Lexium MCode

Programming and Software Reference

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

Intelligent motion systems | Schneider Electric

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 Schneider Electric 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.

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 Schneider Electric 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.

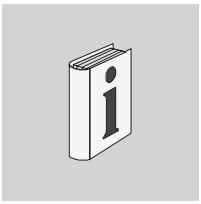
Lexium MCode Programming and Software Reference		
Date	**Revision**	**Changes**
05/02/2013	V1.00, 02.2013	Initial Release
01/10/2013	V1.00, 10.2013	Lexium MDrive Motion Control and Ethernet support release.
04/07/2014	V1.00, 01.2014	Support for firmware release 5.007 - Added status marker for the read 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.

This page intentionally left blank

Table of Contents

Page intentionall left blank

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 you to verify 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 alphanumerical 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 US units, metric values are given in SI units in parenthesis.

Examples:

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

Draft V0.900.08.2011

Page intentionally left blank

INTRODUCTION

1

1.1 About this manual

This manual covers the structure, syntax and use of the Lexium MCode programming and control language for the Lexium Motion products developed and sold by IMS | Schneider Electric Motion USA.

1.2 Applicability - Lexium MCode compatible products

Note: if you are using an MDrivePlus, MDrive Hybrid or MForce Motion Control product, please see the MCode Programming and Reference Manual for MDrivePlus, MDrive Hybrid, and MForce products.

> **MCode Programming and Software Reference**

1.2.1 Lexium MCode compatible product listing

Lexium MDrive Motion Control

Integrated Lexium MDrive Motion Control communicates over an RS-422/485 serial interface.

Lexium MDrive Ethernet TCP/IP

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

Lexium Motion Module (Motion Control)

Uses an adaptation of Lexium MCode for use with the Lexium 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

1.3 Documentation reference

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

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

1.4.1 Related documents

Lexium Software Suite

The Lexium Software Suite Manual documents the installation and use of the programming tool for the Lexium Motion products and Ethernet products.

Associated Ethernet protocols

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

- Modbus/TCP
- EtherNet/IP
- Profinet IO

Documentation for these protocols is available in separate manuals, which may be downloaded from the product manual page at **http://motion.schneider-electric.com**.

1.5 Product software

The Lexium Software Suite is the program used to commission, program and operate the Lexium Motion products. It may be downloaded from the website at:

LEXIUM SOFTWARE SUITE

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

2 SAFETY

2

2.1 Qualification of personnel

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

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

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

2.2 Intended Use

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

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

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

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

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

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

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

2.3 Hazard Categories

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

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

⚠ DANGER

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

⚠WARNING

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

⚠CAUTION

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

CAUTION

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

2.4 General safety instructions

⚠ DANGER

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

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

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

⚠ DANGER

EXPOSED SIGNALS

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

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

⚠ WARNING

LOSS OF CONTROL

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

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

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

Page intentionally left blank

Page intentionally blank

3 INTRO TO LEXIUM MCODE

3

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

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

3.1 Operational modes

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

1) **Immediate**: Commands are issued and executed directly to the controller by user input into the terminal window.
2) **Program:** Program Mode is used to input user programs into the motion controller.

3.2 Basic components of Lexium MCode

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

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

3.2.1 Instructions

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

Motion Motion instructions are those that lead to the movement of a motor. The syntax for these commands is as follows: type the command followed by a space, and then the velocity or position data. For example MA 2000 moves 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. 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. Some command examples would be PG 100, which toggles the system into program mode starting at address 100; BR LP, I1=1, which branches to a program labeled LP if input 1 is true.

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

3.2.2 Variables

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

Factory defined variables Factory defined variables cannot be deleted; they may only have their values modified. When an FD (Factory Default) instruction processes, these variables are reset to their factory default values. There are two types of factory defined variables:

- Read/Writable: These factory defined variables can have their value altered 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 variables contain data that can be viewed or used to affect events inside a program. For example, V (Velocity variable) registers the current velocity of the motor in steps per second.

User defined variables The VA instruction allows the user to assign a 2 or 3 character name to a user variable (32-bit value).

The restrictions for this command are:

1) Using a name that already exist as an MCode mnemonic is disallowed. An error 24, Illegal data entered, is asserted when attempted.

2) The first character must be alpha; the second or third character may be alphanumeric.

3) A variable is limited to two characters.

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

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

3.2.3 Flags

Flags show the status of an event or condition. A flag 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.

3.2.4 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.

3.2.5 Math functions

The Lexium products is 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.

3.3 Program structuring

Proper structuring of your Lexium MCode application ensures your ability to work efficiently and aids in troubleshooting your program. The figure below illustrates how your 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

Figure 3.1 Recommended Lexium MCode program structure

3.3.1 Programming aids

Lexium MDrive Programmer

One of the most powerful tools available to you is the Lexium MDrive Programmer module of the Lexium MDrive Software Suite. The Lexium MDrive Programmer is visual IDE (Integrated Development Environment) for developing, debugging, simulating and deploying Lexium MDrive programs written in Lexium 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 Lexium MCode programming language allows for 192 user labels for your programs, subroutines, and user variables and flags. A label consists of 2 characters, the first of which must be a letter, the second may be alphanumeric. A label cannot use the same character combination as any of the mnemonics used in the Lexium 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

Example labeling

```
VA Q1      'Create user variable Q1

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

K1         'Declare Subroutine K1
```

Comments

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

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

Programming reference

Another powerful tool is this manual. Section 5 contains detailed explanations and usage examples of each mnemonic in the Lexium MCode Programming Language. These are organized alphabetically. In Section 8 there are a number of fully commented example programs that can be used to learn the basics of programming and using the various functions of your Lexium MCode compatible device.

3.4 Commonly used variables and instructions

3.4.1 Variables

MS (Microstep resolution)

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

Motor rotation:	1.8° per step (200 steps/rev.)
Microsteps/step:	MS
Microsteps/rev:	MS * 200
MS default:	256 microsteps/step 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

As we continue you will see that all motion variables use this value.

P (Position)

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

Open loop:	Position from Counter 1 (C1) in motor steps
Encoder enabled:	Position from Counter 2 (C2) in encoder counts
To read:	PR P
To write:	P=0 will clear the position
Notes:	MS values are predefined to 20 resolutions. See command details

VI (Initial velocity)

Initial velocity in steps per second.

Default:	1000 steps/sec
To read:	PR VI
To write:	VI=<integer>
Notes:	VI will return an error is 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. (Step size is a function of the value of ms).

Default:	768000 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:	1000000 steps/sec^2
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:	1000000 steps/sec^2
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)

3.4.2 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 either microsteps per second or encoder counts (pulses) per second. When the encoder is disabled (EE=0), or hMTechnology is enabled (AS=1/2/3) motion scales in step clock pulses. In encoder mode, (EE-1), the motion commands are scaled in encoder counts.
- For example, using the default microstep resolution setting (MS=512): 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. Section 7 of this document, Application and programming notes, 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 plus direction
PR P       'the terminal screen will read 20000
MA 3000    'move 3000 steps from 0 in the plus direction
PR P       'the terminal screen will read 3000
```

Absolute moves are always relative to 0 (zero).

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

MR (Move relative)

Move the number of steps programmed relative to current position.

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

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

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

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

SL (Slew axis)

Move at a constant velocity.

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

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

H (Hold)

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

Below is a usage example.

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

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

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

3.4.3 I/O instructions

Is (Set input function) This command configures the Line, Type and Active state of inputs 1-4.

Type	Function	Description
0	GP	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). Triggers an Error 83: Positive limit reached when a + limit activates.
3	Limit minus (—)	Functions as specified by the limit variable (LM). Triggers 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.
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
- Inputs may be set globally or locally (inside a program)
- The syntax to read the settings of the inputs is PR Is

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 all inputs as decimal)

Used to read the decimal equivalent of the 4-bit binary nibble represented by all inputs collectively. Note the Input 4 is the Most Significant Bit.

Os (Set output function)

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 1 only		
26	Encoder A‡	Sets as the encoder channel A output
Available on output 2 only		
27	Encoder B‡	Sets as the encoder channel B output
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 Lexium MDrive products only*		
‡*Not available on Lexium MDrive 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 inputs 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 all outputs as BCD)

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

3.4.4 System instructions

The following system instructions will be used frequently.

CP (Clear program memory) The CP instruction is used to clear program memory space. CP must be followed by a save commend S.

FD (Restore facory defaults) The FD instruction is used to return the device to its factory default state.

ESC (Stop motion and program) <esc> The ESCAPE key will stop the user program and stop the motor with no decel rate.

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

3.4.5 Program instructions

PG (Begin program mode) This instruction toggles the device into or out of program mode.

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

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

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

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

The restrictions for this command are:

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

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

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

BR (Branch) Used to branch conditionally or unconditionally to a routine.

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

CL (Call subroutine) Used to call a subroutine conditionally or unconditionally to a routine.

```
PG 200     'Switch to program mode at address 200
LB K1              'Label command will name the program
 xxxxx
 xxxxx     'Program named by LB command
 xxxxx
 CL X1     'Unconditional call to subroutine label X1
E          'End program
PG         'Switch out of program mode

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

E (End program) Designates the end of a program.

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

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

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

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

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

RT (Return from subroutine) Required to return from a subroutine to the program.

```
PG 200     'Switch to program mode at address 200
LB K1              'Label command will name the program
 xxxxx
 xxxxx     'Program named by LB command
 xxxxx
 CL X1     'Unconditional call to subroutine label X1
E          'End program
PG         'Switch out of program mode

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

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

