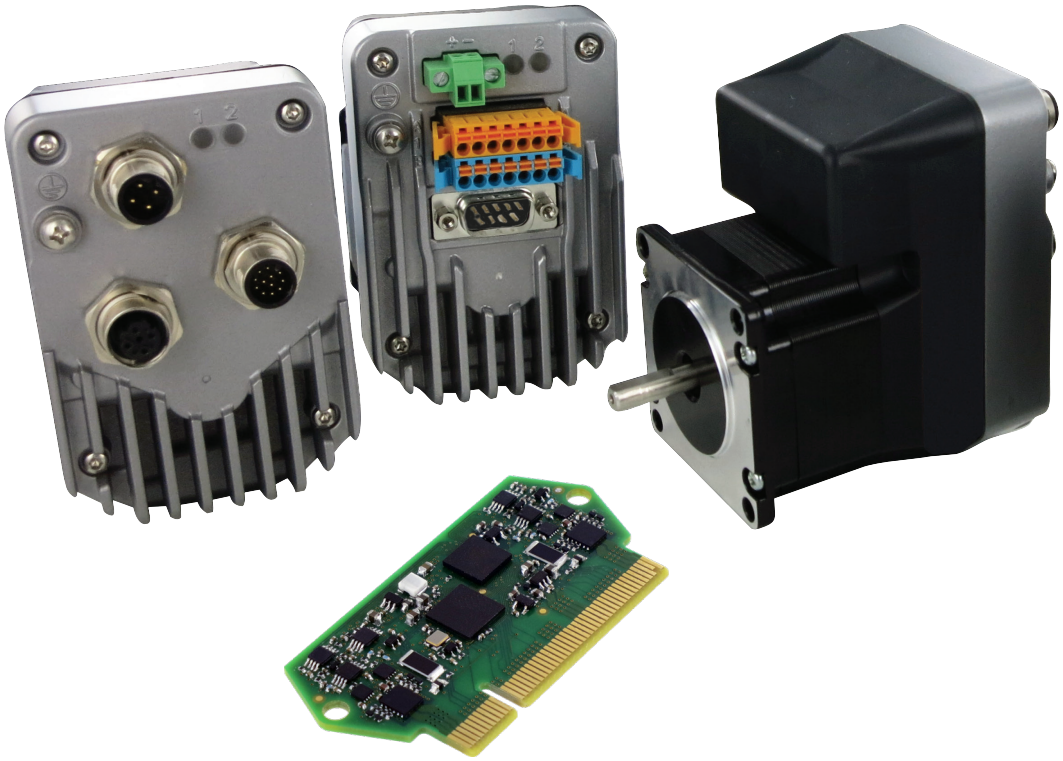
Programming and Software Reference

LMD MCode

LMD MDrive Motion Control
LMD MDrive Ethernet TCP/IP
LMD Motion Module

Publication LMD-MCODE-V2.06
03/2022



c UL us LISTED CE RoHS REACH

Novanta IMS

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof.

Neither Novanta IMS nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

You agree not to reproduce, other than for your own personal, noncommercial use, all or part of this document on any medium whatsoever without permission of Novanta IMS, given in writing. You also agree not to establish any hypertext links to this document or its content.

Novanta IMS does not grant any right or license for the personal and noncommercial use of the document or its content, except for a non-exclusive license to consult it on an “as is” basis, at your own risk. All other rights are reserved.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed. Failure to use Novanta IMS software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

For information on the availability of products, go to https://novantaims.com/

The information contained in the present document is subject to change without notice. The technical characteristics of the devices described in the present document also appear online. The characteristics that are presented in the present document should be the same as those characteristics that appear online. In line with our policy of constant improvement, we may revise content over time to improve clarity and accuracy. If there is a difference between the document and online information, use the online information as reference. All details provided are technical data which do not constitute warranted qualities.

Some of the product designations are registered trademarks of their respective owners, even if this is not explicitly indicated.

Table of Contents

Safety Information

Important Information

NOTICE Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, service, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a “Danger” or “Warning” safety label or message indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert of potential personal injury hazards. Obey all safety messages and labels that follow this symbol to avoid possible injury or death.

⚠ DANGER
DANGER indicates a hazardous situation which, if not avoided, **will result** in death or serious injury.

⚠ WARNING
WARNING indicates a hazardous situation which, if not avoided, **could result** in death or serious injury.

⚠ CAUTION
CAUTION indicates a hazardous situation which, if not avoided, **could result** in minor or moderate injury.

NOTICE
NOTICE is used to address practices not related to physical injury.

CyberSecurity Standards and Certification

Novanta IMS follows local regulations and uses additional industry established frameworks to conform to cyber security standards. Novanta IMS also takes an active part in the evolution of today’s industrial cyber security standards, contributing to these standards and frameworks.

In accordance with US California Senate Bill No. 327, and under direct guidance from Novanta, Novanta IMS has implemented a level of cyber-secure protection in the Ethernet-based Liberty MDrive (LMD) product line in order to protect these devices from outside cyber attacks. By choosing to disable these features, the user is acknowledging their acceptance of potential unauthorized outside access.

Qualification of Personnel

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Novanta IMS for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

Intended Use

The product may only be used in compliance with all applicable safety regulations and directives, the specified requirements, and the technical data.

Prior to using the product, perform a risk assessment in view of the planned application. Based on the results, the appropriate safety measures must be implemented.

Since the product is used as a component in an entire system, ensure the safety of persons by means of the design of this entire system (e.g., machine design).

Operate the product only with the specified cables and accessories. Use only genuine accessories and spare parts. The product must NEVER be operated in explosive atmospheres (e.g., hazardous locations, Ex areas).

Any use other than the use explicitly permitted is prohibited and can result in hazards.

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel.

Product Related

⚠ DANGER

HAZARD OF ELECTRICAL SHOCK, EXPLOSION, OR ARC FLASH

Remove all power from all devices before connecting or disconnecting inputs or outputs to any terminal or installing or removing any hardware.

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

When the system is started, the drives are usually out of the operator's view and cannot be visually monitored.

⚠ DANGER

EQUIPMENT OPERATION

Only start the system if there are no persons in the zone of operation.

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

Drives may perform unintended movements because of incorrect wiring, incorrect parameter settings, incorrect data, or other errors. Further, interference (e.g., electromagnetic interference (EMI)) may cause unpredictable responses in the system.

⚠WARNING

UNINTENDED MOVEMENT

- Carefully install the wiring in accordance with the electromagnetic compatibility (EMC) requirements.
- Do not operate the drive system with unknown parameter settings or data.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

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.

⚠WARNING

LOSS OF CONTROL

- 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, serious injury, or equipment damage.

[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”

Drives may perform unintended movements due to mechanical damage to connectors. Mechanical damage to the connectors may cause erratic or uncontrolled operation. Installation with a bent or broken mounting flange, motor shaft, or misaligned coupling may cause unintended behavior and possible destruction of system components as a result.

⚠WARNING

LOSS OF CONTROL, ERRATIC OPERATION AND DESTRUCTION OF MECHANICS

- Do not drop product.
- Leave product in protective packaging until ready for use.
- Carefully inspect connectors prior to installation in a system for mechanical damage.
- Carefully inspect motor shaft and ensure shaft rotates freely without binding.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Opening Liberty MDrive (LMD) heat sinks can affect factory-set encoder alignment and impact Hybrid Motion Technology (hMT) performance. Tamper seals are used to ensure factory hardware settings remain unaltered and match the encoder alignment set during the manufacturing process. If a seal is broken, the LMD product warranty is void.

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not open the LMD housing for any reason.
- Contact a Novanta IMS service representative if the product exhibits unexplained, erratic, or incorrect operation.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

When working on the wiring, inserting or removing connectors may cause unintended behavior and possible destruction of the system components.

⚠CAUTION

UNINTENDED BEHAVIOR

Remove all power before working on the wiring.

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

Radial (side) loading or axial (thrust) impacts on the shaft may result in premature bearing failure.

NOTICE

EXCESSIVE RADIAL OR AXIAL LOADS

- Do not place unsupported radial or side loads on motor shaft.
- Do not allow the shaft to be subject to impact forces or otherwise struck by external objects.

Failure to follow these instructions can result in equipment damage.

About this Manual

This manual covers the structure, syntax and use of the LMD MCode programming and control language for the LMD products developed and sold by Novanta IMS.

Writing Conventions and Symbols

Work Steps

If work steps must be performed consecutively, this sequence of steps is represented as follows:

- ■ Special prerequisites for the following work steps
- ▶ Step 1
- ◁ Specific response to this work step
- ▶ Step 2

If a response to a work step is indicated, this allows for verification that the work step has been performed correctly.

Unless otherwise stated, the individual steps must be performed in the specified sequence.

Bulleted Lists

The items in bulleted lists are sorted alphanumerically or by priority. Bulleted lists are structured as follows:

- Item 1 of bulleted list
- Item 2 of bulleted list
  - Subitem for 2
  - Subitem for 2
- Item 3 of bulleted list

Making Work Easier

Information on making work easier is highlighted by this symbol:

Sections highlighted this way provide supplementary information on making work easier.

Parameters

Parameters are shown as follows

RC Motor Run Current

Units of Measure

Measurements are given in both imperial and metric values. Metric values are given in parenthesis.

Examples:

1.00 in (25.4 mm)
100 oz-in (70 N-cm)

Documents and Software

The latest version of this document, supporting documentation, and software is available for download from: https://novantaims.com/dloads/

The following user's manuals are available for the LMD 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.

NOTICE: For the MDrivePlus or MForce Motion Control product, see the MCode Programming and Reference Manual for MDrivePlus and MForce products.

Product Software

LMD Software Suite (LSS)

The LSS Manual documents the installation and use of the programming tool for the LMD Motion and Ethernet products.

Associated Ethernet Protocols

The LMD Ethernet TCP/IP products support multiple industrial networking protocols:

- Modbus/TCP
- EtherNet/IP
- Profinet IO
- MCode/TCP

Documentation for these protocols is available in separate manuals. The LSS is the program used to commission, program and operate the LMD Motion products. The supporting documents and software can be downloaded from:
https://novantaims.com/dloads/

Instructions for installation and use of this software may be found in the LSS product manual.

LMD MCode Compatible Product List

LMD Programmable Motion Control

Integrated LMD Programmable Motion Control communicates over an RS-422/485 serial interface.

LMD MCode/TCP

LMD MCode/TCP is the LMD MCode language adapted to communicate over Ethernet TCP/IP networks. The function and usage are identical as with LMD Motion Control products with the exception that the commands related to RS-422/485 communication and serial party mode are disabled. LMD MCode/TCP connects to TCP or UDP port 503. Multidrop addressing is done using IPv4.

LMD Motion Module (Motion Control)

Uses an adaptation of LMD MCode for use with the LMD Motion Module communicating via the UART. Functional differences are:

- Closed loop hMTechnology functions not supported
- Adds PWM tuning functionality
- Analog input is not configurable in software

Website Directory

NOTE: Direct links are subject to change as website updates occur. Each of the websites below can also be accessed through menu options on the Novanta IMS Main Page: https://novantaims.com/

Downloads:
https://novantaims.com/dloads/

Resources:
https://novantaims.com/resources/

Certifications and Listing Information:
https://novantaims.com/dloads/certificationssustainability/

Contact and Support:
https://novantaims.com/contacts/

CyberSecurity Information:
https://novantaims.com/all-products/cybersecurity/

Knowledge Based Solutions:
https://knowledge.imshome.com/s/

Chapter 1
Introduction to LMD MCode

What's in this Chapter?

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

This chapter includes the following topics:

Operational Modes

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

1. **Immediate**: Commands are issued to and executed directly by the controller by user input into the terminal window.
2. **Program Execution Mode**: Program Execution Mode is used to input user programs into the motion controller, which can be stored for execution at a later time.

Basic Components of LMD MCode

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

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

Instructions

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

Motion

Motion instructions are those that lead to the movement of a motor. The syntax for these commands is as follows: type the command followed by a space or an equal sign, and then position data. Example: Type MA 2000 or MA=2000 and press Enter to 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 a space or an equal sign, then the data and press Enter. Example: O2=0 or O2 0 sets output 2 to 0.

Program

A program instruction allows program manipulation. The syntax of these varies due to the nature of the command. Examples: PG 100 toggles the system into program mode starting at address 100; BR LP, I1=1, which branches to a program location 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). Examples: EX 100 executes a program located at address 100 of program memory space, or EX K1 executes a location labeled K1.

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 336 user program labels and variables available. The syntax for each variable may differ.

Factory Defined Parameters, Registers, Keywords,and Reserved Words

These cannot be deleted and may only have the values modified where applicable. When an FD (Factory Default) instruction executes, these functions are reset to their factory default values.
There are two types of factory defined functions:

- **Read/Writable**: These can have the value altered 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 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 microsteps/sec (if EE=0) or encoder counts/sec (if EE=1).

User Defined Variables

The VA instruction allows the user to assign a name to a user variable (32-bit integer).

The restrictions for this command are:

1. Using a name that already exist as an MCode mnemonic is disallowed and will result in an *Error 24, Illegal data entered* if attempted.
2. In naming user variables, the first character must be alpha. It must be followed by 1 alpha character OR a number from 0 to 31.

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.

Flags

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

Factory Defined Flags

Factory defined flags are part of the MCode operating system and may not be deleted. When an FD (Factory Defaults) instruction executes given, these flags are returned to their factory default state. There are two types of factory defined flags:

- **Read/Writable**: This kind of flag is user alterable. They are typically used to set a condition or mode of operation for the 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 on 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.

Keywords

Keywords operate 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.

Math Functions

The LMD products are capable of either integer math or double-precision floating point math.

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

For floating point calculations, eight (8) registers are provided (F1 - F8).
Available floating point math functions are: ABS, SIN, COS, TAN, ARCSIN, ARCCOS, ARCTAN, PI, SQRT, LOG_2, LOG_{10}

NOTE: Floating-point calculations may only be performed using the registers provided (F1-F8). Registers R1-R4, MCode variables and user variables declared using the VA instruction are only capable of integer math.

Program Structuring

Proper structuring of the LMD MCode application ensures the ability to work efficiently and aids in troubleshooting the program. The following figure illustrates how the application can be blocked out to group the global system declarations, the program main body, and the subroutines.

```
'[VARIABLES]
Is = 1,0,0
Ms=256
A=1000000
D=A
Hc=2
Rc=75
P=0
```

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

```
'[PROGRAMS]
PG 1
  LB X1
   P=0
  LB X2
   CL Z1, I1-1
   H 10
   BR X1
```

Block 2: Main program
The first line places the device in program mode starting at address 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 Z1
   MA 51200
   H
   MA 0
   H
   RT
```

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

```
E
PG
```

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

Programming Aids

Motion Control Interface

One of the most powerful tools available is the Motion Control Interface module of the LMD Software Suite (LSS). The Motion Control Interface is a visual IDE (Integrated Development Environment) for developing, debugging, simulating and deploying LMD MDrive programs written in LMD MCode.

It features a program text editor, terminal emulation, program simulation and graphing. Program development may be accomplished by direct entry or by selecting an action and filling out a dialog.

User Labels

The LMD MCode programming language allows for 336 user labels for 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 LMD MCode programming language.

For purpose of this manual we have used the following example labels:

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

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

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

Label Example:

```
VA Q1              'Create user variable Q1
PG 100             'Program label for program 100
 LB G1             'Subroutine G1
 CL K1, I2=1       'Call Subroutine K1 if Input 2 is HIGH
 BR G1             'Unconditional Branch to G1
LB K1              'Declare Subroutine K1
```

Comments

LMD MCode allows for comments to be inserted in the program code. The comment character for the LMD 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 assist in trouble shooting the program.

Programming Reference

Chapter 2, "Command Summary" on page 26 contains detailed explanations and usage examples of each mnemonic in the LMD MCode Programming Language. These are organized alphabetically.

Appendix A, "Sample Programs" on page 161 contains a number of fully commented example programs that can be used to learn the basics of programming and using the various functions of the LMD MCode compatible device.

Commonly Used Variables and Instructions

Variables

MS (Microstep Resolution)

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

Motor rotation:	1.8° per step (200 steps/rev.)
Microsteps/step:	MS
Microsteps/rev:	MS * 200
MS default:	256 microsteps/full steps or 256 * 200 = 51200 microsteps/rev
To read:	PR MS
To write:	MS=<integer>
Notes:	MS values are predefined to 20 resolutions. See command details

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.

Open loop/AS>0:	Position from Counter 1 (C1) in microsteps
Encoder enabled:	Position from Counter 2 (C2) in encoder counts
To read:	PR P
To write:	P=0 will clear the position

VI (Initial Velocity)

Initial velocity in steps per second.

Default:	If EE= 0 - 1000 steps/sec, If EE=1 - 78 steps/sec
To read:	PR VI
To write:	VI=<integer>
Notes:	VI will return an error if set to a value greater than VM. The size of the step is a function of MS

VM (Maximum Velocity)

Maximum or final velocity in steps per second.

Default:	If EE= 0 - 768000 steps/sec If EE=1 - 60000 steps/sec
To read:	PR VM
To write:	VM=<integer>
Notes:	VM will return an error if set to a value less than VI. The size of the step is a function of MS

A (Acceleration)

Acceleration in steps per second^2.

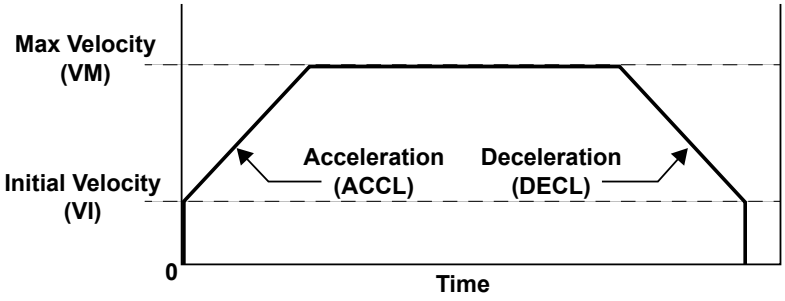
Default:	If EE= 0 - 1000000 steps/sec^2 If EE=1 - 78125 steps/sec
To read:	PR A
To write:	A=<integer>
Notes:	The velocity of the motor increases by <A> every second until VM, or the velocity commanded by a slew (SL)

D (Deceleration)

Deceleration in steps per second^2.

Default:	If EE= 0 - 1000000 steps/sec^2 If EE=1 - 78125 steps/sec
To read:	PR D
To write:	D=<integer>
Notes:	The velocity of the motor decreases by <D> every second until VI, or the velocity commanded by a slew (SL)

Refer to the following graphic to see the correlation between the commonly used variables when applied to motion control:



Motion Instructions

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

- All motion is programmed in either microsteps or encoder counts. When the encoder is disabled (EE=0), or hMTechnology is enabled (AS=1/2/3), motion is expressed in microsteps. In encoder mode, (EE=1), the motion commands are expressed in encoder counts.
- For example, using the default microstep resolution setting (MS=256): MR 51200 indexes the axis one revolution
- In encoder mode (EE=1) with a 1000 line (4000 count) encoder, the following applies MR 4000 indexes the axis one revolution.
- All motion is directly affected by the motion commands and variables. There are some factors impacting motion instructions. Chapter 3, "Software, Programming, and Application Notes" on page 131, covers these factors in detail.

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 positive direction
PR P        'the terminal screen will read 20000
MA 3000     'from the 20000 position, move in reverse to a position
            '3000 steps from 0
PR P        'the terminal screen will read 3000
```

Absolute moves are always relative to 0 (zero).

To program moves in the minus direction, type 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 positive direction
PR P        'the terminal screen will read 20000
MR 3000     'move 3000 steps from the current position in
            'the positive direction
PR P        'notice the position read is 23000 and not 3000
```

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

To program moves in the minus direction, type 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”.
- It is possible to 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 holding up program execution is not desired.

I/O Instructions

Is (Input Setup)

This command configures the Line, Type and Active state of inputs 1-4.

Type	Function	Description
0	General Purpose	Typical usage: to trigger events within a program
1	Home	When active triggers the homing routine as defined by the homing variable (HM)
2	Limit plus (+)	Functions as specified by the limit variable (LM). Results in an *Error 83: Positive limit reached* when a + limit activates.
3	Limit minus (—)	Functions as specified by the limit variable (LM). Results in an *Error 84: Negative limit reached* when a — limit activates.
4	G0	Executes a program at address 1 upon activation.
5	Soft stop	Stops motion with deceleration and halts program execution. If the program is paused (PS), the input is ignored.
6	Pause	Pause/resume program with motion.
7	Jog plus (+)	When active, jogs the motor in the positive direction at maximum velocity (VM). The jog enable (JE) flag must be active (JE=1) for this to function.

Type	Function	Description
8	Jog minus (—)	When active, jogs the motor in the minus direction at maximum velocity (VM). The jog enable (JE) flag must be active (JE=1) for this to function.
11	Reset	When set as a RESET input, the action is equivalent to a CTRL+C (^C) entered into a terminal. Note: If the input is in a sourcing configuration, active when high, ground the input first, or a reset occurs.
High speed capture input - available on input 1 only		
12	Capture	When set as a capture input is a momentary high-speed input that operates with the Trip Capture (TC) variable to run a subroutine upon the trip. It features variable input filtering ranging from 50 nS to 12.9 µS.
Clock input options - paired on inputs 3 and 4 only		
13	Step/Direction	Step clock (IN3) and direction (IN4) inputs
14	Encoder A/B	Encoder channel A (IN3) and B (IN4) inputs for following
15	Step Up/Down	Step up (IN3) and down (IN4) inputs

The syntax for setting up an input is

```
Is = <input #>, <type>, <active>
```

Set input 1 as general purpose active low	Is =**1,0,0**
Set input 2 as jog+ active high	Is =**2,7,1**
Set inputs 3 and 4 as Limit +/Limit –, active low	Is =**3,2,0** Is =**4,3,0**
Set input 1 as a capture input active high	Is =**1,12,1**

- Only input 1 may be set to the Capture function
- LMD NEMA 17 (42 mm) models are not equipped with Input 1
- Inputs may be set globally or locally (inside a program)
- The syntax to read the settings of the inputs is PR Is
- Clock input types (13, 14, 15) are only available on models with Firmware 6.001+

I<1-4> (Read Input State)

Used to retrieve the value of an individual input.

PR I1 reads the logic state of input 1 and display it in the terminal window.

BR K5, I2=0 branches to the program address tabled K5 when Input 2 is LOW

IN (Read Inputs as Group)

Used to read the decimal equivalent of the 4-bit binary nibble represented by all inputs collectively.

NOTE: Input 4 is the Most Significant Bit.

OS (Output Setup)

Sets the function of an output.

Type	Function	Description
16	General Purpose	Defines the output as a general purpose user output
17	Moving	Activates when the axis is in motion
18	Error	Indicates a software error condition occurred
19	Stall	Indicates a stalled condition exists. Stall detection mode (SM) must be enabled, and hMT must be off (AS=0), and encoder functions must be enabled (EE=1)
20	Velocity changing	Indicates when the axis is accelerating or decelerating
21	Locked rotor*	Indicates a locked rotor condition exists
23	Moving to position	Indicates when the axis is moving to a specified position
24	hMT active*	Indicates as when hMTechnology is active
25	Make-up active*	Indicates when position make-up is in process
29	Attention	Configurable to trigger on an attention event as defined by the AO variable
Available on output 3 only		
28	Trip	Indicates a trip condition (Output 3 ONLY, active low only)
**Grayed cells indicate an hMTechnology function, applicable to closed loop LMD products only.*		
Outputs 1 and 2 are not available on LMD NEMA 17 (42mm) products.		

The syntax for setting up an output is:

```
Os = <output #>, <type>, <active>
```

Set output 1 as general purpose active low	Os =1,16,0
Set output 2 as moving active high	Os =2,17,1
Set input 3 as a trip output active high	Os =3,28,1

- Only output 3 may be set to the trip function
- Outputs may be set globally or locally (inside a program)
- The syntax to read the settings of the outputs is PR Os

O <1 - 3> (Set Output)

Used to set the state of an output point.

O2=1 will set Output 2 TRUE

OT (Set Output Total)

Used to set the 3 bit binary equivalent of the decimal number represented by all 3 outputs collectively. Note the output 3 is the most significant bit.

OT=5 will set the outputs to 101

System Instructions

The following system instructions will be used frequently.

CP (Clear Program Memory)

The CP instruction is used to clear program memory space. CP must be followed by a save command S.

FD (Restore Factory 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 instantaneously (without deceleration).

CTRL+C (Software Reset)

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

Program Instructions

PG (Begin Program Mode)

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

```
PG 200     'Enter program mode at address 200
 xxxxx     'Program starting at address 200
 xxxxx     '     |
 xxxxx     '     |
PG         'Exit program mode
```

LB (User Label)

LMD 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 or a number from 0 to 31.
3. A program labeled SU will run on power-up

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

```
PG 10      'Enter program mode at address 10
LB k1      'Label command will name the program K1
 xxxxx     'Program named by LB command     xxxxx
 xxxxx     '
PG         'Exit program mode
```

BR (Branch)

Used to branch conditionally or unconditionally into another routine.

```
PG 30      'Enter program mode at address 30
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         'Exit program mode
```

E (End Program)

Designates the end of a program.

```
PG 15      'Enter program mode at address 15
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         'Exit program mode
```

CL (Call Subroutine)

Used to call a subroutine conditionally or unconditionally into another routine.

```
PG 45      'Enter program mode at address 45
LB K1      'Label command will name the program
 xxxxx
 xxxxx     'Program named by LB command
 xxxxx
 CL X1     'Unconditional call to subroutine label X1
E          'End program
PG         'Exit program mode

'[SUBROUTINES]
LB X1      'Label subroutine X1
 xxxxx     'Subroutine named by LB command
RT         'Return from subroutine
```

RT (Return From Subroutine)

Required to return from a subroutine to the program.

```
PG 100     'Enter program mode at address 100
LB K1      'Label command will name the program
 xxxxx
 xxxxx     'Program named by LB command
 xxxxx
 CL X1     'Unconditional call to subroutine label X1
E          'End program
PG         'Exit program mode

'[SUBROUTINES]
LB X1      'Label subroutine X1
 xxxxx     'Subroutine named by LB command
RT         'Return from subroutine
```

H (Hold Program Execution)

Delays program execution in milliseconds.

```
PG 25      'Enter program mode at address 25
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         'Exit program mode
```

PR (Print)

Outputs specified text and parameter values to a terminal or terminal software on a host PC.

```
PG 150             'Enter program mode at address 150
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                 'Exit program mode
```

VA (Define User Variable)

Command used to define a user variable consisting of 2 alphanumeric characters.

