instance, PR UV would print the state of all the user-defined variables to the screen. IP would restore all the factory variables from the NVM.

### 3.2.5 Math functions

Math functions are used to perform various arithmetic functions on numeric data stored in registers or variables. Supported functions are +, -, x, ÷, >, <, =, <=, >=, <>, AND, OR, XOR, NOT.

*Example*

Addition .....K2†=P+R2

Subtraction.....K3†=R1-P

Multiplication ..... A=A*2

Division ..... A=A/2

†User-defined variable used as an example.

## 3.3 Program structuring

Proper structuring of your MCode program will ensure your ability to work efficiently and will aid in trouble shooting your program. The figure below illustrates how your program can be blocked out to group the global system declarations, the main program body and the subroutines.

```
'System configuration
S1=0,0
Ms=256
A=1000000
D=A
Hc=2
Rc=75
P=0
```

**Block 1: System configuration**
Contains the variables and flag that define the operating parameters of the device during the execution of the program.

```
'Main program
PG 1
  LB aa
   P=0
  LB a1
   CL bb, I1-1
   H 10
   BR a1
```

**Block 2: Main program**
The first line places the device in program mode starting at adress 1. This will contain the main program label, as well as trigger events which will call subroutines to perform some action.

```
'Subroutine from
"trigger event
 LB bb
   MA 51200
   H
   MA 0
   H
   RT
```

**Block 3: Subroutines**
Subroutines are called from the main program block based upon a condirion such as an input state, a trip, or a numeric value stored in a register orvariable. They may also be called unconditionally.

```
E
PG
```

**Block 4: End program**
E - designates the end of the program
PG - exits program mode

Figure 3.1 Recommended MCode program structure

### 3.3.1 Programming aids

*IMS Terminal*

One of the most powerful tools available to you is the IMS Terminal software. IMS Terminal is an integrated Terminal and Program Text editor. The text editor will automatically color-code the Variables, Flags and Instructions that make up your program, as well as automatically indent program blocks for easy visual identification of your program elements.

The IMS Terminal also has several features that assist in program troubleshooting.

For more information on IMS Terminal see Section 6 Installing and Using IMS Terminal.

*User Labels*

The MCode programming language allows for 192 user labels for your programs, subroutines, and user variables and flags. A label consists of 2 characters, the first of which must be a letter, the second may be alphanumeric. A label cannot use the same character combination as any of the mnemonics used in the MCode programming language.

For purpose of this manual we have used the following example labels because the beginning alpha character is not used in any instruction set mnemonic:

Program label (G).....Example: G1, G8, Ga

Subroutine label (K) ..... Example: K7, K2, Ks

User variable label (Q) ..... Example: Q3, Q9, Qz

**Example labeling**

```
VA Q1              'Create user variable Q1

PG 100     'Enter Program mode
 LB G1     'Label Program G1
 CL K1, I2=1       'Call Subroutine K1 if Input 2 is HIGH
 BR G1     'Unconditional Branch to G1

K1         'Declare Subroutine K1
```

*Comments*

MCode allows for comments to be inserted in your program code. The comment character for the MCode language is the Apostrophe (‘). The device will ignore the text string following the apostrophe. Please note that the maximum length of a single line of program code is 64 characters, this includes program text, spaces and comments.

Using comments will be of assistance in trouble shooting your program.

*Programming reference*

Another powerful tool is this manual. Section 3 contains detailed explanations and usage examples of each mnemonic in the MCode Programming Language. In Section 8 there are a number of fully commented example programs that can be used to learn the basics of programming and using the various functions of your MCode compatible device.

## 3.4 Commonly used variables and instructions

### 3.4.1 Variables

*MS (Microstep resolution)*

MS (Microsteps Select) defines the resolution of the stepping motor.

- A motor rotates 1.8° per step or 200 steps per revolution.
- The MS selection divides the number of steps to yield a finer resolution.
- An MS value of 256 x 200 would yield 51200 microsteps per revolution. (Each Motor step will be divided into 256 Microsteps.)
- The MS default is 256.
- To read the MS value, type PR MS and press enter
- To write the MS value, type MS=<number> and press enter
- As we continue you will see that all motion variables use this value.

*P (Position)*

P indicates the position in either steps or encoder counts depending upon the enable/disable state of encoder functions.

- P takes its reading from C1 (Counter 1) when encoder functions are disabled. The reading is taken from C2 (Counter 2) when encoder functions are enabled.
- To read the position, type PR P or PR C1/C2 then press enter
- To zero the position, type P=0 then press enter

*VI (Initial velocity)*

Initial velocity in steps per second. (step size is a function of the value of MS).

- To read the initial velocity, type PR VI then press enter
- To write to the Initial velocity, type VI=<number> then press enter
- The VI default is 1000

*VM (Maximum velocity)*

Maximum or final velocity in steps per second. (Step size is a function of the value of ms).

- To read the final velocity, type PR VM then press enter
- To write to the final velocity, type VM=<number> then press enter
- The default VM Value is 768000

*A (Acceleration)* Acceleration in steps per second2. (Steps per second, per second.)

- The velocity of the motor will increase by the value of the acceleration rate every second until it reaches the programmed velocity in SL mode or it reaches VM.
- To read the acceleration, type PR A then press enter
- To write to the acceleration, type A=<number> then press enter
- The Acceleration Default value is 1000000

*D (Deceleration)* Deceleration in steps per second2. (Steps per second, per second.)

- The velocity of the motor will decrease by the value of the deceleration rate every second until it reaches the programmed velocity in SL mode or it reaches VI.
- To read the deceleration, type PR D then press enter
- To write to the deceleration, type D=<number> then press enter
- The deceleration default value is 1000000

### 3.4.2 Motion instructions

Motion instructions are those that cause the motors to move or affect the movement of the motor. There are a few factors that must be considered when programming motion commands. Linear distances, number of revolutions, degrees of rotation and timed moves can be calculated and programmed from these factors.

- All motion is programmed either microsteps per second or (when the encoder is enabled) encoder counts (pulses) per second.
- All motion is directly affected by the motion command and the program variables.
- There are a number of factors impacting motion instructions. These are addressed in detail in Section 7: Application and programming notes.

*MA (Move absolute)* Move to an absolute position relative to a defined zero position.

For example, type the following commands followed by pressing enter:

```
P=0        'set the current position to 0 (zero)
MA 20000   'move 20000 steps from 0 in the plus direction
PR P       'the terminal screen will read 20000
MA 3000    'move 3000 steps from 0 in the plus direction
PR P       'the terminal screen will read 3000
```

Absolute moves are always relative to 0 (zero).

You may program moves in the minus direction by typing the minus sign (-) before the value.

*MR (Move relative)* Move the number of steps programmed relative to current position.

For example, type the following commands followed by pressing enter:

```
P=0         'set the current position to 0 (zero)
MR 20000    'move 20000 steps from the current position in
            'the plus direction
PR P        'the terminal screen will read 20000
MR 3000     'move 3000 steps from the current position in
            'the plus direction
PR P        'notice the position read is 23000 and not 300
                                                          0
```

Relative moves are cumulative and are either added to or subtracted from the current position.

You may program moves in the minus direction by typing the minus sign (-) before the value.

*SL (Slew axis)* Move at a constant velocity.

```
SL 200000   'the motor moves at a constant velocity 200000
            'steps per second
```

- The slew command overrides the VM (maximum velocity) parameter.
- The value of the slew command may be changed “on the fly”.
- You may program moves in the minus direction by typing the minus sign (-) before the value.

*H (Hold)* An H (hold command) should typically follow any MA or MR commands in a program so that program execution is suspended until the motion is complete.

Below is a usage example.

```
PG 100      'enter program mode at address 100
LB M1               'label program M1
MR 20000    'set mode to relative, move relative 20000
steps
H           'hold until motion completes
MR -20000   'move relative -20000 steps
H           'hold until motion completes
E           'end program
PG          'exit program mode
```

A delay time value (1 to 65000 milliseconds) may be programed with the hold command.

(Note: There are circumstances where you may not want to hold up program execution.)

### 3.4.3 I/O instructions

Not all I/O instructions appply to all MCode device models. See section 4: command details for device compatability.

*S<1-4> <9-12> (Set I/O Point)*

This command configures the Type and Active state of I/O points 1-4.

- Using the PR command to read I/O parameters
- Read I/O1 Setup – “PR S1”
- Read I/O2 Setup – “PR S2”

Set IO 3 parameters – “S3=0,1” Sets IO3 as a General Purpose Input, Active High

For example: To set I/O4 as a jog+ input, active low, sinking

```
S4 =7,0,0
```

*I<1-4><9-12> (Read input state)*

Used to read the state of an individual input.

PR I1 will read the state of input 1 and display it to the terminal window.

BR K5, I2=0 will branch to the program address labled K5 when Input 2 is LOW

*IN (Read all inputs as decimal)*

Used to read the decimal equivalent of the 8 bit binary number represented by all 8 inputs collectively. Note the Input 12 is the Most Significant Bit.

PR IN will print the decimal value of the inputs.

IN will only read inputs 1-4 on devices equipped with only the standard I/O Set!

*IL (Read inputs 1-4 as decimal)*

Used to read the decimal equivalent of the 4 bit binary number represented by inputs 1 - 4 collectively. Note the input 4 is the most significant bit.

Pr il will print the decimal value of the standard input set.

*IH (Read inputs 9-12 as decimal)*

Used to read the decimal equivalent of the 4 bit binary number represented by inputs 9 - 12 collectively. Note the input 12 is the most significant bit.

PR IH will print the decimal value of the enhanced input set.

*0<1-4><9-12> (Set output state)*

Used to set the state of an output.

O2=1 will set Output 2 TRUE

O<9-12> are applicable to the enhanced I/O models only!

*OT (Set all outputs as BCD)*

Used to set the 8 bit binary equivalent of the decimal number represented by all 8 outputs collectively. Note the output 12 is the most significant bit.

OT=214 will set the outputs to 11010110

OT will only set outputs 1-4 on devices equipped with only the standard I/O Set!

*OL (Set outputs 1-4 as BCD)*

Used to set the 4 bit binary equivalent of the decimal number represented by outputs 1 - 4 collectively. Note the output 4 is the most significant bit.

OT=10 will set the standard outputs to 1010.

*OH (Set outputs 9-12 as BCD)*

Used to set the 4 bit binary equivalent of the decimal number represented by outputs 1 - 4 collectively. Note the output 12 is the most significant bit.

Ot=13 will set the enhanced outputs to 1101.

### 3.4.4 System instructions

The following system instructions will be used frequently.

*CP (Clear program memory)*

The CP instruction is used to clear program memory space.

*FD (Restore facory defaults)*

The FD instruction is used to return the device to its factory default state.

*<esc> (Stop motion and program)*

<esc> The ESCAPE key will stop the user program and stop the motor with no decel rate.

*<CTRL+C > (Reboot)*

<CTRL+C> CTRL+C will reboot the unit. This includes reloading of the programs stored in nonvolatile memory into RAM and executing any programs residing at label SU (Start Up).

### 3.4.5 Program instructions

*PG (Begin program mode)*

This instruction toggles the device into or out of program mode.

```
PG 200     'Switch to program mode at address 200
 xxxxx     'Program starting at address 200
 xxxxx     '    |
 xxxxx     '    |
PG         'Switch out of program mode
```

*LB (Lablel program, subroutine or branch process)*

MCode also offers the user the convenience of naming programs, subroutines and processes to ease in branching from one part of a program to another, or calling a subroutine.

These labels, once set, will act as pointers to locations in program memory space.

The LB, or label instruction, allows the user to assign a 2 character name to a program or branch process within a program or subroutine.

The restrictions for this command are:

1) A label cannot be named after an instruction, variable or flag.
2) The first character must be alpha, the second character may be alpha-numeric.
3) A label is limited to to characters.
4) A program labeled SU will run on power-up

Please Note: Any program labeled “SU” will execute on power-up.

```
PG 200     'Switch to program mode at address 200
LB k1             'Label command will name the program K1
 xxxxx     'Program named by LB command  xxxxx
 xxxxx     '
PG         'Switch out of program mode
```

*BR (Branch)*

Used to branch conditionally or unconditionally to a routine.

```
PG 200     'Switch to program mode at address 200
LB K1             'Label command will name the program
 xxxxx
 xxxxx     'Program named by LB command
 xxxxx
 BR K1     'Unconditional branch to Program Label K1
PG         'Switch out of program mode
```

*E (End program)*

Designates the end of a program.

```
PG 200     'Switch to program mode at address 200
LB K1             'Label command will name the program
 xxxxx
 xxxxx     'Program named by LB command
 xxxxx
 BR K1     'Unconditional branch to Program Label K1
```

```
E          'End Program
PG         'Switch out of program mode
```

*H (Hold program execution)* Delays program execution in milliseconds.

```
PG 200     'Switch to program mode at address 200
LB K1              'Label command will name the program
 xxxxx
 xxxxx     'Program named by LB command
 xxxxx
 H 2000    'Hold 2 seconds before execution of program
 BR K1     'Unconditional branch to Program Label K1
E          'End Program
PG         'Switch out of program mode
```

*PR (Print)* Outputs specified text and parameter values to a terminal or terminal software on a host PC.

```
PG 200             'Switch to program mode at address 200
LB K1              'Label command will name the program
 xxxxx
 xxxxx             'Program named by LB command
 xxxxx
 H 2000            'Hold 2 seconds before execution.
 PR "Position =", P       'Print position
 BR K1             'Uncond branch to Program Label K1
E                  'End Program
PG                 'Switch out of program mode
```

*VA (Create user variable)* Command used to define a user variable consisting of 2 alphanumeric characters.

```
PG 200             'Switch to program mode at address 200
VA N1                     'Define user variable N1
LB K1                     'Label command will name the
program
 xxxxx
 xxxxx             'Program named by LB command
 xxxxx
 H 2000            'Hold 2 seconds before execution
 PR "Position =", P       'Print position
 BR K1, N1<10      'Cond branch to K1 if N1 less than 10
E                  'End Program
PG                 'Switch out of program mode
```

# 4 Command summary

MCode supports multiple families of motion control devices. Not all instructions, variables and flags apply to all motion control products.

See the detailed descriptions for each command in Section 5 of this document for device compatibility

## 4.1 Setup instructions, variables and flags

| Mnemonic | Function | Unit | Range | Syntax Example |
|---|---|---|---|---|
| AL | Retrieve All Parameters | – | – | PR AL |
| BD | Communications BAUD Rate | BAUD | 48, 96, 19, 38, 11 | BD=<baud> |
| CE | Control C Software Reset | – | – | CE=<0/1> |
| CK | Check Sum Enable | – | – | CK=<1/0> |
| CM | Clock Mode Enable | – | 1/0 | CM=<1/0> |
| CR | Clock Mode Ratio | – | – | CR=1 |
| CW | Clock Mode Output Step Width | – | 0 to 255 | CW=100 |
| DE | Enable/Disable Drive | – | 1/0 | DE=<1/0> |
| DG | Disable Global Response | – | 1/0 | DG=<1/0> |
| DN | Device Name | Character | a–z, A–Z, 0–9 | DN=<char> |
| EM | Echo Mode 0 (def)=Full Duplex, 1=Half Duplex | Mode | <0 to 3> | EM=<mode> |
| FD | Restore Factory Defaults | – | – | FD |
| IP | Initial Parameters from NVM | – | – | IP |
| PY | Enable/Disable Party Mode | Mode | 1/0 | PY=<mode> |
| QD | Queued | – | 1/0 | QD=<1/0> |
| UG | Upgrade Firmware | Code | 2956102 | IMS Term. Upgrader |
| WT | Warning Temperature | Degrees C | 0 to 84 | WT=<0 – 84> |

## 4.2 Misc instructions, variables and flags

| Mnemonic | Function | Unit | Range | Syntax Example |
|---|---|---|---|---|
| AL | All Parameters, Used with PR (Print) | – | – | PR AL |
| BY | BSY Flag 1=Prog. Running | – | 0/1 | PR BY |
| CM | Clock Mode | — | 0/1 | CM=1 |
| CR | Clock Mode Ratio | – | – | CR=1 |
| CW | Clock Mode Output Step Width | – | 0 to 255 | CW=100 |
| EF | Error Flag | – | 0/1 | PR EF |
| ER | Error Number Variable | Number | – | PR ER |
| ES | Escape | – | – | ES=0 |
| FD | Return to Factory Defaults | – | – | FD |
| IF | Input Variable Pending Flag | – | – | PR IF |
| IT | Internal Temperature | Degrees Celsius | -55°C to 125°C | PR IT |
| IV | Input Into Variable | Number | – | IV <var> |
| PN | Part Number | – | – | PR “Position=” |
| PR | Print Selected Data and/or Text | – | – | PR <data/text string> |
| R1 | User Register 1 | Number | Signed 32 bit | R1=<number> |
| R2 | User Register 2 | Number | Signed 32 bit | R2=<number> |
| R3 | User Register 3 | Number | Signed 32 bit | R3=<number> |
| R4 | User Register 4 | Number | Signed 32 bit | R4=<number> |
| SN | Serial Number | – | – | PR SN |
| VR | Firmware Version | Number | – | PR VR |
| UV | Read User Variables | – | – | PR UV |

## 4.3 Motion instructions, variables and flags

| Mnemonic | Function | Unit | Range | Syntax Example |
|---|---|---|---|---|
| (–) | Do Previously Set Mode to/at This Value | per mode | | –<number> |
| A | Set Acceleration | Steps/Sec2 | 1000000000 | A=<accel> |
| D | Set Deceleration | Steps/Sec2 | 1000000000 | D=<decel> |
| HC | Set Hold Current | % (Percent) | 0 to 100 | HC=<percent> |
| HT | Set Hold Current Delay Time | milliseconds | 0–65000 | HT=<msec> |
| JE | Jog Enable Flag | – | 0/1 | JE<0/1> |
| LM | Limit Stop Mode | – | 1–6 | LM=<number> |
| MA | Set Mode and Move to Abs. Position | ±Position | Signed 32 bit | MA <±pos> |
| MD | Motion Mode Setting | – | – | – |
| MR | Set Mode and Move to Relative Position | ±Distance | Signed 32 bit | MR <±dist> |
| MS | Set Microstep Resolution | Microsteps/ step | MSEL Table | MS=<param> |
| MT | Motor Settling Delay Time | milliseconds | 0–65000 | MT=<msec> |
| MV | Moving Flag | – | – | PR MV |
| NE | Numeric Enable | – | 0/1 | NE<0/1> |
| RC | Set Run Current | % (Percent) | 1 to 100 | RC=<percent> |
| SL | Set Mode and Slew Axis | Steps/sec | ±5000000 | SL <velocity> |
| V | Read Current Velocity | Steps/sec | ±5000000 | PR V |
| VC | Velocity Changing Flag | – | – | BR<addr>, VC |
| VI | Set Initial Velocity | Steps/sec | 1–5000000 | VI=<velocity> |
| VM | Set Maximum Velocity | Steps/sec | 1–5000000 | VM=<velocity> |

## 4.4 Program instructions, variables and flags

| Mnemonic | Function | Unit | Range | Syntax Example |
|---|---|---|---|---|
| BP | Break Point | – | – | BP <addr>, <mode> |
| BR | Branch (Conditional/Unconditional) | – | – | BR <addr>, <cond> |
| BY | Program Executing | – | – | PR BY |
| CL | Call Subroutine (Conditional/ Unconditional) | – | – | CL <addr>, <cond> |
| CP | Clear Program | Address | 1-767 | CP <addr> |
| DC | Decrement Variable | – | – | DC <var/ureg> |
| E | End Program Execution | – | – | E |
| ES | Set Escape Mode | – | 0 – 3 | ES=<0–3> |
| EX | Execute Program at Address Using Selected Trace Mode | 1 to 767 | | EX <addr>, <mode> |
| H | Hold Prog. Execution Blank/0=Motion stops | milliseconds | Blank(0) 1-65000 | H=<msec> |
| IC | Increment Variable | – | – | IC <var> |
| L | List Program | Address | 1-767 | L <addr> |
| LB | Create a Program Address Label Name | | | |
| LK | Lock User Program | | 0/1 | LK=<0/1> |
| OE | On Error Handler 0=Disabled | Address | 0/1-767 | OE <addr> |
| PG | Start Program Entry at Specified Address | – | Blank/1-767 | PG <addr> |
| PS | Pause Program Execution | – | – | PS |
| RT | Return from Subroutine | – | – | RT |
| RS | Resume Paused Program | – | – | RS |
| S | Save to NVM | – | – | S |
| SU | Start Program on Power On | – | – | LB SU |
| VA | Create A User Variable Name | | | Mnemonic |

## 4.5 I/O instructions, variables and flags

| Mnemonic | Function | Unit | Range | Syntax Example |
|---|---|---|---|---|
| D1 | Set Input 1 Digital Filtering | Milliseconds | 0–255 | D1=<time> |
| D2 | Set Input 2 Digital Filtering | Milliseconds | 0–255 | D2=<time> |
| D3 | Set Input 3 Digital Filtering | Milliseconds | 0–255 | D3=<time> |
| D4 | Set Input 4 Digital Filtering | Milliseconds | 0–255 | D4=<time> |
| D5 | Set Input 5 Digital Filtering | Milliseconds | 0–255 | D5=<time> |
| D9 - D12 | Input Switch Debounce | Milliseconds | 0–255 | D1=0 |
| FC | Filter Capture | – | – | FC=3 |
| FM | Filter Motion | – | – | FC=3 |
| I1 -I4 | Read Input 1-4 | – | 0/1 | PR Ix, BR Ix,<cond> |
| I5 | Read Input 5 (Analog) | – | 0–1024 | PR I5, BR I5,<cond> |
| I6 | Read Encoder Index Mark Low true | – | – | PR 16 |
| I9 - I12 | Read Input | – | – | PR 12 |
| IL | Read Inputs 1–4 as One Value | data | 0–15 | PR IL |
| IH | Read Inputs 9 -12 as One Value | data | 0–15 | PR IH |
| IN | Read Inputs 1–4 and 9-12 as One Value | data | 0–255 | PR IN |
| IT | Internal Temperature | Degrees Celsius | -55°C to 125°C | PR IT |
| O1-O4 | Set Output x to Logic State | – | 0/1 | Ox=<1/0> |
| O9-O12 | Set Output x to Logic State | – | 0/1 | Ox=<1/0> |
| OL | Write Data to Outputs 1–4 as One Value | data | 0–15 | OL=<data> |
| OH | Write Data to Outputs 9–12 as One Value | data | 0–15 | OT=<data> |
| OT | Write Data to Outputs 1–4 and 9-12 as One Value | data | 0–255 | OT=<data> |
| S1-S4 | Setup IO Points 1-4 | Type, Active | Type Table, 0/1 | Sx=<type>,<active> |
| S9-S12 | Setup IO Points 9-12 | Type, Active | Type Table, 0/1 | Sx=<type>,<active> |
| S5 | Set/Print I/O Point 5 | – | 9 = 0 to +5 V / 10 = 4 to 20 mA | S5=<type> |
| S7 - S8 | Setup I/O Point Type/Active State | – | – | S7=34,0 |
| S9 - S12 | Setup I/O Point Type/Active State | – | – | S9=2,0 |
| S13 | Setup I/O Point Type/Active State | – | – | S13=60,0 |
| TI | Trip on Input | – | – | TI <input>,<addr> |
| TE | Trip Enable | See Table | <1–4> | TE=<num> |

## 4.6 Position related instructions, variables and flags

| Mnemonic | Function | Unit | Range | Syntax Example |
|---|---|---|---|---|
| C1 | Set Counter 1 | Motor Counts | Signed 32 bit | C1=<counts> |
| HM | Home to Home Switch | Type | 1 to 4 | HM <type> |
| P | Set/Read Position | Motor/Encoder Counts | Signed 32 bit | P=<counts> |
| PC | Read Captured Position at Trip | Motor/Encoder Counts | Signed 32 bit | PR PC |
| TP | Trip on Position | Position | – | TP <pos>, <addr> |
| TE | Trip Enable | See Table | <0-3> | TE=<num> |

## 4.7 Encoder related instructions, variables and flags

| Mnemonic | Function | Unit | Range | Syntax Example |
|---|---|---|---|---|
| C2 | Set Counter 2 | Encoder Counts | Signed 32 bit | C2=<counts> |
| DB | Set Encoder Deadband | Encoder Counts | 0-65000 | DB=<counts> |
| EE | Enable/Disable Encoder Functions | – | 1/0 | EE=<1/0> |
| HI | Home to Encoder Index | Type | 1 to 4 | HI=<type> |
| I6 | Read Encoder Index Mark | – | – | I6 |
| PM | Position Maintenance Enable Flag | | 0/1 | PM=<0/1> |
| SF | Set Stall Factor | Encoder Counts | 0-65000 | SF=<counts> |
| SM | Set Stall Mode | 0=Stop Motor/1=Don't Stop | 1/0 | SM=<mode> |
| ST | Stall Flag | – | 0/1 | PR ST |

## 4.8 Hybrid specific instructions, variables and flags

| Mnemonic | Function | Unit | Range | Syntax Example |
|---|---|---|---|---|
| AF | Read Hybrid Status | — | 128 bit binary | PR AF<br>BR <address>, AF=x<br>CL <address>, AF=x |
| AS | Set Hybrid mode | — | 0 = Off<br>1 = Fixed current (RC, HC)<br>2 = Variable current (RC max) (default)<br>3 = Torque | AS=2 |
| CA | Set calibration mode | — | 0 = Hard Stop Tolerant (fixed default)<br>Second Parameter: Torque compensation<br>0 = Off (default)<br>1 = On | CA=0,0 |
| CB | Set control bounds | — | 0 = 1.1 Full step (torque)<br>1 = 1.3 Full step (default)<br>2 = 1.5 Full step<br>3 = 1.7 Full step (speed) | CB=2 |
| CC | Set calibration current | % | 1 to 100 (100 default) | CC=25 |
| CF | Clear locked rotor fault | — | — | CF |
| CT | Set calibration time | mS | 2 to 65535 (200 default) | CT=2000 |
| EL | Remote encoder line count | Lines | — | EL=512 |
| LD | Set rotor lead limit | step | 0 to 2147483647 (102400 default) | LD=256000 |
| LL | Read position lead/lag limit | steps | -2147483647 to +2147483647 | PR LL<br>BR <address>, LL=x<br>CL <address>, LL=x |
| LG | Set rotor lag | step | 0 to 2147483647 (102400 default) | LG=256000 |
| LR | Locked rotor flag | — | 0/1 | |
| LT | Set locked rotor timeout | mS | 2 to 65535 (2000 default) | LT=1000 |
| MF | Set make-up frequency | Hz | 306 to 5000000 (768000 default) | MF=51200 |

| | | | | |
|---|---|---|---|---|
| MU | Set make-up mode | — | 0 = Off (default)<br>1 = Use MF (make-up freq)<br>2 = Use SS (system speed)<br>Clear parameter (not saved)<br>1 = Use LL<br>2 = Clear LL | MU=1,1 |
| SC | Start calibration/ configuration test | — | 0 = Calibration<br>1 - Configuration test | SC=1 |
| SS | Set system speed | Steps/ sec | 38,910 — 5,000,000 | SS=50000 |
| TA | Trip on Hybrid status | | 1 = Calibration done<br>2 = Hybrid active<br>4 = Locked rotor<br>8 = Lag limit<br>16 = Lead limit | TA=4,<address,label> |
| TD | Set torque direction | — | 0 = Minus<br>1 = Plus (default) | TD=1 |
| TQ | Set torque percent | % | 1 to 100 (25 default) | TQ-50 |
| TS | Set torque mode speed | Steps/ sec | 38,910 — 5,000,000 | TS=50000 |

## 4.9 Math functions

| Symbol | Function |
|---|---|
| + | Add Two Variables and/or Flags |
| - | Subtract Two Variables and/or Flags |
| * | Multiply Two Variables and/or Flags |
| / | Divide Two Variables and/or Flags |
| <> | Not Equal |
| = | Equal |
| < | Less Than |
| <= | Less Than and/or Equal |
| > | Greater Than |
| >= | Greater Than and/or Equal |
| & | AND (Bitwise) |
| \| | OR (Bitwise) |
| ^ | XOR (Bitwise) |
| ! | NOT (Bitwise) |

# 5 MCode command details

## A (Acceleration)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | A |
| **Function** | Set acceleration |
| **Type** | Variable |
| **Description** | The A Variable sets the acceleration rate when changing velocity in steps per second$^2$. If the A was set at 76800 per second$^2$ the motor would accelerate at a rate of 76800 counts per second, every second. If the maximum velocity was set at 768000 microsteps per second it would take 10 seconds to reach maximum speed if VI=0. |
| **Syntax** | A=<mode> |
| **Units** | Steps/Sec2 (EE=0)/Counts/Sec2 (EE=1) |
| **Range** | 91 to 1525878997 (Steps), 91 to 61035160 (Encoder Counts) |
| **Default** | Default: 1000000(EE=0), 40000 (EE=1) |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `A=20000 'Set Accel to 20000 steps/sec2`<br>`A=Q1 'Set Accel to user var Q1` |
| **Related** | D |

## AL (Retrieve all parameters)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | AL |
| **Function** | Retrieve all parameters |
| **Type** | Variable |
| **Description** | The AL variable is used with the PR (PRINT) instruction to print the value/state of all variables and flags to the terminal program. |
| **Syntax** | PR AL |
| **Usage** | Immediate, Read |
| **Code Example** | `PR AL` |

# AF (Hybrid flags)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

| | | |
|---|---|---|
| **Mnemonic** | AF | |
| **Function** | Read Hybrid status | |
| **Type** | Read-only status flag | |
| **Description** | The AF flag will print the status of the Hybrid logic. | |
| **Syntax** | PR AF<br>BR <address/label>, AF=<condition><br>CL <address/label>, AF=<condition> | |
| **Units** | — | |
| **Range** | 128-bit binary | |
| **Conditions** | 1 | Lead limit reached |
| | 2 | Lag limit reached |
| | 4 | Maximum lead/lag limit reached |
| | 8 | Locked rotor |
| | 16 | Hybrid mode is active |
| | 32 | Hardware fault condition exists |
| | 64 | At zero |
| | 128 | Calibration is complete |
| | 256 | Calibration fault |
| **Notes** | If multiple conditions exist the result is additive. i.e. At zero (64) and Calibration complete (128) AF=192 | |
| **Usage** | Program/Immediate, Read only | |
| **Code Example** | `PR AF  'print AF to the terminal`<br>`BR XY, AF=128 'branch to XY when calibration complete` | |
| **Related** | AS, TA | |

## AS (Hybrid mode)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

| | | |
|---|---|---|
| **Mnemonic** | AS | |
| **Function** | Set Hybrid operational mode | |
| **Type** | Variable | |
| **Description** | The AS variable will set the operating mode for the Hybrid motion control device to one of four modes, which are detailed below. These modes will determine the active state of the Hybrid circuitry and the behavior characteristics of the MDrive Hybrid based upon motor current or motor torque. | |
| **Syntax** | AS=<mode> | |
| **Units** | — | |
| **Range** | 0-3 | |
| **Modes** | 0 | Hybrid circuitry off. |
| | 1 | Fixed current mode, motor current will be as specified by the run current (RC) and hold current (HC) variables at the speed specified by the system speed variable (SS). |
| | 2 | Variable current mode, motor current will vary as needed to move/position the load with a maximum current level established by the run current (RC) variable.. |
| | 3 | Torque mode, motor torque and speed will vary as needed to move/position the load at the maximum torque specified by the set torque percent variable (TQ) at the maximum speed as specified by the set torque speed variable (TS). |
| **Default** | 2 (variable current mode) | |
| **Usage** | Program/Immediate, Read/Write | |
| **Code Example** | `AS=1  'set hybrid mode to fixed current` | |
| **Related** | RC, HC, HT, TD, TQ, TS | |

## AT (Reserved)

| | |
|---|---|
| **Mnemonic** | AT |
| **Function** | Reserved for internal use |
| **Description** | Reserved for internal use, will return an error if used.<br>Do not use as a user variable, flag or program label. |

# BD (Communications BAUD rate)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive Hybrid
MForce
MForce (Plus² expanded features)

| | | |
|---|---|---|
| **Mnemonic** | BD | |
| **Function** | Set communications BAUD rate | |
| **Type** | Variable | |
| **Description** | This variable sets the baud rate for serial communications with the MCode device. It sets the rate for the RS-422/485 interface. The baud rate is set by indicating the first two digits of the desired rate as shown in the range section below.<br>In order for the new BAUD rate to take effect, the user must issue the S (SAVE) instruction and then reset the device. When the MCode device is reset, it will communicate at the new BAUD rate. | |
| **Syntax** | BD=<mode> | |
| **Units** | Bits/second | |
| **Range** | 48, 96, 19, 38, 11 | |
| **Modes** | 48 | 4800 bps |
| | 96 | 9600 bps (default) |
| | 19 | 19200 bps |
| | 38 | 38400 bps |
| | **11** | 115200 bps - Not recommended for party mode communications configurations. |
| **Default** | 96 (9600 bps) | |
| **Usage** | Program/Immediate, Read/Write | |
| **Code Example** | `BD=19  'set BAUD rate to 19200 bps`<br>`S  'save to NVM` | |
| **Notes** | If you change the Baud Rate in the MCode device it must be matched in IMS Terminal.<br>A delay time between the command requests to the device must be considered to allow it time to interpret a command and answer the host before a subsequent command can be sent. The time between requests is dependent on the command and the corresponding response from the device. | |
| **Related** | CK | |

# BP (Break point)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | BP |
| **Function** | Set break point/execution mode |
| **Type** | Instruction |
| **Description** | The BP, or Break Point Instruction allows the user to set break points within an MCode program to help in debugging the program.<br><br>To use the BP instruction the program must be executed in either trace or single-step mode. The program will then run normally the number of times specified by the count, then go into single-step mode at the address or label specified by BP. Press the spacebar to step through the program if in single step mode.<br><br>To disable the break point, set BP=0. |
| **Syntax** | BP <address/label>,<count> |
| **Usage** | Program/Immediate |
| **Code Example** | `BP X1, 3  'Break  at label X1 after 3 cycles`<br><br>`EX P1, 1  'Execute  P1 in trace mode`<br>`or`<br>`EX P1,2  'Execute P1 in single step mode` |
| **Related** | `EX` |

# BR (Program branch)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | BR |
| **Function** | Conditional or unconditional program branch. |
| **Type** | Instruction |
| **Description** | The branch instruction can be used to perform a conditional or unconditional branch to a routine in a MCode device program. It can also be used to perform loops and IF THEN logic within a program.<br><br>There are two parameters to a branch instruction. These are used to perform two types of branches:<br><br>**Conditional Branch**<br><br>This type of branch first specifies an address or user label where program execution should continue if the second parameter, the condition, is true. The condition parameter may include flags as well as logical functions that are to be evaluated. Only one condition may exist.<br><br>**Unconditional Branch**<br><br>In this type of branch the second parameter is not specified, then the execution will continue at the label or address specified by the first parameter. |
| **Syntax** | BR <address/label>,<condition> |
| **Usage** | Program |
| **Code Example** | `BR 256, I2=1  'Cond. branch to address 256 'if input 1 = ACTIVE`<br>`BR G1  'Unconditional branch to 'program label G1`<br>`BR G2, Q4<10  'Cond branch to program G2 if 'user var Q4 is less than 10` |
| **Related** | EX |

# BY (Program executing)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | BY |
| **Function** | Busy (program executing) |
| **Type** | Status flag |
| **Description** | The BY flag will indicate the status of program execution, |
| **Syntax** | PR BY |
| **Response** | 0 - No programs running, 1 - program executing |
| **Default** | 0 |
| **Usage** | Immediate, Read only |
| **Code Example** | PR BY |
| **Related** | PR |

# C1 (Counter 1)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | C1 |
| **Function** | Counter 1 (position counter) |
| **Type** | Variable |
| **Description** | This variable contains the count of the clock pulses generated by the MCode compatible device. Counter 1 may be preset if necessary |
| **Syntax** | C1=<steps> |
| **Units** | Motor steps |
| **Range** | -2147483648 to +2147483647 |
| **Default** | 0 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `C1=20000 'Set counter 1 to 20000 motor steps`<br>`PR C1 'Print the value of C1 to the terminal screen`<br>`CL K5,C1>2100000 'Call subroutine K5 if C1>2100000` |
| **Related** | C2, P |

# C2 (Counter 2)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | C2 |
| **Function** | Counter 2 (encoder counter) |
| **Type** | Variable |
| **Description** | This variable contains the encoder edge count received by the MCode compatible device. Counter 1 may be preset if necessary |
| **Syntax** | C2=<counts> |
| **Units** | Encoder counts |
| **Range** | -2147483648 to +2147483647 |
| **Default** | 0 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `C2=512 'Set counter 2 to 512 counts`<br>`PR C2 'Print the value of C2 to the terminal screen`<br>`CL K5,C2>2100000 'Call subroutine K5 if C2>2100000` |
| **Related** | C1, P |

# CA (Calibration mode - HW ver 1.8+)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

ASI Firmware > 4.0.07
HW version 1.8+

**⚠CAUTION**

**MOVEMENT ON POWER APPLICATION**

Hybrid Motion Technology's functionality requires that the rotor and stator of the motor be in precise alignment. To accomplish this, the product will perform an initial calibration move upon power up consisting of a 6 motor full step (10.8°) move in the clockwise direction, followed by a 3 motor full step (5.4°) move inthe counter-clockwise direction.