```
PG 200              'Switch to program mode at address 200
VA 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                  'Switch out of program mode
```

4 COMMAND SUMMARY

4

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

4.1 Compatibility

4.1.1 All Lexium MDrive products

The commands listed in Table 4.1A-D are compatible with all Lexium 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.

Abbreviations
Access: RO = Read only, RW = Read/Write, RC = Read/Clear, WO=Write only
Usage: I - Immediate, P = Program, I/P = Immediate or program

Mnemonic	Function	Access	Usage	Type	Compatibility notes
A	Acceleration	RW	I/P	Variable	—
AJ	Acceleration Jerk	RW	I/P	Variable	Firmware Version 6.0.01 +
AL	List All Parameters	—	I	Instruction	—
AO	Attention Output mask	RW	I/P	Variable	Certain attention events are product specific. See details
AT	Acceleration Type	RW	I/P	Variable	Firmware Version 6.0.01 +
BE	Backlash Enable	RW	I/P	Flag	Firmware Version 6.0.01 +
BL	Backlash Amount	RW	I/P	Variable	Firmware Version 6.0.01 +
BM	Backlash Mode	RW	I/P	Variable	Firmware Version 6.0.01 +
BP	Break Point	—	I/P	Instruction	—
BR	Branch	—	P	Instruction	—
BY	Program Executing (busy)	RO	P	Flag	—
C1	Counter 1 (step count)	RW	I/P	Variable	—
C2	Counter 2 (encoder)	RW	I/P	Variable	Encoder required
CE	CTRL+C reset enable	RW	I/P	Flag	—
CL	Call Subroutine	—	P	Instruction	—
CP	Clear Program memory	—	I	Instruction	—
CW	Clock Width	RW	I/P	Variable	—
D<1-4>	Input Filter	RW	I/P	Variable	—
D5	Analog Input Filter	RW	I/P	Variable	—
D	Deceleration	RW	I/P	Variable	—
DB	Encoder Dead-band	RW	I/P	Variable	Encoder required
DC	Decrement Variable	—	I/P	Math	—

Table 4.1A: MCode command summary - All Lexium Products

Abbreviations
Access: RO = Read only, RW = Read/Write, RC = Read/Clear, WO=Write only
Usage: I - Immediate, P = Program, I/P = Immediate or program

Mnemonic	Function	Access	Usage	Type	Compatibility notes
DE	Drive Enable/Disable	RW	I/P	Flag	—
DJ	Deceleration Jerk	RW	I/P	Variable	—
DT	Deceleration Type	RW	I/P	Variable	—
E	End Program	—	P	Instruction	—
EE	Encoder Enable/Disable	RW	I/P	Flag	Encoder required
EF	Error Flag	RC	I/P	Flag	—
EL	Encoder Lines	RW	I/P	Variable	Lexium Motion Module Only
EM	Echo Mode	RW	I/P	Flag	—
ER	Error Register	RC	I/P	Variable	—
ES	Escape <esc> Mode	RW	I/P	Flag	—
EX	Execute Program	—	I	Instruction	—
F<1-8>	Floating Point Register	RW	I/P	Variable	Firmware Version 6.0.01 +
FC	Filter Capture Input	RW	I/P	Variable	—
FD	Restore Factory Defaults	—	I/P	Instruction	—
FL	Following Mode Enable	RW	I/P	Flag	Firmware Version 6.0.01 +
FS	Index Offset Setting	RW	I/P	Variable	Firmware Version 6.0.01 + Encoder required
FT	Reserved	—	—	—	—
H	Hold Program Execution	—	P	Instruction	—
HC	Hold Current	RW	I/P	Variable	—
HF	Home to Index Offset	—	I/P	Instruction	Firmware Version 6.0.01 + Encoder required
HI	Home to Index	—	I/P	Instruction	Firmware Version 6.0.01 + Encoder required
HM	Home to Home Switch	—	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 Encode Index	RO	I/P	Variable	Encoder required
IC	Increment Variable	—	I/P	Instruction	—
IF	Variable Input Pending	RC	P	Flag	—
IN	Read Inputs as BCD	RO	I/P	Variable	—
IP	Initialize Parameters	—	I	Variable	—
IS<1-4>	Input 1 - 4 Setup	RW	I/P	Variable	—
IS 5	Analog Input Setup	RW	I/P	Variable	Parameter settings do not impact LMM analog input function
IS 6	Encoder Index Setup	RW	I/P	Variable	Encoder required
IT	Internal Temperature	RW	I/P	Variable	—
IV	Input to Variable	—	P	Variable	—
JE	Jog Enable/Disable	RW	I/P	Flag	—
L	List Program Space	—	I	Instruction	—
LB	Label	—	P	Instruction	—
LK	Lock Program	RW	I	Flag	—
LM	Limit Mode	RW	I/P	Variable	—

Table 4.1B: MCode command summary - All Lexium Products

Abbreviations
Access: RO = Read only, RW = Read/Write, RC = Read/Clear, WO=Write only
Usage: I - Immediate, P = Program, I/P = Immediate or program

Mnemonic	Function	Access	Usage	Type	Compatibility notes
LS	Software Limit	RW	I/P	Variable	Firmware Version 6.0.01 +
MA	Move Absolute	—	I/P	Instruction	—
MD	Motion Mode	RW	I/P	Variable	—
MP	Moving to Position	RO	I/P	Flag	—
MR	Move Relative	—	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	Flag	—
O<1-3>	Write Output State	WO	I/P	Variable	—
OE	On Error Handler	—	P	Instruction	—
OF	Output Fault	RC	I/P	Variable	—
OS	Output Setup	RW	I/P	Variable	—
OT	Write All Outputs	WO	I/P	Variable	—
P	Position Counter	RW	I/P	Variable	—
PC	Captured Position	RO	I/P	Variable	—
PF	Print Format	RW	I/P	Variable	Firmware Version 6.0.01 +
PG	Program Mode	—	I	Instruction	—
PK	Reserved	—	—	—	—
PM	Position Maintenance	RW	I/P	Flag	Encoder required
PN	Part Number	RO	I/P	Keyword	—
PR	Print Specified Data/Text	—	I/P	Instruction	—
PS	Pause Program	—	I/P	Instruction	—
R<1-4>	User Register	RW	I/P	Variable	—
RA	Radians/degrees	RW	I/P	Variable	Firmware Version 6.0.01 +
RC	Run Current	RW	I/P	Variable	—
RD	Reverse Direction	—	I/P	Instruction	—
RS	Resume Program	—	I/P	Instruction	—
RT	Return from Subroutine	—	I/P	Instruction	—
S	Save to FLASH	—	I/P	Instruction	—
SF	Stall Factor	RW	I/P	Variable	Encoder required
SL	Slew at Velocity	—	I/P	Instruction	—

Table 4.1C: MCode command summary - All Lexium Products

Abbreviations
Access: RO = Read only, RW = Read/Write, RC = Read/Clear, WO=Write only
Usage: I - Immediate, P = Program, I/P = Immediate or program

Mnemonic	Function	Access	Usage	Type	Compatibility notes
SM	Stall Detect Mode	RW	I/P	Variable	Encoder required
SN	Serial Number	RO	I/P	Keyword	—
SU	Start Up	—	P	Keyword	—
ST	Stall Flag	RO	I/P	Flag	Encoder required
TE	Trip Enable	RW	I/P	Flag	—
TC	Trip on Capture	RW	I/P	Variable	—
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	—	I	Instruction	—
UV	User Variables	—	I	Keyword	—
V	Current Velocity	RO	I/P	Variable	—
VA	Declare User Variable	—	I/P	Instruction	—
VC	Velocity Changing	RO	I/P	Flag	—
VI	Initial Velocity	RW	I/P	Variable	—
VM	Max. Velocity	RW	I/P	Variable	—
VR	Version	—	I/P	Keyword	—
VT	Read Voltage	RO	I/P	Variable	—
WT	Warning Temperature	RW	I/P	Variable	—

Table 4.1D: MCode command summary - All Lexium Products

4.1.2 Lexium serial products

The commands listed in Table 4.2 apply specifically to Lexium Motion products with a serial interface (RS-422/485/UART)

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

These commands are disabled on Lexium MDrive Ethernet TCP/IP products and will return an Error 37: Command/Variable/Flag not available if used.

Abbreviations
Access: RO = Read only, RW = Read/Write, RC = Read/Clear, WO=Write only
Usage: I - Immediate, P = Program, I/P = Immediate or program

Mnemonic	Function	Access	Usage	Type	Compatibility notes
BD	BAUD Rate	RW	I	Variable	—
CK	Checksum Mode	RW	I/P	Variable	—
DG	Disable Global Response	RW	I/P	Flag	—
DN	Device Name	RW	I	Variable	—
PY	Party Mode Enable	RW	I/P	Flag	—
QD	Device Queued	Rw	I	Flag	—

Table 4.2: MCode command summary - Serial Communications specific commands

4.1.3 Lexium Motion Module

The commands listed in Table 4.3 apply specifically to Lexium Motion Module. These will return an Error 37: Command/Variable/Flag not available if used.

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

Abbreviations
Access: RO = Read only, RW = Read/Write, RC = Read/Clear, WO=Write only
Usage: I - Immediate, P = Program, I/P = Immediate or program

Mnemonic	Function	Access	Usage	Type	Compatibility notes
EL	Encoder Lines	RW	I/P	Variable	—
PW	PWM Mask	RW	I/P	Variable	—
SA	Step Angle	RW	I/P	Variable	—

Table 4.3: MCode command summary - Lexium Motion Module specific commands

4.1.4 hMTechnology specific

The commands listed in Table 4.4 apply specifically to Lexium MDrive closed loop products with the hMTechnology functions. These will return an Error 37: Command/Variable/Flag not available if used on open loop or Lexium Motion Module products.

- Lexium MDrive Motion Control (P/N LMDCMxxx)
- Lexium MDrive Ethernet TCP/IP (P/N LMDCExxx)

Abbreviations
Access: RO = Read only, RW = Read/Write, RC = Read/Clear, WO=Write only
Usage: I - Immediate, P = Program, I/P = Immediate or program

Mnemonic	Function	Access	Usage	Type	Compatibility notes
AF	hMT Status	RO	I/P	Flag	—
AS	hMT Mode	RW	I/P	Variable	—
AV	Actual hMT Velocity	RO	I/P	Variable	Firmware Version 6.0.01 +
CB	Control Bounds	RW	I/P	Variable	—
CF	Clear Locked Rotor	—	I/P	Instruction	—
LD	Lead Limit	RW	I/P	Variable	—
LG	Lag Limit	RW	I/P	Variable	—
LL	Position Lead/Lag	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	—
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	—

Table 4.4: MCode command summary - hMTechnology specific commands

4.2 Math functions

The MCode math, comparison, logic and trigonometric operators shown in Table 4.5 are compatible with all Lexium Motion Control and Ethernet TCP/IP products. The advanced floating point math and trigonometric functions are the ONLY available in models with Firmware Version 6.0.01 +.

Note that math and trigonometric functions performed outside the floating point registers (F1 - F8) will be rounded down to the nearest integer.

Operator	Function	Usage
+	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 and/or Equal	R1 <= R2
>	Greater Than	R1 > R2
>=	Greater Than and/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 6.0.01 +		
AB	Absolute Value	F1 = AB R1
CS	Cosine	F1 = CS F2
C_	Arc Cosine	F1 = C_ F2
LO	Logarithm (Base 2)	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

Table 4.5: MCode command summary - Math and Trigonometric functions

5 COMMAND DETAILS

5

This section consists of two main subsections, 5.1: Commands, which has detailed explanation of each Lexium MCode command, and 5.2: Math, Logic and Trigonometric operators.

5.1 Commands

5.1.1 A (Acceleration)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

Description

Defines the acceleration rate when changing velocity. If the value of A is 76800 steps per second^2, the motor accelerates at a rate of 76800 counts per second, every second at the default linear acceleration type.

With VM (Maximum Velocity) 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.

The 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**	steps/sec^2	**Default**	1000000
Range (Encoder)	6 to 44 X 10^6		counts/sec^2		78125
Syntax	A=<integer>, PR A				

Code example

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

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0000
	0x66	1	0x01		

5.1.2 AF (hMT Status)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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.

In most cases the flag will return a status code 128: hMT Initialization complete, as hMT will initialize on power up/reset.

Example: PR AF returns a status code of 5, 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	See above	Units	—	Default	—
Syntax	PR AF, BR <label/address>, AF = <value>				

Code example

`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=4`	Call subroutine Q1 if a lead or lag limit is reached

Related	AO (Attn Output)	AS (hMT Mode)	TA (hMT Status Trip)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x008F
	0x6A	1	0x01		

5.1.3 AJ (Acceleration Jerk)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

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

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 Lexium 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). For example: With AJ=64, the Acceleration ramp will be constant for 128 samples, or 64 samples on either side of the ramp center.

See Figure 5.1: Acceleration Jerk, for example.

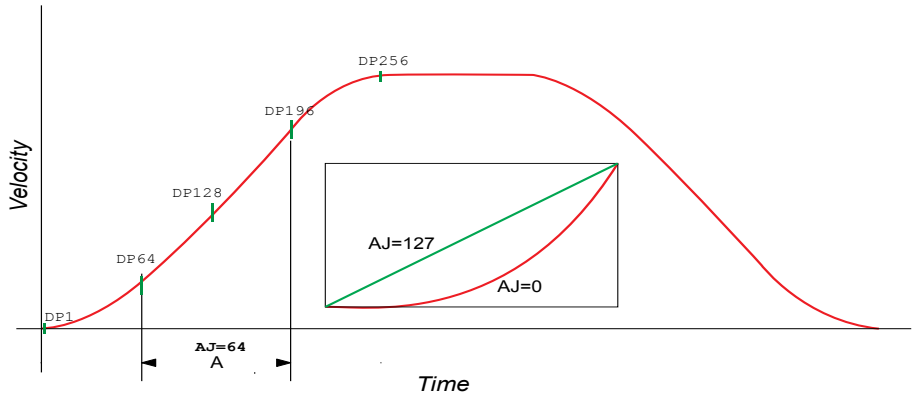


Figure 5.1 Acceleration jerk

Range	0 to 127	Units	—	Default	0
Syntax	AJ=<value>,, PR AJ				

Code example

`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)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.4 AL (List All Parameters)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
AL	Return All Parameters	Keyword	RO	Immediate

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.

Range	—	Units	—	Default	—
Syntax	PR AL				

Code example

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

Related	FD (Factory Defaults)	IP (Initialize Parameters)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.5 AO (Attention Output Mask)

Compatibility	LMD(O) LMD(C) LMM	Notes	Trigger events vary with hMT

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

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. i.e. Lead limit (4) and Lag limit (8) AO=12, Moving flag (16384) and Stall Flag (32768) AO=49152

Note that the available trigger events will vary depend on the model Lexium Motion Control product. Highlighted events apply only to Lexium MDrive models with hMTechnology.

*External encoder required for function

Range	0 - 4,294,967,295	Units	—	Default	0
Syntax	AO=<mask>,				

Code example

`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)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	See Modbus/TCP Fieldbus Manual Section 4.3: Mfg Specific Function Codes
	0x67	1	0x01		

5.1.6 AS (hMTechnology Mode Select)

⚠WARNING

EXECUTION OF MOTION

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

- Motion will occur immediately on AS=3

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

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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 Lexium MDrive 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.**

Range	0-3	Units	—	Default	0
Syntax	AS=<mode>, PR AS				

Code example

`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)	MS (Microstep resolution)	TD (Torque Dir)
	TQ (Torque %)	TS (Torque Speed)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x008E
	0x6A	1	0x02		

5.1.7 AT (Acceleration Type)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

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

Description

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

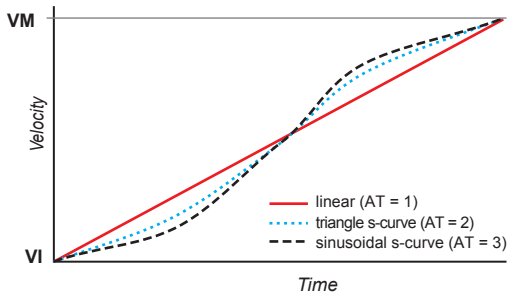


Figure 5.2 Acceleration ramp types

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 example

`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)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.8 AV (Actual hMT Velocity)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

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

Description

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

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

Code example

`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)	

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	—
	—	—	—		

5.1.9 BD (BAUD Rate)

Compatibility	LMD(O) LMD(C) LMM	Notes	Serial RS-422/485/UART only

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

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.

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 Lexium device is reset, it will communicate at the new BAUD rate. Additionally. when the BAUD is changed, it MUST be matched in Lexium Motion Control Programmer.

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 BAUD command is incompatible with Lexium MDrive TCP/IP products. If used, an Error 37: Command not available, will return when queried.

Mode	Operation
48	4800 bps
96	**9600 bps (default)**
19	19200 bps
38	38000 bps
11	115000 bps

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

Range	See table above	Units	—	Default	96 (9600 bps)
Syntax	BD =<mode>, PR BD				

Code example

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)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.10 BE (Backlash Enable)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

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

Lexium Motion Products are able to compensate for that amount, eliminating any positional errors due to backlash.

using the BM (Backlash Compensation Mode) and BL (Backlash Compensation Amount) variables.

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

Range	0/1	Units	—	Default	0
Syntax	BE=<0/1> | PR BE				

Code example

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

Related	BL (Backlash Amount)	BM (Backlash Mode		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.11 BL (Backlash Amount)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

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

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).

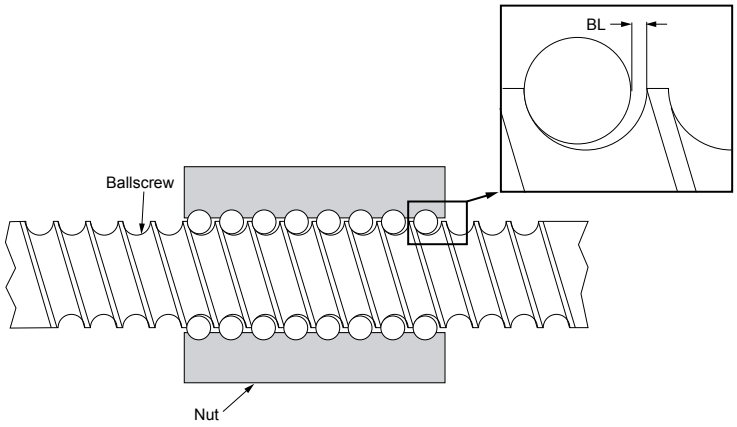


Figure 5.3 Backlash amount parameter

Range	±2147483648	Units	steps / counts	Default	0
Syntax	BL=<steps> | PR BL				

Code example

`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)

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.12 BM (Backlash Mode)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

Mnemonic	Function	Function Group	Access	Usage
BM	Set/Read Backlash Mode	Motion Variable	RW	Program/Immediate
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 employed the value of BL (Backlash Amount) adds 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.

MCode parameters

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

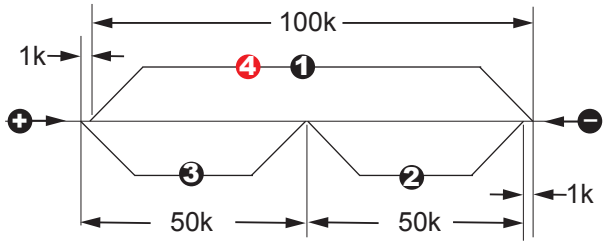


Figure 5.4 Backlash compensation Mode 0: Mathematical

Figure 5.4 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
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 (±) of BL has the value specified by BL added to it. A

separate move is then made relative to the sign (±) of BL to take up the backlash amount and “load” the axis. Whenever possible, program more backlash than there actually is.

MCode parameters

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

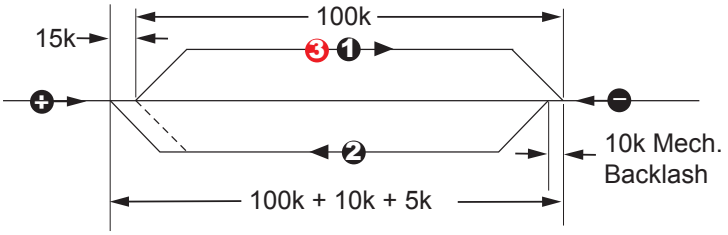


Figure 5.5 Backlash compensation Mode 1: Mechanical

Figure 5.5 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 Figure 5.5 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.

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 example

`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)		

Networking protocol equivalents

EtherNet/IP	—	**Modbus/TCP**	—

5.1.13 BP (Break Point)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

Description

The Break Point Instruction is a debugging tool used to set break points within a program to assist in troubleshooting and optimizing your Lexium 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 example

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

Related	EX (Execute program)			

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.14 BR (Branch)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

Description

The branch instruction is used to perform a conditional or unconditional branch to a location in a Lexium MCode program. It can also be used to perform loops and IF THEN logic within a program.

There are two parameters to a branch instruction. These are used to perform two types of branches:

Conditional Branch

Two parameters define a conditional branch: the first specifies an address or user label where program execution should continue when the conditions defined by the second parameter occur. The condition parameter 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 this type of 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	(unconditional) BR <label/address> | (conditional) BR <label/address>, [VAR/FLG/IN]<math><condition>				

Code example

`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)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.15 BY (Program Busy)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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

Range	0/1	Units	—	Default	—
Syntax	PR BY				

Code example

`PR BY`	Return the state of the busy flag

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

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.16 C1 (Motor Step Counter 1)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

This variable contains the 32-bit integer count of the clock pulses generated by the Lexium MCode compatible device. Counter 1 supplies the position count for P (Position Counter) when the Lexium 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 rolls over to the limit value of the opposite signed count and counts up or down from there.

For example

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

Range	-2147483648 to +2147483647	Units	motor counts	Default	—
Syntax	C1=<counts> | PR C1 | BR <label/address>, C1=<counts>				

Code example

`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

Snippet File [Download Snippet]

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, the call a subroutine to increment the rollover counter variable, then zero C1 before returning to the program.

To use: Download the program sample and extract from the zip file. Open c1-counter-rollover.ixt in Motion Control Programmer and download to your Lexium Motion product. Enter EX X1 to execute.

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

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

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	**Modbus/TCP**	0x0005
	0x68	1	0x01		

5.1.17 C2 (Encoder Counter 2)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

Description

This variable contains the 32-bit integer value of the encoder counts read by the Lexium MCode compatible device. In encoder mode Counter 2 supplies the position count for P (Position Counter) when the Lexium Motion product is operating in open loop mode without an encoder or EE (Encoder Enable) is set to one (1/enabled).

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.

For 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	-2147483648 to +2147483647	Units	encoder counts	Default	—
Syntax	C2=<counts> | PR C2 | BR <label/address>, C2=<counts>				

Code example

`C2=10000`	Set the value of counter 2 to 10000
`PR C2`	Read the value of counter 2 to the terminal
`BR Q1, C2=40000`	Conditional branch to named location Q1 when counter 2 = value
`CL X5, C2=40000`	Conditional call to named subroutine X5 when counter 12 = value

Related	C1 (Counter 1)	EE (Encoder Enable)	EL (Encoder Lines)	P (Position Counter)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0007
	0x69	1	0x01		

5.1.18 CB (Control Bounds)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

Description

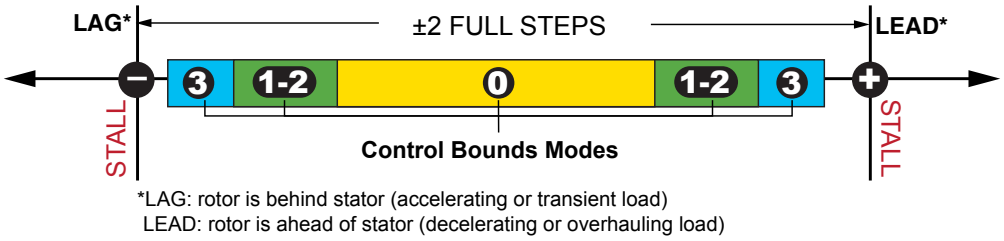
The CB (Control Bounds) variable defines the operational tolerance for the closed loop hMTechnology. The four (4) settings that are used to tune the control tolerance to optimize the device for torque, speed or balanced torque-speed performance.

The mode settings represent 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.



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

Figure 5.6 Control bounds variable for hMTI

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

Range	0-3	Units	—	Default	1
Syntax	CB=<mode> | PR CB				

Code example

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)			

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0091
	0x6A	1	0x03		

5.1.19 CE (Software Reset Enable)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

Description

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

Mode	Operation
0	Disabled, Lexium 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 example

`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)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0009
	0x64	1	0x01		

5.1.20 CF (Clear Locked Rotor)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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

A locked rotor is indicated by both the LR (Locked Rotor Flag), by the assertion of an Error 104, 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 example

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

Related	AS (hMT Mode)	LR (Locked Rotor)	LT (Locked Rotor Timeout)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0093
	0x6A	1	0x04		

5.1.21 CK (Checksum Mode)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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 checksum 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 example

`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

Download	Checksum Calculator (*.xlsx)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0009
	0x64	1	0x01		

5.1.22 CL (Call Subroutine)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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 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 condition setting includes variables, 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	(unconditional) CL <label/address> (conditional) CL <label/address>, [VAR/FLG/IN]<math><condition>				

Code example

`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)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.23 CP (Clear Program Memory)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

Description

Clears the program space in the 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.

An S (Save) command 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 example

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)

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.24 CW (Clock Width)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

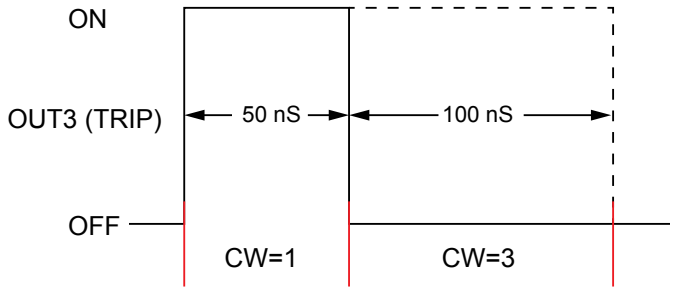


Figure 5.7 Clock width command impact on Trip output pulse width

Range	0 - 255	Units	50 nSec	Default	10 (x 50 nSec)
Syntax	CW=<time>, PR CW				

Code example

`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)			

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x000E
	0x64	1	0x02		

5.1.25 D1 - D4 (Digital Input Filter)

Compatibility	LMD(O) LMD(C) LMM	Notes	

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

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 example

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

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

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x000F 0x0010 0x0011 0x0012
	0x67	1	0x02 — 0x06		

5.1.26 D5 (Analog Input Filter)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

Description

The Analog Filter is a continuously updating process. It does a running average (A_A) 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 example

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

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

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0013
	0x67	1	0x06		

5.1.27 D (Deceleration)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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 (Clock mode)	66 to 1100 X 10^6	**Units**	steps/sec^2	**Default**	1000000
Range (Encoder)	66 to 44 X 10^6		counts/sec^2		40000
Syntax	D=<integer> | PR D				

Code example

`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)	VI (Initial Velocity)
	VM (Max Velocity)	EE (Encoder Enable)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	**Modbus/TCP**	0x0018
	0x66	1	0x02		

5.1.28 DB (Encoder Deadband)

Compatibility	LMD(O) LMD(C) LMM	Notes	LMM operability requires connected and configured encoder

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

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	0 — 65000	Units	counts	Default	1
Syntax	DB=<counts>, PR DB				

Code example

`DB=10`	Set the encoder deadband to ±10 counts
`PR DB`	Read the value of DB to the terminal

Related	C2 (Counter 2)	EE (Encoder Enable)	PM (Position Maint.)	SF (Stall Factor)
	SM (Stall Mode)	ST (Stall Flag)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x001A
	0x69	1	0x02		

5.1.29 DC (Decrement Variable)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

Description

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

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

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

Syntax	DC <var>

Code example

`DC V1`	Decrement user variable V1

Related	IC (Increment Variable)	VA {Create User Var)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	0x001B

5.1.30 DE (Drive Enable/disable))

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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), returns an Error 94: attempting motion while the drive is disabled.

Mode	Operation
0	Bridge outputs disabled. Attempting motion returns an error 94.
1	Bridge outputs enabled (default).

Range	0/1	Units	—	Default	1 - Enabled
Syntax	DE=<mode>				

Code example

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

Related	—			

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x001C
	0x64	1	0x03		

5.1.31 DG (Disable Global)

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

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

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.

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 that 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 example

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)		

5.1.32 DJ (Deceleration Jerk)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

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

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 Lexium 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. See Figure 5.8: Deceleration Jerk, for example.

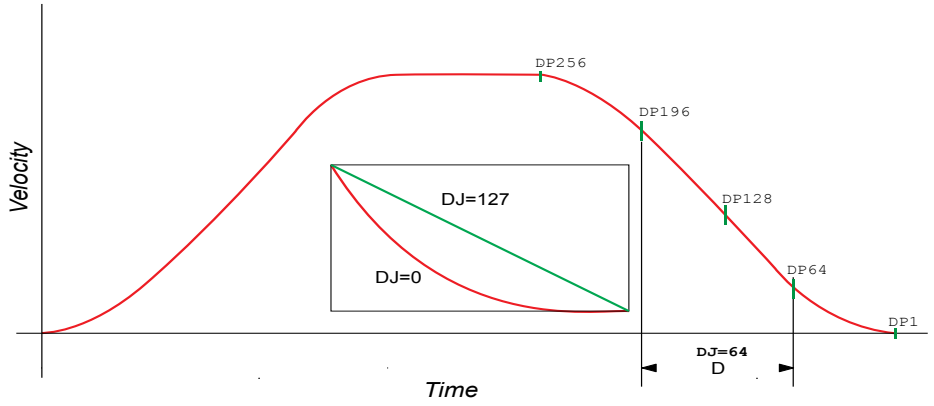


Figure 5.8 Deceleration jerk

Range	0 to 127	Units	—	Default	0
Syntax	DJ=<value> | PR DJ				

Code example

`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)	DT (Decel Type)
	VI (Initial Velocity)	VM (Max Velocity)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.33 DN (Device Name)

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

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

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.

The name is case sensitive.

Resetting the device to the default character (!) requires an FD (Factory Default Reset). The device name must be saved or it will be lost on power cycle or factory reset.

Range	a-z, A-Z, 0-9	Units	ASCII	Default	!
Syntax	DN="<char>" | PR DN				

Code example

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

Related	PY (Party Mode)			

5.1.34 DT (Deceleration Type)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

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

Description

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

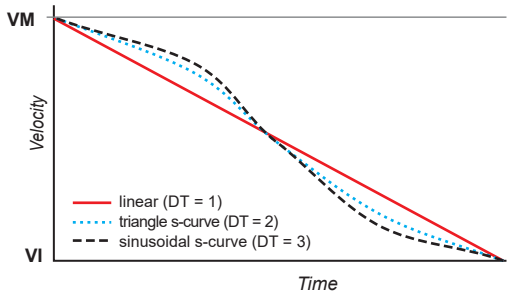


Figure 5.9 Deceleration types

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	1- 3	Units	—	Default	1 - Linear
Syntax	DT=<type> | PR DT				

Code example

`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)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.35 E (End Program)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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.

Syntax	E

Code example

E	End program

Related	EX (Execute Program)	PG (Program Mode)		

5.1.36 EE (Encoder Enable)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
EE	Encoder functions enable	Configuration flag		

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 mainteneance 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.

Range	0/1	Units	—	Default	0 (disabeled)
Syntax	EE=<mode> | PR EE				

Code example

EE=1	Enable encoder functions
PR EE	Return the status of encoder functions

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

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x001E
	0x69	1	0x03		

5.1.37 EF (Error Flag)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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 read the value of the ER (Error) variable or set ER=0.

If an external indication of the EF status is desired, 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 Lexium MDrive products.

There is an instruction, OE (On Error), which allows the user to specify the execution of a subroutine in the program memory when an error occurs. The subroutine might contain instructions to read the ER variable that would clear the EF flag.

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

Range	0/1	Units	—	Default	—
Syntax	PR EF				

Code example

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

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

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x001F
	0x65	1	0x03		

5.1.38 EL (Encoder Lines)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

Description

The Lexium 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 minimum = (EL x 8) ÷ <FS_{REV}>

The following example uses a 512 line encoder and a 1.8° stepper motor (200 F:

512 X 8 = 4096, 4096 ÷ 200 = 20.48

Minimum MS = 25 µsteps/step (5000 steps/rev.)

Range	1 to 2000	Units	lines	Default	1000
Syntax	EL=<lines>, PR EL				

Code example

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

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.39 EM (Echo Mode)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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 example

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

Related	—			

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.40 ER (Error Register)

Compatibility	LMD(O) LMD(C) LMM	Notes	List of error codes will vary between products

Mnemonic	Function	Function Group	Access	Usage
ER	Error Register	Variable	R/W	Program/Immediate

Description

Holding register for the most recent error that has occurred. The ER variable must be read or set to zero to clear the error code and reset 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. Section 9: Error codes lists the error codes for each product:

Section 9.1: Lexium MDrive (Open Loop)

Section 9.2: Lexium MDrive (Closed Loop with hMTechnology)

Section 9.3: Lexium Motion Module

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

Code example

Set to a value	
`ER=0`	Clear stored error code
Display error	
`PR ER`	Return last error to the terminal window
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 Mask)	BR (Branch)	CL (Call Subroutine)	EF (Error Flag)
	OE (On Error)	RT (Return)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x00121
	0x65	1	0x03		

5.1.41 ES (Escape Mode)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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 example

Set escape mode	
`ES=0`	Set escape to trigger on [CTRL + E] keypress
Display mode setting	
`PR ES`	Return last error to the terminal window

Related	—			

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.42 EX (Execute Program)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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.

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 example

`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)			

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0041, 0x0024
	0x64	1	0x06		

5.1.43 F1 — F8 (Floating Point Registers)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

Mnemonic	Function	Function Group	Access	Usage
F1, F2, ... F8	Floating Point Registers	Mathematics Variable	RW	Program/Immediate

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.

Range	max +	1.7976931348623158^{308}	Units	—	Default	0
	min +	$4.9406564584124655^{-324}$				
Syntax		F<num>=<fpvalue>, F<num><math/trig><var/reg>				

Code example

Set to a value	
`F2=3.256`	Set F2 to a value of 3.256
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)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.44 FC (Filter Capture)

Compatibility	LMD(O) LMD(C) LMM	Notes	Not applicable to LMDxx42

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

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

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

Note that the FC command is not available on Lexium MDrive NEMA 17 Motion Control and Ethernet TCP/IP product.

Range	0 — 9	Units	—	Default	0
Syntax	FC=<mode>, PR FC				

Code example

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)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0024
	0x67	1	0x07		

5.1.45 FD (Factory Defaults)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
FD	Restore factory default settings	Instruction	—	Immediate

Description

Issuance of the FD (Factory Defaults) command resets the device to factory default state WITHOUT WARNING upon entering FD followed by a carriage return. An FD results in the loss of all saved data: programs, user variables, and stored parameter values.

NVM values will be retained. PN (Part Number), SN (Serial Number), PW (PWM Mask) and FS (Index Offset) settings.

Syntax	FD

Code example

`FD`	Reset the device to factory default state

Related	CP (Clear Program)	IP (Initialize Parameters)		

5.1.46 FL (Following Mode Enable)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

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

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: Configuration of IN3 and IN4 as an 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 that the FL command is not applicable to the encoder inputs on the Lexium 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 example

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)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.47 FM (Filter Motion Inputs)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

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

Description

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

Prerequisite: Configuration of IN3 and IN4 as an 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	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	FM=<mode>, PR FM				

Code example

`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)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.48 FS (Index Offset Setting)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+, if using an LMM a connected and configured encoder is required.

Mnemonic	Function	Function Group	Access	Usage
FS	Index Offset Setting	Configuration Variable	RW	Program/Immediate

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 Programmer application. To configured, select View > Set Shaft Flat Position and follow the instructions on the configuration dialogs.

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. You may verify the index by entering 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) Retrieve 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 Lexium MDrive or Lexium Motion Module with a connected and configured encoder is required.

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

Code example

`FS=10246`	Set the offset to 10346 counts
`PR FS`	Read the value of the shaft flat offset

Related	HF (Home to Offset)	I6 (Index Mark)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.49 FT (Reserved)

Mnemonic	Function	Function Group	Access	Usage
FT	Reserved for Factory	Reserved	—	—

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	—

5.1.50 H (Hold Program Execution)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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)

The only 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 example

`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)	MA (Move Absolute)	MR (Move Relative)
	HF(Home to Shaft Flat)	HI (Home to Index)	HM (Home to Home Switch)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.51 HC (Hold Current)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

Description

Defines the motor holding current as 0, or 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 standstill current.

Notes:

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

The Lexium Motion Module is a 1.5A RMS standalone integrated driver/controller. The actual drive output current is derived thus: HC=5 results in a holding current level of 0.075A 1.5A * 0.05 = 0.75A.

Range	0 (disabled), 1 to 100	Units	Percent (%)	Default	5%
Syntax	HC=<percent>, PR HC				

Code example

`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)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0029
	0x66	1	0x03		

5.1.52 HF (Home to Offset)

For a detailed explanation on the homing types see [homing parameter details section]. The homing types are common to all the homing instructions.

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+ Encoder

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

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.

Diagrammed in detail in [homing parameter details section] are the four combinations for this command, as well as for the related HI (Home to Index Mark) and HM (Home to Home Switch) instructions.

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

Range	1 to 4	Units	—	Default	—
Syntax	HF <type>				

Code example

`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)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.53 HI (Home to Index Mark)

For a detailed explanation on the homing types see [homing parameter details section]. The homing types are common to all the homing instructions.

Compatibility	LMD(O) LMD(C) LMM	Notes	Encoder

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

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 shaft flat position clears.

Diagrammed in detail in [section var] are the four combinations for this command, as well as for the related HF (Home to Index Offset) and HM (Home to Home Switch) instructions.

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

Range	1 to 4	Units	—	Default	—
Syntax	HI <type>				

Code example

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

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

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x002A
	0x69	1	0x04		

5.1.54 HM (Home to Home Switch)

For a detailed explanation on the homing types see [homing parameter details section]. The homing types are common to all the homing instructions.

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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.

To use HM (Home to Home Switch), a switch connected to an input defined as home using the IS (Input Setup) command thus IS=<input #>, 5,<active>. For example, Is=2,5,0 configures Input 2 as a homing input, active when low.

Diagrammed in detail in [homing parameter details section] are the four combinations for this command, as well as for the related HF (Home to Index Offset) and HI (Home to Index) instructions.

Note that HM is the only homing function available without an encoder.

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

Range	1 to 4	Units	—	Default	—
Syntax	HM <type>				

Code example

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

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

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x002B
	0x68	1	0x02		

5.1.55 HT (Holding Current Delay)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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.

Thus 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 returns an Error 21: Illegal data value entered.

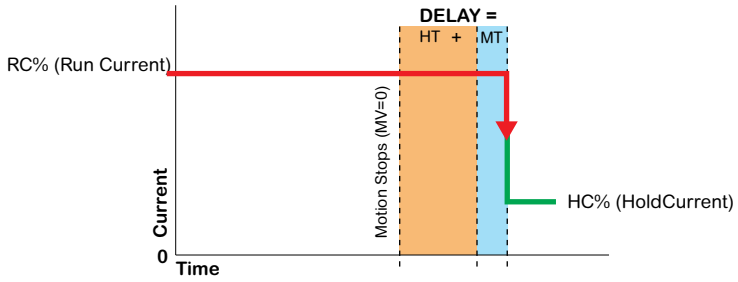


Figure 5.10 HT (Hold Current Delay) and MT (Motor Settling Delay) relationship

Range	0 to (65535-MT)	Units	milliseconds	Default	500
Syntax	HT=<time>, PR HT				

Code example

`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)	

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x002C
	0x66	1	0x04		

5.1.56 I<1-4> (Read Input 1-4)

Compatibility	LMD(O) LMD(C) LMM	Notes	Input 1 is not available on NEMA 17 (42 mm) models.

Mnemonic	Function	Function Group	Access	Usage
I1, I2, I3, I4	Read input logic state	I/O variable	RO	Program/Immediate

Description

Reads 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 example

`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 (outut setup)	OT Set all outputs)	

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x002D 0x002E 0x002F 0x0030
	0x67	1	0x09 — 0x0C		

5.1.57 I5 (Read Analog Input)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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 4096	Units	counts	Default	—
Syntax	PR I5, BR <label/address>, I5 =<integer>, CL <label/address>, I5=<integer>				

Code example

`PR I5`	Return the vslue 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)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0031
	0x67	1	0x0D		

5.1.58 I6 (Read Encoder Index)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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 would 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 example

`PR I6`	Read the value of the encoder index
`BR L2,I6=0`	Branch to labeled location L2 when I6 is zero
`CL Q5,I5=1`	Call subroutine Q5 when I6 is one

Related	IS (Input setup)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0037
	0x69	1	0x05		

5.1.59 I7 - I13 (Reserved)

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

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

5.1.60 IC (Increment Variable)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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) asserts an Error 26: attempting to increment or decrement an illegal variable.

Syntax	IC <var>

Code example

