MCode Programming and Software Reference

MDrivePlus and MForce Motion Control

IMS™ INTELLIGENT MOTION SYSTEMS, INC.

by Schneider Electric

MDrivePlus Motion Control Software Reference Change Log		
Date	Revision	Changes
03/08/2006	R030806	Initial Release
04/24/2006	R042406	Expanded Explanation of CR (pg 15), Corrected EL (pg 18), added new parameters to ES (pg 19), Expanded explanation of HT (pg 23) and MT (pg 31). Added Warning Temp WT (pg 44). Updated Screen captures of MDrive CD and IMS Terminal Installation instructions (pg A4-A5).
05/25/2006	R052506	Expanded information on CP (pg 21) and FD (pg 26). Expanded the Usage Example on IV (pg 33). Added explanation the I/O 13 is Position Capture Input/Position Trip Output ONLY (pg 45).
08/04/2006	R080406	Minor additions to command descriptions, added SU Label Description: All programs labeled SU will execute on power up.
10/13/2006	R101306	Changed the name of the MDrive Programming Language to MCode. Updated references throughout.
11/21/2006	R112206	Added MForce MicroDrive to list of MCode compatible products. Added BP (Break Point Instruction), Added MP (Moving to Position Flag.)
12/20/2006	R122006	Re-wrote Appendix B: IMS Terminal to enhance clarity and include features/functions for IMS Terminal 4.2.000, Corrected range for HT and MT, Added Current Settings (RC & HC) for MForce Products
02/16/2007	R021607	Added TR as a reserved for future use word.
04/20/2007	R042007	Revised manual to reflect firmware version MDI3.006. Added D5, analog input filter variable, various corrections and expansions of commands, updated error table.
10/25/2007	R102507	Revised manual to reflect firmware version MDI3.007
02/27/2008	R022708	Revised Manual to reflect firmware version MDI3.008. Added QD (Qeued - Page 45) command and PW variable (Page 44 and Appendex G: MForce PWM Current Control).
06/16/08	R061608	Updated covers, added disclaimer information. Expanded on Trip definitions. Expanded on Error 92. Minor corrections throughout.
08/14/2008	R081408	Added NE (Numeric Enable Command. Relevant to Firmware Version 3.009

NOTE: MCode Compatible Devices:

- MDrivePlus Motion Control Models
- Motion Control MForce MicroDrive and PowerDrive

These are referred to in this document as “MCode compatible device” or “device”

MCode Software Reference

Revision R081408
Firmware Version MDI3.009

Important information

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

Qualification of personnel

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

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

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

Intended Use

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

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

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

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

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

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

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

This page intentionally left blank

MCode Programming Reference

Section 1: Introduction to MCode Programming 3

Section Overview 3
Operational Modes 3
Basic Components of MCode Software 3
Instructions 3
Variables 3
Flags 4
Keywords 4
Math Functions 4
Program Structuring 5
Programming Aids 5

Section 2: MCode Command Set Summary 7

Section Overview 7
Setup Instructions, Variables and Flags 8
Miscellaneous Instructions, Variables and Flags 8
Motion Instructions, Variables and Flags 8
Motion Instructions, Variables and Flags (Continued) 9
I/O Instructions, Variables and Flags 9
I/O Instructions, Variables and Flags (Continued) 10
Position Related Instructions, Variables and Flags 10
Encoder Related Instructions, Variables and Flags 10
Program Instructions, Variables and Flags 11
Mathematical Functions 11

Section 3: MCode Instructions, Variables, and Flags 12

A, AL, AS, AT 12
BD, BP 13
BR, BY 14
C1, C2, CC, CE 15
CK, CL 16
CM, CP, CR, CW 17
D, D1-D4, D9-D12 18
DB, DC 19
DE, DG, DN 20
E, EE, EF 21
EL, EM, ER 22
ES, EX 23
FC, FD 24
FM, FT 25
H, HC, HI 26
HM 27
HT 29
I1-I4, I5, I6, I7-8, I13 30
I9-12, IC, IF, IP 31
IL, IH 32
IN, IT 33
IV, JE 34
L, LB, LK 35
LM, LR 36
MA, MD, MP 37
MR, MS 38
MT, MV 39
O1-O4, 09-O12, OE 40
OL, OH, OT 41
P, PC, PG 42
PM, PN, PR, PS 43