- Only power the system if there are no persons in the hazardous area.

**Failure to follow these instructions can result in injury or equipment damage.**

| | | |
|---|---|---|
| **Mnemonic** | CA | |
| **Function** | Set calibration mode | |
| **Type** | Variable | |
| **Description** | Hybrid logic requires a calibration to set the initial relationship between the rotor and stator. A calibration is performed on power up to bring the rotor into physical alignment with the stator.<br>During calibration, the motor and position lag / lead logic is cleared, and any incoming steps are ignored.<br>Calibration occurs automatically these conditions:<br>• Power on<br>• Reset<br>• Hybrid functionality enabled<br>• Bridge is re-enabled after being disabled<br>• MSEL is changed.<br>• A Clear Locked Rotor )CF) command is issued. | |
| **Syntax** | CA=<0>,<method> | |
| **Units** | — | |
| **Range** | 0 | |
| **Modes** | 0 | Hard Stop Tolerant (HST) See detailed description |
| **Methods** | 0 | Torque compensation off |
| | 1 | Torque compensation on (not typically used for SSM mode). |
| **Default** | 0 (HST mode) | |
| **Usage** | Program/Immediate, Read/Write | |
| **Code Example** | `CA=0,1 'set torque compensation` | |
| **Notes** | For MDrive Hybrid HW versions earliuer thatn 1.8, please contact factory for support. | |
| **Related** | CC, CT, SC | |

*Detailed HST description*

The purpose of calibration is to bring the rotor and stator into alignment which allows operation of the Hybrid Motion logic. The Hard Stop Tolerant (HST) calibration technique seeks to ensure proper alignment when

the rotor is mechanically prohibited from movement by a hard stop or excessive stick/slip.

HST tests for freedom of movement by attempting a 6 full step (10.8 deg) move in the CW direction followed by a 3 full step (5.4 deg) move in the CCW direction. The sequence proceeds as follows.

Pause for CT(Calibration Time) allowing system to settle.

- Perform 6 full step CW move.
- Wait for CT.
- Perform 3 full step CCW move.
- Pause for CT allowing rotor to settle
- Set Calibration complete flag.

**If motion is blocked the following occurs:**

- If the 1st move can not complete it is abandoned.
- If the 2nd move can not complete it is abandoned then the sequence will be restarted one time.
- If neither move can be completed after 2 times through the sequence an error 109 will be generated, the calibration complete flag will set, and the bridges will be disabled (a locked rotor response).
- To correct, ensure that the motor shaft is free to in both directions. Torque compensation mode may be used for high inertia loads. Calibration current my be increased as well.

**Calibration time and distance required when the motor is free to move in both directions.**

- The time required typically is 3.12 seconds when CT is 250 mS. This value of CT gives the best compromise between time and proper calibration.
- Motor position will be advanced 3 full steps CW from initial position after calibration is complete. Motor position can be confirmed using the encoder counter C2. The step counter C1 does not count calibration steps

**Calibration time and distance when motor is blocked.**

- The time will vary significantly based on where in the motion and which motion is blocked. When the motor can not move at all in either direction the failure will typically indicate in 10 seconds.
- The motor position will also vary significantly based on which motion is effected and where within the motion.
- The motor position can be determined using the encoder counter C2. The step counter C1 does not count calibration steps.

**The following indicate when calibration is complete.**

- On power up, calibration is complete when the sign on message is sent.
- The flag register AF can be polled for the Calibration Complete flag.
- The trip on flag capability (TA to define, TE to enable) can be used also.
- The flags are useful after a manually initiated calibration.

**For mechanical systems with a large amount of prevailing torque a secondary calibration step is available, Torque Compensation (CA=0,1).**

- Th process measures and compensates for the prevailing torque.
- This typically adds 50 mS to the total calibration time and 1 encoder edge of movement

## CB (Control bounds)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

| | |
|---|---|
| **Mnemonic** | CB |
| **Function** | Set control bounds |
| **Type** | Variable |
| **Description** | There are four limits, or control bounds that can be selected. They are 1.1, 1.3, 1.5 and 1.7 full motor steps. Bounds of 1.1 will produce greater torque performance, though maximum speed will be reduced. Bounds of 1.7 will produce greater speed performance, though transient response is decreased. Best overall torque-speed performance is achieved with bounds set at 1.3 or 1.5. |
| **Syntax** | CB=<bounds> |
| **Units** | full motor step |
| **Range** | 0 — 3 |
| **Bounds** | 0 1.1 full motor steps — best torque performance<br>1 1.3 full motor steps<br>2 1.5 full motor steps — best overall performance<br>3 1.7 full motor steps — best speed performance |
| **Default** | 1 (1.3 full motor steps) |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `CB=2 'set control bounds to 1.5 steps` |
| **Notes** | For torque mode operation the bounds are preset. |
| **Related** | |

## CC (Calibration current)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

| | |
|---|---|
| **Mnemonic** | CC |
| **Function** | Set calibration current |
| **Type** | Variable |
| **Description** | |
| **Syntax** | CC=<percent> |
| **Units** | Percent |
| **Range** | 1 — 100 |
| **Default** | 100% |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `CC=25 'set calibration current to 25%` |
| **Related** | CA, CT |

## CE (Software reset enable)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | | |
|---|---|---|
| **Mnemonic** | CE | |
| **Function** | Software reset enable (CTRL+C) | |
| **Type** | Setup flag | |
| **Description** | This setup flag will configure the device to respond or not respond to a CTRL+C software reset. | |
| **Syntax** | CE=<mode> | |
| **Range** | 0 - 2 | |
| **Modes** | 0 | Disables CTRL+C response |
| | 1 | Enables CTRL+C response (default) |
| | 2 | Is addressable in party mode (PY=1), CTRL+C will respo0nd the same as CE=1 when not in party mode. |
| **Default** | 1 | |
| **Usage** | Program,/Immediate, Read/Write | |
| **Code Example** | `CE=0 'disable CTRL+C response` | |

## CF (Clear locked rotor fault)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

| | |
|---|---|
| **Mnemonic** | CF |
| **Function** | Clear locked rotor flault |
| **Type** | Instruction |
| **CCWDescription** | The CF instruction will clear a locked rotor fault, re-enable the output bridge and initiate a Hard Stop tolerant calibration, resulting in motion (10.8°then 5.4° CCW) |
| **Syntax** | CF |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `CF 'clear locked rotor fault` |
| **Related** | CA, CT, LR |

# CK (Checksum enable)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive Hybrid
MForce
MForce (Plus² expanded features)

| | | |
|---|---|---|
| **Mnemonic** | CK | |
| **Function** | Check sum enable | |
| **Type** | Flag | |
| **Description** | This setup flag will configure the device operate in check sum mode where a check sum is required following the command. | |
| **Syntax** | CK=<mode> | |
| **Range** | 0 - 2 | |
| **Modes** | 0 | Checksum mode disabled (default) |
| | 1 | Puts the device into Check Sum Mode. When enabled, all communications with the device require a Check Sum to follow the commands. The Check Sum is the 2's complement of the 7 bit sum of the ASCII value of all the characters in the command "OR"ed with 128 (hex = 0x80). The command will be acknowledged with a NAK (0x15) if the Check Sum is incorrect or an ACK (0x06) when the command is correctly processed (no error). |
| | 2 | CK=2 will enable check sum mode, however NAK only sent for bad check sum. "ACK" is not echoed if a program is running. Only a NAK is echoed if an error occurs. In immediate mode both ACK or NAK characters are echoed. |
| **Default** | 0 | |
| **Usage** | Program,/Immediate, Read/Write | |

**Code Example**

```
MR 1                          'MCode Command
77  82  32  49               'Decimal Value
4D  52  20  31                'Hex
77 + 82 + 32 + 49 = 240       'Add decimal values together
1111 0000  240                'Change 240 decimal to binary
0000 1111                     '1's complement
0001 0000                     'Add 1 (results in the 2's
1000 0000                     'complement)
1001 0000  144                'OR result with 128
                              'result Check Sum value in
                              'decimal
```

| | |
|---|---|
| **Notes** | To Send the checksum, in IMS terminal use ALT+ Checksum (In the example ALT+0144) The Response will be 06 |
| **Related** | BD |

## CL (Call subroutine)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | CL |
| **Function** | Call subroutine |
| **Type** | Instruction |
| **Description** | This function can be used to invoke a subroutine within a program. This allows the user to segment code and call a subroutine from a number of places rather than repeating code within a program.<br><br>There are two parameters to the CL instruction. The first specifies the program address or label of the subroutine to be invoked if the second parameter, the condition, is true. If the second parameter is not specified, the subroutine specified by the first parameter is always invoked. The condition parameter can include flags as well as logical functions that are to be evaluated. There can only be one condition.<br><br>The subroutine should end with a RT (Return) instruction. The RT instruction will cause program execution to return to the line following the CL instruction. |
| **Syntax** | CL=<address/label>,<condition> |
| **Usage** | Program,/Immediate, Read/Write |
| **Code Example** | `CL 256, I5<512  'Call Sub at 256, analog in < than 512`<br>`CL K5  'Unconditional call to subroutine label K5`<br>`CL K8, I4=0  'Call sub k8 if input 4 is INACTIVE` |
| **Notes** | The called subroutine should end with a return command (RT) |
| **Related** | RT |

## CM (Clock mode enable)

Compatible with Motion Control products:
MDrive (Plus² expanded features)
MForce (Plus² expanded features)

**⚠ WARNING**

If CM=1 (Following Mode Enabled) Limit functions will not operate.

To overcome this limits must either be handled by the controlling electronics or the MCode compatible device can be programmed to Trip On Input (See TI)

**Failure to follow these instructions can result in equipment damage.**

| | |
|---|---|
| **Mnemonic** | CM |
| **Function** | Clock mode enable |
| **Type** | Flag |
| **Description** | This flag (when CM=1) will enable the clock I/O 7 and 8. If S7 and S8 are set as Input type, the device will be in follower mode. If S7 and S8 are set as output types, signal will be motor step counts out. |
| **Syntax** | CM=<0/1> |
| **Range** | 0 - 1 |
| **Default** | 0 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `CM=1  'Enable Clock Mode` |
| **Related** | S7, S8, CR, CW, FM |

## CP (Clear program)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | CP |
| **Function** | Clear program |
| **Type** | Instruction |
| **Description** | This instruction will clear the program space in the NVM as specified by the instruction parameter. Programs are stored directly to the NVM and executed from there. Will clear program addresses only. Will not clear globally declared user variable or flags. FD will clear program memory as well. |
| **Syntax** | CP <address/label> |
| **Usage** | Immediate |
| **Code Example** | `CP 256  'Clear prog. space beginning at addr. 256`<br>`CP G3  'Clear program space beginning at label G3`<br>`CP  'Clear all of program space` |
| **Related** | FD, IP |

## CR (Clock ratio)

Compatible with Motion Control products:
MDrive (Plus² expanded features)
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | CR |
| **Function** | Clock ratio |
| **Type** | Variable |
| **Description** | Clock Ratio value for electronic gearing. The value selected will set the ratio from the clock input on I/O 7 (Step Clock) and I/O 8 (Direction) to the drive output. A Clock Ratio set to 0.50 will cause the frequency pulses sent to the driver section of the device to be 0.50 of the frequency input to the device, this would cause the device to “follow” at half the velocity of the primary axis. |
| **Syntax** | CR <ratio> |
| **Range** | 0.001 to 2.000 |
| **Default** | 1.000 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `CR=1.500  'Ratio set to 1.5` |
| **Notes** | A value of 1.00 has no acceleration or deceleration. All other values use the value of A and D.<br>The value of CR will have no impact on a clock output. CR will impact Step/Direction in, Clock Up/ Clock Down in and Quadrature in when I/O 7 and 8 are configured as inputs. |
| **Related** | S7, S8, CM, CW |

## CT (Calibration time)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

| | |
|---|---|
| **Mnemonic** | CT |
| **Function** | Set calibration time |
| **Type** | Variable |
| **Description** | The CT variable will set the calibration time in milliseconds. |
| **Syntax** | CT=<time> |
| **Units** | Milliseconds |
| **Range** | 2 — 65535 |
| **Default** | 200 mS |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `CT=1000 'set calibration time to 1S` |
| **Related** | CA, CC |

## CW (Clock width)

Compatible with Motion Control products:
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | CW |
| **Function** | Set clock width |
| **Type** | Variable |
| **Description** | When CM=1 and I/O 7 and 8 are configured as outputs, CW sets the pulse width of the output signal. The setting will be a multiplier x 50 nS. CW will set the output clock pulse width for: Step Clock out, Clock Up/Clock Down out, Quadrature out and the Trip output (I/O 13). |
| **Syntax** | CW=<time> |
| **Units** | Nanoseconds |
| **Range** | 0 - 255 |
| **Default** | 500 nS |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `CW=100 'output step width to 5µS (100 x 50nS)` |
| **Related** | S7, S8, S13, CM, PC |

# D (Deceleration)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | D |
| **Function** | Set deceleration |
| **Type** | Variable |
| **Description** | The D variable sets the deceleration of the device in steps per second². If the D was set at 76800 per second² the motor would decelerate at a rate of 76800 per second, every second. If the device was running at a maximum velocity of 768000 microsteps per second it would take 10 seconds to decelerate if VI=0. |
| **Syntax** | D=<steps/counts> |
| **Units** | Steps/Sec² (EE=0)/Counts/Sec² (EE=1) |
| **Range** | 91 to 1525878997 (Steps EE=0),<br>91 to 61035160 (Counts EE=1) |
| **Default** | 1000000(EE=0), 40000 (EE=1) |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `D=20000 'set acceleration to 20000 step/sec2`<br>`D=A 'set deceleration equal to acceleration` |
| **Related** | A, C1, C2, P |

# D1-D4 (Input switch debounce)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | D1-D4 |
| **Function** | Set input switch debounce |
| **Type** | Variable |
| **Description** | This variable will set the digital filtering to be applied to the selected input 1 - 4. The input must be stable for “time” amount of milliseconds before a change in state is detected. |
| **Syntax** | D1-4=<time> |
| **Units** | Milliseconds |
| **Range** | 0 to 255 |
| **Default** | 0 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `D1=0 'No debounce`<br>`D1=150 'Set filtering to 150 msec` |
| **Related** | I1 - I4 |

## D5 (Analog input filter)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | D5 |
| **Function** | Set analog input filter |
| **Type** | Variable |
| **Description** | D5 is a continuous filtering process. It does a running average by computing:<br>(((X-1)/X)*current reading) + (1 / X) If X = 10, then: ((current averaged value * 9)/10) + (new reading / 10) == NEW current averaged value. |
| **Syntax** | D5=<X> |
| **Units** | Counts |
| **Range** | 0 to 255 |
| **Default** | 0 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `D5=0 'No debounce`<br>`D5=100 'Set filtering to 100 counts` |

## D9-D12 (Input switch debounce)

Compatible with Motion Control products:
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | D9-D12 |
| **Function** | Set input switch debounce |
| **Type** | Variable |
| **Description** | This variable will set the digital filtering to be applied to the selected input 9 - 12. The input must be stable for “time” amount of milliseconds before a change in state is available. |
| **Syntax** | D9-12=<time> |
| **Units** | Milliseconds |
| **Range** | 0 to 255 |
| **Default** | 0 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `D10=0 'No debounce on input 10`<br>`D11=150 'Set filtering to 150 msec on input 11` |
| **Related** | I9 - I12 |

## DB (Encoder deadband)

Compatible with Motion Control products:
MDrive
MDrive (Ethernet)
MDrive (Plus² expanded features)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | DB |
| **Function** | Set encoder deadband |
| **Type** | Variable |
| **Description** | This variable defines the plus (+) and minus (-) length of the encoder deadband in encoder counts.<br>When the encoder is enabled, a move is not completed until motion stops within DB. If PM=1 device will correct if pushed outside of DB value once in position. |
| **Syntax** | DB=<counts> |
| **Units** | Encoder counts |
| **Range** | 0 to 65000 |
| **Default** | 1 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `DB=10  'Set encoder deadband to ±10 counts` |
| **Related** | EE, C2, SF, SM, ST, PM, EL |

## DC (Decrement variable)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | DC |
| **Function** | Decrement variable |
| **Type** | Instruction |
| **Description** | The DC instruction will decrement the specified variable by one. |
| **Syntax** | DC <variable> |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `DC R1  'Decrement register r1`<br>`DC K5  'Decrement user variable K5` |
| **Related** | IC |

## DE (Drive enable)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | | |
|---|---|---|
| **Mnemonic** | DE | |
| **Function** | Set drive enabled/disabled | |
| **Type** | Variable | |
| **Description** | The DE flag enables or disables the drive portion of the MCode compatible device. | |
| **Syntax** | DE=<0/1> | |
| **Range** | 0/1 | |
| **States** | 0 | Drive output bridge disabled. |
| | 1 | Drive output bridge enabled (default). |
| **Default** | 1 | |
| **Usage** | Program/Immediate, Read/Write | |
| **Code Example** | `DE=0 'Disable the motor driver section`<br>`DE=1 'Enable the motor driver section` | |

## DG (Disable global response)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive Hybrid
MForce
MForce (Plus² expanded features)

| | | |
|---|---|---|
| **Mnemonic** | DG | |
| **Function** | Enable/disable global response in party mode | |
| **Type** | Flag | |
| **Description** | The DG flag enables or disables device response to global commands made while in Party Mode. In the default state (DG=1) the device will not respond but will execute global commands. By setting the the DG flag to 0, that device will respond to global commands. | |
| **Syntax** | DG=<0/1> | |
| **Range** | 0/1 | |
| **States** | 0 | Response to global commands enabled. |
| | 1 | Response to global commands disabled (default). |
| **Default** | 1 (disabled) | |
| **Usage** | Program/Immediate, Read/Write | |
| **Code Example** | `DG=0 'Enable response to global commands`<br>`DG=1 'Disable Response to global commands no echo` | |
| **Related** | DN, PY | |

## DN (Device name)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive Hybrid
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | DN |
| **Function** | Set device name for party mode. |
| **Type** | Variable |
| **Description** | DN sets the name of the device for party mode communications. The acceptable range of characters is a-z, A-Z, 0-9. The factory default is "!" Once named, the device name must precede the instruction to that drive. When assigning a device name, the character MUST be within quotation marks.<br>The name is case sensitive. Refer to Section 2.2 in the Hardware Reference for specific Party Mode configuration and use instructions.<br>Once the default character has been changed it can not be reset except on a FD (Factory Default Reset). |
| **Syntax** | DN=<"ascii character"> |
| **Units** | ASCII Characters |
| **Range** | a-z, A-Z, 0-9 |
| **Default** | ! |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `DN="A"  'Set the device name to the character A`<br>`DN="65"  'Set the device name to the character A*` |
| **Notes** | Note: Must enter S (Save) prior to any reset of the DN will be lost. |
| **Related** | PY, S |

## E (End program)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | E |
| **Function** | End program |
| **Type** | Instruction |
| **Description** | Stops the execution of a program. Used in program mode to designate the end of the program |
| **Syntax** | E |
| **Usage** | Program |
| **Code Example** | `E  'End program` |
| **Related** | PG, EX |

## EE (Encoder enable)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | EE |
| **Function** | Enable/disable encoder |
| **Type** | Flag |
| **Description** | The EE flag enables or disables the optional encoder mode of the MCode compatible device. When in Encoder Mode, all moves are done by Encoder Counts. The 512 line Encoder generates counts in a Quadrature format which results in 2048 counts per revolution. |
| **Syntax** | EE=<0/1> |
| **Range** | 0/1 |
| **States** | 0 Disable the encoder (default state)<br>1 Enable the encoder |
| **Default** | 0 (disabled) |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `EE=0 'Disable the encoder`<br>`EE=1 'Enable encoder mode` |
| **Related** | DB, C2, SF, SM, ST, PM, FM, EL |

## EF (Error flag)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)\
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | EF |
| **Function** | Error condition exists |
| **Type** | Flag |
| **Description** | The Error flag will indicate whether or not an error condition exists. It is automatically cleared when a new program is executed. The only way to manually clear the EF flag is to read the value of the ER variable or set ER=0.<br>There is an instruction, OE, which allows the user to specify the execution of a subroutine in the program memory when an error occurs. The subroutine might contain instructions to read the ER variable which would clear the EF flag. |
| **Syntax** | PR EF |
| **Response** | 0/1 |
| **States** | 0 No error exists<br>1 Error condition exists |
| **Default** | 0 (disabled) |
| **Usage** | Program/Immediate, Read |
| **Code Example** | `PR EF 'Read the state of the error flag`<br>`'Response = 0: No error exists`<br>`'Response = 1: Error condition exists,`<br>`'Error value exists` |
| **Related** | ER, OE |

## EL (Remote encoder line count)

Compatible with Motion Control products:
MDrive (Plus² expanded features)
MDrive Hybrid
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | EL |
| **Function** | Set remote encoder line count. |
| **Type** | Variable |
| **Description** | This variable defines the number of encoder lines that the device will see in a revolution. Counter 2 will read 4 x EL, or 4 counts per line.<br>Note : The setting for MS (Microstep Resolution) is relative to the EL Setting. To calculate the minimum value for MS use the following equation:<br>MS minimum = (EL x 8) ÷ 200<br>Example for 1000 line encoder: 1000 X 8 = 8000, 8000 ÷ 200 = 40<br>Minimum Microstep Resolution = 50, or 10000 steps/rev. Note: We recommend leaving the Microstep Resolution at the default of 256. |
| **Syntax** | EL=<lines> |
| **Units** | Encoder lines |
| **Default** | 512 (1000 for MDrive Hybrid) |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `EL=1000 'Config EL variable for 1000 line encoder`<br>`MS=50 'Config microstep res to 10000 steps/rev` |
| **Related** | C2, MS, SF, SM, ST, PM, FM |

## EM (Echo mode)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | | |
|---|---|---|
| **Mnemonic** | EM | |
| **Function** | Set echo mode | |
| **Type** | Flag | |
| **Description** | The Echo Mode Flag will set the full/half duplex configuration of the RS-485 channel. | |
| **Syntax** | EM=<0-3> | |
| **Range** | 0 - 3 | |
| **States** | 0 | Echo all information back over communications line. CR/LF indicates command accepted (full duplex) (default). |
| | 1 | Don't echo the information, only send back prompt. CR/LF indicates command accepted (half duplex). |
| | **2** | Does not send prompt, only responds to PRINT (PR) and LIST (L) commands. |
| | 3 | Saves Echo in Print Queue then executes. Prints after command is terminated. |
| **Default** | 0 (disabled) | |
| **Usage** | Program/Immediate, Read | |
| **Code Example** | `EM=1 'Don't echo the information,` | |
| **Related** | ER, OE | |

## ER (Error)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | ER |
| **Function** | Display error code |
| **Type** | Variable |
| **Description** | The ER variable indicates the MCode device error code for the most recent error that has occurred. The ER variable must be read or set to zero to clear the EF flag.<br>A Question Mark <?> in place of the normal cursor indicates an ERROR. A list of Error codes are located at the end of this section.. |
| **Syntax** | PR ER |
| **Units** | Numeric error code |
| **Default** | 0 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `PR ER  'Read the error number, result = <Value>`<br>`ER 0  'Set the error value to zero` |
| **Notes** | Note: See Error code section for full listing of error codes. |
| **Related** | EF, OE |

## ES (Escape)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | | |
|---|---|---|
| **Mnemonic** | ES | |
| **Function** | Set escape mode | |
| **Type** | Flag | |
| **Description** | ESC flag to switch between ESC and CTRL+E. An Escape will stop both the program and the motion. | |
| **Syntax** | ES=<0-3> | |
| **Range** | 0 - 3 | |
| **Modes** | 0 | Escape Flag set to respond to CTRL+E |
| | 1 | Escape Flag set to respond to ESC keypress (default) |
| | **2** | Escape Flag set to respond addressable CTRL+E (party mode) |
| | 3 | Escape Flag set to respond to addressable ESC keypress (party mode) |
| **Default** | 1 | |
| **Usage** | Program/Immediate, Read/Write | |
| **Code Example** | `ESC=0  'ESC responds to CTRL+E` | |

# EX (Execute program)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | | |
|---|---|---|
| **Mnemonic** | EX | |
| **Function** | Execute program | |
| **Type** | Instruction | |
| **Description** | Execute program at a specified address or label using a selected trace mode. Used in immediate mode.<br>There are three modes of program execution. | |
| **Syntax** | EX <address/label>,<mode> | |
| **Modes** | 0 | Normal execution, is specified by a mode of 0 (or simply leaving the mode blank). |
| | 1 | Trace mode is specified by a mode of 1. This means that the program executes continuously until the program E is encountered, but the instructions are “traced” to the communications port so the user can see what instructions have been executed. |
| | **2** | Single step mode is specified by a mode of 2. In this mode, the user can step through the program using the space bar to execute the next line of the program. The program can be resumed at normal speed in this mode by pressing the enter key. |
| **Usage** | Immediate | |
| **Code Example** | `EX 1 'Execute program at address 1 normally`<br>`EX G2,1 'Execute program G2 in trace mode`<br>`EX 200,2 'Execute program at address 200 in 'single-step mode` | |

# FC (Filter capture)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | FC |
| **Function** | Set filtering for the capture input. |
| **Type** | Variable |
| **Description** | This variable will set the digital filtering to be applied to I/O 13 when configured as a Capture input. The input must be stable for "time" amount of milliseconds before a change in state is detected. |

**Input 13 Filter Capture Settings**

| Range | Min Pulse | Cutoff Frequency |
|---|---|---|
| 0 | 50 nS | 10 MHz |
| 1 | 150 nS | 3.3 MHz |
| 2 | 200 nS | 2.5 MHz |
| 3 | 300 nS | 1.67 MHz |
| 4 | 500 nS | 1.0 MHz |
| 5 | 900 nS | 555 kHz |
| 6 | 1.7 µS | 294.1 kHz |
| 7 | 3.3 µS | 151 kHz |
| 8 | 6.5 µS | 76.9 kHz |
| 9 | 12.9 µS | 38.8 kHz |

| | |
|---|---|
| **Syntax** | FC=<0-9> |
| **Range** | 0 to 9 |
| **Units** | Numeric |
| **Default** | 0 (min pulse 50 nS, cutoff of 10 MHz) |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `FC=2 'Set filtering for Input 13 min pulse to 200 'nS/CO 2.5 MHz` |
| **Related** | I13, TC, S13 |

## FD (Restore factory defaults)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | FD |
| **Function** | Restore factory default settings |
| **Type** | Instruction |
| **Description** | FD will clear all program memory and return the MCode compatible device to factory default settings. The response will be the sign on message. FD will clear programs and initialize parameters. |
| **Syntax** | FD <carriage return> |
| **Usage** | Immediate |
| **Code Example** | `FD 'Restore device to factory default state`<br>`Response "Copyright 2001-2007 by Intelligent Motion Systems, Inc."` |
| **Notes** | FD will generate an error 73 if the unit is in motion. |
| **Related** | CP, IP |

## FM (Filter motion)

Compatible with Motion Control products:
MDrive (Plus² expanded features)
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | FM |
| **Function** | Set filtering for the motion inputs. |
| **Type** | Variable |
| **Description** | Filter Motion inputs, I/O 7 & 8 for electronic gearing and optional encoder. |

**Input 7 & 8 Filter Settings**

| Range | Min Pulse | Cutoff Frequency |
|---|---|---|
| 0 | 50 nS | 10 MHz |
| 1 | 150 nS | 3.3 MHz |
| 2 | 200 nS | 2.5 MHz |
| 3 | 300 nS | 1.67 MHz |
| 4 | 500 nS | 1.0 MHz |
| 5 | 900 nS | 555 kHz |
| 6 | 1.7 µS | 294.1 kHz |
| 7 | 3.3 µS | 151 kHz |
| 8 | 6.5 µS | 76.9 kHz |
| 9 | 12.9 µS | 38.8 kHz |

| | |
|---|---|
| **Syntax** | FM=<0-9> |
| **Range** | 0 to 9 |
| **Units** | Numeric |
| **Default** | Default: 2 (min pulse 200 nS, cutoff of 2.5 MHz) |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `FM=3 'Set filtering for motion min pulse to 300 'nS/CO 1.67 MHz` |
| **Related** | I7, I8, C2, EL, EE, S7, S8 |