```
PG 65              'Enter program mode at address 65
VA Q1              'Define user variable Q1
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, Q1<10      'Cond branch to K1 if Q1 less than 10
E                  'End Program
PG                 'Exit program mode
```

Chapter 2
Command Summary

What's in this Chapter?

LMD MCode supports multiple families of motion control devices. Not all instructions, variables and flags apply to all motion control products. This chapter covers the Commands, each command's compatibility with LMD Motion and Ethernet TCP/IP products, and an explanation of each command.

This chapter includes the following topics:

Command Compatibility

All LMD Products

The commands listed in the following table are compatible with all LMD Motion and Ethernet TCP/IP products. Some function of the command may differ slightly between products. Attention should be paid to the command details for compatibility notes.

Mnemonic	Function	Access	Usage	Type	Notes	Compatibility			
Abbreviations Access: RO = Read only, RW = Read/Write, RC = Read/Clear, WO=Write only, CMD=Command/Instruction Usage: I - Immediate, P = Program, I/P = Immediate or program						LMD(O)	LMD(C)	LMD(A)	LMM
A	Acceleration	RW	I/P	Variable	—	✓	✓	✓	✓
AB	Absolute Value	—	I/P	Advanced Math/ Trigonometry	Firmware Version 6.001 +	✓	✓	✓	✓
AJ	Acceleration Jerk	RW	I/P	Variable	Firmware Version 6.001 +	✓	✓	✓	✓
AL	List All Parameters	RO	I	Instruction	—	✓	✓	✓	✓
AO	Attention Output Mask	RW	I/P	Variable	Certain attention events are product specific.	✓	✓	✓	✓
AT	Acceleration Type	RW	I/P	Variable	Firmware Version 6.001 +	✓	✓	✓	✓
AV	Approximate Velocity	RO	I/P	Variable	Firmware Version 6.001 +		✓	✓	
BE	Backlash Enable	RW	I/P	Flag	Firmware Version 6.001 +	✓	✓	✓	✓
BL	Backlash Amount	RW	I/P	Variable	Firmware Version 6.001 +	✓	✓	✓	✓
BM	Backlash Mode	RW	I/P	Variable	Firmware Version 6.001 +	✓	✓	✓	✓
BP	Break Point	CMD	I/P	Instruction	—	✓	✓	✓	✓
BR	Branch	CMD	P	Instruction	—	✓	✓	✓	✓
BY	Program Executing (busy)	RO	I	Flag	—	✓	✓	✓	✓
C_	Arc Cosine	—	I/P	Advanced Math/ Trigonometry	Firmware Version 6.001 +	✓	✓	✓	✓
C1	Counter 1 (motor count)	RW	I/P	Variable	—	✓	✓	✓	✓
C2	Counter 2 (encoder)	RW	I/P	Variable	LMD Closed Loop LMM Encoder required		✓	✓	✓
CE	Software Reset Enable/Disable	RW	I/P	Flag	—	✓	✓	✓	✓
CL	Call Subroutine	CMD	P	Instruction	—	✓	✓	✓	✓
CP	Clear Program memory	CMD	I	Instruction	—	✓	✓	✓	✓
CS	Cosine	—	I/P	Advanced Math/ Trigonometry	Firmware Version 6.001 +	✓	✓	✓	✓
CW	Clock Width	RW	I/P	Variable	—	✓	✓	✓	✓
D	Deceleration	RW	I/P	Variable	—	✓	✓	✓	✓
D<1-4>	Digital Input Filter	RW	I/P	Variable	—	✓	✓	✓	✓
D5	Analog Input Filter	RW	I/P	Variable	—	✓	✓		✓
DB	Encoder Deadband	RW	I/P	Variable	Encoder required		✓	✓	✓
DC	Decrement Variable	WO	I/P	Instruction	—	✓	✓	✓	✓
DE	Drive Enable/Disable	RW	I/P	Flag	—	✓	✓	✓	✓
DJ	Deceleration Jerk	RW	I/P	Variable	—	✓	✓	✓	✓
DT	Deceleration Type	RW	I/P	Variable	Firmware Version 6.001 +	✓	✓	✓	✓
E	End Program	CMD	I/P	Instruction	—	✓	✓	✓	✓
EE	Encoder Enable/Disable	RW	I/P	Flag	Encoder required		✓	✓	✓
EF	Error Flag	RO	I/P	Flag	—	✓	✓	✓	✓
EL	Encoder Lines	RW	I/P	Variable	—				✓
EM	Echo Mode	RW	I/P	Variable	—	✓	✓	✓	✓
ER	Error Register	RC	I/P	Variable	—	✓	✓	✓	✓
ES	Escape <esc> Mode	RW	I/P	Variable	—	✓	✓	✓	✓
EX	Execute Program	CMD	I	Instruction	—	✓	✓	✓	✓
F<1-8>	Floating Point Register	RW	I/P	Variable	Firmware Version 6.001 +	✓	✓	✓	✓

Mnemonic	Function	Access	Usage	Type	Notes	Compatibility			
Abbreviations Access: RO = Read only, RW = Read/Write, RC = Read/Clear, WO=Write only, CMD=Command/Instruction Usage: I - Immediate, P = Program, I/P = Immediate or program						LMD(O)	LMD(C)	LMD(A)	LMM
FC	Filter Capture Input	RW	I/P	Variable	Not applicable to LMDxx42	✓	✓	✓	✓
FD	Restore Factory Defaults	CMD	I	Instruction	—	✓	✓	✓	✓
FL	Following Mode Enable	RW	I/P	Flag	Firmware Version 6.001 +	✓	✓	✓	✓
FM	Filter Motion Inputs	RW	I/P	Variable	Firmware Version 6.001 +	✓	✓	✓	✓
FS	Index Offset Setting	RW	I/P	Variable	Firmware Version 6.001 + Encoder required		✓	✓	✓
FT	Reserved	—	—	—	—	—	—	—	—
H	Hold Program Execution	CMD	P	Instruction	—	✓	✓	✓	✓
HC	Hold Current	RW	I/P	Variable	—	✓	✓	✓	✓
HF	Home to Index Offset	WO	I/P	Instruction	Firmware Version 6.001 + Encoder required		✓	✓	✓
HI	Home to Index	WO	I/P	Instruction	Encoder required		✓	✓	✓
HM	Home to Home Switch	WO	I/P	Instruction	—	✓	✓	✓	✓
HT	Hold Current Delay	RW	I/P	Variable	—	✓	✓	✓	✓
I<1-4>	Read Input 1 - 4	RO	I/P	Variable	—	✓	✓	✓	✓
I5	Read Analog Input	RO	I/P	Variable	—	✓	✓		✓
I6	Read Encoder Index Mark	RO	I/P	Variable	Encoder required		✓	✓	✓
I7-I13	Reserved	—	—	—	—	—	—	—	—
IC	Increment Variable	CMD	I/P	Instruction	—	✓	✓	✓	✓
IF	Variable Input Pending	RC	I/P	Flag	—	✓	✓	✓	✓
IN	Read Inputs as Group	RO	I/P	Keyword	—	✓	✓	✓	✓
IP	Initialize Parameters	WO	I/P	Instruction	—	✓	✓	✓	✓
IS<1-4>	Input 1 - 4 Setup	RW	I/P	Instruction	Clock input types (13, 14, 15) available with Firmware 6.001+; No I1 on LMD NEMA 17 (42 mm)	✓	✓	✓	✓
IS 5	Analog Input Setup	RW	I/P	Instruction	Analog input on the LMM is not configurable.	✓	✓		✓
IS 6	Encoder Index Setup	RW	I/P	Instruction	Encoder required		✓	✓	✓
IT	Internal Temperature	RO	I/P	Keyword	—	✓	✓	✓	✓
IV	Input to Variable	CMD	P	Instruction	—	✓	✓	✓	✓
JE	Jog Enable/Disable	RW	I/P	Flag	—	✓	✓	✓	✓
L	List Program Space	CMD	I	Instruction	—	✓	✓	✓	✓
L_	Logarithm (Base 10)	—	I/P	Advanced Math/ Trigonometry	Firmware Version 6.001 +	✓	✓	✓	✓
LB	Label	CMD	P	Instruction	—	✓	✓	✓	✓
LK	Lock Program	RW	I/P	Flag	—	✓	✓	✓	✓
LM	Limit Mode	RW	I/P	Variable	—	✓	✓	✓	✓
LO	Logarithm (Base e)	—	I/P	Advanced Math/ Trigonometry	Firmware Version 6.001 +	✓	✓	✓	✓
LS	Software Limit	RW	I/P	Variable	Firmware Version 6.001 +	✓	✓	✓	✓
MA	Move Absolute	WO	I/P	Instruction	—	✓	✓	✓	✓
MD	Motion Mode	RO	I/P	Variable	—	✓	✓	✓	✓
MP	Moving to Position	RO	I/P	Flag	—	✓	✓	✓	✓
MR	Move Relative	WO	I/P	Instruction	—	✓	✓	✓	✓
MS	Microstep Resolution	RW	I/P	Variable	—	✓	✓	✓	✓
MT	Motor Settling Delay Time	RW	I/P	Variable	—	✓	✓	✓	✓
MV	Moving	RO	I/P	Flag	—	✓	✓	✓	✓
NE	Numeric Enable/Disable	RW	I/P	Variable	—	✓	✓	✓	✓

Mnemonic	Function	Access	Usage	Type	Notes	Compatibility			
Abbreviations Access: RO = Read only, RW = Read/Write, RC = Read/Clear, WO=Write only, CMD=Command/Instruction Usage: I - Immediate, P = Program, I/P = Immediate or program						LMD(O)	LMD(C)	LMD(A)	LMM
O<1-3>	Set Output Number	WO	I/P	Instruction	—	✓	✓	✓	✓
OE	On Error Handler	WO	I/P	Instruction	—	✓	✓	✓	✓
OF	Output Fault	RO	I/P	Variable	—	✓	✓	✓	
OS<1-3>	Setup Outputs 1 to 3	RW	I/P	Instruction	LMD NEMA 17 (42 mm) has O3 only. LMD(A) NEMA 23 (57 mm) and NEMA 34 (85 mm) do not have an O2. Some hMT specific functions not available on all products	✓	✓	✓	✓
OT	Write All Outputs	WO	I/P	Instruction	LMD NEMA 17 (42 mm) has O3 only.	✓	✓	✓	✓
P	Position Counter	RW	I/P	Instruction	—	✓	✓	✓	✓
PC	Captured Position	RW	I/P	Instruction	—	✓	✓	✓	✓
PF	Print Format	RW	I/P	Variable	Firmware Version 6.001 +	✓	✓	✓	✓
PG	Program Mode	CMD	I/P	Instruction	—	✓	✓	✓	✓
PI	Value of Pi 3.141592654	—	I/P	Advanced Math/ Trigonometry	Firmware Version 6.001 +	✓	✓	✓	✓
PK	Reserved	—	—	—	—	—	—	—	—
PM	Position Maintenance	RW	I/P	Flag	Encoder required		✓	✓	✓
PN	Part Number	RO	I/P	Variable	—	✓	✓	✓	✓
PR	Print Specified Data/Text	WO	I/P	Instruction	—	✓	✓	✓	✓
PS	Pause Program	CMD	I/P	Instruction	—	✓	✓	✓	✓
PW	PWM Mask Setting	RW	I/P	Variable	LMM Only				✓
R<1-4>	User Register	RW	I/P	Variable	—	✓	✓	✓	✓
RA	Radians/degrees	RW	I/P	Variable	Firmware Version 6.001 +	✓	✓	✓	✓
RC	Run Current	RW	I/P	Variable	—	✓	✓	✓	✓
RD	Reverse Direction	CMD	I/P	Variable	—	✓	✓	✓	✓
RP	Referenced Position	RO	I/P	Variable	—	✓	✓	✓	✓
RS	Resume Program	CMD	I	Instruction	—	✓	✓	✓	✓
RT	Return from Subroutine	CMD	P	Instruction	—	✓	✓	✓	✓
S	Save Program/Parameters	WO	I/P	Instruction	—	✓	✓	✓	✓
S_	Arc Sine	—	I/P	Advanced Math/ Trigonometry	Firmware Version 6.001 +	✓	✓	✓	✓
SA	Step Angle	RW	I/P	Variable	—				✓
SC	System Configuration Test	WO	I/P	Instruction	—		✓	✓	
SF	Stall Factor	RW	I/P	Variable	Encoder required LMD Closed Loop		✓	✓	✓
SI	Sine	—	`F1 = SI F2`	Advanced Math/ Trigonometry	Firmware Version 6.001 +	✓	✓	✓	✓
SL	Slew at Velocity	WO	I/P	Instruction	—	✓	✓	✓	✓
SM	Stall Detect Mode	RW	I/P	Variable	Encoder required LMD Closed Loop		✓	✓	✓
SN	Serial Number	RO	I/P	Keyword	—	✓	✓	✓	✓
SQ	Square Root	—	I/P	Advanced Math/ Trigonometry	Firmware Version 6.001 +	✓	✓	✓	✓
ST	Stall Flag	RO	I/P	Flag	Encoder required LMD Closed Loop		✓	✓	✓
SU	Start Up	CMD	I/P	Keyword	—	✓	✓	✓	✓
T_	Arc tangent	—	I/P	Advanced Math/ Trigonometry	Firmware Version 6.001 +	✓	✓	✓	✓
TC	Trip on Capture	RW	I/P	Variable	—	✓	✓	✓	✓

Mnemonic	Function	Access	Usage	Type	Notes	Compatibility			
Abbreviations Access: RO = Read only, RW = Read/Write, RC = Read/Clear, WO=Write only, CMD=Command/Instruction Usage: I - Immediate, P = Program, I/P = Immediate or program						LMD(O)	LMD(C)	LMD(A)	LMM
TE	Trip Enable	RW	I/P	Variable	Specific product limitations	✓	✓	✓	✓
TG	Tangent	—	I/P	Advanced Math/ Trigonometry	Firmware Version 6.001 +	✓	✓	✓	✓
TI	Trip on Input	RW	I/P	Variable	—	✓	✓	✓	✓
TM	Trip on Main Power Loss	RW	I/P	Variable	—	✓	✓	✓	✓
TP	Trip on Position	RW	I/P	Variable	—	✓	✓	✓	✓
TR	Trip on Relative	RW	I/P	Variable	—	✓	✓	✓	✓
TT	Trip on Time	RW	I/P	Variable	—	✓	✓	✓	✓
UG	Upgrade Firmware	WO	I/P	Instruction	—	✓	✓	✓	✓
UV	User Variables	RO	I/P	Keyword		✓	✓	✓	✓
V	Current Velocity	RO	I/P	Keyword	—	✓	✓	✓	✓
VA	Define User Variable	CMD	I/P	Instruction	—	✓	✓	✓	✓
VB	Abs Encoder Backup Voltage	RO	I/P	Variable	Absolute Encoder only			✓	
VC	Velocity Changing	RO	I/P	Flag	—	✓	✓	✓	✓
VI	Initial Velocity	RW	I/P	Variable	—	✓	✓	✓	✓
VM	Max. Velocity	RW	I/P	Variable	—	✓	✓	✓	✓
VR	Version	RO	I/P	Keyword	—	✓	✓	✓	✓
VT	Read Voltage	RO	I/P	Keyword	—	✓	✓	✓	✓
WT	Warning Temperature	RW	I/P	Variable	—	✓	✓	✓	✓

LMD Serial Products

The commands listed in following table apply specifically to LMD Motion products with a serial interface (RS-422/485/UART).

- LMD Motion Control (P/N LMDxM)
- LMD Motion Module (P/N LMM-15-M)

These commands will return an *Error 37: Command/Variable/Flag not available* if used with any other product.

Mnemonic	Function	Access	Usage	Type	Compatibility			
Abbreviations Access: RO = Read only, RW = Read/Write, RC = Read/Clear, WO=Write only Usage: I - Immediate, P = Program, I/P = Immediate or program					LMD(O)	LMD(C)	LMD(A)	LMM
BD	BAUD Rate	RW	I/P	Variable	✓	✓	✓	✓
CK	Checksum Mode	RW	I/P	Variable	✓	✓	✓	✓
DG	Disable Global Response	RW	I/P	Flag	✓	✓	✓	✓
DN	Device Name	RW	I/P	Variable	✓	✓	✓	✓
PY	Party Mode Enable	RW	I/P	Variable	✓	✓	✓	✓
QD	Device Queued	RW	I	Flag	✓	✓	✓	✓
SC	System Configuration Test	WO	IP	Instruction		✓	✓	

LMD Motion Module

The commands listed in the following table apply specifically to LMD Motion Module.

- LMD Motion Module (P/N LMM-15-M)

These commands will return an *Error 37: Command/Variable/Flag not available* if used with any other product.

Mnemonic	Function	Access	Usage	Type	Notes	Compatibility			
Abbreviations Access: RO = Read only, RW = Read/Write, RC = Read/Clear, WO=Write only Usage: I - Immediate, P = Program, I/P = Immediate or program						LMD(O)	LMD(C)	LMD(A)	LMM
PW	PWM Mask	RW	I/P	Variable	—				✓
SA	Step Angle	RW	I/P	Variable	—				✓

hMTechnology Specific

The commands listed in the following table apply specifically to LMD closed loop and absolute products with the hMTechnology functions.

- LMD Motion Control (P/N LMDCMxxx and LMDAMxxx)
- LMD Ethernet TCP/IP (P/N LMDCExxx and LMDAExxx)

These commands will return an *Error 37: Command/Variable/Flag not available* if used on open loop or LMD Motion Module products.

Mnemonic	Function	Access	Usage	Type	Notes	Compatibility			
Abbreviations Access: RO = Read only, RW = Read/Write, RC = Read/Clear, WO=Write only, CMD=Command/Instruction Usage: I - Immediate, P = Program, I/P = Immediate or program						LMD(O)	LMD(C)	LMD(A)	LMM
AF	hMT Status	RO	I/P	Flag	—		✓	✓	
AS	hMT Mode	RW	I/P	Variable	—		✓	✓	
CB	Control Bounds	RW	I/P	Variable	—		✓	✓	
CF	Clear Locked Rotor	CMD	I/P	Instruction	—		✓	✓	
LD	Lead Limit	RW	I/P	Variable	—		✓	✓	
LG	Lag Limit	RW	I/P	Variable	—		✓	✓	
LL	Position Lead/Lag Register	RO	I/P	Variable	—		✓	✓	
LR	Locked rotor	RO	I/P	Flag	—		✓	✓	
LT	Locked Rotor Timeout	RW	I/P	Variable	—		✓	✓	
MF	Makeup Frequency	RW	I/P	Variable	—		✓	✓	
MU	Position Makeup Mode	RW	I/P	Variable	—		✓	✓	
TA	Trip on hMT Status	RW	I/P	Variable	—		✓	✓	
TD	Torque Direction	RW	I/P	Variable	—		✓	✓	
TQ	Torque Percent	RW	I/P	Variable	—		✓	✓	
TS	Torque Speed	RW	I/P	Variable	—		✓	✓	
VF	Torque Velocity Filter	RW	I/P	Variable	—		✓	✓	

Math Functions

The MCode math, comparison, logic, and trigonometric operators shown in the following table are compatible with all LMD Motion Control and Ethernet TCP/IP products. The advanced floating point math and trigonometric functions are the ONLY available in models with Firmware Version 6.001 +.

NOTE: Math and trigonometric functions performed outside the floating point registers (F1 - F8) will be rounded down to the nearest integer.

Operator	Function	Usage	Compatibility			
			LMD(O)	LMD(C)	LMD(A)	LMM
+	Add Two Variables and/or Flags	`R1 + R2`	✓	✓	✓	✓
-	Subtract Two Variables and/or Flags	`R1 - R2`	✓	✓	✓	✓
*	Multiply Two Variables and/or Flags	`R1 * R2`	✓	✓	✓	✓
/	Divide Two Variables and/or Flags	`R1/R2`	✓	✓	✓	✓
<>	Not Equal	`R1 <> R2`	✓	✓	✓	✓
=	Equal	`R1 = R2`	✓	✓	✓	✓
<	Less Than	`R1 < R2`	✓	✓	✓	✓
<=	Less Than or Equal	`R1 <= R2`	✓	✓	✓	✓
>	Greater Than	`R1 > R2`	✓	✓	✓	✓
>=	Greater Than or Equal	`R1 >= R2`	✓	✓	✓	✓
&	AND (Bitwise)	`R1 = R2 & R3`	✓	✓	✓	✓
|	OR (Bitwise)	`R1 = R2 | R3`	✓	✓	✓	✓
^	XOR (Bitwise)	`R1 = R2 ^ R3`	✓	✓	✓	✓
!	NOT (Bitwise)	`R1 = R2 ! R3`	✓	✓	✓	✓
Floating point and trigonometric functions - Firmware Version 6.001 +						
AB	Absolute Value	`F1 = AB R1`	✓	✓	✓	✓
CS	Cosine	`F1 = CS F2`	✓	✓	✓	✓
C_	Arc Cosine	`F1 = C_ F2`	✓	✓	✓	✓
LO	Logarithm (Base e)	`F1 = LO F2`	✓	✓	✓	✓
L_	Logarithm (Base 10)	`F1 = L_ F2`	✓	✓	✓	✓
PI	Value of Pi 3.141592654	`F1 = PI`	✓	✓	✓	✓
SI	Sine	`F1 = SI F2`	✓	✓	✓	✓
SQ	Square Root	`F1 = SQ F2`	✓	✓	✓	✓
S_	Arc Sine	`F1 = S_ F2`	✓	✓	✓	✓
TG	Tangent	`F1 = TG F2`	✓	✓	✓	✓
T_	Arc tangent	`F1 = T_ F2`	✓	✓	✓	✓

Commands

Mnemonic	Function	Function Group	Access	Usage
A	Set/Read Acceleration	Motion variable	RW	Program/Immediate

Compatibility: ■ LMD(O) ■ LMD(C) ■ LMD(A) ■ LMM

Notes: —

DESCRIPTION

Defines the acceleration rate when changing velocity.

If the value of **A** is 768000 steps per second^2, the motor accelerates at a rate of 768000 steps per second^2, every second at the default linear acceleration type. If the **VM** (Maximum Velocity) is set at 768000 microsteps per second, it takes 10 seconds to reach **VM** from an Initial Velocity (**VI**) of 0 (axis stopped).

The primary factor determining the range and units applied to the acceleration profile is the logic state of the **EE** (Encoder Enable) flag. When disabled (**EE**=0), acceleration is measured in steps/sec^2. When enabled (**EE**=1), the value represents encoder counts/sec^2. When **EE** is changed, **A**, **D**, **VI** and **VM** are recalculated.

Secondary factors impacting acceleration is the configuration of **AT** (Acceleration Type) and **AJ** (Acceleration Jerk). **AT** adds triangle and sinusoidal S-curve capability to the default linear acceleration type. The **AJ** variable allows the user to set a constant value to compensate for load oscillations.

Range:	**Clock Mode**	66 to 1100 X 10^6	**Units:**	**Clock Mode**	steps/sec^2	**Default:**	**Clock Mode** 1000000
	Encoder	85937496		**Encoder**	counts/sec^2		**Encoder** 78125

Syntax: A=<integer>, PR A

CODE EXAMPLES

`A=20000`	Set acceleration to 20000 steps/sec^2
`A=Q1`	Set acceleration to be equal to user variable Q1
`PR A`	Print acceleration value

RELATED

AJ (Accel Jerk) EE (Encoder Enable)
AT (Accel Type) VI (Initial Velocity)
D (Deceleration) VM (Max Velocity)

NETWORK PROTOCOL EQUIVALENTS

	Class	Instance	Attribute	Data Type		
Ethernet/IP	0x66	1	0x01	UDINT	**Modbus/TCP**	0x0000

Mnemonic	Function	Function Group	Access	Usage
AF	Read hMT status	Status Flag	RO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

The **AF** status flag holds the status code reflecting the last hMT status event. In the case where multiple status conditions exist, the returned result will represent the sum of the active status conditions.

The hMT will initialize following a power up/reset of the system. This results in the **AF** flag showing a status code '128' (hMT Initialization Complete) following a power up/reset of the system.

Example: **PR AF** returns a status code of 133, indicating that **LD** (Lead Limit) and **LL** (Max. Lead/Lag Limit) were reached.

Status Code	Condition
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	hMT initialization complete
256	hMT initialization error

Range:	Units:	Default:
See above	—	—

Syntax: PR AF, BR <label/address>, AF = <value>

CODE EXAMPLES

`PR AF`	Print the status of AF to the terminal
`BR Q1,AF&2`	Branch to Q1 if AF not 0 - indicating LG (Lag Limit) is reached
`CL Q1,AF=132`	Call subroutine Q1 if a lead or lag limit is reached

RELATED

AO (Attention Output)
AS (hMT Mode)
TA (hMT Status Trip)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x008F
	0x6A	1	0x01	UINT		

Mnemonic	Function	Function Group	Access	Usage
AJ	Acceleration Jerk	Motion Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

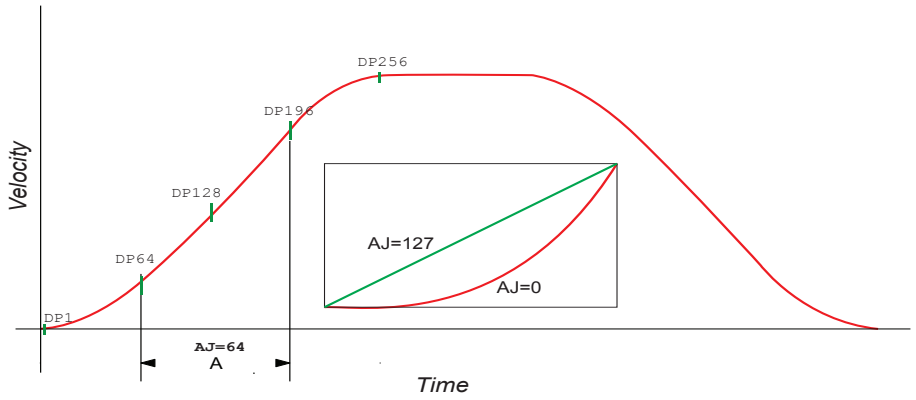
Notes: Firmware 6.001+

DESCRIPTION

Acceleration Jerk is the rate of change of acceleration, or, the derivative of acceleration with respect to time. The acceleration jerk variable only impacts the motion profile when an S-curve acceleration type (**AT**=2 or **AT**=3) is selected. The jerk value may be adjusted to any integer value between 0 and 127 to compensate for load oscillations.

The motion logic in the LMD product samples 256 data points during the acceleration ramp. The value applied to **AJ** represents the number of data points on either side of the center of the acceleration table, at which the acceleration is at a constant, linear acceleration at the value defined by **A** (Acceleration).

Example: As shown in this graphic, with **AJ**=64, the Acceleration ramp will be constant for 128 samples, or 64 samples on either side of the ramp center.



Range:	0 to 127	Units:	—	Default:	0

Syntax: AJ=<value>, PR AJ

CODE EXAMPLES

`AJ=32`	Set acceleration jerk to 32
`PR AJ`	Read the value of AJ to the terminal window

RELATED

A (Acceleration) AT (Accel Type) D (Deceleration) DT (Decel Type) DJ (Decel Jerk) VI (Initial Velocity) VM (Max Velocity)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x66	1	0x14	USINT		Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes

Mnemonic	Function	Function Group	Access	Usage
AL	List All Parameters	Instruction	RO	Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

The **AL** keyword is used with the **PR** (PRINT) instruction to print the value/state of all variables and flags to the terminal program.

NOTE: The **PR AL** command can not be performed during program execution..

Range:	—	Units:	—	Default:	—

Syntax: PR AL

CODE EXAMPLES

`PR AL`	Read the value of all parameters to the terminal window

RELATED

FD (Factory Defaults) IP (Initialize Parameters)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	—	—	—	—		—

Mnemonic	Function	Function Group	Access	Usage
AO	Set/Read Attention Output Mask	I/O variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Trigger events vary with hMT

DESCRIPTION

The **AO** variable will define the condition(s) on which the attention output triggers LED 2, or to the output point assigned to the Attention Output function.

If multiple conditions need to trigger the output, the result is additive.

Example:
Lead limit (4) and Lag limit (8), **AO** will equal 12,
Moving flag (16384) and Stall Flag (32768), **AO** will equal 49152

NOTE: Available trigger events will vary depend on the model LMD Motion Control product.

Mask	Description	LED Color (Green = Default)
1	Error flag	Red
2	Locked rotor*	
4	Lead limit*	
8	Lag limit*	
16	hMT active*	
32	Calibration active*	
64	Over temperature	
128	Software error	
256	At zero cross	
512	Current reduction active	
1024	Make-up active*	
2048	Drive disabled (DE=0)*	
4096	Warning temperature	
8192	Voltage warning	
16384	Moving flag	
32768	Stall flag*	
65536	Position Referenced	

*encoder required for function

Range:	0 - 4,294,967,295	Units:	—	Default:	0

Syntax: AO=<mask>

CODE EXAMPLES

`AO=512`	Attention active when at hold current level
`PR AO`	Return the AO mask value to the terminal

RELATED

O<1-3> (Set Output)

OS (Output Setup)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes
	0x67	1	0x01	UINT		

Mnemonic	Function	Function Group	Access	Usage
AS	Set/Read hMT Mode Select	Motion Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Sets the operating mode for hMTechnology device to one of four modes: Off, Fixed Current, Variable Current and Torque.

These modes will determine the operational characteristics of the closed loop LMD Motion product.

NOTE: **MS** (Microstep Resolution) cannot be set lower than ten (10) when hMTechnology is enabled.

Mode	Operation
0	hMT inactive (default): Motor performs as a traditional stepper.
1	Fixed current mode, motor current will be as specified by the run current (RC) and hold current (HC) variables
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). **IMPORTANT:** **- Motion will commence IMMEDIATELY upon setting AS=3 without warning.** **- Limit inputs do not function in torque mode.**

Range: 0 - 3 **Units:** — **Default:** 0

Syntax: AS=<mode>, PR AS

CODE EXAMPLES

`AS=2`	Set the hMT mode to variable current
`PR AS`	Return the hMT mode setting to the terminal

RELATED

AV (Actual hMT Velocity) RC (Run Current) HC (Hold Current) LR (Locked Rotor) MF (Makeup Freq) MU (Position Makeup) TD (Torque Dir) TQ (Torque %) TS (Torque Speed) MS (Microstep resolution)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x008E
	0x6A	1	0x02	USINT		

Changing hMT mode to torque mode (AS=3) will result in immediate motion at the velocity specified by the torque speed (TS) variable.

⚠WARNING

EXECUTION OF MOTION

- Motion will occur immediately if AS=3

Failure to follow these instructions can result in death, serious injury or equipment damage.

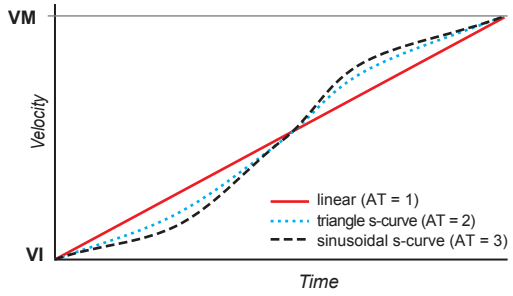
Mnemonic	Function	Function Group	Access	Usage
AT	Set/Read Acceleration Type	Motion Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+

DESCRIPTION

Defines the type of acceleration profile used when a move is executed. There are three (3) acceleration types available for LMD products: Linear (constant), triangle s-curve and sinusoidal s-curve.



Type	Accel Ramp	Description
1	Linear (Default)	Constant smooth (linear) acceleration from initial to max velocity.
2	Triangle	Triangle s-curve profile.
3	Sinusoidal	The sinusoidal s-curve profile is very similar to the triangle s-curve. The main difference is that it has less jerk when starting or stopping.

Range: 1- 3 **Units:** — **Default:** 1 - Linear

Syntax: AT=<type>, PR AT

CODE EXAMPLES

`AT=3`	Set the Acceleration type to sinusoidal s-curve
`PR AT`	Return the configured acceleration type

RELATED

A (Acceleration) AJ (Acceleration Jerk) D (Deceleration) DJ (Decel Jerk) DT (Decel Type) VI (Initial Velocity VM (Max Velocity)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP
	0x66	1	0x15	USINT	Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes

Mnemonic	Function	Function Group	Access	Usage
AV	Approximate hMT Velocity	hMT Variable	RO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+

DESCRIPTION

AV reads the approximate axis velocity when hMT is enabled. The granularity of the output is based upon the setting of the **VF** (Velocity Filter).

Range: — **Units:** — **Default:** —

Syntax: PR AV | [BR/CL] <label/address>, AV<math><num>

CODE EXAMPLES

`PR AV` `0`	Print the actual hMT velocity the hMT velocity is zero
`BR Q1,AV>10000`	Conditional branch to Q1 when AV is greater than 10000

RELATED

AS (hMT Mode)
VF (Velocity Filter)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP
	0x6A	1	0x12	DINT	Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes

Mnemonic	Function	Function Group	Access	Usage
BD	Set/Read Serial BAUD Rate	Communications Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Programmable Motion Control LMD and LMD Motion Module (Serial RS-422/485/UART only)

DESCRIPTION

This variable sets the baud rate for serial communications via the RS-422/485 interface. The baud rate is set by indicating the first two digits of the desired rate as shown in the table below.

Mode	Operation
48	4800 bps
96	9600 bps (default)
19	19200 bps
38	38000 bps
11	115200 bps

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 LMD device is reset, it will communicate at the new baud rate.

A delay time between the command requests to the device must be considered to allow it time to interpret a command and respond to the host before sending a subsequent command.

The time between requests is dependent on the command and the corresponding response from the device.

The **BD** command is incompatible with LMD TCP/IP products. If used, an *Error 37: Command not available* will result when queried.

NOTES:

– When placing the product into firmware upgrade mode UG (Upgrade Firmware) the device will automatically set the BAUD to 19200 bps.

– When the baud is changed, it MUST be matched in Motion Control Interface.

Range:	See table above	Units:	—	Default:	96 (9600 bps)

Syntax: BD =<mode>, PR BD

CODE EXAMPLES

`BD=48`	Set serial baud rate to 4800 bits per second
`PR BD`	Read the value of BD to the terminal window

RELATED

CK (Checksum) EM (Echo mode)

UG (Upgrade)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
BE	Set/Read Backlash Enable	Motion Flag	RW	Immediate/Program

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+

DESCRIPTION

The **BE** flag enables the backlash compensation feature.

Backlash is the amount of mechanical variance within a system. For example, the nut on a leadscrew may require several steps to engage the screw thread. During a direction change, several steps would again be required before the actual motion in the opposite direction would begin.

LMD Motion Products are able to compensate for that amount using the **BM** (Backlash Compensation Mode) and **BL** (Backlash Compensation Amount) variables, eliminating any positional errors due to backlash.

State	Meaning
0	Disable backlash compensation (default)
1	Enable backlash compensation

Range: 0/1 **Units:** — **Default:** 0

Syntax: BE=<0/1> | PR BE

CODE EXAMPLES

`BE=1`	Enable backlash compensation
`PR BE`	Return the state of the backlash compensation enable flag

RELATED

BL (Backlash Amount)

BM (Backlash Mode)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x66	1	0x16	BOOL		Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes

Mnemonic	Function	Function Group	Access	Usage
BL	Set/Read Backlash Amount	Motion Variable	RW	Immediate/Program

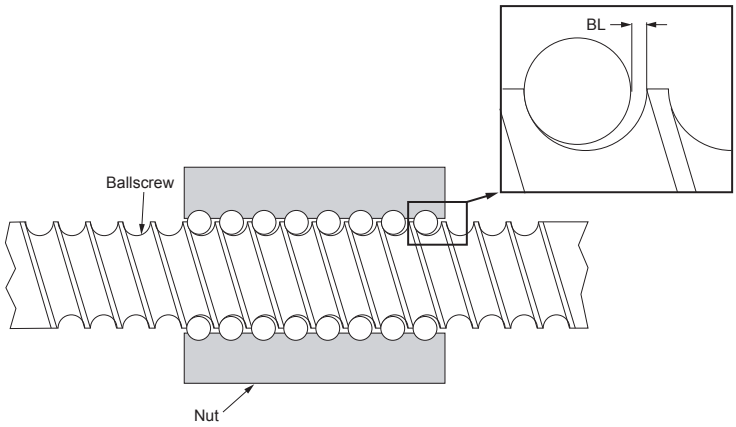
Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+

DESCRIPTION

This variable represents the amount of backlash compensation employed in motor steps, or in encoder counts if encoder functions are enabled (**EE**=1).

The **BL** variable is signed. If no sign precedes the value, it is assumed to be positive. The minus (-) symbol must always be programmed, The sign indicates the direction and is only required when using Backlash Compensation Mode 1 (**BM**=1 - Mechanical Compensation).



Range: ±2147483648 **Units:** steps / counts **Default:** 0

Syntax: BL=<steps> | PR BL

CODE EXAMPLES

`BL=25600`	Set backlash compensation amount to 1/2 revolution @ MS=256 (motor steps)
`BL=2000`	Set backlash compensation amount to 1/2 revolution @ EE=1 (encoder counts)
`PR BL`	Return the amount of backlash compensation to the terminal

RELATED

BE (Backlash Enable)

BM (Backlash Mode)

EE (Encoder Enable)

MS (Microstep Resolution)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x66	1	0x17	DINT		Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes

Mnemonic	Function	Function Group	Access	Usage
BM	Set/Read Backlash Mode	Motion Variable	RW	Program/Immediate
Compatibility: LMD(O) LMD(C) LMD(A) LMM		**Notes**: Firmware 6.001+		

DESCRIPTION

The **BM** (Backlash Mode) variable sets the mode of operation for backlash compensation, either mathematical (mode 0) or mechanical (mode 1). Backlash compensation must be enabled using the **BE** (Backlash Enable) flag in order to function.

Mode 0: Mathematical Compensation

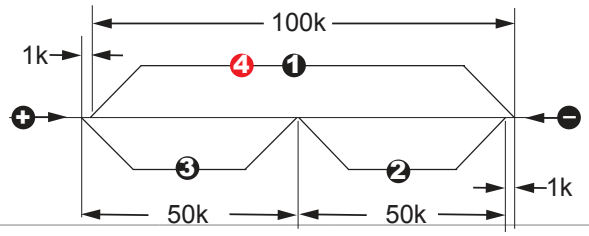
When mathematical backlash compensation has been employed, the value of **BL** (Backlash Amount) is added to each change of direction. On each reversal move, the controller outputs the programmed move plus the backlash units to the driver, taking up the backlash from the change in direction, and completes the move to the correct position.

The figure below illustrates Mode 0 operation using the assumption that backlash is taken up before the first move:

1. Move ❶ is +100k steps
2. Move ❷ is -50k steps. When the motor reverses direction, there are 1000 steps of backlash where no physical motion occurs. When Move ❷ executes on the reversal of direction, the value of **BL** (1000) is added to the value of the motion command: **MA** 50000 + 1000 results in a total motor move distance of 51000 steps, though the load only moves 50000 steps. The Position Counter (**P**) records the total move distance of 51000.
3. Move ❸ is -50k steps. Because the backlash was taken up during Move ❷, Move ❸ is uncompensated.

MCode parameters

```
BE=1        'enable backlash compensation
BL=1000     'set backlash amount to 1k steps
BM=0        'set backlash mode to mathematical
```



4. Because the next move, Move ❹, is a reversal of direction, **BL** is again added to the +100000 steps of Move ❹.

Mode 1: Mechanical Compensation

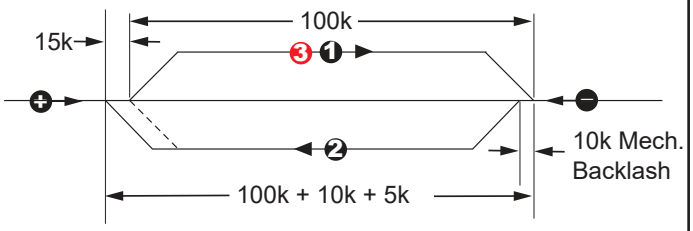
Mechanical backlash compensation always “loads” the axis in the direction of the sign (±) of the **BL**. A move in the direction opposite to that indicated by the sign (±) has the value specified by **BL** added to it. A separate move is then made relative to the sign (±) to take up the backlash amount and “load” the axis. Whenever possible, program more backlash than there actually is.

The figure below illustrates Mode 1 operation using the assumption that backlash is taken up and the axis “loaded” in the plus (+) direction before the first move:

1. Move ❶ is plus (+) 100k steps.
   NOTE: Whenever possible, always enter a larger compensation value than the actual to ensure proper backlash removal and proper axis “loading.”
2. The example in the figure above assumes 10k steps of mechanical backlash, set **BL** (Backlash Amount) to 15000 (or some value greater than 10000).
3. Move ❷ indexes the axis minus (-) 100k steps but due to 10k steps backlash, the (uncompensated) physical movement of the axis would only be 90k steps. Since Move ❷ is opposite the sign of the compensation, 15k sites of compensation is added giving a sum of 115k steps. Because of the physical backlash, the result would be a 5 unit overshoot.

MCode parameters

```
BE=1        'enable backlash compensation
BL=15000    'set backlash amount to 15k steps
BM=1        'set backlash mode to mechanical
```



4. On execution of Move ❸, the axis moves back in the plus (+) direction 15k steps – 10k to take up backlash and 5k to go to the correct position and “load” the axis again.

Range:	0/1	Units:	—	Default:	0
Syntax:	BM=<mode> | PR BM				

CODE EXAMPLES

`BM=1`	Set backlash compensation mode to 1: mechanical compensation
`PR BM`	Return the mode setting for backlash compensation to the terminal

RELATED

BE (Backlash Enable)

BL (Backlash Amount)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x66	1	0x18	BOOL		Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes

Mnemonic	Function	Function Group	Access	Usage
BP	Break Point	Program Instruction	CMD	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

The Break Point Instruction is a debugging tool used to set break points within a program to assist in troubleshooting and optimizing the LMD MCode programs.

The program must execute in either trace or single-step mode for the **BP** instruction to take effect. The program executes for the number of times specified by the count, then goes 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.

While a program is running; typing **BP** without a value will break a program and allow the spacebar to step through the program where it is. As if a BP was set.

To disable the break point, set BP=0.

Range:	Units:	Default:
—	—	—

Syntax: BP <label/address>,<count>

CODE EXAMPLES

`BP X1,3`	Break at label X1 after 3 cycles
`EX P1,2`	Execute program P1 in single-step mode

RELATED

EX (Execute program)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
BR	Branch	Program Instruction	CMD	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

The branch instruction is used to perform a conditional or unconditional branch to a location in an LMD MCode program. It can also be used to perform loops and IF THEN logic within a program. There are two types of branch instructions:

Conditional Branch
There are two parameters to a conditional branch instruction. The first parameter specifies an address or user label where program execution should continue when the condition defined by the second parameter occur.
The second parameter defines the condition, which may include flag states, variable values or logical functions. Only one condition may exist.

Example conditions defining the second parameter include:

- Input logic states: I1=0 (Input 1 is LOW), I2=1 (Input 2 is HIGH)
- Flag logic states: ST=1 (Axis is stalled)
- Variable values (user or factory): V1<=10 (User Variable V1 is less than/equal to 10)

Unconditional Branch
In an unconditional branch, the second parameter is not specified, and then the execution continues at the label or address specified by the first parameter.

Range:	Units:	Default:
—	—	—

Syntax (Conditional): BR <label/address>, [VAR/FLG/IN]<math><condition>

Syntax (Unconditional): BR <label/address>

CODE EXAMPLES

`BR Q1`	Unconditional branch to labeled location Q1
`BR Q1,I1=1`	Conditional branch to labeled location Q1 when input 1 is equal to 1

RELATED

CL (Call Subroutine)
EX (Execute Program)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
BY	Program Busy (executing)	Status Flag	RO	Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

The **BY** flag indicates the status of program execution: (0) program is not executing or (1) program running.

Range:	Units:	Default:
0/1	—	—

Syntax: PR BY

CODE EXAMPLES

`PR BY` Return the state of the busy flag

RELATED

E (End Program) EX (Execute program) PG (Program Mode)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0004
	0x64	1	0x07	BOOL		

Mnemonic	Function	Function Group	Access	Usage
C1	Read/Set Counter 1 (Motor Counts)	Motion Variables	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

This variable contains the 32-bit integer count of the clock pulses generated by the LMD MCode compatible device. Counter 1 supplies the position count for **P** (Position Counter) when the LMD Motion product is operating in open loop mode without an encoder or **EE** (Encoder Enable) is set to zero (0/disabled).

Rollover behavior:
When **C1** reaches its limit in either the plus (+) or minus(-) direction, **C1** rolls over to the limit value of the opposite signed count and counts up or down from there.
Example:

- **C1** = 2147483647, its plus (+) upper limit
- Enter a plus (+) move of 1 motor count
- Issuing **PR C1** returns -2147483648, the minus (-) lower limit.

Absolute Encoder note:
LMD products with a multi-turn absolute encoder read the master encoder position from **C2**, in encoder counts (4000 counts/rev).
The position register, **P**, will read the shaft position from **C1** in motor steps, and all motion commands are issued in motor steps.

On powering or performing a software reset, the LMD will perform a calculation converting the **C2** count value to motor steps, then load **C1** with that value, synchronizing the two counters.

The following program snippet illustrates the declaration of a user variable, **Xr**, to function as a rollover counter. The motion runs until **C1** reaches a predetermined value, then calls a subroutine to increment the rollover counter variable, then zero **C1** before returning to the program.
This snippet may be adapted to duplicate this functionality with the **C2** (Encoder Counter) variable and **P** (Position Counter) by replacing the **C1** references to the appropriate variable.

Global variables:	
`VA Xr=0`	Define user variable Xr (Rollover Counter) and set value to 0
Program contents:	
`PG 1` `LB X1`	Enter program mode @address 1, name program X1
`'***Motion***`	Motion code block
`CL X2, C1>=2000000000` `BR X1`	Call named subroutine X2 when C1 greater than/equal assigned value
`'***Subroutine***`	

`LB X2`
`IC Xr`
`PR Xr`
`PR C1`
`C1=0`
`RT`
	Increment the rollover counter register, reset C1 to zero, return from subroutine X2

`E` `PG`	End, exit program mode

NOTE: Do not modify the counter values during system operation. Changing the value of **C1** during normal operation will desynchronize the **C1/C2** relationship and may cause position errors do to a discrepancy between actual and reported until the counters are resynced via a power cycle or software reset.

Range:	**Units:**	**Default:**
-2147483648 to +2147483647	motorcounts	—

Syntax: C1=<counts> | PR C1 | BR <label/address>, C1=<counts>

CODE EXAMPLES

`C1=10000`	Set the value of counter 1 to 10000
`PR C1`	Read the value of counter 1 to the terminal
`BR Q1, C1=512000`	Conditional branch to named location Q1 when counter 1 = value
`CL X5, C1=512000`	Conditional call to named subroutine X5 when counter 1 = value

RELATED

C2 (Counter 2)
EE (Encoder Enable)
P (Position Counter)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0005 - 0x0006
	0x68	1	0x01	DINT		

Enabling the encoder (EE=1) or modifying the data in Motor Counter C1 or Encoder Counter C2 on LMD models with a multi-turn absolute encoder during operation will desynchronize the relationship between the counters and the Absolute encoder counter, causing a discrepancy between reported and actual shaft position.

⚠CAUTION

UNINTENDED OPERATION

- Do not modify, manually or by program, counters C1 or C2 during operation.
- Do not move outside the range of the counter, either by setting it manually or by rolling over the counter on LMD models with an absolute encoder.

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

Mnemonic	Function	Function Group	Access	Usage
C2	Read/Set Counter 2 (Encoder Counts)	Motion Variables	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes:
LMD Closed loop
LMM with encoder

DESCRIPTION

This variable contains the 32-bit integer value of the encoder counts read by the LMD MCode compatible device. When encoder is enabled (EE=1), C2 provides the position count for P (Position Counter).

Rollover behavior:

When **C2** reaches its limit in either the plus (+) or minus(-) direction rolls over to the limit value of the opposite signed count and counts up or down from there.
Example:

- **C2** = 2147483647, its plus (+) upper limit
- Enter a plus (+) move of 1 encoder count
- Issuing **PR C2** returns -2147483648, the minus (-) lower limit.

Range:	Units:	Default:
-167772160 to +167772160	encoder counts	—

Syntax: C2=<counts> | PR C2 | BR <label/address>, C2=<counts>

CODE EXAMPLES		RELATED
`C2=10000`	Set the value of counter 2 to 10000	C1 (Counter 1)
`PR C2`	Read the value of counter 2 to the terminal	EE (Encoder Enable)
`BR Q1,C2=40000`	Conditional branch to named location Q1 when counter 2 = value	EL (Encoder Lines)
`CL X5,C2=40000`	Conditional call to named subroutine X5 when counter 2 = value	P (Position Counter)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0007 - 0x0008
	0x69	1	0x01	DINT		

Enabling the encoder (EE=1) or modifying the data in Motor Counter C1 or Encoder Counter C2 on LMD models with a multi-turn absolute encoder during operation will desynchronize the relationship between the counters and the Absolute encoder counter, causing a discrepancy between reported and actual shaft position.

⚠CAUTION

UNINTENDED OPERATION

- Do not modify, manually or by program, counters C1 or C2 during operation.
- Do not move outside the range of the counter, either my setting it manually or by rolling over the counter on LMD models with an absolute encoder.

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

Mnemonic	Function	Function Group	Access	Usage
CB	Read/Set Control Bounds (for hMT)	Motion Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

The **CB** (Control Bounds) variable defines the operational tolerance for the closed loop hMTechnology. The four (4) settings are used to tune the control tolerance, optimizing the device for torque, speed, or balanced torque-speed performance.

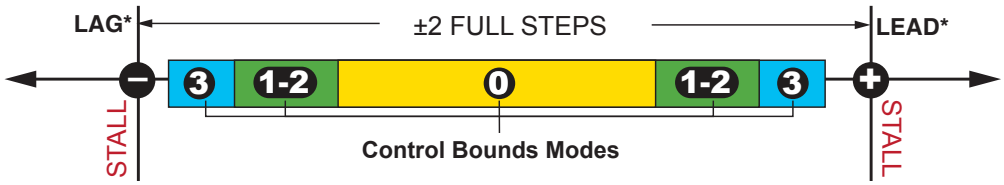
The setting represents a range value in full motor steps. The hMTechnology feature keeps the relationship between the rotor and the stator within the tolerance by the particular mode setting.

For example, **CB**=0 provides the tightest control bounds for optimal torque performance. The hMT algorithm keeps the rotor-stator relationship within 1.1 full steps. **CB** =3 opens the performance gap between the rotor and the stator to 1.7 steps for better speed performance.

CB (Control Bounds) is only applicable when hMTechnology is in fixed or variable current modes (**AS**=1 or **AS**=2).

When hMT torque mode (**AS**=3) is active, control bounds are predefined at 1.1 motor steps (**CB**=0) and may not be adjusted.

Setting	Operation	
0	± 1.1 motor full steps provides optimal torque performance	
1	± 1.3 motor full steps (default)	best overall balanced torque-speed performance
2	± 1.5 motor full steps	
3	± 1.7 motor full steps provides optimal speed performance	



*LAG: rotor is behind stator (accelerating or transient load)
LEAD: rotor is ahead of stator (decelerating or overhauling load)

Range:	0-3	Units:	—	Default:	1

Syntax: CB=<mode> | PR CB

CODE EXAMPLES

`CB=2`	Set the control bounds mode for hMT to 1.5 motor full steps
`PR CB`	Return the control bounds mode to the terminal

RELATED

AS (hMT Mode)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0091
	0x6A	1	0x03	USINT		

Mnemonic	Function	Function Group	Access	Usage
CE	Software reset enable/disable	Configuration flag	R/W	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

This setup flag will configure the device to respond or not respond to a CTRL+C software reset.

Mode	Operation
0	Disabled, LMD device will not respond to a CTRL+C input
1	Enabled (default) CTRL+C entry will assert a software reset, stopping motion and running programs. Unsaved user variables and data will be lost.
2	Is addressable in party mode (PY=1), CTRL+C will respond the same as CE=1 when not in party mode.

Range: 0 – 2 **Units:** — **Default:** 1 (enabled)

Syntax: CE=<mode>, PR CE

CODE EXAMPLES

`CE=0`	Disable response to software reset command CTRL+C
`PR CE`	Return the software reset mode to the terminal

RELATED

DN (Device Name)

PY (Party Mode)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0009
	0x64	1	0x01	BOOL		

Mnemonic	Function	Function Group	Access	Usage
CF	Clear Locked Rotor Error	hMT instruction	CMD	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

The **CF** instruction clears a locked rotor fault and re-enables the output bridge.

A locked rotor is indicated by the **LR** (Locked Rotor) flag, the assertion of an *Error 104: hMT Locked Rotor*, or by a latched state on the Attention Output, if so configured using the **AO** (Attention Output Mask) variable.

A power cycle will also clear a locked rotor.

Range: — **Units:** — **Default:** —

Syntax: CF

CODE EXAMPLES

`CF`	Clear locked rotor condition, re-enable output bridges

RELATED

AS (hMT Mode)

LR (Locked Rotor)

LT (Locked Rotor Timeout)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0093
	0x6A	1	0x04	BOOL		

Mnemonic	Function	Function Group	Access	Usage
CK	Checksum mode select	Configuration Variable	R/W	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

This setup variable configures the device to operate in checksum mode.

In this mode, appending the ASCII character representing the value of the checksum is required following the command string.

To calculate the checksum, using an example motion command: **MR** 51200 (move relative one revolution):

Command	M	R	<space>	5	1	2	0	0		← 7-bits →		
ASCII:	77	82	32	53	49	50	48	48	sum →	439		
Action:	Convert to the sum to binary									1	1011	0111
Action:	One's complement the result									0	0100	1000
Action:	Adding one results in a two's complement									0	0100	1001
Action:	Or result with 128									0	1100	1001
Result:	Checksum (decimal) =									201		
ASCII Table lookup DEC 201 provides check sum character =										É		

Enter **`MR 51200[ALT]+0201`** OR paste **`MR 51200É`** into the active terminal window to execute command via checksum mode

To assist in calculating the checksum, we have provided a Microsoft® Excel spreadsheet which calculates the checksum and displays the checksum character. See the Resource Download portion of this table to download the Checksum Calculator.

Mode	Operation
0	Checksum mode disabled (default)
1	Puts the device into checksum mode. When enabled, all communications with the device require a check-sum to follow the commands. The checksum 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 is acknowledged with an NAK (0x15 – Checksum verification failure) if the checksum is incorrect or an ACK (0x06 - checksum verification successful) when the command correctly processes (no error).
2	Enables checksum mode. However, "NAK" only sent for bad checksum. "ACK" is not echoed if a program is running, NAK is only echoed if an error occurs. In immediate mode, both ACK or NAK characters are echoed.

Range: 0-2	**Units:** —	**Default:** 0 (disabled)

Syntax: CK=<mode>, PR CK

CODE EXAMPLES

`CK=1`	Enable checksum verification in mode 1: ACK and NAK always sent
`PR CK`	Return the selected checksum mode to the terminal

RELATED

BD (BAUD Rate)

RESOURCE DOWNLOAD

Checksum Calculator (*.xlsx)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0009
	0x64	1	0x01	BOOL		

Mnemonic	Function	Function Group	Access	Usage
CL	Call Subroutine	Program Instruction	CMD	Program

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

This instruction is used to invoke a subroutine within a program, allowing the user to segment code and call a subroutine from multiple places rather than repeating code within a program.

There are two parameters to the **CL** (Call Subroutine) instruction.

The first parameter specifies the program address or label of the subroutine to be invoked if the second parameter, the condition, is satisfied. If the second parameter is not used or blank, the subroutine indicated by the first parameter is always invoked.

The second parameter defines the condition. This setting includes variables and flags, as well as logical and input functions, that are to be evaluated. There can only be one condition.

The subroutine must end with an **RT** (Return) instruction. The **RT** instruction will cause program execution to return to the address line following the line invoking a subroutine call.

Range:	—	Units:	—	Default:	—

Syntax (Conditional): (conditional) CL <label/address>, [VAR/FLG/IN]<math><condition>

Syntax (Unconditional): (unconditional) CL <label/address>

CODE EXAMPLES

`CL Q3`	Unconditionally call subroutine at labeled location Q3
`CL Q3,I1=1`	Conditionally call subroutine at labeled location Q3 when input 1 is 1.

RELATED

BR (Branch)

RT (Return from Subroutine)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
CP	Clear Program Memory	Program Instruction	CMD	Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Clears the program space in the non-volitile memory (NVM) as specified by the instruction parameter. Programs are stored and executed directly from NVM. The **CP** instruction will empty program memory only. It will not erase globally declared user variables or flags.

A save command (**S**) must be issued following the invocation of a **CP** (Clear Program).

Issuing an **FD** (Factory Defaults) will also clear program memory space.

CP may be used with a parameter to determine whether or not to leave user variables.

Parameter	Operation
0	Retain user variables
1	Delete user variables

Range:	—	Units:	—	Default:	—

Syntax: CP <label/address> | CP

CODE EXAMPLES

`CP` `S`	Clear all of program memory and save
`CP 0,P1`	Clear program memory occupied by labeled program P1, retain user variables

RELATED

FD (Factory Defaults)

IP (Initialize Parameters)

S (Save)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
CW	Clock Width for Trip Output	I/O Variable	RW	Program/Immediate

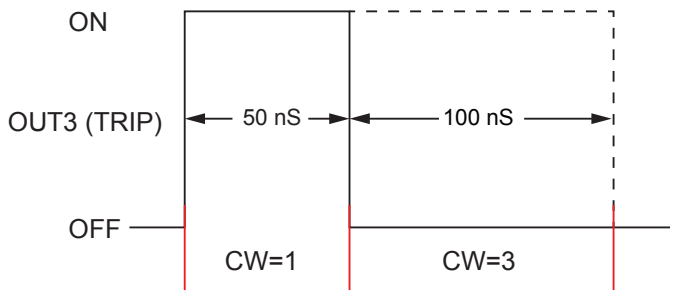
Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION:

CW sets the pulse width duration for the trip output in 50 nanosecond increments. The trip output will be active for the duration specified by the **CW** variable.

Clock width command impact on Trip output pulse width



Range:	0 - 255	Units:	50 nSec	Default:	10 (x 50 nSec)

Syntax: CW=<time>, PR CW

CODE EXAMPLES

`CW=100`	Set Trip output clock width to 5000 nSec (100 * 50 nS)
`PR CW` `100`	Read the value of CW to the terminal Clock width is 100 nSec

RELATED

PC (Position Capture)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x000E
	0x64	1	0x02	USINT		

Mnemonic	Function	Function Group	Access	Usage
D	Set/Read Deceleration	Motion variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Defines the deceleration rate when changing velocity. If the value of **D** is 76800 steps per second^2, the motor decelerates at a rate of 76800 steps per second, every second at the default linear acceleration type.

The primary factor determining the range and units applied to the deceleration profile is the logic state of the **EE** (Encoder Enable) flag. When disabled (**EE**=0) deceleration is measured in steps/sec^2. When enabled (**EE**=1) the value represents encoder counts/sec^2. When **EE** is changed, **A**, **D**, **VI** and **VM** are recalculated.

The secondary factors impacting deceleration is the configuration of **DT** (Deceleration Type) and **DJ** (Deceleration Jerk). **DT** adds triangle and sinusoidal S-curve capability to the default linear deceleration type. The **DJ** variable allows the user to set a constant value to compensate for load oscillations.

Range:		Units:		Default:	
Clock Mode	66 to 1100 X 10^6	**Clock Mode**	steps/sec^2	**Clock Mode**	1000000
Encoder	85937496	**Encoder**	counts/sec^2	**Encoder**	78125

Syntax: D=<integer> | PR D

CODE EXAMPLES

`D=2000`	Set deceleration to 2000 steps/sec^2
`PR D` `2000`	Print deceleration to the terminal screen deceleration is set to 2000 steps/sec^2

RELATED

A (Acceleration) DJ (Decel Jerk)
DT (Decel Type) EE (Encoder Enable)
VI (Initial Velocity) VM (Max Velocity)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0018 - 0x0019
	0x66	1	0x02	UDINT		

Mnemonic	Function	Function Group	Access	Usage
D1 - D4	Read/Set Digital Input Filter	I/O Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

Variable defines the time in milliseconds that the input is allowed to settle following a state transition, a factor common to mechanical switches.

Filtering is applied separately to each input.

Range:	0 (no filtering) - 255	**Units:**	milliseconds	**Default:**	0

Syntax: D[1-4]=<time> | PR D[1-4]

CODE EXAMPLES