```
IC R1        Increment register 1
IC V2        Increment user variable V2
```

Related	DC (Decrement Variable)			

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.61 IF (Variable Input Pending)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

Description

The IF instruction is automatically set to 1 when 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 that IF may only be cleared, not manually set to 1.

Range	0/1	Units	—	Default	0
Syntax	IF=0				

Code example

`IF=0`	Clear the input variable pending flag

Related	IV (Input to Variable)			

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.62 IN (Read Inputs as group)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

Description

Reads the binary state of the inputs and returns them as a decimal value. When used thus, Input 1 is the Least Significant Bit (LSb) and Input 4* is the Most Significant Bit (MSb). It may be used in conjunction with PR (Print), BR (Branch) and CL (Call Subroutine) instructions.

The value is a function of the actual state of the IO 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.

** Lexium MDrive NEMA 17 (42 mm) products have only three inputs. In these products input 3 will be the MSb and the total range is IN=<0-7>*

Range	0 - 15	Units		Default	
Syntax	PR IN, BR <addr/lbl>,IN=<0-15>, CL <addr/lbl>, IN<0-15>				

Code example

`PR IN >07`	Print value of IO4-IO1
`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)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x003B
	0x67	1	0x0E		

5.1.63 IP (Initialize Parameters)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

Description

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

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

Syntax	IP

Code example

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

Related	FD (Factory Defaults)	S (Save)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	0x003C

5.1.64 IS <1-4> (Input Setup IN1-IN4)

Compatibility	LMD(O) LMD(C) LMM	Notes	Clock input types (13, 14, 15) are only available on models with Firmware 6.00.00+

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

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 response	0 (LOW active), 1 (HIGH active)	0 (LOW active)

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	See LM (Limit Response)
3	**Limit —**; functionality is defined by the LM instruction	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 the 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 the 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	

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

Code example

`IS=1,2,1`	Set 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`
	Retun 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)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	MFG specific function code, See MODBUS/ TCP manual, Section 4.3
	0x67	1	0x0F		

5.1.65 IS <5> (Analog Input Setup)

Compatibility	LMD(O) LMD(C) LMM	Notes	The analog input on the Lexium Motion Module is not configurable.

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

Description

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

Lexium MDrive

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)	

Lexium Motion Module

NOTE: The Lexium Motion Module 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 return an Error 24: Illegal data entered. Refer to the LMM hardware manual for example interface circuits.

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

Code example

`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 voltage mode with a 0 to 10V 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)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	MFG specific function code, See MODBUS/ TCP manual, Section 4.3
	0x67	1	0x0F		

5.1.66 IS <6> (Encoder Index Setup)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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)	

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

Code example

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

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

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	MFG specific function code, See MODBUS/ TCP manual, Section 4.3
	0x67	1	0x0F		

5.1.67 IT (Read Internal Temperature)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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

Range	-20 to 100	Units	°C	Default	—
Syntax	PR IT,<param>				

Code example

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,

Related	WT (Warning Temperature)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	MFG specific function code, See MODBUS/TCP manual, Section 4.3
	0x65	1	0x04		

5.1.68 IV (Input to Variable)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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.

Syntax	IV <var/reg>

Code example

IV used with conditional loop example

IV F1	Input numeric into floating point register 1
LB X2	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
BR X2,IF=1	

Related	IF (Variable Input Pending)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.69 JE (Jog Enable)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

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

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 example

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

Related	IS (Input Setup)			

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x003F
	0x66	1	0x05		

5.1.70 L (List Program Space)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
L	List the contents of program memory	Program instruction	—	Immediate

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.

Syntax	L <label/address>

Code example

`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)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.71 LB (Declare User Label)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
LB	Label Program or Subroutine	Program Instruction	—	Program

Description

The Label Instruction allows the user to assign a 2 character 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 192, 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 Lexium MCode Instruction, Variable, Flag or Keyword.
2. The first character must be alpha, the second character may be alpha-numeric.
3. A label is limited to two alpha characters or 1 alpha and 2 numeric characters
4. A program labeled SU will run on power-up
5. 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.

Syntax	LB <alpha><blank or alpha-num>

Code example

`LB G1`	Label program or location to G1

Related	BR (Branch)	CL (Call Subroutine)	EX (Execute Program)

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.72 LD (Lead Limit)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
LD	Lead limit	hMT Variable	RW	Program/Immediate

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.

Conditions causing the rotor position to lead the stator position:

1) Deceleration rate to high for load

Note that Lead 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	LD=<steps>, PR LD				

Code example

`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)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0095 –0x0096
	0x6A	1	0x06		

5.1.73 LG (Lag Limit)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
LG	Lag limit	hMT Variable	RW	Program/Immediate

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 interuption in the load due to load inertia change or mechanical changes in the system.

Note that 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 example

`LG=51200`	Set the lead limit for hMT to 51200 motor steps
`PR LG` `>51200`	Read the value of the lead limit

Related	LD (Lead Limit)	LL (Position Lead-Lag)	

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0097 –0x0098
	0x6A	1	0x08		

5.1.74 LL (Position Lead/Lag Count)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
LL	Position Lead/Lag Register	hMTechnology Variable	RO	Program/Immediate

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 that LL values are only measured when AS (hMTechnology Mode) is set to 1, 2, or

Range	-2147483647 to +2147483647	Units	motor steps	Default	—
Syntax	PR LL, CL <label/address>, LL<operator><steps>, BR <label/address>, LL<operator><steps>				

Code example

`CL k5,LL>102500`	Call k5 if LL is greater than 102500
`PR LL` `>0`	Read the value of the Lead/Lag register

Related	LD (Lead Limit)	LG (Lag Limit)	

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0099 – 0x009A
	0x6A	1	0x07		

5.1.75 LK (Lock User Program)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
LK	Lock user program space	Program Flag	RW	Program/Immediate

Description

LK may be used to prohibit user interaction with stored MCode programs by disallowing:

1) Program upload
2) Modification
3) Listing

Once enabled, attempting to list or modify the stored program space will assert an Error 44: User program space locked.

Once saved 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).

State	Description
0	**User program space unlocked (default)**
1	User program space locked

Range	0/1	Units	—	Default	0
Syntax	LK=<0/1>, PR LK				

Code example

`LK=1`	Lock user program space to prevent upload/list/modification
`PR LK` `>1`	Print the status of the LK flag.

Related	L (List Program Space)		

5.1.76 LM (Limit Response Mode)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
LM	Limit Response Mode	I/O Variable	RW	Program/Immediate

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.

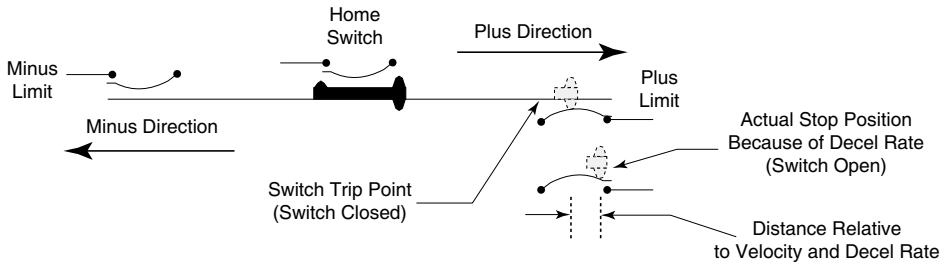


Figure 5.11 LM Limit response

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 example

`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)	LS (Soft Limits)	IS (Input Setup)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0042
	0x066	1	0x06		

5.1.77 LR (Locked Rotor)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
LR	Locked Rotor	hMT Status Flag	RO	Program/Immediate

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.

he 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 assert an error 104 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.

State	Description
0	**Rotor unlocked (default)**
1	Rotor locked

Range	0/1	Units	—	Default	0
Syntax	PR LR				

Code example