## FT (Internal use only)

| | |
|---|---|
| **Mnemonic** | FT |
| **Function** | Factory use only |
| **Description** | Will return an error if used. Do not use for a user program label, variable or flag.. |

## H (Hold program execution)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | H |
| **Function** | Hold program execution |
| **Type** | Instruction |
| **Description** | The hold instruction is used in a program to suspend program execution. If no parameter is specified the execution of the program will be suspended while motion is in progress. This will typically be used following a MA, MR, HI or HM instruction.<br>A time in milliseconds may be placed as a parameter to the hold instruction, This will suspend program execution for the specified number of milliseconds. |
| **Syntax** | H <time> |
| **Units** | Milliseconds |
| **Range** | 1 - 65000 |
| **Usage** | Program/Immediate |
| **Code Example** | `MA 1000  ' Move Absolute 1000 steps`<br>`H  'Suspend prog ex. until motion completes`<br>`'(used after a move command)`<br><br>`H 2000  'Suspend program execution for 2 seconds` |
| **Related** | PG, E, EX |

# HC (Motor holding current)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | HC |
| **Function** | Motor holding current |
| **Type** | Variable |
| **Description** | This variable defines the motor holding current in percent. |

| HC=(%) | MDrive MDrive Hybrid (All) | MForce MicroDrive (A RMS) | MForce PowerDrive (A RMS) |
|---|---|---|---|
| 10 | MDrive Range 0 To 100% | 0.3 | 0.5 |
| 20 | | 0.6 | 1.0 |
| 30 | | 0.9 | 1.5 |
| 40 | Actual Current Not required as Motor is appropriately sized to the device. | 1.2 | 2.0 |
| 50 | | 1.5 | 2.5 |
| 60 | | 1.8* | 3.0 |
| 70 | | 2.1 | 3.5 |
| 80 | | 2.4 | 4.0 |
| 90 | | 2.7 | 4.5 |
| 100 | | 3.0 | 5.0 |

| | |
|---|---|
| **Syntax** | HC=<percent> |
| **Units** | Percent |
| **Range** | 0 to 100 |
| **Default** | 5% |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `HC=25  'set holding current to 25%` |
| **Related** | RC, HT |

# HI (Home to index mark)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | | |
|---|---|---|
| **Mnemonic** | HI | |
| **Function** | Home to index mark | |
| **Type** | Instruction | |
| **Description** | This instruction will find the the encoder index mark. There are four combinations for this command. (See Use below.)<br>When HI is executed, the axis moves in the direction specified by the (S) at VM until it reaches the index mark. It then creeps off of the index in the direction specified by the sign of (C) at VI. Motion is stopped as soon as the index changes state. Note that Speed and Creep is set by the VM and VI commands.<br>1. Speed: Specifies the direction and speed that the axis will move until the switch is activated (VM).<br>2. Creep: Specifies the direction and speed that the axis will move off the switch until it becomes inactive again (VI). | |
| **Syntax** | HI <type> | |
| **Range** | 1 - 4 | |
| **Modes** | 1 | Slew at VM in the minus direction and Creep at VI in the plus direction. |
| | 2 | Slew at VM in the minus direction and Creep at VI in the minus direction. |
| | 3 | Slew at VM in the plus direction and Creep at VI in the minus direction. |
| | 4 | Slew at VM in the plus direction and Creep at VI in the plus direction. |
| **Usage** | Program/Immediate | |
| **Code Example** | `HI 2 'Slew at VM minus direction and Creep at VI in the 'minus direction` | |
| **Related** | VM, VI, EE, I6, HM | |

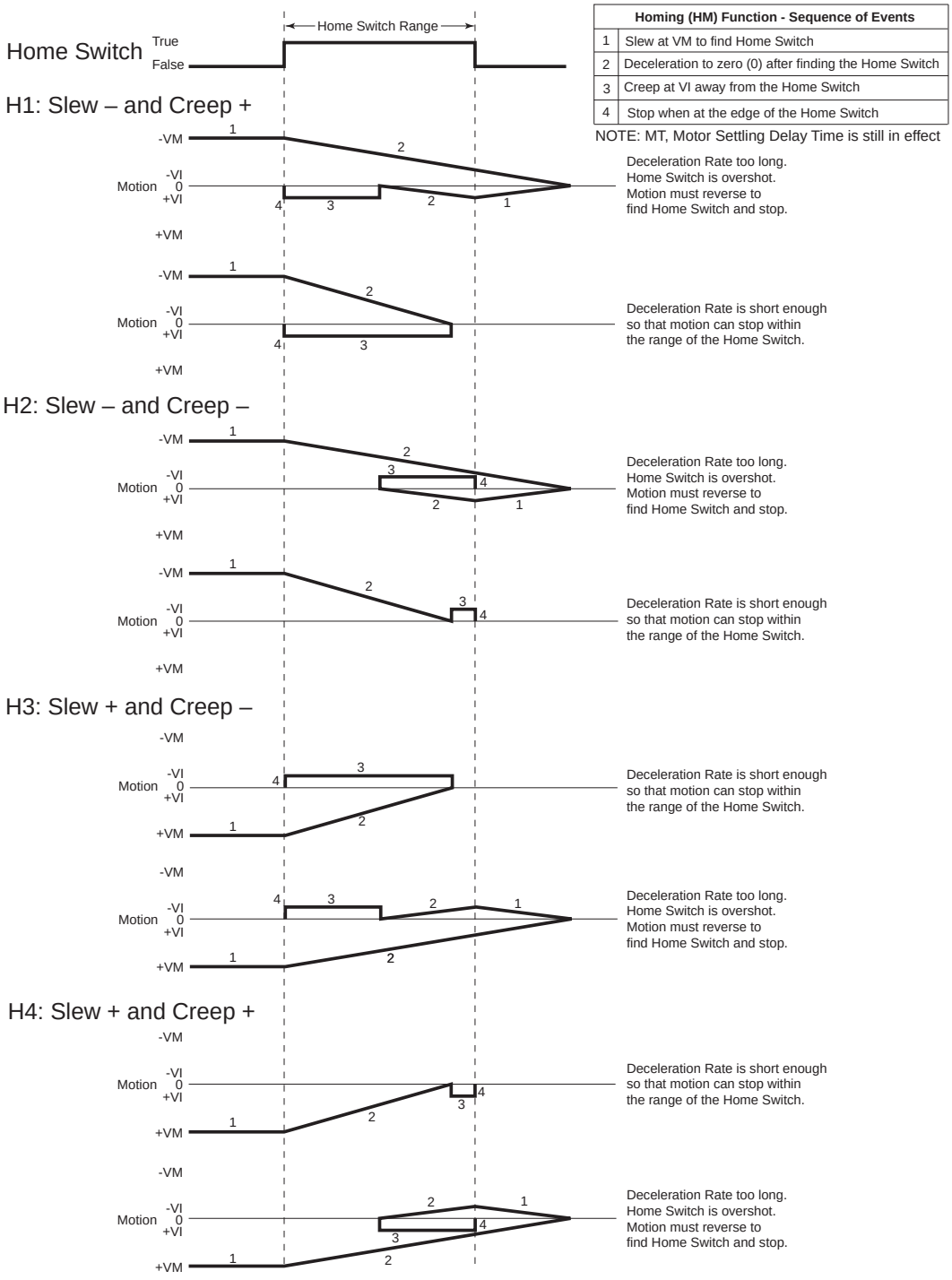
# HM (Home to home switch)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | HM |
| **Function** | Home to home switch |
| **Type** | Instruction |
| **Description** | This instruction will find the selected I/O switch assigned to “Home”.<br>1. Speed: Specifies the direction and speed that the axis will move until the switch is activated (VM).<br>2. Creep: Specifies the direction and speed that the axis will move off the switch until it becomes inactive again (VI).<br>When HM is executed, the axis moves at VM in the direction specified by the sign of speed. It then creeps off of the switch at VI in the direction specified by the sign of creep. Motion is stopped as soon as the switch becomes deactivated. Note that Speed and Creep is set by the VM and VI commands.<br>The diagram on the following page illustrates the different scenarios possible during the Homing (HM) sequence. The diagrams represent the four HM commands. Below are the four combinations of the HM command. |
| **Syntax** | HI <type> |
| **Range** | 1 - 4 |
| **Modes** | 1 Slew at VM in the minus direction and Creep at VI in the plus direction.<br>2 Slew at VM in the minus direction and Creep at VI in the minus direction.<br>3 Slew at VM in the plus direction and Creep at VI in the minus direction.<br>4 Slew at VM in the plus direction and Creep at VI in the plus direction. |
| **Usage** | Program/Immediate |
| **Code Example** | `HM 2 'Slew at VM minus direction and Creep at VI in the 'minus direction` |
| **Related** | VM, VI, EE, I6, HI, LM, S<1-4>, <9-12> |

The key to the diagrams is as follows.

1. Slew at VM to find the Index Mark.
2. Decelerate to zero (0) after finding the Index Mark.
3. Creep at VI away from the Index Mark.
4. Stop when at the edge of the Index Mark.




| Homing (HM) Function - Sequence of Events | |
|---|---|
| 1 | Slew at VM to find Home Switch |
| 2 | Deceleration to zero (0) after finding the Home Switch |
| 3 | Creep at VI away from the Home Switch |
| 4 | Stop when at the edge of the Home Switch |

NOTE: MT, Motor Settling Delay Time is still in effect

Figure 5.1 Homing functions sequence of events

# HT (Hold current delay time)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | HT |
| **Function** | Holding current delay time |
| **Type** | Variable |
| **Description** | The HT variable sets the delay time in milliseconds between the MV=0 and when the device shifts to the holding current level specified by the HC (Motor Holding Current) variable. The delay time is also effected by the MT (Motor Settling Delay Time) variable in that the total time to current change is represented by the sum of MT + HT. The total of MT+HT cannot add up to more than 65535, thus the value of MT is included in the HT range.<br>Thus the Maximum setting for HT=(65535-MT). ***If HT=0, the current will not reduce.*** |
| **Syntax** | HT=<time> |
| **Units** | Milliseconds |
| **Range** | 0 or 2 - 65535 |
| **Default** | 500 ms |
| **Usage** | Program/Immediate. Read/Write |
| **Code Example** | `HT=1500 'set hold current delay time to 1.5 seconds` |
| **Related** | HC, MT, RC |

# HT (Hold current delay time)

Compatible with Motion Control products:
MDrive Legacy

**⚠ CAUTION**

The HT command functions differently on legacy MDrive products than on current MDrive products in that if HT is set to 0, the current will reduce immediately.

**Failure to follow these instructions can result in equipment damage.**

| | |
|---|---|
| **Mnemonic** | HT |
| **Function** | Holding current delay time |
| **Type** | Variable |
| **Description** | The HT variable sets the delay time in milliseconds between the MV=0 and when the device shifts to the holding current level specified by the HC (Motor Holding Current) variable. The delay time is also effected by the MT (Motor Settling Delay Time) variable in that the total time to current change is represented by the sum of MT + HT. The total of MT+HT cannot add up to more than 65535, thus the value of MT is included in the HT range.<br>Thus the Maximum setting for HT=(65535-MT). ***If HT=0, the current will reduce. immediately.*** |
| **Syntax** | HT=<time> |
| **Units** | Milliseconds |
| **Range** | 0 or 2 - 65535 |
| **Default** | 500 ms |
| **Usage** | Program/Immediate. Read/Write |
| **Code Example** | `HT=1500 'set hold current delay time to 1.5 seconds` |
| **Related** | HC, MT, RC |

# I1, I2, I3 and I4 (Read inputs 1, 2 ,3 or 4)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | I1 - I4 |
| **Function** | Read the state of inputs 1 - 4 |
| **Type** | Variable |
| **Description** | This variable will read the state of the specified input 1 - 4. Can be used with PR (Print), BR (Branch) and CL (Call Subroutine) instructions. Can also be used with R1 - R4 and User Variables.<br>The value of the bit state will be dependant on active (low/high) state of the input, specified by the S<1-4> 2nd parameter. |
| **Syntax** | PR I<1-4><br>BR <addr/lbl>, I<1-4>=<1/0><br>CL<addr/lbl>, I<1-4>=<1/0> |
| **Units** | Logic 0 or 1 |
| **Range** | 0/1 |
| **Usage** | Program/Immediate. Read |
| **Code Example** | `PR I2 'Prints the logic state of input 2`<br>`BR 128,I3=1 'Cond branch to address 125, Input 3 ACTIVE`<br>`CL K9, I4=0 'Call subroutine K9, Input 4 INACTIVE` |
| **Related** | IL, IN, O1-O4, S1-S4 |

## I5 (Read analog input)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | I5 |
| **Function** | Read the value of the analog input |
| **Type** | Variable |
| **Description** | This variable will read the value of the correlating bit value seen on the Analog Input. Can be used with PR (Print), BR (Branch) and CL (Call Subroutine) instructions. The value read will between 0 and 1023.<br>This value represents a voltage or current being seen on the analog input. for example, if in current mode (0 - 20mA range) 1023 would represent 20 mA, 512 would represent 10 mA. |
| **Syntax** | PR I5<br>BR <addr/lbl>,I5=<0-1023><br>CL<addr/lbl>, I5<0-1023> |
| **Units** | Numeric value |
| **Range** | 0 to 1023 |
| **Usage** | Program/Immediate. Read |
| **Code Example** | `PR I5 'Print the value of I5`<br>`BR G3,I5>512 'Branch to prog G3 if I5 is > 512`<br>`CL 423,I5<220 'Call sub at address 423 if I5 is < 220` |
| **Related** | S5 |

## I6 (Read encoder index mark)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | I6 |
| **Function** | Read the state of the encoder index mark |
| **Type** | Variable |
| **Description** | This variable will read the on/off state of the Encoder Index Mark. Can be used with PR (Print), BR (Branch) and CL (Call Subroutine) instructions. The value read will be 0 (off mark) or 1 (on mark). |
| **Syntax** | PR I6<br>BR <addr/lbl>,I6=<0/1><br>CL<addr/lbl>, I6<0/1> |
| **Units** | Logic state 0 or 1 |
| **Range** | 0/1 |
| **Usage** | Program/Immediate. Read |
| **Code Example** | `PR I6 'Print state of the encoder index mark`<br>`BR 324,I6=1 'Branch to address 324 if I6 is ACTIVE`<br>`CL K3,I6=1 'Call subroutine K3 if I6 is ACTIVE` |
| **Related** | S5 |

## I7, I8 (Reserved)

| | |
|---|---|
| **Mnemonic** | I7, I8 |
| **Function** | Reserved, do not use as a user variable, flag or label. |

## I9, I10, I11 and I2 (Read inputs 9, 10, 11 or 12)

Compatible with Motion Control products:
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | I9 - I12 |
| **Function** | Read the state of inputs 9 - 12 |
| **Type** | Variable |
| **Description** | This variable will read the state of the specified input 9 - 12. Can be used with PR (Print), BR (Branch) and CL (Call Subroutine) instructions. Can also be used with R1 - R4 and User Variables.<br>The value of the bit state will be dependant on active (low/high) state of the input, specified by the S<9-12> 2nd parameter. |
| **Syntax** | PR I<9-12><br>BR <addr/lbl>, I<9-12>=<1/0><br>CL<addr/lbl>, I<9-12>=<1/0> |
| **Units** | Logic 0 or 1 |
| **Range** | 0/1 |
| **Usage** | Program/Immediate. Read |
| **Code Example** | `PR I12 'Prints the logic state of input 12`<br>`BR 128,I9=1 'Cond branch to address 125, Input 9 ACTIVE`<br>`CL K9, I8=0 'Call subroutine K9, Input 8 INACTIVE` |
| **Related** | IL, IN, O1-O4, S1-S4 |

## I13 (Reserved)

| | |
|---|---|
| **Mnemonic** | I13 |
| **Function** | Reserved, do not use as a user variable, flag or label. |

## IC (Increment variable)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | IC |
| **Function** | Increment variable |
| **Type** | Instruction |
| **Description** | The IC instruction will increment the specified variable by one. |
| **Syntax** | IC <variable> |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `IC R1 'Increment register r1`<br>`IC K5 'Increment user variable K5` |
| **Related** | DC |

## IF (Input variable pending)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | IF |
| **Function** | Input variable pending |
| **Type** | Flag |
| **Description** | The IF instruction is automatically set to 1 when IV command is executed. The IF flag reflects an input value from serial port is pending, not that one has been received. IF will be cleared to zero (0) with a carriage return or can be reset manually. |
| **Syntax** | <user input><CR><br>IF=<0/1> |
| **States** | 0 No variable input pending<br>1 Variable input pending |
| **Default** | 0 |
| **Usage** | Program/Immediate, Read |
| **Code Example** | `No Usage Example, Flag set automatically by IV` |

## IL (Read inputs 1 - 4 as a group)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | IL |
| **Function** | Read the state of inputs 1 - 4 as one value |
| **Type** | Variable |
| **Description** | This keyword will read the binary state of inputs 1-4 and print them as a decimal value. When used thus, Input 1 is the Least Significant Bit (LSb) and Input 4 is the Most Significant Bit (MSb). It may be used in conjunction with the R1-R4 (Registers), PR (Print), BR (Branch) and CL (Call Subroutine) instructions. The value is a function of the actual voltage level of the I/O where 1 = +V and 0 = Ground. (Not a function of the active state defined in S1 to S4 variables). |
| **Syntax** | PR IL<br>BR <addr/lbl>,IL=<0-15><br>CL<addr/lbl>, IL<0-15> |
| **Units** | Decimal |
| **Range** | 0 —15 |
| **Usage** | Program/Immediate. Read |
| **Code Example** | `PR IL ‘Print decimal value of IO4-IO1`<br>`BR 324,IL=8 ‘Branch to address 324 if IL=8`<br>`CL K3,IL=13 ‘Call subroutine K3 if IL=13` |
| **Related** | IH, IN, OL, OH, OT, S1-S4 |

## IH (Read inputs 9 - 12 as a group)

Compatible with Motion Control products:
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | IH |
| **Function** | Read the state of inputs 9 - 12 as one value |
| **Type** | Variable |
| **Description** | This keyword will read the binary state of inputs 9-12 and print them as a decimal value. When used thus, Input 9 is the Least Significant Bit (LSb) and Input 12 is the Most Significant Bit (MSb). It may be used in conjunction with the R1-R4 (Registers), PR (Print), BR (Branch) and CL (Call Subroutine) instructions.<br><br>The value is a function of the actual state of the I/O where 1 = +V and 0 = Ground. (Not a function of the active state defined in S9 to S12 variables) |
| **Syntax** | PR IH<br>BR <addr/lbl>,IH=<0-15><br>CL<addr/lbl>, IH<0-15> |
| **Units** | Decimal |
| **Range** | 0 —15 |
| **Usage** | Program/Immediate. Read |
| **Code Example** | `PR IH  'Print decimal value of IO12-IO9`<br>`BR 324,IH=8  'Branch to address 324 if IH=8`<br>`CL K3,IH=13  'Call subroutine K3 if IH=13` |
| **Related** | IL, IN, OL, OH, OT, S1-S4, S9-S12 |

## IN (Read inputs 1 - 4 and 9 - 12 as a group)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | IN |
| **Function** | Read the state of inputs 1 - 4 and 9 - 12 as one value |
| **Type** | Variable |
| **Description** | This keyword will read the binary state of inputs 1-4 and 9-12 and print them as a decimal value. When used thus, Input 1 is the Least Significant Bit (LSb) and Input 12 is the Most Significant Bit (MSb). It may be used in conjunction with PR (Print), BR (Branch) and CL (Call Subroutine) instructions.<br><br>The value is a function of the actual state of the IO where 1 = +V and 0 = Ground. (Not a function of the active state defined in S1 to S4 and S9 to S12 variables). On Standard Plus Models IN will only read the lower group (Inputs 1-4). The debounce function has no effect on the data read. |
| **Syntax** | PR IN<br>BR <addr/lbl>,IN=<0-255><br>CL <addr/lbl>, IN<0-255> |
| **Units** | Decimal |
| **Range** | 0 —255 |
| **Usage** | Program/Immediate. Read |
| **Code Example** | `PR IN  'Print value of IO4-IO1 and IO9-IO12`<br>`BR 324,IN=225  'Branch to address 324 if IN=225`<br>`CL K3,IN=113  'Call subroutine K3 if IN=113` |
| **Notes** | While tailored to Mdrive and MForce models with 8 I/O points, this command will function the same as IL on units with 4 I/O. |
| **Related** | IH, IL, OL, OH, OT, S1-S4, S9-S12 |

## IP (Initialize parameters)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | IP |
| **Function** | Initialize parameters |
| **Type** | Instruction |
| **Description** | The IP instruction will return all of the device variable and flag parameters to their stored values.<br>If IP is used while the motor is moving an Error 74: Tried to Initialize Parameters of Clear Program while moving will be issued. |
| **Syntax** | IP |
| **Usage** | Program/Immediate |
| **Code Example** | `IP  'initialize parameters` |
| **Related** | S, FD |

## IT (Read internal temperature)

Compatible with Motion Control products:
MDrive 34 DC and AC
MDrive 34 (Plus²) DC and AC
MDrive Hybrid
MForce PowerDrive
MForce (Plus²) PowerDrive

| | |
|---|---|
| **Mnemonic** | IT |
| **Function** | Read the internal temperature of the device |
| **Type** | Variable |
| **Description** | Internal Temperature of the MCode compatible driver electronics. Only used on MDrive 34, AC input MDrive, MForce PowerDrive and MDrive Hybrid. |
| **Syntax** | PR IT |
| **Units** | Degrees |
| **Range** | -55°C to 125°C |
| **Usage** | Program/Immediate. Read |
| **Code Example** | PR IT  ‘Print the internal temperature to the terminal screen |
| **Related** | WT |

## IV (Input to variable)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | IV |
| **Function** | Input data into variable |
| **Type** | Instruction |
| **Description** | With the IV command, a user may input new variable values. These values must be numeric and will be input into the variable specified in the IV command.<br>The variable used for the IV may be a system or USER Variable. A USER Variable must be declared prior to the IV command.<br>When waiting for user input, there must be a conditional program loop based upon the state of IF (Input Pending) until the variable is input by the user. |
| **Syntax** | IV <user variable/R1-R4> |
| **Usage** | Program/Immediate |
| **Code Example** | `IV R1  'Input data into R1`<br>`VA K5  'Create user variable K5`<br>`IV K5  'Input data into user variable K5`<br><br>`IV R1  'input value into register 1`<br>`  LB k1  'label program loop k1`<br>`  BR k1,If=1 'branch to k1 awaiting user input` |
| **Related** | IF |

## JE (Jog enable)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | | |
|---|---|---|
| **Mnemonic** | JE | |
| **Function** | Enable/disable jog functions | |
| **Type** | Flag | |
| **Description** | This command will enable Jog Mode if I/O are set for Jog Plus and/or Jog Minus. | |
| **Syntax** | JE=<0/1> | |
| **States** | 0 | Jog functions disabled |
| | 1 | Jog functions enabled |
| **Default** | 0 | |
| **Usage** | Program/Immediate, Read/Write | |
| **Code Example** | `JE=0  'Disable Jog mode`<br>`JE=1  'Enable Jog mode` | |

# L (List program space)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | L |
| **Function** | List contents of program space |
| **Type** | Instruction |
| **Description** | The L instruction will print the contents of program space beginning at the specified address to the end. If no address is specified it will list beginning at address 1. |
| **Syntax** | L <address/label> |
| **Usage** | Immediate |
| **Code Example** | `L  'List program space contents from address 1`<br>`L G5  'List program space contents from label G5` |
| **Related** | CP, FD |

# LB (Label program or subroutine)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | LB |
| **Function** | Label program or subroutine |
| **Type** | Instruction |
| **Description** | The LB, or Label Instruction, allows the user to assign a 2 character name to a program, (BR) branch process or (CL) call subroutine. There is a limit of 192 labels.<br>The restrictions for this command are:<br>1. A label cannot be named after an MCode Instruction, Variable or Flag or Keyword.<br>2. The first character must be alpha, the second character may be alpha-numeric.<br>3. A label is limited to two characters.<br>4. A program labeled SU will run on power-up<br>5. Labels ARE NOT case sensitive. |
| **Syntax** | LB <label> |
| **Usage** | Program |
| **Code Example** | `PG 100  'Start Program at address 100`<br>`LB G1  'Name program G1`<br><br>`PG 1`<br>`LB SU  'Label Program to execute on power up.` |
| **Related** | BR, CL, EX, TI, TP, L, CP |

## LD (Lead limits)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

| | |
|---|---|
| **Mnemonic** | LD |
| **Function** | Set rotor lead limit |
| **Type** | Variable |
| **Description** | LD will set the lead limit in motor steps. |
| **Syntax** | LD=<steps> |
| **Units** | steps |
| **Range** | 0 — 2147483647 |
| **Default** | 102400 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `LD=52100 'set lead limit to 52100` |
| **Notes** | When the lead limit is reached, error 106, lead limit reached will be active |
| **Related** | LG, LL |

## LG (Lag limits)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

| | |
|---|---|
| **Mnemonic** | LG |
| **Function** | Set rotor lag limit |
| **Type** | Variable |
| **Description** | LD will set the lag limit in motor steps. |
| **Syntax** | LG=<steps> |
| **Units** | steps |
| **Range** | 0 — 2147483647 |
| **Default** | 102400 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `LG=52100 'set lag limit to 52100` |
| **Notes** | When the lag limit is reached, error 107, lag limit reached will be active |
| **Related** | LD, LL |

## LL (Position lead/lag)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

| | |
|---|---|
| **Mnemonic** | LL |
| **Function** | Read rotor lead/lag |
| **Type** | Variable |
| **Description** | Represent the number of counts that the rotor leads or lags the stator.<br>A positive value indicates position lag. A negative value indicates position lead |
| **Syntax** | PR LL<br>CL <address/label>, LL <condition><steps><br>BR <address/label>, LL <condition><steps> |
| **Units** | steps |
| **Range** | -2147483647 to +2147483647 |
| **Usage** | Program/Immediate, Read only |
| **Code Example** | `PR LL  'read rotor lead/lag`<br>`CL k5, LL>102500 'Call subroutine k5 if 'LL is greater than '102500 steps` |
| **Related** | LD, LG |

## LK (Lock user program)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | LK |
| **Function** | Lock access to user program space |
| **Type** | Flag |
| **Description** | This flag allows the user to lock the program from being listed or modified. It can only be reset by clearing the entire program space: CP (no address). If CP (address,/label), L (address/label) or PG (address/label) are entered, then error 44 (Program Locked) will be set and nothing else will happen.<br>To clear LK, don't save (S) then do a Ctrl-C or Cycle Power and the LK will be reset to previous unlocked state. (Program is automatically stored in NVM as it is entered.) Or you may clear program (CP or FD to list or reprogram the device.). This will clear the program and reset LK to 0 |
| **Syntax** | LK=<0/1> |
| **States** | 0 User programs may be modified or listed<br>1 User programs may not be modified or listed |
| **Default** | 0 (disabled)_ |
| **Usage** | Immediate, Read/Write |
| **Code Example** | `LK=1  'Lock programs from being listed or changed` |

# LM (Limit stop modes)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | | |
|---|---|---|
| **Mnemonic** | LM | |
| **Function** | Limit stop modes | |
| **Type** | Variable | |
| **Description** | The LM variable specifies the Limit Stop Mode for the MCode compatible device. There are six LM modes. | |
| **Syntax** | LM=<mode> | |
| **Range** | 1 - 6 | |
| **Modes** | 1 | Normal Limit function with a decel ramp.<br>The I/O must be set for Limits (S1-S4, S9-S12 Command). If the limit switch in the direction of travel is reached, the motion will decel to a stop. That is, the plus limit works only in the plus direction of travel and the minus limit works only in the minus direction of travel.<br>In Figure 5.2, the Limit is activated at a given position but because of the deceleration rate the motion continues for the duration of the deceleration time. This position may be beyond the trip point of the limit and a subsequent move in the same direction will not stop. A crash may be imminent.<br>If the limit is activated and maintained the software will allow motion only in the opposite direction.<br>If Homing (HM) is active and a limit is reached, the motion will decel to a stop and then reverse direction and seek the Homing Switch. If the Homing Switch is not activated on the reverse and the opposite limit is reached all motion will stop with a decel ramp. (See HM) |
| | 2 | A Limit stops all motion with a deceleration ramp but no Homing. |
| | 3 | A Limit will stop all motion with a deceleration ramp and stop program execution. |
| | 4 | Functions as LM=1 but with no deceleration ramp |
| | 5 | Functions as LM=2 but with no deceleration ramp. |
| | 6 | Functions as LM=3 but with no deceleration ramp. |
| **Usage** | Program/Immediate, Read/Write | |
| **Code Example** | `LM=2 'Set Limit stop with a decel ramp, no homing.` | |
| **Related** | H,I, HM, JE, MA, MR, SL | |

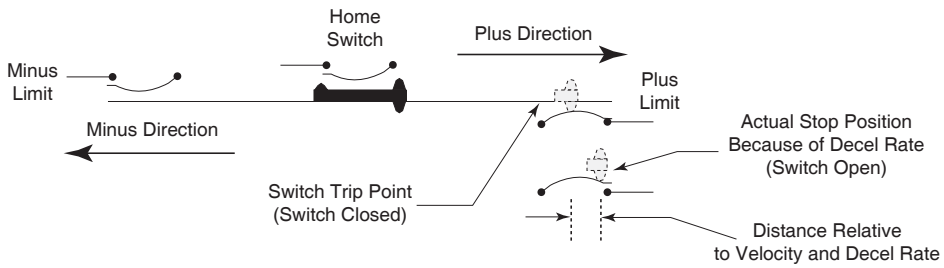



Figure 5.2 Limit modes of operation

## LR (Locked rotor)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

A locked rotor is defined as no rotor movement while at the maximum allowed lag for a specified period of time. When lag becomes equal to the bounds, a timer starts to count down. Upon reaching zero, a locked rotor will be indicated by the assertion of a status flag. The timer reloads on any encoder movement. The timer timeout period is user selectable from 2mS to 65.5 seconds.

When Hybrid is configured AS=1 or 2, a locked rotor will also cause an internal fault (LR) disabling the output bridge. The bridges may be re-enabled by cycling power, cycling the enable input, or via software command.