```
D2=50          Set the digital filter of input 2 to 50 msec
PR D2          Read the value of D2 to the terminal
   50            Filtering for input 3 is 50 msec
```

RELATED

D5 (Analog Filter)
I[1-4] (Read Input 1-4)
IS (Input Setup)

NETWORK PROTOCOL EQUIVALENTS

	Class	Instance	Attribute	Data Type		
Ethernet/IP	0x67	1	0x02 – 0x05	USINT	**Modbus/TCP**	0x000F 0x0010 0x0011 0x0012

Mnemonic	Function	Function Group	Access	Usage
D5	Set/Read Analog Input Filter	I/O Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

The Analog Filter is a continuously updating process. A running average (Aa) is performed by computing the equation shown below, where **D5** (Analog Filter) is a value between 0 and 1000 and **I5** (Read Analog Input) is the current reading between 0 and 4095.

Aa = ((Aa * (**D5** - 1)) + **I5**) / **D5**

Range:	0 — 1000	**Units:**	milliseconds	**Default:**	0

Syntax: D5=<counts> | PR D5

CODE EXAMPLES

```
D5=50          Set the analog filter to 50 counts
PR D5          Read the value of D5 to the terminal
   50            the analog filter is set to 50
```

RELATED

D[1-4] (Input Filter)
I[1-4] (Read Input)
IS (Input Setup)

NETWORK PROTOCOL EQUIVALENTS

	Class	Instance	Attribute	Data Type		
Ethernet/IP	0x67	1	0x06	UINT	**Modbus/TCP**	0x0013

Mnemonic	Function	Function Group	Access	Usage
DB	Encoder Deadband	Motion Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: LMM operability requires connected and configured encoder

DESCRIPTION

This variable defines the plus (+) and minus (-) length of the encoder dead-band in encoder counts.

A move completes when motion stops within the range defined by the DB (Encoder Deadband) parameter. If PM (Position Maintenance) is enabled, (PM=1), the position corrects when pushed outside of DB value once in position.

Encoder functions must be enabled (EE=1) for the DB to take effect.

Range:	Units:	Default:
0 — 65000	counts	1

Syntax: DB=<counts>, PR DB

CODE EXAMPLES		RELATED		
`DB=10`	Set the encoder deadband to ±10 counts	C2 (Counter 2)	EE (Encoder Enable)	PM (Position Maint.)
`PR DB`	Read the value of DB to the terminal	SM (Stall Mode)	ST (Stall Flag)	SF (Stall Factor)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x001A
	0x69	1	0x02	UINT		

Mnemonic	Function	Function Group	Access	Usage
DC	Decrement Variable	Instruction	WO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Decrements the specified factory or user variable by one (1).

Attempting to decrement an unspecified or read-only variable results in an *Error 25: variable is read-only*.

Attempting to decrement a mode select or configuration variable, for example MS (Microstep Resolution) results in an *Error 26: attempting to increment or decrement an illegal variable*.

Range:	Units:	Default:
—	—	—

Syntax: DC <var>

CODE EXAMPLES		RELATED	
`DC V1`	Decrement user variable V1	IC (Increment Variable)	VA {Create User Var)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x001B
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
DE	Drive Enable/disable	Configuration Flag	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Enables (1- default) or disables (0) the drive output bridges.

Issuing a motion command, for example, MA (Move Absolute), MR (Move Relative), SL (Slew), or any homing command while the drive is disabled (DE=0), results in an *Error 94: attempting motion while the drive is disabled*.

Mode	Operation
0	Bridge outputs disabled. Attempting motion results in an *Error 94*.
1	Bridge outputs enabled (default).

Range: 0/1 **Units:** — **Default:** 1 - Enabled

Syntax: DE=<mode>

CODE EXAMPLES

`DE=0` Set drive enabled state to 0 (disabled)

RELATED

—

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x64	1	0x03	BOOL		0x001C

Mnemonic	Function	Function Group	Access	Usage
DG	Enable/disable global response in party mode	Configuration Flag	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Serial RS-422/485 models and LMM only

DESCRIPTION

Enables or disables device response to global commands made while in party mode (PY=1). In the default state (DG=1) the device executes global commands without sending back a response. By setting DG=0, that device responds global commands.

The asterisk character (*) is the drive name for global commands. Commands preceded by this character will be recognized by every MCode compatible device in the system that has DG = 0.

Mode	Operation
0	Enable global response to commands (commands echo back to terminal)
1	Disable global response to commands (default — command does not echo back to terminal)

Note: DG only impacts operation when the device is in serial party mode (PY=1).

Range: 0/1 **Units:** — **Default:** 1

Syntax: DG=<mode> | <dn>DG=0

CODE EXAMPLES

`DG=0` Enable global response to commands

`aDG=0` Enable global response to commands on named device "a"

RELATED

DN (Device name)

PY (Party Mode)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	—	—	—	—		—

Mnemonic	Function	Function Group	Access	Usage
DJ	Read/Set Deceleration Jerk	Motion Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+

DESCRIPTION

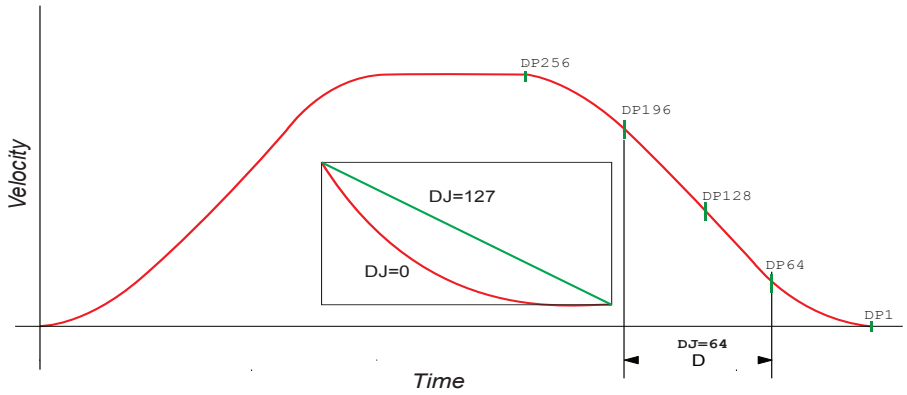
Deceleration Jerk is the rate of change of Deceleration, or, the derivative of Deceleration with respect to time.

The Deceleration jerk variable only impacts the motion profile when an S-curve Deceleration type (**DT**=2 or **DT**=3) is selected.

The jerk value may be adjusted to any integer value between 0 and 127 to compensate for load oscillations. The motion logic in the LMD product samples 256 data points during the deceleration ramp.

The value applied to **DJ** represents the number of data points on either side of the center of the deceleration table, at which the deceleration is at a constant, linear deceleration at the value defined by **D** (Deceleration).

For example: With **DJ**=64, the deceleration ramp will be constant for 128 samples, or 64 samples on either side of the ramp center. This graphic shows an example of Deceleration Jerk.



Range: 0 to 127 | **Units:** — | **Default:** 0

Syntax: DJ=<value> | PR DJ

CODE EXAMPLES

`DJ=32`	Set deceleration jerk to 32
`PR DJ` `32`	Read the value of DJ to the terminal window decel jerk is set to 32

RELATED

A (Acceleration)
AJ (Accel Jerk)
AT (Accel Type)
D (Deceleration)
DT (Decel Type)
VI (Initial Velocity)
VM (Max Velocity)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x66	1	0x19	USINT		Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes

Mnemonic	Function	Function Group	Access	Usage
DN	Device name for party mode	Communication variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: Serial RS-422/485 models and LMM only

DESCRIPTION

Configures 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. Attempting to assign a device name without enclosing it within quotation marks returns an *Error 21: Tried to SET to an incorrect value*.

The name is case sensitive.

Resetting the device to the default character (!) requires an **FD** (Factory Default Reset).
The device name will be lost following a factory reset.

Range:	Units:	Default:
a-z, A-Z, 0-9	ASCII	!

Syntax: DN="<char>" | PR DN

CODE EXAMPLES

`DN="a"`	Set the device name to the character "a"
`PR DN`	Return the device name to the terminal window

RELATED

PY (Party Mode)

NETWORK PROTOCOL EQUIVALENTS

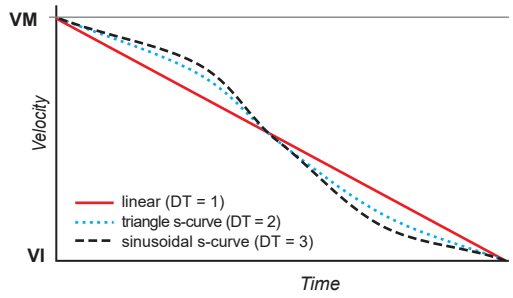
Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP
	—	—	—	—	—

Mnemonic	Function	Function Group	Access	Usage
DT	Read/Set Deceleration Type	Motion Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: Firmware 6.001+

DESCRIPTION

Defines the type of deceleration profile used when a move is executed. There are three (3) deceleration types available for LMD products: Linear (constant), triangle s-curve, and sinusoidal s-curve.



Type	Accel Ramp	Description
1	Linear (default)	Constant smooth (linear) deceleration from initial to max velocity.
2	Triangle	Triangle s-curve profile.
3	Sinusoidal	The sinusoidal s-curve profile is very similar to the triangle s-curve. The main difference is that it has less jerk when starting or stopping.

Range:	Units:	Default:
1- 3	—	1 - Linear

Syntax: DT=<type> | PR DT

CODE EXAMPLES

`DT=3`	Set the deceleration type to sinusoidal s-curve
`PR DT`	Return the configured deceleration type

RELATED: A (Acceleration), AJ (Acceleration Jerk), D (Deceleration), DJ (Decel Jerk), DT (Decel Type), VI (Initial Velocity), VM (Max Velocity)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP
	0x66	1	0x1A	USINT	Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes

Mnemonic	Function	Function Group	Access	Usage
E	End program	Program Instruction	CMD	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

The operation of the **E** (End Program) instruction differs between immediate and program mode.

Program mode**:**
In program mode, the **E** instruction is used to designate the end of a program.

Immediate mode**:**
An **E** issued while in immediate mode ends the currently executing program. If a move is in progress, the program ends after motion completes.

Range: —

Units: —

Default: —

Syntax: E

CODE EXAMPLES

`E` End program

RELATED

EX (Execute Program) PG (Program Mode)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
EE	Encoder Enable/Disable	Configuration flag	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Enables or disables encoder mode. Once placed in encoder mode, all motion-related variables and commands register in encoder counts. The value of **P** (Position Counter) will update from **C2** (Encoder Counter).

Encoder functions such as stall detection and position maintenance require that encoder functions be enabled (**EE**=1).

Mode	Operation
0	Disable (default) encoder functions, motion registers in motor step counts.
1	Enable encoder functions, motion registers in encoder counts. Functions such as stall detection and position maintenance are available.

When **EE** is changed, **A**, **D**, **VI** and **VM** are recalculated.

Range: 0/1

Units: —

Default: 0 (disabled)

Syntax: EE=<mode> | PR EE

CODE EXAMPLES

`EE=1` Enable encoder functions

`PR EE` Return the status of encoder functions

RELATED

C2 (Counter 2) DB (Deadband) EL (Encoder Lines) FM (Filter Motion) PM (Position Maintenance) SF (Stall Factor) SM (Stall Mode) ST ((Stall Flag)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x001E
	0x69	1	0x03	BOOL		

Enabling encoder functions on LMD models with a multi-turn absolute encoder during operation will delete absolute encoder position information and lead to unintended motor shaft positions.

⚠CAUTION

DAMAGE TO COMPONENTS

- Do not enable encoder functions (EE=1) when using the absolute encoder for position tracking.

Failure to follow these instructions can result in equipment damage.

Mnemonic	Function	Function Group	Access	Usage
EF	Error Flag	Status Flag	RO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

EF Indicates whether or not an error condition exists. It clears automatically when a new program executes. The only way to manually clear **EF** is to print the value of the **ER** variable (**PR ER**) or set **ER** to 0 (**ER**=0).

For an external indication of the **EF** status, the **AO** (Attention Output Mask) may be set to one (**AO**=1). The **EF** state displays on the output point configured as the attention output or on LED 2 on LMD products.

Value	Operation
0	No error exists (default).
1	Error condition exists

The instruction **OE** (On Error) 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.

Range: 0/1 **Units:** — **Default:** —

Syntax: PR EF

CODE EXAMPLES

`PR EF` Read the value of the error flag to the terminal

RELATED

AO (Attention Output)
ER (Error)
OE (On Error)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x65	1	0x02	BOOL	0x001F	

Mnemonic	Function	Function Group	Access	Usage
EL	Encoder Lines	Configuration Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

The LMD Motion Module features quadrature encoder inputs (A/B/Index).
EL (Encoder Lines) sets the line count for the connected encoder and is used as the scaling factor for calculating encoder moves, **C2** (Counter 2) reads 4 x **EL** or 4 counts per line.

MS (Microstep Resolution) is relative to **EL**.
To calculate the minimum value for **MS**, use the following equation:

MS_{MIN} = (EL x 8) ÷ Full Steps per Revolution

MS variables need to be scaled to the equivalent ratio, as shown in the **MS** Command Summary. The settings for a particular velocity profile can change dramatically based on the setting of **MS**.

For this example: **EL** = 1000 and **FS** = 200 (1.8° stepper motor)

MS_{MIN} = (1000 x 8) ÷ 200
= 8000 ÷ 200
= 40 (**MS** scaling needs to be rounded up to the next available **MS** value. For this example, the **MS** value will be 50)

All shown steps per revolution values assume the 1.8° motor standard with LMD products. If using a custom integrated product, or an LMD Motion Module and motor with a step angle other than 1.8° refer to the **SA** (Step Angle) command.

Range: 1 to 2000 **Units:** lines **Default:** 1000

Syntax: EL=<lines>, PR EL

CODE EXAMPLES

Set encoder lines:

`EL=500` Set the encoder lines to match a 500 line encoder (2000 counts/rev)

Display encoder lines setting:

`PR EL` Return the encoder line count to the terminal window

RELATED

C2 (Counter 2)
EE (Encoder Enable)
MS (Microstep Resolution)
SA (Step Angle)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x6A	1	0x05	UINT	Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes	

Mnemonic	Function	Function Group	Access	Usage
EM	Read/set Echo Mode	Configuration Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Sets the echo configuration of the communications channel.

Mode	Operation
0	Echo all entered commands and data back to the terminal. Carriage return/line feed indicates that the command accepted (full duplex) (default) by the display of the prompt character ">."
1	Do not echo entered commands and data back to the terminal, only return the cursor. CR/LF indicates command accepted by the display of a blinking cursor. Printed values display to the terminal, i.e. PR EM returns "1."
2	Does not return prompt, only echoes data requested by PRINT (PR) and LIST (L) commands.*
3	Command and data echo stored in the print queue, returns to the terminal upon termination of the command string.

Range: 0 — 3 **Units:** — **Default:** 0 (echo all)

Syntax: EM=<mode>, PR EM

CODE EXAMPLES

```
EM=1      Set the echo mode to 1, do not echo commands and data except for a print.
PR EM     Return the echo mode to the terminal window
  1         Echo is set to mode 1
```

RELATED

—

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
ER	Error Register	Variable	R/C	Program/Immediate
Compatibility: LMD(O) LMD(C) LMD(A) LMM		**Notes**: List of error codes will vary between products		

DESCRIPTION

Holding register for the most recent error that has occurred. The **ER** variable must set to zero (**ER**=0) to clear the error code(s). Setting **ER** to 0 will also reset the **EF** (Error Flag).

An error condition is indicated by question mark character (?) in place of the prompt (>).

A command, **OE** (On Error Handler) is used to execute a subroutine when an error condition occurs. While **OE** activates on any error, subroutines may be executed for specific error codes using **BR** (Branch) and **CL** (Call Subroutine) instructions.

While many error codes are common across the product family, each particular device has error codes associated with it specifically. Refer to the following sections for a list of error codes for each product:

"hMTechnology Specific" on page 31

"LMD Error Codes" on page 180

"LMD Motion Module Error Codes" on page 182

Range: —	Units: —	Default: —

Syntax: ER=0 | PR ER | BR <label/address>, ER=<code> | CL <label/address>, ER=<code>

CODE EXAMPLES

Set to a value:

`ER=0`	Clear stored error code

Display error:

`PR ER`	Return last error to the terminal window
`PR ER,10`	Return up to 9 errors encountered on the device to the terminal window. If error list is cleared (ER=0), no errors will be displayed

Program Flow:

`BR Q1,ER=86`	Branch to labeled location Q1 on error code 86 (motor stall)
`CL Z2,ER=104`	Call labeled subroutine Z2 on error code 104 (hMT locked rotor)

RELATED

AO (Attention Output)
BR (Branch)
CL (Call Subroutine)
EF (Error Flag)
OE (On Error)
RT (Return)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0021
	0x65	1	0x03	UNIT		

Mnemonic	Function	Function Group	Access	Usage
ES	Set Escape Mode	Configuration Variable	R/W	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Sets the mode of escaping a program or motion event, either using the [ESC] key or by keying in [CTRL+E]. Modes 2 and 3 add an addressability function to the escape for operation in **PY** (Party Mode).

Mode	Operation
0	Escape triggers on [CTRL + E] keypress
1	Escape triggers on {ESC] keypress (default)
2	Addressable escape set to trigger on <device-name>[CTRL + E] keypress
3	Addressable escape set to trigger on <device-name>[ESC] keypress

Range: 0 — 3 **Units:** — **Default:** 1

Syntax: ES=<mode>, PR ES

CODE EXAMPLES

Set escape mode:

`ES=0` Set escape to trigger on [CTRL + E] keypress

Display mode setting:

`PR ES` Return escape mode setting

RELATED

—

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
EX	Execute Program	Program Instruction	CMD	Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Executes a specified program label or address at a selected mode of execution. If the mode is unspecified or 0, the program executes in normal mode.

Modes 1 and 2 aid in application development and troubleshooting by adding trace and single-step modes.

A custom factory label, **SU** (Start Up) is provided to execute a program so named on power cycle/software reset [CTRL + C].

There are three modes of program execution.

NOTE: Attempting to execute an undefined label will return an *Error 30: Unknown Label or User Variable*.

Mode	Operation
0	Normal execution
1	The program executes continuously until the program E (End), but the instructions are “traced” to the communications port so the user can see the instructions as they process
2	The user can step through the program using the space bar to process each line of the program. The program can be resumed at normal speed in this mode by pressing the enter key

Range: <label/address>, 0 — 2 **Units:** — **Default:** —

Syntax: EX <label/address>,<mode>

CODE EXAMPLES

`EX G1` Execute program at named location G1 normally

`EX G1,2` Execute program at named location G1 in single-step mode

RELATED

SU (Start Up)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes
	0x64	1	0x06	STRING		

Mnemonic	Function	Function Group	Access	Usage
F1 — F8	Floating Point Registers	Mathematics Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+

DESCRIPTION

Double precision 64-bit floating point registers are used to perform calculations requiring a floating decimal point. For use with advanced math and trigonometric operators.

When transferred to a user variable or an integer register **R1 – R4** (User Registers), the fraction portion of the floating point number is discarded.

When a motion command is used with a floating point register, for example **SL=F2** or **MA F5**, the axis will move at the rate or to the position represented by the register value, rounded down to the nearest integer.

The display format for the data contained in floating point registers derives from the **PF** (Print Format) command.

NOTE: Float values sent to non-floating data types will return a whole value.

Range:	**max +:**	1.7976931348623158^{308}	**Units:** —	**Default:** 0
	min +:	$4.9406564584124655^{-324}$		

Syntax: F<num>=<fpvalue>, F<num><math/trig><var/reg>

CODE EXAMPLES

Set to a value:

`F2=300.256`	Set F2 to a value of 300.256
`SL = F2`	Returns a velocity of 300

Math and trig function:

`F2=CS R3`	Set F2 to the cosine value of register R3
`F1=R2*F2`	Multiply R2 by F2

RELATED

R<1-4> (User Registers)

PF (Print Format)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
FC	Set capture input filtering	I/O Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Not applicable to LMDxx42

DESCRIPTION

Sets the digital filtering to be applied to Input 1 when configured as a Capture Input (type = 12).

NOTE: The **FC** command is not available on LMD NEMA 17 Motion Control and Ethernet TCP/IP product.

Mode	Min Pulse	Cutoff Frequency
0 (default)	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

Range: 0 — 9 **Units:** — **Default:** 0

Syntax: FC=<mode>, PR FC

CODE EXAMPLES

`FC=2`	Set capture input to filter signals with a pulse width <150 nS, or of frequency greater than 3.3 MHz
`PR FC`	Return the filter setting for the capture input

RELATED

IS (Input Setup)
TC (Trip Capture)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0024
	0x67	1	0x07	USINT		

Mnemonic	Function	Function Group	Access	Usage
FD	Restore Factory Default settings	Instruction	CMD	Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

The **FD** (Factory Defaults) command resets the device to MCode factory default state.

Entering **FD** followed by a carriage return (ENTER) will result in the loss of all saved data, which includes: programs, user variables, and stored parameter values.

Non-volitile Memory (NVM) values will be retained, which includes:. **PN** (Part Number), **SN** (Serial Number), **PW** (PWM Mask) and **FS** (Index Offset).

NOTE: Once the **FD** command has been entered, there is no warning message to indicate that all programming will be cleared from the device. The programming can be uploaded using the LSS prior to performing the **FD** command to prevent program loss if needed, unless programming is locked.

Range: — **Units:** — **Default:** —

Syntax: FD

CODE EXAMPLES

`FD`	Reset the device to factory default state

RELATED

CP (Clear Program)
IP (Initialize Parameters)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
FL	Following Mode Enable	Configuration Flag	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+

DESCRIPTION

When in an enabled state (**FL**=1), the axis follows the signals on the input pins 3 and 4 at a 1:1 ratio.

Prerequisite:
Configure **IN3** and **IN4** as clock inputs: Step/Direction, ENC A/ENC-B or Step Up/Step Down is required.

Mode	Operation
0	Normal operation, axis motion controlled program or immediate command
1	Axis motion follows inputs 3 and 4 at a 1:1 ratio

NOTE: The **FL** command is not applicable to the encoder inputs on the LMD Motion Module. These inputs are strictly for encoder mode of operation.

Range: 0/1 **Units:** — **Default:** 0 (disabled)

Syntax: FL=<0/1>, PR FL

CODE EXAMPLES

Prerequisite:

```
IS=3,13,1
IS=4,13,1
```
Configure IN3 & IN4 to step direction

Operation:

```
FL=1
```
Enable following mode

Return status:

```
PR FL
```
Return the following mode setting

RELATED

FM (Filter Motion)
I<3-4> (Inputs 3-4)
IS (Input Setup)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x64	1	0x10	BOOL		Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes

Mnemonic	Function	Function Group	Access	Usage
FM	Filter Motion Inputs	I/O Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+

DESCRIPTION

Sets the digital filtering applied to Inputs 3 and 4 when configured as clock inputs.

Prerequisite:
Configure **IN3** and **IN4** as clock inputs: Step/Direction, ENCA/ENCB or Step Up/Step Down is required.

Mode	Min Pulse	Cutoff Frequency
0	50 nS	10 MHz
1	150 nS	3.3 MHz
2 (default)	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

Range: 0 — 9 **Units:** — **Default:** 2

Syntax: FM=<mode>, PR FM

CODE EXAMPLES

`IS=3,13,1` `IS=4,13,1`	Configure IN3 & IN4 to step direction
`FM=2`	Set the motion inputs to filter signals with a pulse width <150 nS, or of frequency greater than 3.3 MHz
`PR FM` `2`	Return the motion filter setting Motion input filter is set to 200 nS

RELATED

FL (Following Mode Enable)
I<3-4> (Inputs 3-4)
IS (Input Setup)

NETWORK PROTOCOL EQUIVALENTS

	Class	Instance	Attribute	Data Type		
Ethernet/IP	0x67	1	0x08	USINT	**Modbus/TCP**	—

Mnemonic	Function	Function Group	Access	Usage
FS	Index Offset Setting	Configuration Variable	RW	Program/Immediate
Compatibility: LMD(O) LMD(C) LMD(A) LMM		**Notes**: Firmware 6.001+ if using an LMM a connected and configured encoder is required.		

DESCRIPTION

FS (Index Offset) sets the reference position for **HF** (Home to Index Offset) operation. It represents the offset between the encoder index mark and the manually set shaft flat position.

FS is configured using a utility included in the Motion Control Interface application. To configure, select **View > Set Shaft Flat Position** and follow the instructions on the configuration dialogs.

For Ethernet TCP/IP models, the Application protocol must be set to MODBUS using the Ethernet TCP/IP Configuration Utility.

To manually calculate the value of **FS** (no load on shaft):

1. Perform a Home to Index operation. For example HI 1 will home the axis to the encoder index mark. To verify the index, enter **PR I6** in the terminal. A returned value of "1" indicates the index mark is aligned.
2. Zero the encoder counter by entering **C2**=0.
3. Disable the driver by entering **DE**=0 to allow free rotation of the shaft.
4. Manually move the motor shaft to the desired position.
5. Read the value of **C2** by entering **PR C2**.
6. Calculate **FS** = **C2***12.8 and enter **FS**=<result>
7. Re-enable the driver (**DE**=1)

Make a positional move, **HF** <mode> will home the axis to the Index offset position.

NOTE: A closed loop LMD or LMD Motion Module with a connected and configured encoder is required.

Range:	±25600	**Units:**	Counts	**Default:**	0
Syntax:	FS=<counts>, PR FS				

CODE EXAMPLES		RELATED
`FS=10246`	Set the offset to 10246 counts	HF (Home to Offset)
`PR FS`	Read the value of the shaft flat offset	I6 (Index Mark)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP
	0x68	1	0x07	DINT	Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes

Mnemonic	Function	Function Group	Access	Usage
FT	Reserved for Factory	Reserved	—	—
Compatibility: —		**Notes**: —		

DESCRIPTION

FT is reserved for factory use. Attempting to use **FT** as a user variable or label will return an *Error 24: Illegal data entered*.

Range:	—	**Units:**	—	**Default:**	—
Syntax:	—				

CODE EXAMPLES	RELATED
—	—

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP
	—	—	—	—	—

Mnemonic	Function	Function Group	Access	Usage
H	Hold program execution	Program Instruction	CMD	Program

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

The hold instruction is used in a program to suspend program execution. There are two ways to use the hold instruction:

H, when not followed by the time parameter suspends program execution until the motion completes. Used without the time parameter, a Hold should always follow a programmed motion instruction such as **MA** (Move Absolute) or **MR** (Move Relative). **H** should also follow the homing instructions: **HI** (Home to Index), **HM** (Home to Home Switch) or **HF** (Home to Offset)

Adding a time parameter to the hold instruction suspends program execution for the specified number of milliseconds.

Range: 1 - 65000 **Units:** milliseconds **Default:** —

Syntax: H, H <time>

CODE EXAMPLES

`MA 512000` `H`	Execute an absolute move, suspend program execution until motion completes
`H 20000`	Suspend program execution for 20 seconds

RELATED

E (End Program)
EX (Execute Program)
HI (Home to Index)
HF(Home to Shaft Flat)
HM (Home to Home Switch)
MA (Move Absolute)
MR (Move Relative)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
HC	Motor holding current	Motion variable	R/W	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Defines the motor holding current as 0 (OFF), or as a percentage value from 1 to 100%. The transition from **RC** (Run Current) to **HC** (Hold Current) is impacted by two other commands: **HT** (Hold Current Delay) and **MT** (Motor Settling Delay Time). These two variables are additive, with the sum being the total time to transition from the **RC** (Run Current) level to the specified stand-still current.

NOTES:

– For LMD products, the current is only given in a percentage range as the driver is already sized and tuned to the integrated motor.

– The LMD Motion Module is a 1.5A RMS standalone integrated driver/controller. The actual drive output current is derived by multiplying 1.5A times the **HC** percentage.
Example:
HC=5
1.5A * 0.05 = 0.075A
Holding current level is 0.075A.

Range: 0 (disabled), 1 to 100 **Units:** Percent (%) **Default:** 5%

Syntax: HC=<percent>, PR HC

CODE EXAMPLES

`HC=0`	Disable holding current, motor is at set RC (Run Current) at all times
`PR HC`	Read the value of the holding current

RELATED

HT (Hold Current Delay time)
MT (Motor Settling Delay Time
RC (Run Current)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0029
	0x66	1	0x03	USINT		

Mnemonic	Function	Function Group	Access	Usage
HF	Home to Index Offset	Motion instruction	WO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+, Encoder

DESCRIPTION

This instruction moves the axis to an offset position of the encoder index mark position specified by **FS** (Offset Setting).

When **HF** executes, the axis moves in specified direction at **VM** (Maximum Velocity) until it reaches the preset position. It then creeps away from the home position in the direction specified at **VI** (Initial Velocity). Motion ceases as soon as the shaft flat position clears.

Type	Slew (VM) direction	Creep (VI) direction
1	(-) minus	(+) plus
2	(-) minus	(-) minus
3	(+) plus	(-) minus
4	(+) plus	(+) plus

"Homing Types" on page 69 diagrams in detail the four combinations for this command, as well as for the related **HI** (Home to Index Mark) and **HM** (Home to Home Switch) instructions.

Range: 1 to 4 **Units:** — **Default:** —

Syntax: HF <type>

CODE EXAMPLES

`HF 3` Seek index offset position in the (+) direction, creep off position in (-) minus direction

RELATED

EE (Encoder Enable) FS (Offset Setting)
HI (Home to Index) HM (Home to Home Switch
VI (Initial Velocity) VM (Maximum Velocity)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x68	1	0x08	USINT		Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes

Mnemonic	Function	Function Group	Access	Usage
HI	Home to Index Mark	Motion instruction	WO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Encoder required

DESCRIPTION

This instruction homes the axis to the encoder index mark.

When **HI** executes, the axis moves in specified direction at **VM** (Maximum Velocity) until it reaches the encoder index. It then creeps away from the index position in the direction specified at **VI** (Initial Velocity). Motion ceases as soon as the index mark clears.

Type	Slew (VM) direction	Creep (VI) direction
1	(-) minus	(+) plus
2	(-) minus	(-) minus
3	(+) plus	(-) minus
4	(+) plus	(+) plus

"Homing Types" on page 69 diagrams in detail the four combinations for this command, as well as for the related **HF** (Home to Index Offset) and **HM** (Home to Home Switch) instructions.

Range: 1 to 4 **Units:** — **Default:** —

Syntax: HI <type>

CODE EXAMPLES

`HI 1` Seek encoder index in the (-) minus direction, creep off in (+) plus direction

RELATED

HF (Home to Offset) EE (Encoder Enable) FS (Offset Setting)
HM (Home to Home) VI (Initial Velocity) VM (Maximum Velocity)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x002A
	0x69	1	0x04	USINT		

Mnemonic	Function	Function Group	Access	Usage
HM	Home to Home Switch	Motion instruction	WO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

This instruction homes the axis to Home Switch.

When **HM** executes, the axis moves in specified direction at **VM** (Maximum Velocity) until it reaches the home switch. It then creeps away from the switch in the direction specified at **VI** (Initial Velocity). Motion ceases as soon as the switch deactivates.

Type	Slew (VM) direction	Creep (VI) direction
1	(-) minus	(+) plus
2	(-) minus	(-) minus
3	(+) plus	(-) minus
4	(+) plus	(+) plus

To use **HM** (Home to Home Switch), a switch connected to an input defined as home using the **IS** (Input Setup) command:
IS=<input #>, 1,<active>
For example: Is=2,1,0 (configures Input 2 as a homing input, active when low).

"Homing Types" on page 69 diagrams in detail the four combinations for this command, as well as for the related **HF** (Home to Index Offset) and **HI** (Home to Index) instructions.

NOTE: **HM** is the only homing function available without an encoder.

Range: 1 to 4 **Units:** — **Default:** —

Syntax: HM <type>

CODE EXAMPLES

`HM 1` Seek home switch in the (-) minus direction, creep off in (+) plus direction

RELATED

HF (Home to Offset) EE (Encoder Enable) FS (Offset Setting)
HI (Home to Index) IS (Input Setup)
LM (Limit Method) VI (Initial Velocity) VM (Maximum Velocity)

NETWORK PROTOCOL EQUIVALENTS

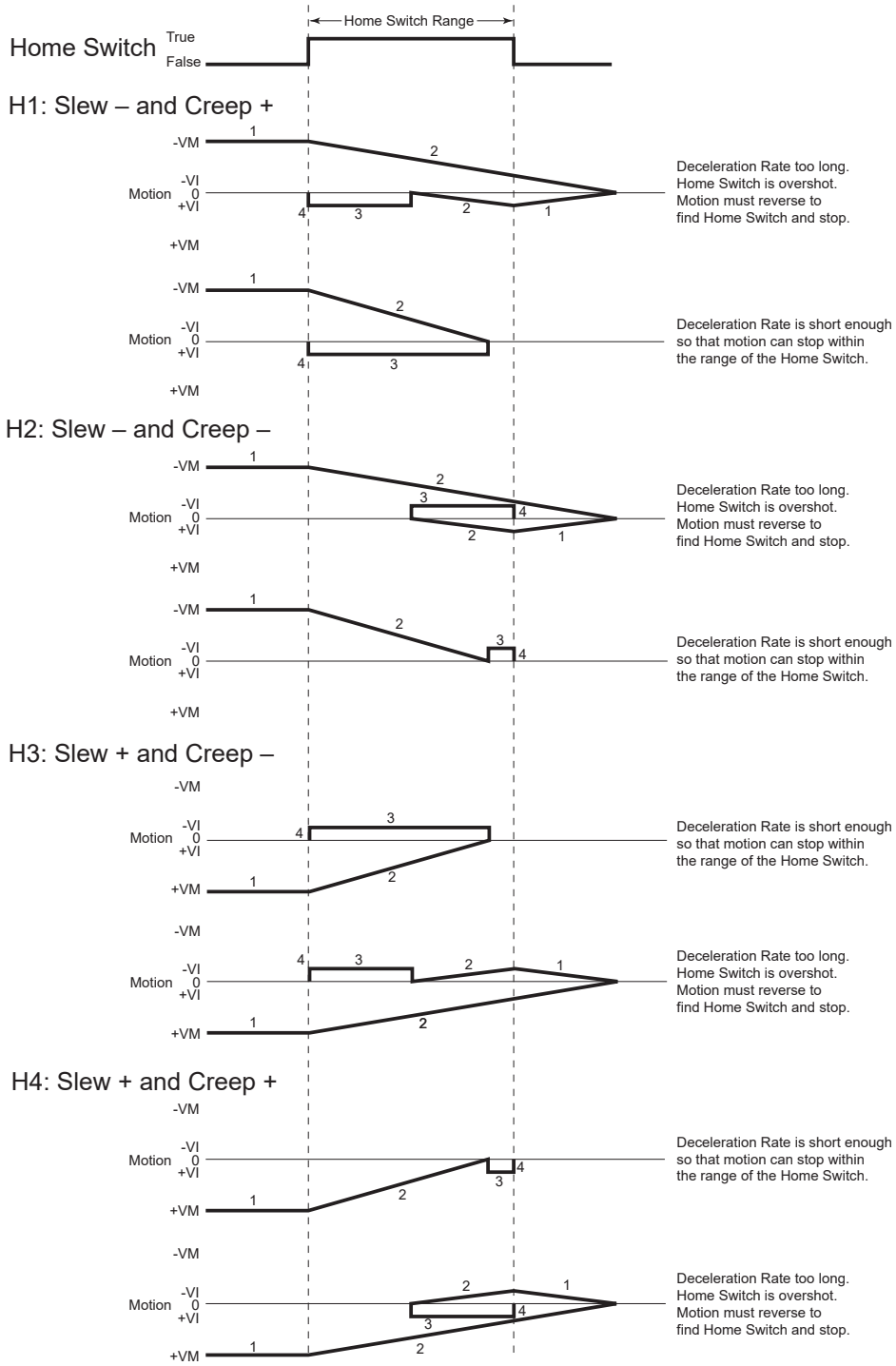
Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x002B
	0x68	1	0x02	USINT		

Homing Types

The diagram below displays the Homing (HM) Function sequence of events. 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.

NOTE: MT (Motor Settling Delay) time is still in affect.



Mnemonic	Function	Function Group	Access	Usage
HT	Holding current delay	Motion variable	R/W	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

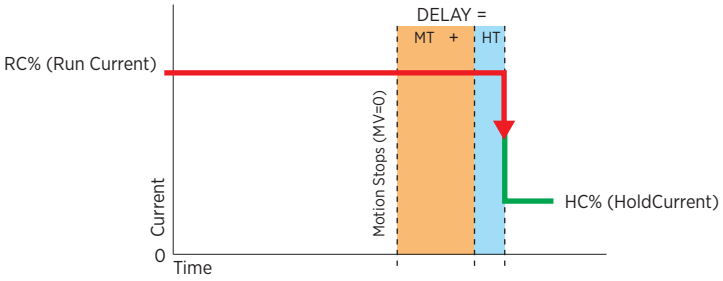
Delay in milliseconds between the **RC** (Motor Run Current) and **HC** (Motor Hold Current)., The delay time is also impacted by the **MT** (Motor Settling Delay) variable. The sum of **MT+HT** represents the total time delay between **RC** and **HC**.

The total of **MT+HT** cannot add up to more than 65535, thus, the value of **MT** is included in the total delay.

Therefore, the maximum setting for **HT**=(65535-**MT**). If setting **HT** to 0, **MT** is still in effect. If both **HT** and **MT** are set to 0, the current will not reduce, but maintain the **RC** (Run Current) percentage.

Exceeding this maximum results in an *Error 21: Illegal data value entered*.

This graphic shows the relationship between **HT** (Hold Current Delay) and **MT** (Motor Settling Delay)



Range:	0 to (65535-MT)	Units:	milliseconds	Default:	500

Syntax: HT=<time>, PR HT

CODE EXAMPLES

`HT=0`	Disable HT, motor will still delay the set MT value
`PR HT`	Read the value of HT (Hold Current Delay)

RELATED

HC (Hold Current)
MT (Motor Settling Delay)
RC (Run Current)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x66	1	0x04	UINT		0x002C

Mnemonic	Function	Function Group	Access	Usage
I1 — I4	Read input logic state	I/O variable	RO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Input 1 is not available on NEMA 17 (42 mm) models.

DESCRIPTION

The **I(x)** command is used to read the state of the specified input 1 - 4. **I(x)** is used with **PR** (Print), **BR** (Branch) and **CL** (Call Subroutine) instructions and with registers and user variables.

The response to the input state will be dependent on active (low/high) setting of the input.

Range:	0/1	Units:	—	Default:	—

Syntax: PR I<x>, BR <label/address>, I<x>=<0/1>, CL <label/address>, I<x>=<0/1>

CODE EXAMPLES

`PR I2`	Return the logic state of input 2
`BR L2,I3=1`	Branch to labeled location L2 when input 3 is HIGH
`CL Q5,I4=0`	Call subroutine Q5 when I4 is zero

RELATED

IN (Read all inputs)
IS (Input setup)
O<1-3> (Set output)
OS (output setup)
OT Set all outputs)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x67	1	0x09 — 0x0C	BOOL		0x002D 0x002E 0x002F 0x0030

Mnemonic	Function	Function Group	Access	Usage
I5	Read analog input	I/O variable	RO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

Reads the current value of the 12-bit analog input, which ranges from 0 to 4096 counts. The counts represent the signal amplitude sensed on the analog input.

Range: 0 to 4095 **Units:** counts **Default:** —

Syntax: PR I5, BR <label/address>, I5 =<integer>, CL <label/address>, I5=<integer>

CODE EXAMPLES

`PR I5`	Return the value of the analog input
`BR L2,I5>2048`	Branch to labeled location L2 when I5 is greater than 2048 counts
`CL Q5,I5=<2048`	Call subroutine Q5 when I5 is equal or less than 2048 counts

RELATED

IS (Input setup)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x67	1	0x0D	UINT	0x0031	

Mnemonic	Function	Function Group	Access	Usage
I6	Read encoder index mark	I/O variable	RO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

Reads the logic state of the encoder index mark. This will either be one or zero, as there are no configuration settings for the index mark.

Typical uses for this variable include: running a subroutine or incrementing a counter variable when the index mark is active.

Range: 0 or 1 **Units:** — **Default:** —

Syntax: PR I6, BR <label/address>, I6 =<0/1>, CL <label/address>, I6=<0/1>

CODE EXAMPLES

`PR I6`	Read the value of the encoder index
`BR L2,I6=0`	Branch to labeled location L2 when I6 is zero
`CL Q5,I6=1`	Call subroutine Q5 when I6 is one

RELATED

IS (Input setup)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x69	1	0x05	BOOL	0x0032	

Mnemonic	Function	Function Group	Access	Usage
I7 — I13	Reserved	Reserved	—	—

Compatibility: — **Notes**: —

DESCRIPTION

Reserved for factory use. Attempting to use as a user variable or label will result in an *Error 24: Illegal data entered*.

Range: — **Units:** — **Default:** —

Syntax: —

CODE EXAMPLES — **RELATED** —

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	—	—	—	—	—	

Mnemonic	Function	Function Group	Access	Usage
IC	Increment variable	Instruction	CMD	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Increments the specified variable by adding one.

Attempting to increment an unspecified or a read-only variable asserts an *Error 25: variable is read-only*.

Attempting to increment a mode select or configuration variable, for example **MS** (Microstep Resolution) results in an *Error 26: attempting to increment or decrement an illegal variable*.

Range: —

Units: —

Default: —

Syntax: IC <var>

CODE EXAMPLES

`IC R1` Increment register 1

`IC V2` Increment user variable V2

RELATED

DC (Decrement Variable)

Network Protocol Equivalents:

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0037
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
IF	Variable input pending	Conditional Flag	R/C	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

The **IF** instruction is automatically set to 1 when the **IV** command is executed. The **IF** flag reflects an input value from the communications 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 by entering **IF**=0.

NOTE: The IF instruction can only be cleared, not manually set to 1.

Range: 0/1

Units: —

Default: 0

Syntax: IF=0

CODE EXAMPLES

`IF=0` Clear the input variable pending flag

RELATED

IV (Input to Variable)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
IN	Read all inputs as a group	I/O keyword	RO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Reads the binary state of the inputs and returns them as a decimal value. When used, 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 **PR** (Print), **BR** (Branch) and **CL** (Call Sub-routine) instructions.

The value is a function of the actual state of the I/O where 1 = input voltage (+5 to +24 +VDC) and 0 = Ground. The level used to define the active state is a parameter of the **IS** (Input setup) variable.

Digital input filtering (**D1-D4**) has no effect on the data read.

* LMD NEMA 17 (42 mm) products have only three inputs. In these products input 3 will be the MSb and the total range is IN=<0-14>

Range:	Units:	Default:
0 - 15	—	—

Syntax: PR IN, BR <addr/lbl>,IN=<0-15>, CL <addr/lbl>, IN<0-15>

CODE EXAMPLES

`PR IN >07`	Print value of I4-I1
`BR L5,IN=07`	Branch to named location L5 if IN=07
`CL K3,IN=13`	Call subroutine K3 if IN=13

RELATED

IS (Input setup)
OS (Output setup)
OT (Set Outputs as Group)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x67	1	0x0E	USINT	0x003B	

Mnemonic	Function	Function Group	Access	Usage
IP	Initialize parameters	Instruction	WO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Restores all of the device variable and flag parameters to their stored values. This instruction does not delete user variables, but it restores the last saved value to the user variable.

If **IP** is used while the motor is moving, an *Error 74: Tried to initialize parameters or clear program while moving* occurs.

Range:	Units:	Default:
—	—	—

Syntax: IP

CODE EXAMPLES

`IP`	Return all saved variable values and flag states to the last saved.

RELATED

FD (Factory Defaults) S (Save)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	—	—	—	—	—	

Mnemonic	Function	Function Group	Access	Usage
IS <1-4>	Setup Inputs 1 to 4	I/O Instruction	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Clock input types (13, 14, 15) are only available on models with Firmware 6.001+
LMD NEMA 17 (42 mm) models are not equipped with Input 1.

DESCRIPTION

This instruction is used to configure the input parameters. These parameters define the function and active state. When used as a keyword (**PR IS**), the instruction will return the configuration of all inputs.

Input Parameters

Param	Description	Values	Default
1	Input line number	1 - 4	—
2	Input function type	(see type table)	0 (General Purpose User)
3	Input Active HI/LO state*	0 (Software LOW when input electrically energized) 1 (Software HIGH when input electrically de-energized)	1 (Software HIGH)

* Refer to "Active HI/LO States Defined" on page 136 for additional explanation and examples.

Input Function Types

Type	Function	Notes/restrictions
0	**General purpose user**; (default for all inputs) typically used to trigger events within a program.	—
1	**Homing**; functionality is defined by the HM instruction.	See HM (Home to Home)
2	**Limit +**; functionality is defined by the LM instruction. Note Limits do not function in hMT torque mode (AS=3).	See LM (Limit Response)
3	**Limit –**; functionality is defined by the LM instruction. Note Limits do not function in hMT torque mode (AS=3).	See LM (Limit Response)
4	**G0**; will execute a program stored at address 1 on 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 +**; jogs motor in a positive direction at VM (Maximum Velocity). JE (Jog Enable) must be set for this to function.	See JE (Jog Enable) and VM (Max. Velocity)
8	**Jog –**; jogs motor in a minus direction at VM (Maximum Velocity). JE (Jog Enable) must be set for this to function.	
11	**Reset**; equivalent to a ^C entered into a terminal. Note: If setting the input to sourcing, HIGH, ground the input first or a reset will occur.	—
12	**Capture**; operates with the Trip Capture (TC) trip to run a subroutine when active.	High speed function available on Input #1 ONLY. Not available on NEMA 17 (42 mm) models.
13	**Step (IN3) / Direction (IN 4)**; step clock and direction inputs.	The clock functions may only be assigned to inputs 3 and 4 as a pair. Setting one of the inputs to a clock function will automatically change the opposite input to the corresponding type.
14	**ENC A (IN3) / ENC B (IN 4)**; encoder channel A & B inputs.	
15	**StepUp (IN3) / StepDown (IN 4)**; step up and step down inputs.	

Range: — **Units:** — **Default:** —

Syntax: IS=<1-4>,<type>,<active> | PR IS

CODE EXAMPLES

`IS=1,1,1`	Set input 1 to Homing function, HIGH active
`IS=3,13,0`	Set input 3 to step clock function type. Input 4 will automatically set to direction type

`PR IS`
`  IS = 1, 2, 1`
`  IS = 2, 0, 1`
`  IS = 3, 13, 0`
`  IS = 4, 13, 0`
`  IS = 6, 0, 1`
	Return the input settings
	Response: notice IN4 automatically sets to the corresponding clock type

RELATED

D<1-4> (Input Filter)
I<1-4> (Read Input)
IN (Read All Inputs)
O<1-4> (Set Output)
OS (Output Setup)
OT (Set All Outputs)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x67	1	0x0F	STRING		Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes

Mnemonic	Function	Function Group	Access	Usage
IS <5>	Setup Analog Input	I/O Instruction	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: The analog input on the LMD Motion Module is not configurable.

DESCRIPTION

This instruction is used to configure the analog input sense and range for the LMD products. When used as a keyword (**PR IS**), the instruction will return the configuration of all inputs.

NOTE: The LMD Motion Module (LMM) analog input is fixed at voltage mode with an unbuffered range limit of 0 to 3.6V. Attempting to set the LMM analog parameters will result in an *Error 24: Illegal data entered*. Refer to the LMM hardware manual for example interface circuits.

LMD Parameters

Param	Description	Values	Default
1	Input line number	5 (Analog input)	—
2	Sense	9 (Voltage)	9 (Voltage)
		10 (Current)	
3	Range	0 (0 to 5V / 0 to 20 mA)	0 (0 to 5V)
		1 (0 to 10V / 4 to 20 mA)	

Range: — **Units:** — **Default:** —

Syntax: IS=<5>,<sense>,<range> | PR IS

CODE EXAMPLES

`IS=5,9,1` Set the analog input to voltage mode with a 0 to 10V range.

`IS=5,10,0` Set the analog input to current mode with a 0 to 20mA range.

Related:

D5 (Analog Input Filter) I<1-4> (Read Input)
IN (Read All Inputs) O<1-4> (Set Output)
OS (Output Setup) OT (Set All Outputs)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x67	1	0x0F	STRING		Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes

Mnemonic	Function	Function Group	Access	Usage
IS <6>	Setup Encoder Index	I/O Instruction	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Encoder Required

DESCRIPTION

This applies strictly to the encoder index mark.

The only user configurable parameter is the active state.

Param	Description	Values	Default
1	Input line number	6 (Index input)	6 (Index input)
2	Active	0 (LOW active)	0 (LOW active)
		1 (HIGH active)	

Range: — **Units:** — **Default:** —

Syntax: IS=<5>,<sense>,<range> | PR IS

CODE EXAMPLES

`IS=6,1` Set the encoder index input response to HIGH active

RELATED

EE (Encoder Enable) IN (Read All Inputs)
FS (Index Offset) HF (Home to Index Offset)
I6 (Read Index) HI (Home to Index

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x67	1	0x0F	STRING		Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes

Mnemonic	Function	Function Group	Access	Usage
IT	Read internal temperature	Status Keyword	RO	Program/Immediate

Compatibility: ■ LMD(O) ■ LMD(C) ■ LMD(A) ■ LMM

Notes:

DESCRIPTION

This keyword, when used with the **PR** (Print instruction) will return the internal temperature of the device electronics, measured at two locations, in the following order.

1. Driver dual h-bridge
2. Microcontroller

Param	Description
<blank>	Read both sensors, bridge first, then µController
1	Read the bridge sensor
2	Read the µController sensor

NOTE: For devices with Absolute Encoder, the **PR IT** command will only display the µController temperature. Adding additional parameters to the **PR IT** command will result in an *Error 24: Illegal Data Entered*.

Range: -20 to 100 **Units:** °C **Default:** —

Syntax: PR IT,<param>

CODE EXAMPLES

```
PR IT
   34,37
```
Return the internal temperature
Bridge temp = 34 °C, controller temp = 37 °C

```
PR IT,1
   34
```
Return the internal temperature of the bridge sensor
Bridge temp = 34 °C,

```
PR IT,2
   37
```
Return the internal temperature of the µController sensor
µController temp = 37 °C,

With Absolute Encoder:

```
PR IT
   37
```
Return the internal temperature of the µController sensor
µController temp = 37 °C,

RELATED

WT (Warning Temperature)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x65	1	0x04	STRING		Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes

Mnemonic	Function	Function Group	Access	Usage
IV	Input to Variable	Program instruction	CMD	Program

Compatibility: ■ LMD(O) ■ LMD(C) ■ LMD(A) ■ LMM

Notes: —

DESCRIPTION

The **IV** instruction facilitates the input of numeric data into a system or user-defined variable. User variables MUST be declared prior to issuing an **IV**.

When using **IV**, a conditional program loop using the logic state of the **IF** (Variable Input Pending) flag is needed.

Range: — **Units:** — **Default:** —

Syntax: IV <var/reg>

CODE EXAMPLES

IV used with conditional loop example:

```
IV F1
```
Input numeric into floating point register 1

```
LB X2
BR X2,IF=1
```
Conditional loop to suspend program while the variable input is pending, once the input is satisfied the IF flag will clear and the program will continue, with the value input stored in variable F1

RELATED

IF (Variable Input Pending)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	—	—	—	—		—

Mnemonic	Function	Function Group	Access	Usage
JE	Enable jog functions	Configuration flag	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

JE enables/disables input jog functions. Jogging the motor with using an input point requires the two parameters be configured.

1. The **JE** (Jog Enable) must be set to 1 (enabled). By default it is 0 (disabled)
2. Jog - and/or Jog + input function must be assigned to the appropriate inputs.

State	Description
0	Jog functions disabled (default)
1	Jog functions enabled

Range: 0/1 **Units:** — **Default:** 0

Syntax: JE

CODE EXAMPLES

`JE=1`	Enable jog functions
`PR JE` `>1`	Return the enabled/disabled state of jog functions

RELATED

IS (Input Setup)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x003F
	0x66	1	0x05	BOOL		

Mnemonic	Function	Function Group	Access	Usage
L	List the contents of program memory	Program instruction	CMD	Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

Retrieves the contents of program memory beginning at the specified label or address to the end of user program space. If no parameter is given it will list the full contents of user program space beginning at address 1.

Range: — **Units:** — **Default:** —

Syntax: L <label/address>

CODE EXAMPLES

`L`	Return the contents of program memory, beginning at address 1
`L G1`	Return the contents of program memory, beginning at label G1

RELATED

AL (List all Parameters)
CP (Clear Program Space)
FD (Factory Defaults
IP (Initialize Parameters)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
LB	Label Program or Subroutine	Program Instruction	CMD	Program

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

The Label Instruction allows a 2 character user-defined name to a program, program location, or subroutine. This label is then accessed within a program using the **BR** (Branch) and **CL** (Call Subroutine) instructions.

Labels applied to a program may be executed from immediate mode using the **EX** (Execute Program) command, or be label target subroutines for the various MCode trip functions.

There is a limit of 336, an amount shared with user variable names created using the **VA** (Create User Variable) instruction.

The restrictions for this command are:

1. A label cannot be named after an LMD MCode Instruction, Variable, Flag, or Keyword.
2. In naming user labels, the first character must be alpha, followed by 1 alpha character OR a number from 0 to 31.
3. A program labeled **SU** will run on power-up
4. Labels ARE NOT case sensitive.

Usage Tip:

Establish labeling conventions prior to beginning to write a program. For example: G1, G2, G3... for executable programs, V1, V2, V3 ... for user variables, Q1, Q2, Q3 ... for subroutines, B1, B2, B3 ... for branch targets, T1, T2, T3 ... for trip routines and etc.

Range:	Units:	Default:
—	—	—

Syntax: LB <alpha><alpha> or LB <alpha><0-31>

CODE EXAMPLES

`LB G1` Label program or location to G1

RELATED

BR (Branch)
CL (Call Subroutine)
EX (Execute Program)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	—	—	—	—		—

Mnemonic	Function	Function Group	Access	Usage
LD	Lead limit	hMT Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

LD sets the limit in motor steps in which the rotor may lead the stator for hMTechnology. When this limit is reached, an *Error 106: Lead limit reached*, is asserted.

A deceleration rate too high for load can cause the rotor position to lead the stator position.

NOTE: Lead Limit values are only active when AS (hMTechnology Mode) is set to 1, 2, or 3

Range:	Units:	Default:
0 to 2147483647	motor steps	102400

Syntax: LD=<steps>, PR LD

CODE EXAMPLES

`LD=51200` Set the lead limit for hMT to 51200 motor steps

`PR LD >51200` Read the value of the lead limit

RELATED

LG (Lag Limit)
LL (Position Lead-Lag)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x6A	1	0x06	UDINT		0x0095 –0x0096

Mnemonic	Function	Function Group	Access	Usage
LG	Lag limit	hMT Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

LG sets the limit in motor steps in which the rotor may lag the stator for hMTechnology. When this limit is reached, an *Error 107: Lag limit reached*, is asserted.

Conditions causing the rotor position to lag the stator position:

1. Acceleration rate to high for load
2. Transient load, sudden interruption in the load due to load inertia change or mechanical changes in the system.

NOTE: Lag Limit values are only active when **AS** (hMTechnology Mode) is set to 1, 2, or 3

Range: 0 to 2147483647 **Units:** motor steps **Default:** 102400

Syntax: LG=<steps>, PR LG

CODE EXAMPLES

`LG=51200`	Set the lead limit for hMT to 51200 motor steps
`PR LG` `>51200`	Read the value of the lead limit

RELATED

AS (hMT Mode)

LD (Lead Limit)

LL (Position Lead-Lag)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x6A	1	0x08	UDINT	0x0097 –0x0098	

Mnemonic	Function	Function Group	Access	Usage
LL	Position Lead/Lag Register	hMTechnology Variable	RO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Read only register holding the number of counts that the rotor leads or lags the stator. A positive value indicates position lag. A negative value indicates position lead.

hMTechnology will use this counter for position correction.

NOTE: **LL** values are only measured when **AS** (hMTechnology Mode) is set to 1, 2, or 3.

Range: -2147483647 to +2147483647 **Units:** motor steps **Default:** —

Syntax: PR LL, CL <label/address>, LL<operator><steps>, BR <label/address>, LL<operator><steps>

CODE EXAMPLES

`CL k5,LL>102500`	Call k5 if LL is greater than 102500
`PR LL` `>0`	Read the value of the Lead/Lag register

RELATED

AS (hMT Mode)

LD (Lead Limit)

LG (Lag Limit)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x6A	1	0x07	DINT	0x0099 – 0x009A	

Mnemonic	Function	Function Group	Access	Usage
LK	Lock user program space	Program Flag	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

LK may be used to prohibit user interaction with stored MCode programs by disallowing:

1. Program upload
2. Modification
3. Listing

State	Description
0	User program space unlocked (default)
1	User program space locked

Once enabled, attempting to list or modify the stored program space will result in an *Error 44: User program space locked*.

The program space may only be unlocked by issuing a full **CP** (Clear Program Space) without parameters or by entering an **FD** (Reset to Factory Defaults).

If not saved, a lock may be cleared by a power cycle or software reset (CTRL+C).

Range:	Units:	Default:
0/1	—	0

Syntax: LK=<0/1>, PR LK

CODE EXAMPLES

```
LK=1          Lock user program space to prevent upload/list/modification
PR LK         Print the status of the LK flag.
  >1
```

RELATED

L (List Program Space)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
LM	Limit Response Mode	I/O Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

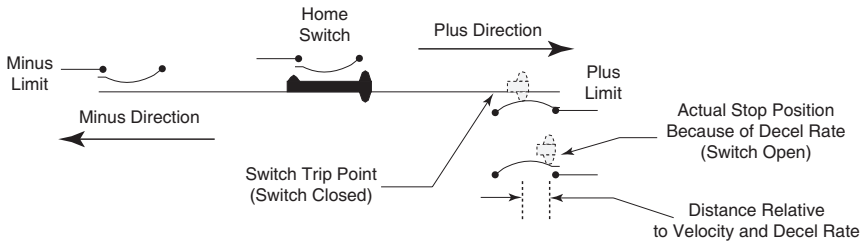
Notes: —

DESCRIPTION

LM defines the response taken when a limit is reached. The mode for **LM** applies to both hardware I/O limit switches or position limits set in software.

Prerequisites:

1. Limits must be configured, either hardware switch limits using the IS (Input Setup) command, or software limits configured using the **LS** (Software Limits) variable.
2. Limits only work in the defined direction of travel; i.e., +limit only works in the positive direction, — limits only work in the minus direction.
3. If the limit is active and maintained, the software will only allow motion in the opposite direction.
4. If homing is active **HM** (Home to Home Switch), motion will decelerate to a stop, then reverse direction to seek the home switch. If the home switch is not reached on the reverse and the opposite limit is reached, all motion will stop with a deceleration ramp.



Limit Response Modes

Mode	Operation
1	**Normal limit function with a deceleration ramp: motion stops, unless homing**. If the limit is active and maintained, the software will only allow motion in the opposite direction. If homing is active HM (Home to Home Switch), motion will decelerate to a stop, then reverse direction to seek the home switch/ If the home switch is not reached on the reverse and the opposite limit is reached, all motion will stop with a deceleration ramp.
2	A limit stops all motion with a deceleration ramp whether or not homing is active
3	A limit stops 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

Range:	1 to 6	Units:	Mode	Default:	1

Syntax: LM=<mode>, PR LM

CODE EXAMPLES

`LM=6`	Stop motion without deceleration and program execution when a limit is reached
`PR LM` `6`	Return the current limit stop mode

RELATED

HM (Home to Home)
IS (Input Setup)
LS (Soft Limits)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0042
	0x66	1	0x06	USINT		

Mnemonic	Function	Function Group	Access	Usage
LR	Locked Rotor	hMT Status Flag	RO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

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 hMT is configured **AS**=1 or 2, a locked rotor will also cause an internal fault (**LR**) disabling the output bridge.

State	Description
0	Rotor unlocked (default)
1	Rotor locked

The flag may be cleared and the bridges re-enabled by cycling power, or via software command **CF**: Clear Locked Rotor Fault. A locked rotor condition will result in an *Error 104: hMT Locked Rotor* as well.

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.

Range:	Units:	Default:
0/1	—	0

Syntax: PR LR

CODE EXAMPLES

```
PR LR
  0
```
Return the lock status of the rotor

RELATED

AO (Attention Output) | AF (hMT Status) | AS (hMT Mode)
CF (Clear Locked Rotor) | LT (Locked Rotor Timeout) | TA (Trip on hMT Status)

Network Protocol Equivalents:

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x009B
	0x6A	1	0x09	BOOL		

Mnemonic	Function	Function Group	Access	Usage
LS	Set/Read ± Software Limits	Motion Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: Firmware 6.001+

DESCRIPTION

Sets the direction, position and enabled state for software limit switches. There are three parameters:

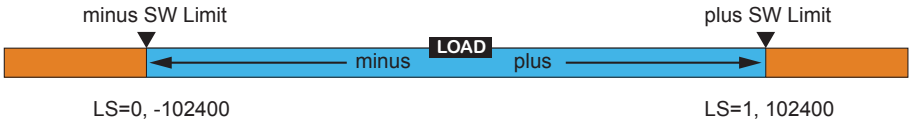
The first parameter provides the limit direction.

The second parameter provides the position at which the limits will respond.
NOTE: The limits must have a logical gap, meaning that the negative limit must be set to a value more negative than the positive limit.

Param	Description	Values	Default
1	Limit direction	0 (lower limit)	0
		1 (upper limit)	1
2	Position	-2147483648 to 214748364	0
3	Enable/disable	0 (disable)	0 (disabled)
		1 (enable)	

The third parameter enables or disables the limit function.

When a software limit is reached, the product will respond as specified by the **LM** (Limit Response Mode) variable.



Range: See parameter table above **Units:** See parameter table above **Default:** See parameter table above

Syntax: LS=<0/1>,<±position>,<0/1> | PR LS

CODE EXAMPLES

```
LS=0,-102400,1
LS=1,102400,1
```
Set minus and plus software limits, enable

```
PR LS
  LS=0,-102400,1
  LS=1,102400,1
```
Return the software limit configuration

RELATED

HM (Home to Home)
LM (Limit Stop Mode)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP
	0x66	1	0x1B	STRING	Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes

Mnemonic	Function	Function Group	Access	Usage
LT	Set/Read Locked Rotor Timeout	hMT Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

Defines the time in milliseconds between the assertion of an **LR** (Locked Rotor) condition and the output H-bridges being disabled

NOTE: If the LMD is in hMTechnology Torque Mode (**AS**=3), the output bridges will not disable upon a locked rotor condition

Range: 2 to 65535 **Units:** milliseconds **Default:** 2000

Syntax: LT=<time>, PR LT

CODE EXAMPLES

```
LT=50
```
Set the locked rotor timeout to 50 msec

```
PR LT
   50
```
Read the locked rotor timeout

RELATED

AS (hMT Mode)
LR (Locked Rotor)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP
	0x6A	1	0x0A	UINT	0X009C – 0x009D

Mnemonic	Function	Function Group	Access	Usage
MA	Move to Absolute Position	Motion Instruction	WO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

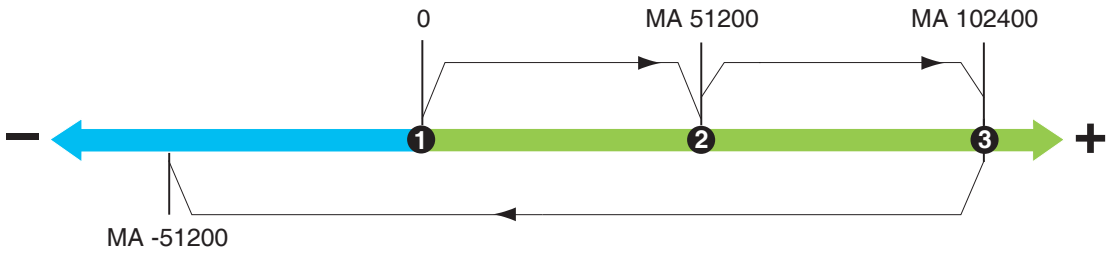
Set mode for absolute move and move to an absolute position relative to (0) zero. **MD** (Motion Mode) will be set to **MA**. **MA** moves the axis to a position in motor steps relative to zero (0).

In the case of the profile shown below:

The first **MA** (51200) will result in the axis ending 51200 motor steps from 0 (position 2).

The second **MA** (102400), moves the axis an additional 51200 steps, ending at position 3 or 102400 steps from 0.

The third **MA** (-51200) will index the axis 153600 steps in the negative direction from position 3, to a final position of -51200 absolute from 0.



NOTES:

- The **MA** command will not operate during a homing sequence.
- An in-progress **MA** can be stopped by pressing [ESC] on the keyboard or entering an **SL** 0 command.
- The time required to calculate each move is 20 µSec.
- The position value following each **MA** move will be stored in **P** (Position Counter).

In addition to the commanded position, there are two optional parameters to define specific functions within the move.

Param	Description	Values	Default
1	± Motor position	-2147483648 to +2147483648	—
2	Party Mode (PY) response	0 - no operation	0
		1 - send DN (Device Name) out the communications port following move completion. The device name will be sent regardless of the PY setting.	
3	Motion	0 (or blank) Stop motion after reaching position	0
		1 continue moving after position is reached	

Range:	See table	**Units:**	motor steps	**Default:**	See table

Syntax: MA <±position>, <param>, <param>

CODE EXAMPLES

`MA 102400`	Move to absolute position 102400
`MA 102400,0,1`	Move to absolute position 102400, do not stop motion upon position

RELATED

MD (Motion Mode)
MR (Move Relative)
P (Position Counter)
SL (Slew)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0043 - 0x0044
	0x66	1	0x07	DINT		

Mnemonic	Function	Function Group	Access	Usage
MD	Motion Mode	Status Variable	RO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Read-only status variable holds the last used motion instruction. It is used in the (invisibly to the user) with the **NE** (Numeric Enable) flag to facilitate repeated move types (absolute position, relative position or slew) by entering a numeric value instead of the full command string.

May be used as a keyword with the **PR** (Print) instruction to view the last move type. The device will respond with the command mnemonic: **MA** (Move Absolute), **MR** (Move Relative) or **SL** (Slew).

Range:	Units:	Default:
MA, MR, or SL	—	—

Syntax: PR MD

CODE EXAMPLES

```
MA 51200      Perform a move
PR MD         Display the previous move type
   MA
```

RELATED

MA (Move Absolute) MR (Move Relative)
SL (Slew) NE (Numeric Enable)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
MF	Read/Set hMT Make-up Frequency	hMT Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Defines the frequency at which missed steps are re-inserted into the move profile when **MU** (Make-up Mode) is set to mode 1.

Can be used as a keyword with the **PR** (Print) command to display the stored value for **MF**.

Range:	Units:	Default:
92 to VM	motor steps/sec	768000

Syntax: MF=<steps/sec>, PR MF

CODE EXAMPLES

```
MF=512000     Set hMT make-up to 512000 steps/sec
PR MF         Read the current value of MF
   512000
```

RELATED

AS (hMT Mode)
MU (Make-up Mode)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x009E – 0x009F
	0x6A	1	0x0B	UDINT		

Mnemonic	Function	Function Group	Access	Usage
MP	Moving to a Position	Status Flag	RO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Read-only status flag is active (1) when the axis is indexing to a position. This moving to position flag can be used in writing a subroutine to wait while positional moves are in progress.

NOTE: **MP** will be active for the total move, which includes the delays added to compensate for **HT** (Hold Current Delay) and **MT** (Motor Settling Delay).

State	Description
0	Not moving to position
1	Positional move in progress

The moving to position flag may be used to give external indication via an output point specifically configured for the Moving to Position type (Os=<output>,23,<active>).

Range: 0/1 **Units:** — **Default:** —

Syntax: CL <label/address>,MP=<0/1> | PR MP

CODE EXAMPLES