PW, PY 44
QD, R1-R4 45
RC, RS, RT S 46
S1-S4 47
S5 48
S7-S8 49
S9-S12, S13 50
SF, SL, SM 51
SN, ST, SU 52
TC, TE, TI 53
TP,,TR, TP, TT 54
UG, UV 55
V, VA, VC 56
VI, VM, VR 57
WT 58
Error Codes 59

Appendices

Appendix A: ASCII Table A-3

Appendix B: IMS Terminal A-4

Section Overview A-4
Installation A-4
Introduction To IMS Terminal A-6
The Menu Bar Structure and Operation A-7
File Menu Functions A-7
Edit Menu Operation A-7
View Menu Operation A-9
Transfer Menu Operation A-9
Upgrade Menu Operation A-10
Window Menu Operation A-10
Help Menu Operation A-10
The Button Bar Structure and Operation A-12
Program Editor Active A-12
Terminal Window Active A-12
Configuring Program Editor Format Preferences A-13
Configuring Terminal Window Format Preferences A-14
Configuring Communications Settings A-14
Configuring Function Keys A-17
Function Key Control Commands A-18
Setting Up A Function Key A-18
Creating, Downloading and Uploading Programs A-19
Creating a New Program A-19
Downloading a Program A-19
Uploading a Program A-20
Program Troubleshooting Using IMS Terminal A-21
Single Step Mode A-21
Trace Mode A-21
The Capture Function A-21

Appendix C: Upgrading Firmware A-23

Before Upgrading the MCode Firmware A-23
Upgrading the Firmware A-23

Appendix D: Most Commonly Used Variables and Instructions A-27

Variables A-27
Motion Instructions A-27
I/O Instructions A-28
System Instructions A-30
Program Instructions A-30

Appendix E: MCode Program Samples ***A-32***

Sample Programs A-32
Move on an Input *A-32*
Change Velocity During A Move *A-33*
Binary Mask *A-34*
Closed Loop *A-36*
User Input into Variables *A-37*
Closed Loop with Homing *A-38*
Input Trip *A-39*

Appendix F: Factors Impacting Motion Commands ***A-40***

Motor Steps A-40
Microsteps: (MS) *A-40*
Move Command A-40
Closed Loop Control With an Encoder A-40
Linear Movement A-40
Calculating Axis Speed (Velocity) *A-41*
Calculating Rotary Movement A-42
Programming with the Optional Encoder Enabled A-43
Linear Movement A-30
Calculating Axis Speed (Velocity) *A-31*
Calculating Rotary Movement A-32
Programming with the Optional Encoder Enabled A-33

Appendix G: MForce PWM Configuration (PW) ***A-47***

Description: A-47
PWM Mask <mask> Parameter A-47
Maximum PWM Duty Cycle (%) <period> Parameter A-48
PWM Frequency <sfreq> Parameter A-48
PWM Checksum <chksum> Parameter (Boot Write Only) A-49
Boot Writes Remaining <boot_writes_remaining> Parameter (Read Only) *A-49*
Example PWM Settings By Motor Specifications *A-49*

List of Figures

MCode Programming Reference

Figure 1.1: MCode Program Organization and Structure 5
Figure 3.1 Homing Functions Sequence of Events 28
Figure 3.2: Limit Mode Operation 36