In torque mode, a locked rotor does not disable the bridges. The locked rotor flag (LR) can be used to indicate the rotor has been stopped at the specified torque for a preset amount of time.

| | |
|---|---|
| **Mnemonic** | LR |
| **Function** | Locked rotor |
| **Type** | Read only status flag |
| **Description** | Indicates the state of the rotor as locked (1) or unlocked (0). |
| **Syntax** | PR LR |
| **Range** | 0/1 |
| **Usage** | Program/Immediate, Read only |
| **Code Example** | `PR LR  'read rotor lead/lag`<br>`TA 2,k3 'trip to sub k3 on locked otor` |
| **Related** | LT, TA |

## LT (Locked rotor timeout)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

| | |
|---|---|
| **Mnemonic** | LT |
| **Function** | Set locked rotor timeout |
| **Type** | Variable |
| **Description** | The LT variable will set the locked rotor timeout in milliseconds. This setting will determine the time in milliseconds that the output bridge will disable following the locked rotor (LR) flag being active. |
| **Syntax** | LT=<time> |
| **Units** | Milliseconds |
| **Range** | 2 — 65535 |
| **Default** | 2000 mS |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `LT=1000  'set locked rotor time to 1S` |
| **Notes** | The output bridge will not disable if the Hybrid is in torque mode (AS=3) |
| **Related** | AS, CB, LR |

## MA (Move to an absolute position)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | MA |
| **Function** | Move to an absolute position |
| **Type** | Instruction |
| **Description** | Set mode for absolute move and move to an absolute position relative to (0) zero. MD (Motion Mode) will be set to MA. The time required to calculate the move is 2.5 mSec.<br>MA command will not operate during a homing sequence.<br>An in progress MA can be stopped with and "esc" entry or an SL 0 command. |
| **Syntax** | MA <±position>,<parameter 2>, <parameter 3> |
| **Parameter 2** | 0 (or blank) no action taken<br>1 Device name will be sent out of the serial port on move completion. |
| **Parameter 3** | 0 (or blank) no action taken.<br>1 Motor will continue moving after reaching position. |
| **Usage** | Program/Immediate |
| **Code Example** | `MA 200000 'Move to absolute position 200000`<br>`MA 100000,1 'Move to abs. pos. 100000, send DN when 'complete`<br>`MA 512000,0,1 'Move to abs. pos. 512000, continue 'motion after position is reached.` |
| **Related** | MD, MR, MS, P, SL |

## MD (Motion mode)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | MD |
| **Function** | Motion mode |
| **Type** | Variable |
| **Description** | Indicates what the last motion command was. When just a number is entered, then it will execute the move type according to the previous move type entered. This allows the user to apply numeric data to the last motion command without having to enter the command itself. |
| **Syntax** | PR MD |
| **Units** | Motor steps/Encoder counts |
| **Usage** | Program/Immediate. Read |
| **Code Example** | `PR MD 'Return the last motion command used to the 'terminal screen. Response will be the last 'motion command ie. MR`<br><br>`MR 10000 'Move Relative 10000 steps`<br>`5000 'Motor will move relative 5000 steps` |
| **Notes** | Note that if the IF flag is pending, numeric entry will be applied to the IV (Input Variable). |
| **Related** | MA, MR, MS, P, PR, SL |

## MF (Make up frequency)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

| | |
|---|---|
| **Mnemonic** | MF |
| **Function** | Set make up frequency |
| **Type** | Variable |
| **Description** | Defines the frequency at which missed steps are re-inserted into the move profile. |
| **Syntax** | MF=<frequency> |
| **Units** | Hz |
| **Range** | 306 — 5000000 |
| **Default** | 768000 Hz |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `MF=256000 'set make up freq to 25.6 kHz` |
| **Notes** | MF will be used for make up if MU=1 |
| **Related** | MU, SS |

## MP (Moving to position)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | MP |
| **Function** | Moving to position |
| **Type** | Flag |
| **Description** | This flag will=1 when the axis is moving to a position following a MA or MR instruction until MT expires.. |
| **Syntax** | PR MP |
| **Range** | 0/1 |
| **Status Response** | 0 Not moving |
| | 1 Moving to a position |
| **Usage** | Program/Immediate, Read |
| **Code Example** | `PR MP 'read the state of the MP flag to terminal` |
| **Related** | MT |

# MR (Move to a relative position)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | MR |
| **Function** | Move to a relative position |
| **Type** | Instruction |
| **Description** | Set mode for relative move and move a relative distance. MD (Motion Mode) will be set to MR. The time required to calculate the move is 2.5 mSec.<br>MR command will not operate during a homing sequence.<br>An in progress MR can be stopped with and "esc" entry or an SL 0 command. |
| **Syntax** | MR <±position>,<parameter 2>, <parameter 3> |
| **Parameter 2** | 0 (or blank) no action taken<br>1 Device name will be sent out of the serial port on move completion. |
| **Parameter 3** | 0 (or blank) no action taken.<br>1 Motor will continue moving after reaching position. |
| **Usage** | Program/Immediate |
| **Code Example** | `MR 200000 'Move to relative position 200000`<br>`MR 100000,1 'Move to rel. pos. 100000, send DN when 'complete`<br>`MR 512000,0,1 'Move to rel. pos. 512000, continue 'motion after position is reached.` |
| **Related** | MD, MR, MS, P, SL |

# MS (Microstep resolution select)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | MS |
| **Function** | Microstep resolution select |
| **Type** | Variable |
| **Description** | The MS variable controls the microstep resolution of the MCode compatible device. There are 20 different microstep resolutions that can be used with the device. The table below illustrates the parameter settings and their associated resolutions for the 1.8° stepping motor used with the MCode compatible device.<br>The MS parameters given in the table below are the only valid parameters that will be accepted by the device. |
| **Syntax** | MS=<parameter> |
| **Units** | Microsteps per full motor step |
| **Default** | 256 |

**Parameters**

**Binary resolution parameters**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| per step | 1 | 2 | 4 | 8* | 16 | 32 | 64 | 128 | 256 |
| per rev, | 200 | 400 | 800 | 1600 | 3200 | 6400 | 12800 | 25600 | 51200 |

**Decimal resolution parameters**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| per step | 5* | 10 | 25 | 50 | 100 | 125 | 200 | 250 |
| per rev, | 1000 | 2000 | 5000 | 10000 | 20000 | 25000 | 40000 | 50000 |

**Additional resolution parameters**

| | | | |
|---|---|---|---|
| per step | 108 | 127 | 180 |
| per rev, | 21600 (1 Arc Minute/µStep) | 25400 (0.001mm/µStep) | 36000 (0.01°/µStep) |

*Not recommended for MDrive Hybrid

| | |
|---|---|
| **Usage** | Program/Immediate. Read/Write |
| **Code Example** | MS=5 'microstep resolution to 1000 steps/rev |
| **Related** | MA, MR, MS, P, PR, SL, C1 |

# MT (Motor settling delay time)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | MT |
| **Function** | Motor settling delay time |
| **Type** | Variable |
| **Description** | Specifies the motor settling delay time in milliseconds. MT allows the motor to settle following a move. This is the time between moves if consecutive motions are executed. the MV flag will be active during this time. |
| **Syntax** | PR MT |
| **Units** | Milliseconds |
| **Range** | 0 to 65000 |
| **Default** | 0 |
| **Usage** | Program/Immediate. Read/Write |
| **Code Example** | `MT=50 'Set motor settling delay time to 50 milliseconds` |
| **Notes** | MT is added into HT (Hold Current Delay Time). The total of the two cannot exceed 65535. Thus the maximum setting for MT=(65535-HT).<br>MT should be at least 50 mS when Encoder function is enabled (EE=1) |
| **Related** | HC, HT, RC |

## MU (Make up mode)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

Automatic position maintenance can be enabled which will insert steps as required, when conditions allow, in the appropriate direction to bring the position difference between the commanded number of steps and actual steps taken to zero. The speed of position maintenance can be performed at one (1) of two (2) speeds as specified by the make up (MU) variable. Insertion can be at a specified speed or can be set at the maximum speed the load will allow. There is no acceleration or deceleration applied to position make up, therefore make up could be abrupt at high speed.

| | | |
|---|---|---|
| **Mnemonic** | MU | |
| **Function** | Set make up mode | |
| **Type** | Variable | |
| **Description** | Determines the mode for position maintenance | |
| **Syntax** | MU=<mode>,<parameter> | |
| **Range** | 0 — 2 | |
| **Modes** | 0 | Off |
| | 1 | Use make up frequency (MF) as make up frequency |
| | 2 | Use system speed (SS) as make up frequency |
| **Parameters** | 0 | Use lead/lag (LL) |
| | 1 | Clear lead/lag (LL) |
| **Default** | 0 (Off) | |
| **Usage** | Program/Immediate, Read/Write | |
| **Code Example** | `MU=2,1  'use SS, clear LL` | |
| **Notes** | The make up parameter is not saved. Position maintenance will only occur when motor lag/lead is within 1.1 motor steps. Make up steps may be interleaved with motion steps and made after a move has completed. Where position maintenance occurs is dependant on motor lag/lead, motion frequency and selected make up speed. Make up mode will be cleared when bridges are disabled and Hybrid is enabled. | |
| **Related** | MF, SS | |

## MV (Moving)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | MV |
| **Function** | Axis in motion |
| **Type** | Flag |
| **Description** | Moving flag will be in a logic 1 state when a motion is occurring. This flag will set an output ACTIVE if S<1-4, 9-12>=17 |
| **Syntax** | PR MV |
| **Range** | 0/1 |
| **Status Response** | 0 Not moving |
| | 1 Moving |
| **Usage** | Program/Immediate, Read |
| **Code Example** | `PR MV  'read the state of the MV flag to terminal` |
| **Related** | VC, S1-S4, S9-S12 |

## NE (Numeric enable)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | NE |
| **Function** | Numeric enable flag |
| **Type** | Flag |
| **Description** | Numeric enable flag will enable (default) or disable the ability of the device to execute the last motion command upon the entry of a numeric value without the preceding command being entered. By default, if a motion command is executed i.e. MR 20000, when the user enters -20000, the device will move relative -20000.<br>If disabled, the user must enter a motion command to execute a move, i.e. MA 100000, MR -50000, SL 300000 etc. |
| **Syntax** | NE=<1/0> |
| **Range** | 0/1 |
| **Default** | 1 (enabled) |
| **Status Response** | 0 Disabled<br>1 Enabled (Default) |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `NE=0 'disable motion mode on numeric entry` |
| **Related** | MA, MR, SL |

## O1, O2, O3 and O4 (Set outputs 1, 2 ,3 or 4)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | O1 - O4 |
| **Function** | Set output logic state |
| **Type** | Variable |
| **Description** | This variable will set the logic state of the specified output to 1 or 0.<br>The voltage level will be dependant on the active (low/high) state of the output, specified by the S<1-4> variable. |
| **Syntax** | Syntax:O<1-4>=<0/1> |
| **Units** | Logic 0 or 1 |
| **Range** | 0/1 |
| **Usage** | Program/Immediate. Write |
| **Code Example** | `O2=1 'Set output 2 ACTIVE` |
| **Related** | O9-O12, OL, OH, OT, S1-S4 |

## O9, O10, O11 and O12 (Set outputs 9, 10 ,11 or 12)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | O9 - O12 |
| **Function** | Set output logic state |
| **Type** | Variable |
| **Description** | This variable will set the logic state of the specified output to 1 or 0.<br>The voltage level will be dependant on the active (low/high) state of the output, specified by the S<1-4> variable. |
| **Syntax** | O<9-12>=<0/1> |
| **Units** | Logic 0 or 1 |
| **Range** | 0/1 |
| **Usage** | Program/Immediate. Write |
| **Code Example** | `O10=1 'Set output 10 ACTIVE` |
| **Related** | O1-O4, OL, OH, OT, S1-S4 |

## OE (On error handler)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | OE |
| **Function** | On error handler |
| **Type** | Instruction |
| **Description** | When an error occurs, the specified subroutine is called. If a program was running when the fault occurs, once the error routine completes, program execution continues with the instruction after the one that caused the error. After OE, the command is executed. A program need not be running for the subroutine specified by OE to run.<br>The ON ERROR function is disabled by setting the address parameter to 0 or resetting the device with an FD or CP OE Subroutine MUST have an RT at the end.<br>OE will not execute during programming. |
| **Usage** | Program |
| **Code Example** | `OE K1 'run subroutine K1 on an error` |
| **Related** | EF, ER |

## OL (Set outputs 1 - 4 as a group)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive Hybrid
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | OL |
| **Function** | Set the state of outputs 1 - 4 as one value |
| **Type** | Variable |
| **Description** | The OL variable allows the user to set Outputs 1-4 as one 4 bit binary value. The value is entered in decimal, with a range of 0-15 in binary where Output 1 will be the LSb and Output 4 will be the MSb. |
| **Syntax** | OL=<0-15> |
| **Units** | Decimal |
| **Range** | 0 —15 |
| **Usage** | Program/Immediate. Write |
| **Code Example** | `OL=13 'set output group 1-4 to 1101` |
| **Related** | IH, IN, IL, OH, OT, S1-S4 |

## OH (Set outputs 9 - 12 as a group)

Compatible with Motion Control products:
MDrive (Plus² expanded features)
MDrive Hybrid
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | OH |
| **Function** | Set the state of outputs 9 - 12 as one value |
| **Type** | Variable |
| **Description** | The OH variable allows the user to set Outputs 9-12 as one 4 bit binary value. The value is entered in decimal, with a range of 0-15 in binary where Output 9 will be the LSb and Output 12 will be the MSb. |
| **Syntax** | OH=<0-15> |
| **Units** | Decimal |
| **Range** | 0 —15 |
| **Usage** | Program/Immediate. Write |
| **Code Example** | `OH=13 'set output group 9-12 to 1101` |
| **Related** | IH, IN, IL, OL, OT, S9-S12 |

## OT (Set outputs 1- 4 and 9 - 12 as a group)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive Hybrid
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | OT |
| **Function** | Set the state of outputs 1 - 4 and 9 - 12 as one value |
| **Type** | Variable |
| **Description** | The OT variable allows the user to set Outputs 1-4 and 9-12 as one 8 bit binary value. The value is entered in decimal, with a range of 0-255 in binary where Output 1 will be the LSb and Output 12 will be the MSb. |
| **Syntax** | OH=<0-255> |
| **Units** | Decimal |
| **Range** | 0 — 255 |
| **Usage** | Program/Immediate. Write |
| **Code Example** | `OT=214 'set the standard output group to 11010110` |
| **Notes** | NOTE: On Standard Plus Models OT will only set the lower group (Outputs 1-4) |
| **Related** | IL, IH, IN, OL, OH, S1-S4, S9-S12 |

## P (Position counter)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | P |
| **Function** | Position counter |
| **Type** | Variable |
| **Description** | This instruction is used to set or print the value of the MCode compatible device position counter. The position will read in Motor Steps from C1 (Counter 1) by default, if encoder functions are enabled, the position counter will read in Encoder Counts from C2 (Counter 2).<br><br>Modifying P in essence changes the frame of reference for the axis for Move Absolute (MA) instructions. P will probably be set once during system set up to reference or “home” the system. |
| **Syntax** | P=<±position>,<br>PR P |
| **Units** | Motor Steps (EE=0)/Encoder Counts (EE=1 |
| **Range** | -2147483648 to +2147483647 |
| **Default** | 0 |
| **Usage** | Program/Immediate. Read/Write |
| **Code Example** | `PR P 'sprint the position to the terminal window` |
| **Related** | C1, C2 |

## PC (Position capture at trip)

Compatible with Motion Control products:
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | PC |
| **Function** | Position capture at trip |
| **Type** | Instruction |
| **Description** | Captures motor or encoder position during a trip event. Activation will occur upon any trip function EXCEPT a position trip (TP). Will display in either motor steps (EE=0) or encoder counts (EE=1) |
| **Syntax** | PR PC |
| **Usage** | Immediate |
| **Code Example** | `PR PC    'Display captured position` |
| **Related** | TE, TI, TC, TT, S13 |

## PG (Program mode))

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | PG |
| **Function** | Enter/exit program mode |
| **Type** | Instruction |
| **Description** | This instruction toggles the device into or out of program mode. |
| **Syntax** | PG <address> |
| **Usage** | Immediate/Program |
| **Code Example** | `PG 100 'enter program at address 100`<br>`...`<br>`...  'program body`<br>`...`<br>`PG   'end program return to immediate mode` |
| **Related** | CP |

## PM (Position maintenance enable)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | PM |
| **Function** | Position maintenance enable |
| **Type** | Flag |
| **Description** | This flag will enable the position maintenance functions of an MCode compatible device with encoder. The position maintenance velocity will be at the setting for VI (Initial Velocity). If moved beyond the value of DB (DeadBand), unit will correct<br>If SM = 0 and PM = 1, Position Maintenance will take place provided the position does not exceed the Stall Factor (SF).<br>If SM = 1 and PM = 1, Position Maintenance will take place even if the Stall Factor (SF) is exceeded, unless VI is set too high causing the motor to stall. |
| **Syntax** | PM=<1/0> |
| **Range** | 0/1 |
| **Default** | 1 (enabled) |
| **Status Response** | 0 Disabled (Default)<br>1 Enabled |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `PM=1 'Enable position maintenance` |
| **Related** | VI, EE, SM, DB, C2, SF |

## PN (Part number)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | PN |
| **Function** | Read device part number |
| **Type** | Variable |
| **Syntax** | PR PN |
| **Usage** | Immediate, Read |
| **Code Example** | `PR PN 'read the device part number` |

## PR (Print specified data or text)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | PR |
| **Function** | Print specified data or text |
| **Type** | Instruction |
| **Description** | This instruction is used to output text and parameter value(s) to the host PC. Text should be enclosed in quotation marks while parameters (variables and flags) should not. Text strings and parameters which are to be output by the same PR instruction should be separated by commas. The information being output is followed by a carriage return unless a semicolon (;) is included at the end of the PR instruction to indicate that the cursor should remain on the same line.<br><br>It is important to note that the receive buffer for the MCode device is 64 characters, this includes the PR instruction itself, any spaces, text characters, etc. If the buffer length is exceeded a CR/LF will ocur and set error 20. |
| **Syntax** | PR <text>, <data> |
| **Usage** | Immediate |
| **Code Example** | `PR "Position =", P 'print axis position`<br>`PR MS 'Print the µStep Resolution setting` |
| **Notes** | A delay time between the print requests to the device must be considered to allow the device time to interpret a command and answer the host before a subsequent command can be sent. |

## PS (Pause program execution)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | PS |
| **Function** | Pause program execution |
| **Type** | Instruction |
| **Description** | This instruction is used to pause an executing program and invoke normal deceleration of any motion being executed to Zero. Immediate mode instructions are allowed while a program is in a paused state. To resume the program the RS instruction is used. |
| **Syntax** | PS |
| **Usage** | Immediate |
| **Code Example** | `PS 'Pause running program` |
| **Related** | RS, SL, MA, MR, HI, HM |

# PW (PWM configuration)

Compatible with Motion Control products:
MForce
MForce (Plus$^2$ expanded features)

**⚠ CAUTION**

This variable is only applicable to the MForce Product Line and is used to tune the PWM Settings to optimize the current control of the MForce Driver. It should not be used unless erratic motion or positional accuracy problems are being experienced.

Note that there are other factors that could contribute to these problems. Ensure that all wiring conforms to the guidelines in the MForce Hardware Manual.

Be aware that this parameter, when used with the checksum will write to the boot sector of memory. This sector only allows eight write cycles. Ensure that the settings you choose are optimal prior to storing the parameter to the boot.

Read this section closely before making changes to the PWM settings. Please contact application support for questions concerning the use of this command.

| | |
|---|---|
| **Mnemonic** | PW |
| **Function** | PWM configuration |
| **Type** | Variable |
| **Description** | The PW variable is only used on the MForce product line. It is not a reserved word on the MDrive product line and may be used as a User Variable or Label.<br><br>This variable is used to set the PWM Current Control settings of the MForce ONLY! It does not apply in any function to the MDrive series and may be used as a label or user variable or flag.<br><br>See Appendix G of this document for parameter settings and usage. It is recommended that these settings not be modified from the factory default unless the motor exhibits rough motion, audible noise or poor zero-crossing performance. If these symptoms exist, please check for other issues such as electrical noise coupled onto logic signals or ground loops before modifying these settings. |
| **Syntax** | PW=<mask>,<period>,<sfreq>, <chksum><br><br>PR PW (Response = <mask>,<period>,<sfreq>,<boot_writes_remaining> |
| **Units** | Vary |
| **Range** | See tables in Section 7.4: MForce PWM Settings, of this document. |
| **Default** | 204,95,170 |
| **Usage** | Immediate. Read/Write |

# PY (Party mode enable)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive Hybrid
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | PY |
| **Function** | Party mode enable |
| **Type** | Flag |
| **Description** | The party flag must be set to 1 if the device is being used in a multidrop communication system.<br><br>When Party Mode is enabled, each device in the system must be addressed by the host computer by using the device name specified by the DN instruction. This name will precede any command given to a specified unit in the system and be terminated with a Control J (CTRL + J). One CTRL + J must be issued after power up or entering the Party Mode to activate the Party Mode. By default the DN assigned at the factory is the exclamation character (!) .<br><br>The global Drive Name is the asterisk character (*). Commands preceded by this character will be recognized by every MCode compatible device in the system.<br><br>After the Party Mode is enabled, send CTRL + J (^J) to activate it. Type commands with Device Name (DN) and use CTRL + J as the Terminator. |
| **Syntax** | PY=<0/1> |
| **Range** | 0/1 |
| **Default** | 0 (disabled) |
| **Status** | 0 Disabled (Default)<br>1 Enabled |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `PY=1 'Enable Party Mode Communications` |
| **Notes** | Note: A delay time between the command requests to the device must be considered to allow the device time to interpret a command and answer the host before a subsequent command can be sent. The time between requests is dependent on the command and the corresponding response from the Device. |
| **Related** | DN, DG |

## QD (Queued)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive Hybrid
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | QD |
| **Function** | Queued |
| **Type** | Flag |
| **Description** | Function is to queue drives on party lines. Works similar to uLynx QUED flag.<br>If a drive or drives are Queued, then, when they see the address "^", they will respond to it. All other, non-queued drives will ignore the command |
| **Syntax** | QD=<0/1> |
| **Range** | 0/1 |
| **Default** | 0 (disabled) |
| **Status** | 0 Disabled (Default)<br>1 Enabled |
| **Usage** | Immediate, Read/Write |
| **Code Example** | `QD=1 'Queue Drive` |
| **Related** | PY, DN |

## R1, R2, R3 and R4 (User registers)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | R1 - R4 |
| **Function** | User registers |
| **Type** | Variable |
| **Description** | The MCode compatible device has four 32 bit user registers to contain numerical data. These registers may contain up to 11 digits including the sign and may be used to store and retrieve data to set variables, perform math functions, store and retrieve moves and set conditions for branches and call subroutine. |
| **Syntax** | R<1-4>=<number> |
| **Range** | -2147483647 to 2147483647 |
| **Usage** | Immediate, Read/Write |
| **Code Example** | `R1=50000 'Set Register 1 to 50000`<br>`R2=Q2 'Set Reg. 2 to the value of User Variable Q2` |

## RC (Motor run current)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | RC |
| **Function** | Motor run current |
| **Type** | Variable |
| **Description** | This variable defines the motor run current in percent. |

| HC=(%) | MDrive MDrive Hybrid (All) | MForce MicroDrive (A RMS) | MForce PowerDrive (A RMS) |
|---|---|---|---|
| 10 | MDrive Range 0 To 100% | 0.3 | 0.5 |
| 20 | | 0.6 | 1.0 |
| 30 | | 0.9 | 1.5 |
| 40 | Actual Current Not required as Motor is appropriately sized to the device. | 1.2 | 2.0 |
| 50 | | 1.5 | 2.5 |
| 60 | | 1.8* | 3.0 |
| 70 | | 2.1* | 3.5 |
| 80 | | 2.4* | 4.0 |
| 90 | | 2.7* | 4.5 |
| 100 | | 3.0* | 5.0 |

*Older versions of the MForce MicroDrive would output 2.0 A RMS. For these drives 67% run current will output 2.0 A. The 3.0 A MicroDrives are marked 3.0 Amps on the label.

| | |
|---|---|
| **Syntax** | RC= <percent> |
| **Units** | Percent |
| **Range** | 1 to 100 |
| **Default** | 25% |
| **Usage** | Program/Immediate |
| **Code Example** | `RC=50  'set run current to 50%` |
| **Related** | HC |

## RS (Resume program execution)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive Hybrid
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | RS |
| **Function** | Resume program execution |
| **Type** | Instruction |
| **Description** | This instruction is used to resume a program that has been paused using the PS instruction. Any move that was paused will resume as well. Motion will resume using the normal acceleration profiles. |
| **Syntax** | RS |
| **Usage** | Immediate |
| **Code Example** | `RS  'Resume paused program` |
| **Related** | PS, SL, MA, MR, HI, HM |

## RT (Return from subroutine)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | RT |
| **Function** | Return from called or on-error subroutine |
| **Type** | Instruction |
| **Description** | This instruction defines the end of a subroutine. This instruction is required and will be the final instruction in the subroutine executed by the CL or OE instruction. When used, it will return to the program address immediately following the instruction which executed the subroutine. |
| **Syntax** | RT |
| **Usage** | Program |
| **Code Example** | `CL K8  'Call Subroutine K8`<br>`LB K8`<br>`*****SUBROUTINE K8*****`<br>`RT     'Go back to main program` |
| **Related** | CL, OE |

## S (Save to NVM)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | S |
| **Function** | Save to NVM |
| **Type** | Instruction |
| **Description** | Saves all variables and flags currently in working memory (RAM) to nonvolatile memory (NVM). The previous values in NVM are completely overwritten with the new values.<br>When the user modifies variables and flags, they are changed in working memory (RAM) only. If the S instruction is not executed before power is removed from the control module, all modifications to variables & flags since the last S will be lost. |
| **Syntax** | S |
| **Usage** | Program/Immediate |
| **Code Example** | `S    'Save all variable and flag states to NVM` |
| **Notes** | Note: Communications during a Save could corrupt communications. If a Save is performed during the execution of a motion command, trips may be delayed.<br>Use of the S command during a move (MA or MR) will generate an error 73, the save will not occur. |

# S1, S2, S3, S4, S9, S10, S11, S12 (Setup I/O points 1 - 4 & 9 - 12)

Compatible with Motion Control products:

***S1 - S4***
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

***S9 - S12***
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | S1 - S4, S9 - S12 |
| **Function** | Set up I/O point |
| **Type** | Instruction |
| **Description** | This instruction is used to setup the I/O type and active states, and sink/source setting. Each of the device I/O points may be programmed as either general purpose inputs and outputs, or to dedicated input function or dedicated output function.<br><br>When programmed as inputs, these points can be sinking or sourcing, and may be programmed such that they are active when pulled to ground, or active when left floating. By default each point is configured as a general purpose input, active when LOW.<br><br>When configured as outputs on devices with Plus² expanded features the outputs may be sinking or sourcing. For standard MDrive and MForce devices the outputs are sinking only.<br><br>There are three parameters attached to this instruction:<br>1. The type specifies the function of the I/O point (see tables - following page).<br>2. The second parameter sets the active state, which defines the point as (0 - Default) LOW or (1) HIGH ACTIVE.<br>3. The third parameter specifies whether the point will be (0 - Default) sinking or sourcing (1). |
| **Syntax** | S<point number>=<type>,<0/1>,<0/1> |

| **Input Types** | |
|---|---|
| 0 | General purpose input |
| 1 | Homing function, will function as specified by the homing command (HM). |
| 2 | Limit +, will function as specified by the limit command (LM). |
| 3 | Limit —, will function as specified by the limit command (LM). |
| 4 | G0 input, will run program at address 1 upon activation. |
| 5 | Soft stop, stops motion with deceleration and halts program execution. If program is paused (PS), input is ignored. |
| 6 | Pause, pause/resume program with motion |
| 7 | Jog +, Will Jog motor in the positive direction at Max. Velocity (VM). The Jog Enable (JE) Flag must be set for this to function. |
| 8 | Jog —, Will Jog motor in thenegative direction at max. velocity (VM). The jog enable (JE) flag must be set for this to function. |
| 11 | Reset, When set as RESET input, then the action is equivalent to a ^C entered into a terminal. Note: If setting the input to sourcing, active true, ground the input first or a reset will occur. |

| | | |
|---|---|---|
| **Output Types** | 16 | General purpose |
| | 17 | Moving, output will be active while the moptor is in motion. |
| | 18 | Fault, will be active when an error occurs. |
| | 19 | Stall, will be in the Active State when a stall is detected. Encoder Required, Stall Detect Mode (SM) must be enabled. |
| | 20 | Velocity changing, will be in the Active State when the velocity is changing. |
| | 21 | Locked rotor (MDrive Hybrid ONLY). |
| | 23 | Moving to position, will be in an active state while moving to an absolute position. |
| | 24 | Hybrid active (MDrive Hybrid ONLY). |
| | 25 | Make up active (MDrive Hybrid ONLY). |
| **Parameter 2** | 0 | Active when LOW (default) |
| | 1 | Active when HIGH |
| **Parameter 3** | 0 | Sinking (default) |
| | 1 | Sourcing |
| **Default** | 0 (Off) | |
| **Usage** | Program/Immediate, Read/Write | |
| **Code Example** | `S1=1,0,0 'IO1 to homing input, active LOW, sinking`<br>`S2=4,1,1 'IO2 = G0 input active HIGH, sourcing`<br>`S3=17,1,0 'IO3 = moving output, active HIGH, sinking` | |
| **Notes** | 1. Output Types are SINKING ONLY on Plus devices.<br>2. Once set wait debounce time before using. | |
| **Related** | I1-4, I9-12, IN, O1-4, O9-12, OT, D1-D4 | |

*Output Circuit Conditions*

| | |
|---|---|
| S1=16,1,0 | O1 = 1 (Sink OFF, High Impedance) |
| Output, Active High, Sinking | O1 = 0 (Sink ON) |
| S1=16,0,0 | O1 = 1 (Sink ON) |
| Output, Active Low, Sinking | O1 = 0 (Sink OFF, High Impedance) |
| S1=16,1,1 (Plus$^2$ Only) | O1 = 1 (Source ON) |
| Output, Active High, Sourcing | O1 = 0 (Source OFF, High Impedance) |
| S1=16,0,1 (Plus$^2$ Only) | O1 = 1 (Source Source OFF, High Impedance) |
| Output, Active Low, Sourcing | O1 = 0 (Source ON) |

# S5 (Set analog input)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | S5 |
| **Function** | Set analog input |
| **Type** | Instruction |
| **Description** | This I/O point configures the analog input as either a current or voltage input. The value of this input will be read using the I5 instruction, which has a range of 0 to 1023.<br>The first parameter sets the input type as either voltage or current.<br>The second parameter specifies the voltage/current range:<br>Note that for 4 to 20 mA current mode, the range for I5 is 205 to 1023. |
| **1st Parameter** | 9 Sets the analog input as a voltage input (default).<br>10 Sets the analog input to a current input. |
| **2nd Parameter** | 0 Sets the input range to 0 to 5 V, or 4 to 20 mA (default).<br>1 Sets the input range to 0 to 10 V, or 0 to 20 mA. |
| **Syntax** | S5= <type>, <0/1> |
| **Default** | Voltage reference, 0 to +5 VDC |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `S5=9,0 'Analog Input = voltage, 0-5 VDC`<br>`S5=9,1 'Analog Input = voltage, 0-10 VDC`<br>`S5=10,0 'Analog Input = current, 0-20 mA`<br>`S5=10,1 'Analog Input = current, 4-20 mA` |
| **Related** | I5, JE |

# S7 & S8 (Setup clock I/O points)

Compatible with Motion Control products:
MDrive (Plus² expanded features)
MDrive (Ethernet)
MForce (Plus² expanded features)

| | | |
|---|---|---|
| **Mnemonic** | S7 - S8 | |
| **Function** | Set up clock I/O point | |
| **Type** | Instruction | |
| **Description** | Sets up I/O 7 and I/O 8 clock type. Can be set as inputs or outputs, I/O 7 and I/O 8 are setup in pairs. The clock types are step clock/direction, up/down, and quadrature. | |
| **Syntax** | S<7/8>=<type>,<active> | |
| **Input Types** | 33 | Step/Direction , sets I/O 7 and 8 to receive step and direction inputs from an external source. The motion will occur based on the input frequency seen at I/O 7 in the Direction relative to the logic state of I/O 8. The step rate will be based upon the ratio set by Clock Ratio (CR) |
| | 34 | Quadrature, sets I/O 7 and 8 to receive Channel A and Channel B Quadrature inputs from an external source such as an encoder. The motion will follow the Quadrature Input. The clock rate will be based upon the ratio set by Clock Ratio (CR) |
| | 35 | Up/Down, sets I/O 7 and 8 to receive Clock Up/Clock Down inputs from an external source. The motion will occur based upon the input clock frequency in the direction relative to the input being clocked. The step rate will be based upon the ratio set by Clock Ratio (CR) |
| **Output Types** | 49 | Step clock pulses will be output from Point 7, Direction from Point 8. The step clock output rate will be based upon the Pulse Width set by Clock Width (CW). The logic state of the Direction output will be with respect to the direction of the motor. |
| | 50 | Will output Quadrature signals. |
| | 51 | Will output Clock Up/Clock Down signals. The step clock output rate will be based upon the Pulse Width set by Clock Width (CW). The Active output will be based on the motor direction. |
| **Parameter 2** | 0 | Active when LOW (default) |
| | 1 | Active when HIGH |
| **Usage** | Program/Immediate, Read/Write | |
| **Code Example** | `S7=33,0 'IO7 = step/direction input, active LOW`<br>`S8=33,0 'IO8 = step/direction input, active LOW`<br><br>`S7=50,1 'IO7 = quadrature output, active HIGH`<br>`S8=50,1 'IO8 = quadrature output, active HIGH` | |
| **Related** | FM, CW, CM | |

## S13 (Setup capture input/trip output)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | | |
|---|---|---|
| **Mnemonic** | S13 | |
| **Function** | Set up capture/trip I/O point | |
| **Type** | Instruction | |
| **Description** | Sets up high speed position capture input and position trip output. | |
| **Syntax** | S13=<type>,<active> | |
| **Input Type** | 60 | The Capture input is a momentary high speed input that operates with the Trip Capture (TC) variable to run a subroutine upon the trip. It feature variable input filtering ranging from 50 nS to 12.9 µS |
| **Output Type** | 61 | The trip output will activate on Position Trips (TP) or Trip Relative (TR). The output will pulse out at the trip point. The pulse width will be determined by Clock Width (CW) |
| **Parameter 2** | 0 | Active when LOW (default) |
| | 1 | Active when HIGH |
| **Usage** | Program/Immediate, Read/Write | |
| **Code Example** | `S13=60,0 'Set up IO13 as High Speed Capture`<br>`'Input, active LOW (default)` | |
| **Notes** | Note that this I/O Point is for position capture and trip ONLY. It is not effected by any other trips. | |
| **Related** | FC, CW, TP, TR, PC | |