```
MA 5120000
PR MP
   1
```

Make a positional move, return the MP status while axis is moving,

RELATED

MA (Move Absolute)

MR (Move Relative)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0045
	0x66	1	0x08	BOOL		

Mnemonic	Function	Function Group	Access	Usage
MR	Move to Relative Position	Motion Instruction	WO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

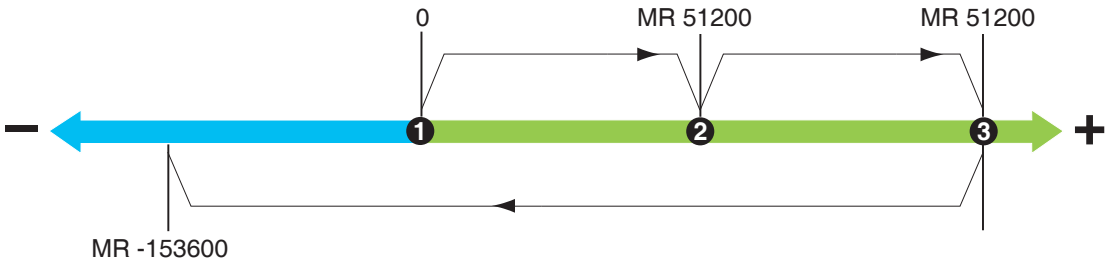
Set mode for relative move and move to a position relative to the current position. **MD** (Motion Mode) will be set to MR.

MR moves the axis to a position in motor steps relative to the current motor position. In the case of the profile shown in the figure above, the start position (1) of the axis is zero (0).

The result of the first **MR** (51200) will result in the axis moving 51200 steps from the start position, or 51200 motor steps (position 2 in the example) from 0.

The second **MR** (51200) will result in the axis moving an additional 51200 steps from its current position, ending at position 3 or 102400 steps from 0.

The third **MR** (-153600) will index the axis -153600 steps in the negative direction from position 3, with a final position of -51200 relative to the starting position of 0.



NOTES:

- The **MR** command will not operate during a homing sequence.
- An in progress **MR** can be stopped with an [ESC] or an **SL** 0 command entry.
- The time required to calculate each move is 20 µSec.
- The axis position following each **MR** move will be stored in **P** (Position Counter).

In addition to the commanded position, there are two optional parameters to define specific functions within the move.

Param	Description	Values	Default
1	± Motor position	-2147483648 to +2147483648	—
2	Party Mode (PY) response	0 - no operation	0
		1 - send DN (Device Name) out the communications port following move completion. The device name will be sent regardless of the PY setting.	
3	Motion	0 (or blank) Stop motion after reaching position	0
		1 continue moving after position is reached	

Range:	See table	**Units:**	motor steps	**Default:**	See table

Syntax: MR <±position>, <param>, <param>

CODE EXAMPLES

`MR 102400`	Move 102400 steps relative to the current motor position
`MR 102400,0,1`	Move 102400 steps relative to the current motor position, do not stop motion upon position

RELATED

MA (Move Absolute)
MD (Motion Mode)
P (Position Counter)
SL (Slew)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0046 - 0x0047
	0x66	1	0x09	DINT		

Mnemonic	Function	Function Group	Access	Usage
MS	Set/read Microstep Resolution	Motion Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

This command sets the Microstep Resolution for the device. There are 20 fixed microstep resolutions that the LMD Motion product will accept ranging from full step (**MR**=1) to 256 microsteps per full step, or **MR**=256.

It is important to consider that when changing **MS** (Microstep Resolution), other motion variables will automatically scale to the equivalent ratio, as shown in the table below. The settings for a particular velocity profile can change dramatically based on the setting of **MS**:

	MS=<param>	Steps/rev	VI (Initial V)	VM (Max V)	A	D
Default	MS=256	51200	1000	768000	1000000	1000000
Change	MS=2	400	4	3000	3906	3906

The parameter table below is based upon the LMD products with 1.8° (200 Step/Rev) motor. If using an LMD modular product with a different motor, the motor resolution will apply. For example, a 0.9° motor has 400 steps per revolution. The following equation applies where **SA** is the setting of the Step Angle variable.

Steps/Rev = (360/SA)***MS**

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)		
*Do not use with hMT active									

All shown steps per revolution values assume the 1.8° motor standard with LMD products. If using a custom integrated product, or an LMD Motion Module with a motor with a step angle other than 1.8°, refer to the **SA** (Step Angle) command.

Range: See parameter table **Units:** steps/full step **Default:** 256

Syntax: MS=<param>, PR MS

CODE EXAMPLES

`MS=4`	Set microstep resolution to 4 steps/full step
`PR MS 4`	Return the microstep resolution setting

RELATED

A (Acceleration) D (Deceleration)
VI (Initial Velocity) MA (Move Absolute)
SA (Step Angle) MR (Move Relative)
SL (Slew at Velocity) VM (Maximum Velocity)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0048
	0x66	1	0x0A	UINT		

Mnemonic	Function	Function Group	Access	Usage
MT	Set/Read Motor Settling Delay	Motion Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

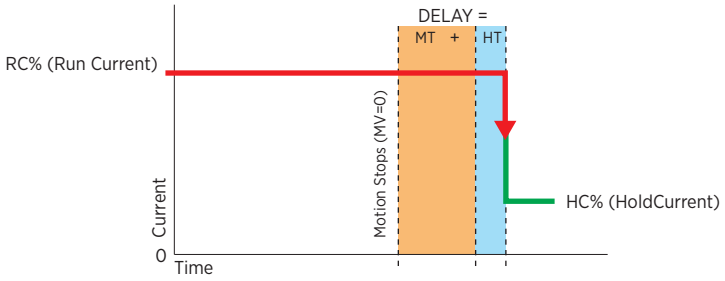
DESCRIPTION

Delay, measured in milliseconds, given to allow the motor to settle into position following a move. Total delay time between **RC** (Run Current) and HC (Hold Current) is the sum of the **MT** and **HT** (Hold Current Delay) values.

ThIS diagram below shows the relationship between **MT** and **HT**.

The total of **MT + HT** cannot exceed 65535. Exceeding this value results in an *Error 21: Illegal data value entered*. Therefore, the maximum setting for **MT** = (65535 - **HT**).

If **HT** is set to 0, **MT** is still in effect. If both **HT** and **MT** are set to 0, the current will not reduce, but maintain the **RC** (Run Current) percentage.



NOTE: **MT** should be at least 50 mS when encoder functions are enabled (**EE**=1).

Range: 0 to (65535-HT) **Units:** milliseconds **Default:** 500

Syntax: MT=<time>, PR MT

CODE EXAMPLES

`MT=0`	Disable MT, motor will still delay the set HT value
`PR MT` `0`	Read the value of MT (Motor Settling Delay)

RELATED

HC (Hold Current)
HT (Hold Current Delay)
RC (Run Current)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x66	1	0x0B	UINT	0x0049	

Mnemonic	Function	Function Group	Access	Usage
MU	Set/Read Make-Up Mode for hMT	hMT Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

Defines the mode for hMT position make-up. Make-up only occurs when motor lag/lead is within 1.1 motor steps. Make up steps may be integrated with motion steps and made after a move has completed.

Where make-up occurs is dependent on motor lag/lead, motion frequency and selected make up speed.

Make up mode will be cleared when bridges are disabled and hMT is enabled (**AS**=1 or 2).

Mode	Description
0	Make up position without regard to time (default)
1	Use make up frequency (MF) as make up frequency
2	Use system speed (SS), an internally defined velocity limited to 2560000 steps/sec (3000 RPM) as make up frequency

NOTE: Make-up is an advanced hMT function covered in detail in "Hybrid Motion Technology (hMT)" on page 152 of this document.

Range: 0 to 2 **Units:** — **Default:** 0

Syntax: MU=<mode>

CODE EXAMPLES

`MU=1`	Set make-up to make up position using MF as the reference velocity
`PR MU` `1`	Return the set make-up mode

RELATED

AS (hMT Mode)
MF (Make-up Frequency)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x6A	1	0x0C	STRING	0x00A0	

Mnemonic	Function	Function Group	Access	Usage
MV	Axis is Moving	Status Flag	RO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Read-only status flag is active (1) when the axis is moving, regardless of the move type.

NOTES:

– **MV** will be active for the total move, which includes the delays added to compensate for **HT** (Hold Current Delay) and **MT** (Motor Settling Delay).

State	Description
0	Not moving
1	Moving

– The moving flag may be used to give external indication via either an output point specifically configured for the Moving type (**Os**=<output>,17,<active>) or by setting the attention output mask variable (**AO**=16384) to indicate on LED 2 (LMD Motion Control models only) or an output defined as the Attention Output type (**OS**=<output>,29,<active>).

Range: 0/1 **Units:** — **Default:** —

Syntax: CL <label/address>,MV=<0/1> | PR MV

CODE EXAMPLES

```
SL 51200        Slew the axis
PR MV           Return the MV status
   1
```

RELATED

MA (Move Absolute) OS (Output Setup)
MR (Move Relative) SL (Slew)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x004A
	0x66	1	0x0C	BOOL		

Mnemonic	Function	Function Group	Access	Usage
NE	Enable/disable Numeric Functions	Setup Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Facilitates repeated move types (absolute position, relative position or slew) by entering a numeric value instead of the full command string.

When a move is executed, the type of move (**MA**, **MR**, or **SL**) is stored in the **MD** (Motion Mode) variable. This stored value will be used as the move type whenever **NE** is in an enabled state.

Value	Description
0	Disabled (default)
1	Numeric functions enabled

If disabled, the user must enter a motion command to execute a move, i.e., **MA** 100000, **MR** -50000, **SL** 300000 etc.

Range: 0/1 **Units:** — **Default:** 0

Syntax: NE=<0/1>, PR NE

CODE EXAMPLES

```
NE=1            Enable numeric functions

PR NE           Return the numeric enable state
   1              numeric functions are enabled
```

RELATED

MA (Move Absolute)
MD (Motion Mode)
MR (Move Relative)
SL (Slew)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
O1, O2, O3	Set Output #	I/O Instruction	RO	Program/Immediate
Compatibility: ■ LMD(O) ■ LMD(C) ■ LMD(A) ■ LMM		**Notes**: Outputs 1 and 2 are not available on NEMA 17 (42 mm) models.		

DESCRIPTION

Sets the state of the specified output to 1 or 0 for output type 16 (General Purpose User).

The output response is determined by the third parameter of **OS** (Output Setup), which defines the output as active when HIGH (1) or LOW (0).

NOTE: On LMDxM42x or LMDxE42x (NEMA 17) Outputs 1 and 2 are not present. Use of this command will result in an *Error 37: Command, variable or flag not available*.

Setting	Output Config	Output State
O<output>=0	OS=x,16,0	INACTIVE
	OS=x,16, 1	ACTIVE
O<output>=1	OS=x,16,0	ACTIVE
	OS=x,16, 1	INACTIVE

Range:	0/1	Units:	—	Default:	—

Syntax: O<1/2/3>=<0/1>

CODE EXAMPLES

`O1=1`	Set Output 1 to a value of 1
`O1=0`	Set Output 1 to a value of 0

RELATED

OF (Output Fault)
OS (Output Setup)
OT (Write All Outputs)

NETWORK PROTOCOL EQUIVALENTS

	Class	Instance	Attribute	Data Type		
Ethernet/IP	0x67	1	0x10 (O1) 0x11 (O2) 0x12 (O3)	BOOL	**Modbus/TCP**	0x004B (O1) 0x004C (O2) 0x004D (O3)

Mnemonic	Function	Function Group	Access	Usage
OE	On Error Handler	Program Instruction	WO	Program/Immediate
Compatibility: ■ LMD(O) ■ LMD(C) ■ LMD(A) ■ LMM		**Notes**: —		

DESCRIPTION

OE declares the label or address of the subroutine which will execute when an error code **ER** (Error Code) is received and **EF** (Error Flag) activated.

Attempting to target **OE** to a non-existent subroutine will result in an *Error 30: Unknown User Label or Variable*.

An **RT** (Return From Subroutine) must be inserted at the end of the subroutine called in an **OE**. After the subroutine completes, the program will return to the line following the command string that caused the error.

NOTES:

1. **OE** may be declared inside a program, between the opening and closing **PG** (Program Mode) tags.
2. **OE** may be declared in immediate mode ONLY if the target subroutine is resident in program memory space. The program need not be running.
3. Subroutines targeted by an **OE** will execute when an error is encountered during immediate operations. The target subroutine need only be resident in program memory space, it does not need to be running. A return from the subroutine should be programmed as well.
4. **OE** will not execute during programming.

Range:	—	Units:	—	Default:	—

Syntax: OE=<label/address>

CODE EXAMPLES

`OE Q1`	Execute subroutine Q1 when an Error is asserted

RELATED

EF (Error Flag) ER (Error Code)

NETWORK PROTOCOL EQUIVALENTS

	Class	Instance	Attribute	Data Type		
Ethernet/IP	—	—	—	—	**Modbus/TCP**	—

Mnemonic	Function	Function Group	Access	Usage
OF	Output Over Current Fault	I/O Variable	RO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Applicable to Outputs 1 & 2 only. Not applicable to LMM products

DESCRIPTION

Read-only status variable indicates an over-current fault condition on the power outputs (Outputs 1 and 2).

NOTE: An output over current condition will result in an *Error 1: OUT 1 fault* or *Error 2: OUT 2 fault*. Since the **ER** (Error Register) only displays the last error received, perform a read of the Output Fault to confirm if one or both outputs are be faulted.

Status code	Description
0	No Fault (default)
1	Over current fault on output 1
2	Over current fault on output 2
3	Over current fault on both output 1 and 2

Range:	0 to 4	**Units:**	—	**Default:**	0

Syntax: PR OF, <CL/BR> <label/address>,OF=<status>

CODE EXAMPLES

```
PR OF
0
```
Return the status of the outputs
no output fault conditions exist

```
CL Q1,OF=1
```
Call Q1 if an over current fault occurs on output 1

RELATED

EF (Error Flag)
ER (Error Register)
OE (On Error Handler)
OS (Output Setup)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x004E
	0x67	1	0x13	BOOL		

Mnemonic	Function	Function Group	Access	Usage
OS <1-3>	Setup Outputs 1 to 3	I/O Instruction	RW	Program/Immediate

Compatibility:
LMD(O) LMD(C) LMD(A) LMM

Notes:
LMD NEMA 17 (42 mm) models are equipped with OUT 3 (Signal Output) only.
LMD(A) NEMA 23 (57 mm) and NEMA 34 (85 mm) models do not have an OUT 2.
Some output functions are hMTechnology specific and not available on all products

DESCRIPTION

This instruction is used to configure the output parameters. These parameters define the function and active state.

When used as a keyword (**PR OS**), the instruction will return the configuration of all outputs.

LMD NEMA 17 (42 mm) models are equipped with OUT 3 (Signal Output) only. Attempting to setup Outputs 1 or 2 will generate an *Error 10: Illegal I/O number*.

Output Parameters

Param	Description	Values	Default
1	Output line number	1 - 3	—
2	Output function type	(see type table)	16 (General Purpose User)
3	Active HI/LO State	0 (LOW active), 1 (HIGH active)	0 (LOW active)

LMD NEMA 23 (57 mm) and NEMA 34 (85 mm) with Absolute Encoder are not equipped with OUT 2. Attempting to setup Output 2 will generate an *Error 10: Illegal I/O number*.

Some output functions are hMTechnology specific and not available on all products. Such variances are noted in the type table below.

Output Function Types

Type	Function	Notes/Restrictions
16	**General purpose user:** (default for all inputs) typically used to trigger events external to the device using O1-O3 and OT	See O<1-3> (Set Output) and OT (Set All Outputs)
17	**Moving**: active when the axis is in motion or awaiting the expiration of HC and MT delays.	See MV (Moving)
18	**Error**: active when an error condition exists, cleared by PR ER or ER=0	Se ER (Error)
20	**Velocity Changing**: active when the axis is changing velocity, such as acceleration and deceleration, linked to the VC (Velocity Changing) flag	See VC (Velocity Changing)
21	**Locked Rotor**: active when an hMT Locked Rotor condition exists	See LR (Locked Rotor) and CF (Clear Locked Rotor) [hMT LMD ONLY]
23	**Moving to a Position**: active while the axis is moving to a position from and MA (Move Absolute) or MR (Move Relative). Includes HT and MT delays, Linked to the status of the MP (Moving to Position) flag.	See MP (Moving to Position)
24	**hMT Active**: active whenever hMTechnology is compensating for load variances.	See AS (hMT Mode) [hMT LMD ONLY]
25	**Make-up Steps Active**: active whenever hMTechnology Make-up function is compensating for position errors.	See MU (Make-up Mode) [hMT LMD ONLY]
28	**Trip**: active when an assigned trip event occurs. Available on Output 3 (Signal Output) only. The trip function is active when LOW only.	See TA (Trip on hMT), TC (Trip on Capture), TD (Torque Direction), TE (Trip Enable), TI (Trip on Input), TM (Trip on Main Power Loss), TP (Trip on Position), TQ (Torque Percent), TR (Trip on Relative Position), TS (Torque Speed), and TT (Trip on Time)
29	**Attention**: active with regard to the AO (Attention Output Mask) setting.	See AO (Attention Output Mask)

Range: — **Units:** — **Default:** —

Syntax: OS=<1-3>,<type>,<active> | PR OS

CODE EXAMPLES

`OS=1,17,1`	Set output 1 to moving function, HIGH active
`OS=3,29`	Set output 3 to trip function type.

`PR OS`
`OS = 1, 17, 1`
`OS = 2, 16, 0`
`OS = 3, 29, 0`
	Return the output settings
	Response: settings of all output points

RELATED

IS (Input Setup)
O<1-3> (Set Output)
OT (Set All Outputs)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes
	0x67	1	0x14	STRING		

Mnemonic	Function	Function Group	Access	Usage
OT	Set the state of all outputs	I/O Instruction	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: LMD NEMA 17 (42 mm) models are equipped with OUT 3 (Signal Output) only.

DESCRIPTION

Allows the user to set outputs 1-3 as one 3 bit binary value. The value is entered in decimal, with a range of 0-7 in binary where Output 1 will be the LSb and Output 3 will be the MSb.

Range: 1 - 7 **Units:** — **Default:** —

Syntax: OT=<1-7>

CODE EXAMPLES

`OT=7` Set Output total to 7, all outputs will be active

RELATED

O<1-3> (Set Output) OS (Output Setup)

Network Protocol Equivalents:

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0056
	0x67	1	0x15	UINT		

Mnemonic	Function	Function Group	Access	Usage
P	Position Counter	Instruction	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Reads or writes the value of the position counter. The position will read in Motor Steps from **C1** (Counter 1) by default, if encoder functions are enabled on closed loop models, the position counter will read in Encoder Counts from **C2** (Counter 2).

Modifying **P** changes the frame of reference for the axis for Move Absolute (**MA**) instructions. **P** will likely be set once during system set up to reference or "home" for the system.

Range:		Units:		Default:
EE=0:	-2147483648 to +2147483647	**EE=0:**	Motor steps	—
EE=1:	-167772160 to +167772160	**EE=1:**	Encoder counts	

Syntax: P=<counts>, PR P, <CL/BR> <label/address>, P=<value>

CODE EXAMPLES

`P=0` zero the position counter

`PR P`
`0` Read the value of the position counter
the position counter is a 0

RELATED

C1 (Counter 1)
C2 (Counter 2)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0057 - 0x0058
	0x68	1	0x03	DINT		

Enabling the encoder (EE=1) or modifying the data in Motor Counter C1 or Encoder Counter C2 on LMD models with a multi-turn absolute encoder during operation will desynchronize the relationship between the counters and the Absolute encoder counter, causing a discrepancy between reported and actual shaft position.

NOTICE

DAMAGE TO COMPONENTS

- Do not modify, manually or by program, counters C1 or C2 while device is in motion.
- Do nor move outside the range of the counter, either by setting it manually or by rolling over the counter on LMD models with an absolute encoder.

Failure to follow these instructions can result in equipment damage.

Mnemonic	Function	Function Group	Access	Usage
PC	Position Capture at Trip	Instruction	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

Captures motor or encoder position during a trip event. Activation will occur upon any trip function EXCEPT a position trip (**TP** or **TR**). Will display in either motor steps (**EE**=0) or encoder counts (**EE**=1)

Range: — **Units:** — **Default:** —

Syntax: PR PC

CODE EXAMPLES

```
PR PC
   0
```
Return the captured position count
the captured position count is zero

RELATED

TA (Trip on hMT Status) TE (Trip Enable)
TC (Trip Capture) TI (Trip on Input)
TM (Trip on Main Power Loss) TT (Trip on Time)

Network Protocol Equivalents:

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0059 - 0x005A
	0x68	1	0x04	DINT		

Mnemonic	Function	Function Group	Access	Usage
PF	Set Print Format for Floating Point Registers	System Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: Firmware 6.001+

DESCRIPTION

Sets the format for displaying the contents of the floating point registers **F1** through **F8**.

This command is used to format floating point values for setting the width, digits following the decimal, notation and justification.
NOTE: This setting will not truncate the floating point register values for numbers that extend beyond the **PF** setting.

Param	Description	Values	Default
1	width	0 to 16 (includes ±sign and decimal	10
2	decimal	The number of digits to the right of the decimal	6
3	notation	0 (normal notation), 1 (scientific notation)	0
4	justification	0 (right), 1 (left)	0

Range: See parameter table **Units:** — **Default:** 10,6,0,0

Syntax: PF=<width>,<dec>,<0/1>,<0/1>, PR PF

CODE EXAMPLES

```
PR PF
   10,6,0,0
PR F1
   0.000000
```
Return the print format setting
default PF setting
Read the value of F1
formatted register contents

```
PF=8,4,1,1
PR F1
    0.0000E+00
```
Set print format to format
Read the value of F1
Value returned showing new PF - 4 digits after the decimal and scientific notation.

RELATED

F<1-8>(Floating Point Register)
PR (Print)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
PG	Enter/Leave Program Mode	Program Instruction	CMD	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Toggles the device into or out of program mode.

Range: — **Units:** — **Default:** —

Syntax: PG <address>

CODE EXAMPLES

```
PG 1                Enter program mode at address 1
[MAIN PROG]
[SUBROUTINES]

PG                  Exit program mode
E                   Designated end of program
```

RELATED

CP (Clear Program Memory)

FD (Factory Defaults)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
PK	Reserved	Reserved	—	—

Compatibility: —

Notes: —

DESCRIPTION

Reserved for factory use. Attempting to use as a user variable or label will result in an *Error 24: Illegal data entered*.

Range: — **Units:** — **Default:** —

Syntax: —

CODE EXAMPLES — **RELATED** —

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
PM	Position Maintenance enable/disable	Encoder Flag	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Encoder required

DESCRIPTION

Enables the position maintenance functions of an LMD 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.

Encoder functions must be enabled (**EE**=1) for position maintenance.

Param	Description
0	Position maintenance disabled (default)
1	Position maintenance enabled

The method for position maintenance will depend on the setting of the **SM** (Stall Detect Mode) variable:

PM=	SM=	Position Maintenance
1	0	Position maintenance occurs provided position is within the setting of SF (Stall Factor)
	1	Position maintenance occurs regardless of SF (Stall Factor) setting

NOTES:

– Do not confuse Position maintenance with the hMT **MU** (Position Makeup) function. While similar, the method for correcting and maintaining position are different.

– Encoder functions (**EE**=1) must be enable for PM to take effect.

Range: 0/1 **Units:** — **Default:** 0

Syntax: PM=<0/1>

CODE EXAMPLES

```
PM=1          Enable position maintenance
PR PM         Return the status of position maintenance
  1             position maintenance is enabled
```

RELATED

C2 (Encoder Counter) DB (Encoder Deadband)
EE (Encoder Enable) SM (Stall Detect Mode)
SF (Stall Factor)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x005C
	0x69	1	0x06	BOOL		

Mnemonic	Function	Function Group	Access	Usage
PN	Read Part Number	Identification Variable	RO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Read only register holds the factory defined part number.

Range: — **Units:** — **Default:** —

Syntax: PR PN

CODE EXAMPLES

```
PR PN          Return the stored part number
  LMDCM571       LMD Motion Control NEMA 23 (57 mm)
```

RELATED

SN (Serial Number)
VR (Version)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	0x65	1	0x05	STRING		

Mnemonic	Function	Function Group	Access	Usage
PR	Print Text and/or Data	System Instruction	WO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes:

DESCRIPTION

Displays text and parameter value(s) to the communications host.

– Text strings are enclosed in quotation marks while parameters (variables and flags) are not.

– Text strings and parameters 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.

The receive buffer for the LMD 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** occurs and results in an *Error 20: Tried to set unknown variable or flag*.

ASCII Control Codes

ASCII control codes may be used to enhance the performance of the **PR** instruction. The ASCII code must be enclosed within quotes. For example:

PR P, " motor steps\r"

This would terminate a string requesting the axis position with a carriage return.

This table shows the most commonly used escape codes, though most ASCIII escape codes used with terminal emulators may be used.

Param	Description
;	Semicolon character suppresses the CR/LF at the end of a line.
\b	Backspace
\c	CTRL +C (software reset)
\e	ESC
\g	Bell/beep
\n	Line feed
\r	Carriage return
\t	Tab

Range:	—	Units:	—	Default:	—

Syntax: PR <var/flg/keyword>, PR "<text> ", <var/flg/keyword>

CODE EXAMPLES

```
PR P
    12345
```
Read the value of the position counter
Position is 12345

```
PR "Position = ",P
    Position = 12345
```
Read the value of the position counter with descriptive text
Position = 12345

RELATED

PF (Print Format)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
PS	Pause executing program	Program Instruction	CMD	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Pauses an executing program with normal deceleration ramp. Immediate mode instruction may be issued and will be executed while a program is paused.

The **RS** (Resume Paused Program) is used to resume the paused program.

Range: — **Units:** — **Default:** —

Syntax: PS

CODE EXAMPLES

`PS` Pause executing program

RELATED E (End Program) EX (Execute Program) PG (Program Mode) RS (Resume Paused Program)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	—	—	—	—		Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes

Mnemonic	Function	Function Group	Access	Usage
PW	PWM Mask Setting	Configuration Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: LMD Motion Module only

DESCRIPTION

The **PW** variable is used on the LMD Motion Module product only. It is not a reserved word on the LMD products and may be used as a user variable or label.

This variable is used to set the PWM current control settings of the LMD Motion Module (LMM) only. It does not apply in any function to the LMD and may be used as a label or user variable or flag for an LMD product. See "Software, Programming, and Application Notes" on page 131 of this document for parameter settings and usage.

The **PW** variable is defaulted to IMS NEMA 17 (42 mm) motors. Recommended settings for additional motor sizes offered by IMS are located in the Software, Programming, and Application Notes section of this document. A settings dialog is also available from the View Menu when the LMMxM drive type is selected in the terminal settings.

Range: See Chapter 3 on page 131 **Units:** See Chapter 3 on page 131 **Default:** —

Syntax: PW=<mask>,<period>,<sfreq>,<ctrl> | PR PW

CODE EXAMPLES See Chapter 3 on page 131.

RELATED —

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	—	—	—	—		—

Mnemonic	Function	Function Group	Access	Usage
PY	Party Mode Enable/disable	Communications Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: See LMM Note below

DESCRIPTION

The party flag must be set to 1 if the device is being used in a multidrop communication system.

– When Party Mode is enabled, each device in the system must be addressed by the host computer using the device name specified by the **DN** instruction. By default the **DN** assigned at the factory is the exclamation character (!)

– The device 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.

– The asterisk character (*) is the drive name for global commands. Commands preceded by this character will be recognized by every MCode compatible device in the system that has **DG** = 0.

After the Party Mode is enabled, send CTRL + J (^J) to activate it. Type a command with Device Name (**DN**) and use CTRL + J as the terminator.

To configure the MDrive in Party Mode, perform the following steps:

1. Connect in single mode RS-422 and initiate communication, download any programs if required.
2. Assign a device name (DN=”<A-Z, a-z or 0-9>”) i.e DN=”A”.
3. Set the party flag to 1 (PY=1).
4. Press CTRL+J to activate party mode.
5. Type in [Device Name]S and press CTRL+J (Saves the DN and Party Configuration) i.e. AS CTRL+J.
6. Remove power and label the drive with the assigned DN.
7. If the party system uses 2 wire RS-485, set Echo Mode to 2 or the drive will echo its own transmissions.
8. Repeat for each system MDrive.

Refer to the device's associated hardware manual for additional information.

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.

Value	Description
0	Disabled (default)
1	Party Mode enabled

For LMM:
The LMD Motion Module features hardware inputs for device name (address). Whenever any of these inputs is active, the LMM will automatically enable Party Mode.

Range:	0/1	Units:	—	Default:	0

Syntax: PY=<0/1>, PR PY

CODE EXAMPLES

`PY=1[Enter][CTRL+J]`	Enable party mode

`!MR 512000[CTRL+J]`
`!PR P[CTRL+J]`
`512000`
	Device ! (default) move relative 10 revolutions
	Return the position of device !
	Position is 512000 steps

RELATED

DG (Disable Global)
DN (Device Name)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
QD	Device Queued	Communications Flag	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Function is to queue drives on party lines. **QD** may be set outside of party mode, but will only take effect if **PY** (Party Mode) is enabled (**PY**=1).

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.

Range:	Units:	Default:
0/1	—	0

Syntax: <dn>QD=<0/1>, <dn>PR QD

CODE EXAMPLES

`!QD=1 [CTRL+J]`	Set device ! as queued
`^MA 0 [CTRL+J]`	Move all queued devices to absolute position 0

RELATED

DN (Device Name)

PY (Party Mode)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	—	—	—	—		—

Mnemonic	Function	Function Group	Access	Usage
R1-R4	User Integer Register	Mathematics Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

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.

These registers contain integer values only, to perform floating point calculations, use **F<1-8>** (Floating Point Registers).

Range:	Units:	Default:
-2147483647 to 2147483647	—	0

Syntax: R<1-4>=<integer>, R1=<var>, R<1-4>=R<1-4><MATH><R<1-4>, PR R1

CODE EXAMPLES

`R1=12345`	Set R1 to 12345
`PR R1` `12345`	Read the value of R1 R1=12345
`R1=R2+R3`	Set R1 to the sum of R2+R3
`CL Q2, R1<25`	Call subroutine Q2 if R1 is less than 25

RELATED

F<1-8> (User Floating Point Registers)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x65	1	R1: 0x06 R2: 0x07 R3: 0x08 R4: 0x09	DINT		R1: 0x005F - 0x0060 R2: 0x0061 - 0x0062 R3: 0x0063 - 0x0064 R4: 0x0065 - 0x0066

Mnemonic	Function	Function Group	Access	Usage
RA	Set Radians or degrees	Configuration Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+

DESCRIPTION

Selects the Radians or Degrees as the units for trigonometric calculations. When used as a keyword with the **PR** (Print) statement it will display the setting on the terminal.

Value	Description
0	Degrees
1	Radians (default - faster)

Range: 0/1 **Units:** — **Default:** 1

Syntax: RA=<0/1> | PR RA

CODE EXAMPLES

```
RA=0          Calculate trigonometric functions in degrees
PR RA         Read the units for trig functions
  0             trig functions calculate in degrees
```

RELATED

F1-F8 (Floating Point)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
RC	Motor Running Current	Motion variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Defines the motor run current as a percentage value from 1 to 100%. The transition from **RC** (Run Current) to **HC** (Hold Current) is impacted by two other commands: **HT** (Hold Current Delay) and **MT** (Motor Settling Delay Time). These two variables are additive, with the sum being the total time to transition from the **RC** (Run Current) level to the specified standstill current.

NOTES:
For LMD products the current is only given in a percentage range as the driver is already sized and tuned to the integrated motor.

The LMD Motion Module is a 1.5A RMS standalone integrated driver/controller. The actual drive output current is derived thus: **RC**=75 results in a run current level of 1.12 A - 1.5A * 0.75 = 1.12A.

Range: 1 to 100 **Units:** Percent (%) **Default:** 25%

Syntax: RC=<percent>, PR RC

CODE EXAMPLES

```
RC=75         Set RC (Run Current) to 75%
PR RC         Read the value of the holding current
```

RELATED

MT (Motor Settling Delay Time RC (Run Current)
HT (Hold Current Delay time)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0067
	0x66	1	0x0D	USINT		

Mnemonic	Function	Function Group	Access	Usage
RD	Reverse Direction	Motion Variable	RW	Program/Immediate

Compatibility: ■ LMD(O) ■ LMD(C) ■ LMD(A) ■ LMM **Notes**: —

DESCRIPTION

This variable, when TRUE will reverse the default +/- motor direction reference. An RD command issued when the axis is in motion will result in an *Error 95: Not allowed to change Rotation Direction (RD) while in motion.*

Value	Description
0	Default +/- direction (default)
1	Direction reversed

Range: 0/1 **Units:** — **Default:** 0

Syntax: RD=<0/1>, PR RD

CODE EXAMPLES

```
RD=1          Reverse axis direction reference

PR RD         Read the value of RD
  1             RD is true, the +/- direction is reversed
```

RELATED —

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP
	0x66	1	0x13	BOOL	Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes

Use of the RD command in LMD Motion product or Ethernet (Closed Loop models) with firmware versions 5.007 or earlier may, under certain conditions, result in unintended motion.

⚠CAUTION

UNINTENDED MOTION

- Upgrade the device firmware to 5.009 or greater.

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

Mnemonic	Function	Function Group	Access	Usage
RP	Referenced Position	Motion Variable	RO	Program/Immediate

Compatibility: ■ LMD(O) ■ LMD(C) ■ LMD(A) ■ LMM **Notes**: —

DESCRIPTION

This status variable indicates whether or not the device has been homed.

Value	Description
0	Device has not been homed.
1	Device has been homed

Range: 0/1 **Units:** — **Default:** 0

Syntax: PR RP, CL Q1, RP=0

CODE EXAMPLES

```
PR RP         Read the homing status
1               The axis has been homed

CL Q1, RP=0   Call homing subroutine if axis has not been homed.
```

RELATED

HF (Home to Offset)
HI (Home to Index Mark)
HM (Home to Home Switch)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP
	0x68	1	0x09	DINT	—

Mnemonic	Function	Function Group	Access	Usage
RS	Resume Program Execution	Program Instruction	CMD	Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Resumes an program that has been paused using the **PS** (Pause Program Execution) command.

If the pause was issued during a move, the move will restart with the configured acceleration profile.

Range:	Units:	Default:
—	—	—

Syntax: RS

CODE EXAMPLES

`RS` Resume paused program

RELATED

E (End Program)
EX (Execute Program)
PG (Program Mode)
PS (Pause Program Execution)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
RT	Return From Subroutine	Program Instruction	CMD	Program

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

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 or by a trip subroutine. When used, it will return to the program address immediately following the instruction which executed the subroutine.

Range:	Units:	Default:
—	—	—

Syntax: RT

CODE EXAMPLES

`RT` Return from Subroutine

RELATED

CL (Call Subroutine)
OE (On Error Handler)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
S	Save Programs and Parameters	System Instruction	WO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

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.

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.

NOTE: Sending or requesting data during a save could corrupt communications. Use of the **S** command during a move (**MA** or **MR**) may generate an *Error 73: Tried to SAVE/RTFD/PG while Moving*, and the save may not occur.

TIP: Programs may be automatically saved on load by adding an **S** after the final **PG**. The line following the **S** should have a comment line to guarantee the <CR/LF> after the save.

Range:	Units:	Default:
—	—	—

Syntax: S

CODE EXAMPLES

`S`	Save all variable data, flag states and programs to NVM

`E`
`PG`
`S`
`'keep this line`
	Final lines of a program to save on program download

RELATED —

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0076
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
SA	Set/Read Step Angle	Motion Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: LMM only

DESCRIPTION

Step angle is a floating point variable to configure the step angle of the motor for the LMD Motion Module only.

The setting is represented by the equation: Motorcounts = **MS** * (360/SA)

Example:
MS = 256
SA = 0.9
Motorcounts = 256 * (360 / 0.9) = 102400

Common step angles for Hybrid stepper motors are shown in the table below.

Angle	Steps/rev
0.45	800
0.72	500
0.9	400
1.8	200
1.875	192
2	180
2.5	144
3.6	100
5	72

Range: See Table above **Units:** Degrees **Default:** 1.8

Syntax: SA=<angle>, PR SA

CODE EXAMPLES

```
SA=0.9          Set step angle for 0.9 degree motor.
PR SA           Display the step angle setting
  0.900000        The step angle is 0.9 degrees.
```

RELATED

MS (Microstep Resolution)
PW (Motor PWM Settings)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
SC	System Configuration Test	System Instruction	WO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: LMD Closed loop only.

DESCRIPTION

Tests the encoder direction and resolution by moving the motor shaft 1/2 revolution (180 degrees).

Ensure the shaft is disconnected from load and free to move unhindered prior to running this test.

A misconfigured encoder will return one of the following errors:

Error 100: Config Test Done - Encoder Res Mismatch
Error 101: Config Test Done - Encoder Dir Wrong
Error 102: Config Test Done - Encoder Res + Dir Wrong
Error 103: Config NOT Done - Drive not enabled.

Range: — **Units:** — **Default:** —

Syntax: SC 1

CODE EXAMPLES

```
SC 1        Start configuration test
```

RELATED —

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x00A1
	0x6A	1	0x0D	USINT		

Mnemonic	Function	Function Group	Access	Usage
SF	Set/Read Stall Factor	Encoder Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes:
LMD Closed loop
LMM with encoder

DESCRIPTION

If the encoder is enabled (**EE** = 1) and encoder position differs from the commanded position by more than the specified factor, a motor stall error is asserted. If **SM** (Stall Detect Mode) 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. Motion will attempt to continue. If the attempted velocity is above the speed at which the drive can re-sync, the motion will be unsuccessful and the position of the device will be inaccurate. An **ST** will return an *Error 86: Stall Detected* on stall.

Range: 0 to 65000 **Units:** Encoder counts **Default:** 15

Syntax: SF=<counts>, PR SF

CODE EXAMPLES

`SF=20`	Set the stall Factor to 20 encoder counts
`PR SF` `20`	Read the value of the Stall Factor The stall factor is 20 counts

RELATED

EE (Encoder Enable)
PM (Position Maintenance)
SM (Stall Detect Mode)
ST (Stall Flag)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x69	1	0x07	UINT		0x0077

Mnemonic	Function	Function Group	Access	Usage
SL	Slew Axis at Velocity	Motion Instruction	WO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Slews the axis at the commanded velocity in steps per second. The axis will accelerate at the rate specified by the **A** (Acceleration) variable.

NOTE: The maximum slew velocity is independent of the maximum velocity specified by the **VM** variable. 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.

Range:		Units:		Default:
(EE=0)	±2560000	(EE=0)	Microsteps	—
(EE=1)	±200000	(EE=1)	Encoder Counts	

Syntax: SL <velocity.

CODE EXAMPLES

`SL 20000`	Slew axis at 20000 steps /sec
`PR V` `20000`	Return the axis velocity the axis is moving at 20000 steps/sec.

RELATED

MA (Move Absolute)
MR (Move Relative)
VI (Initial Velocity)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	
	0x66	1	0x0E	DINT		0x0078 - 0x0079

Mnemonic	Function	Function Group	Access	Usage
SM	Set/Read Stall Detect Mode	Encoder Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes:
LMD Closed loop
LMM with encoder

DESCRIPTION

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. If the attempted velocity is above the speed at which the drive can re-sync, the motion will be unsuccessful. In either case **ST** (Stall Flag) will be set.

The functionality of **SM** when used with Position Maintenance (**PM**) is listed below:

Param	Description
0	Motion stops on stall detect (default)
1	Motion will attempt to continue

The method for position maintenance will depend on the setting of the **SM** (Stall Detect Mode) variable:

PM=	SM=	Position maintenance
1	0	Position maintenance occurs provided position is within the setting of SF (Stall Factor)
	1	Position maintenance occurs regardless of SF (Stall Factor) setting

Range: 00/1 **Units:** — **Default:** 0

Syntax: SM=<0/1>, PR SM

CODE EXAMPLES

`SM=1`	Set stall detection mode to mode 1
`PR SM` `1`	Return the stall mode setting Stall detection is in mode 1

RELATED

EE (Encoder Enable) PM (Position Maintenance)
SM (Stall Detect Mode) ST (Stall Flag)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x007A
	0x69	1	0x08	BOOL		

Mnemonic	Function	Function Group	Access	Usage
SN	Read Serial Number	Keyword	RO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Allows user to read the device serial number using the PR (Print) statement.

Range: — **Units:** — **Default:** —

Syntax: PR SN

CODE EXAMPLES

`PR SN`	Return the serial number

RELATED

PN (Part Number) VR (Version)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	0x65	1	0x0A	STRING		

Mnemonic	Function	Function Group	Access	Usage
ST	Stall Flag	Encoder Flag	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes:
LMD Closed loop
LMM with encoder

DESCRIPTION

The stall flag is set active (1) when the motor stalls. An *Error 86: Stall Detected* will also be asserted.

NOTES:

– The Stall Flag must be manually reset to 0 (**ST=**0).
– Clearing the error state will not clear the **SF** flag.
– The product will respond to motion commands while the **ST** flag is active.
– A subroutine triggered by the **OE** (On Error) instruction containing Encoder functions must have the encoder enabled (**EE**=1).

Param	Description
0	Axis is not stalled
1	Axis is stalled

Range: 0/1 **Units:** — **Default:** —

Syntax: ST=<0/1>, PR ST

CODE EXAMPLES