```
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)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x009B
	0x6A	1	0x09		

5.1.78 LS (Software Limits)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

Mnemonic	Function	Function Group	Access	Usage
LS	Set/Read ± Soft Limits	Motion Variable	RW	Program/Immediate

Description

Sets the direction, position and enabled state for software limit switches. There are three parameters:

Param	Description	Values	Default
1	Limit direction	0 (minus)	0
		1 (plus)	1
2	Position	-2147483648 to 214748364	0
3	Enable/disable	0 (disable)	0 (disabled)
		1 (enable)	

The first parameter provides the limit direction. The second parameter provides the position at which the limits will respond, note that the limits must have a logical gap, meaning that the negative limit must be set to a value more negative than the positive limit.

Finally 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.

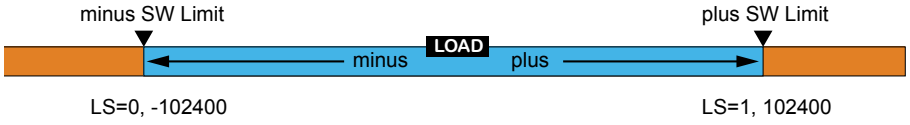


Figure 5.12 LS (Soft Limits)

Range	See table	Units	See table	Default	See table
Syntax	LS=<0/1>,<±position>,<0/1> | PR LS				

Code example

`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)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.79 LT (Locked Rotor Timeout)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
LT	Set/Read Locked Rotor Timeout	hMT Variable	RW	Program/Immediate

Description

Defines the time in milliseconds between the assertion of an LR (Locked Rotor) condition and the output H-bridges being disabled

Note that if the Lexium MDrive 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 example

`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)	

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0X009C – 0x009D
	0x6A	1	0x0A		

5.1.80 MA (Move Absolute)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
MA	Move to Absolute Position	Motion Instruction	—	Program/Immediate
Description				

Set mode for absolute move and move to an absolute position relative to (0) zero. MD (Motion Mode) will be set to MA.

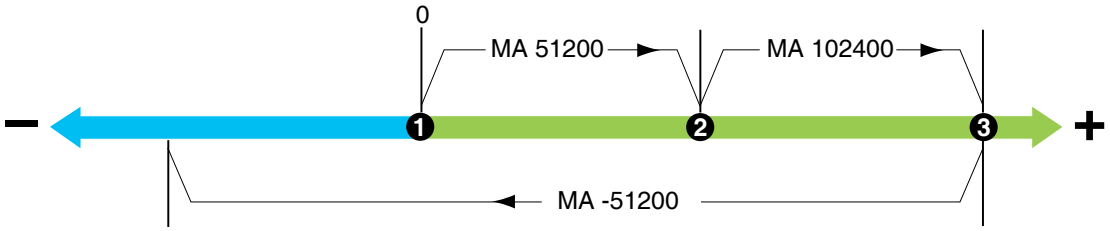


Figure 5.13 MA - move to an absolute position

MA moves the axis to a position in motor steps relative to zero (0). In the case of the profile shown in Figure 5:13 the end position of the first move (2) is + 51200, or 51200 motor steps from 0, a value which will be stored in the P (Position Counter).

The second move, MA 102400 moves the axis an additional 51200 steps, ending at position 3 or 102400 steps from 0. The third MA will index the axis 153600 steps in the negative direction from position 3, with a final position of -51200 absolute from 0.

The time required to calculate each move is 20 µSec.

NOTES:

The MA command will not operate during a homing sequence.

An in progress MA can be stopped with an [ESC] or an SL 0 command entry.

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 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 example

`MA 102400`	Move to absolute position 102400
`MA 1024900,0,1`	Move to absolute position 102400, do not stop motion upon position

Related	MD (Motion Mode)	MR (Move Relative)	P (Position Counter)
	SL (Slew)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0043-0x0044
	0x66	1	0x07		

5.1.81 MD (Motion Mode)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
MD	Motion Mode	Status Variable	RO	Program/Immediate

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	MA. MR. SL	Units	—	Default	—
Syntax	PR MD				

Code example

```
MA 51200
PR MD
   MA
```

Move, then return the previous move type

Related	MA (Move Absolute)	MR (Move Relative)	NE (Numeric Enable)
	SL (Slew)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.82 MF (Make-up Frequency)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
MF	Read/Set hMT Make-up Frequency	hMT Variable	RW	Program/Immediate

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.

When used as a keyword with the PR (Print) command it will return the stored value for MF.

Range	92 to VM	Units	motor steps/sec	Default	768000
Syntax	MF=<steps/sec>, PR MF				

Code example

`MF=512000`	Set hMT make-up to 512000 steps/sec
`PR MF` `512000`	Read the current value of MF

Related	AS (hMT Mode)	MU (Make-up Mode)	

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x009E – 0x009F
	0x6A	1	0x0B		

5.1.83 MP (Moving to Position)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
MP	Moving to a Position	Status Flag	RO	Program/Immediate

Description

Read-only status flag is active (1) when the axis is indexing to a position.

Example use: wait subroutine while positional moves are in process.

Note that 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 example

```
MA 5120000
PR MP
   1
```

Make a positional move, return the MP status while axis is moving,

Related	MA (Move Absolute)	MR (Move Relative)	

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0045
	0x66	1	0x08		

5.1.84 MR (Move Relative)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
MR	Move to Relative Position	Motion Instruction	—	Program/Immediate
Description				

Set mode for relative move and move to a position relative to the current position. MD (Motion Mode) will be set to MR.

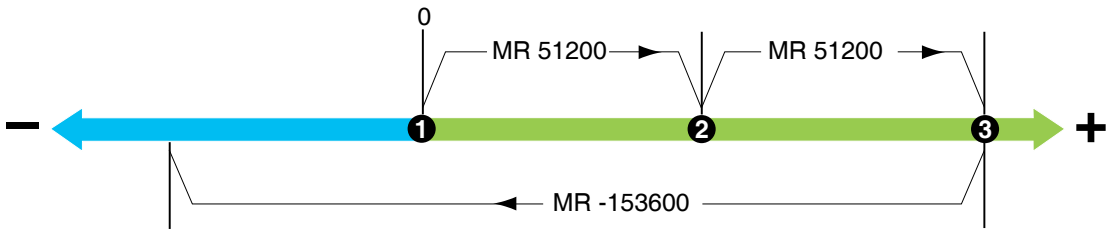


Figure 5.14 MR - move to a relative position

MR moves the axis to a position in motor steps relative to the current motor position, zero (0). In the case of the profile shown in Figure 5:14 the end position of the first move (2) is + 51200, or 51200 motor steps from 0, a value which will be stored in the P (Position Counter).

The second move, MR 51200 moves the axis an additional 51200 steps, ending at position 3 or 102400 steps from 0. The third MR 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.

The time required to calculate each move is 20 µSec.

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.

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 response	0 - no operation	0
		1 - send DN (Device Name) out the communications port following move completion. DN (Device Name) will be sent whether or not the device is in party mode (PY=1)	
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 example

`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	MD (Motion Mode)	MA (Move Absolute)	P (Position Counter)
	SL (Slew)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0046-0x0047
	0x66	1	0x09		

5.1.85 MS (Microstep Resolution)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
MS	Set/read Microstep Resolution	Motion Variable	RW	Program/Immediate

Description

Sets the Microstep Resolution for the device. There are 20 fixed microstep resolutions that the Lexium 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 (Mcrostep 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 table below is based upon the Lexium MDrive products with 1.8° (200 Step/Rev) motor. If using a Lexium 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 Lexium MDrive products. If using a custom integrated product, or a Lexium 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 example

`MS=4`	Set microstep resolution to 4 steps/full step
`PR MS` `4`	Return the microstep resolution setting

Related			
	A (Acceleration)	D (Deceleration)	MA (Move Absolute)
	MR (Move Relative)	SA (Step Angle)	SL (Slew at Velocity)
	VI (Initial Velocity)	VM (Maximum Velocity)	

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0048
	0x66	x	0x0A		

5.1.86 MT (Motor Settling Delay)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
MT	Set/Read Motor Settling Delay	Motion Variable	RW	Program/Immediate

Description

Delay in milliseconds given to allow the motor to settle into position following a move., The delay time is also impacted by the HT (Hold Current 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.

Thus the maximum setting for MT=(65535-HT). 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 returns an Error 21: Illegal data value entered.

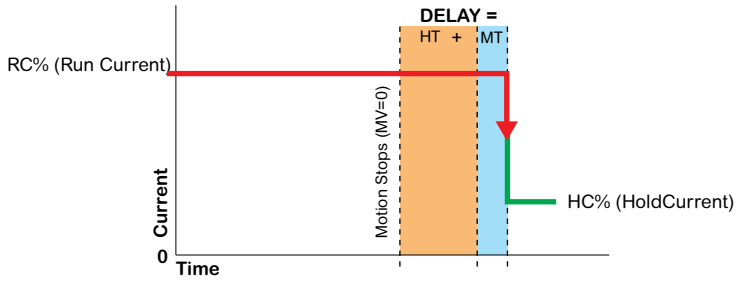


Figure 5.15 MT (Motor Settling Delay) and HT (Hold Current Delay) relationship

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 example

`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)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0049
	0x66	1	0x0B		

5.1.87 MU (Make-up Mode)

NOTE: Make-up is an advanced hMT function covered in detail in Section 8: hMTechnology, of this document

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
MU	Set/Read Make-Up Mode for hMT	hMT Variable	RW	Program/Immediate

Description

Defines the mode for hMTechnology position make-up. Make-up only occurs when motor lag/lead is within 1.1 motor steps. Make up steps may be interleaved with motion steps and made after a move has completed.

Where make-up occurs is dependant on motor lag/lead, motion frequency and selected make up speed.

Make up mode will be cleared when bridges are disabled and hMTechnology 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 , an internally defined velocity limited to 2560000 steps/sec (3000 RPM) as make up frequency

Range	0 to 2	Units	—	Default	0
Syntax	MU=<mode>				

Code example

`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 (hMTechnology Mode)	MF (Make-up Frequency)	

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x00A0
	0x6A	1	0x0C		

5.1.88 MV (Moving)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
MV	Axis is Moving	Status Flag	RO	Program/Immediate

Description

Read-only status flag is active (1) when the axis is moving, regardless of the move type.

Note that 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
1	Moving

NOTES:

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 (Lexium MDrive 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 MVP				

Code example

```
SL 51200
PR MV
   1
```

Slew the axis, return the MV status,

Related	MA (Move Absolute)	MR (Move Relative)	OS (Output Setup)
	SL (Slew)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x004A
	0x66	1	0x0C		

5.1.89 NE (Numeric Enable/Disable)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
NE	Enable/disable Numeric Functions	Setup Variable	RW	Program/Immediate

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.

If disabled, the user must enter a motion command to execute a move, i.e. MA 100000, MR -50000, SL 300000 etc.

Value	Description
0	Disabled (default)
1	Numeric functions enabled

Range	0/1	Units	—	Default	0
Syntax	NE=<0/1>, PR NE				

Code example

`NE=1`	Enable numeric functions
`PR NE` `1`	Return the numeric enable state numeric functions are enabled

Related	MA (Move Absolute)	MD (Motion Mode)	MR (Move Relative)
	SL (Slew)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.90 O1, O2, O3 (Set Output)

Compatibility	LMD(O) LMD(C) LMM	Notes	Outputs 1 and 2 are not available on NEMA 17 (42 mm) models.

Mnemonic	Function	Function Group	Access	Usage
O1/O2/O3	Set Output #	I/O Instruction	RO	Program/Immediate

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).

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

NOTES:

On LMDxM42x or LMDxE42x (NEMA 17) Outputs 1 and 2 are not present. Use of this command will return an Error 37: Command, variable or flag not available.

Range	0/1	Units	—	Default	—
Syntax	O<1/2/3>=<0/1>				

Code example

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)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	
	0x67	1	0x10 (O1) 0x11 (O2) 0x12 (O3)		0x004B (O1) 0x004C (O2) 0x004D (O3)

5.1.91 OE (On Error Handler)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
OE	On Error Handler	Program Instruction	RW	Program/Immediate

Description

OE declares the label or address of the subroutine which will execute when an error code ER (Error Code) is asserted and EF (Error Flag) activated.

Attempting to target OE to a non-existent subroutine will throw an Error 30: Unknown User Label or Variable.

Standard rules for subroutines apply to subroutines called by OE: and RT must be inserted at the end of the subroutine, 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.
4) OE will not execute during programming

Syntax	OE=<label/address>

Code example

`OE Q1`	Execute subroutine Q1 when an Error is asserted

Related	EF (Error Flag)	ER (Error Code)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.92 OF (Output Fault)

Compatibility	LMD(O) LMD(C) LMM	Notes	Applicable to Outputs 1 & 2 only Not applicable to LMM products

Mnemonic	Function	Function Group	Access	Usage
OF	Output Over Current Fault	I/O Variable	RO	Program/Immediate

Description

Read-only status variable indicates an over-current fault condition on the power outputs (Outputs 1 and 2).

Though an Error code 1 or 2 will also be asserted, read the Output Fault from OF, as the ER (Error Register) will only hold the last asserted error, indicating a single output fault condition when in fact both outputs may 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 example

`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)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x004E
	0x67	1	0x13		

5.1.93 OS <1-3> (Output Setup OUT1 - OUT3)

Compatibility	LMD(O) LMD(C) LMM	Notes	Lexium MDrive NEMA 17 (42 mm) models are equipped with OUT 3 (Signal Output) only Some output functions are hMTechnology specific and not available on all products.

Mnemonic	Function	Function Group	Access	Usage
OS	Setup Outputs 1 to 3	I/O Instruction	RW	Program/Immediate

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.