Appendices

Figure B.1: Product CD Main Index Page A-4
Figure B.3: Product CD Software Setup Command A-5
Figure B.2: Product CD Software Selection Screen A-5
Figure B.4: IMS Terminal Main Window A-6
Figure B.5: IMS Terminal File Menu Functions A-7
Figure B.6: IMS Terminal Edit Menu Functions A-8
Figure B.7:IMS Terminal View Menu Functions A-9
Figure B.8: IMS Terminal Transfer Menu Functions A-9
Figure B.9: IMS Terminal Upgrade Menu Functions A-10
Figure B.10: IMS Terminal Edit Menu Functions A-10
Figure B.11 IMS Terminal Help Menu Functions A-10
Figure B.12: IMS Terminal Interactive Tutorials A-11
Figure B.13: IMS Terminal Button Bar (Program Editor Active) A-12
Figure B.14: IMS Terminal Button Bar (Terminal Active) A-12

Figure B.15: IMS Terminal Program Editor Preferences A-13
Figure B.16: Terminal Window Preferences A-14
Figure B.17: Communications Preferences A-15
Figure B.18: IMS Sign On Message A-16
Figure B.19: Windows Device Manager showing COM Port A-16
Figure B.20: Function Key Dialog A-17
Figure B.21: F1 Setup A-18
Figure B.22: New Editor Window A-19
Figure B.23: Download Program To MCode Device A-20
Figure B.24: Upload Programs to Program Editor Window A-20
Figure C.1: IMS Terminal Information Page A-23
Figure C.2: MDrive Firmware Upgrade Dialog Progression 1 ..... A-24
Figure C.3: MDrive Firmware Upgrade Dialog Progression 2 ..... A-25
Figure C.4: MDrive Firmware Upgrade Dialog Progression 3 ..... A-26
Figure F.1: Trapezoidal Move Profile A-40
Figure F.2: Rotary Drive Example 1 ..... A-42
Figure F.3: Rotary Drive Example 2 ..... A-42
Figure F.4: Quadrature Encoder Counts A-43
Figure F.5: EE=1 Flowchart A-45
Figure G.1: PWM Mask Bits A-47
Figure G.2 PWM Frequency Range A-48

List of Tables

MCode Programming Reference

Table 2.1: MCode New and Expanded Instructions 7
Table 2.2: Setup Instructions, Variables and Flags 8
Table 2.3: Miscellaneous Instructions, Variables and Flags 8
Table 2.4: Motion Instructions, Variables and Flags 9
Table 2.5: I/O Instructions, Variables and Flags 10
Table 2.6: Position Related Instructions, Variables and Flags 10
Table 2.7: Position Related Instructions, Variables and Flags 10
Table 2.8: Program Instructions, Variables and Flags 11
Table 2.9: Mathematical Functions 11
Table 3.1: Input 13 Filter Capture Settings 20
Table 3.2: Input 7 & 8 Filter Capture Settings 21
Table 3.3: Microstep Resolution Settings 30
Table 3.4: I/O Types and Settings 37
Table 3.5: Output Circuit Conditions 38
Table 3.6: Clock Output Functions 39
Table 3.7: High Speed I/O Types 40

Appendices

Table B.1: Function Key Control Commands B-18
Table G.1: PWM Mask Settings A-47
Table G.2: Typical PWM Mask Settings A-48
Table G.3: Maximum and Initial PWM Frequency A-48

INTELLIGENT MOTION SYSTEMS, INC.

Excellence in Motion™

MCODE PROGRAMMING & SOFTWARE REFERENCE

Section 1: Introduction to MCode Programming

Section 2: MCode Command Set Summary

Section 3: MCode Instructions, Variables and Flags

Page Intentionally Left Blank

SECTION 1

Introduction to MCode Programming

Section Overview

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

- Operational Modes.
- Basic components of the MCode Programming Language.

Operational Modes

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

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

Basic Components of MCode Software

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

- Instructions
- Variables
- Flags
- Keywords
- Math Functions

Instructions

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

Motion

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

I/O

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

Program

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

System

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

Variables

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

Factory Defined Variables

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

defined variables:

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

User Defined Variables

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

The restrictions for this command are:

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

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

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

Flags

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

Factory Defined Flags

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

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

Keywords

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

Math Functions

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

Example

Addition..... K2†=P+R2

Subtraction..... K3†=R1-P
Multiplication ..... A=A*2
Division .....A=A/2
†*User-defined variable used as an example.*

Program Structuring

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

Programming Aids

IMS Terminal

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

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

For more information on IMS Terminal see Appendix B: Installing and Using IMS Terminal.