```
ST=0          Clear the state of the stall flag
PR ST         Read the value of the stall flag
0               no stall condition exists
```

RELATED

EE (Encoder Enable) OE (On Error)
SF (Stall Factor) SM (Stall Mode)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x007B
	0x69	1	0x09	BOOL		

Mnemonic	Function	Function Group	Access	Usage
SU	Execute on Startup	Factory Label	CMD	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

A program with an **LB SU** label will automatically execute from the location of the label in the program on power-up or if CTRL+C is received.

Range: — **Units:** — **Default:** —

Syntax: LB SU

CODE EXAMPLES

```
LB SU         Label program to execute on startup
```

RELATED —

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Programs labeled with the SU label will execute on system power application or software reset. Depending on the program structure this could result in immediate motion on power application or system restart.

⚠ DANGER

UNINTENDED CONSEQUENCES OF EQUIPMENT OPERATION

- Only use the SU label in instances or applications where operation does not represent a hazard to personnel or equipment.

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

Mnemonic	Function	Function Group	Access	Usage
TA	Trip on hMT Status	Trip Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Executes a subroutine address or label on the trip.

The trip can be set to occur on one or any combination of the following conditions: calibration done, hMT active, locked rotor, or lead/lag limit reached.

NOTE: The conditions are additive, e.g., **TA**=3 will trip on calibration complete and/or hybrid active status but, would not trip on locked rotor, lag limit or lead limit.

Param	Description
0	Off
1	Calibration done
2	hMTechnology active
4	Locked rotor
8	Lag limit reached
16	Lead limit reached

Range: 0 - 31 **Units:** — **Default:** 0

Syntax: TA=<label/address>,<0-31>

CODE EXAMPLES

`TA=4,k6` execute subroutine k6 when there is a locked rotor condition

`TA=12,b3` execute subroutine b3 when lag limited has been reached and there is a locked rotor

RELATED

TE (Trip Enable)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP
	—	—	—	—	—

Mnemonic	Function	Function Group	Access	Usage
TC	Trip Capture	Trip Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Sets the capture input event (trip) for input 1. When input 1 goes active, the subroutine programmed in the **TC** variable will execute.

NOTE: The **TE** (Trip Enable/Disable TC) command 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 (**RT**) to exit the subroutine. Use of a BRANCH will cause stack errors.

Range: — **Units:** — **Default:** —

Syntax: TC=<label/address>

CODE EXAMPLES

`TC=K1` Run subroutine K1 on capture

RELATED

IS (Input Setup)

TE (Trip Enable)

PC (Position Capture)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP
	0x68	1	0x06	STRING	—

Mnemonic	Function	Function Group	Access	Usage
TD	Read/Set Torque Direction	hMT Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Sets torque direction to + or –

Param	Description
0	Minus (CCW facing shaft)
1	Plus (CW facing shaft) (default)

Range: 0/1 **Units:** — **Default:** 1

Syntax: TD=<0/1>, PR TD

CODE EXAMPLES

```
TD=0          Switch torque direction to minus
PR TD         Display the torque direction
  1             Torque direction is plus
```

RELATED

AS (hMT Mode)
TQ (Torque)
TS (Torque Speed)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x00A5
	0x6A	1	0x0E	BOOL		

Mnemonic	Function	Function Group	Access	Usage
TE	Set/Read Trip Enable	Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Not all trip functions are available with all products

DESCRIPTION

Sets an event (trip) for one or any combination of the following conditions: input enabled, position enables, capture enabled, time enabled, relative position, hMT status, or main power loss.

NOTES:

– A trip must be defined prior to being enabled. Enabling an undefined trip will throw an *Error 27: Trip not defined*.

– Trip functions may be combined by adding trip numbers. For example: **TE**=3 will trip on input or on position, **TE**=127 enables all trips.

– When multiple trips are used, only the activated trip functions need to be re-enabled. The other trips remain enabled.

Param	Description	Compatibility
0	Disabled (default)	All
1	Trip on input enabled	All
2	Trip on position enabled	All
4	Trip on capture enabled	All NEMA 23 and 34
8	Trip on time enabled	All
16	Trip on relative position	All
32	Trip on hMTechnology status	LMD Closed Loop only
64	Trip on main power loss	All

Range: 0 - 127 **Units:** — **Default:** 0 (disabled)

Syntax: TE=<param> | PR TE

CODE EXAMPLES

```
TE=127        Enable all trip functions
PR TE         Return enabled trips
  127           All trips are enabled
```

RELATED

I1-I4 (Read Inputs 1 - 4) IS (Input Setup)
TA (Trip on hMT Status) TC (Trip Capture) TI (Trip on Input)
TM (Trip on Main Power) TP (Trip on Position) TT (Trip on Time)
TR (Trip on Relative Position)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x007D
	0x68	1	0x05	USINT		

Mnemonic	Function	Function Group	Access	Usage
TI	Trip on Input	Trip Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

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.

NOTES:

– The Trip subroutine must use a RETURN (**RT**) to exit the subroutine, use of a BRANCH will cause stack errors.

– The **TE** is reset when a Trip occurs. **TE** must be re-enabled prior to the next Trip if it is to be repeated.

Range: —

Units: —

Default: —

Syntax: TI <input>,<label/address>

CODE EXAMPLES

```
TI 1,Q1
TE=1
```

Set trip to execute Q1 when input 1 is active
Enable trip on input

RELATED

I1-I4 (Read Inputs 1 - 4) | IS (Input Setup) | TA (Trip on hMT Status) | TC (Trip Capture) | TI (Trip on Input) | TM (Trip on Main Power) | TP (Trip on Position) | TT (Trip on Time) | TR (Trip on Relative Position)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
TM	Trip on Main Power Loss	Trip Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Sets up an event (trip) to run a subroutine if main power is lost. In order for this to be used the auxiliary power supply must be powered and connected.

NOTES:

– The **TE** (Trip Enable which Enables/Disables TP) is reset when a Trip occurs. **TE** must be re-enabled prior to the next Trip if it is to be repeated.

– The Trip subroutine must use a RETURN (**RT**) to exit the subroutine, use of a BRANCH will cause stack errors.

– Trips should be set BEFORE motion commands in the program.

Range: —

Units: —

Default: —

Syntax: TM=<label/address>

CODE EXAMPLES

```
TM=Q1
TE=64
```

Execute Q1 on loss of main power
(Re)enable trip

RELATED

TE (Trip Enable)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
TP	Trip on Position	Trip Variable	RW	Program/Immediate
Compatibility: LMD(O) LMD(C) LMD(A) LMM		**Notes**: —		

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.

NOTES:

– 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 (**RT**) to exit the subroutine, use of a BRANCH will cause stack errors.

– Trips should be set BEFORE motion commands in the program.

– Only a single position trip type may be used at a time. **TR** cannot be used simultaneously with **TP**.

Range:	Units:	Default:
—	—	—

Syntax: TP=<position>,<label/address>

CODE EXAMPLES

```
TP=51200,Q1     Set trip to trigger Q1 at 51200 steps
TE=2            (Re)enable trip
```

RELATED

P (Position Counter)
TE (Trip Enable)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
TQ	Read/Set Torque	hMT Variable	RW	Program/Immediate
Compatibility: LMD(O) LMD(C) LMD(A) LMM		**Notes**: —		

DESCRIPTION

Sets the maximum output torque of the motor to a percentage. The parameter is only used with hMT (AS=3).

Range:	Units:	Default:
1 — 100	% (percent)	25

Syntax: TQ=<percent> | PR TQ

CODE EXAMPLES

```
TQ=50           Set the Torque to 50%

PR TQ           Read the value of TQ
  50              The torque is set to 50%
```

RELATED

AS (hMT Setting)
TD (Torque Direction)
TS (Torque Speed)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x00A6
	0x6A	1	0x0F	USINT		

Mnemonic	Function	Function Group	Access	Usage
TR	Trip on Relative Position	Trip Variables	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Sets up an event (trip) for the specified relative position.

There are three parameters for the **TR** 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, if no subroutine address or label is specified then only the High Speed Trip Output will activate.
- The third parameter specifies the number of times the trip will repeat. If 0 (default) the trip will repeat infinite times, otherwise the range is 1- 65000.

NOTES:

– 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.
Example: If TR=10000,0,25, Output 3 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).

– The Trip subroutine must use a RETURN (**RT**) to exit the subroutine, use of a BRANCH will cause stack errors.

– Trips should be set BEFORE motion commands in the program.

– Output 3 must be configured as a trip output (Os=3,28,0).

– **TR** will always use motor counts unless the encoder is enabled (**EE**=1).

– The maximum rate of trip is 20 kHz. Exceeding this may cause communications errors.

– Only a single position trip type may be used at a time. **TR** cannot be used simultaneously with **TP**.

Range:	—	Units:	—	Default:	—

Syntax: TR=<position>,<label/address>, <repeat>

CODE EXAMPLES

```
TR=51200,Q1,15   Set TR to Trip every revolution for 15 revolution
TE=16            (Re)enable Trip
```

RELATED

TE (Trip Enable) TP (Trip on Position)
OS (Output Setup) CW (Clock Width)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	0x64	1	0x04	USINT		

Mnemonic	Function	Function Group	Access	Usage
TS	Set/Read Torque Speed	hMT Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Defines the system speed for Torque mode (**AS**=3). This configuration variable will only take effect if hMT is in torque mode.

NOTE: The value for **TS** may be changed while the axis is in motion. Changing velocity during a torque move may result in an *Error 106: Reached Lead Limit count* or *Error 107: Lag limit error.*

Range:	46,512 — 2560000	Units:	steps/sec	Default:	0

Syntax: TS=<steps/sec>

CODE EXAMPLES

```
TS=51200      Set torque speed to 51200 steps per second

PR TS         Read the value of TS
   51200        TS is 51200 steps/sec
```

RELATED

AS (hMTechnology Mode)
TD (Torque Direction)
TQ (Torque Percent)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x00A3 - 0x00A4
	0x6A	1	0x10	UDINT		

Mnemonic	Function	Function Group	Access	Usage
TT	Trip on time	Trip Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Sets up an event (trip) based on time.

There are two parameters for the **TT** variable:

- The first parameter is time in mSec.
- The second parameter specifies the subroutine that should be executed when the time is expired.

NOTE: The Trip subroutine must use a RETURN (**RT**) to exit the subroutine, use of a BRANCH will cause stack errors

Range:	Units:	Default:
1 to 65535	milliseconds	—

Syntax: TT=<time>,<label/address>

CODE EXAMPLES

```
TT=10000,Q1     Execute subroutine Q1 every 10 seconds
TE=8            Enable trip
```

RELATED

TE (Trip Enable)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
UG	Process Firmware Upgrade	Upgrade Instruction	WO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

When an upgrade is initiated, the upgrade command and code will be automatically entered by the Upgrader Utility in the Motion Control Interface or Motion Terminal software programs.

Once initiated, the firmware Upgrade MUST be completed.

Range:	Units:	Default:
—	—	—

Syntax: UG 2956102

CODE EXAMPLES

```
UG 2956102    Enter upgrade mode
```

RELATED

VR (Version)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
UV	Read User Variable	Keyword	RO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Keyword used with the **PR** (Print) command to read the value of all user defined variables. The keyword will return the user variables, the scope, either global or local, and the value.

The response will come in the form of [var] = [**G**lobal/**L**ocal] [value] or example **Q1** = **G** 25

Range: —

Units: —

Default: —

Syntax: PR UV

CODE EXAMPLES

```
VA Q1=25          Create user variable Q1 and set to 25
PR UV             Read user variables, scope and values
  Q1 = G 25         Q1 is a global variable with a value of 25
```

RELATED

PR (Print)

VA (Create User Variable)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
V	Read Axis Velocity	Keyword	RO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Keyword used with the **PR** (Print) command to read the current velocity of the axis. The value of **V** is signed based on the direction of motion.

NOTES:

– **V** will not return an accurate value if hMTechnology is active.

– In Torque Mode, (**AS**=3), **V** will return a zero value.

Range: —

Units: —

Default: —

Syntax: PR V
BR <label/address>,V=<value>
CL <label/address>, V=<value>

CODE EXAMPLES

```
PR V              Read the Velocity
CL Q1,V=123456    Execute subroutine Q1 when V=123456 motor steps/sec
```

RELATED

MA (Move Absolute)

MR (Move Relative)

SL (Slew at Velocity)

VI (Initial Velocity)

VM (Maximum Velocity)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0085 - 0x0086
	0x66	1	0x0F	DINT		

Mnemonic	Function	Function Group	Access	Usage
VA	Define User Variable	Keyword	CMD	Program/Immediate
Compatibility: LMD(O) LMD(C) LMD(A) LMM		**Notes**: —		

DESCRIPTION

The **VA** instruction creates a user variable with a logical name and optionally variable value.

The restrictions for this command are:

1. A variable cannot be named after an LMD MCode Instruction, Variable, Flag, or Keyword.
2. In naming user variables, the first character must be alpha, followed by 1 alpha character OR a number from 0 to 31.
3. The system is limited to a combined total of 336 variables and labels (**LB**).

Range:	—	Units:	—	Default:	—

Syntax: VA <char><char>=<value>
VA <char><0-31>=<value>

CODE EXAMPLES

```
VA Q1=25
PR Q1
   25
```

Create user variable Q1 and set to 25
Read user variable Q1
Q1 is a global variable with a value of 25

RELATED

UV (Read User Variables)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0085
	—	—	—	—		

Mnemonic	Function	Function Group	Access	Usage
VB	Read backup voltage	Variable	RO	Program/Immediate
Compatibility: LMD(A)		**Notes**: Absolute encoder models only		

DESCRIPTION

The **VB** variable holds the voltage level for the backup voltage for the Absolute Encoder. The variable is read-only and consists of two parameters:

- Internal backup voltage
- External battery pack voltage

Range:	—	Units:	—	Default:	—

Syntax: PR VB

CODE EXAMPLES

```
PR Vb
   5.123, 5.1
```

Read user variable Q1
Internal backup level at 5.123V, external battery at 5.1V

RELATED —

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	0x69	1	0x0A	STRING		

Mnemonic	Function	Function Group	Access	Usage
VC	Read Velocity Changing	Status Flag	RO	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

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.

An output may be set to be ON when **VC** is active using **OS**=<output>,20,<active>.

Param	Description
0	**V** is not changing
1	**V** is changing

Range: 0/1 **Units:** — **Default:** 0

Syntax: PR VC | {BR/CL} <label/address>,VC=<state>

CODE EXAMPLES

```
PR VC
0
```
Read the state of the velocity changing flag
velocity is constant

```
CL Q1,VC=1
```
Call subroutine Q1 when the axis velocity is changing

RELATED

OS (Output Setup)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0088
	0x66	1	0x10	BOOL		

Mnemonic	Function	Function Group	Access	Usage
VF	Read/Set hMT Velocity Filter	hMT Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

VF takes a value of 0 to 1000. It can be defined as 0 = no filtering and 1000 = most filtering.

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.

Range: 0 to 1000 **Units:** counts **Default:** 0

Syntax: VF=<counts> | PR VF

CODE EXAMPLES

```
VF=500
```
Set the torque velocity filter to 500 counts

```
PR VF
500
```
Read the torque velocity filter
the torque velocity filter is 500 counts

RELATED

AS (hMTecnology Mode)
TQ (Torque Percent)
TS (Torque Speed)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x00A7
	0x6A	1	0x11	UINT		

Mnemonic	Function	Function Group	Access	Usage
VI	Set/Read Initial Velocity	Motion Variable	RW	Program/Immediate

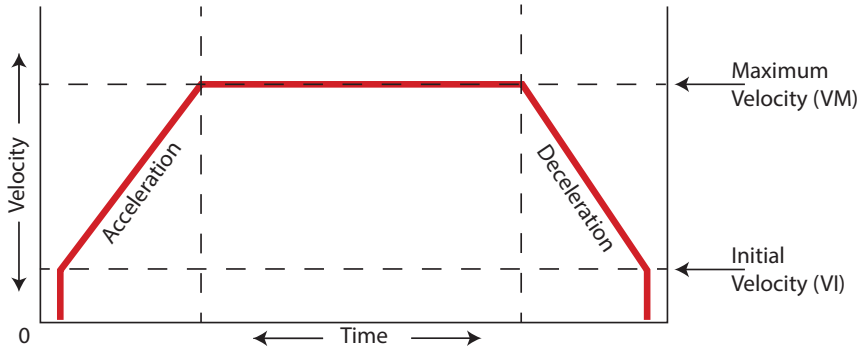
Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

Initial velocity for all motion commands. The factory default value is 1000 clock pulses (steps) per second. The graphic below demonstrates the Motion Control Profile for velocity.

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).

When **EE** is changed, **A**, **D**, **VI** and **VM** are recalculated.



Range:	1 to (VM –1)	**Units:**	EE=0	steps/sec	**Default:**	EE=0	1000
			EE=1	counts/sec		EE=1	40

Syntax: VI=<velocity> | PR VI

CODE EXAMPLES

`VI=5000`	Set the initial velocity to 5000
`PR VI` `5000`	Read the value of the initial velocity The initial velocity is 5000 steps/sec

RELATED

A (Acceleration)
D (Deceleration)
VM (Max Velocity)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x0089 - 0x008A
	0x66	1	0x11	UDINT		

Mnemonic	Function	Function Group	Access	Usage
VM	Read/Set Maximum Velocity	Motion Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

The **VM** variable specifies the maximum velocity in steps/counts per second that the axis will reach during a move command. When **EE** is changed, **A**, **D**, **VI** and **VM** are recalculated.

NOTES:

– The maximum setting of **VM** is dependent on the setting of the Microstep Resolution and is equal to **MS***10000.

– **VM** must be greater than **VI**.

– Changes to **VM** made during motion will not take effect until the current move completes.

Range:	(VI + 1) to (MS*10000)	**Units:**	**EE=0**	steps/sec	**Default:**	**EE=0**	768000
			EE=1	counts/sec		**EE=1**	60000

Syntax: VM=<velocity> | PR VM

CODE EXAMPLES

`VM=500000`	Set maximum velocity to 500000 steps sec.
`PR VM` `500000`	Read the value of VM VM is set to 500k steps/sec

RELATED

A (Acceleration)
D (Deceleration)
VI (Initial Velocity)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x008B - 00x008C
	0x66	1	0x12	UDINT		

Mnemonic	Function	Function Group	Access	Usage
VR	Read Firmware/Hardware Version	Identification Keyword	RO	Program/Immediate

Compatibility: ■ LMD(O) ■ LMD(C) ■ LMD(A) ■ LMM

Notes: —

DESCRIPTION

Keyword used with **PR** (Print) to read the firmware and hardware versions of the core code.

The keyword will return two values, the first is the device µController firmware (field upgradable), the second is the FPGA hardware version (factory upgrade only).

Range: —

Units: —

Default: —

Syntax: PR VR

CODE EXAMPLES

```
PR VR
   LMMCM 6.002, Hw: 3.2
```

Read the device version
Firmware version and hardware version

RELATED

UG (Upgrade)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	—
	0x65	1	0x0B	STRING		

Mnemonic	Function	Function Group	Access	Usage
VT	Read Voltage	Status Keyword	RO	Program/Immediate

Compatibility: ■ LMD(O) ■ LMD(C) ■ LMD(A) ■ LMM

Notes: —

DESCRIPTION

The **VT** keyword is used in conjunction with the **PR** (Print) instruction to read the status and voltage of the device.

Using this keyword will return a status followed by a voltage level.

Status	Aux V	+VDC	Notes
0	In range	In range	Normal for LMD with Auxiliary voltage connected
1	Out of range/ Unused	In range	Normal for LMM or LMD without Auxiliary voltage connected
2	In range	Out of range	Abnormal condition, Results in an *Error 78: Aux V out of range (too high or too low)*
3	Out of range	Out of range	Abnormal condition, Results in an *Error 79: Plus V out of range (too high or too low)*

An optional parameter may be used to read the voltage and status of a specific voltage:

- Auxiliary Voltage (LMD products only)
- +VDC

Param	Description
<blank>	Read both sensors, bridge first, then µController
1	Read the device status
2	Read the Aux V level
3	Read the +V level

Range: —

Units: —

Default: —

Syntax: PR VT, <param>

CODE EXAMPLES

LMM

```
PR VT
   1, 23
```

Read the status and voltage
+V in range, 23 VDC

LMD

```
PR VT
   0,23,36
```

Read the status and voltage
Aux V and +V in range. Aux V: 23 VDC, +V: 36 VDC

```
PR VT, 3
   0,36
```

Read the voltage
+V in range. +V: 36 VDC

RELATED

IT (Internal Temperature)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	Refer to the Modbus/TCP User Manual for Mfg Specific Function Codes
	0x65	1	0x0C	STRING		

Mnemonic	Function	Function Group	Access	Usage
WT	Set/Read Warning Temperature	System Variable	RW	Program/Immediate

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

The Warning Temperature variable allows the user to set a threshold temperature at which the device will assert an *Error 71: Internal Temperature Warning* to the terminal screen if the set temperature is exceeded.

NOTE: This single setting will set the warning level for both temperature sensors. Either sensor reaching the set threshold will result in the error code.

Range:	0-84	**Units:**	ºC	**Default:**	80

Syntax: WT=<temperature> | PR WT

CODE EXAMPLES

`WT=75`	Set warning temperature threshold to 75 ºC
`PR WT` `75`	Read the warning temperature setting WT is set to 75 ºC

RELATED

IT (Internal Temperature)

NETWORK PROTOCOL EQUIVALENTS

Ethernet/IP	Class	Instance	Attribute	Data Type	Modbus/TCP	0x008D
	0x64	1	0x05	USINT		

Math, Logic, and Trigonometric Operators

NOTE:
Firmware versions prior to Firmware 6.001+ do not support advanced floating point math and trigonometric functions.
Math functions execute from left to right and not by standard order of operation.

Symbol	Function	Function Group
+	Addition	Basic Math

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

Adds the contents of variables.

Syntax: <sum target>=<augend>+<addend>+...

CODE EXAMPLES

```
VA Q1=25
VA Q2=30
VA Q3=40
```
Setup sample user variables and assign value

```
R1=Q1+Q2+Q3
PR R1
    95
```
Add Q1, Q2 and Q3 together, store sum in Register 1
Read the Value of R1
R1 is 95

Symbol	Function	Function Group
–	Subtraction	Basic Math

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

Subtracts the contents of two variables.

Syntax: <difference target>=<menuend>–<subtrahend>

CODE EXAMPLES

```
VA Q1=25
VA Q2=30
```
Setup sample user variables and assign value

```
R1=Q2-Q1
PR R1
    5
```
Subtract Q1 from Q2, store difference in Register 1
Read the Value of R1
R1 is 5

Symbol	Function	Function Group
*	Multiplication	Basic Math

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

Multiplies the contents of two variables.

Syntax: <product target>=<multiplicand>*<multiplier>

CODE EXAMPLES

```
VA Q1=25
VA Q2=30
```
Setup sample user variables and assign value

```
R1=Q1*Q2
PR R1
    750
```
Multiply Q1 and Q2, store product in Register 1
Read the Value of R1
R1 is 750

Symbol	Function	Function Group
/	Division	Basic Math

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

Divides the contents of one variable with another variable.

NOTE: When dividing integer values that require a more precise quotient, the quotient may be stored in **F1-F8** (Floating Point Registers).

Syntax: <quotient target>=<dividend>/<divisor>

CODE EXAMPLES

```
VA Q1=25
VA Q2=30
```
Setup sample user variables and assign value

```
F1=Q2/Q1
PR F1
    1.200000
```
Divide Q2 by Q1, store quotient in Floating Point Register 1
Read the Value of F1
F1 is 1.200000

Symbol	Function	Function Group
=	Equal	Comparison operator

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

Set a variable equal to another variable or number, comparison operator for **BR** (Branch) and **CL** (Call Subroutine) program operations.

Syntax: <target var>**=**<source var>| [BR/CL] <label/address>,<var/flg/io>=<var/flg/num >

CODE EXAMPLES

`VA Q1=25`	Setup sample user variables and assign value
`A=Q1`	Set acceleration equal to user variable Q1
`CL X1,I1=1`	Call subroutine X1 when input is active

Symbol	Function	Function Group
<>	Not Equal	Comparison operator

Compatibility: LMD(O) LMD(C) LMD(A) LMM **Notes**: —

DESCRIPTION

Compare two variables and execute stated action when the values are not equal.

Syntax: [BR/CL] <label/address>,<var/flg/io>**<>**<var/flg/num>

CODE EXAMPLES

`CL X1,Q1<>25`	Call subroutine when user variable Q1 is not equal to 25

Symbol	Function	Function Group
<	Less Than	Comparison operator
Compatibility: LMD(O) LMD(C) LMD(A) LMM	**Notes**: —	
DESCRIPTION Tests if Variable is less than a second variable.		
Syntax: [BR/CL] <label/address>,<var/flg/io><<var/flg/num>		
CODE EXAMPLES: `CL X1,Q1<=25` Call subroutine when user variable Q1 is less than 25		

Symbol	Function	Function Group
<=	Less Than or Equal	Comparison operator
Compatibility: LMD(O) LMD(C) LMD(A) LMM	**Notes**: —	
DESCRIPTION Tests if Variable is less than or equal to a second variable		
Syntax: [BR/CL] <label/address>,<var/flg/io>**<=**<var/flg/num>		
CODE EXAMPLES `CL X1,Q1<=25` Call subroutine when user variable Q1 is less than or equal to 25		

Symbol	Function	Function Group
>	Greater Than	Comparison operator
Compatibility: LMD(O) LMD(C) LMD(A) LMM	**Notes**: —	
DESCRIPTION Tests if Variable is greater than to a second variable		
Syntax: [BR/CL] <label/address>,<var/flg/io>**>**<var/flg/num>		
CODE EXAMPLES `CL X1,Q1>25` Call subroutine when user variable Q1 is greater than 25		

Symbol	Function	Function Group
>=	Greater Than or Equal	Comparison operator
Compatibility: LMD(O) LMD(C) LMD(A) LMM	**Notes**: —	
DESCRIPTION Tests if Variable is greater than or equal to a second variable.		
Syntax: [BR/CL] <label/address>,<var/flg/io>**>=**<var/flg/num>		
CODE EXAMPLES `CL X1,Q1>=25` Call subroutine when user variable Q1 is greater than or equal to 25		

Symbol	Function	Function Group
&	AND	Logic operator
Compatibility: LMD(O) LMD(C) LMD(A) LMM	**Notes**: —	

DESCRIPTION

Performs a Logic AND operation on two variables.

Syntax: <target var>=<var/flg>**&**<var/flg/num>

CODE EXAMPLES

```
R1=25
R2=30
```
Assign value to user registers

```
R3=R1&R2
```
AND R1 and R2 together, store in R3

```
PR R3
   24
```
Read the value of R3
R3 is 24

Symbol	Function	Function Group
|	OR	Logic operator
Compatibility: LMD(O) LMD(C) LMD(A) LMM	**Notes**: —	

DESCRIPTION

Performs a Logic OR operation between two variables.

Syntax: <target var>=<var/flg>**|**<var/flg/num>

CODE EXAMPLES

```
R1=25
R2=30
```
Assign value to user registers

```
R3=R1|R2
```
OR R1 and R2 together, store in R3

```
PR R3
   25
```
Read the value of R3
R3 is 31

Symbol	Function	Function Group
^	XOR	Logic operator
Compatibility: LMD(O) LMD(C) LMD(A) LMM	**Notes**: —	

DESCRIPTION

Performs a Logic XOR operation between two variables.

Syntax: <target var>=<var/flg>**^**<var/flg/num>

CODE EXAMPLES

```
R1=25
R2=30
```
Assign value to user registers

```
R3=R1^R2
```
XOR R1 and R2 together, store in R3

```
PR R3
   7
```
Read the value of R3
R3 is 7

Symbol	Function	Function Group
!	NOT	Logic operator

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: —

DESCRIPTION

Performs a Logic NOT operation.

Syntax: <target var>=<var/flg>!<var/flg>

CODE EXAMPLES

```
R1=25                Assign value to user registers
R3=!R1               Evaluate the NOT value of R1. Store the value in R3
PR R3                Read the value of R3
  -26                  R3 is -26
```

To invert a boolean or flag:

```
R1=1                 Assign a value of 1 to user register
R2=!R1+2             Evaluate the NOT value of R1 and add 2. Store the result in R2
PR R2                Read the value of R2
  0                    R2 is 0
R1=0                 Assign a value of 0 to the user register
R2=!R1+2             Evaluate the NOT value of R2 and add 2. Store the result in R2
PR R2                Read the value of R2
  1                    R2 is 1
```

Symbol	Function	Function Group
AB	Absolute Value	Advanced Math/Trigonometry

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+
Calculation should be performed using the double-precision floating point registers F1-F8 (Floating Point Registers).

DESCRIPTION

Performs an Absolute on the specified register or variable.

Syntax: <target fpreg>=AB <var/num>

CODE EXAMPLES

```
MA -51200            Move negative 51200 steps (1 Rev)
PR P                 Read the position counter
  -51200               Position counter is at -51000 steps
F1=AB P              Perform absolute on position counter, store in F1
PR F1                Read the value of F1
  51200.00000          F1 is 51200.00000
```

Symbol	Function	Function Group
CS	Cosine	Advanced Math/Trigonometry

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+
Calculation should be performed using the double-precision floating point registers F1-F8 (Floating Point Registers).

DESCRIPTION

Performs cosine on the specified register.

Syntax: <target fpreg>=CS <var/flg/num>

CODE EXAMPLES

`VA Q1=51200`	Create and assign value to user register Q1
`F1=CS Q1`	Store cosine of Q1 in F1
`PR F1` `-0.106072`	Read the value of F1 F1 is -0.106072

Symbol	Function	Function Group
C_	Arc Cosine	Advanced Math/Trigonometry

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+
Calculation should be performed using the double-precision floating point registers F1-F8 (Floating Point Registers).

DESCRIPTION

Performs an arc cosine on the specified register.

Syntax: <target fpreg>=C_ <var/flg/num>

CODE EXAMPLES

`VA Q1=51200`	Create and assign value to user register Q1
`F1=CS Q1`	Store cosine of Q1 in F1
`PR F1` `-0.106072`	Read the value of F1 F1 is -0.106072
`F2=C_ F1`	Store Arc Cosine of F1 in F2
`PR F2` `1.677068`	Return the value of F2 F2 is 1.677068

Symbol	Function	Function Group
LO	Logarithm Base e	Advanced Math/Trigonometry

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+
Calculation should be performed using the double-precision floating point registers F1-F8 (Floating Point Registers).

DESCRIPTION

Performs an logarithm (base e) on the specified register.

Syntax: <target fpreg>=LO <var/flg/num>

CODE EXAMPLES

`VA Q1=51200`	Create and assign value to user register Q1
`F1=LO Q1`	Store log (base e) of Q1 in F1
`PR F1` `10.843495`	Read the value of F1 F1 is 10.843495

Symbol	Function	Function Group
L_	Logarithm Base 10	Advanced Math/Trigonometry

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+
Calculation should be performed using the double-precision floating point registers F1-F8 (Floating Point Registers).

DESCRIPTION

Performs an logarithm (base 10) on the specified register.

Syntax: <target fpreg>=L_ <var/flg/num>

CODE EXAMPLES

`VA Q1=51200`	Create and assign value to user register Q1
`F1=L_ Q1`	Store log (base 10) of Q1 in F1
`PR F1` `4.709270`	Read the value of F1 F1 is 4.7092705

Symbol	Function	Function Group
PI	PI (3.141592654)	Advanced Math/Trigonometry

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+
Calculation should be performed using the double-precision floating point registers F1-F8 (Floating Point Registers).

DESCRIPTION

Holds the value of PI.

Syntax: <target fpreg>=<reg/var><math>PI

CODE EXAMPLES

`VA Q1=51200`	Create and assign value to user register Q1
`F1=Q1*PI`	Multiply User var Q1 times PI
`PR F1` `160849.543885`	Read the value of F1 F1 is 160849.543885

Symbol	Function	Function Group
SI	Sine	Advanced Math/Trigonometry

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+
Calculation should be performed using the double-precision floating point registers F1-F8 (Floating Point Registers).

DESCRIPTION

Calculates the sine of the specified register.

Syntax: <target fpreg>=SI <var/flg/num>

CODE EXAMPLES

```
VA Q1=51200
```
Create and assign value to user register Q1

```
F1=SI Q1
```
Store Sine of Q1 in F1

```
PR F1
   -0.994358
```
Read the value of F1
F1 is -0.994358

Symbol	Function	Function Group
S_	Arc Sine	Advanced Math/Trigonometry

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+
Calculation should be performed using the double-precision floating point registers F1-F8 (Floating Point Registers).

DESCRIPTION

Calculates the arc sine of the specified register.

Syntax: <target fpreg>=S_ <var/flg/num>

CODE EXAMPLES