Output parameters

Param	Description	Values	Default
1	Output line number	1 - 3	—
2	Output function type	(see type table)	16 (General Purpose User)
3	Output active	0 (LOW active), 1 (HIGH active)	0 (LOW active)

Input 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) 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	**hMTechnology Active:** active whenever hMTechnology is compensating for load variances.	See AS (hMTechnology Mode) **[hMT LMD ONLY]**
25	**Make-up Steps Active:** active whenever hMTechlogy 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 MCode Trip Functions
29	**Attention:** active with regard to the AO (Attention Output Mask) setting.	See AO (Attention Output Mask)

Lexium MDrive 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.

Some output functions are hMTechnology specific and not available on all products. Such variances are noted in the type table on the previous page.

Syntax	OS=<1-3>,<type>,<active> | PR OS

Code example

`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)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	**Modbus/TCP**	MFG specific function code, See MODBUS/ TCP manual, Section 4.3
	0x67	1	0x14		

5.1.94 OT (Set Output Total)

Compatibility	LMD(O) LMD(C) LMM	Notes	Lexium MDrive NEMA 17 (42 mm) models are equipped with OUT 3 (Signal Output) only.

Mnemonic	Function	Function Group	Access	Usage
OT	Set the state of all outputs	I/O Instruction	RW	Program/Immediate

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=				

Code example

OT=7	Set Output total to 7, all outputs will be active

Related	O<1-3> (Set Output)	OS (Output Setup)	

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0056
	0x67	1	0x015		

5.1.95 P (Position Counter)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
P	Position Counter	Instruction	RW	Program/Immediate

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 in essence 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		Motor steps		—
	EE=1	-2147483648 to +2147483647		Encoder counts		
Syntax		P=<counts>, PR P, <CL/BR> <label/address>, P=<value>				

Code example

`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)	

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0057-0x0058
	0x68	1	0x03		

5.1.96 PC (Position Capture at Trip)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
PC	Position Capture at Trip	Instruction	RW	Program/Immediate

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)

Syntax	PR PC

Code example

`PR PC` `0`	Return the captured position count the captured position count is zero

Related	TA (Trip on hMT Status)	TC (Trip Capture)	TE (Trip Enable)
	TI (Trip on Input)	TM (Trip on Main Power Loss)	TT (Trip on Time)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0059 - 0x005A
	0x68	1	0x04		

5.1.97 PF (Print Format)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

Mnemonic	Function	Function Group	Access	Usage
PF	Set Print Format for Floating Point Registers	System Variable	RW	Program/Immediate

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
3	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 example

`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 sci notation.

Related	F<1-8>(Floating Point Register)	PR (Print)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.98 PG (Program Mode)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
PG	Enter/Leave Program Mode	Program Instruction	—	Program/Immediate

Description

Toggles the device into or out of program mode.

Syntax	PG <address>

Code example

`PG 1`
`[MAIN PROG]`
`[SUBROUTINES]`
	Enter program mode at address 1

`PG` `E`	Exit program mode designated end of program

Related	CP (Clear Program Memory)	FD (Factory Defaults)	

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	—
	—	—	—		

5.1.99 PK (Reserved)

Compatibility	LMD(O) LMD(C) LMM	Notes	

Mnemonic	Function	Function Group	Access	Usage
PK	Null	Reserved	—	—
Description				

Reserved for factory/future use. DO not use as a user label or variable.

5.1.100 PM (Position Maintenance)

Compatibility	LMD(O) LMD(C) LMM	Notes	Encoder required

Mnemonic	Function	Function Group	Access	Usage
PM	Position Maintenance enable/disable	Encoder Flag	RW	Program/Immediate

Description

Enables the position maintenance functions of an Lexium 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 disable

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

Position maintenance is not to be confused with hMTechnology 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 example

`PM=1`	Enable position maintenance
`PR PM` `1`	Return the status of position maintenance position maintenance is enabled

Related	C2 (Encoder Counter)	DB (Encoder Deadband)	EE (Encoder Enable)
	SF (Stall Factor)	SM (Stall Detect Mode)	

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x005C
	0x69	1	0x06		

5.1.101 PN (Part Number)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
PN	Read Part Number	Identification variable	RO	Program/Immediate
Description				

Read only register holds the factory defined part number.

Syntax	PR PN

Code example

```
PR PN
    LMDCM571
```

Return the stored part number
Lexium MDrive Motion Control NEMA 23 (57 mm)

Related	SN (Serial Number)	VR (Version)	

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	MFG specific function code, See MODBUS/TCP manual, Section 4.1
	0x65	1	0x05		

5.1.102 PR (Print specified data and/or text)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
PR	Print Text and/or Data	System Instruction	RW	Program/Immediate

Description

Outputs text and parameter value(s) to the communications host. Text strings are enclosed in quotation marks while parameters (variables and flags) should not. Text strings and parameters which are to be output by the same PR instruction should be separated by commas. The information being output is followed by a carriage return unless a semicolon (;) is included at the end of the PR instruction to indicate that the cursor should remain on the same line.

It is important to note that the receive buffer for the Lexium 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 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. They must be enclosed within quotes, for example PR P, " motor steps\r" would terminate a string requesting the axis position with a carriage return.

The table below 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

Syntax	PR <var/flg/keyword>, PR "<text> ", <var/flg/keyword>

Code example

`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)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.103 PS (Pause Program)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
PS	Pause executing program	Program Instruction	—	Program/Immediate

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.

Syntax	PS

Code example

PS	Pause executing program

Related	E (End Program)	EX (Execute Program)	PG (Program Mode)
	RS (Resume Paused Program)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	See Section 4.3.9: Pause Program in the Modbus/ TCP Fieldbus Manual

5.1.104 PW (PWM Mask)

Compatibility	LMD(O) LMD(C) LMM	Notes	Lexium Motion Module only

Mnemonic	Function	Function Group	Access	Usage
PW	PWM Mask Setting	Configuration Variable	RW	Program/Immediate

Description

The PW variable is only used on the Lexium Motion Module product only. It is not a reserved word on the Lexium MDrive products and may be used as a user variable or label.

This variable is used to set the PWM current control settings of the LMM ONLY! It does not apply in any function to the Lexium MDrive and may be used as a label or user variable or flag. See Section 7: Programming and Applications Notes of this document for parameter settings and usage.

The PW variable is defaulted to SEM NEMA 17 (42 mm) motors. Recommended settings for additional motor sizes offered by SEM are located in Section 7 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 Section 7	Units	See Section 7	Default	
Syntax	PW=<mask>,<period>,<sfreq>,<ctrl> | PR PW				

Code example

See Section 7

Related	—		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.105 PY (Party Mode)

Compatibility	LMD(O) LMD(C) LMM	Notes	See LMM Note below

Mnemonic	Function	Function Group	Access	Usage
PY	Party Mode Enable/disable	Communications Variable	RW	Program/Immediate

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 by using the device name specified by the DN instruction. This name will precede any command given to a specified unit in the system and be terminated with a Control J (CTRL + J). One CTRL + J must be issued after power up or entering the Party Mode to activate the Party Mode. By default the DN assigned at the factory is the exclamation character (!) .

The global Drive Name is the asterisk character (*). Commands preceded by this character will be recognized by every Lexium MCode compatible device in the system.

After the Party Mode is enabled, send CTRL + J (^J) to activate it. Type commands with Device Name (DN) and use CTRL + J as the Terminator.

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

LMM Note:
The Lexium Motion Module features hardware inputs for device name (address). When ever any og these inputs is active, the LMM will automatically enable Party Mode.

Range	0/1	Units	—	Default	0
Syntax	PY=<0/1>, PR PY				

Code example

`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)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.106 QD (Queued)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
QD	Device Queued	Comm Flag	RW	Program/Immediate

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	o/1	Units	—	Default	0
Syntax	<dn>QD=<0/1>, <dn>PR QD				

Code example

`!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)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.107 R1-R4 (User Register)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
R<1-4>	User Integer Register	Mathematics Variable	RW	Program/Immediate

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	-2147483647 to 2147483647	Units	—	Default	0
Syntax	R<1-4>=<integer>, R1=<var>, R<1-4>=R<1-4><MATH><R<1-4>, PR R1				

Code example

`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)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	
	0x65	1	R1: 0x06 R2: 0x07 R3: 0x08 R4: 0x09		R1: 0x005F - 0x0060 R2: 0x0061 - 0x0062 R3: 0x0063 - 0x0064 R4: 0x0065 - 0x0066

5.1.108 RA (Radians or degrees)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

Mnemonic	Function	Function Group	Access	Usage
RA	Set Radians or degrees	Configuration Variable	RW	Program/Immediate

Description

Selects the Radians or Degrees as the units for trigonometric calculations. When used as a keyword with the PR (Print) statement it will return the setting to the terminal.

Value	Description
0	Degrees
1	Radians (default - faster)

Range	0/1	Units	—	Default	
Syntax	RA=<0/1> | PR RA				

Code example

`RA=0`	Calculate trigonometric functions in degrees
`PR RA` `0`	Read the units for trig functions trig functions calculate in degrees

Related	F1-F8 (Floating Point)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.109 RC (Run Current)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
RC	Motor Running Current	Motion variable	R/W	Program/Immediate

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 Lexium MDrive products the current is only given in a percentage range as the driver is already sized and tuned to the integrated motor.

The Lexium 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	HC=<percent>, PR HC				

Code example

`RC=75`	Set RC (Run Current) to 75%
`PR RC`	Read the value of the holding current

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

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0029
	0x66	1	0x03		

5.1.110 RD (Rotation of Direction)

⚠WARNING

UNINTENDED MOTION

Use of the RD command in Lexium Motion product or Ethernet (Closed Loop models) with firmware versions 5.007 or earlier may, under certain conditions, result in unintended motion.

- Upgrade the device firmware to 5.009 or greater.

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

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
RD	Rotation of Direction	Motion Variable	RW	Program/Immediate

Description

This variable, when TRUE will reverse the default +/- motor direction reference. Cannot be issued when the axis is in motion or error 95 will be asserted.

Value	Description
0	Default +/- direction (default)
1	Direction reversed

Range	0/1	Units	—	Default	0
Syntax	RD=<0/1>, PR RD				

Code example

`RD=1`	Reverse axis direction reference
`PR RD` `1`	Read the value of RD RD is true, the +/- direction is reversed

Related	—		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	See 4.3 Manufacturer specific function codes in the Modbus/TCP Fieldbus Manual
	0x66	1	0x13		

5.1.111 RS (Resume Program Execution)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
RS	Resume Program Execution	Program Instruction	—	Immediate

Description

Resumes and executing 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.

Syntax	RS

Code example

```
RS          Resume paused program
```

Related	E (End Program)	EX (Execute Program)	PG (Program Mode)
	PS (Pause Program Execution)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	See 4.3 Manufacturer specific function codes in the Modbus/TCP Fieldbus Manual

5.1.112 RT (Return From Subroutine)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
RT	Return From Subroutine	Program Instruction	—	Program

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. When used, it will return to the program address immediately following the instruction which executed the subroutine.

Syntax	RT

Code example

```
RT          Return from Subroutine
```

Related	CL (Call Subroutine)	OE (On Error Handler)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.113 S (Save to FLASH)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
S	Save Programs and Parameters)	System Instruction	RW	Program/Immediate

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. If a save is performed during the execution of a motion command, trips may be delayed.

Use of the S command during a move (MA or MR) will generate an error 73, the save will 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.

Syntax	S

Code example

`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	—		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	0x0076

5.1.114 SA (Step Angle)

Compatibility	LMD(O) LMD(C) LMM	Notes	LMM only

Mnemonic	Function	Function Group	Access	Usage
SA	Set/Read Step Angle	Motion Variable	RW	Program/Immediate

Description

Step angle is a floating point variable to configure the step angle of the motor for the Lexium Motion Module only.

The setting is represented by the equation: Motorcts = MS * (360/SA)

Ex:
MS = 256
SA = 0.9
MtrCts = 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	Units	Degrees	Default	1.8
Syntax	SA=<angle>, PR SA				

Code example

`SA=0.9`	Set step angle for 0.9 degree motor.
`PR SA` `0.900000`	Return the step angle setting The step angle is 0.9 degrees.

Related	PW (Motor PWM Settings)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.115 SC (System Configuration Test)

Compatibility	LMD(O) LMD(C) LMM	Notes	LMD Closed loop only.

Mnemonic	Function	Function Group	Access	Usage
SC	Start Configuration Test	System instruction	RW	Program/Immediate

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 an error.

Syntax	SC 1

Code example

SC 1	Start configuration test

Related	—		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x00A1
	0x6A	1	0x0D		

5.1.116 SF (Stall Factor)

Compatibility	LMD(O) LMD(C) LMM	Notes	LMD Closed loop LMM with encoder

Mnemonic	Function	Function Group	Access	Usage
SF	Set/Read Stall Factor	Encoder Variable	RW	Program/Immediate

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 is set to 0, then the motor will be stopped when a stall is detected. If SM=1, the motor will not be stopped upon detection of a stall. ST will return an Error 86 on stall.

Range	0 to 65000	Units	Encoder counts	Default	15
Syntax	SF=<counts>, PR SF				

Code example

`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)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0077
	0x69	1	0x07		

5.1.117 SL (Slew at Velocity)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
SL	Slew Axis at Velocity	Motion Instruction	RW	Program/Immediate

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 that 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	±5000000 (EE=0)/±200000 (EE=1)	Units	Motor Steps (EE=0)/Encoder Counts (EE=1)
Syntax	SL <velocity.		

Code example

`SL 20000`	Slew axis at 2000 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)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0078
	0x66	1	0x0E		

5.1.118 SM (Stall Detect Mode)

Compatibility	LMD(O) LMD(C) LMM	Notes	LMD Closed loop LMM with encoder

Mnemonic	Function	Function Group	Access	Usage
SM	Set/Read Stall Detect Mode	Encoder Variable	RW	Program/Immediate

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. 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 example

`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)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x007A
	0x69	1	0x08		

5.1.119 SN (Serial Number)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
SN	Read Serial Number	Keyword	RO	Program/Immediate

Description

Allows user to read the deviece serial number using the PR (Print) statement.

Syntax	PR SN

Code example

`PR SN`	Return the serial number

Related	PN (Part Number)	VR (Version)	

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	See Section 4.1 in the Modbus/TCP Fieldbus Manual
	0x65	1	0x0A		

5.1.120 ST (Stall Flag)

Compatibility	LMD(O) LMD(C) LMM	Notes	LMD Closed loop LMM with encoder

Mnemonic	Function	Function Group	Access	Usage
ST	Stall Flag	Encoder Flag	RW	Program/Immediate

Description

The stall flag is set active (1) when the motor stalls. An Error 86 will also be asserted.

It is important to note that the Stall Flag must be manually reset to 0 (ST-0) clearing the error state will not clear the stall flag. The product will respond to motion commands while the ST flag is active. A subroutine triggered by the OE (One Error) instruction containing

Encoder functions must be 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 example

`ST=0`	Clear the state of the stall flag
`PR ST` `0`	Read the value of the stall flag no stall condition exists

Related	EE (Encoder Enable)	OE (On Error)	SF (Stall Factor)
	SM (Stall Mode)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x007B
	0x69	1	0x09		

5.1.121 SU (Execute Program on Startup)

⚠ DANGER

UNINTENDED CONSEQUENCES OF EQUIPMENT OPERATION

Programs labeled with the SU label will execute on sytem power application or software reset. Depending on the program structure this could result in immediate motion on power application or system restart.

• 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.

Compatibility	■ LMD(O) ■ LMD(C) ■ LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
SU	Execute on Startup	Factory Label	RW	Program/Immediate

Description

The Start up label will cause any program labeled SU to automatically execute on power-up.

Syntax	LB SU

Code example

```
LB SU          Label program to execute on startup
```

Related	—		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.122 TA (Trip on hMT Status)

Compatibility	LMD(O) LMD(C) LMM	Notes	LMD Closed Loop only

Mnemonic	Function	Function Group	Access	Usage
TA	Trip on hMT Status	Trip Variable	RW	Program/Immediate

Description

Executes a subroutine address or label on the trip. The trip can be set to occurs on any or all of three conditions: calibration done, hybrid active, locked rotor or lead/lag limit reached conditions.

Param	Description
0	Off
1	Calibration done
2	hMTechnology active
4	Locked roto
8	Lag limit reached
16	Lead limit reached

The conditions are additive, eg. TA=3 will trip on calibration complete and hybrid active status.

There is no error generation when enabling trip on locked rotor, lag limit or lead limit.

Range	0 - 7	Units	—	Default	0
Syntax	TA=<label/address>,<0-7>				

Code example

`TA=4, k6`	execute subroutine k6 when there is a locked rotor condition

Related	TE (Trip Enable)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	See Section 4.3 in the Modbus/ TCP Fieldbus Manual

5.1.123 TC (Trip on Capture)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
TC	Trip Capture	Trip Variable	RW	Program/Immediate

Description

Sets the Capture input trip for input 1. Sets one parameter for trip address. The TE command (Trip Enable/ Disable TC) is reset when trip occurs. TE must be re-enabled in the main program prior to the next trip if it is to be repeated. The Trip subroutine must use a RETURN (RT) to exit the subroutine, use of a BRANCH will cause stack errors..