## SC (Start calibration)

Compatible with Motion Control products:
MDrive Hybrid

| | |
|---|---|
| **Mnemonic** | SC |
| **Function** | Start calibration |
| **Type** | Instruction |
| **Description** | Starts calibration |
| **Syntax** | SC |
| **Units** | — |
| **Usage** | Program/Immediate, Write |
| **Code Example** | `SC 'start calibration` |
| **Notes** | Start calibration mode 0 will initiate a Hard Stop Calibration resulting in a small motion of 10.8° CCW followed by 5.4° CCW to align the rotor to the stator. |
| **Related** | CA, CC, CT, CF |

## SF (Stall factor)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | SF |
| **Function** | Stall factor |
| **Type** | Variable |
| **Description** | If the encoder is enabled (EE = 1) and the encoder differs from the commanded position by more than the specified factor, a STALL is indicated. If SM is set to 0, then the motor will be stopped when a STALL is detected. If SM=1, the motor will not be stopped upon detection of a stall. ST will return an ER=86 on stall. |
| **Syntax** | SF=<counts> |
| **Units** | Encoder counts |
| **Range** | 0 to 65000 |
| **Default** | 15 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `SF=20 'Set the stall factor for 20 counts` |
| **Related** | EE, SM, ST, PM |

## SL (Slew axis)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | SL |
| **Function** | Slew Axis |
| **Type** | Instruction |
| **Description** | The SL instruction will slew the axis at the specified velocity in steps per second. The axis will accelerate at the rate specified by the A (Acceleration) variable.<br>Note that the maximum slew velocity is independent of the maximum velocity specified by the VM variable. If a slew is commanded at a velocity greater than the setting of VM, the axis will accelerate to the SL velocity regardless of the setting of VM. |
| **Syntax** | SL <velocity> |
| **Units** | Motor Steps (EE=0)/Encoder Counts (EE=1) |
| **Range** | ±5000000 (EE=0)/±200000 (EE=1) |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `SL 20000 'Slew at a rate of 20000 steps/sec` |
| **Notes** | If 'SL 0' is issued after a MA/MR, motion has to come to a stop before issuing another motion command. This can be accomplished automatically with an 'H', <HOLD>, in user program mode. |
| **Related** | MA, MR, VI |

## SM (Stall detect mode)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | SM |
| **Function** | Stall detection mode |
| **Type** | Variable |
| **Description** | The SM variable specifies the action which will be taken by the device when a stall is detected. When set to 0 (default) the motion will be stopped upon a stall detection. When SM=1, the motor will try to continue the move. In either case ST (Stall Flag) will be set.<br>The functionality of SM when used with Position Maintenance (PM) is listed below:<br>If SM = 0 and PM = 1, Position Maintenance will take place provided the position does not exceed the Stall Factor (SF).<br>If SM = 1 and PM = 1, Position Maintenance will take place even if the Stall Factor (SF) is exceeded, unless VI is set too high causing the motor to stall. |
| **Syntax** | SM <mode> |
| **Modes** | 0 Motion stops on stall detect (default)<br>1 Motion will attempt to continue |
| **Default** | 0 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `SM=1 'Change Stall mode that the motor doesn't stop on stall detect` |
| **Related** | EE, SM, ST, PM, SF |

## SN (Serial number)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive Hybrid
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | SN |
| **Function** | Serial number |
| **Type** | Variable |
| **Description** | Keyword allows the user to read the device serial number. |
| **Syntax** | PR SN |
| **Usage** | Read |
| **Code Example** | `PR SN 'read the device serial number` |

## SS (System speed)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

| | |
|---|---|
| **Mnemonic** | SS |
| **Function** | Set system speed |
| **Type** | Instruction |
| **Description** | Sets maximum response frequency for fixed or variable current modes (AS=1 or 2) |
| **Syntax** | SS=<integer> |
| **Units** | steps/sec |
| **Range** | 38,910 — 5,000,000 |
| **Default** | 0 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `SS=50000 'set max. osc freq to 50000 'steps/sec` |
| **Related** | AS, TS, MU |

## ST (Stall flag)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | ST |
| **Function** | Stall flag |
| **Type** | Flag |
| **Description** | The ST flag will be set to 1 when a stall is detected. It is the responsibility of the user to reset it to zero (0). Encoder function must be enabled (EE=1) in order for a stall flag to set. |
| **Syntax** | PR ST<br>BR <addr>, ST=1<br>CL <addr>, ST=1 |
| **Status** | 0 Not stalled (default)<br>1 Stalled |
| **Default** | 0 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `CL K5,ST=1 'Call subroutine K5 if motor stalls`<br>`ST=0 'Clear stall flag` |
| **Related** | EE, SF, OE |

## SU (Execute program on power up)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | SU |
| **Function** | Execute program labeled SU on power up |
| **Type** | Predefined label |
| **Description** | The Start up label will cause any program labeled SU to automatically execute on power-up. |
| **Syntax** | LB SU |
| **Usage** | Program |
| **Code Example** | `LB SU 'Label program to executes on power up.` |

## TA (Trip on Hybrid status)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

| | | |
|---|---|---|
| **Mnemonic** | TA | |
| **Function** | Trip on hybrid status | |
| **Type** | Variable | |
| **Description** | Will execute a subroutine address or label on the trip. The trip can be set to occure on any or all of three conditions: calibration done, hybrid active, locked rotor or lead/lag limit reached conditions. | |
| **Syntax** | TA <flag>,<address/label> | |
| **Units** | — | |
| **Range** | 0 — 7 | |
| **Trip Condition** | 1 | Calibration done |
| | 2 | Hybrid active |
| | 4 | Locked rotor |
| | **8** | Lag limit reached |
| | 16 | Lead limit reached |
| **Default** | 0 (Off) | |
| **Usage** | Program/Immediate, Read/Write | |
| **Code Example** | `TA=4,k6 'execute subroutine k6 on locked rotor condition` | |
| **Notes** | The conditions are additive, eg. TA=3 will trip on calibration complete and hybrid active status.<br>There is no error generation when enabling trip on locked rotor, lag limit or lead limit. | |

## TC (Trip capture)

Compatible with Motion Control products:
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | TC |
| **Function** | Trip capture |
| **Type** | Variable |
| **Description** | Sets the Capture input trip for I/O 13. Sets one parameter for trip address. The TE command (Trip Enable/Disable TC) is reset when trip occurs. TE must be re-enabled in the main program prior to the next trip if it is to be repeated. The Trip subroutine must use a RETURN (RET) to exit the subroutine, use of a BRANCH will cause stack errors. |
| **Syntax** | TC=<address/label> |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `TC=K1 'Run subroutine K1 on Input Trip`<br>`TE=4 'Re-enable Trip` |
| **Related** | TE, S13 |

## TD (Torque direction)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

| | | |
|---|---|---|
| **Mnemonic** | TD | |
| **Function** | Set torque direction | |
| **Type** | Variable | |
| **Description** | Sets torque direction to + or – | |
| **Syntax** | TD=<dir> | |
| **Units** | — | |
| **Range** | 0 — 1 | |
| **Modes** | 0 | Minus (CCW facing shaft) |
| | 1 | Plus (CW facing shaft) |
| **Default** | 1 | |
| **Usage** | Program/Immediate,Read/Write | |
| **Code Example** | `TD=0 'set torque direction to minus` | |
| **Notes** | `Takes effect when torque mode starts` | |
| **Related** | AS, TQ, TS | |

# TE (Trip enable)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | | |
|---|---|---|
| **Mnemonic** | TE | |
| **Function** | Trip enable | |
| **Type** | Flag | |
| **Description** | The trip functions may be combined by adding trip numbers. For example TE=3 will trip on input or on position, TE=63 enables all trips. When multiple trips are used only the activated trip function needs to be re-enabled, the other trips will still be enabled. | |
| **Syntax** | TE=<trip> | |
| **Range** | 0 — 63 | |
| **Trip** | 0 | Disabled (default) |
| | 1 | Trip on input enabled |
| | 2 | Trip on position enabled |
| | 4 | Trip on capture (I/O 13) enabled |
| | 8 | Trip on time enabled |
| | 16 | Trip on relative position |
| | 32 | Trip on Hybrid status |
| **Default** | 0 (Off) | |
| **Usage** | Program/Immediate, Read/Write | |
| **Code Example** | `TE=1 'Enable trip on input function`<br>`TE=8 'Enable trip on time function`<br>`TE=6 'Trip on Position or Capture input.` | |
| **Related** | I1-I4, I9-I12, P, S1-S4, S9-S12, TI, TP, TC, TT | |

# TI (Trip on input)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive Hybrid
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | TI |
| **Function** | Trip on input |
| **Type** | Variable |
| **Description** | Sets up an input event (Trip) for the specified input. There are two parameters for the TI variable. The first specifies which input line to monitor. The second specifies the subroutine that should be executed when the input goes to true. The Trip subroutine must use a RETURN (RET) to exit the subroutine, use of a BRANCH will cause stack errors<br>The TE is reset when a Trip occurs. TE must be re-enabled prior to the next Trip if it is to be repeated. |
| **Syntax** | TI=<input>,<address/label> |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `TI=2,K3 'execute subroutine K3 when input 2 active`<br>`TE=1 'Re-enable Trip` |
| **Related** | I1-4, S1-4, TE, TP, TT |

## TP (Trip on position)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | TP |
| **Function** | Trip on position |
| **Type** | Variable |
| **Description** | Sets up an event (trip) for the specified position. There are two parameters for the TP variable. The first specifies the position which will cause the event. The second specifies the subroutine that should be executed when the position is detected.<br>The TE (Trip Enable which Enables/Disables TP) is reset when a Trip occurs. TE must be re-enabled in the main program prior to the next Trip if it is to be repeated. The Trip subroutine must use a RETURN (RET) to exit the subroutine, use of a BRANCH will cause stack errors.<br>Trips should be set BEFORE motion commands in the program. |
| **Syntax** | TP=<position>,<address/label> |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `TP=650000,K9  'exe sub K9 when position = 650000`<br>`TE=2  'Re-enable trip` |
| **Notes** | Note that TP will always use motor counts regardless of the encoder enabled state (EE=1) |
| **Related** | P, TI, PC, S13, CW |

## TQ (Set torque)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

| | |
|---|---|
| **Mnemonic** | TQ |
| **Function** | Set torque |
| **Type** | Variable |
| **Description** | Sets the maximum out put torque of the motor to a percentage. |
| **Syntax** | TQ=<percent> |
| **Units** | Percent |
| **Range** | 1 — 100 |
| **Default** | 25% |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `TQ=50  'set torque to 50%` |
| **Related** | AS, CT |

# TR (Trip on relative position)

Compatible with Motion Control products:
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | TR |
| **Function** | Trip on relative position |
| **Type** | Variable |
| **Description** | Sets up an event (trip) for the specified relative position. There are three parameters for the TR variable.<br><br>The first specifies the position which will cause the event.<br><br>The second specifies the subroutine that should be executed when the position is detected, if no subroutine address or label is specified then the High Speed Trip Output will activate. The Trip subroutine must use a RETURN (RET) to exit the subroutine, use of a BRANCH will cause stack errors<br><br>The third parameter specifies the number of times the trip will repeat. If 0 (default) the trip will repeat infinite times, other wise the range is 1- 65000<br><br>The TE (Trip Enable which Enables/Disables TR) is reset after repeating the number of relative trips specified. TE must be re-enabled in the main program prior to the next series of Trip on Relative if it is to be repeated. For exampl, if TR=10000,0,25, the Output (S13) will trip 25 times in succession at 100,000 counts relative to the last position. Following these 25 trips the trip must be re-enabled (TE=16).<br><br>Trips should be set BEFORE motion commands in the program. |
| **Syntax** | TR=<±dist>,<address/label>, <repeat> |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `TR=650000,0,0  'Activate trip out when at 650000`<br>`'counts relative`<br>`TE=16  'Re-enable trip`<br><br>`TR=512000,K2,27  'Run Sub K2 when at 512000 rel,`<br>`'repeat 27 times`<br>`TE=16  'Re-enable trip` |
| **Notes** | Note: Output S13 must be configured as a trip output (S13=61,1/0)<br><br>Note that TR will always use motor counts unless the encoder is enabled (EE=1).<br><br>Note: The maximum rate of trip is 20 kHz. Exceeding this may cause communications errors |
| **Related** | P, TI, PC, S13, CW |

## TS (Set torque speed)

Compatible with Motion Control products:
MDrive Hybrid
MDrive Hybrid (Ethernet)

| | |
|---|---|
| **Mnemonic** | TS |
| **Function** | Set torque speed |
| **Type** | Instruction |
| **Description** | Determines the system speed for torque mode (AS=3)<br>Hybrid will perform the following calculation based upon the value of TS: |
| **Syntax** | TS=<integer> |
| **Units** | steps/second |
| **Range** | 38,910 — 5,000,000 |
| **Default** | 0 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `TS=50000 'set spd to 50000 steps/sec` |
| **Related** | AS, SS, MSEL |

## TT (Trip on position)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | TT |
| **Function** | Trip on time |
| **Type** | Variable |
| **Description** | Sets up a trip based on time. The first parameter is time in mSec. The second parameter specifies the subroutine that should be executed when the time is expired. The Trip subroutine must use a RETURN (RET) to exit the subroutine, use of a BRANCH will cause stack errors |
| **Syntax** | TT=<time>,<address/label> |
| **Umits** | Milliseconds |
| **Range** | 1 to 65535 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `TT=2000,K3 'Trip on time 2000 mS, run- sub K3`<br>`TE=8 'Re-enable trip` |
| **Related** | TE, TP, T |

## UG (Upgrade firmware)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | UG |
| **Function** | Upgrade firmware |
| **Type** | Instruction |
| **Description** | Upgrade Firmware Instruction. Upgrade code is 2956102. This will put the device in Upgrade Mode. Once set, the firmware Upgrade MUST be completed. |
| **Usage** | Immediate |

## UV (Read user variables)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | UV |
| **Function** | Read user variables |
| **Type** | Variable |
| **Description** | Read User Variables is used with the PR (Print) Instruction to read the value of all user variables. |
| **Syntax** | PR UV |
| **Usage** | Program/Immediate, Read |
| **Code Example** | `PR UV  'Read the value of all user variables` |
| **Related** | PR, VA |

## V (Read velocity)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | V |
| **Function** | Read axis velocity |
| **Type** | Variable |
| **Description** | The velocity variable is used in conjunction with the PR (print) instruction to read the current velocity of the axis in counts per second. This variable can also be used with the BR and CL instructions to set a condition based upon a velocity. |
| **Syntax** | PR V<br>BR <address/label>, V=<velocity><br>BR <address/label>, V=<velocity> |
| **Usage** | Program/Immediate, Read |
| **Code Example** | `PR V  'Read the velocity`<br>`CL Ka, V=20000  'Execute sub Ka when velocity is`<br>`'20000/steps sec in the + direction`<br>`CL Kb, V=-20000 'Execute sub Kb when velocity is`<br>`'-20000/steps sec in the - irection` |
| **Notes** | Note that V is signed. When using an Hybrid product V will not return an accurate value when Hybrid make up is active. In torque mode V will return 0. |
| **Related** | VI, VM, SL, MA, MR |

## VA (Create user variable)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | VA |
| **Function** | Create user variable name |
| **Type** | Instruction |
| **Description** | The VA instruction creates a user variable with a 1 or 2 character name. Can optionally set value assigned to that variable.<br>The restrictions for this command are:<br>1. A variable cannot be named after a MCode Instruction, Variable or Flag or Keyword<br>2. The first character must be alpha, the second character may be alpha-numeric.<br>3. A variable is limited to two characters.<br>4. Limited to 192 variables and labels. |
| **Syntax** | VA <char><char>=<value> |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `VA Q3 'Create user Variable Q3`<br>`VA Q3=20000 'Create user Variable Q3, set to 20000` |
| **Related** | UV |

## VC (Velocity changing)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | | |
|---|---|---|
| **Mnemonic** | VC | |
| **Function** | Velocity changing | |
| **Type** | Flag | |
| **Description** | The read-only motion flag will be at an active state (1) when the velocity of the motor is changing, either accelerating or decelerating. | |
| **Syntax** | PR VC<br>BR <addr>, VC=1<br>CL <addr>, VC=1 | |
| **Range** | 0/1 | |
| **Trip** | 0 | Motor stopped or at constant velocity (default) |
| | 1 | Velocity is changing |
| **Default** | 0 (not active) | |
| **Usage** | Program/Immediate, Read/Write | |
| **Code Example** | `CL K5,VC=1 'Call sub K5 if velocity is changing`<br>`PR VC 'Print the state of the VC Flag` | |
| **Related** | MV, VI, VM, S1-S4, S9-S12 | |

## VF (Torque velocity filter)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | VF |
| **Function** | Set torque velocity filter |
| **Type** | Variable |
| **Description** | VF takes a value of 0 to 1000. It can be defined as 0 = no filtering and 1000 = most filtering.<br>Because the Torque Velocity is computed and the encoder is sampled every mSec there can be fluctuation in the result. The filtering compensates for this fluctuation. |
| **Syntax** | VF=<integer> |
| **Units** | Counts |
| **Range** | 0 to 1000 |
| **Default** | 0 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `VF=100 'Set filtering to 100 counts` |

## VI (Initial velocity)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | VI |
| **Function** | Initial velocity |
| **Type** | Variable |
| **Description** | Initial velocity for all motion commands. The factory default value is 1000 clock pulses (steps) per second.<br>The initial velocity for a stepper should be set to avoid the low speed resonance frequency and must be set lower than the pull in torque of the motor. It must also be set to a value lower than VM (Max. Velocity). |
| **Syntax** | VI=<velocity> |
| **Units** | Motor Steps (EE=0)/Encoder Counts (EE=1) |
| **Range** | 1 to (VM –1) |
| **Default** | 1000/40 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `VI=10000 'Set initial vel to 10000 steps/sec.` |
| **Related** | VM, MR, MA, HI, HM, JE |

## VM (Maximum velocity)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | VM |
| **Function** | Maximum velocity |
| **Type** | Variable |
| **Description** | The VM variable specifies the maximum velocity in steps/counts per second that the axis will reach during a move command.<br>VM must be greater than VI. |
| **Syntax** | VM=<velocity> |
| **Units** | Motor Steps (EE=0)/Encoder Counts (EE=1) |
| **Range** | (VI + 1) to 5000000 (EE=0)/200000 (EE=1) |
| **Default** | 768000/30720 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `VM=500000 'Set max vel to 500000 steps/counts sec.` |
| **Related** | VI, MR, MA, HI, HM, JE |

## VR (Firmware version)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | VR |
| **Function** | Firmware version |
| **Type** | Variable |
| **Description** | This variable is used in conjunction with the PR instruction to read the version of the firmware installed at the factory. |
| **Syntax** | PR VR |
| **Usage** | Read |
| **Code Example** | `PR VR 'Read the firmware version installed` |
| **Notes** | `Hybrid products will return a second value for the hardware version.` |
| **Related** | UG |

# WT (Warning temperature)

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive (Ethernet)
MDrive Hybrid
MDrive Hybrid (Ethernet)
MForce
MForce (Plus² expanded features)

| | |
|---|---|
| **Mnemonic** | WT |
| **Function** | Warning temperature |
| **Type** | Variable |
| **Description** | The Warning Temperature variable allows the user to set a threshold temperature at which the device will print an error 71 to the terminal screen if the set temperature is exceeded. |
| **Syntax** | WT=<temp> |
| **Units** | Degrees C |
| **Range** | 0 to 84 |
| **Default** | 80 |
| **Usage** | Program/Immediate, Read/Write |
| **Code Example** | `WT=75 'set the warning temperature to 75 deg. C` |
| **Notes** | NOTE: This functionality is only standard on MDrive 34 (DC and AC), MForce PowerDrive and all Hybrid devices. |
| **Related** | IT |

# 6 Supporting Software

There are two programs provided for configuring, controlling and programming the Motion Control products. These are:

1. **IMS Terminal:** integrated ASCII program editor/ANSI terminal emulator used to program and immediate mode control the Motion Control products.
2. **TCP/IP Configuration Utility:** configuration tool used to setup and test the Ethernet MDrive Motion Control products. When using the product in MCode/TCP mode, this tool is primarily used to set the device IP address and test connectivity and functionality before using IMS Terminal for configuration and programming.

## 6.1 IMS Terminal

IMS Terminal is an Integrated Program Editor and Terminal Emulator designed to communicate with and program IMS products including the MCode devices, the Motion Control MDrive, MForce and Hybrid products. For purpose of this document the focus will be on the devices utilizing the MCode language.

The upgrader utility included with IMS Terminal is required if you desire to upgrade the firmware in your MCode device.

### 6.1.2 Features

*General features*

- Multiple program editor windows allowed.
- Multiple Terminal Windows allowed which can be connected to multiple devices and device types.
- Configurable initialization file (lynxterm.lxt) allows Program Editor and Terminal Window states and configurations to be remembered and opened upon startup.

*Program editor window*

- Color-coded text to easily differentiate command types.
- Auto-indent for program blocks.
- Lines of code my be commented using the apostrophe (‘) character.
- Files may be text (*.txt) formatted or MCode (*.mxt) formatted. Note the color-coded text is only available in the MCode format.
- Color-coding, indenting, font type, size and style are user configurable through the preferences dialog.

*Terminal emulator window*

- Programmable function keys set in groups of ten.
- Multiple function key groupings.
- Special “Capture Mode” allows the capture of all terminal communications to a text file.
- Terminal Window Status, i.e. Connected/Disconnected, Port #, BAUD Rate and etc. displayed on a clickable bar across the bottom of the Terminal Window.

## 6.1.3 Installing IMS Terminal

*System requirements*

- IBM Compatible PC.
- Windows XP Service Pack 2 or greater.
- A free USB or serial communications port.

*Installation*

1) Download the software from the IMS web site at http://motion.schneider-electric.com/downloads/software_interfaces.html.
2) Extract to a location on your hard drive.
3) In the folder location of the extracted files, double click “setup.exe”
4) Follow the on-screen prompts to complete the installation of IMS Terminal.

## 6.1.4 Screen and menu overview

*IMS Terminal main screen*

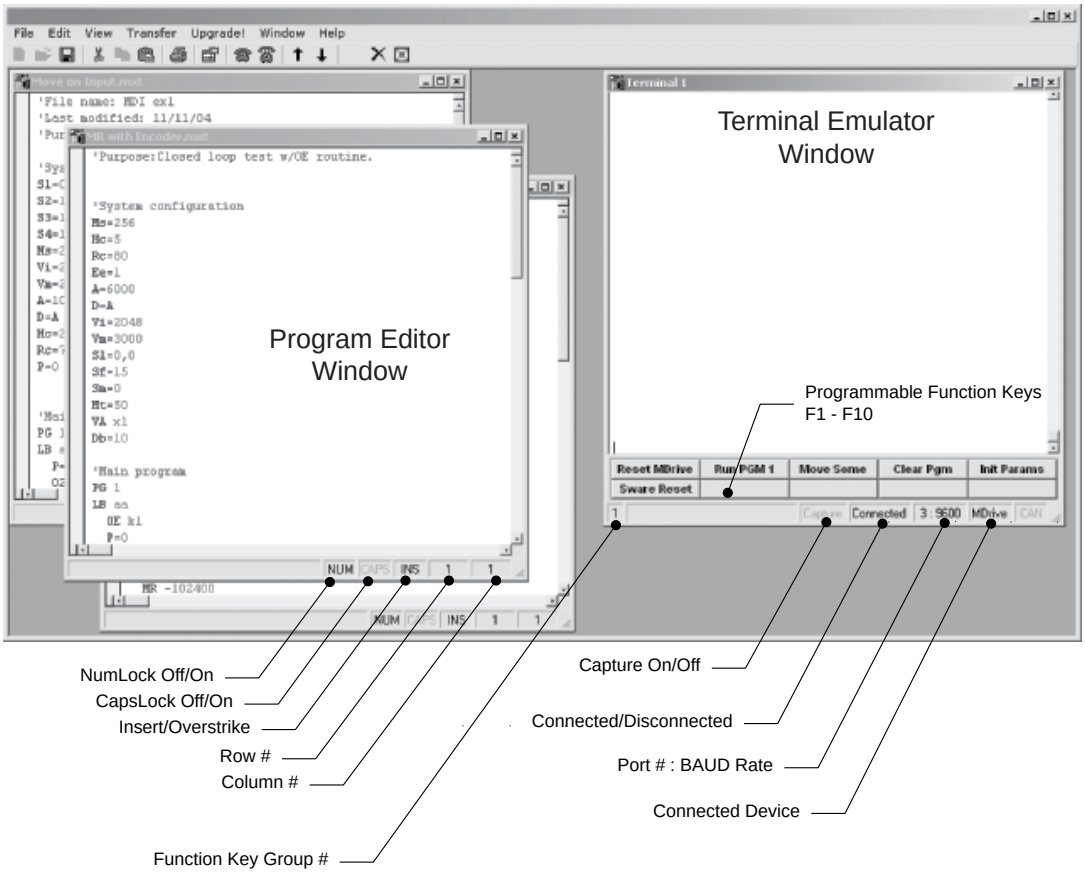



Figure 6.1 IMS Terminal main screen

## 6.4.2 The menu bar structure and operation

Most of the functions of the IMS Terminal are accessible through the menu bar. Please note that some of the menu bar functions will differ based upon the type of window which is active, either Program Editor or Terminal.

*File menu functions*

File menu functions functions, when used in the program editor, are common to most windows programs. When the terminal window is in an active state, the new, open and save items will disable. Save as will allow the user to save the contents of the terminal window, including the scroll back buffer, if desired to a text file.

## 6.4.3 Edit menu operation

As with the file menu, many of the edit menu functions are familiar to windows users such as redo, undo, copy, cut, paste, delete, select all, find, find next and replace. These will function as they do with other windows-based software programs. Of these, only copy and paste will be available if a terminal window is active.

*Preferences*

The preferences item will open a tabbed dialog with the various window configuration settings. If selected with the program editor window active, it will display the font, color, style selections for the program window.

If selected with the terminal window active, it will display the visual configurations settings for the terminal window. These configurations will be discussed in detail later in this section.

***Open preferences***

The preferences settings are all saved in a single *.ltm file, the default being lynxterm.ltm. These files contain the following preference settings:

- Text editor preferences
- Program editor preferences
- Terminal format preferences
- Terminal window communications settings
- Open program or text files
- Open terminal windows
- Function key and group configuration

IMS terminal will automatically save the settings on exit to the filename you have loaded.

***Save preferences***

- Save preferences will save the current preferences to the loaded file.
- Save preferences as
- Save preferences as different *.ltm filename.

*View menu operation*

The only applicable functions to MCode devices are the menu items:

- New edit window
- New terminal window

The operation of these options are covered later in this section

*Transfer menu operation*

The transfer menu controls the transfer of data to and from the MCode device.

**Upload**

Upload will open a dialog which will allow the user to selectively transfer the stored global variable and flag declarations and programs to a new or open program editor window.

**Download**

The download menu item will open a dialog allowing the user to selectively download variable and flag declarations and programs to the MCode compatible device from either an open program editor window or a saved *.mxt file.

**Capture**

The capture menu item will place the active terminal window into a special “capture mode”. When in capture mode all terminal activity is captured to a text file in real time. The on/off state of capture mode is displayed on the bottom indicator bar of the active terminal window.

When in capture mode, two additional menu items appear on the transfer menu:

- Pause capture
- Stop capture

These will function as described by their label.

*Upgrade menu operation*

The upgrade! menu item will open the IMS Terminal upgrader utility for upgrading the firmware in your MCode device. It will not place your device in the upgrade mode for upgrading.

*Window menu operation*

The Window menu offers the options common to most Microsoft Windows™ programs.

## 6.1.5 Configuring IMS Terminal

*Configuring program editor format preferences*

The program editor features a number of enhancements designed to aid the user in programming IMS MCode products by the use of color-coded text and automatic tabbing to separate program blocks and subroutines for easier code editing and debugging.

While the default format of the program editing window is sufficient for most users, you may configure the format to your preference.

To open the program editor preferences dialog:

1) Right-click into the program editor window.
2) Select preferences.
3) The program editor format tab of the preferences dialog will open.
4) Set the format to that which you desire.

Note that the use of a mono-spaced font such as the default, courier new, makes code editing and debugging much easier than using a variable spaced font.

*Configuring terminal window format preferences*

The terminal window features similar formatting preferences as the program editor.

1) To open the terminal format preferences dialog:
2) Right-click into the terminal window.
3) Select preferences.
4) The terminal format tab of the preferences dialog will open.
5) Set the format to that which you desire.

*Configuring communications settings*

The communications settings are configured by means of the preferences dialog.

The optimum communications settings for the MCode compatible device are set by default. The only thing that will need to be set to begin communicating with your MCode device is to set the COM port to which your rs-422/485 communications converter is connected, or the IP address if using an Ethernet based product (default 192.168.33.1).

1) Open the communications preferences dialog by either opening the preferences from the menu bar and selecting the comm settings tab, or double clicking the port: baud rate field on the status bar at the bottom of the terminal window.
2) Select the device you are communicating with:
   MDI - MDrive or MForce
   ASI - Hybrid
   The communication settings will automatically be set for the device selected.
3) The window size and function key settings are optional.
4) Once you have selected the correct com port, click ok.
5) Verify hardware connections and apply power to the MCode device.
6) If not already connected, connect to the device by clicking the “connect” icon on the button bar, or by double clicking the disconnected field on the status bar of the terminal window.
7) Key in ctrl+c.
8) The sign-on message below should appear.

The sign-on message should appear:

```
Copyright 2001 - 2010 by Schneider Electric Motion USA
>
```

The sign-on message indicates that you are up and running. You may now begin to issue immediate mode commands and/or download programs to your device!

## 6.1.6 Troubleshooting communications

*Troubleshooting the communications converter*

1) Go to the start menu on Windows XP.
2) Right click “my computer”, select “properties”.
3) On the system properties dialog, click the tab labeled “hardware”.
4) Click the device manager button.
5) On the device manager window, click the “+” sign next to ports (com and lpt) to expand the category.
6) The RS-422 converter should be listed there with its port number.
7) Verify that the port is the one configured in the communications settings for IMS Terminal.
8) If the communications converter does not show in the device manager verify that the drivers for the converter are installed per the converter manufacturer documentation.
9) If the converter will not install, contact the device manufacturer support desk.
10) Communications converter installed and functioning
11) Verify that all mating connectors are firmly seated.

If not using an IMS MD-CC40x-001 and appropriate adapter, verify the wiring.

Check the following connections:

| Converter | MCode Device |
|---|---|
| RX+ | TX+ |
| RX- | TX- |
| TX+ | RX+ |
| TX- | RX- |
| GROUND | CGND |

## 6.1.7 Configuring function keys

The ability to program MCode functions and code strings and assign a function key to them is one of the most powerful features of IMS Terminal.

Function Keys can also be grouped in whatever fashion the user desires. IMS Terminal supports up to 999 function groups.

The illustration below shows the function key dialog with example functions programmed in.

To open the function key dialog:

Right-click any of the function keys on the bottom of the terminal window or key-in CTRL+[F1-F10].