```
VA Q1=51200
```
Create and assign value to user register Q1

```
F1=SI Q1
```
Store Sine of Q1 in F1

```
PR F1
   -0.994358
```
Read the value of F1
F1 is -0.994358

```
F2=S_ F1
```
Store Arc Sine of F1 in F2

```
PR F2
     -1.464524
```
Return the value of F2
F2 is -1.464524

Symbol	Function	Function Group
SQ	Square Root	Advanced Math/Trigonometry

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+
Calculation should be performed using the double-precision floating point registers F1-F8 (Floating Point Registers).

DESCRIPTION

Calculates the square root of the specified register.

Syntax: <target fpreg>=SQ <var/flg/num>

CODE EXAMPLES

`VA Q1=51200`	Create and assign value to user register Q1
`F1=SQ Q1`	Store Square Root of Q1 in F1
`PR F1` `226.274170`	Read the value of F1 F1 is 226.274170

Symbol	Function	Function Group
TG	Tangent	Advanced Math/Trigonometry

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+
Calculation should be performed using the double-precision floating point registers F1-F8 (Floating Point Registers).

DESCRIPTION

Calculates the tangent of the specified register.

Syntax: <target fpreg>=TG <var/flg/num>

CODE EXAMPLES

`VA Q1=51200`	Create and assign value to user register Q1
`F1=TG Q1`	Store Tangent of Q1 in F1
`PR F1` `9.374376`	Read the value of F1 F1 is 9.374376

Symbol	Function	Function Group
T_	Arc Tangent	Advanced Math/Trigonometry

Compatibility: LMD(O) LMD(C) LMD(A) LMM

Notes: Firmware 6.001+
Calculation should be performed using the double-precision floating point registers F1-F8 (Floating Point Registers).

DESCRIPTION

Calculates the arc tangent of the specified register.

Syntax: <target fpreg>=T_ <var/flg/num>

CODE EXAMPLES

`VA Q1=51200`	Create and assign value to user register Q1
`F1=T_ Q1`	Store Tangent of Q1 in F1
`PR F1` `1.570777`	Read the value of F1 F1 is 1.570777

Chapter 3
Software, Programming, and Application Notes

What's in this Chapter?

This section will provide information on the LMD Software Suite (LSS) and LMD MCode Programming and Applications, including Party Mode communications, programming the Input/Output (I/O), and factors impacting motion commands.

This chapter includes the following topics:

LMD Software Suite (LSS)

The software associated with LMD Motion products is contained within the LMD Software Suite (LSS). This software package is available for download https://novantaims.com/dloads/

Applicable modules include:

1. **Motion Control Interface**

   - Graphic User Interface (GUI) for developing and simulating LMD MCode programs.
   - ANSI Terminal emulation with the ability for multiple terminal tabs to be open on different COM ports.
   - Program editor tabs with color coding.
   - Programmable function keys
   - Program simulator allows for quick test and debugging of LMD MCode programs.
   - For RS-422/485 and Ethernet LMD Motion product products
   - Motion Control Firmware upgrade utility.

2. **Ethernet Configuration Utility**

   - For LMD Motion product Ethernet products
   - Configure basic TCP/IP parameters such as:
     - IP address
     - Subnet mask
     - Gateway address
   - Firmware upgrades to Ethernet controller firmware

These modules are documented in the LSS Manual, available for download from:

https://novantaims.com/dloads/

Party Mode Communications

The following communication formats are used by LMD MCode compatible devices:

{} .....The contents between the {} symbols are transmitted.
{0D} ..... Hex equivalent for an ASCII CR (Carriage Return).
{0A} .....Hex equivalent for an ASCII 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 less than 100 microseconds.

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.

Response to Echo Mode - Party and Check Sum are Zero (0)				
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

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=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 CK=0	(DN) (command) (0A)	–	CET command (0D) (0A)	Queued response. The last character sent is the LF

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=0 CK=1	(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=0 CK=1	(DN) (command) (0A)	–	CET (0D) (0A)	The last character sent is LF
EM=2 & PY=0 CK=1	(DN) (command) (0A)	–	–	No response except to PR and L commands
EM=3 & PY=0 CK=1	(DN) (command) (0A)	–	CET command (0D) (0A)	Queued response. The last character sent is the LF

Response to echo mode - party and check sum are one (1)				
Parameter Setting	**Transmission**	**Initial Response**	**Final Response**	**Notes**
EM=0 & PY=1 CK=1	(DN) (command) (CS) (0A)	(command) 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 CK=1	(DN) (command) (CS) (0A)	–	CET (ACK) or (NAK)>	The last character sent is ACK or NAK
EM=2 & PY=1 CK=1	(DN) (command) (CS) (0A)	–	–	No response except to PR and L commands
EM=3 & PY=1 CK=1	(DN) (command) (CS) (0A)	–	CET command (CS) (ACK) (NAK)	Queued response. The last character sent is ACK or NAK

Using Checksum

For communication using checksum, 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 the string by typing: MRr 1(ALT key + 0144) (use the symbol of 0144 in the string by holding down the ALT key and typing 0144).

NOTE: 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.
   - If not zero, the command is ignored and NAK echoed.
   - 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.

Immediate Party Mode Sample Codes

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

Move device A, B or C 10000 steps

Assuming there are three devices set up in party mode.

- To move MDrive unit "A", press CTRL+J and then type: aMR^10000 and press CTRL+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: cVA=<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.

Programming the I/O

I/O Availability per Device Type

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

I/O Function	NEMA 17 (42mm)	NEMA 23 (57mm)	NEMA 34 (85mm)
+5 to +24 isolated input points. Programmable to multiple functions. Sink or source.	3	4	4
Analog input	1	1	1
+5 to +24 VDC isolated (power) outputs, dry contact configuration. Programmable to multiple functions	—	2	2
High-speed, low current isolated output. Programmable to multiple functions including Trip.	1	1	1

Input Functions

The following table lists the programmable input functions.

Digital Input Functions				
Function	**Description**	**Line**	**Type**	**Active**
General Purpose	General purpose input function used to control program branches, subroutine calls or bcd functions when input bank is used as a group.	1 – 4	0	0/1
Home	Homing input. Will function as specified by the home (hm) command.	1 – 4	1	0/1
Limit +	Positive limit input. Will function as specified by the limit (lm) command.	1 – 4	2	0/1
Limit –	Negative limit input. Will function as specified by the limit (LM) command.	1 – 4	3	0/1
G0	G0 input. Will run program located at address 1 on activation.	1 – 4	4	0/1
Soft Stop	Soft stop input. Stops motion with deceleration and stops program execution.	1 – 4	5	0/1
Pause	Pause/resume program with motion.	1 – 4	6	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.	1 – 4	7	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.	1 – 4	8	0/1
Reset	When set as reset input, then the action is equivalent to a ^c entered into a terminal.	1 – 4	11	0/1
Capture	Capture input will operate with the Trip Capture (TC) trip to run a subroutine when active. Only applicable to input 1. Capture function not available on the NEMA 17 (42 mm) LMD Motion product models.	1	12	0/1
Step/Direction	Step clock (IN3) and direction (IN4) inputs. Paired on inputs 3 and 4 only.	3 – 4	13	0/1
Encoder A/B	Encoder channel A (IN3) and B (IN4) inputs for following. Paired on inputs 3 and 4 only.	3 – 4	14	0/1
Step Up/Down	Step up (IN3) and down (IN4) inputs	3 – 4	15	0/1

Active HI/LO States Defined

Active HI/LO State Parameter Definition - Inputs

The Active HI/LO State parameter determines the relationship between an input's electrical activity and its state in the MCode software. Use the Active HI/LO State parameter of the IS command to change this relationship as shown in the examples below.

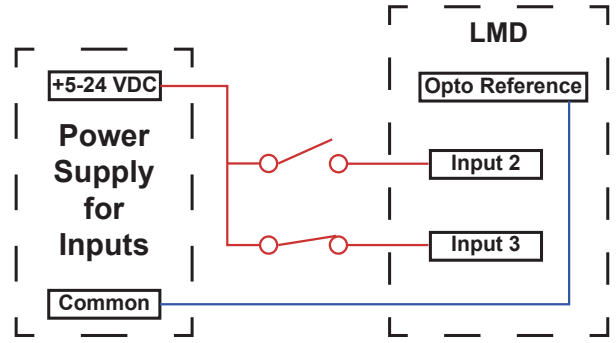
If the Active HI/LO State parameter of an input is 0, it will be high in the MCode when it is electrically active.

If the Active HI/LO State parameter of an input is 1, it will be low in the MCode when it is electrically inactive.

- **To determine whether an input is electrically energized:**
  Measure the voltage between the input in question and the Opto Reference.
  If the voltage is below about 1 volt, that input is not electrically energized.
- **To determine if that input is active in the MCode software in the drive:**
  Send PR Ix where x is the input number. The return of a one (1) means it is high in software, a return of a zero (0) means it is low in software.

Examples:

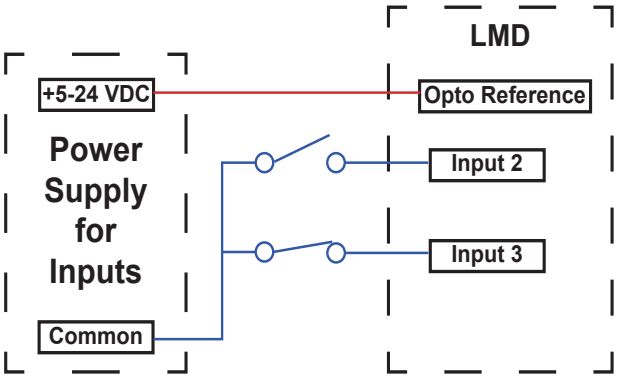
Input Wiring: Opto Reference Low:



Input Setting	Input State in Software on LMD
IS = 2,0,1* IS = 2,0,0	PR I2 response is 1 PR I2 response is 0
IS = 3,0,1* IS = 3,0,0	PR I3 response is 0 PR I3 response is 1

*Default

Input Wiring: Opto Reference High:



Input Setting	Input State in Software on LMD
IS = 2,0,1* IS = 2,0,0	PR I2 response is 1 PR I2 response is 0
IS = 3,0,1* IS = 3,0,0	PR I3 response is 0 PR I3 response is 1

*Default

Active HI/LO State Parameter Definition - Outputs

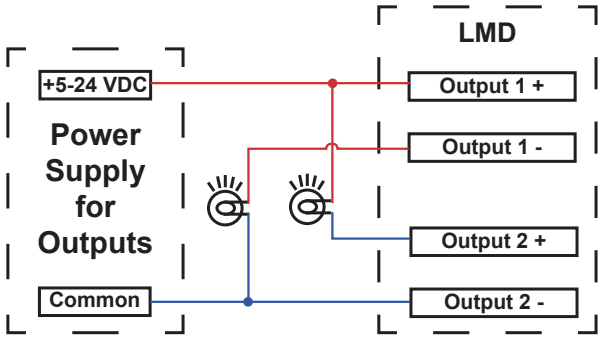
The Active HI/LO State parameter determines the relationship between an output's state in the MCode software and its electrical conductivity. Use the Active HI/LO State parameter of the OS command to change this relationship as shown in the examples below.

If the Active HI/LO State parameter of an output is 0, it will conduct when Hi (1) in the software.

If the Active HI/LO State parameter of an output is 1, it will conduct when Lo (0) in the software.

Examples:

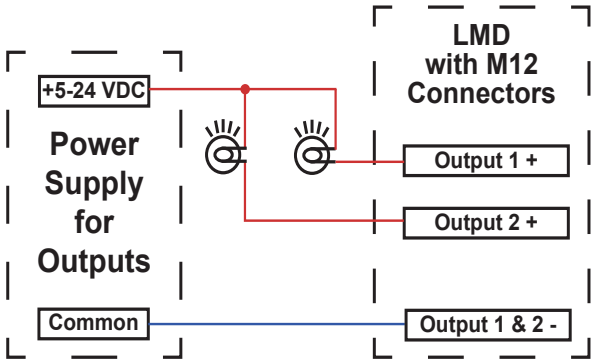
Output Wiring: IP20 Version



Output Setting	Output Conducting
OS = 1,0,0* OS = 1,0,1	Output 1 will conduct when O1=1 Output 1 will conduct when O1=0
OS = 2,0,0* OS = 2,0,1	Output 2 will conduct when O2=1 Output 2 will conduct when O2=0

*Default

Output Wiring: IP65 Version



Output Setting	Output Conducting
OS = 1,0,0* OS = 1,0,0	Output 1 will conduct when O1=1 Output 1 will conduct when O1=0
OS = 2,0,0* OS = 2,0,1	Output 2 will conduct when O2=1 Output 2 will conduct when O2=0

*Default

Digital Input Functions

The inputs may be interfaced as sourcing or sinking, but can not be mixed sourcing and sinking since the common is shared. An input may be programmed to be a general purpose user input, or to one of 11 dedicated input functions. These may then be programmed to have an active state of either high or low.

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

```
IS=<line number>,<type>,<active low/high>
```

Example:

```
IS=3,3,0          'set input 3 = limit-, active low
IS=2,0,1          'set input 2 = gen. purpose, active high
```

NOTE: The Sink/Source Function is defined by the bias of the Opto Reference input. Connecting the opto reference to a +5 to +24 VDC supply will provide for sinking inputs. Connecting the opto reference to Ground will provide for sourcing inputs.
Refer to the LMD Motion Product’s hardware manual for examples.

Digital Output Functions

The outputs may be configured as general purpose or set to dedicated functions, such as fault or moving. These outputs will sink up to 100 mA (one channel of two banks) and may be connected to an external VDC source.

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

```
OS=<line>,<type>,<active low/high>
```

Examples:

```
OS=1,17,0 'set output 3 to moving, active high
OS=3,0,0          'set output 3 to be error, active low
```

Output Functions

Output functions may be programmed to be a general purpose user output with the following functions. Shaded areas apply only to units with an internal encoder installed.

Digital Output Functions				
Function	**Description**	**Line**	**Type**	**Active**
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.	1 — 3	16	0/1
Moving	Will be in the active state when the motor is moving.	1 — 3	17	0/1
Software error	Will be in the Active State when a error occurs. .	1 — 3	18	0/1
Stall	Will be in the active state when a stall is detected. Encoder required, stall detect mode (SM) must be enabled.	1 — 3	19	0/1
Velocity Changing	Will be in the active state when the velocity is changing. Example: during acceleration and deceleration.	1 — 3	20	0/1
Locked Rotor	Will be in an active state when the rotor is locked on MDrive Hybrid products	1 — 3	21	0/1
Moving to Position	Will be active when the motor is indexing to a commanded position.	1 — 3	23	0/1
Hybrid Active	Will be active when the Hybrid control circuitry is engaged.	1 — 3	24	0/1
Make Up Active	Will be active when the Hybrid is correcting lead/lag conditions.	1 — 3	25	0/1
Trip	Trip output applies to output 3 only, active low only	3	28	0
Attention	When active, indicates a status or statuses as configured by the AO variable.	1 — 3	29	0/1

Programmable Input/Output Usage Examples

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

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

Example 1: Input Interface - Go Input

The following programming example applies when a switch connected to activate Input 3.

Code Sample:

For this code sample, the input will be set up as a G0 input, Active HI/LO State = 0. When electrically active, the input will launch the program beginning at address 1 in device memory:

```
'[VARIABLES]
Is=3,4,0 'Set Input 3 TO Be A G0 Input, ON In MCode When Electrically Active
D3=50    'Set Input 3 debounce to 50 milliseconds
'[PROGRAMS]
PG 1              'Enter Program Mode at location 1
  P=0             'Set P to zero
  MR 20000        'Move +20000 Steps Relative TO Current Pos.
  H               'Hold Program Execution Until Motion Completes
  MR -20000       'Move -20000 Steps
  H               'Hold Program Execution Until Motion Completes
  PR "  P=",P     'Print position
  E               'End Program
PG                'Exit Program Mode
S
'[END]
```

Example 2: Input Interface - Soft Stop

The following programming example applies when a switch to activate Input 1.

Code Sample:

For this code sample, this input will be set up as a Soft Stop input, Active HI/LO State = 1. When electrically inactive, the input will stop the motor:

```
Is=2,5,1    'Set Input 2 TO Soft Stop, on in MCode When electrically inactive
SL 200000   'Slew the Motor At 200000 Steps/Sec
```

When the output is electrically inactive, the motor will decelerate to a stop.

Example 3: Output Interface - Velocity Changing

The following programming example applies when a load is connected to an output point configured for Velocity Changing.

Code Sample:

For this code sample, the load will be an LED. The motor is configured so the LED will be lit while the motor is stopped or at constant velocity. Input 2 will be set up as a Soft Stop input, Active HI/LO State = 1. When electrically inactive, the input will stop the motor:

```
A = 50000
D = A
Is=2,5,1 'Set Input 2 TO Soft Stop, On In Mcode When Electrically Inactive
Os=1,20,0 'Set Output 1 Be ON When Velocity Changing
SL 200000 'Slew The Motor At 200000 Steps/Sec
```

While the motor is accelerating, the LED will be on. Once the motor reaches a constant velocity, the LED will turn off. When the Soft Stop input goes off, the motor will begin to decelerate. The LED will go on again while velocity is changing.

Example 4: Output Interface - General Purpose

The following programming example applies when a load is connected to output 1.

Code Sample:

For this code sample, the load is an LED. The output is configured to be a general purpose user output that will be turned on briefly when a range of motion is complete:

```
'[VARIABLES]
Os = 1,16,0 'Set Output 1 = General Purpose, Electrically Active When Set TO 1
'[PROGRAMS]
PG 1 'Enter Program Mode At Address 1
  MR 2000000      'Move 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 ON
  H 1500          'Hold for 1500 milliseconds
  O1 = 0          'Set Output 1 OFF
PG                'Exit Program Mode
S                 'Save to Flash
'[END]
```

Enter EX 1 to execute the program. The motion will occur, and the output will turn on.

Example 5: Reading Inputs as a Group

The inputs may be read as a group using the IN keyword. This will display as a decimal between 0 to 15, representing the 4 bit binary number. The IN keyword will function on the 42mm (NEMA 17) devices, but will only read inputs 2 - 4.

NOTE: Results of the IN keyword are not affected by debounce or setting of Active Hi/Lo State in the IS command.

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

Example 6: Interfacing Outputs as a Group

Outputs may be written to as a group using the OT keywords. This will set the outputs as a binary number displaying a decimal between 0 to 7, representing the 3 bit binary number on the 57 mm (NEMA 23) and 85 mm (NEMA 34) devices, but will have no practical use on 42 mm (NEMA 17) devices. The outputs should be configured to the general purpose user type (OS=<line>,16).

```
OT=5 'set the binary state of the combined I/O to 101
```

Analog Input Usage

The analog input is configured from the factory as a 0 to 5V, 12 bit resolution input (IS = 5,9,0). 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 4095, where 0 = 0 volts and 4095 = 5.0 volts. The analog input may also be configured as 0 to 10 vols (IS = 5,9,1) for a 4 to 20 mA (IS = 5,10,0) or 0 to 20 mA Analog Input (IS = 5,10,1). If used as a 4 to 20mA input the range is 0 to 3200 units.

Sample Usage:

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

IS=5,9,0                        'set analog to read 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
```

Factors Impacting Motion Commands

Motor Steps

All LMD MCode examples assume 200 step motors. They rotate at 1.8° per full step. 200 steps would equal 1 revolution.

Microsteps divide the 200 motor steps into smaller steps to improve smoothness and resolution of the LMD 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 MS is set to 128, one revolution of the motor would now require 25600 microsteps.

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 MS is set at 256 and programmed 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 programming a move of 51200, the motor would turn 2 full revolutions.

Control with Encoder Enabled (EE=1)

If the encoder is enabled the move commands use different values. The encoder has 1000 lines and yields 4000 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 4000.

The value of MS will remain at its previous set value if the Encoder is enabled (EE=1).

When using an LMD with Encoder Enabled, the minimum setting for MS is 50.
To determine the minimum value of MS when using an LMM, refer to EL to calculate.

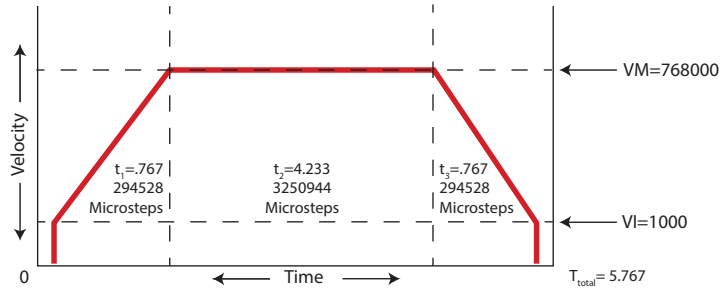
Linear Movement

If using 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. 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, it is possible to calculate the actual time it takes to move the axis the required distance. This is done with a trapezoidal profile as shown below.



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/Sec^2.
D ..... 1000000 Steps/Sec^2.
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).

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

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

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

$$= 294911 \text{ steps}$$

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.

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

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.

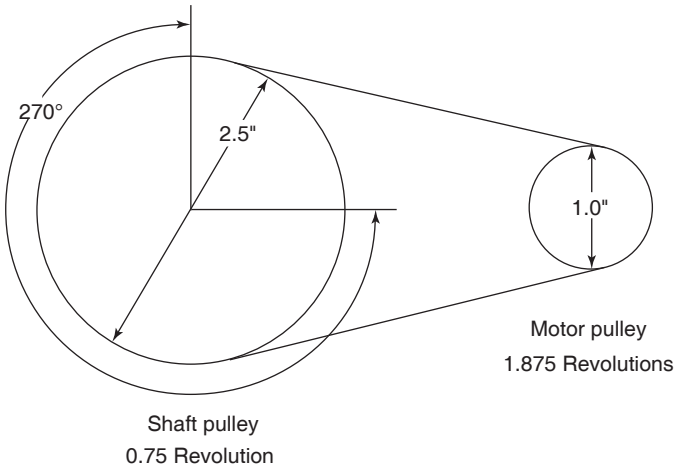
$$\frac{768000}{51200} \times 0.1 \times 60 = \mathbf{90\ IPM}$$

Calculating Rotary Movement

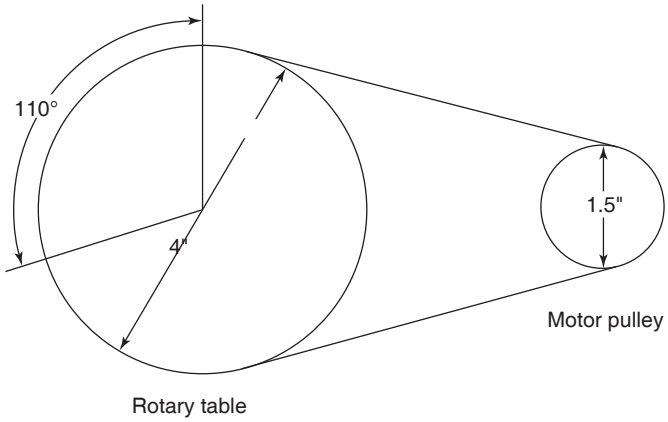
Assume that MS is set to 256 and using the motor to drive a shaft with a timing belt and pulley arrangement. As shown below, the pulley is 1” in diameter and the shaft pulley is 2.5” in diameter. The shaft must turn 270°.

- 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).

Many of the calculations can be performed in reverse to calculate motor moves to meet a required move of the device. A linear or rotational move as well as speed may be translated into an LMD MCode command.



Rotary drive example 1



Rotary drive example 2

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°.

$$110 \times 2.66 = \mathbf{293.3°}$$

If 51200 steps = 1 revolution then 1° = 142.222 steps.

$$\frac{51200}{360} = \mathbf{142.222\ steps}$$

The LMD MCode device must be programmed to move 41713 steps to rotate 293.3°.

```
142.222 steps x 293.3° = 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 VM is set 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.

$$\frac{8}{60} = 0.133333 \text{ RPS}$$

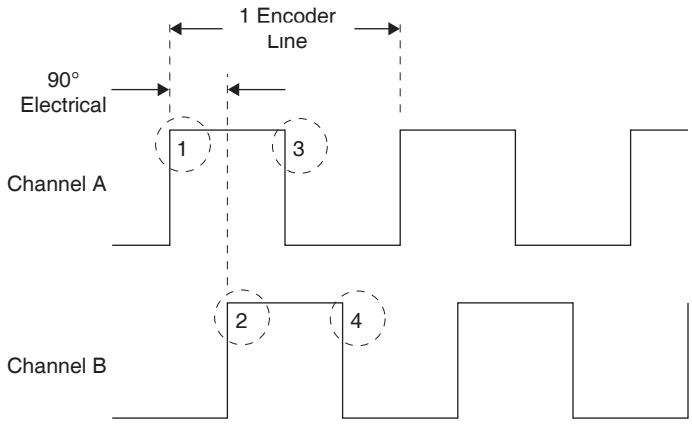
In order to rotate at 0.13333 RPS the VM must be set at 6827 steps/sec.

```
51200 x 0.133333 = 6827
VM = 6827
```

NOTE: These numbers will vary slightly depending on Acceleration and Deceleration rates.

Programming with the Optional Encoder Enabled

An optional 1000 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 4000 counts per revolution (1000 × 4 = 4000). (See Figure below.) If programming motion using the MR (Move Relative) or MA (Move Absolute) commands the motor would rotate a distance equal to the encoder counts.



(4 Encoder Pulses per Encoder Line)

Example:

A programmed move of 14000 counts would result in the motor rotating 3.5 revolutions at a velocity controlled by VM.

```
14000 ÷ 4000 = 3.5 revolutions
```

If programming motion using the SL (Slew) command the motor would rotate at a “counts per second” rate based on the programmed value.

Example:

A SL (Slew) rate of 14000 counts was programed. The motor will rotate at 14000 counts/sec., 3.5 RPS, or 210 RPM.

14000 ÷ 4000 = 3.5 RPS × 60 = 210 RPM

When the Encoder is enabled, the parameters are also changed to be compatible with the 4000 counts.

If EE=0 and the device is at default settings, setting EE to 1 will cause the following parameters to be recalculated and set to the following values:

VM.....60000 Counts/Sec.
VI.....78 Counts/Sec.
A.....78125 Counts/Sec
D.....78125 Counts/Sec.

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.

The encoder counts is continuously updated and stored in C2, regardless of the EE setting.

When EE=0, motion settings and commands are

NOTE:
— Once enabled, the encoder is always enabled and counting in C2
— Set EE at the beginning of the program and to avoid confusion during programming. Changing the EE setting will recalculate A, D, VI, and VM.

Any motion will now be programmed in encoder counts. It is possible to calculate the distance or velocity needed 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 LMD MCode device to the motor which turns the encoder. The encoder sends counts back to the drive to complete the motion. If programming 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 EE=1 mode.

NOTE: If MT has no value, the motor may hunt and never satisfy the position check.

LMM PWM Configuration

Description

This variable defines the parameter settings for the PWM and should not be used unless problems with motion, such as smoothness or positional accuracy, are experienced.

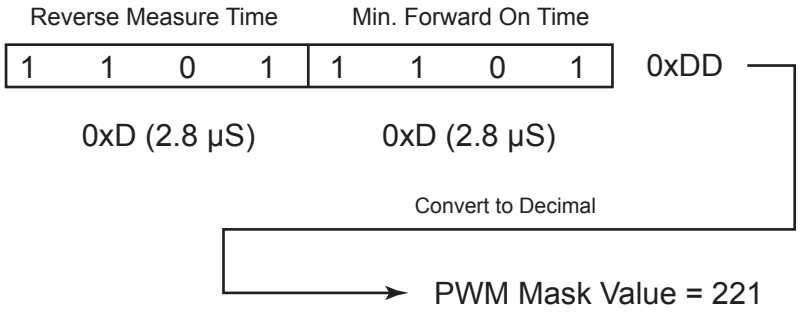
The PW variable consists of four components: <mask>, <period>, <sfrq> and <ctrl>.

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 inherent in stepper motor systems, this may be minimized or eliminated by adjusting this value. The range of this value is 0 to 255 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: 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.

PWM Mask Settings							
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



Typical PWM Mask Settings							
Mask (hex)	**Mask (dec)**	**REVTM**	**FORTM**	**Mask (hex)**	**Mask (dec)**	**REVTM**	**FORTM**
0x00	0	600 ns	600 ns	0x88	136	1.6 µs	1.6 µs
0x11	17	700 ns	700 ns	0x99	153	1.8 µs	1.8 µs
0x22	34	800 ns	800 ns	0xAA	170	2.0 µs	2.0 µs
0x33	51	900 ns	900 ns	0xBB	187	2.2 µs	2.2 µs
0x44	68	1.0 µs	1.0 µs	0xCC	204	2.5 µs	2.5 µs
0x55	85	1.1 µs	1.1 µs	0xDD	221	2.8 µs	2.8 µs
0x66	102	1.2 µs	1.2 µs	0xEE	238	3.1 µs	3.1 µs
0x77	119	1.4 µs	1.4 µs	0xFF	255	3.4 µs	3.4 µs

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%

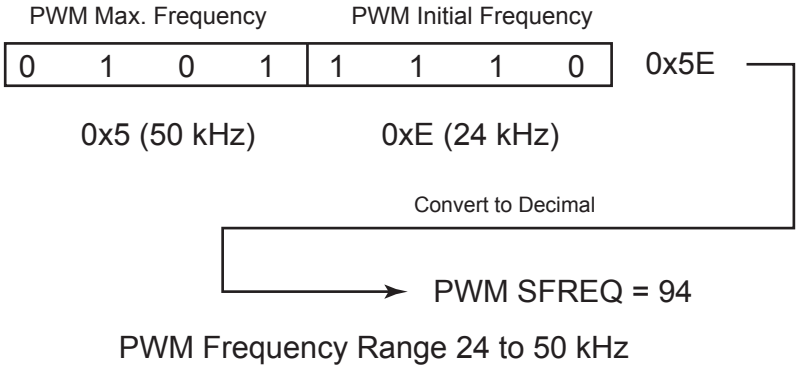
The PWM Frequency Parameter sets the initial and maximum frequencies for the PWM. As with the MASK parameter, the PWM Frequency is a two part 8-bit hex number which is entered as a decimal value ranging from 0 to 255.

The default for this 170 (0xAA) with an initial PWM Frequency of 20 kHz and a Maximum of 60 kHz.

PWM Frequency <sfrq> Parameter

Initial PWM Frequency, kHz							
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

Maximum PWM Frequency, kHz							
Hex	**Freq.**	**Hex**	**Freq.**	**Hex**	**Freq.**	**Hex**	**Freq.**
0x0	40	0x4	48	0x8	56	0xC	2.5 µs
0x1	42	0x5	50	0x9	58	0xD	2.8 µs
0x2	44	0x6	52	0xA	60	0xE	3.1 µs
0x3	46	0x7	54	0xB	62	0xF	3.4 µs



PWM Frequency Range 24 to 50 kHz

PWM Control <pwm_ctl> Parameter

Quiet		SYNC-EN	Recirc	TODLY			Enable
0	-	1	0	0	0	1	1

PWM Control Settings		
Parameter	**Description**	**Settings**
Quiet	– Minimal bridge activity to electrically “quiet” – Only useful stopped or at very low speed	0 = Disabled (Default) 1 = Enabled
SYNC-EN	– Synchronizes phase current to step once per cycle. – Prevents audible “beating”of step to PWM	0 = Disabled 1 = Enabled (Default)
Recirc	– Designates where bridge current recirculates	0 = Lower Half (Default) 1 = Upper Half
TODLY	– Delay before any bridge FET turns on	0 = 350 ns, 50 ns resolution 1 = 50 ns (Default)
Enable:	– May be cleared by an external event	0 - Bridge Disabled 1 = Bridge Enabled (Default)

Example PWM Settings by Motor Specifications

The following settings are based upon 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**: These are example settings ONLY!

Example PWM Settings								
Frame Size	**Stack Size**	**Phase Current (ARMS)**	**Phase Resistance (Ω)**	**Phase Inductance (mH)**	**MASK <mask>**	**Duty Cycle <period>**	**Frequency <sfreq>**	**Control <pwm_ctl>**
11	Single	1.0	3.5	2.3	204	95	170	37
	Double	1.4	1.77	1.56	136	95	170	37
	Triple	1.5	1.65	1.48	136	95	170	37
14	Single	0.75	4.30	4	102	90	170	35
17	Single	1.5	1.30	2.1	136	90	170	35
	Double	1.5	2.10	5.0	136	90	170	35
	Triple	1.5	2.00	3.85	136	90	170	35
23	Single	2.4	0.95	2.4	136	90	170	35
	Double	2.4	1.20	4.0	136	90	170	35
	Triple	2.4	1.50	5.4	136	90	170	35
34	Single	6.3	0.25	1.6	168	95	170	35
	Double	6.3	0.35	3.3	220	95	170	35
	Triple	6.3	0.50	6.6	253	95	170	35
LMM Default		—	—	—	204	95	170	—

Chapter 4
Hybrid Motion Technology (hMT)

NOTE: This section only applies to LMD Motion Control and LMD Ethernet Closed Loop products.

What's in this Chapter?

This chapter includes the following topics:

hMT Overview

hMT is a proprietary closed loop technology combining stepper motors with servo control that prevents the loss of synchronization due to transient or continued overload, extreme acceleration or deceleration, or excessive slew speed.

hMT Basics

hMT control technology enables the multi-mode functionality of the LMD by overcoming two major limitations inherent to stepper motor systems:

- Loss of motor synchronization and subsequent stalling.
- Excessive motor heated due to limited current control options

Loss of Synchronization

Synchronized motion in a stepper motor requires that the lead/lag relationship between the rotor and stator be within +/- 2 motor full steps. As this relationship drifts toward the 2 step point the torque available to the load is reduced, with maximum constant torque available at the <= 1 full step point.

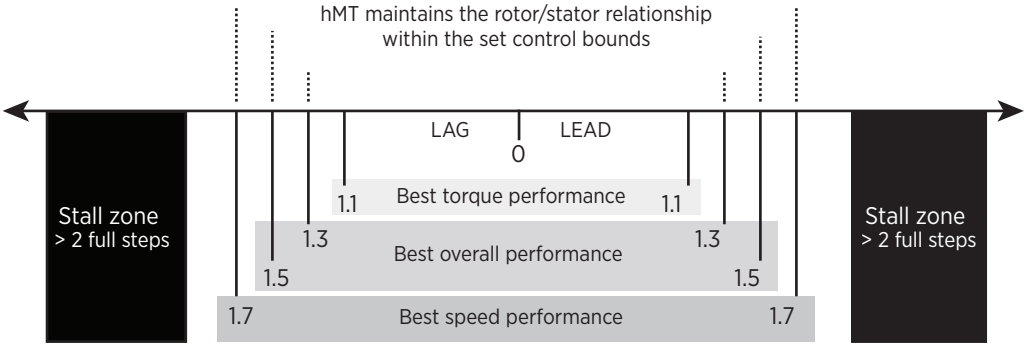
Conditions that can cause the stepper motor to lose synchronization and stall are:

Rotor lags stator:

- Acceleration is too rapid to apply enough torque to overcome the inertia of the load.
- Transient load condition at velocity; i.e., load being increased on a conveyor.

Rotor leads stator:

- Deceleration is too rapid to hold the load within the +/- 2 full step range.
- Overhauling load condition where the momentum of the load is greater than the torque supplied to maintain constant velocity.



Variable Current Control

Historically, stepper motor drivers operate at two adjustable current levels:

1. Running current (the current level in use when the shaft is moving).
2. Holding or reduction current (the current level in use when the shaft is at rest).

Variable current control uses hMT to accurately measure and track the rotor-stator relationship and apply current as needed, such as during acceleration or deceleration, then reducing the current to the level required to move the load when the axis is at velocity. This can lead to greater power efficiency and a cooler running motor.

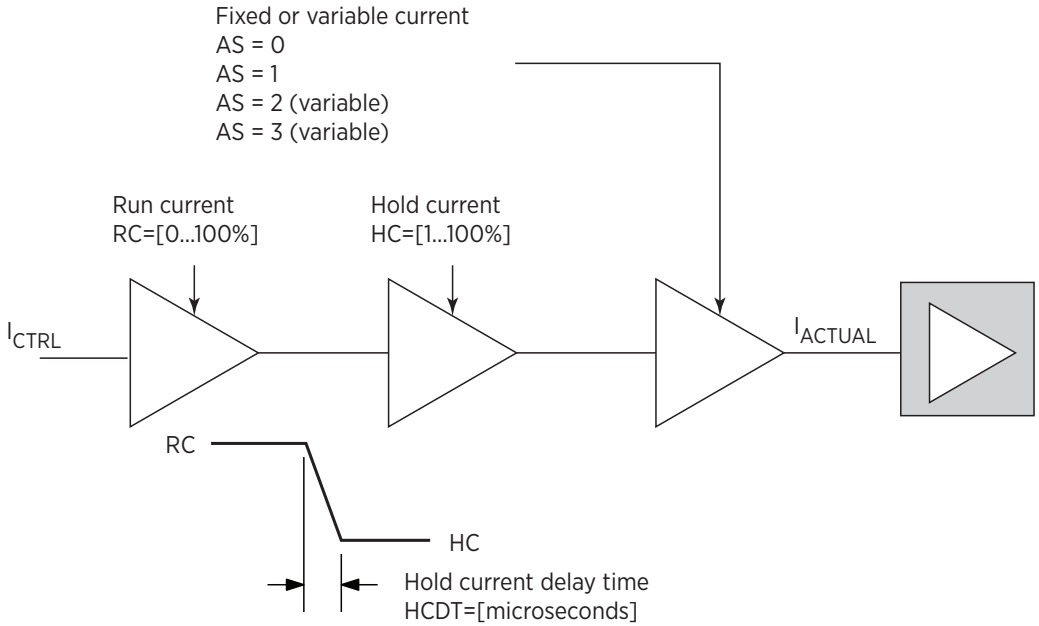
Position Make-up

When active, the position make-up function stores the difference between commended pulses and actual motor steps in a register. At the completion of the move, the lead or lag pulses will be reinserted into the profile and moved to the commanded position at one of two velocity presets.

Overview of Motor Phase Current

The motor phase current of the drive is influenced by the following factors:

- The setting of RC (Run Current).
- The setting of HC (Hold Current).
- The setting of HT (Hold Current Delay Time).
- Current control defined as fixed or variable.



hMT Modes of Operation

There are four (4) operational modes for the hMT, which is configured using AS (hMTechnology Mode):

1. hMT Off (**AS=0**)
2. hMT On (**AS=1**) fixed current
3. hMT On (**AS=2**) variable current)
4. Torque control (**AS=3**)

The selected mode will have a major effect on how the device will operate during a move.

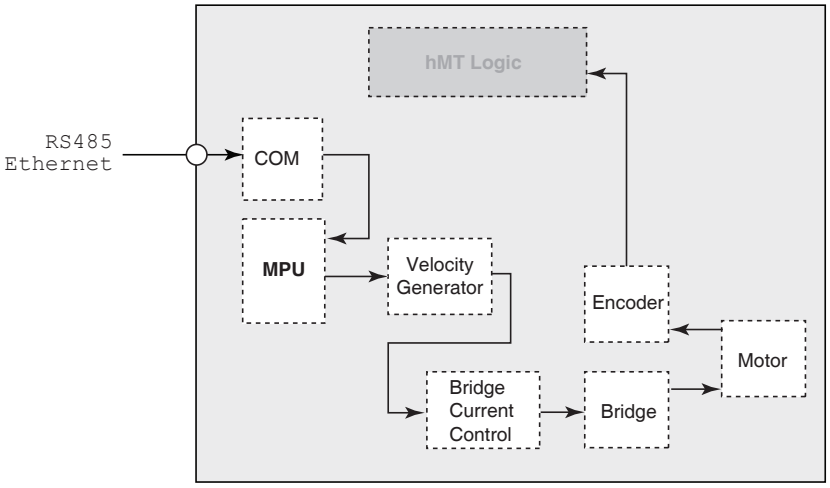
The hMT operating mode may also be changed either through programming or immediately, provided a move is not in progress.

If AS=1, 2, or 3, the encoder should not be enabled. Ensure that EE=0.

hMT Off (AS=0)

With the hMT disabled (**AS=0**), the motion block of the device will operate as a standard integrated stepper controller/drive/motor.

Commands for absolute (MA) or relative (MR) positioning, or slew at velocity (SL) are received via the communications port and processed as commanded, bypassing the hMT logic block.



The current control will be fixed at the set RC (Run Current) and HC (Hold Current) percent levels.

NOTE: Encoder functions are not available in this mode.

hMT On (Fixed Current) (AS=1)

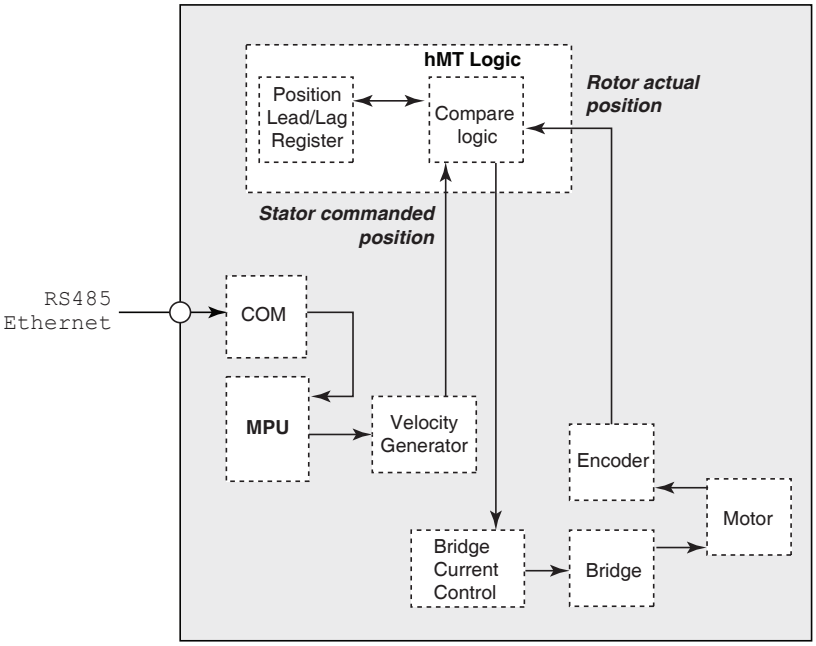
In fixed current mode (**AS=1**) the rotor/stator relationship is maintained within set control bounds using the integrated encoder.

Commands for absolute (MA) or relative (MR) positioning, or slew at velocity (SL) are received via the communications port and processed through the hMT logic block. Feedback from the encoder is compared with commanded clock pulses from the velocity generator. The output of this comparison is used to keep the rotor-stator relationship within the control bounds, thus eliminating loss of synchronization.

The variance between commanded position and actual position is stored in the lead/lag register (LL) and is used to perform a position correction move if make-up (MU) is enabled.

The device will use the RC (Run Current) and HC (Hold Current) settings for bridge current.

Common encoder functions, such as stall detection and position maintenance, are disabled when fixed current mode is selected.



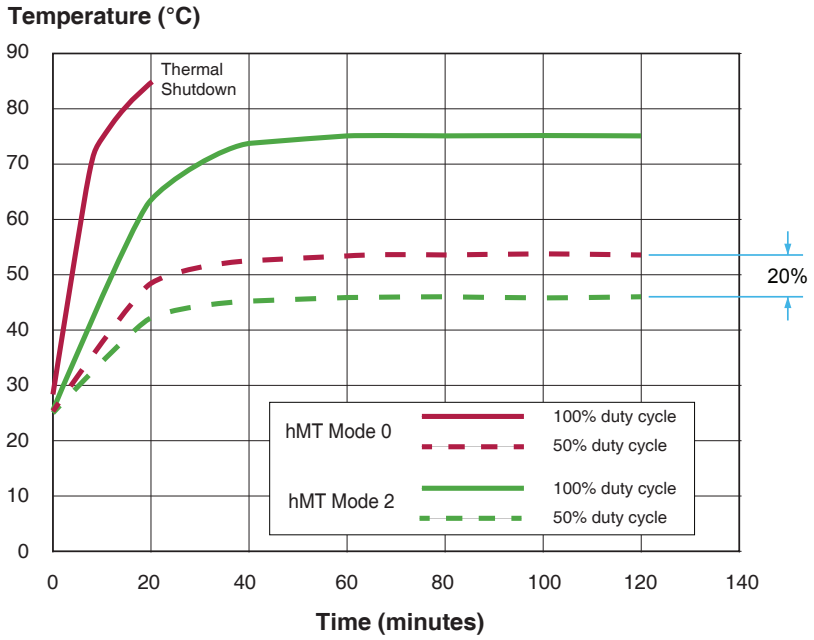
hMT On (Variable Current) (AS=2)

With hMT enabled in variable current mode (**AS=2**), the hMT will function as described for hMT On (Fixed Current), with the exception that current control will be in variable mode.

In variable current mode, the hMT will adjust the bridge current to the amount required to move the load. The set run current (RC) will be used as the maximum threshold.

Using hMT Mode 2 can significantly increase the energy efficiency and reduce the motor heating. The following graphic shows thermal performance of an LMDCM572 NEMA 23 (57 mm) running at 25% current at a speed of 2000 motor steps per second.

The first set of measurements reflect the motor running at a constant velocity. The second set show the motor running back and forth at a duty cycle of 50%.



With hMT in variable current mode, the device will use less power and run cooler, depending on load and duty cycle.

Common encoder functions, such as stall detection and position maintenance, are disabled when variable current mode is selected.

hMT On (Torque Mode) (AS=3)

With hMT in torque mode (**AS=3**), the hMT will maintain constant torque on the load at the speed required to maintain that torque.

The amount of torque used is set using the torque percent (TQ) parameter. The maximum speed for torque mode is set using the torque speed (TS) parameter. The torque direction (TD) flag may be used to control the direction of rotation.

Common encoder functions such as stall detection and position maintenance are disabled when torque mode is selected.

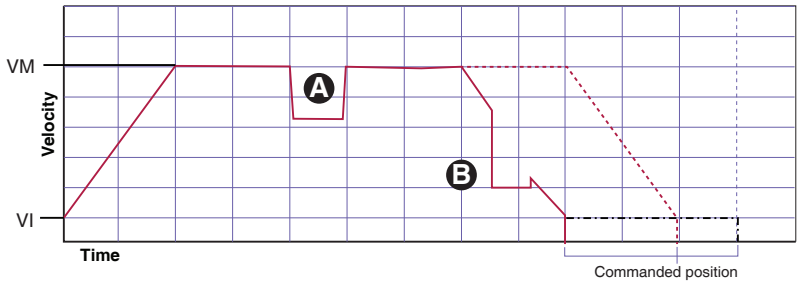
Make-up (MU) is disabled when in torque mode.

Position Make-up

Make-up mode is active whenever hMT is on in fixed (AS=1) or variable (AS=2) current mode. Make-up compensates for position errors resulting from a disturbance during a move by reinserting missed steps into a motion profile as conditions allow. The MU mode selected defines how that compensation occurs.

MU=0: Make-up happens without regard to time. In this mode missed steps are added to the motion profile to end the move at the commanded position. The speed at which the error compensation occurs is determined by the point in which the disturbance leading to the error occurs.

As shown in the diagram below, should the disturbance occur at during acceleration or at velocity, steps are added at the set maximum velocity (VM). Should the disturbance occur during deceleration, the axis will creep into position at the set initial velocity (VI).

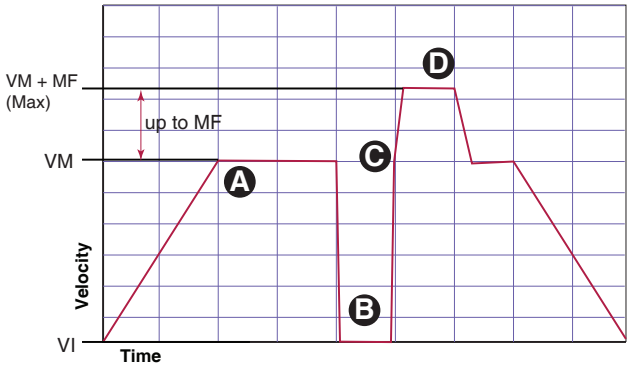


A Disturbance at velocity, error compensation occurs at VM

B Disturbance during decel, error compensation occurs at VI

MU=1: Make-up occurs as the load allows with regard to the timing of the move. In this mode, error compensation occurs by missed steps being inserted into the profile. The hMT algorithm will integrate steps into the move, attempting to complete the motion profile on time. Missed steps are reinserted when the lead/lag relationship of the rotor and stator is <=1.1 motor full steps.

As shown in the diagram below, during make-up active in mode 1, the steps will be generated at a rate (frequency) that is a composite of the maximum velocity (VM) or commanded slew rate (SL) and the set make-up (MU) frequency. This frequency will be the greater of 2 X (VM or SL) or MU.



A Axis accelerates to max. velocity (VM)

B Event (transient load, ect) stops/slows motor shaft resulting in missed steps

C hMT overcomes event, motion resumes

D Missed steps re-inserted into profile at a composite frequency that will be the greater of 2 times (VM or SL) or MF.

MU=2: In mode 2 error compensation will occur similar to mode 1 at the highest velocity the load will allow without regard to VM, but at a velocity not exceeding 2560000 steps/sec (3000 RPM).

NOTE: When the motor shaft is torqued out of position without any commanded motion, make-up will occur at ≤ MF (MU=1) or at ≤ 3000 RPM (MU=2).

Acceleration During Make-up

Make-up acceleration occurs at 16,763,806 steps/sec^2. The VI setting for make-up is 916 steps/sec. These are fixed values that cannot be changed by the user.

Locked Rotor

A locked rotor is defined as no rotor movement while at the maximum allowed lag for a specified period of time, after which a LR (Locked Rotor) condition is activated and results in an Error code. When lag becomes equal to the bounds, a timer starts to count down. Upon reaching zero, a locked rotor will be indicated by a status flag. The timer reloads on any encoder movement. The timer timeout period is user selectable from 2mS to 65.5 seconds using the LT (Locked Rotor Timeout) variable.

When configured as a step/direction drive or in speed control mode, a locked rotor will also cause an internal fault disabling the motor bridges. 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 can be used to indicate the rotor has been stopped at the specified torque for a pre-set amount of time.

hMT Specific Error Codes

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. The Locked Rotor flag will also be active (LR=1). Clear by issuing a CF (Clear Locked Rotor Fault).
105	Maximum position count reached
106	Lead limit reached
107	Lag limit reached
108	Lead/lag not zero at the end of a move
109	Calibration failed because drive not enabled.
110	Make-up disabled.
111	Factory calibration failed

Glossary of Terms

Control bounds: Control bounds establish the rotor/stator lead and lag relationship. Control bounds may be set to one of 4 parameters ranging from 1.1 to 1.7 motor full steps. When hMT is active, the technology will maintain the relationship within those boundaries, eliminating motor stalls.

Lag: The amount (in full motor steps) that the rotor lags the stator. Lag conditions are caused by loading on the motor shaft, as during transient loading or rapid acceleration.

Lead: The amount (in full motor steps) that the rotor leads the stator. Lead conditions are caused by an overhauling load, as during periods of rapid deceleration.

Loss of synchronization: In traditional stepper systems, when the lead/lag relationship of the rotor and stator reaches two full motor steps, the alignment of the magnetic fields is broken and the motor will stall in a freewheeling state. hMT eliminates this.

Locked rotor: When the lag/lead limit is reached, a timer starts a countdown that is determined by the user. The locked rotor will assert itself by triggering a flag and, depending on the selected mode, by disabling the output bridge.

Position lead/lag: hMT continually tracks the position lead or lag error, and may use it to correct position.

Position make-up: When active, the position make-up can correct for position errors occurring due to transient loads. The lost steps may be interleaved with incoming steps, or reinserted into the profile at the end of a move.

Variable current control: When active, variable current control will control the motor current as such to maintain the torque and speed on the load to what is required by the profile. This leads to reduced motor heating and greater system efficiency.

Appendix A
Sample Programs

This appendix contains example programs designed to aid the user in discovering the LMD MCode programming language.

Additional sample programs and resources are available from the following website: https://novantaims.com/resources/

NOTE: Each of the programs can be copied and pasted into the Motion Control Programmer. However, confirm that the program pasted correctly before downloading and running it on the device. If a line return is missed or punctuation does not appear as expected, the program will not run correctly and errors may occur.

Move on an Input