Syntax	TC=<label/address>

Code example

`TC=K1`	Run subroutine K1 on capture

Related	IS (Input Setup)	TE (Trip Enable)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.124 TD (Torque Direction)

Compatibility	LMD(O) LMD(C) LMM	Notes	LMD Closed Loop only

Mnemonic	Function	Function Group	Access	Usage
TD	Read/Set Torque Direction	hMT Variable	RW	Program/Immediate

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 example

`TD=0`	Switch torque direction to minus
`PR TD` `1`	Return the torque direction Torque direction is minus

Related	AS (hMT Mode)	TQ (Torque)	TS (Torque Speed)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x00A5
	0x6A	1	0x0E		

5.1.125 TE (Trip Enable)

Compatibility	LMD(O) LMD(C) LMM	Notes	Not all trip functions are available with all products

Mnemonic	Function	Function Group	Access	Usage
TE	Set/Read Trip Enable	Variable	RW	Program/Immediate

Description

The 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 function needs to be re-enabled, the other trips will still be 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

NOTES: A trip must be defined prior to being enabled. Enabling an undefined trip will throw an Error 27: Trip not defined

Range	0 - 127	Units	—	Default	0 (disabled)
Syntax	TE=<param> | PR TE				

Code example

`TE=127`	Enable all trip functions
`PR TE` `127`	Return enabled trips 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)	TR (Trip on Relative Position)	TT (Trip on Time)

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.126 TI (Trip on Input)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
TI	Trip on Input	Trip Variable	RW	Program/Immediate

Description

Sets up an input event (Trip) for the specified input. There are two parameters for the TI variable. The first specifies which input line to monitor. The second specifies the subroutine that should be executed when the input goes to true. The Trip subroutine must use a RETURN (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.

Syntax	TI <input>,<label/address>

Code example

`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)	TR (Trip on Relative Position)	TT (Trip on Time)

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	See Section 4.3 in the Modbus/TCP Fieldbus Manual

5.1.127 TM (Trip on Main Power Loss)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
TM	Trip on Main Power Loss	Trip Variable	RW	Program/Immediate

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.

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.

Syntax	TM=<label/address>

Code example

`TM=Q1` `TE=64`	Execute Q1 on loss of main power (Re)enable trip

Related	TE (Trip Enable)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.128 TP (Trip on Position)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
TP	Trip on Position	Trip Variable	RW	Program/Immediate

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.

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.

Syntax	TP=<position>,<label/address>

Code example

`TP=51200, Q1` `TE=2`	Set trip to trigger Q1 at 51200 steps (Re)enable trip

Related	P (Position Counter)	TE (Trip Enable)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	See Section 4.3 in the Modbus/TCP Fieldbus Manual

5.1.129 TQ (Torque Percent)

Compatibility	LMD(O) LMD(C) LMM	Notes	LMD Closed Loop only

Mnemonic	Function	Function Group	Access	Usage
TQ	Read/Set Torque	hMT Variable	RW	Program/Immediate

Description

Sets the maximum out put torque of the motor to a percentage.

Range	1 — 100	Units	% (percent)	Default	25
Syntax	TQ=<percent> | PR TQ				

Code example

`TQ=50`	Set the Torque to 50
`PR TQ` `50`	Read the value of TQ The torque is set to 50%

Related	AS (hMT Setting)	TD (Torque Direction)	TS (Torque Speed)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x00A6
	0x6A	1	0x0F		

5.1.130 TR (Trip on Relative Position)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
TR	Trip on Relative Position	Trip Variables	RW	Program/Immediate

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 the High Speed Trip Output will activate. The Trip subroutine must use a RETURN (RT) to exit the subroutine, use of a BRANCH will cause stack errors

The third parameter specifies the number of times the trip will repeat. If 0 (default) the trip will repeat infinite times, other wise the range is 1- 65000

The TE (Trip Enable which Enables/Disables TR) is reset after repeating the number of relative trips specified. TE must be re-enabled in the main program prior to the next series of Trip on Relative if it is to be repeated. For exampl, if TR=10000,0,25, 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).

Trips should be set BEFORE motion commands in the program.

Note: Output 1 must be configured as a trip output (Os=3,28,0)

Note that TR will always use motor counts unless the encoder is enabled (EE=1).

Note: The maximum rate of trip is 20 kHz. Exceeding this may cause communications errors

Note that only a single position trip type may be used at a time. TR cannot be used simultaneously with TP

Syntax	TR=<postition>,<label/address>, <repeat>

Code example

`TR=512000,Q1,15` `TE=16`	Set TR to Trip every revolution for 15 revolution (Re)enable Trip

Related	TE (Trip Enable)	TP (Trip on Position)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	See Section 4.3 in the Modbus/TCP Fieldbus Manual

5.1.131 TS (Torque Speed)

Compatibility	LMD(O) LMD(C) LMM	Notes	LMD Closed Loop only

Mnemonic	Function	Function Group	Access	Usage
TS	Set/Read Torque Speed	hMT Variable	RW	Program/Immediate

Description

Defines the system speed for Torque mode (AS=3). This configuration variable will only take effect if hMT is in torque mode.

Note that the value for TS may be changed while the axis is in motion, though changing velocity during a torque move may throw an Error 106: Reached Lead Limit count.

Range	46,512 — 2560000	Units	steps/sec	Default	0
Syntax	TS=<steps/sec>				

Code example

`TS=51200`	Set torque speed to 51200 steps per second
`PR TS` `51200`	Read the value of TS TS is 51200 steps/sec

Related	AS (hMTechnology Mode)	TD (Torque Direction)	TQ (Torque Percent)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x00A7
	0x6A	1	0x10		

5.1.132 TT (Trip on Time)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
TT	Trip on time	Trip Variable	RW	Program/Immediate

Description

Sets up a trip based on time. The first parameter is time in mSec. The second parameter specifies the subroutine that should be executed when the time is expired. The Trip subroutine must use a RETURN (RT) to exit the subroutine, use of a BRANCH will cause stack errors

Range	1 to 65535	Units	milliseconds	Default	—
Syntax	TT=<time>,<label/address>				

Code example

`TT=10000,Q1` `TE=8`	Execute subroutine Q1 every 10 seconds Enable trip

Related	TE (Trip Enable)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	See Section 4.3 in the Modbus/TCP Fieldbus Manual

5.1.133 UG (Firmware Upgrade)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
UG	Process Firmware Upgrade	Upgrade Firmware	RW	Program/Immediate

Description

The upgrade command and code will be automatically entered by the Upgrader Utility in the Motion Control Interface or SEM Terminal software programs.

Once initiated, the firmware Upgrade MUST be completed.

Syntax	UG 2656102

Code example

UG 2956102	Enter upgrade mode

Related	VR (Version)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	—
	0x66	1	0x0E		

5.1.134 UV (Read User Variable)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
UV	Read User Variable	Keyword	RO	Program/Immediate

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

Syntax	PR UV

Code example

`VA Q1=25`
`PR UV`
`  Q1 = G 25`
	Create user variable Q1 and set to 25
	Read user variables, scope and values
	  Q1 is a global variable with a value of 25

Related	PR (Print)	VA (Create User Variable)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	—

5.1.135 V (Read Axis Velocity)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
V	Read Axis Velocity	Keyword	RO	Program/Immediate

Description

Keyword used with the PR (Print) command to read the current velocity of the axis velocity. The value of V is signed based on the direction of motion.

NOTE: V will not return an accurate value if hMTechnology is active. In Torque Mode, (AS=3), V will return a zero value.

Syntax	PR V | BR <label/address>,V=<value> | CL <label/address>, V=<value>

Code example

```
VA Q1=25
PR UV
  Q1 = G 25
```

Create user variable Q1 and set to 25
Read user variables, scope and values
Q1 is a global variable with a value of 25

Related	MA (Move Absolute)	MR (Move Relative)	SL (Slew at Velocity)
	VI (Initial Velocity)	VM (Maximum Velocity)	

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	0x0085

5.1.136 VA (Define User Variable)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
VA	Define User Variable	Keyword	RO	Program/Immediate

Description

The VA instruction creates a user variable with a 1 or 2 character name. Can optionally set value assigned to that variable.

The restrictions for this command are:

1. A variable cannot be named after a Lexium MCode Instruction, Variable or Flag or Keyword
2. The first character must be alpha, the second character may be alpha-numeric.
3. A variable is limited to two characters.
4. Limited to 192 variables and labels.

Syntax	VA<char><char>=<value>

Code example

`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)		

Networking protocol equivalents

EtherNet/IP	—	Modbus/TCP	0x0085

5.1.137 VC (Velocity Changing)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
VC	Read Velocity Changing	Status Flag	RW	Program/Immediate

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.

Param	Description
0	Minus (CCW facing shaft)
1	**Plus (CW facing shaft) (default)**

An output may be set to be ON when VC is active using OS=<output>,20,<active>.

Range	0/1	Units	—	Default	0
Syntax	PR VC | {BR/CL} <label/address>,VC=<state>				

Code example

`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)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0088
	0x66	1	0x10		

5.1.138 VF (hMT Velocity Filter)

Compatibility	LMD(O) LMD(C) LMM	Notes	LMD Closed Loop

Mnemonic	Function	Function Group	Access	Usage
VF	Read/Set hMT Velocity Filter	hMT Variable	RW	Program/Immediate

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 example

`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)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x00A7
	0x6A	1	0x11		

5.1.139 VI (Initial Velocity)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
VI	Set/Read Initial Velocity	Motion Variable	RW	Program/Immediate

Description

Initial velocity for all motion commands. The factory default value is 1000 clock pulses (steps) per second.

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).

Range	1 to (VM –1)	Units	steps/sec (EE=0) counts/sec (EE-1)	Default	1000 (EE=0) 40 (EE-1)
Syntax	VI=<velocity> | PR VI				

Code example

`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)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x0089 - 0x008A
	0x66	1	0x11		

5.1.140 VM (Maximum Velocity)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
VM	Read/Set Maximum Velocity	Motion Variable	RW	Program/Immediate

Description

The VM variable specifies the maximum velocity in steps/counts per second that the axis will reach during a move command.

The maximum setting of VM is dependant 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	steps/sec (EE=0)	Default	786000 (EE=0)
			counts/sec (EE-1)		307200 (EE-1)
Syntax	VM=<velocity> | PR VM				

Code example

`VM=500000`	Set maximum velocity to 5000000 steps sec.
`PR VM` `5000000`	Read the value of VM VM is set to 500k steps/sec

Related	A (Acceleration)	D (Deceleration)	VI (Initial Velocity)

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	0x008B
	0x66	1	0x12		

5.1.141 VR (Version)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
VR	Read Firmware/Hardware Version	Identification Keyword	RO	Program/Immediate

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).

Syntax	PR VR

Code example

```
PR VR
   LMMCM 6.002, Hw: 3.2
```

Read the device version
Firmware version and hardware version

Related	UG (Upgrade)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	See Modbus/TCP Fieldbus Manual Section 4.1: Device ID
	0x65	1	0x0B		

5.1.142 VT (Read Voltage)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
VT	Read Voltage	Status Keyword	RO	Program/Immediate

Description

The VT keyword is used in conjunction with the PR (Print) instruction to read the status and voltage of the device.

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, Error 78 asserted
3	Out of range	Out of range	Abnormal condition, Error 79 asserted

An optional parameter may be used to read the voltage and status of a specific voltage:

1) Auxiliary Voltage (LMD products only)
2) +VDC

Param	Description
<blank>	Read both sensors, bridge first, then µController
1	Read the Aux V level
2	Read the +V level

Syntax	PR VT, <param>

Code example

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

Related	IT (Internal Temperature)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	See Modbus/TCP Fieldbus Manual Section 4.3: Mfg Specific Function Codes
	0x65	1	0x0C		

5.1.143 WT (Warning Temperature)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Mnemonic	Function	Function Group	Access	Usage
WT	Set/Read Warning Temperature	System Variable	RW	Program/Immediate

Description

The Warning Temperature variable allows the user to set a threshold temperature at which the device will assert an error 71 to the terminal screen if the set temperature is exceeded.

Note that this is a single setting that will set the warning level for bother temperature sensors. If either reaches the set threshold the error code will be asserted

Range	0-84	Units	ºC	Default	80
Syntax	WT=<temperature> | PR WT				

Code example

`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)		

Networking protocol equivalents

EtherNet/IP	class	instance	attribute	Modbus/TCP	—
	0x64	1	0x05		

5.2 Math, logic and trigonometric operators

NOTE: Firmware versions prior to Firmware 6.00.00+ do not support advanced floating point math and trigonometric functions.

5.2.1 Addition (+)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Symbol	Function	Function Group
+	Addition	Basic Math

Description

Adds the contents of variables

Syntax	<sum target>=<augend>+<addend>+...

Code example

`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

5.2.2 Subtraction (-)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Symbol	Function	Function Group
-	Subtraction	Basic Math

Description

Subtracts the contents of two variables

Syntax	<difference target>=<menuend>-<subtrahend>

Code example

`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

5.2.3 Multiplication (*)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Symbol	Function	Function Group
*	Multiplication	Basic Math

Description

Multiplies the contents of two variables

Syntax	<product target>=<multiplicand>*<multiplier>

Code example

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

5.2.4 Division (/)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Symbol	Function	Function Group
/	Division	Basic Math

Description

Divides the contents of one variable with another variables

Note that if you are dividing integer values and require a more precise quotient, the quotient may be stored in F1-F8 (Floating Point Registers).

Syntax	<quotient target>=<dividend>/<divisor>

Code example

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

5.2.5 Equal (=)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Symbol	Function	Function Group
=	Equal	Comparison operator

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 example

`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

5.2.6 Not Equal (<>)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Symbol	Function	Function Group
<>	Not Equal	Comparison operator

Description

Test if two variables are not equal.

Syntax	[BR/CL] <label/address>,<var/flg/io><><var/flg/num>

Code example

`CL X1,Q1<>25`	Call subroutine when user variable Q1 is not equal 25

5.2.7 Less Than (<)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Symbol	Function	Function Group
<=	Less Than	Comparison operator
Description		

Tests if Variable is less than a second variable

Syntax	[BR/CL] <label/address>,<var/flg/io>**<**<var/flg/num>

Code example

`CL X1,Q1<=25`	Call subroutine when user variable Q1 is less than 25

5.2.8 Less Than or Equal (<=)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Symbol	Function	Function Group
<=	Less Than or Equal to	Comparison operator
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 example

`CL X1,Q1<=25`	Call subroutine when user variable Q1 is less than or equal 25

5.2.9 Greater Than (>)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Symbol	Function	Function Group
>	Greater than	Comparison operator
Description		

Tests if Variable is greater than to a second variable

Syntax	[BR/CL] <label/address>,<var/flg/io>**>**<var/flg/num>

Code example

`CL X1,Q1>25`	Call subroutine when user variable Q1 is greater than 25

5.2.10 Greater Than or Equal (>)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Symbol	Function	Function Group
>=	Greater than or Equal	Comparison operator
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 example

`CL X1,Q1>=25`	Call subroutine when user variable Q1 is greater than or equal 25

5.2.11 AND (&)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Symbol	Function	Function Group
&	AND	Logic operator

Description

Performs a Logic AND operation on two variables.

Syntax	<target var>=<var/flg>&<var/flg/num>

Code example

`R1=25` `R2=30`	Assign value to user registers
`R3=R1&R2`	AND R1 and R2 together, store in R3
`PR R3` `30`	Read the value of R3 R3 is 30

5.2.12 OR (|)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Symbol	Function	Function Group
|	OR	Logic operator

Description

Logic OR operation between two variables.

Syntax	<target var>=<var/flg>|<var/flg/num>

Code example

`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 25

5.2.13 XOR (^)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Symbol	Function	Function Group
^	XOR	Logic operator
Description		

Logic XOR operation between two variables.

Syntax	<target var>=<var/flg>^<var/flg/num>

Code example

R1=25 R2=30	Assign value to user registers
R3=R1^R2	AND R1 and R2 together, store in R3
PR R3 25	Read the value of R3 R3 is 25

5.2.14 NOT (!)

Compatibility	LMD(O) LMD(C) LMM	Notes	—

Symbol	Function	Function Group
!	NOT	Logic operator
Description		

Logic NOT operation.

Syntax	<target var>=<var/flg>!<var/flg/num>

Code example

R1=25 R2=30	Assign value to user registers
R3=!R1	AND R1 and R2 together, store in R3
PR R3 -26	Read the value of R3 R3 is -26

The advanced math and trigonometric calculation should be performed using the double-precision floating point registers F1-F8 (Floating Point Registers).

5.2.15 AB (Absolute Value)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

Symbol	Function	Function Group
AB	Absolute Value	Advanced Math/Trigonometry
Description		

Preforms an Absolute on the specified register.

Syntax	<target fpreg>=AB <var/flg/num>

Code example

MA -51200 PR P -51200	Move negative 51200 steps (1 Rev) Read the position counter Position counter is at -51000 steps
F1=AB P	Perform absolute on position counter, store in F1
PR F1 51200.00000	Read the value of F1 F1 is 51200.00000

5.2.16 CS (Cosine)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

Symbol	Function	Function Group
CS	Cosine	Advanced Math/Trigonometry
Description		

Preforms an cosine on the specified register.

Syntax	<target fpreg>=**CS** <var/flg/num>

Code example

VA Q1=51200	Create and ssign 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

5.2.17 C_ (Arc Cosine)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

Symbol	Function	Function Group
C_	Arc Cosine	Advanced Math/Trigonometry
Description		

Preforms an arc cosine on the specified register.

Syntax	<target fpreg>=**C_** <var/flg/num>

Code example

Continues example from 5.2.4.2 CS (Cosine)

`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`**

5.2.18 LO (Logarithm Base 2)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

Symbol	Function	Function Group
LO	Logarithm Base 2	Advanced Math/Trigonometry
Description		

Preforms an logarithm (base 2) on the specified register.

Syntax	<target fpreg>=**LO** <var/flg/num>

Code example

`VA Q1=51200`	Create and assign value to user register Q1
`F1=LO Q1`	Store log (base 10) of Q1 in F1
`PR F1` `10.843495`	Read the value of F1 F1 is `10.843495`

5.2.19 L_ (Logarithm Base 10)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

Symbol	Function	Function Group
L_	Logarithm Base 10	Advanced Math/Trigonometry
Description		

Preforms an logarithm (base 10) on the specified register.

Syntax	<target fpreg>=**L_** <var/flg/num>

Code example

`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`

5.2.20 PI (3.141592654)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

Symbol	Function	Function Group
PI	PI (3.141592654)	Advanced Math/Trigonometry
Description		

Holds the value of PI.

Syntax	<target fpreg>=<reg/var><math>**PI**

Code example

`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`

Calculates the sine of the specified register.