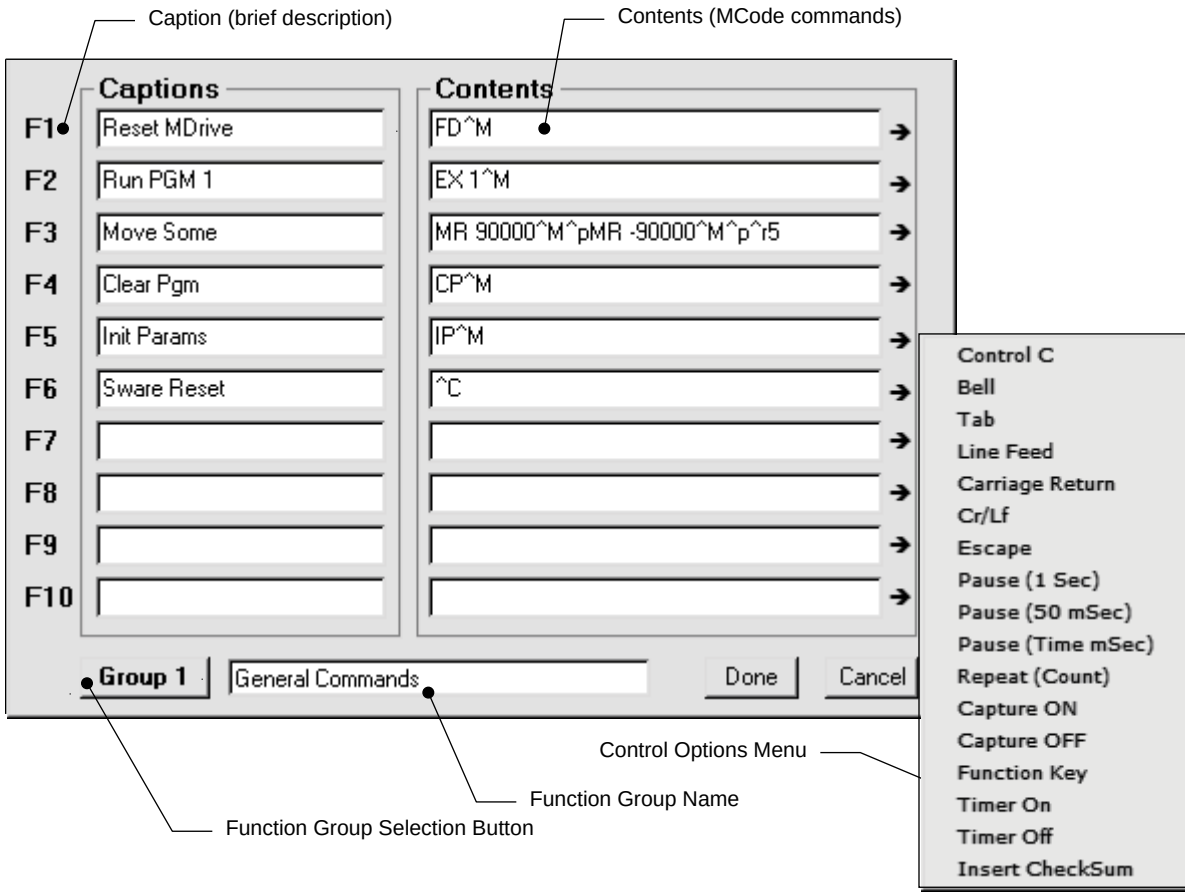



Figure 6.2 Function key setup dialog

*Function key control commands*

The IMS Terminal function keys have a number of control commands that are used to input or impact MCode strings sent to the terminal on function key press.

| Function | Characters | Description |
|---|---|---|
| Control C | ^C | MCode Software Reset |
| Bell | ^G | Computer Beep |
| Tab | ^t | Tabs cursor 5 spaces |
| Line Feed | ^J | Sends a Line Feed |
| Carriage Return | ^M | Sends a Carriage Return |
| Cr/Lf | ^M^J | Sends a Carriage Return with a Line Feed |
| Escape | ^[ | Sends an Escape |
| Pause (1 Sec) | ^p | Pauses operation between command execution for 1 second |
| Pause (50 mSec) | ^m | Pauses operation between command execution for 50 milliseconds |
| Pause (Time mSec) | ^d<x> | Pauses operation between command execution for <x> milliseconds |
| Repeat (Count) | ^r<x> | Repeats the string <x> times |
| Capture ON | ^c | Turns on the IMS Terminal Capture Feature |
| Capture OFF | ^o | Turns off the IMS Terminal Capture Feature |
| Function Key | ^f<1-10> | Activates Function Key <1-10>. Can be used to send multiple command strings from the functions. |
| Timer On | ^t1 | Activates Timer. Time will display on the status bar. The Time will not update until the Timer is turned off |
| Timer Off | ^t0 | Deactivates Timer, updates time display on status bar. |

Table 6.1 Function key control commands

## 6.1.8 Creating, downloading and uploading programs

Existing programs may be edited in the program editor window from a file on a disk, a file on the hard drive or a file uploaded from an MCode compatible device. You may also create a new program in the program editor window.

Note: your system must be connected and running to perform these steps as they are outlined.

*Creating a new program*

Before you create a program you must have a new program editor window open. Follow these steps:

1) Click on the drop-down menu “view”. The following dialog box will be displayed:
2) Click on “new edit window”.
3) You must assign a file name in order to open the new window. If there is no file name the “ok” button will not be highlighted. Name this file <my program.mxt>. The <mxt> extension designates MCode programs.
4) Click “OK” and the new program editor window will be displayed.

Naming the program with the <mxt> extension automatically formats the text color and makes most of the characters appear in upper case. When you type a program the text will be color coded. In complex programs it may be difficult to read the text easily. By formatting indents, the overall appearance and readability will be greatly improved.

*Downloading a program to the device*

NOTE: Before downloading any programs type FD into the terminal window and press ENTER to set the device to the factory defaults.

There are two basic sources from which you can download programs to the MCode compatible device:

1) Directly from the program editor window of the IMS Terminal.
2) From a file folder located on a hard drive or removable disk.

**To download**

1) Activate the Program Editor Window containing the Program MR.mxt.
2) Click the menu item “Transfer > Download”. The download dialog box will open.
3) Click the download button on the main tool bar. The download dialog box will open. Select the “Source Type > Edit Window” option, and click download. The program will transfer to the device. If a program has been previously created and stored, it may be downloaded to the device from the *.mxt file.
4) Once the program is downloaded, type S and press ENTER to

Save the program. (Always save your programs!)

5) Now type EX 1 and press Enter or Click the Function button Run PGM 1. (EX=Execute and 100 is the Program Number.) The motor should move a short distance back and forth.

NOTE: The program can be stopped by pressing the escape button or by pressing <Ctrl+C>.

*Uploading a Program*

NOTE: Be certain the program is stopped by pressing the Escape Button or by pressing <Ctrl+C>.

There are two ways to upload programs from the MCode compatible device:

1) Directly to the program editor window of the IMS Terminal.
2) To a file folder located on a hard drive or removable disk.

There are also two ways to enable the upload dialog box.

1) Click the menu item “Transfer > Upload”. The upload dialog box will open.
2) Click the upload button on the main tool bar. The upload dialog box will open. The upload dialog box is similar in appearance to the download dialog box.

With the upload dialog box open, select the “Destination Type > Edit Window” option, click “Upload”. The program will transfer from the device.

Programs may also be uploaded from the device directly to a text file by selecting “Destination Type > “File” as the destination and typing in a filename in the “File Name” box on the dialog box.

NOTE: When uploading MCode program files they will be slightly changed from the original. The device will upload the program only with the data within the program. That is, the data between the two program modes (PG). Data such as variables entered outside the PG modes will not be uploaded. The uploaded program will also have a header ‘[PROGRAMS] and a footer ‘[END]. These will not affect your program as they are remarked with the apostrophe (‘) or they can be removed during editing.

You may upload the program variables by clicking “Variables” in the upload dialog box. However, this will upload all of the current variables, not just those associated with the program.

## 6.1.9 Program troubleshooting using IMS Terminal

The IMS Terminal offers tools to help you troubleshoot and analyze programs. They are:

- Execute in single step mode
- Execute in trace mode
- The scroll back function
- The capture function

*Single step mode*

The single step mode allows the user to execute a program in the immediate mode one line at a time. This will help the user to define problem areas by process of elimination. To use single step mode, do the following:

It is recommended that you list (L) the program in the terminal window and either print it on paper or cut and paste it to another program edit window. This will allow you to look ahead and see what line is coming up next.

1) Have the system and the program ready to run.
2) To run in single step mode add a comma and the number two (2) to the execute command.

Example: the program label is <aa>. Type EX aa,2. The program will run one line at a time.

3) Each line will be executed and listed in the terminal window and the program will stop.
4) To execute and list the next line, press the space bar.
5) Press the space bar for each successive line until the program has completed.

While the program is executing, it will stop after each line is listed. At this time you may enter immediate commands such as velocity variables or actual moves as tests within the program. After entering immediate commands you may continue running in single step mode by pressing the space bar again.

If you decide to cancel the single step mode press the “enter” key and the program will run in normal mod and finish or press escape (esc) to abort the program.

*Trace mode*

The trace mode allows the user to run a program and list each line as it is executed. Running trace mode in conjunction with the scroll back function or the capture function will enhance your program troubleshooting tasks. To run trace mode:

1) Have the system and the program ready to run.
2) To run in Trace Mode add a comma and the number one (1) to the execute command.

   Example: the program label is <aa>. Type ex aa,1. The program will run in trace mode and each line will be executed and listed in the terminal window.

3) Each line can now be analyzed.

On very large programs all of the lines may not be displayed if the “Scroll Back Buffer” value is set too low. The Scroll Back Buffer can be set to a higher value allowing you to Scroll Back farther in the program.

*The capture function*

The capture function allows you to capture terminal communications into a text file for the purpose of troubleshooting. You may have a program that fails after running a number of times. It may be from an accumulation of position errors or other factors. By enabling the capture function you can store an entire text file of the received communications to your hard drive for analysis.

**Enable the capture function**

The Capture function may be enabled through the drop-down menu under “Transfer”.

When you click on “Capture” a dialog box will be displayed.

Give the file you will be capturing a name and be certain to save it as a [ .txt ] file and click “Save”. Upon clicking Save, the faded (disabled) Capture title below the Function Keys will change to “Capture ON” and to black letters.

You are now ready to run the program. The program in this example will cycle five (5) times. The data will scroll up the Terminal Window while a copy of the data is captured into the text file simultaneously.

### 6.1.10 Upgrading firmware

*Before upgrading the MCode firmware*

IMPORTANT! It is recommended that you review this procedure in its entirety before performing the upgrade.

It is recommended that the most recent version of IMS Terminal Software be installed on your PC prior upgrading the firmware.

To check if you have the most recent version of IMS Terminal Software, click the “HELP” menu item on the IMS Terminal menu bar and then click “About IMS Terminal”. The following information block will appear.

The current version of your IMS Terminal Software will be shown as indicated by the arrow. Compare this version number with the IMS Terminal version number found on the IMS web site at motion.schneider-electric.com/software_interfaces.html. If a more recent version is shown on the web site, you should download and install it on your system before upgrading the firmware.

NOTE: The file you will be downloading is a self-extracting executable file. Download it to your desktop or a known folder.

To install the most recent version of IMS Terminal Software on your system perform the following steps:

NOTE: Skip Steps 1 & 2 if this is a new installation.

1) Open Windows Explorer and proceed to the folder “Program Files”.
2) Locate the folder named “IMS Terminal” and rename it to “IMS TermOLD”. This will preserve any files you want to save which can be retrieved later and it will also ensure a complete new installation of IMS Terminal.
3) Locate the downloaded version of IMS Terminal Software and Double Click the file.
4) A message regarding sharing files will appear. All other applications should be closed. Click OK.
5) In the window that follows, click the button to the left of the message to continue.
6) A dialog box will query you as to which program group you want IMS Terminal to be associated. Click CONTINUE to accept the default.
7) The installation will begin followed by the “Installed Successfully” message box. Click OK and the system is ready.

*Upgrading the firmware*

NOTE: Your MCode compatible device is configured with the most recent firmware at the time of shipment. The main reason for upgrading is to take advantage of new features that your system may need or to correct minor errors that may be causing problems in your system. Albeit, new features and corrections may be appealing, they may have little or no affect on your system operation. If your system is operating as it should, be hesitant about upgrading the firmware for the sake of “upgrading”. Before performing the upgrade procedure, verify the firmware version.

With the system running, type <pr vr> in the Terminal Window and press ENTER. The device will return the firmware version number. Compare this number with the latest version on the IMS web site at motion.schneider-electric.com/flash_code.html.

While at the web site, review the Change Summary for that version of the firmware. If none of the changes will help to correct a problem you may be having or improve your system operation, it is not necessary to upgrade.

Many problems are the result of programming errors. Verify that you do not have a programming problem that may mislead you to believe there is a problem with the firmware or your system.

If it is determined that a firmware upgrade is necessary, download the most recent version into a known folder from http://motion.schneider-electric.com/downloads/firmware.html.

During upgrades, the communication baud rate is switched from 9600 to 19,200 and is more susceptible to electrical noise. Your communications cable should be kept to a minimum length of 6 feet.

When using a laptop PC it is recommended that you power the RS-232 to RS-485 cable with an external +5 VDC power supply. This will fortify communications.

The device remains in the upgrade mode until the upgrade is complete. Cycling power will not clear the upgrade mode.

It is recommended that you use this procedure as it is tailored for the device while the on-screen instructions are designed for several different products.

1) Open “IMS Terminal”. The following screen should be displayed. The left panel is the program edit window and the right panel is the terminal window. The firmware upgrade will superimpose several dialog boxes and instructions over these two windows.
2) Check to see that the terminal window is set for communication.
   - Right click in the terminal window.
   - Click “Preferences” near the bottom of the pop-up menu.
   - A “Preferences” dialog box will be displayed.
   - Click on the “Comm Settings” tab at the top of the box.
   - Confirm that device is selected in the “Devices” block.
     MDI - MDrive or MForce
     ASI - Hybrid
3) Power up the device.
   - The sign on message will appear.

     “Copyright 2001-20010 by Schneider Electric Motion USA.”
4) Check and/or reestablish communications if the sign on message does not appear.
5) Type UG 2956102 in the terminal window and then press <enter>. Include the space between the G and the 2.
   - The device will return a random symbol character (ô or ö) when it is in the upgrade mode.
6) Click the “Upgrade” menu item on the IMS Terminal menu bar.
7) Message appears: “During upgrade, the baud rate is changed to 19,200.”
   - Click “OK”
8) The Windows Explorer page “Select device upgrade file” opens.
   - Browse and select the desired version of the upgrade file.
   - Click “Open” or double click the file.
9) Message appears: Step 2 Select upgrade file.
   - The upgrade version will now appear in the upgrade version window.
   - Click “Next”
10) Message appears: Step 3 Reminder Press cancel if you need to setup COMM port.
    - The COMM port has been setup previously. This is just a reminder.
    - Click “Next”
11) Message appears: Step 4 Connect RS-422 cable to the device.
    - The RS-422 has been connected previously. DO NOT perform this step.

- Click “Next”

12) Message appears: Step 5 If device is not in the upgrade mode, press cancel then type ‘UG 2956102’ in the terminal window.
    - The device was placed in the Upgrade mode previously. DO NOT ENTER CODE AGAIN.
    - Click “Next”
13) Message: Step 6 Power up or cycle power to device.
    - The unit has been previously powered up. Do not cycle power.
    - Click “next”
14) Message: Step 7 Establishing COMM with device.
    - Wait for step 8 to appear.
    - The previous version of firmware will now be displayed in the “Previous Version” window.
15) Message: Step 8 Press upgrade button to start.
    - Click the upgrade button.
    - NOTE: An upper case E will be displayed in the “Type Number” window. This confirms the upgrade is functioning properly.
16) Message: Step 9 Press ABORT to abort upgrade.
    - DO NOT abort the upgrade. The device remains in the upgrade mode and the upgrade must be completed.
    - Monitor the progress in the “Upgrading…%” window.
    - Step 10 will appear when DONE
17) Message: Step 10 Resetting device. Then Press DONE.
    - Click “DONE”
    - Upgrade window will close.
18) Press “Control + C” <Ctrl + C> while the Terminal Window is active to reset the device and exit the upgrade mode.
    - The sign on message will appear. “Copyright 2001-2008 by Intelligent Motion Systems, Inc.”
    - The > cursor will appear.
19) The device firmware has been upgraded.
20) Optional confirmation of the upgrade: Type “PR VR” in the terminal window and press <enter>.
    - The new firmware version is displayed.

## 6.2 TCP/IP Configuration Utility

The TCP/IP Configuration Utility is specifically for setting the Ethernet MDrive Motion Control IP address and basic functionality testing for use with MCode/TCP. We recommend using IMS Terminal for programming and controlling the MDrive after initial TCP/IP configuration.

### 6.2.1 Installation

*System requirements*

- IBM Compatible PC.
- Windows XP Service Pack 2 or greater.
- Ethernet hub with free port.

*Installation*

1) Download the software from the web site at http://motion.schneider-electric.com/downloads/software_interfaces.html.
2) Extract to a location on your hard drive.
3) In the folder location of the extracted files, double click “setup.exe”
4) Follow the on-screen prompts to complete the installation.

### 6.2.2 Screen overview

Configuration SEM Port (Terminal)

Figure 6.3 TCP/IP Configuration Utility

For MCode/TCP two of the 4 tbs on the GUI are available: The MODBUS and Manual tabs will generate an error dialog if you attempt to access them in MCode/TCP mode.

1. Config: Used to read the device MAC address and read/write the IP address and netmask.
2. SEM Port: a lightweight terminal emulator for testing the communication setting.

### 6.2.3 Configuration

The primary function of the utility in MCode/TCP mode is to configure the device IP address and Subnet Mask.

If you are on a corporate network, you may need to check with your IT department to obtain a block of private IP addresses so as not to conflict with computers and other devices on the network.

The assigned IP address should always be within the IPv4 Private Network block (192.168.0.0 — 192.168.255.255).

MCode/TCP will always use port 503 and may communicate via TCP or UDP interchangeably.

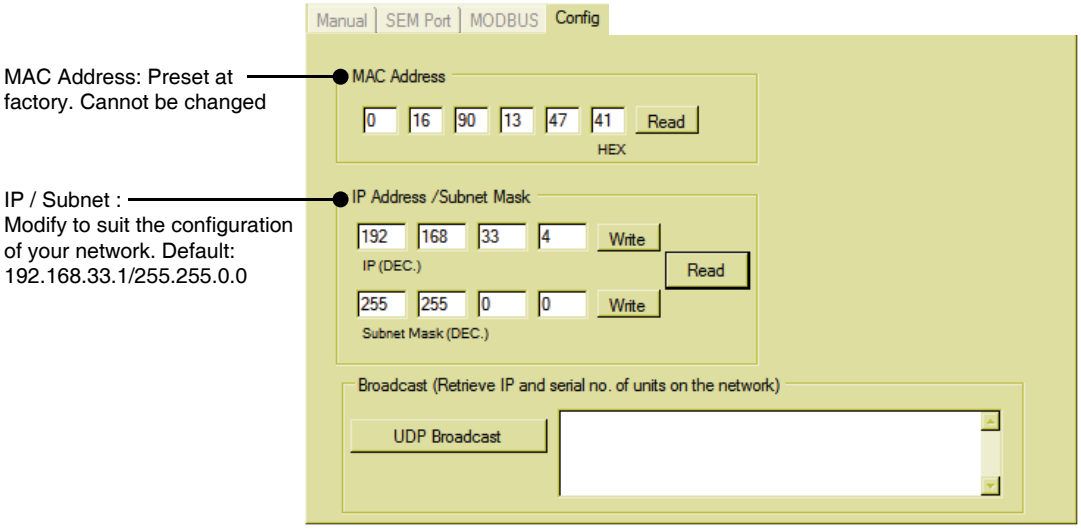



Figure 6.4: Configuration tab

### 6.2.4 SEM Port tab

This tab is used to test the functionality of communication interface and can submit commands to the MDrive

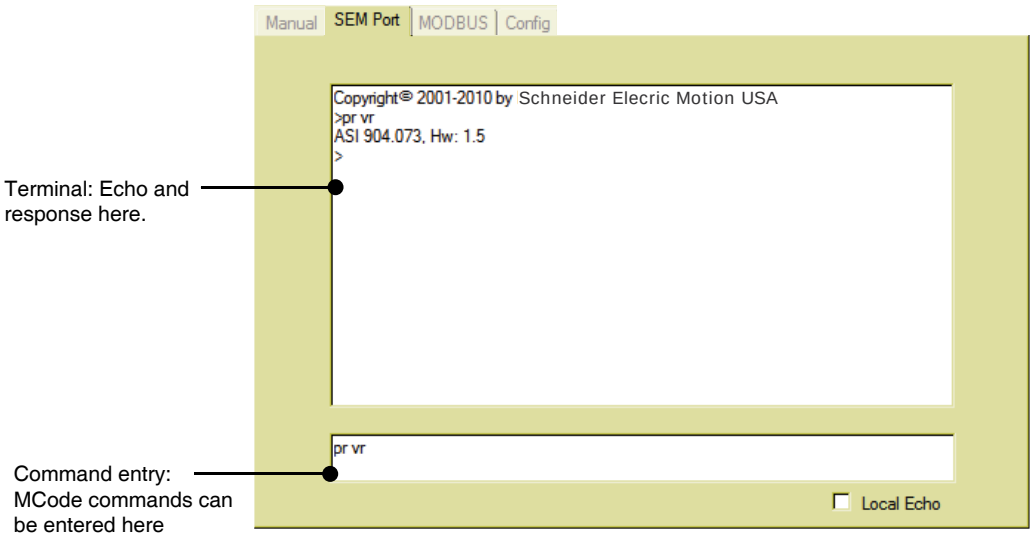



Figure 6.5: SEM Port tab

## 6.3 Upgrading the Ethernet controller firmware

NOTE: This refers strictly to the controller firmware for the Ethernet interface of Ethernet equipped MDrivePlus and MDrive Hybrid models. It is NOT an upgrade to the MDrive operating firmware.

It is recommended that you DO NOT perform this upgrade unless so instructed by the IMS SEM Applications department.

Please review this in detail before performing the upgrade, each step must be completed in order.

*Requirements*

The latest versions of the software and firmware are available on the web site under the downloads tab at http://www.schneider-electric-motion.us

1) MDrivePlus or MDrive Hybrid Motion Control with Ethernet
2) TCP/IP Configuration Tool (Installed)
3) TFTPD Firmware Server (Installed)
4) Ethernet firmware upgrade file
   IMPORTANT: Unzip upgrade *.S19 file to the installation directory of the TCP/IP Configuration Tool

This process will utilize the firmware upgrade area on the configuration tab of the TCP/IP Configuration Tool to set up.

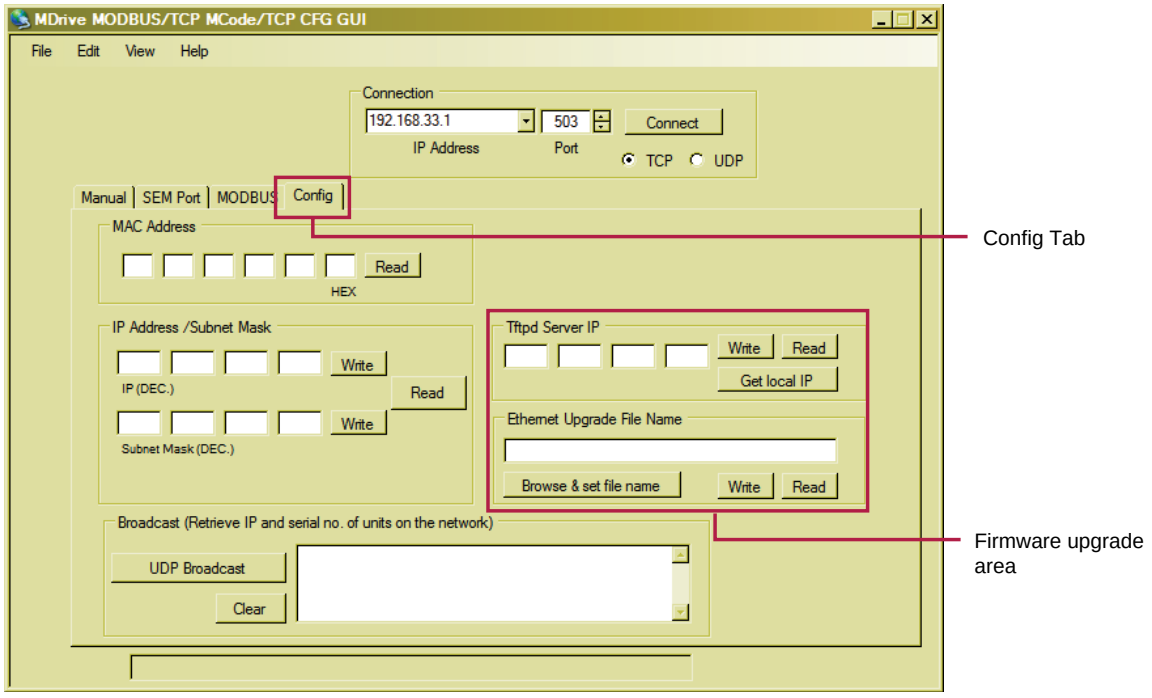



Figure 6.6: TCP/IP Config tab

### 6.3.1 To begin

1) Open the TCP/IP Configuration Tool
2) Click the config tab, if not already active.
3) Connect to your Ethernet MDrive over TCP.

### 6.3.2 Set the Tftpd Server IP

a) Click “Get Local IP”
b) Click “Write”
c) Tftpd Server IP should read ‘OK’

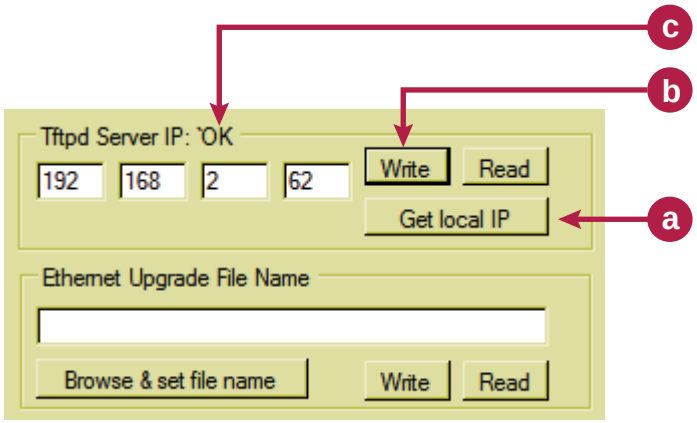



Figure 6.7: Setting the Tftpd Server IP

### 6.3.3 Set the Ethernet upgrade file name

a) Click “Browse & set file name”. In the file open window, browse to the location where you extracted the firmware upgrade *.S19 file. Click “OK”
b) Click “Write”
c) Ethernet Upgrade File Name should read ‘~OK’

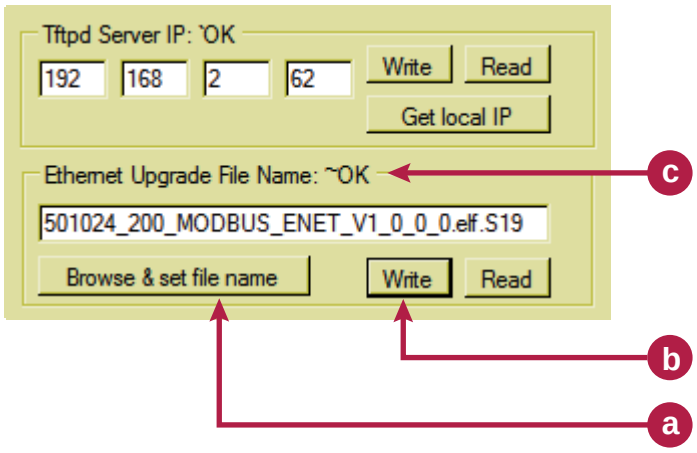



Figure 6.8: Setting the upgrade file name

### 6.3.4 Enter upgrade mode

1) On the «Edit» menu, select «Enter Ethernet Firmware Upgrade Mode»
2) A dialog will open requesting verification of the upgrade file-name. If the name matches, click «Yes».
3) If it does not match, click «No» and repeat step 2.
4) In the dialog, «Enter unlock code to enter upgrade mode», enter the code:

   2956102

5) The message, «Successfully entered Ethernet Firmware up-grade mode» will appear, click «OK».
6) The message «Cycle power to upgrade Ethernet firmware via Tftpd server» will pop up. DO NOT Click OK at this point.

### 6.3.5 Complete upgrade process

1) Remove power from you MDrive.
2) Click “OK” on the dialog referenced in Step 3-e.
3) On the “Edit” menu, select “Select & Enter Tftpd Server”. The browse dialog should open to the install directory. If not, browse to the Tftpd_Server install directory and select “tftpd32. exe”
4) Click “Open”
5) Apply power to the MDrive
6) The upgrade should begin after a few seconds.
7) When complete, close Tftpd server.
8) Cycle power to the MDrive.
9) Reconnect using the default IP: 192.168.33.1 and Subnet mask: 255.255.0.0.
10) Configure device to your system requirements.

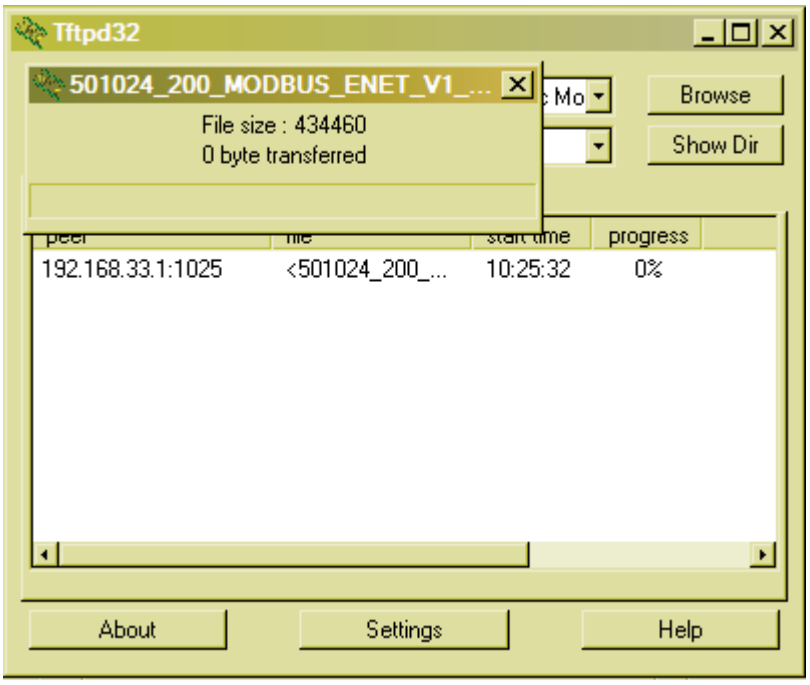



Figure 6.9: Firmware upgrading

# 7 Programming and application notes

This section will cover the following areas of MCode programming and applications in detail.

- Party mode communications
- Programming the I/O
- Factors impacting motion commands
- MForce PWM configuration

## 7.1 Party mode communications

The following communication formats, used by MCode comptible devices.

{} ..... The contents between the {} symbols are transmitted.

{0D} ..... Hex equivalent for a CR (Carriage Return).

{0A} ..... Hex equivalent for a LF (Line Feed).

{DN} ..... Represents the Device Name being sent.

{CS} ..... Check Sum; {ACK} 06 Hex; {NAK} 15 Hex

EM = Echo Mode; PY = PartY Mode; CK= CheckK sum

The word {command} represents the immediate command sent to the device.

Command execution time (CET) is the time the device takes to execute a command. This varies from command to command and usually is in the 1-5 millisecond range.

### 7.1.1 Response to Echo Mode

Dependent on how the echo mode (EM) is set in conjunction with party mode (PY) and check sum (CK), the device will respond differently. The following tables illustrate the various responses based on how the EM, PY and CK parameters are set.

| Parameter Setting | Transmission | Initial Response | Final Response | Notes |
|---|---|---|---|---|
| EM=0 & PY=0 CK=0 | (command) (D) | (command) Echoed back one character at a time as the character is entered. | CET (0D) (0A)> | The last character sent is the prompt > |
| EM=1 & PY=0 CK=0 | (command) (0D) | – | CET (0D) (0A) | The last character sent is LF |
| EM=2 & PY=0 CK=0 | (command) (0D) | – | – | No response except to PR and L commands |
| EM=3 & PY=0 CK=0 | (command) (0D) | – | CET command (0D) (0A) | Queued response. The last character sent is the LF |

Table 7.1 Response to echo mode - party and check sum are zero (0)

| Parameter Setting | Transmission | Initial Response | Final Response | Notes |
|---|---|---|---|---|
| EM=0 & PY=1 CK=0 | (DN) (command) (0A) | (command) Echoed back one character at a time as the character is entered. | CET (0D) (0A)> | The last character sent is the prompt > |
| EM=1 & PY=1 CK=0 | (DN) (command) (0A) | – | CET (0D) (0A) | The last character sent is LF |
| EM=2 & PY=1 CK=0 | (DN) (command) (0A) | – | – | No response except to PR and L commands |

| | | | | |
|---|---|---|---|---|
| EM=3 & PY=1<br>CK=0 | (DN) (command)<br>(0A) | – | CET command<br>(0D) (0A) | Queued response. The last character sent is the LF |

Table 7.2 Response to echo mode - party is one (1) and check sum is zero (0)

| Parameter Setting | Transmission | Initial Response | Final Response | Notes |
|---|---|---|---|---|
| EM=0 & PY=0<br>CK=1 | (DN) (command)<br>(0A) | (command)<br>Echoed back one character at a time as the character is entered. | CET (0D) (0A)> | The last character sent is the prompt > |
| EM=1 & PY=0<br>CK=1 | (DN) (command)<br>(0A) | – | CET (0D) (0A) | The last character sent is LF |
| EM=2 & PY=0<br>CK=1 | (DN) (command)<br>(0A) | – | – | No response except to PR and L commands |
| EM=3 & PY=0<br>CK=1 | (DN) (command)<br>(0A) | – | CET command<br>(0D) (0A) | Queued response. The last character sent is the LF |

Table 7.3 Response to echo mode - party is zero (0) and check sum is one (1)

| Parameter Setting | Transmission | Initial Response | Final Response | Notes |
|---|---|---|---|---|
| EM=0 & PY=1<br>CK=1 | (DN) (command)<br>(CS) (0A) | (command)<br>Echoed back one character at a time as the character is entered. | CET (ACK) or (NAK)> | The last character sent is the prompt > |
| EM=1 & PY=1<br>CK=1 | (DN) (command)<br>(CS) (0A) | – | CET (ACK) or (NAK)> | The last character sent is ACK or NAK |
| EM=2 & PY=1<br>CK=1 | (DN) (command)<br>(CS) (0A) | – | – | No response except to PR and L commands |
| EM=3 & PY=1<br>CK=1 | (DN) (command)<br>(CS) (0A) | – | CET command<br>(CS) (ACK) (NAK) | Queued response. The last character sent is ACK or NAK |

Table 7.4: Response to echo mode - party and check sum are one (1)

### 7.1.2 Using Check Sum

For communication using check sum, the following 2 commands demonstrate sending and receiving.