```
'[VARIABLES]
'This block contains the global variable and system
'configuration information.
Is=1,0,0
Ms=256
Vi=200000
Vm=2500000
A=1000000
D=A
Hc=2
Rc=75
P=0

'[PROGRAMS]
'The program block for this application sets the event
'that triggers the subroutine call when input 1 is active
'and loops when I1=inactive
PG 1
LB Ga          'Program execution label
  P=0
LB G1          'Loop back label
  CL Kb,I1=1
  H 10
  BR G1
E

'Subroutine from trigger event will execute a ten
'revolution positive move, hold, then return to 0 in the
'negative direction and repeat as long I1=1
LB Kb          'subroutine label
  MA 512000
  H
  MA 0
  H
  RT
PG             'exit program
S

'Keep this line to save program on load
'[END]
```

Enter EX Ga or EX 1 in the terminal tab to run.

Change Velocity During a Move

This program will demonstrate ability to change speed during move. The device does not have ability 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 10 times for averaging, expected end position = 102400.

```
'[VARIABLES]
'This block contains the global variable and system
'configuration information.
Hc=20
Rc=100

'[PROGRAMS]
PG 1
'Program label Ga sets local variables and register
'values. These are re-initialized each time the program
'is executed.
LB Ga
  Vi=20000
  Vm=500000
  A=500000
  D=800000000
  R1=0
  R2=0

'Label Gx sets the trip response and Performs Register
'Math to print final position
LB Gx
  P=0
  Tp=51200,Kb
  Te=2
  SL 101200
  H
  H 250
  IC R1
  R2=R2+P
  BR Gx,R1<10
  R2=R2/100
  PR "Average end pos = ",R2
  E

'[SUBROUTINES]
'Subroutine Kb, when called by Tp=51200 increases the 'axis ve-
locity by 50%
LB Kb
  SL 202400
  Tp=102290,Kc
  Te=2
  RT

'Subroutine Kc, when called from Kb ends the motion
'sequence
LB Kc
  SL 0
  H
  RT

PG
S

'Keep this line to save program on load
'[END]
```

Enter EX Ga or EX 1 in the terminal tab to run.

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.

```
'[VARIABLES]
'Define I/O configuration
Is=1,0,0
Is=2,0,0
Is=3,0,0
Is=4,0,0
Os=1,16,0

'Set up system variables
Vi=20000
Vm=1000000
A=500000
D=A
Hc=20
Rc=75

'[PROGRAMS]
'The main program block is labeled SU 'a keyword which
'will execute the program on power up.
PG 1
LB Su
  P=0

'The block G1 will call various subroutines based upon
'the weight of the inputs which is stored in register R1

LB G1
  R1=In
  R1=R1 & 7
  O1=0
  CL K0,R1 = 0
  CL K1,R1 = 1
  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]
'These 8 routines will rotate the motor
'1 time for each input bit and repeat
'the input weight changes
LB K0
  PR "Logic 000"
  MR R1*51200
  H
  O1=1
  H 2000
  RT

LB K1
  PR "Logic 001"
  MR R1*51200
  H
  H 200
  RT
```

```
LB K2
  PR "Logic 010"
  MR R1*51200
  H
  H 200
  RT

LB K3
  PR "Logic 011"
  MR R1*51200
  H
  H 200
  RT

LB K4
  PR "Logic 100"
  MR R1*51200
  H
  H 200
  RT

LB K5
  PR "Logic 101"
  MR R1*51200
  H
  H 200
  RT

LB K6
  PR "Logic 110"
  MR R1*51200
  H
  H 200
  RT

LB K7
  PR "Logic 111"
  MR R1*51200
  H
  H 200
  RT

PG
S
'Keep this line to save program on load
'[END]
```

Program will execute on power on or software reset (CTRL+C)

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.

```
'[VARIABLES]
Rc=80
Mt=50

'hMT off and encoder functions enabled and configured
As=0
Ee=1
Sf=15
Sm=0

'motion variables are scaled to encoder counts instead of 'micro-
steps
A=20000
D=A
Vi=2048
Vm=15000

'user variable created to hold move count
VA Q1

'[PROGRAMS]
'program block Ga sets the on error handle routine to
'call K1
PG 1
LB Ga
  OE K1
  P=0

'program block Gb contains the motion loop which will run
'100 times
LB Gb
  MR 51200
  H
  H 500
  MR -51200
  H
  H 500
  IC Q1
  BR Gb,Q1<100
  CL K1
E

'[SUBROUTINES]
'Subroutine K1 sets the response for the on-error
'handler. It will perform some math to 'determine the
'position error in encoder counts, as well as display the
'error # if one occurs.
LB k1
  R3=C1/25
  R1=R3 - C2
  PR "Counts error = ",R1
  PR "Error = ",Er
  Er=0
  H 20
  RT

PG
S
'Keep this line to save program on load
'[END]
```

User Input into Variables

This program demonstrates the ability to hold up program execution while the user enters multiple variables. Uses variable K1 and K2 to enter the amount and direction of motor rotation.

```
'[VARIABLES]
'System configuration variables
Ms=256
Vi=200000
Vm=2500000
A=1000000
D=A
Hc=10
Rc=75

'Globally defined user variables to contain 'input data
VA K1=0
VA K2=0
VA K3=51200
VA K4=0

'[PROGRAMS]
'Program labeled Su will start on power on 'or software reset.
Will zero the position 'counter and wait 2 sec before dropping to
'program block Z1
PG 1
    LB Su
    P=-0
    PR "At Home Position"
    H 2000

'Block will request a number of desired 'revolutions and insert
the number into 'variable K1
LB Z1
     PR "Enter the number of revolutions in whole numbers"
     IV K1
     LB X1
     BR X1, If=1
     H 50

'Block will request a direction of 'rotation and insert the num-
ber into 'variable K2, then call the appropriate  'subroutine
with error checking for 'invalid entries
 LB X4
     PR "Enter rotation direction (0) neg. (1) pos."
     IV K2
     LB X2
     BR X2, If=1
     H 50
     BR Y1,K2=0
     BR Y2,K2=1
     PR "Invalid Entry"
     BR X4

'X6 will orient the final position of the axis
'to the terminal screen
LB X6
     VA K5
     K5=P/K3
     PR "Axis position is ", K5, " absolute from home"
     H 3000

'Block X5 will initiate following the commanded
'move with an option to re-run or quit
LB X5
     PR "Repeat program (1) or quit (0)"
     IV K4
     LB X3
     BR X3, If=1
```

```
     BR Z1, K4=1
     BR Z2, K4=0
     PR "Invalid Entry"
     BR X5

'[SUBROUTINES]
'The following branch routines will
'calculate the move distance and
'direction and execute the move
LB Y1
     MR -K3*K1
     H
     BR X6

LB Y2
     MR K3*K1
     H
     BR X6

'[END]
LB Z2
PR "Program Ended"
E
PG
S
'Keep this line to save program on load
```

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.

The Homing method used is HM1, which will slew at VM (Max Velocity) in the negative direction, when input 1 is activated, the axis will creep in the plus direction at VI (Initial Velocity). See the MCode Home to home switch command and change the homing method to experiment with different methods of homing. Output 1 is set to activate when the axis is moving. Stalling the motor will generate an error, activating output 2.

```
'[VARIABLES]
'Global variable declarations
Ee=1
Vm=4096
Vi=Vm/50
A=20480
D=A
Hc=50
Rc=50
Mt=50


'Encoder setup
Sf=20
Sm=0
Db=5

'I/O setup
Is = 1, 1, 0  'Homing input
Is = 2, 0, 1  'General purpose input
Os = 1, 17, 1  'Moving output
Os = 2, 18, 1  'Error output
D1=100

'[PROGRAMS]
'Main program will home in mode 1 Slew minus @ VM until 'to find
home switch then creep plus @ VI

PG 1
LB G1
    H 1000
    PR C1 ,C1
    PR C2 ,C2
    Pm=1
    PR "Position counter: " C1
    PR "Encoder counter: " C2
    H 5000
    HM 1
    H
    P=0

'After homing, motor will move @ 7186 steps each move
'printing position each time
 LB G2
    BR G2,I2=1
    MR 7186
    H
    PR "Position: " P
    BR G2
E
PG
S
'Keep this line to save program on load on load
```

Input Trip

This program demonstrates the use input trips. The LMD Motion product will perform a short 1 revolution move in each direction repeating four times when input 1 is toggled.

When using a mechanical switch, remember to set the input filtering to avoid erroneous trips.

IMPORTANT! Trip Rules:

1. Trip must be enabled using Te=<num> following the trip definition.
2. Only a single input trip may be defined in a program.
3. Trip must be re-enabled to re-execute trip.

```
'LMD Motion Module DEMO PROGRAM
'Last modified 12/13/12
'[VARIABLES]
VA Q1
D1=255

'[PROGRAMS]
'Program will run a motion
'profile on an input toggle
PG 1
LB G1
  Ti = 1, X1
  Te = 1
  LB G2
  Q1 = 0
  BR G2
E

'Motion profile
LB X1
  IC Q1
  MR 51200
  H
  MR -51200
  H
  BR X1, Q1 < 4
  Te = 1
  RT
  E
PG      ' End of Program
S
'Keep this line to save program on load
```

Position Teach (Encoder Required)

This program allows the user to "teach" the LMD Motion product a +/- move profile based on manually positioning the motor shaft. The shaft is manually moved to a position, then an input is toggled to store that position in encoder counts to a user variable. The shaft is moved to second position, the input is again toggled to store the second position in a second variable.

The motor will then move between the two stored positions.

```
'LMD Motion Module DEMO PROGRAM
'Last modified 12/14/12
'[VARIABLES]
VA Q1 = 0
VA Q2 = 0
D1 = 255
D2 = 255

'hMT off, encoder enabled
As=0
Ee=1

'[PROGRAMS]
'Program stores a +/- move p profile based on encoder
'counts set by manually positioning the motor shaft
'An input toggle stores the encoder counts to a user 'variable.
PG 1
LB Su
    Er = 0
    C2=0
    Q1 = 0
    O2 = 0
    PR "Move motor to position 1"
    PR "Toggle switch 1 when ready"
LB X1
    BR X1, I1 = 0
    Q1 = C2
    PR Q1
LB X2
    BR X2, I1 = 1
    PR "Move motor to position 2"
    PR "Toggle switch 1 when ready"
LB X3
    BR X3, I1 = 0
    Q2 = C2
    PR Q2
LB X4
    BR X4,I1=1
    PR "Toggle Sw 2 to start cycle"
LB X5
    BR X5, I2 = 0
    LB X6
    MA Q1
    H
    PR P
    H 250
    MA Q2
    H
    PR P
    H 250
    BR X5
E
PG                ' End of Program
S
'Keep this line to save program on load
```

Analog Speed Control

This program demonstrates the use of the analog input in a speed control application.

The program subroutine performs calculations using the user registers R1-R4 and slews the axis bidirectionally based upon the value seen on the analog input.

Hardware requirement: 10kΩ potentiometer connected to the Analog input.

```
'LMD Motion Module DEMO PROGRAM
'Last modified 12/14/12
'[VARIABLES]
Os = 1, 20, 1 'Velocity changing output
A=2000000
D=2000000
R4=80

'[PROGRAMS]
'The main program block calls
'subroutine to calculate a slew rate
'based on the value of I5
PG 1
LB G1
    R1 = I5
    CL Z1
    SL R3
    H 10
    BR G1
E

'Subroutine performs calculation
'to vary the velocity based upon
'the analog input
LB Z1
    R1 = R1-2032
    R2 = 1
    BR Z2, R1>=0
    R2 = -1
    R1 = R1 * R2

LB Z2
    BR Z3,R1<R4
    R1 = R1 * 625
    R3 = R1 * R2
RT

LB Z3
    R3=0
RT

E
PG
S
'Keep this line to save program on load
```

Analog Slew with Stall Detect

This program will use the analog input reading to ram the velocity until the motor stalls. When the stall occurs, an error is generated.

A subroutine is triggered by the error to:

Print the Error number and stalled sate of the motor,

```
'LMD Motion Module DEMO PROGRAM
'Last modified 12/13/12
'[VARIABLES]
As=0
Ee=1
Sf=30

'[PROGRAMS]
'Main program will assign
'a register to do math on the value of the analog
'input and slew the register value. An on-error event
'calls a subroutine to register stall
PG 1
LB Su
    OE X1
    Er=0
    R1=I5
    R1=R1*50
    SL R1
    PR V
    H 250
    BR Su
E

'[SUBROUTINES]
'on error routine
LB X1
    PR "Error! " Er
LB Y1
    BR Y2,Er <> 86
    PR "Stall"
LB Y2
    Er=0
E
PG                      ' End of Program
S
'Keep this line to save program on load
```

Multiple Position Trips

This program will use the position trip function multiple times to change position and velocity, each time printing the position and velocity to the terminal screen.

```
'LMD Motion Module DEMO PROGRAM
'Last modified 02/21/2013

'Distance traveled is V * 12 sec.
' 33 RPM * 0.2 min =  6.6 revs. X 51200 uSteps =  337920 uStp.
' 66 RPM * 0.2 min = 13.2 revs. X 51200 uSteps =  675840 uStp.
'100 RPM * 0.2 min = 19.8 revs. X 51200 uSteps = 1013760 uStp.
' Slew Rate is V/60 * 51200 uSteps/sec.
' 33 RPM /60 Sec.  X 51200 uSteps =  28160 uStp./Sec.
' 66 RPM /60 Sec.  X 51200 uSteps =  56320 uStp./Sec.
'100 RPM /60 Sec.  X 51200 uSteps =  85333 uStp./Sec.
'Step 1  V=33RPM
'Step 2  V=66RPM
'Step 3  V=100RPM
'Step 4  V=66RPM
'Step 5  V=33RPM
'Step 6  V=-33RPM
'Step 7  V=-66RPM
'Step 8  V=-100RPM
'Step 9  V=-66RPM
'Step 10 V=-33RPM

'STEPS

'   1   2    3    4   5    6    7   8    9    10
'
'        ____
'      _____|    |____
'   ___|              |___
'_|_____________________|_____________________________Ds
'                        |___                 ____|
'                            |____     ____|
'                                 |____|
'
'[VARIABLES]
VA SP=33*51200/60 'Step 1 speed 33 RPM * 51200 Stp/rev /60 s/m
VA DS=50000  'Length of first step.

'[PROGRAMS]
PG 1
  R1=0
  P=0       'set position counter to 0
  LB AA
    PR ""
    PR "Starting Step 1   P=",P,"    V=",V
    Tp DS,X1
    Te=2
    Vm=SP       'Step 1 speed 33 RPM * 51200 Stp/rev /60 s/m
    MA DS*9,0,1
    LB G1
      BR G1,R1=0
      R1=0
      BR Aa
    E
```

```
'[SUBROUTINES]
'Each sub will move a dist at a velocity
'then redefine and re-enable the trip
'Step 2 speed 66 RPM * 51200 Stp/rev /60 s/m
  LB X1
    PR " Starting Step 2   P=",P,"   V=",V
    Vm= SP*2
    MA DS*9,0,1
    Tp DS*3,X2
    Te=2
    RT

'Step 3 speed 100 RPM * 51200 Stp/rev /60 s/m
  LB X2
    PR "  Starting Step 3   P=",P,"  V=",V
    Vm= SP*3
    MA DS*9,0,1
    Tp  DS*6,X3
    Te=2
    RT

'Step 4 speed 66 RPM * 51200 Stp/rev /60 s/m
  LB X3
    PR "   Starting Step 4   P=",P,"  V=",V
    Vm= SP*2
    MA DS*9,0,1
    Tp DS*8,X4
    Te=2
    RT

'Step 5 speed 33 RPM * 51200 Stp/rev /60 s/m
  LB X4
    PR "    Starting Step 5   P=",P,"  V=",V
    Vm= SP
    MA DS*9 ',0,1
    Tp DS*9,X5
    Te=2
    RT

'Step 6 speed 33 RPM * 51200 Stp/rev /60 s/m
  LB X5
    PR "     Starting Step 6   P=",P,"  V=",V
    Vm= SP
    MA 0,0,1
    Tp DS*8,X6
    Te=2
    RT

'Step 7 speed 66 RPM * 51200 Stp/rev /60 s/m
  LB X6
    PR "      Starting Step 7   P=",P,"  V=",V
    Vm= SP*2
    MA 0,0,1
    Tp DS*6,X7
    Te=2
    RT

'Step 8 speed 100 RPM * 51200 Stp/rev /60 s/m
  LB X7
    PR "       Starting Step 8   P=",P,"  V=",V
    Vm= SP*3
    MA 0,0,1
    Tp DS*3,X8
    Te=2
    RT
```

```
'Step 9 speed 66 RPM * 51200 Stp/rev /60 s/m
  LB X8
    PR "         Starting Step 9    P=",P,"  V=",V
    Vm= SP*2
    MA 0,0,1
    Tp DS,X9
    Te=2
    RT
'Step 10 speed 33 RPM * 51200 Stp/rev /60 s/m
  LB X9
    PR "          Starting Step 10  P=",P,"   V=",V
    Vm= SP
    MA 0
    H
    PR "           Back at Start     P=",P,"   V=",V
    R1=1
    RT
  PG
S
'Keep this line.
```

Analog to Time Delay

This program demonstrates setting a delay rate based on the analog input.

```
PG 1
OE G1
S1 = 16,0,0
S2 = 16,0,0

LB G2
  PR "Change the analog voltage to show a changing repeat rate"
  H 50
LB G3
  R1=I5             'Save I5 to R1
  R2=R1/10          'Scale the analog input value
  R4=!R4+2          'Toggle R4 (invert it)
  O1=R4             'Toggle output 1
  O2=!R4+2          'Toggle output 2
  PR "R2 =",R2     'Test print string
  H R2              'Hold for the milliseconds in R2
  H 30              'Hold for thirty milliseconds
  BR G3             'Branch back to G3
  E

LB G1
  PR "Error =",Er
  Er=0
  RT

PG
S
'Keep this line
```

MDrive Input Test

This program demonstrates and tests the MDrive Inputs.

```
PG 1
OE G1
R1=0
R2=0

LB G0
   S1=0,0,0
   S2=0,0,0
   S3=0,0,0
   S4=0,0,0
   PR "Toggle inputs to see change of state"
   PR
   PR

LB G6
   PR "I1=",I1," I2=",I2," I3=",I3," I4=",I4
   H 250
   BR G6
   E

LB G1
  PR "Error =",Er
  Er=0
  RT

PG
S
'Keep this line
```

MDrive Output Test

'Program demonstrates and tests the MDrive Outputs.

NOTE: Verify that outputs are not shorted or overloaded before running

```
PG 1
OE G1
R1=0
R2=0

LB G0
   IV R2
   PR "VERIFY THAT OUTPUTS ARE NOT SHORTED OR OVERLOADED BEFORE "
   PR " Running this test"
   PR
   PR "type   1<enter>   to test Outputs"

LB G10
   BR G10,If=1

LB G2
PR "Program is now test cycling the Outputs"
   H 1000
   R1=0
   S1=16,0,0
   S2=16,0,0
   S3=16,0,0
   S4=16,0,0

LB G4
   BR G3,R1<16
   R1=0

LB G3
   PR "R1=",R1
   H 250
   Ot=R1
   IC R1
   BR G4
   E

LB G1
   PR "Error =",Er
   Er=0
   RT

PG
S
'Keep this line
```

Trip on Falling and Rising Edge

Program will trip on both falling and rising edge of Trip/Capture input

```
Ee=1
R1=0
R2=0
A=400000
D=A

PG 1
LB K0
   BR K2,I1=0
   PR " Input 1 is on, it should be off."

LB K2
   BR K2,I1=1
   S13 = 60,0
   Tc=K1
   Te=4
   SL 40000
   R2=0

LB G1
   BR G1
   E

LB K1
   R1 = Pc
   PR "Tripped  R1= ",R1, "  Is=",Is
   PR "R1=",R1
   BR K11,R2=1
   S13 = 60,1
   Te=4
   R2=1
   RT

LB K11
   R2=0
   S13 = 60,0
   Te=4
   RT

PG
S
'Keep this line.
```

Trip on Time

'Program slews the motor at 60 RPM = 1 RPS,.sets a trip on time for 1 second, then slews at 2 RPS and re-enables the trip and repeats.

```
PG 1
  Tt 1000,X1    'Set up trip on time
  R1=0

LB G1
  SL 51200      'Slew at 1 revolution per second
  Te 8          'Enable trip on time

LB G2
                'Other program functions
  BR G2         'Loop to keep program running

LB X1           'Trip on time subroutine
  IC R1         'Increment R1
  SL 0          'Stop
  H             'Wait for Mv=0
  MA 0          'Move absolute back to zero
  H             'Wait for Mv=0
  SL 102400     'Slew at 2 revolutions per second
  PR "Tripped ",R1," Times"  'Send text to update trip count
  Te 8          'Enable trip on time
  RT

PG
S
'Keep this line
```

Appendix B
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.

LMD Error Codes

Error Code	Description
0	no error
I/O Errors	
1	OUT 1 fault
2	OUT 2 fault
6	An IO already set to this Type
7	Tried to SET an Input or Defined I/O not used
8	Tried to SET IO to an incorrect I/O type
9	Tried to Write to IO set as Input or is “TYPED”
10	Illegal I/O number
11	Incorrect CLOCK type
12	INPUT 1 not set to Capture Input type
Data Errors	
20	Tried to SET Unknown Variable/Flag
21	Tried to SET to an incorrect value
22	VI set greater than or equal to VM
23	VM set less than or equal to VI
24	Illegal Data Entered.
25	Variable or Flag is Read Only
26	Variable or Flag not allowed to be Incremented or Decremented
27	Trip Not Defined
28	Trying to Redefine a Program Label or GLOBAL User Variable
29	Trying to Redefine an Embedded Command or Variable
30	Unknown Label or User Variable
31	Program Label/User Variable Table is Full
32	Trying to SET a Label
33	Trying to SET an Instruction
34	Trying to Exec a Variable or Flag
35	Trying to Print Illegal variable or flag
36	Illegal Motor Count to Encoder Count Ratio
37	Command/Variable/Flag Not Available in Drive
38	Missing parameter separator
39	Trip on Position and Trip on Relative distance not allowed together

Error Code	Description
Program Errors	
40	Program Not Running
41	Program Running
42	Illegal Program Address
43	Tried to OverFlow Program STACK
44	Program Locked
45	Trying to Overflow Program Space
46	Not in Program Mode
47	Tried to write to illegal Flash Address
48	Program Execution Stopped by IO set as STOP
Communication Errors	
60	Tried to Enter Unknown Command
61	Trying to set illegal baudrate
62	An INPUT is already pending.
63	Character Over Run
65	SPI Bus Error.
66	Transmit buffer filled while a program is running.
67	(Ethernet Interface Only) Firmware mismatch between Ethernet and Motion Control firmware. This error will be displayed in the TCP/IP App Ver. box.
System Errors	
70	FLASH Check Sum Fault
71	Internal Temperature Warning
72	Internal OVER TEMP Fault. Disabling Drive
73	Tried to SAVE/RTFD/PG while Moving
74	Tried to IP or CP while Moving
75	ASIC STAT/FAULT = true (current/temp/other)
76	MakeUp Frequency is out of range. Must be >= 92 and <= 3000 RPM.
77	VM or VI or SL or TS too large for selected MSEL.
78	Aux V out of range (too high or too low)
79	Plus V out of range (too high or too low)
Motion Errors	
80	HOME Sw. not defined
81	HOME type not defined
82	Went to both LIMITs and didn't find HOME
83	Reached Positive LIMIT Sw
84	Reached Minus LIMIT Sw
85	MOVES not allowed while HOMING and HOME not allowed while MOVING
86	Stall Detected
87	Not allowed to change AS Mode while in motion
88	MOVES not allowed while Calibration in progress.
89	Calibration not allowed while in Motion.
90	Motion Variables (VI and/or VM) are too low
91	Motion Stopped by IO set as STOP
92	Position Error
93	New MR or MA not allowed while correcting position at end of previous MR or MA

Error Code	Description
94	Motion Commanded while Drive Disabled.
95	Not allowed to change Rotation Direction (Rd) while in motion.
96	Not allowed to start motion with no +V.
97	Calculated Final Velocity less than VI.
98	Move generates illegal s-curve Accel Data.
99	Move generates illegal s-curve Decel Data.
hMTechnology Errors	
100	Config Test Done - Encoder Res Mismatch
101	Config Test Done - Encoder Dir Wrong
102	Config Test Done - Encoder Res + Dir Wrong
103	Config NOT Done - Drive not enabled
104	hMT Locked Rotor
105	hMT Reached Max P Count
106	hMT Reached Lead Limit Count
107	hMT Reached Lag Limit Count
108	hMT Lead/lag not zero at end of move
109	hMT Calibration failed because Drive Not Enabled.
110	hMT Make Up Disabled.
111	hMT Factory Calibration failed.
Absolute Encoder Errors	
120	Absolute encoder factory calibration failed
121	Absolute encoder communication failed.
122	Absolute encoder was reset and position was lost.
123	Unable to compute new absolute encoder Count.
124	Status Error.
125	Battery hardware failure, Magnetics.
126	External battery overvoltage

LMD Motion Module Error Codes

Error Code	Description
0	no error
I/O Errors	
1	OUT 1 fault
2	OUT 2 fault
6	An IO already set to this Type
7	Tried to SET an Input or Defined I/O not used
8	Tried to SET IO to an incorrect I/O type
9	Tried to Write to IO set as Input or is «TYPED»
10	Illegal I/O number
11	Incorrect CLOCK type
12	INPUT 1 not set to Capture Input type
13	LMM Motor Phase Over Current Fault
14	LMM Enable Pin set to Disable
Data Errors	
20	Tried to SET Unknown Variable/Flag
21	Tried to SET to an incorrect value

Error Code	Description
22	VI set greater than or equal to VM
23	VM set less than or equal to VI
24	Illegal Data Entered.
25	Variable or Flag is Read Only
26	Variable or Flag not allowed to be Incremented or Decremented
27	Trip Not Defined
28	Trying to Redefine a Program Label or GLOBAL User Variable
29	Trying to Redefine an Embedded Command or Variable
30	Unknown Label or User Variable
31	Program Label/User Variable Table is Full
32	Trying to SET a Label
33	Trying to SET an Instruction
34	Trying to Exec a Variable or Flag
35	Trying to Print Illegal variable or flag
36	Illegal Motor Count to Encoder Count Ratio
37	Command/Variable/Flag Not Available in Drive
38	Missing parameter separator
39	Trip on Position and Trip on Relative distance not allowed together
Program Errors	
40	Program Not Running
41	Program Running
42	Illegal Program Address
43	Tried to OverFlow Program STACK
44	Program Locked
45	Trying to Overflow Program Space
46	Not in Program Mode
47	Tried to write to illegal Flash Address
48	Program Execution Stopped by IO set as STOP
Communication Errors	
60	Tried to Enter Unknown Command
61	Trying to set illegal baudrate
62	An INPUT is already pending.
63	Character Over Run
65	SPI Bus Error.
66	Transmit buffer filled while a program is running.
System Errors	
70	FLASH Check Sum Fault
71	Internal Temperature Warning
72	Internal OVER TEMP Fault. Disabling Drive
73	Tried to SAVE/RTFD/PG while Moving
74	Tried to IP or CP while Moving
75	ASIC STAT/FAULT = true (current/temp/other)
76	MakeUp Frequency is out of range. Must be >= 92 and <= 3000 RPM.
77	VM or VI or SL or TS too large for selected MSEL.
79	Plus V out of range (too high or too low)
Motion Errors	
80	HOME Sw. not defined
81	HOME type not defined
82	Went to both LIMITs and didn't find HOME
83	Reached Positive LIMIT Sw
84	Reached Minus LIMIT Sw

Error Code	Description
85	MOVES not allowed while HOMING and HOME not allowed while MOVING
86	Stall Detected
90	Motion Variables (VI and/or VM) are too low
91	Motion Stopped by IO set as STOP
92	Position Error
93	New MR or MA not allowed while correcting position at end of previous MR or MA
94	Motion Commanded while Drive Disabled.
95	Not allowed to change Rotation Direction (Rd) while in motion.
96	Not allowed to start motion with no +V.
97	Calculated Final Velocity less than VI.
98	Move generates illegal s-curve Accel Data.
99	Move generates illegal s-curve Decel Data.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Warranty

For the latest warranty and product information, visit https://novantaims.com/warranty-and-disclaimer/.

Document Revision History

Document Number: LMD-MCODE-V2.06		
Date	**Revision**	**Changes**
05/02/2013	V1.00, 02.2013	Initial Release
01/10/2013	V1.00, 10.2013	Lexium MDrive Motion Control and Ethernet support release.
04/07/2014	V1.00, 01.2014	Support for firmware release 5.007 - Added status marker for the read voltage level (VT) command.
04/24/2014	V1.00, 04.2014	Support for firmware release 5.009
08/15/2014	V1.00, 08.2014	Support for firmware release 5.010, minor corrections and additions throughout.
12/15/2014	V1.00, 12.2014	Support for firmware release 5.013, minor corrections and additions throughout.
06/04/2015	V1.00, 06.2015	Corrected multiple Ethernet/IP cross-references.
05/26/2016	V2.00, 05.2016	Added support for advanced math functions. Added support for the Lexium Motion Module. Reorganized to optimize usability.
10/12/2018	V2.01, 03.2018	Added RP (Referenced Position) command, added support for LMD with absolute encoder, which adds the read-only variable VB (Backup Voltage).
10/12/2018	V2.02, 01.2019	Updated current to LMDA firmware release 7.003. Added Error Codes for Absolute MDrive products.
07/24/2020	V2.03, 07.2020	Updated document format, added additional sample programs, updated EtherNet and Modbus information per code. Updated Active States info and example programs.
08/03/2020	V2.04, 08.2020	Updated NEMA 11 PWM settings.
09/09/2020	V2.05, 09.2020	Corrected IT command and updated FM default.
03/02/2022	V2.06 03.2022	Corrections to PR, TC, EM, and TR. updated command summary, added compat. to the summary tables. Corrected VT parameter information. Updated to brand Novanta

Novanta IMS is a part of the Precision Motion Group within Novanta, a leading technology company that delivers innovation to medical and advanced industrial OEMS.

As standards, specifications, and designs may change, confirmation of the information given in this publication can be found in the product disclaimer and most recent product information, available online.
https://novantaims.com/all-products/

Photos: Novanta IMS
Print: Novanta IMS
To provide feedback on Novanta IMS documentation, send an e-mail to:
documentation@imshome.com

Novanta IMS
370 North Main Street
Marlborough, CT 06447
Phone: (860) 295-6102
www.novantaims.com