Syntax	<target fpreg>=**SI** <var/flg/num>

Code example

`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`

5.2.22 S_ (Arc Sine)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

Symbol	Function	Function Group
S_	Arc Sine	Advanced Math/Trigonometry

Description

Calculates the arc sine of the specified register.

Syntax	<target fpreg>=**S_** <var/flg/num>

Code example

`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`**

5.2.23 SQ (Square Root)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

Symbol	Function	Function Group
SQ	Square Root	Advanced Math/Trigonometry
Description		

Calculates the square root of the specified register.

Syntax	<target fpreg>=**SQ** <var/flg/num>

Code example

`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`

5.2.24 TG (Tangent)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

Symbol	Function	Function Group
TG	Tangent	Advanced Math/Trigonometry
Description		

Calculates the tangent of the specified register.

Syntax	<target fpreg>=**TG** <var/flg/num>

Code example

`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`

5.2.25 T_ (Arc Tangent)

Compatibility	LMD(O) LMD(C) LMM	Notes	Firmware 6.00.00+

Symbol	Function	Function Group
T_	Arc Tangent	Advanced Math/Trigonometry

Description

Calculates the arc tangent of the specified register.

Syntax	<target fpreg>=**T_** <var/flg/num>

Code example

`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`

6 SUPPORTING SOFTWARE

The software associated withLexium Motion product products is contained within the Lexium Software Suite. This software package is available for download at the Schneider Electric Motion USA web site at http://motion.schneider-electric.com.

The modules applicable are:

1. **Motion Control Interface:**
   - Graphic User Interface (GUI) for developing and simulating Lexium 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 Lexium MCode programs.
   - For RS-422/485 and EthernetLexium Motion product products
   - Motion Control Firmware upgrade utility.
2. **Ethernet Configuration Utility:**
   - For Lexium 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 separate manuals. The Manual for the module be used may be down loaded at:

http://motion.schneider-electric.com

This page intentionally left blank

7 PROGRAMMING AND APPLICATION NOTES

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

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

7.1 Party mode communications

The following communication formats, used by Lexium MCode compatible devices.

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

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

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

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

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

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

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

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

7.1.1 Response to Echo Mode

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

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

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

Parameter Setting	Transmission	Initial Response	Final Response	Notes
EM=0 & PY=1 CK=0	(DN) (command) (0A)	(command) Echoed back one character at a time as the character is entered.	CET (0D) (0A)>	The last character sent is the prompt >
EM=1 & PY=1 CK=0	(DN) (command) (0A)	–	CET (0D) (0A)	The last character sent is LF

EM=2 & PY=1 CK=0	(DN) (command) (0A)	_	_	No response except to PR and L commands
EM=3 & PY=1 CK=0	(DN) (command) (0A)	_	CET command (0D) (0A)	Queued response. The last character sent is the LF

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

Parameter Setting	Transmission	Initial Response	Final Response	Notes
EM=0 & PY=0 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

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

Parameter Setting	Transmission	Initial Response	Final Response	Notes
EM=0 & PY=1 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

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

7.1.2 Using Check Sum

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

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

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

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

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

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

the receiving device should not respond with ACK or NAK.

7.1.3 Immediate party mode sample codes

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

Move device A, B or C 10000 steps

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

- To move mdrive unit “a”, press CTRL+J and then type: aMR∧10000 and 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: c<variable name><number> and press CTRL+J. The variable will be changed. To verify the change type: cPR <variable name> and press CTRL+J. The new value will be displayed.
- All commands and variables may be programmed in this manner.
- To take a device out of party mode type: <device name>PY=0 and press CTRL+J. That unit will be taken out of party mode. To take all units out of party mode type: *PY=0 and press CTRL+J. All units will be taken out of party mode.

7.2 Programming the I/O

7.2.1 I/O availability per device type

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

NEMA size 17

- 3 — +5 to +24 VDC isolated input points. Programmable to multiple functions. Sink or source.
- 1 — analog input.
- 1 — high speed isolated output. Programmable to multiple functions including Trip.

7.2.2 Active states defined

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

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

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

Examples

Input 1 is to be configured as a Jog– input which will activate when a switch is toggled to ground:

```
IS=1,8,0     'set input 1 to jog-, active low
```

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

```
IS=4,1,1     'set input 4 to home, active high
```

7.2.3 Digital input functions

The inputs may be interfaced to a variety of sinking or sourcing devices. An input may be programmed to be a general purpose user input, or to one of 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 ims 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 Input Reference input.

Connecting the input to a +5 to +24 VDC supply will provide for sinking inputs.

Connecting the input to Ground will provide for sourcing inputs.

Refer to Section 6 of theLexium Motion product Hardware Manual for examples.

NEMA Size 23 and 34

- 4 — +5 to +24 VDC isolated input points. Programmable to multiple functions. Sink or source.
- 1 — analog input.
- 2 — +5 to +24 VDC isolated outputs, dry contact configuration. Programmable to multiple functions
- 1 — high speed isolated output. Programmable to multiple functions including Trip.

Input Functions

The following table lists the programmable 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) Lexium Motion product models.	1	12	0/1

Table 7.5 Digital input functions

7.2.4 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 600 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.

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

Table 7.6 Digital output functions

7.2.5 Programmable input usage examples

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

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

Input Interface Example - Switch Input

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

Code Sample

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

```
***Setup Variables***
IS=4,4,0      'set input 4 to be a G0 input, active
LOW

****Program***
PG1
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
E

PG            'End program, exit program mode
```

Input interface example - switch input

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

Code Sample

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

```
IS=1,5,1      'set input 1 to Soft Stop, active when
HIGH

SL 200000     'slew the motor at 200000 µsteps/sec
```

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

Output interface example

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

Code Sample

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

```
IS=1,5,1      'set input 1 to Soft Stop, active when
HIGH
OS=1,20,0     'set output 1 as a Velocity Changing
activeLOW
SL 2000000    'slew the motor at 200000 µsteps/sec
```

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

Output interface example

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

Code Sample

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

```
******Setup Variables******
OS=1,16,1     'set IO 1 = user output, active HIGH


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

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

Reading inputs as a group example

The inputs may 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 1 - 3. Inputs should be configured as user inputs (IS = <line>,0).

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

Interfacing outputs as a group example

Outputs may be written to as a group using the OT keywords\ . This will set the outputs as a binary number representing the decimal between 0 to 7 representing the 3 bit binary number on 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 (S=<line>,16).

```
OT=5    'set the binary state of the combined I/O to
101
```

7.2.6 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 anlog 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
```

7.3 Factors impacting motion commands

7.3.1 Motor steps

All Lexium MCode examples assume 200 step motors. They rotate at 1.8° per clock pulse. 200 steps would equal 1 revolution.

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

7.3.3 Move Command

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

7.3.4 Closed loop control with an encoder

If the encoder is enabled the move commands use different values. The encoder has 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.

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

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

7.3.5 Linear movement

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

7.5 inches divided by 0.1 inches = 75 motor revolutions.

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

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

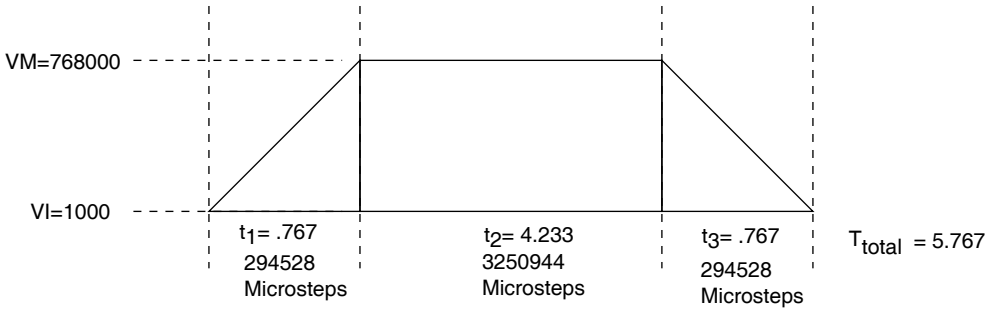


Figure 7.2 Trapezoidal move profile

Calculating axis speed (velocity)

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

Known Values and Parameters:

VM.....768000 Steps/Sec.

VI.....1000 Steps/Sec.

A.....1000000 Steps/Sec2.

D .....1000000 Steps/Sec2.

MA/MR.....3840000 Microsteps

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

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

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

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

$$\textbf{= 294911 steps}$$

Determine the t2 time.

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

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

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

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

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

Note that the average speed includes the Acceleration and Deceleration. The maximum axis speed attained is approximately 90 inches/minute.

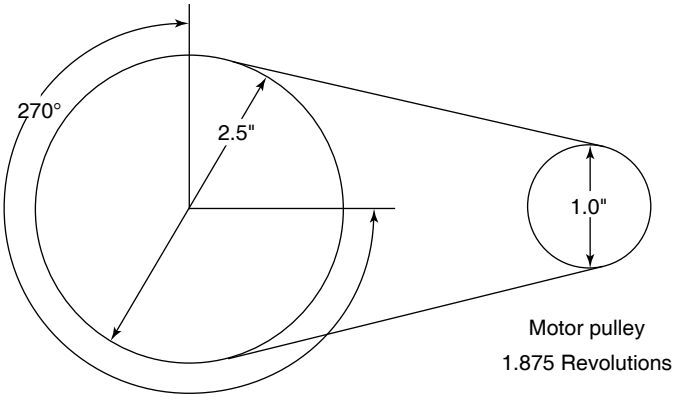
$$\frac{768000}{51200} \text{ x } 0.1 \text{ x } 60 \textbf{ = 90 IPM}$$

7.3.6 Calculating rotary movement

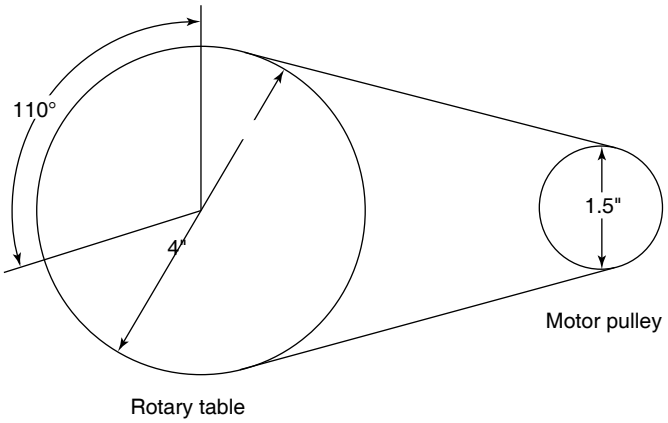
Assume that MS is set to 256. You are using the motor to drive a shaft with a timing belt and pulley arrangement. As shown below, the pulley is 1” in diameter and the shaft pulley is 2.5” in diameter. You must turn the shaft 270°.

- The shaft will rotate 1 full revolution for every 2.5 revolutions of the motor.
- 270° is 0.75 of a revolution.
- 0.75 x 2.5 = 1.875 motor revolutions to turn the shaft 270°.
- If 51200 Microsteps is 1 motor revolution, then the device must be programmed to move 96000 Microsteps (51200 x 1.875).

You may also do many of the calculations in reverse to calculate motor moves to meet a required move of your device. A linear or rotational move as well as speed may be translated into an Lexium MCode command.



Rotary drive example 1



Rotary drive example 2

Figure 7.3 Rotary examples

In the example above, the belt driven rotary table must be turned 110° at 3 RPM. How should the device be set up?

Bear in mind that all the numbers are approximate due to rounding.

Mechanical ratio between the motor and the rotary table is 2.666:1. That is, the motor must rotate 2.666 revolutions for the table to rotate 1 revolution and the table will rotate 2.666 times slower than the motor.

In order to move the table 110° the motor must move 293.3°.

$$110 \times 2.66 = \mathbf{293.3°}$$

If 51200 steps = 1 revolution then 1° = 142.222 steps.

$$\frac{51200}{360} = \mathbf{142.222\ steps}$$

The Lexium MCode device must be programmed to move 41713 steps to rotate 293.3°.

$$142.222\ \text{steps} \times 293.3° = \mathbf{41713\ steps}$$

In order to rotate the table at 3 RPM the motor must turn at 8 RPM.

$$3\ \text{RPM} \times 2.666 = \mathbf{8\ RPM}$$

If you were to set VM at 51200 and MS set at 256 the motor will rotate 1 full revolution (51200 steps) in 1 second or 1 RPS. In order to rotate at 8 RPM, the motor must rotate at 0.13333 RPS.

$$\frac{8}{60} = \mathbf{0.133333\ RPS}$$

In order to rotate at 0.13333 RPS the VM must be set at 6827 steps/sec.

$$51200 \times 0.133333 = \mathbf{6827}$$

VM = 6827

Note: These numbers will vary slightly depending on Acceleration and Deceleration rates.

7.3.7 Programming with the optional encoder enabled

An optional 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 you were to program motion using the MR (Move Relative) or MA (Move Absolute) commands the motor would rotate a distance equal to the encoder counts.

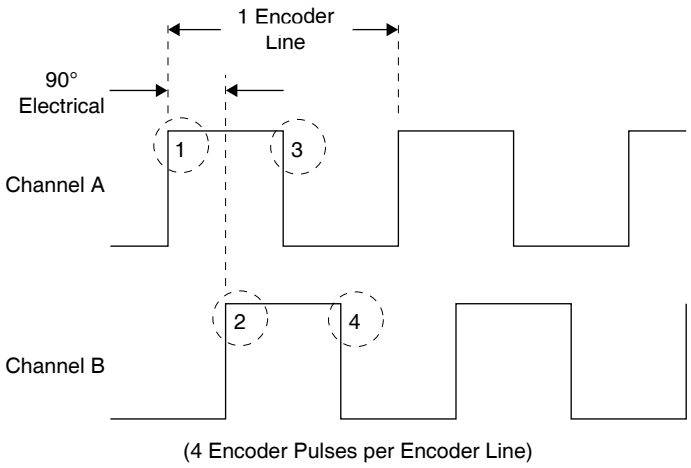


Figure 7.4 Encoder waveform

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 you were to program motion using the SL (Slew) command the motor would rotate at a "counts per second" rate based on the programmed value.

Example: An SL (Slew) rate of 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.

The Encoder Enabled defaults are:

VM.....60000 Counts/Sec.

VI.....78 Counts/Sec.

A.....78125 Counts/Sec

D.....78125 Counts/Sec.

MS.....256 (default for encoder mode.)

To enable the encoder the program syntax is <EE=n> where n is a zero (0) or a one (1). The default is zero (0) which is encoder disabled. To enable the encoder, program EE=1.

Any motion will now be programmed in encoder counts. You can calculate the distance or velocity you need in a similar manner as done previously only with different factors.

Note: The microstep select is defaulted and locked at 256 in the encoder mode to ensure stable, high resolution.

Several Variables work in conjunction with Encoder Enable (EE). They are:

DB .....Encoder Deadband

SF.....The Stall Factor Variable

SM.....The Stall Detection Mode

ST.....Stall Flag

PM.....Position Maintenance

EE .....Encoder Enabled

When the encoder is enabled, all motion is “closed loop”. That is, motion steps are delivered from the Lexium MCode device to the motor which turns the encoder. The encoder sends counts back to the drive to complete the motion. If you programmed a move of 2048 counts, the device would output an appropriate number of microsteps provided the stall factor (SF) value or other fault is not encountered. If no faults were encountered, the device would output the full amount of microsteps. Depending on which variables were set, the driver would then wait until the position (plus or minus the encoder deadband) was read and confirmed.

DB - Encoder Deadband

The Encoder Deadband is a Variable that is set in Encoder Counts. Motion will be deemed complete when the Encoder Counts are within ± the Deadband variable. With DB=5 the motion of 2048 counts would be complete between 2043 and 2053 counts.

SF - Stall Factor

The Stall Factor is a Variable which is entered in Encoder Counts. The Stall Factor is active only in the EE=1 mode. The Stall Factor might be compared to the “following error” or “lag error” of a servo drive. The Stall Factor is triggered by the number of steps output from the device to the motor as compared to the number of counts returned by the encoder. The comparison should always be within the value of the Stall Factor, otherwise a fault will occur and the Stall Flag (ST) will be set. If the Stall Detection Mode is active (SM=0), the motion will be stopped.

Example:

A Stall Factor of 30 counts (SF=30) is programmed. A motion command of 2048 counts is programmed. The device reaches a mechanical bind at 2000 counts. The device will keep outputting steps equivalent to 2030 counts (present position plus the SF value) and then the Stall Flag (ST) will be set. The motor will be stopped if the Stall Detection Mode (SM=0) is active.

SM - Stall Detection Mode

The Stall Detection Mode can be programmed to stop the device (SM=0) or to allow the device to continue (SM=1) when the Stall Factor (SF) is reached. Whether SM is active or not, the Stall Flag will always be set when the SF is encountered.

ST - Stall Flag

The Stall Flag will be set any time the SF is reached regardless of the state of the Stall Detection Mode (SM). If the Stall Flag is set, the user must reset it to zero (0).

PM - Position Maintenance

Position maintenance (PM) is active only after the motion has completed. Position maintenance is used to maintain position when there might be an external force on the drive. If position maintenance is enabled (PM=1) and the stall detection mode is enabled (SM=0), the motor will be driven back to its final position if it was forced out of position provided the stall factor (SF) was not reached.

If position maintenance is enabled (PM=1) and the stall detection mode is disabled (SM=1), the motor will be driven back to its final position if it was forced out of position regardless of whether the stall factor (SF) was reached or not.

There are three other variables, although not directly conned to EE, that do affect the overall operation when in encoder mode, they are:

HC..... Motor Hold Current

HT..... Motor Hold Current Delay Time

MT.....Motor Settling Delay Time

HC.....Hold Current

When motion is complete, the device will switch from motor run current (RC) to motor hold current (HC). The hold current is set at a lower percentage than the run current (rc). However, the hold current must be sufficient to overcome an outside force such as driving a vertical slide which maintains a load on the motor at all times. Actual hold current

values will vary depending on the application and the load on the motor when it is at rest.