1) Check sum set to zero before first character is sent.
2) All characters (ascii values) are added to check sum, including the device name DN (if PY=1), to the end of the command, but not including terminator.
3) Check sum is 2's complement, then "or" ed with hex 80 (prevents check sum from being seen as command terminator).
4) Terminator sent.

Note: Any combination of upper/lower case may be used. In this example, if a lower case <mr> were to be used, the decimal values will change to 109 and 114. Subsequently the result check sum value will change. (Possible entries: MR, mr, Mr, mR.) (M = 77, R =82, m = 109, r = 114) (See ASCII table in Section 9 of this document.)

```
77  82  32  49    Decimal value of M, R, <space> and 1
4D  52  20  31    Hex
77+82+32+49 = 240        Add decimal values together
1111  0000 = 240  Change 240 decimal to binary
0000  1111 1's complement (invert binary)
0001 0000  Add 1 [2's complement]
1000  0000 OR result with 128 (Hex 80)
1001 0000 144     Result Check Sum value
```

Once the result is reached, add the check sum value (144 in this example) to your string by typing: MRr 1(alt key + 0144) (use the symbol of 0144 in your string by holding down the alt key and typing 0144). You must type the numbers from the numlock key pad to the right of the keyboard. The numbers at the top of the keyboard will not work.

1) Check sum set to zero.
2) All characters are added to check sum.
3) When receiving a command terminator, the lower 7 bits of the check sum should be equal to zero.
   A) if not zero, the command is ignored and NAK echoed.
   B) if zero, ACK is sent instead of CR/LF pair.
4) Responses to PR commands will be check summed as above, but the receiving device should not respond with ACK or NAK.

### 7.1.3 Immediate party mode sample codes

Once party mode has been defined and set up as previously described under the heading "multiple devices (party mode)", you may enter commands in the immediate mode in the ims terminal window. Some examples follow.

*Move device A, B or C 10000 steps*

Assuming there are three devices set up in party mode as shown in the sample codes above.

- To move mdrive unit “a”, press CTRL+J and then type: aMR∧10000 and pressCTRL+J. device “a” will move 10000 steps.
- To print the position type: aPR p and press CTRL+J. The position of device “a” will be printed.
- To move device “b” type: bMR 10000 and press CTRL+J. Device “b” will move 10000 steps.
- To move all three devices at the same time type: *MR 10000 and press CTRL+J. All devices will move 10000 steps.
- To change a variable in the “c” unit type: c<variable name><number> and press CTRL+J. The variable will be changed. To verify the change type: cPR <variable name> and press CTRL+J. The new value will be displayed.
- All commands and variables may be programmed in this manner.
- To take a device out of party mode type: <device name>PY=0 and press CTRL+J. That unit will be taken out of party mode. To take all units out of party mode type: *PY=0 and press CTRL+J. All units will be taken out of party mode.

## 7.2 Programming the I/O

### 7.2.1 I/O availability per device type

The product families using the MCode language may have different sets of I/O points and functions. These are

*MDrive, MForce*

- 4 +5 to +24 VDC I/O points programmable as sinking or sourcing inputs or sinking outputs.
- 1 analog input.

*MDrive, MForce, (Plus² expanded features)*

- 8 +5 to +24 VDC I/O points programmable as sinking or sourcing inputs or outputs.
- 1 analog input.
- 2 TTL level step/direction I/O programmable as sinking or sourcing inputs or outputs.
- 1 TTL level high speed point programmable for capture input or trip output functions

*MDrive, MForce (Plus² expanded features with remote encoder inputs)*

- 4 +5 to +24 VDC I/O points programmable as sinking or sourcing inputs or outputs.
- 1 analog input.

- 6 TTL level differential encoder inputs.
- 1 TTL level high speed point programmable for capture input or trip output functions.
- **EXCEPTION:** MDrive34Plus² and MForce PowerDrive Plus² are available with 8 sinking/sourcing I/O points and remote encoder inputs.

*MDrive Hybrid*

- 8 +5 to +24 VDC I/O points programmable as sinking or sourcing inputs or outputs.
- 1 analog input.
- 1 TTL level high speed point programmable for capture input or trip output functions

### 7.2.2 Active states defined

The active state determines at what voltage level the input will be active.

Active HIGH: the input will be active when +5 to +24 VDC is applied to the input.

Active LOW: The input will be active when it is grounded (0 VDC).

*Examples*

IO 1 is to be configured as a Jog– input which will activate when a switch is toggled to ground (sinking input):

```
S1=8,0,0   'set io point 1 to jog-, active low, sinking
```

IO 4 is to be configured as a home input which will activate when instructed by a PLC (+24VDC sourcing input):

```
S4=1,1,1   'set io point 1 to home, active high, sourcing
```

## 7.2.3 Digital input functions

The inputs may be interfaced to a variety of sinking or sourcing devices. An input may be programmed to be a general purpose user input, or to one of nine dedicated input functions. These may then be programmed to have an active state of either high or low.

The inputs are configured using the “S” variable (see Section 5: Command details). The command is entered into the ims terminal or program file as:

```
s<io point>=<io type>,<active state><sink/source>.
```

Example:

```
S9=3,1,0   'set io9 = limit- input, active high, sinking
S3=0,0,1   'set io 3 = general purpose input, active low,
           'sourcing
```

*Input Functions (I/O Points 1-4, 9-12)*

The following table lists the programmable input functions.

| Function | Description | Parameter (S1-S4, S9-S12) | Active | Sink/ Source |
|---|---|---|---|---|
| General Purpose | General purpose input function used to control program branches, subroutine calls or bcd functions when input bank is used as a group | 0 | 0/1 | 0/1 |
| Home | Homing input. Will function as specified by the home (hm) command. | 1 | 0/1 | 0/1 |
| Limit + | Positive limit input. Will function as specified by the limit (lm) command. | 2 | 0/1 | 0/1 |
| Limit – | Negative limit input. Will function as specified by the limit (lm) command. | 3 | 0/1 | 0/1 |
| G0 | G0 input. Will run program located at address 1 on activation. | 4 | 0/1 | 0/1 |
| Soft Stop | Soft stop input. Stops motion with deceleration and stops program execution. | 5 | 0/1 | 0/1 |
| Pause | Pause/resume program with motion. | 6 | 0/1 | 0/1 |
| Jog + | Will jog motor in the positive direction at max. Velocity (vm). The jog enable (je) flag must be set for this to function. | 7 | 0/1 | 0/1 |
| Jog – | Will jog motor in the negative direction at max. velocity (VM). The jog enable (JE) flag must be set for this to function. | 8 | 0/1 | 0/1 |
| Reset | When set as reset input, then the action is equivalent to a ^c entered into a terminal. | 11 | 0/1 | 0/1 |

Table 7.5 Digital input functions

*Input Functions (Points 7 & 8 — Clock Inputs and Point 13 — Capture)*

| Function | Description | Parameter (S7, S8) | Active |
|---|---|---|---|
| Step/Direction | Sets IO 7 and 8 to receive step and direction inputs from an external source. The motion will occur based on the input frequency seen at IO 7 in the direction relative to the logic state of IO 8. The step rate will be based upon the ratio set by clock ratio (cr) | 33 | 0/1 |
| Quadrature | Sets IO 7 and 8 to receive channel a and channel b quadrature inputs from an external source such as an encoder. The motion will follow the quadrature input. | 34 | 0/1 |
| Up/Down | Sets IO 7 and 8 to receive clock up/clock down inputs from an external source. The motion will occur based upon the input clock frequency in the direction relative to the input being clocked. The step rate will be based upon the ratio set by clock ratio (CR) | 35 | 0/1 |
| **Function** | **Description** | **Parameter (S13)** | **Active** |
| High Speed Capture | The capture input is a momentary high speed input that operates with the trip capture (TC) variable to run a subroutine upon the trip. It feature variable input filtering ranging from 50 ns to 12.9 Ms | 60 | 0/1 |

Table 7.6 Clock and high speed input functions

## 7.2.4 Digital output functions

The outputs may be configured as general purpose or set to dedicated functions, fault or moving. These outputs will sink up to 600 mA (one channel of two banks) and may be connected to an external VDC source.

The outputs are set using the “S” command (see Section 5 of this document for precise details on this command). The command is entered into the terminal or program file as:

```
S<IO point>=<IO Type>,<Active State><Sink/Source>.
```

*Examples*

```
S9=17,1,0  'set IO point 9 to be a moving output, active
           'high, sinking
S3=18,0,0  'set IO point 3 to be a fault output, active
           'low, sinking
```

*Output Functions*

Output functions may be programmed to be a general purpose user output with the following functions.

| Function | Description | Parameter (S1-S4, S9-S12) | Active | Sink/ Source |
|---|---|---|---|---|
| General Purpose User | A general purpose output can be set in a program or in immediate mode to trigger external events. When used as a group they can be a BCD output. | 16 | 0/1 | 0/1 |
| Moving | Will be in the active state when the motor is moving. | 17 | 0/1 | 0/1 |
| Fault | Will be in the Active State when a error occurs. . | 18 | 0/1 | 0/1 |
| Stall | Will be in the active state when a stall is detected. Encoder required, stall detect mode (SM) must be enabled. | 19 | 0/1 | 0/1 |
| Velocity Changing | Will be in the active state when the velocity is changing. Example: during acceleration and deceleration. | 20 | 0/1 | 0/1 |
| *Locked Rotor | Will be in an active state when the rotor is locked on MDrive Hybrid products | 21 | 0/1 | 0/1 |
| Moving to Position | Will be active when the motor is indexing to a commanded position. | 22 | 0/1 | 0/1 |
| *Hybrid Active | Will be active when the Hybrid control circuitry is engaged. | 23 | 0/1 | 0/1 |
| *Make Up Active | Will be active when the Hybrid is correcting lead/lag conditions. | 24 | 0/1 | 0/1 |

*Only applies to Hybrid products.

Table 7.7 Digital output functions

*Output Functions (Points 7 & 8 — Clock Outputs and Point 13 — Trip)*

| Function | Description | Parameter (S7, S8) | Active |
|---|---|---|---|
| Step/Direction | Step clock pulses will be output from point 7, direction from point 8. The step clock output rate will be based upon the pulse width set by clock width (CW). The logic state of the direction output will be with respect to the direction of the motor. | 49 | 0/1 |
| Quadrature | Will output Quadrature signals. | 50 | 0/1 |
| Up/Down | Will output clock up/clock down signals. The step clock output rate will be based upon the pulse width set by clock width (CW). The active output will be based on the motor direction. | 51 | 0/1 |
| **Function** | **Description** | **Parameter (S13)** | **Active** |
| High Speed Trip | The trip output will activate on position trips (TP) only. The output will pulse out at the trip point. The pulse width will be determined by clock width (CW) | 61 | 0/1 |

Table 7.8 Clock and high speed output functions

## 7.2.5 Programmable I/O usage examples

The code examples below illustrate possible interface examples for using the digital I/O.

Reference the hardware manual of your device for connection and wiring information

*Input Interface Example - Switch Input (Sinking Input)*

The following example shows a switch connected between an I/O point and power ground.

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive Hybrid
MForce
MForce (Plus² expanded features)

**Code Sample**

For the code sample, this switch will be set up as a G0 sinking input, active when low. When pressed, the switch will launch the program beginning at address 1 in device memory:

```
***Setup Variables***
Sx=4,0,0   'set IO point x to be a G0 input, active when
           'LOW, sinking

****Program***
PG1
MR 20000   'Move +20000 steps relative to current
position
H          'Hold program execution until motion completes
MR -20000  'Move -20000 steps
H          'Hold program execution until motion completes
E
PG         'End program, exit program mode
```

*Input interface example - switch input example (sourcing input)*

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive Hybrid
MForce
MForce (Plus² expanded features)

The following circuit example shows a switch connected between an I/O point and a voltage supply which will source the input to perform a function.

**Code Sample**

For the code sample, the switch will be set up as a soft stop sourcing input, active when high. When pressed, the switches will stop the motor.

```
S1=5,1,1   'set IO point 1 to be a Soft Stop input, active
           'when HIGH, sourcing
SL 200000  'slew the motor at 200000 µsteps/sec
```

When the switch is depressed the motor will decelerate to a stop.

*Output interface example (sinking output)*

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive Hybrid
MForce
MForce (Plus² expanded features)

The following circuit example shows a load connected to an I/O point that will be configured as a sinking output.

For the code sample, the load will ben LED. T motor is a tonfigured such that the LED will be lit while the motor is at constant velocity. Set input 1 up to be a soft stop input using a switch in a sinking configuration this will soft stop the motor.

```
S1=5,0,0   'set IO point 1 to be a Soft Stop input, active
           'when LOW, sinking.
S1=20,0,0  'set IO point 2 to be a Velocity Changing
           'output, active when LOW
SL 2000000 'slew the motor at 200000 µsteps/sec
```

While the motor is accelerating the LED will be dark, but will light up when the motor reaches a constant velocity. When the Soft Stop switch is depressed the motor will begin to decelerate, the LED will go dark again while velocity is changing.

*Output interface example (sourcing output)*

Compatible with Motion Control products:
MDrive (Plus² expanded features)
MDrive Hybrid
MForce (Plus² expanded features)

The following circuit example shows a load connected to an I/O point that will be configured as a sourcing output.

**Code Sample**

For the code sample, the load will be a relay. The output will be configured to be a general purpose user output that will be set active when a range of motion completes.

```
******Setup Variables******
S1=16,1,1  'set IO 1 = user output, active HIGH, sourcing.

******Program******
PG 100     'Enter program at address 100
MR 2000000 'Move x in the positive direction
H          'Hold execution until motion completes
MR -1000000       'Move x distance negative direction
H          'Hold execution until motion completes
O1=1               'Set output 1 HIGH
```

Enter EX 100 to execute the program, the motion will occur and the output will set high.

*Reading inputs as a group example*

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive Hybrid
MForce
MForce (Plus² expanded features)

The inputs may read as a group using the IL. IH and IN keywords. This will display as a decimal between 0 to 15 representing the 4 bit binary number (IL, IH) or as a decimal between 0 and 255 representing the 8 bit binary number on the devices with expanded I/O. The IN keyword will function on the devices with standard I/O but will only read inputs 1 - 4. Inputs will be configured as user inputs (S<point>=0).

**Standard I/O**

```
PR IN              'Reads Inputs 4(MSB) through 1(LSB)
PR IN              'Reads Inputs 4(MSB) through 1(LSB)
```

**Expanded I/O**

```
PR IL      'Reads Inputs 4(MSB) through 1(LSB)
PR IH:     'Reads Inputs 12(MSB) through 9(LSB)
PR IN:     'Reads Inputs 12(MSB) -  9 amd 4 - 1(LSB)
```

*Interfacing outputs as a group example*

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive Hybrid
MForce
MForce (Plus² expanded features)

Outputs may be written to as a group using the OL, OH and OT keywords. This will set the outputs as a binary number representing the decimal between 0 to 15 representing the 4 bit binary number (OL, OH) or as an 8 bit binary number representing the decimal 0 to 255 on texpanded I/O devices. The OT keyword will function on tstandard I/O devices, but will only set outputs 1 - 4. Outputs will be configured as user outputs (S<point>=16).

**Standard I/O**

```
OL=3              'set the binary state of the standard
I/O to 0011
OT=13             'set the binary state of the standard
I/O to 1101
```

**Expanded I/O**

```
OL=5              'set the binary state of the standard
I/O to 0101
OH=9              'set the binary state of the expanded
I/O to 1001
OT=223     'set the binary state of the combined I/O to
1101 1111
```

## 7.2.6 Dedicated digital I/O usage

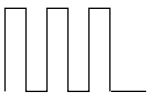
Compatible with Motion Control products:
MDrive (Plus² expanded features)
MForce (Plus² expanded features)

These dedicated I/O lines are used to receive clock inputs from an external device or provide clock outputs to an external device such as a counter or a second device in a system. The clock I/O can be configured as one of three clock types using the S7 and S8 variable:

1) Step/Direction
2) Quadrature
3) Up/Down

*Step/Direction*

The Step/Direction function would typically be used to receive step and direction instructions from a second system device or secondary controller. When configured as outputs the device can provide step and direction control to another system drive for electronic gearing applications.



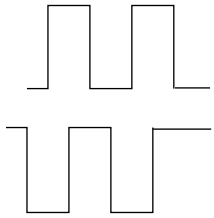

```
Input:  S<7>=33, <active state>
Output: S<7>=49,<active state>

Input:  S<8>=33, <active state>
Output: S<8>=49,<active state>
```

Figure 7.1 Step/direction I/O type & configuration

*Quadrature*

The quadrature clock function would typically be used for following applications where the device would either be a master or slave in an application that would require two motors to move the same distance and speed.

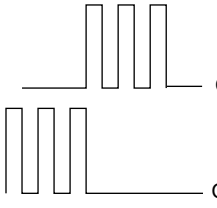



```
Input:  S<7>=34, <active state>
Output: S<7>=50,<active state>
Input:  S<8>=34, <active state>
Output: S<8>=50,<active state>
```

Figure 7.2 Quadrature I/O configuration

*CW/CCW*

The cw/ccw clock would typically be used in a dual-clock direction control application, or to increment/decrement an external counter.




```
Input:  S<7>=35, <active state>
Output: S<7>=51,<active state>
Input:  S<8>=35, <active state>
Output: S<8>=51,<active state>
```

Figure 7.3 CW/CCW I/O configuration

## 7.2.7 Analog input usage

Compatible with Motion Control products:
MDrive
MDrive (Plus² expanded features)
MDrive Hybrid
MForce
MForce (Plus² expanded features)

The analog input is configured from the factory as a 0 to 5V, 10 bit resolution input (S5=9). This offers the user the ability to receive input from temperature, pressure, or other forms of sensors, and then control events based upon the input.

The value of this input will be read using the I5 instruction, which has a range of 0 to 1023, where 0 = 0 volts and 1024 = 5.0 volts. The anlog input may also be configured for a 4 to 20 mA or 0 to 20 mA Analog Input (S5 = 10). If used as a 4 to 20mA input the range is 0 to 800 units.

*Sample Usage*

```
'***********Main Program**************

S5=9,0            'set analog input to read variable
                  'voltage (0 to +5VDC)
PG 100            'start prog. address 100
LB A1             'label program A1
CL A2, I5<500     'Call Sub A2, If I5 is less than 500
CL A3, I5>524     'Call Sub A3, If I5 is greater than 524
BR A1             'loop to A1

'***********Subroutines**************

LB A2      'label subroutine A2
MA 2000    'Move Absolute 2000 steps
H          'Hold program execution until motion ceases
RT         'return from subroutine

LB A3      'label subroutine A3
MA -2000   'Move Absolute -2000 steps
H          'Hold program execution until motion ceases
RT         'return from subroutine
E          'End
PG         'Exit program
```

## 7.3 Factors impacting motion commands

### 7.3.1 Motor steps

All MCode examples assume 200 step motors. They rotate at 1.8° per clock pulse. 200 steps would equal 1 revolution. MCode devices such as the MForce line may be used with different step resolution motors, such as 0.9° motors.

### 7.3.2 Microsteps: (MS)

Microsteps divide the 200 motor steps into smaller steps to improve smoothness and resolution of the MCode compatible device. Using the default setting of 256 for MS, the 200 motor steps are increased to 51200 microsteps. One motor revolution requires 51200 microsteps with the ms set at 256. If you were to set the ms to 128, one revolution of the motor would now require 25600 microsteps.

### 7.3.3 Move Command

The move absolute (MA) and the move relative (MR) commands are programmed in microsteps or if the encoder is enabled, encoder counts. If the ms was set at 256 and you were to program a move of 51200 microsteps, the motor would turn one full revolution. If the ms was set to 128, one full revolution of the motor would be 25600 microsteps (128 x 200). If you programmed a move of 51200, the motor would turn 2 full revolutions.

### 7.3.4 Closed loop control with an encoder

If the encoder is enabled the move commands use different values. The encoder has 512 lines and yields 2048 counts or counts per revolution. Therefore, the MR and MA command values are programmed in encoder counts. One full revolution would be programmed as mr or ma 2048.

When the encoder is enabled, the MS value is defaulted to 256. It cannot be changed.

Knowing these factors you can program a multitude of different movements, speeds, and time intervals.

## 7.3.5 Linear movement

You have a rack and pinion or a ball screw to move a linear axis. The rack and pinion or ball screw moves the linear axis 0.1 inches for each revolution. You need to move 7.5 inches.

7.5 inches divided by 0.1 inches = 75 motor revolutions.

Assuming an MS of 256 (51200 Microsteps) is programmed, 51200 Microsteps x 75 revolutions requires a move of 3840000 microsteps.

Knowing the values of the variables as well as the required move, you can calculate the actual time it takes to move the axis the required distance. This is done with a trapezoidal profile as shown below.

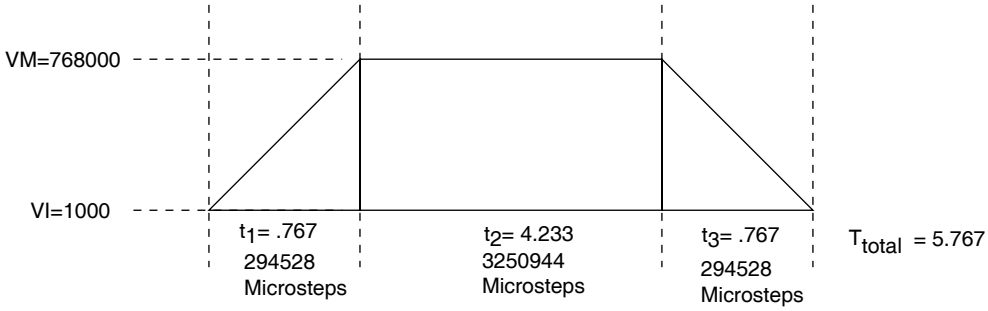



Figure 7.4 Trapezoidal move profile

*Calculating axis speed (velocity)*

There are several steps required to determine the actual axis speed. They are all based on the Trapezoidal Profile above.

Known Values and Parameters:

VM.....768000 Steps/Sec.

VI.....1000 Steps/Sec.

A.....1000000 Steps/Sec2.

D .....1000000 Steps/Sec2.

MA/MR.....3840000 Microsteps

Determine the Acceleration (A) and Deceleration (D) times (t1 and t3). Since the Deceleration (D) value is also 1000000 Steps/Sec. the Deceleration time (t3) will be the same as the Acceleration time (t1).

$$(t1 \text{ and } t3) = \frac{VM - VI}{A} \textbf{ OR } \frac{768000 - 1000}{1000000} \textbf{ = 0.767 Seconds}$$

Determine the distance (Steps) traveled in t1 or t3.

$$\text{Distance} = \frac{VM + VI}{2} \times t1 \quad \textbf{OR} \quad \frac{768000 + 1000}{2} \times 0.767$$

**= 294911 steps**

Determine the t2 time.

The t2 time is calculated by dividing the remainder of MA/MR by VM.

The remainder of MA/MR = MA/MR - (t1 steps + t3 steps) or 3840000 - 589056 = 3250944.

$$t2 = \frac{3250944}{768000} \textbf{ = 4.233 Seconds}$$

Determine the total time. (t1 + t2 + t3) or (0.767 + 4.233 + 0.767) = 5.767 Seconds

The linear axis took 5.767 seconds to move 7.5 inches or an average speed of 78 inches/minute.

Note that the average speed includes the Acceleration and Deceleration. The maximum axis speed attained is approximately 90 inches/minute.

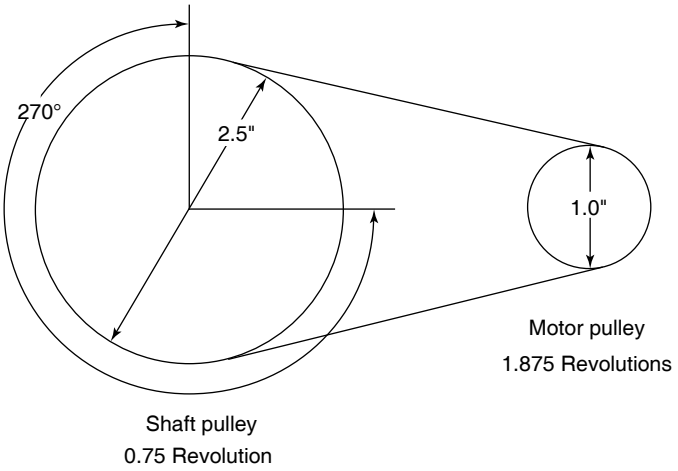
$$\frac{768000}{51200} \times 0.1 \times 60 \textbf{ = 90 IPM}$$

## 7.3.6 Calculating rotary movement

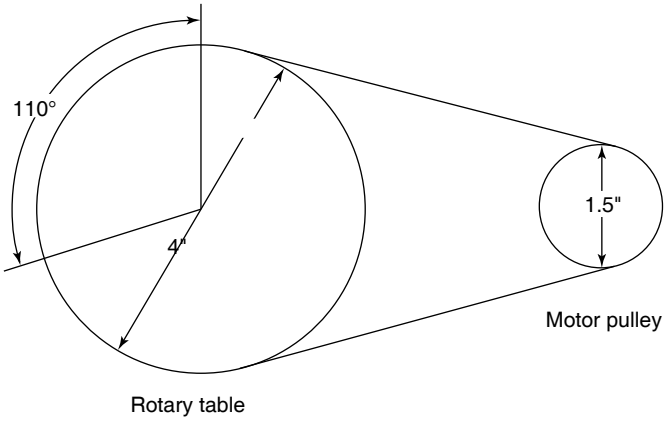
Assume the MS is set to 256. You are using the motor to drive a shaft with a timing belt and pulley arrangement. As shown below, the pulley is 1” in diameter and the shaft pulley is 2.5” in diameter. You must turn the shaft 270°.

- The shaft will rotate 1 full revolution for every 2.5 revolutions of the motor.
- 270° is 0.75 of a revolution.
- 0.75 x 2.5 = 1.875 motor revolutions to turn the shaft 270°.
- If 51200 Microsteps is 1 motor revolution, then the device must be programmed to move 96000 Microsteps (51200 x 1.875).

You may also do many of the calculations in reverse to calculate motor moves to meet a required move of your device. A linear or rotational move as well as speed may be translated into an MCode command.




**Rotary drive example 1**




**Rotary drive example 2**

Figure 7.5 Rotary examples

In the example above, the belt driven rotary table must be turned 110° at 3 RPM. How should the device be set up?

Bear in mind that all the numbers are approximate due to rounding.

Mechanical ratio between the motor and the rotary table is 2.666:1. That is, the motor must rotate 2.666 revolutions for the table to rotate 1 revolution and the table will rotate 2.666 times slower than the motor.

In order to move the table 110° the motor must move 293.3°.

`110 x 2.66` **`= 293.3º`**

If 51200 steps = 1 revolution then 1° = 142.222 steps.

$$\frac{51200}{360} = \textbf{142.222 steps}$$

The MCode device must be programmed to move 41713 steps to rotate 293.3°.

`142.222 steps x 293.3º` **`= 41713 steps`**

In order to rotate the table at 3 RPM the motor must turn at 8 RPM.

`3 RPM x 2.666` **`= 8 RPM`**

If you were to set VM at 51200 and MS set at 256 the motor will rotate 1 full revolution (51200 steps) in 1 second or 1 RPS. In order to rotate at 8 RPM, the motor must rotate at 0.13333 RPS.

$$\frac{8}{60} = \textbf{0.133333 RPS}$$

In order to rotate at 0.13333 RPS the VM must be set at 6827 steps/sec.

`51200 x 0.133333` **`= 6827`**

**`VM = 6827`**

Note: These numbers will vary slightly depending on Acceleration and Deceleration rates.

### 7.3.7 Programming with the optional encoder enabled

An optional 512 line magnetic encoder is available. When the Encoder is enabled (EE=1) the programming also changes. All motion must now be programmed by the encoder counts. The Encoder operates in the "Quadrature" format. That is, there are four Encoder counts for each Encoder line or 2048 counts per revolution (512 × 4 = 2048). (See Figure below.) If you were to program motion using the MR (Move Relative) or MA (Move Absolute) commands the motor would rotate a distance equal to the encoder counts.

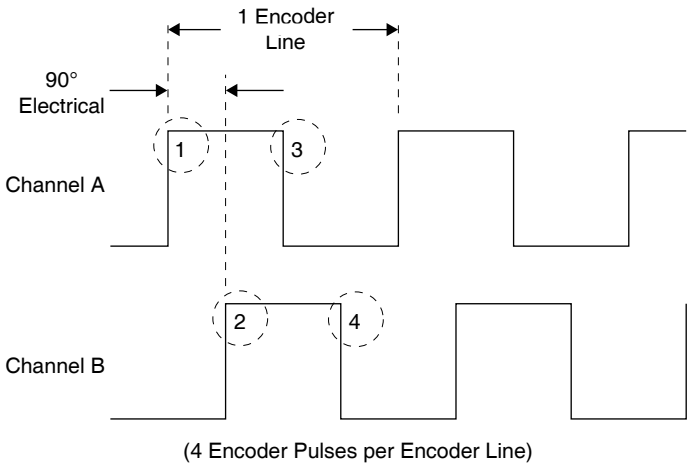



Figure 7.6 Encoder waveform

*Example:*

A programmed move of 7168 counts would result in the motor rotating 3.5 revolutions at a velocity controlled by VM.

```
7168 ÷ 2048 = 3.5 revolutions
```

If you were to program motion using the SL (Slew) command the motor would rotate at a "counts per second" rate based on the programmed value.

*Example:*

An SL (Slew) rate of 7168 counts was programed. The motor will rotate at 7168 counts/sec., 3.5 RPS, or 210 RPM.

```
7168 ÷ 2048 = 3.5 RPS × 60 = 210 RPM
```

When the Encoder is enabled, the parameters are also changed to be compatible with the 2048 counts.

The Encoder Enabled defaults are:

VM.....30720 Counts/Sec.

VI.....40 Counts/Sec.

A.....40000 Counts/Sec

D.....40000 Counts/Sec.

MS.....256 (default for encoder mode.)

To enable the encoder the program syntax is <EE=n> where n is a zero (0) or a one (1). The default is zero (0) which is encoder disabled. To enable the encoder, program EE=1.

Any motion will now be programmed in encoder counts. You can calculate the distance or velocity you need in a similar manner as done previously only with different factors.

Note: The microstep select is defaulted and locked at 256 in the encoder mode to ensure stable, high resolution.

Several Variables work in conjunction with Encoder Enable (EE). They are:

DB .....Encoder Deadband

SF.....The Stall Factor Variable

SM..... The Stall Detection Mode

ST.....Stall Flag

PM..... Position Maintenance

EE .....Encoder Enabled

When the encoder is enabled, all motion is “closed loop”. That is, motion steps are delivered from the MCode device to the motor which turns the encoder. The encoder sends counts back to the drive to complete the motion. If you programmed a move of 2048 counts, the device would output an appropriate number of microsteps provided the stall factor (SF) value or other fault is not encountered. If no faults were encountered, the device would output the full amount of microsteps. Depending on which variables were set, the driver would then wait until the position (plus or minus the encoder deadband) was read and confirmed.

*DB - Encoder Deadband*

The Encoder Deadband is a Variable that is set in Encoder Counts. Motion will be deemed complete when the Encoder Counts are within ± the Deadband variable. With DB=5 the motion of 2048 counts would be complete between 2043 and 2053 counts.

*SF - Stall Factor*

The Stall Factor is a Variable which is entered in Encoder Counts. The Stall Factor is active only in the EE=1 mode. The Stall Factor might be compared to the “following error” or “lag error” of a servo drive. The Stall Factor is triggered by the number of steps output from the device to the motor as compared to the number of counts returned by the encoder. The comparison should always be within the value of the Stall Factor, otherwise a fault will occur and the Stall Flag (ST) will be set. If the Stall Detection Mode is active (SM=0), the motion will be stopped.

**Example:**

A Stall Factor of 30 counts (SF=30) is programmed. A motion command of 2048 counts is programmed. The device reaches a mechanical bind at 2000 counts. The device will keep outputting steps equivalent to 2030 counts (present position plus the SF value) and then the Stall Flag (ST) will be set. The motor will be stopped if the Stall Detection Mode (SM=0) is active.

*SM - Stall Detection Mode*

The Stall Detection Mode can be programmed to stop the device (SM=0) or to allow the device to continue (SM=1) when the Stall Factor (SF) is reached. Whether SM is active or not, the Stall Flag will always be set when the SF is encountered.

*ST - Stall Flag*

The Stall Flag will be set any time the SF is reached regardless of the state of the Stall Detection Mode (SM). If the Stall Flag is set, the user must reset it to zero (0).

*PM - Position Maintenance*

Position maintenance (PM) is active only after the motion has completed. Position maintenance is used to maintain position when there might be an external force on the drive. If position maintenance is enabled (PM=1) and the stall detection mode is enabled (SM=0), the motor will be driven back to its final position if it was forced out of position provided the stall factor (SF) was not reached.

If position maintenance is enabled (PM=1) and the stall detection mode is disabled (SM=1), the motor will be driven back to its final position if it was forced out of position regardless of whether the stall factor (SF) was reached or not.

There are three other variables, although not directly conned to EE, that do affect the overall operation when in encoder mode, they are:

HC..... Motor Hold Current

HT..... Motor Hold Current Delay Time

MT.....Motor Settling Delay Time

HC.....Hold Current

When motion is complete, the device will switch from motor run current (RC) to motor hold current (HC). The hold current is set at a lower percentage than the run current (rc). However, the hold current must be sufficient to overcome an outside force such as driving a vertical slide which maintains a load on the motor at all times. Actual hold current

values will vary depending on the application and the load on the motor when it is at rest.

*HT - Motor Hold Current Delay Time*

The motor hold current delay time (HT) is a variable that delays the change from run current (RC) to hold current (HC) at the end of a move. The end of the move is triggered by the device when it has completed outputting the correct number of steps. Depending on the application, including velocity, deceleration, load and inertia, the device may lag behind a few counts. The ht will allow the device to finish its move before applying the lower HC.

*MT - Motor Settling Delay Time*

A stepping motor may ring or oscillate in minuscule amounts at the completion of a move until it satisfies the target position. The amount of this “ringing” is dependent on the application including velocity, deceleration, inertia, friction and load. The motor settling delay time (MT) allows the motor to stop “ringing” before checking the position count. If the device tried to check the position count during this ringing, it would assume a position error and try to correct an already moving motor and possibly cause ringing of a larger magnitude and longevity. Typically, the MT is set between 50 and 100 milliseconds. It is recommended that there is always a Motor Settling Time programmed any time you are in EE=1 mode.

Note: If MT has no value, the motor may hunt and never satisfy the position check.

## 7.4 MForce PWM configuration

**⚠ CAUTION**

This variable is only applicable to the mforce product line and is used to tune the PWM Settiings to optimize the current control of the MForce driver. It should not be used unless erratic motion or positional accuracy problems are being experienced.

Note that there are other factors that could contribute to these problems. Ensure that all wiring conforms to the guidelines in the mforce hardware manual.

Be aware that this parameter, when used with the checksum will write to the boot sector of memory. This sector only allows eight write cycles. Ensure that the settings you choose are optimal prior to storing the parameter to the boot.

Read this section closely before making changes to the PWM settings. Please contact application support for questions concerning the use of this command.

### 7.4.1 Description:

This variable defines the parameter settings for the PWM and should not be used unless motion problems such as smoothness and positional accuracy are experienced.

The PW variable consists of four components: <mask>, <period>, <sfrq> and either <checksum>, if writing, or <boot_writes_remaining> if reading using the PR keyword.

### 7.4.2 PWM Mask <mask> Parameter

The PWM mask signal prevents the premature end of the forward period caused by switching transients when the motor phase current is at low levels. Adjusting this value can impact the zero-crossing performance of the motor. If experiencing the “tick” which is inherit in stepper motor systems, this may be minimized or eliminated by adjusting this value. The range of this value is 0 to 255d and will be entered as a decimal value.

The Mask will act as a filter on the PWM signal to allow time for any ringing in the output circuitry to settle.

This range represents a 8-bit Hex value that specifies the Bridge Reverse Measure Time (REVTM) and the Minimum Bridge Forward On Time (FORTM) ranging from 600 nS to 3.4 µS each (see table and diagram below). Typically these values would be balanced. The table below shows the decimal value for each time.

Note that these are typical values and the currents may be unbalanced to fine tune the motor performance.

The default value for this parameter is 204 (0xCC), which represents a Reverse Measure Time and Minimum Forward On Time of 2.5 µS.

| Hex | Time | Hex | Time | Hex | Time | Hex | Time |
|---|---|---|---|---|---|---|---|
| 0x0 | 600 ns | 0x4 | 1.0 µs | 0x8 | 1.6 µs | 0xC | 2.5 µs |
| 0x1 | 700 ns | 0x5 | 1.1 µs | 0x9 | 1.8 µs | 0xD | 2.8 µs |
| 0x2 | 800 ns | 0x6 | 1.2 µs | 0xA | 2.0 µs | 0xE | 3.1 µs |
| 0x3 | 900 ns | 0x7 | 1.4 µs | 0xB | 2.2 µs | 0xF | 3.4 µs |

Table 7.8 PWM Mask Settings

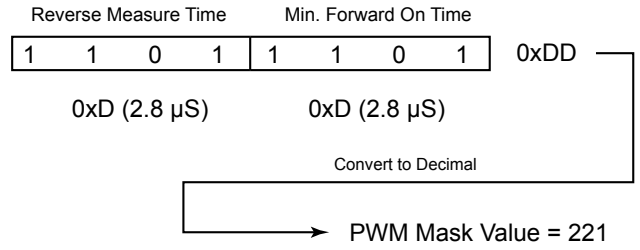



Figure 7.7 PWM mask bits

| Mask (hex) | Mask (dec) | REVTM | FORTM |
|---|---|---|---|
| 0x00 | 0 | 600 ns | 600 ns |
| 0x11 | 17 | 700 ns | 700 ns |
| 0x22 | 34 | 800 ns | 800 ns |
| 0x33 | 51 | 900 ns | 900 ns |
| 0x44 | 68 | 1.0 µs | 1.0 µs |
| 0x55 | 85 | 1.1 µs | 1.1 µs |
| 0x66 | 102 | 1.2 µs | 1.2 µs |
| 0x77 | 119 | 1.4 µs | 1.4 µs |

| Mask (hex) | Mask (dec) | REVTM | FORTM |
|---|---|---|---|
| 0x88 | 135 | 1.6 µs | 1.6 µs |
| 0x99 | 153 | 1.8 µs | 1.8 µs |
| 0xAA | 170 | 2.0 µs | 2.0 µs |
| 0xBB | 187 | 2.2 µs | 2.2 µs |
| 0xCC | 204 | 2.5 µs | 2.5 µs |
| 0xDD | 221 | 2.8 µs | 2.8 µs |
| 0xEE | 238 | 3.1 µs | 3.1 µs |
| 0xFF | 255 | 3.4 µs | 3.4 µs |

Table 7.9 Typical PWM Mask Settings

### 7.4.3 Maximum PWM duty cycle (%) <period> parameter

This parameter sets the maximum duty cycle as a percentage of the bridge PWM oscillator period. The range for this parameter is 0 to 95%. Entries above 95% will generate an out of range error (Error 21) and will not allow the setting to be written to the boot.

The default value for this parameter is 95%.

### 7.4.4 PWM frequency <sfreq> parameter

The PWM Frequency Parameter sets the initial and maximum frequencies for the PWM. As with the MASK parameter, the PWM Frequency is a two part 8-bit hex number which is entered as a decimal value ranging from 0 to 255.

The default for this 170 (0xAA) with an initial PWM Frequency of 20 kHz and a Maximum of 60 kHz.

| Hex | Freq. | Hex | Freq | Hex | Freq | Hex | Freq |
|---|---|---|---|---|---|---|---|
| 0x0 | 40 | 0x4 | 48 | 0x8 | 56 | 0xC | 64 |
| 0x1 | 42 | 0x5 | 50 | 0x9 | 58 | 0xD | 66 |
| 0x2 | 44 | 0x6 | 52 | 0xA | 60 | 0xE | 68 |
| 0x3 | 46 | 0x7 | 54 | 0xB | 62 | 0xF | 70 |

Table 7.10 Maximum PWM frequency

| Hex | Freq. | Hex | Freq | Hex | Freq | Hex | Freq |
|---|---|---|---|---|---|---|---|
| 0x0 | 10 | 0x4 | 14 | 0x8 | 18 | 0xC | 22 |
| 0x1 | 11 | 0x5 | 15 | 0x9 | 19 | 0xD | 23 |
| 0x2 | 12 | 0x6 | 16 | 0xA | 20 | 0xE | 24 |
| 0x3 | 13 | 0x7 | 17 | 0xB | 21 | 0xF | 25 |

Table 7.11 Initial PWM frequency

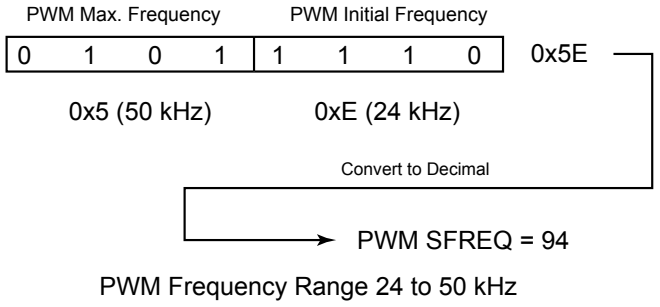



Figure 7.8 PWM frequency range

### 7.4.5 PWM checksum <chksum> parameter (boot write only)

**⚠ CAUTION**

This parameter should be left blank for testing parameters. Only insert a checksum if you have verified that these parameters cause the motor to perform as desired as the presence of the checksum WILL write the PWM settings to the boot, Limited Writes (8) are available.

The PWM checksum parameter is only used on the write cycle to write the PWM parameters to the boot. To write the PWM parameters to an Mforce a checksum must be sent along with the parameters.

PW = mask, period, frequency, checksum

To compute the checksum use the following steps:

1. Add the decimal values for mask, period, and frequency. Total = mask + period + frequency
2. Use A, B, or C to calculate checksum
   A. If the total is less than or equal to 256:
   (Total <= 256) checksum = 256 - Total
   B. If the total is greater than 256 but less than or equal to 512:
   (256 < Total <= 512) checksum = 512 - Total
   C. If the total is greater than 512, then:
   (Total > 512)checksum = 768 - Total

*Examples*

Add 102 + 90 + 10 = 202. The total is less than 256 so use A. to calculate checksum.

Checksum = 256 - 202
Checksum = 54
PW = 102,90,10,54

Add 238 + 95 + 170 = 503. The total is greater than 256 but less than or equal to 512 so use B. to calculate checksum.

Checksum = 512 - 503
Checksum = 9
PW = 238,95,170,9

Add 255 + 90 + 170 = 515. The total is greater than 512 so use C. to calculate checksum.

Checksum = 768 - 515
Checksum = 253
PW = 255,90,170,253

### 7.4.6 Boot writes remaining <boot_writes_remaining> parameter (read only)

This parameter will be the fourth parameter shown during a read of the PWM settings (PR PW).

The range is from 8 to 0 and represents the number of times remaining that the PWM settings can be written to the boot of the drive.

### 7.4.7 Example PWM settings by motor specifications

The following settings are based upon IMS settings per motor specifications and should serve as a baseline to work from with regard to the manufacturer specifications of the motor being utilized. Note that these are example settings ONLY!

| Frame Size | Stack Size | Phase Current ($A_{RMS}$) | Phase Resistance (Ω) | Phase Inductance (mH) | MASK <mask> | Duty Cycle <period> | Frequency <sfreq> | Checksum <chksum> |
|---|---|---|---|---|---|---|---|---|
| 14 | Single | 0.75 | 4.30 | 4 | 102 | 90 | 170 | 250 |
| 17 | Single | 1.5 | 1.30 | 2.1 | 136 | 90 | 170 | 116 |
| | Double | 1.5 | 2.10 | 5.0 | 136 | 90 | 170 | 116 |
| | Triple | 1.5 | 2.00 | 3.85 | 136 | 90 | 170 | 116 |
| 23 | Single | 2.4 | 0.95 | 2.4 | 136 | 90 | 170 | 116 |
| | Double | 2.4 | 1.20 | 4.0 | 136 | 90 | 170 | 116 |
| | Triple | 2.4 | 1.50 | 5.4 | 136 | 90 | 170 | 116 |
| 34 | Single | 6.3 | 0.25 | 1.6 | 168 | 95 | 170 | 79 |
| | Double | 6.3 | 0.35 | 3.3 | 220 | 95 | 170 | 27 |
| | Triple | 6.3 | 0.50 | 6.6 | 253 | 95 | 170 | 250 |
| MForce Default | | — | — | — | 204 | 95 | 170 | — |

Table 7.9 Example PWM settings

*Usage Example*

```
PW=102,90,150      'set pwm, DO NOT WRITE TO BOOT!
PW=136,90,170,116 'set pwm settings, write to boot

PR PW                        'read PWM
Response = 136,90,170,7      'last integer notes 7 boot
                             'writes remaining
```

This page intentionally left blank

# 8 Sample programs

This section is made up of several example programs designed to aid the user in discovering the MCode programming language.

## 8.1 Move on an input

```
'Last modified: 01/26/2006
'Purpose: Demonstrate move on input.

'System configuration
S1=0,0     'set IO1 to gen. purpose input, active LOW,
           'sinking
Ms=256     'set µstep resolution to 256 µsteps/step
Vi=200000  'set initial velocity to 200000 steps/sec
Vm=2500000 'set max velocity to 2500000 steps/sec
A=1000000  'set acceleration to 1000000 steps/sec2
D=A                'set deceleration equal to acceleration
Hc=2               'set motor holding current to 2%
Rc=75              'set motor run current to 75%
P=0                'set position counter to 0


'Main program
PG 1               'enter program mode at address 1
LB Ga              'label program Ga
  P=0              'initialize position counter
LB G1              'label program G1
  CL Kb,I1=1       'call subroutine Kb on input HIGH state
  H 10     'hold program execution 10 msec
  BR G1    'loop to G1

'Subroutine from trigger event
LB Kb              'declare subroutine Kb
  MA 51200         'move to absolute motor position 51200
  H                'suspend program execution until motion
                   'completes
  MA 0             'move to absolute motor position 0
  H                'suspend program execution until motion
                   'completes
  RT               'return from subroutine

E          'designate end of program
PG         'exit to immediate mode
```

## 8.2 Change velocity during a move

This program will demonstrate ability to change speed during move. The device does not have abiltity to change speed during point to point move, so we use the slew command with position trips. End position trip, decel and slew speed determine actual ending position. Program is written to print ending position to serial port 100 times for averaging, expected end position = 102400.

```
'System configuration
Ms=256      'set µstep resolution to 256 µsteps/step

Hc=20       'set motor holding current to 20%
Rc=100      'set motor run current to 75%

'Main program
PG 1
LB Ga       'Program Label Ga sets up local variables and
            'register values
  Vi=20000
  Vm=500000
  A=500000
  D=800000000
  R1=0
  R2=0
LB Gx               'Program label Gx sets up position
                    'trips and math functions
  P=0
  Tp=51200,Kb       'set position trip at P=51200
  Te=2
  SL 101200
  H
  H 250
  R1=R1+1           'increment R1
  R2=R2+P           'add position to r2 to set up position
calculation
  BR Gx,R1<100      'loop to Gx if R1 indicates less than
                    '100 moves
  R2=R2/100         'after 100 moves have completed, r2 is
                    'divided by 100
  PR "Average end pos = ",R2      'to obtain and print
                                  'final Pos.
  E

'Subroutines
LB Kb               'Subroutine called by position trip in
                    'Gx which doubles
  SL 202400         'the motor speed
  Tp=102290,Kc
  Te=2
  RT

LB Kc               'Subroutine executed by position trip
in Kb
  SL 0
  H
  RT

PG
```

## 8.3 Binary mask

This program will demonstrate ability to execute various subroutines depending on the binary value of inputs 1-3 while masking all i/o above input 3.

```
'System configuration
S1=0,1     'setup IO points 1-4, 9-11 as General purpose
user
S2=0,1
S3=0,1
S4=0,1
S9=0,1
S10=0,1
S11=0,1
S12=16,0   'set up IO point 12 as a gen. purp. output
Ms=256     'global system variable declarations
Vi=20000
Vm=1000000
A=500000
D=A
Hc=20
Rc=75

'Main program
PG 1
LB Ga
  P=0
LB G1
  R1=In       'capture input combined value to register 1
  R1=R1 & 7    'bits 00000111=7
  CL k0,R1 = 0     'Subroutine calls based on the decimal
                   'value of IO
  CL k1,R1 = 1     'points 1-3
  CL k2,R1 = 2
  CL k3,R1 = 3
  CL k4,R1 = 4
  CL k5,R1 = 5
  CL k6,R1 = 6
  CL k7,R1 = 7
  H 10
  BR G1
  E

'Subroutines
LB k0                    'Declare sub K0 executed if R1=0
  PR "Logic 000"
  MR 0*51200
  H
  H 200
  RT

LB k1                    'Declare sub K1 executed if R1=1
  PR "Logic 001"
  MR 1*51200
  H
  H 200
  RT

LB k2                    'Declare sub K2 executed if R1=2
  PR "logic 010"
  MR 2*51200
  H
  H 200
```

```
  RT

LB k3                    'Declare sub K3 executed if R1=3
  PR "Logic 011"
  MR 3*51200
  H
  H 200
  RT

LB k4                    'Declare sub K4 executed if R1=4
  PR "Logic 100"
  MR 4*51200
  H
  H 200
  RT

LB k5                    'Declare sub K5 executed if R1=5
  PR "Logic 101"
  MR 5*51200
  H
  H 200
  RT

LB k6                    'Declare sub K6 executed if R1=6
  PR "Logic 110"
  MR 6*51200
  H
  H 200
  RT

LB k7                    'Declare sub K7 executed if R1=7
  PR "Logic 111"
  MR 7*51200
  H
  H 200
  RT

E
PG
```

## 8.4 Closed Loop

This program illustrates closed loop control with an On Error (OE) routine which will perform math functions on the counters to display the position error.

```
'System configuration
Ms=256     'declare global system variables and flags
Hc=5
Rc=80
Ee=1               'encoder enabled
A=60000
D=A
Vi=2048
Vm=30000
S1=0,0
Sf=15              'encoder stall variables declared
Sm=0
Mt=50
VA Q1              'user variable Q1 declared

'Main program
PG 1
LB Ga              'Ga declares the error call and locally
                   'sets the position
  OE k1    'counter to 0 encoder counts
  P=0
LB Gb              'Gb performs ± motions and increments
                   'Q1 after each
  MR 51200 'until Q1 reaches 100
  H
  H 500
  MR -51200
  H
  H 500
  IC Q1
  BR Gb,Q1<100
E

'Subroutines
LB k1              'Sub K1 calculates the position error
                   'by dividing
  R3=C1/25         'actual motor steps moved by 25 and
                   'subtracts
  R1=R3 - C2       'the number of encoder counts in C2 to
                   'determine
  PR "Counts error = ",R1 'the counts error
  PR "Error = ",Er
  Er=0
  H 20
  RT

E
PG
```

## 8.5 User input into variables

This program demonstrates the ability to hold up program execution while the user enters multiple variables. Uses registers R1-R3 and a user declared flag for program control.

```
'System configuration
Ms=256     'Global variable declarations
Vi=10000
Vm=50000
A=10000
D=A
Hc=5
Rc=70
P=0
R1=0               'Registers set to 0
R2=0
R3=0
VA X1=0    'User flag X1 declared and set to 0

'Main program
PG 1
LB G1              'Local var-flg settings
  P=0
  X1=0
LB G2              'label for program hold loop
  H 20
  BR G1,X1=0 'command for pg hold loop
LB G3
  X1=0     'reset x1 to 0.
  A=R1     'set A to R1 value
  D=A
  Vm=R2
  MR R3
  H
  H 500
  BR G2
PG
```

## 8.6 Closed loop with homing

This program demonstrates the use of the home to home switch instruction (HM) in closed loop, also there is a move on input routine.

```
Ee=1                'Global variable and flag declarations
Vm=4096
Vi=Vm/50
A=20480
D=A
Hc=50
Rc=50
Mt=50
Sf=20
Sm=0
Db=5
S1=1,0     'Home input
S2=0,0     'Move on input
S3=17,0    'Moving output
S4=19,0    'Fault output

D1=100

'Program
PG 1
LB G1
 H 1000
 PR "C1 ",C1
 PR "C2 ",C2
 Pm=1
 PR "C1 ",C1
 PR "C2 ",C2
 H 5000
 HM 1
 H
 P=0
 LB G2
  BR G2,I2=0
 MR 7186
 H
 PR "p=",P
 BR G2
E
PG
```

## 8.7 Input trip

This program demonstrates the use input trips

```
'System configuration
Ms=256
Hc=0
Rc=100
D=800000000
Vi=10000
Vm=50000
S1=0,0
S2=16,0
S3=16,0
S4=16,0
O2=1
O3=1
O4=1

'Main program
PG 1
LB Ga
  CL K1  'call to configure 1st input trip
  SL 50000
LB Gb
  H 10
  BR Gb,Mv>0
  R3=R2-R1
  PR "Distance between inputs = ",R3
  H 1000
  PR " "
  BR Ga
  E

'Subroutines
LB K1         'Config for 1st input trip
  S1=0,0
  Ti=1,K2
  Te=1
  RT

LB K2          'Config for 2nd input trip
  R1=Pc
  S1=0,1
  Ti=1,K3
  Te=1
  RT

LB K3          'Sub for 2nd input trip
  R2=Pc
  SL 0

LB K4
  BR K4,Vc=1
  RT

E
PG
```

# 9 Error codes

A question mark <?> Displayed as a cursor indicates an error. To determine what the error is, type <pr er> in the terminal window. The device will respond with an error number displayed in the terminal window. The error number may then be referenced to this list.

| | |
|---|---|
| 0 | No Error |

## 9.1 I/O errors

| | |
|---|---|
| 6 | An I/O is already set to this type. Applies to non-General Purpose I/O. |
| 8 | Tried to set an I/O to an incorrect I/O type. |
| 9 | Tried to write to I/O set as Input or is “TYPED”. |
| 10 | Illegal I/O number. |
| 11 | Incorrect CLOCK type. |
| 12 | Illegal Trip / Capture |

## 9.2 Data errors

| | |
|---|---|
| 20 | Tried to set unknown variable or flag. Trying to set an undefined variable of flag. Also could be a typo. |
| 21 | Tried to set an incorrect value. Many variables have a range such as the Run Current (RC) which is 1 to 100%. As an example, you cannot set the RC to 110%. |
| 22 | VI is set greater than or equal to VM. The Initial Velocity is set equal to, or higher than the Maximum Velocity. VI must be less than VM. |
| 23 | VM is set less than or equal to VI. The Maximum Velocity is set equal to, or lower than the Initial Velocity. VM must be greater than VI. |
| 24 | Illegal data entered. Data has been entered that the device does not understand. |
| 25 | Variable or flag is read only. Read only flags and variables cannot be set. |
| 26 | Variable or flag is not allowed to be incremented or decremented. IC and DC cannot be used on variables or flags such as Baud and Version. |
| 27 | Trip not defined.Trying to enable a trip that has not yet been defined. |
| 28 | WARNING! Trying to redefine a program label or variable. This can be caused when you download a program over a program already saved. Before downloading a new or edited program, type <FD> and press ENTER to return the device to the Factory Defaults. You may also type <CP> and press ENTER to Clear the Program. |
| 29 | Trying to redefine a built in command, variable or flag. |
| 30 | Unknown label or user variable. Trying to Call or Branch to a Label or Variable that has not yet been defined. |

| | |
|---|---|
| 31 | Program label or user variable table is full. The table has a maximum capacity of 22 labels and/or user variables. |
| 32 | Trying to set a label (LB). You cannot name a label and then try to set it to a value. Example: Lable P1 (LB P1 ). The P1 cannot be used to set a variable such as P1=1000. |
| 33 | Trying to SET an Instruction. |
| 34 | Trying to Execute a Variable or Flag |
| 35 | Trying to Print Illegal Variable or Flag |
| 36 | Illegal Motor Count to Encoder Count Ratio |
| 37 | Command, Variable or Flag Not Available in Drive |
| 38 | Missing parameter separator |
| 39 | Trip on Position and Trip on Relative Distance not allowed together |

## 9.3 Program errors

| | |
|---|---|
| 40 | Program not running. If HOLD (H) is entered in Immediate Mode and a program is not running. |
| 41 | Stack overflow |
| 42 | Illegal program address. Tried to Clear, List, Execute, etc. an incorrect Program address. |
| 43 | Tried to overflow program stack. Calling a Sub-Routine or Trip Routine with no Return. |
| 44 | Program locked. User Programs can be Locked with the <LK> command. Once Locked, the program cannot be listed or edited in any way. |
| 45 | Trying to Overflow Program Space. |
| 46 | Not in Program Mode. |
| 47 | Tried to Write to Illegal Flash Address |
| 48 | Program Execution stopped by I/O set as Stop. |

## 9.4 Communications errors

| | |
|---|---|
| 60 | Not used |
| 61 | Trying to set illegal BAUD rate. The only Baud Rates accepted are those listed on the Properties Page of IMS Terminal. (4,800, 9,600, 19,200, 38,400, 115,200) |
| 62 | IV already pending or IF Flag already TRUE. |
| 63 | Character over-run. Character was received. Processor did not have time to process it and it was over-written by the next character. |
| 64 | Startup Calibration failed (Hybrid only) |

## 9.5 System errors

| | |
|---|---|
| 70 | FLASH Check Sum Fault |
| 71 | Internal Temperature Warning, 10C to Shutdown |
| 72 | Internal Over TEMP Fault, Disabling Drive |
| 73 | Tried to SAVE while moving |
| 74 | Tried to Initialize Parameters (IP) or Clear Program (CP) while Moving |
| 75 | Linear Overtemperature Error (For units without Internal Over Temp) |

## 9.6 Motion errors

| | | |
|---|---|---|
| 80 | | HOME switch not defined. Attempting to do a HOME (H) sequence but the Home Switch has not yet been defined. |
| 81 | | HOME type not defined. The HOME (HM or HI) Command has been programmed but with no type or an illegal type. (Types = 1, 2, 3, or 4) |
| 82 | | Went to both LIMITS and did not find home. The motion encroached both limits but did not trip the Home switch. Indicates a possible bad switch or a bad circuit. |
| 83 | | Reached plus LIMIT switch. The LIMIT switch in the plus direction was tripped. |
| 84 | | Reached minus LIMIT switch. The LIMIT switch in the minus direction was tripped. |
| 85 | | MA or MR isn't allowed during a HOME and a HOME isn't allowed while the device is in motion. |
| 86 | | Stall detected. The Stall Flag (ST) has been set to 1. |
| 87 | **MDrive** | In Clock Mode, JOG not allowed |
| | **Hybrid** | Not allowed to change AS mode while calibrating |
| 88 | **MDrive** | Following error |
| | **Hybrid** | Moves not allowed while calibration is in progress. |
| 89 | **MDrive** | Reserved |
| | **Hybrid** | Calibration not allowed while motion is in progress. |
| 90 | | Motion Variables are too low switching to EE=1 |
| 91 | | Motion stopped by I/O set as Stop. |
| 92 | | Position Error in Closed loop. motor will attempt tp position the shaft within the deadband, After failing 3 attempts Error 92 will be generated. Axis will continue to function normally. |
| 93 | | MR or MA not allowed while correcting position at end of previous MR or MA. |

## 9.7 Hybrid errors

| | |
|---|---|
| 100 | Configuration test done, encoder resolution mismatch |
| 101 | Configuration test done, encoder direction incorrect |
| 102 | Configuration test done, encoder resolution and direction incorrect |
| 103 | Configuration not done, drive not enabled |
| 104 | Locked rotor |
| 105 | Maximum position count reached |
| 106 | Lead limit reached |
| 107 | Lag limit reached |
| 108 | Reserved |
| 109 | Calibration failed |

# WARRANTY

## TWENTY-FOUR (24) MONTH LIMITED WARRANTY

Schneider Electric Motion USA warrants only to the purchaser of the Product from Schneider Electric Motion USA (the “Customer”) that the product purchased from Schneider Electric Motion USA (the “Product”) will be free from defects in materials and workmanship under the normal use and service for which the Product was designed for a period of 24 months from the date of purchase of the Product by the Customer. Customer’s exclusive remedy under this Limited Warranty shall be the repair or replacement, at Company’s sole option, of the Product, or any part of the Product, determined by Schneider Electric Motion USA to be defective. In order to exercise its warranty rights, Customer must notify Company in accordance with the instructions described under the heading “Obtaining Warranty Service”.

*NOTE: MDrive Motion Control electronics are not removable from the motor in the field. The entire unit must be returned to the factory for repair.*

This Limited Warranty does not extend to any Product damaged by reason of alteration, accident, abuse, neglect or misuse or improper or inadequate handling; improper or inadequate wiring utilized or installed in connection with the Product; installation, operation or use of the Product not made in strict accordance with the specifications and written instructions provided by IMS; use of the Product for any purpose other than those for which it was designed; ordinary wear and tear; disasters or Acts of God; unauthorized attachments, alterations or modifications to the Product; the misuse or failure of any item or equipment connected to the Product not supplied by Schneider Electric Motion USA; improper maintenance or repair of the Product; or any other reason or event not caused by Schneider Electric Motion USA.

Schneider Electric Motion USA HEREBY DISCLAIMS ALL OTHER WARRANTIES, WHETHER WRITTEN OR ORAL, EXPRESS OR IMPLIED BY LAW OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. CUSTOMER’S SOLE REMEDY FOR ANY DEFECTIVE PRODUCT WILL BE AS STATED ABOVE, AND IN NO EVENT WILL IMS BE LIABLE FOR INCIDENTAL, CONSEQUENTIAL, SPECIAL OR INDIRECT DAMAGES IN CONNECTION WITH THE PRODUCT.

This Limited Warranty shall be void if the Customer fails to comply with all of the terms set forth in this Limited Warranty. This Limited Warranty is the sole warranty offered by Schneider Electric Motion USA with respect to the Product. Schneider Electric Motion USA does not assume any other liability in connection with the sale of the Product. No representative of Schneider Electric Motion USA is authorized to extend this Limited Warranty or to change it in any manner whatsoever. No warranty applies to any party other than the original Customer.

Schneider Electric Motion USA and its directors, officers, employees, subsidiaries and affiliates shall not be liable for any damages arising from any loss of equipment, loss or distortion of data, loss of time, loss or destruction of software or other property, loss of production or profits, overhead costs, claims of third parties, labor or materials, penalties or liquidated damages or punitive damages, whatsoever, whether based upon breach of warranty, breach of contract, negligence, strict liability or any other legal theory, or other losses or expenses incurred by the Customer or any third party.

## OBTAINING WARRANTY SERVICE

If the Product was purchased from an Schneider Electric Motion USA Distributor, please contact that Distributor to obtain a Returned Material Authorization (RMA). If the Product was purchased directly from Schneider Electric Motion USA, please contact Customer Service at info@motion.schneider-electric.com or 860-295-6102 (Eastern Time Zone).

Customer shall prepay shipping charges for Products returned to Schneider Electric Motion USA for warranty service and Schneider Electric Motion USA shall pay for return of Products to Customer by ground transportation. However, Customer shall pay all shipping charges, duties and taxes for Products returned to Schneider Electric Motion USA from outside the United States.

**Schneider Electric Motion USA**
370 North Main Street, P.O. Box 457
Marlborough, CT 06447 - U.S.A.
Tel. +00 (1) 860 295-6102 - Fax +00 (1) 860 295-6107
e-mail: info@imshome.com
http://motion.schneider-electric.com

© Schneider Electric Motion USA All Rights Reserved.
REV062812
*Product Disclaimer and most recent product information at motion.schneider-electric.com.*