HT - Motor Hold Current Delay Time

The motor hold current delay time (HT) is a variable that delays the change from run current (RC) to hold current (HC) at the end of a move. The end of the move is triggered by the device when it has completed outputting the correct number of steps. Depending on the application, including velocity, deceleration, load and inertia, the device may lag behind a few counts. The ht will allow the device to finish its move before applying the lower HC.

MT - Motor Settling Delay Time

A stepping motor may ring or oscillate in minuscule amounts at the completion of a move until it satisfies the target position. The amount of this “ringing” is dependent on the application including velocity, deceleration, inertia, friction and load. The motor settling delay time (MT) allows the motor to stop “ringing” before checking the position count. If the device tried to check the position count during this ringing, it would assume a position error and try to correct an already moving motor and possibly cause ringing of a larger magnitude and longevity. Typically, the MT is set between 50 and 100 milliseconds. It is recommended that there is always a Motor Settling Time programmed any time you are in EE=1 mode.

Note: If MT has no value, the motor may hunt and never satisfy the position check.

8 HMTECHNOLOGY

NOTICE
This section only applies to Lexium MDrive Motion Control and Lexium MDrive Ethernet Closed Loop products.

8.1 hmTechnology overview

hMTechnology is a proprietary closed loop control technology that, when applied to step motors, prevents the loss of synchronization due to transient or continued overload, extreme acceleration or deceleration, or excessive slew speed.

8.1.1 Glossary of Terms

Because hMT is a unique technology, some of the terms used to describe its operation are defined:

8.1.1.1 hMTechnology (hMT)

A motor control technology representing a new paradigm in brushless motor control. By bridging the gap between stepper and servo performance, hMT offers system integrators a third choice in motion system design.

8.1.1.2 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.

8.1.1.3 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.

8.1.1.4 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.

8.1.1.5 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. hMTechnology eliminates this.8.1.1.6 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.

8.1.1.7 Position lead/lag

hMT continually tracks the position lead or lag error, and may use it to correct position.

8.1.1.8 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.

8.1.1.9 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.

8.1.2 hMTechnology Basics

hMTechnology is the core control technology that enables the multi-mode functionality of the Lexium MDrive by overcoming many of the limitations inherent in stepper systems. Two major limitations addressed by this technology are:

- Loss of motor synchronization and subsequent stalling.
- Excessive motor heated due to limited current control options

8.1.2.1 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.

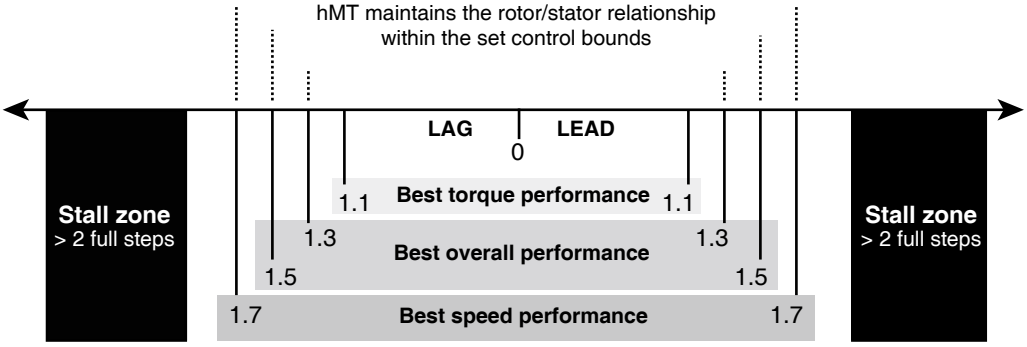


Figure 8.1 Motion block, hMTechnology disabled

8.1.2.2 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 cooler running motor.

8.1.2.3 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.

8.1.3 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.

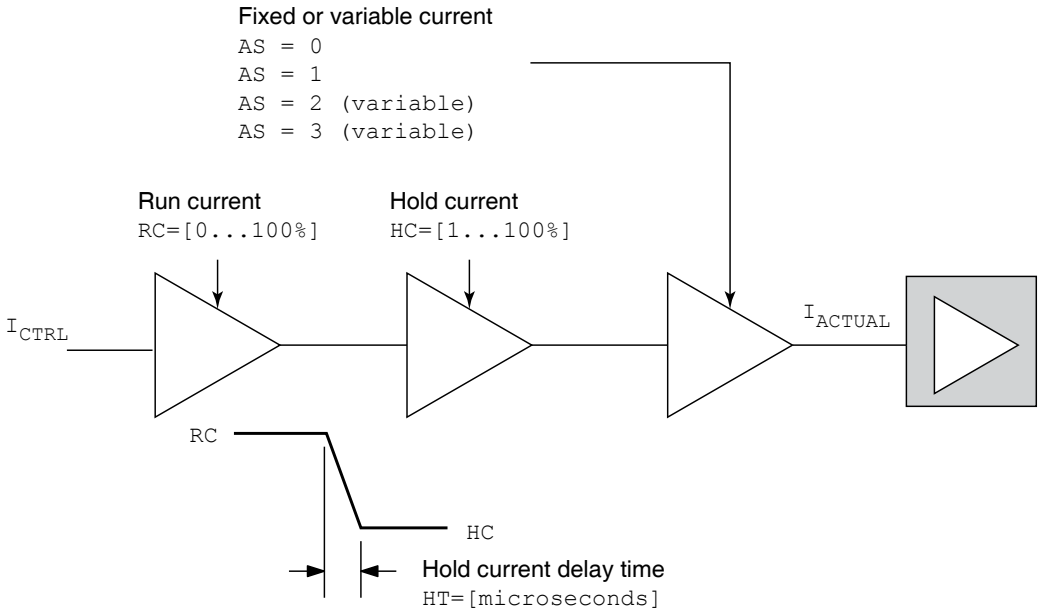


Figure 8.2 Overview of motor phase current

8.2 hMTechnology modes of operation

There are four operational modes for the hMTechnology, which are configured using **AS (hMTechnology Mode)**:

1) hMT Off, or bypass(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 programmatically or immediately provided a move is not in progress.

8.2.1 hMT off (bypass) (AS=0)

With the hMTechnology 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.

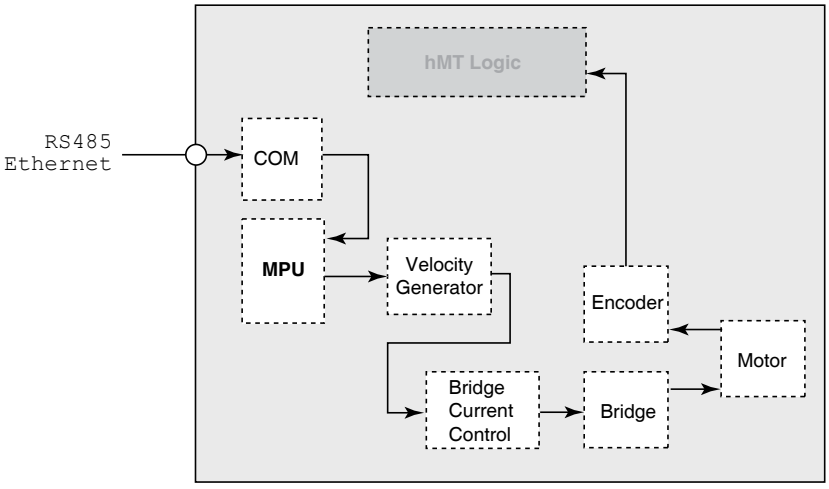


Figure 8.3 Motion block, hMTechnology disabled

In bypass mode, the current control will be fixed at the set run **RC (Run Current)** and hold **HC (Hold Current)** current percent levels.

Encoder functions are not available in bypass mode.

8.2.2 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. hMTechnology

Common encoder functions such as stall detection and position maintenance are disabled when fixed current mode is selected.

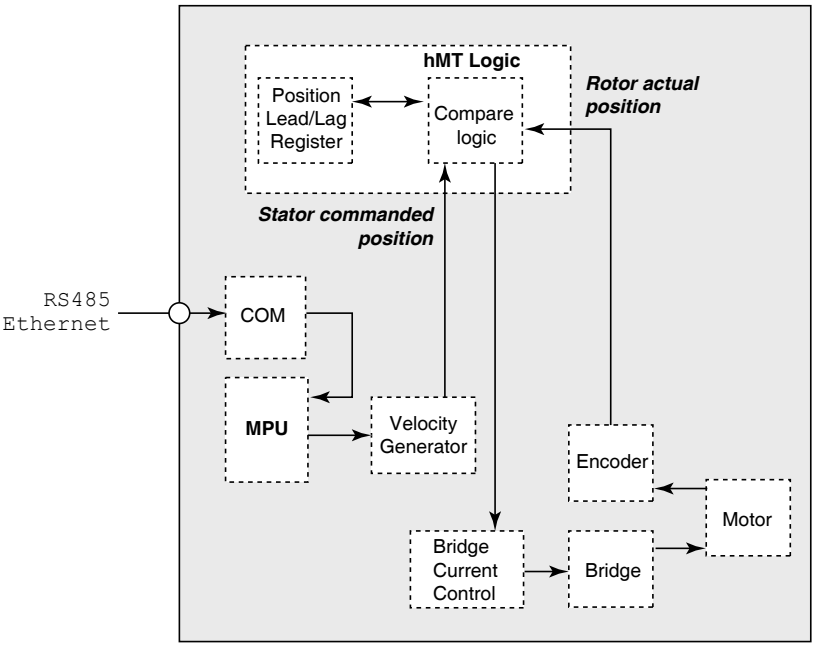


Figure 8.4 Block diagram, hMTechnology enable (AS=1/2)

8.2.3 hMT on (variable current) (AS=2)

With hMT enabled in variable current mode (AS=2) the hMT will function as described in Sub-section 8.2.2 with the difference 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 hMTechnology Mode 2 can significantly increase the energy efficiency and reduce the motor heating. The graph in Figure 8.4 shows the 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%.

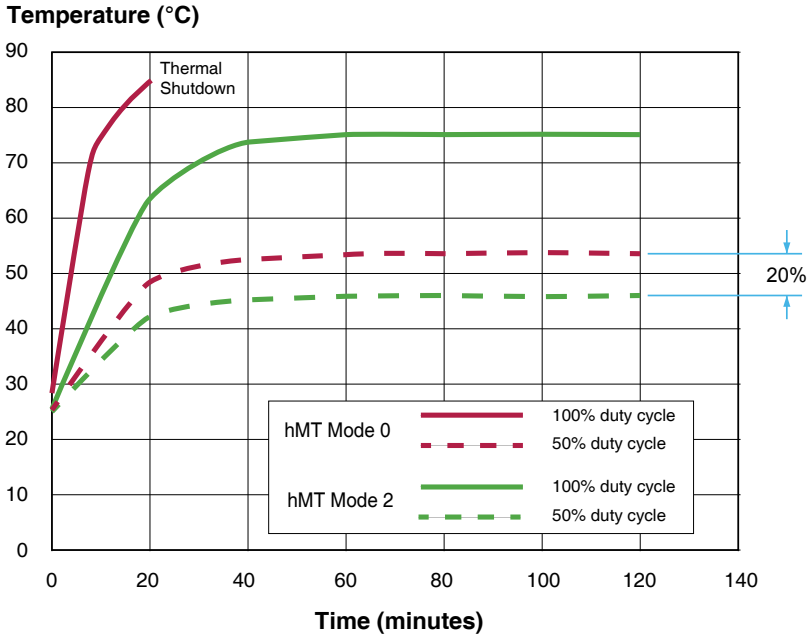


Figure 8.5 hMT Mode 2 - Variable current mode thermal performance

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.

8.2.4 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 (TQ)** 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.

8.3 Position Make-up

Make-up mode is active when ever hMTechnology is on in fixed (AS=1) or variable (AS=2) current mode. Make-up compensates for position errors resultant 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.

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)**.

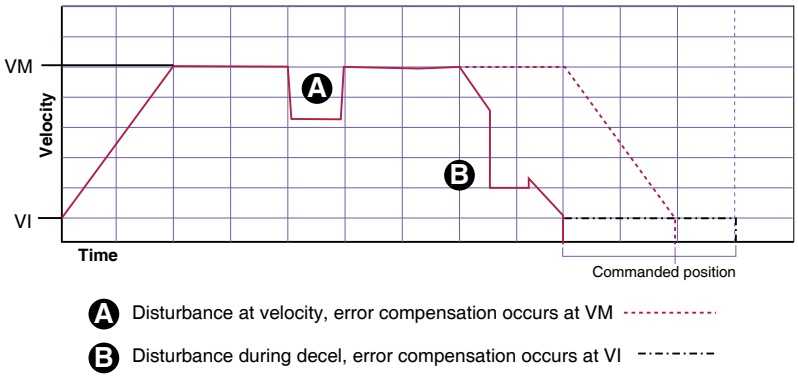
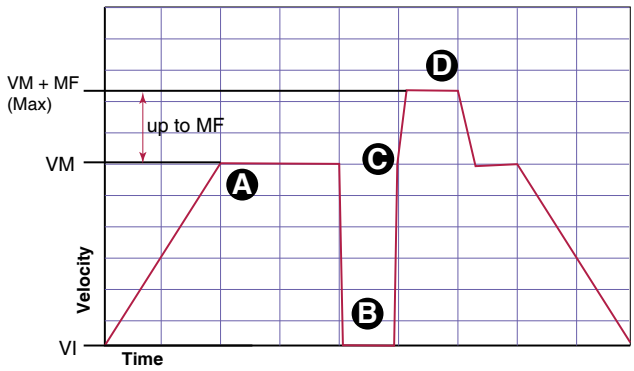


Figure 8.6 Make up mode MU=0

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 interleave 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.

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 stepes re-inserted into profile at a composite frequency that will be the greater of 2 times (VM or SL) or MF.

Figure 8.7 Make up mode MU=1

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 that when the the mtor 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.

8.4 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 an Error code asserted. 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 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.

8.5 hMTechnology 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

9 SAMPLE PROGRAMS

This section is made up of several example programs designed to aid the user in discovering the Lexium MCode programming language.

Download sample programs

All the sample programs from this section may be downloaded from the web site at http://motion.schneider-electric.com in *.ixt ((Motion Control Programmer) format.

> **DOWNLOAD NOW**

9.1 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 termianl tab to run

9.2 Change velocity during a move

This program will demonstrate ability to change speed during move. The device does not have abiltity to change speed during point to point move, so we use the slew command with position trips. End position trip, decel and slew speed determine actual ending position. Program is written to print ending position to serial port 10 times for averaging, expected end position = 102400.

Use the file change_speed_during_a_move.ixt in the **sample_programs.zip** file

```
'[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 velocity 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

9.3 Binary mask

This program will demonstrate ability to execute various subroutines depending on the binary value of inputs 1-3 while masking all i/o above input 3.

Use the file binary_mask.ixt in the **sample_programs.zip** file

```
'[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 cally various subroutines based upon
'the wieght 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 wieght 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)

9.4 Closed Loop

This program illustrates closed loop control with an On Error (OE) routine which will perform math functions on the counters to display the position error.

Use the file closed_loop_on_error.ixt in the **sample_programs.zip** file

```
'[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
'microsteps
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]
```

9.5 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.

Use the file user_input_into_variables.ixt in the **sample_programs.zip** file

```
'[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 number 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 orint 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
```

9.6 Closed loop with homing

This program demonstrates the use of the home to home switch instruction (HM) in closed loop, also there is a move on input routine.

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.

Use the file closed_loop_with_homing.ixt in the **sample_programs.zip** file.

```
'[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
```

9.7 Input trip

This program demonstrates the use input trips. TheLexium 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.

Use the file trip_on_input.ixt in the **sample_programs.zip** file.

```
'Lexium 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
```

9.8 Position teach (encoder required)

This program allows the user to “teach” theLexium 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.

Use the file position_teach.ixt in the **sample_programs.zip** file.

```
'Lexium 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
```

9.9 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 bi-directionally based upon the value seen on the analog input.

Hardware requirement: 10kΩ potentiometer connected to the Analog input.

Use the file analog_speed_control.ixt in the **sample_programs.zip** file

```
'Lexium 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
```

9.10 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,

Use the file analog_slew_with_stall_detect.ixt in the **sample_programs.zip** file.

```
'Lexium 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
```

9.11 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.

Use the file multiple_position_trips.ixt in the **sample_programs.zip** file.

```
'Lexium 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.
```

Page intentionally left blank

10 ERROR CODES

A question mark <?> displayed as a cursor indicates an error. To determine what the error is, type <pr er> in the terminal window. The device will respond with an error number displayed in the terminal window. The error number may then be referenced to this list.

0 No Error

10.1 Lexium MDrive Error Codes

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
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.
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 postion 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 SCurve Accel Data.
99	Move generates illegal SCurve 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.

10.2 Lexium Motion Module Error Codes

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
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
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 postion 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 SCurve Accel Data.
99	Move generates illegal SCurve Decel Data.

Page intentionally left blank

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

T

U

V

W

X

WARRANTY

Reference the web site at www.motion.schneider-electric.com for the latest warranty and product information.

USA SALES OFFICES

East Region
Tel. 610-573-9655
e-mail: e.region@imshome.com

Northeast Region
Tel. 860-368-9703
e-mail: n.region@imshome.com

Central Region
Tel. 630-267-3302
e-mail: c.region@imshome.com

Western Region
Tel. 602-578-7201
e-mail: w.region@imshome.com

EUROPEAN SALES MANAGEMENT
Tel. +33/4 7256 5113 – Fax +33/4 7838 1537
e-mail: europe.sales@imshome.com

TECHNICAL SUPPORT
Tel. +00 (1) 860-295-6102 – Fax +00 (1) 860-295-6107
e-mail: etech@imshome.com

Schneider Electric Motion USA

370 N. Main Street
Marlborough, CT 06447 USA

www.motion.schneider-electric.com

Owing to changes in standards and equipment, the characteristics given in the text and images in this document are not binding until they have been confirmed with us.
Print: Schneider Electric Motion USA
Photos: Schneider Electric Motion USA

Date : 05 / 2016