User Labels

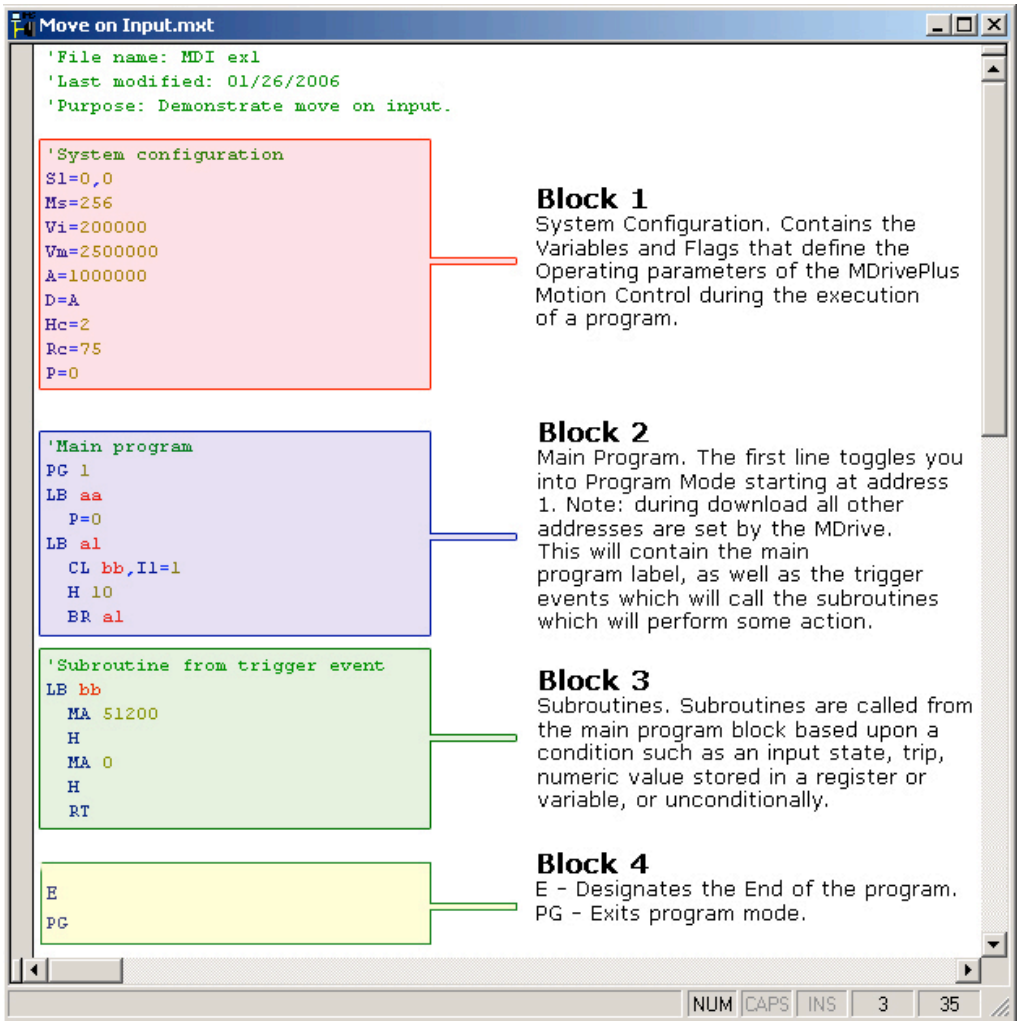


Figure 1.1: MCode Program Organization and Structure

The MCode Programming Language allows for 192 User Labels for your Programs, Subroutines, and user Variables and Flags. A label consists of 2 characters, the first of which must be a letter, the second may be alphanumeric. A label cannot use the same character combination as any of the mnemonics used in the MCode programming language.

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

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

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

User Variable Label (Q) .....Example: Q3, Q9, Qz

Note: User labels may not use the same character combination as any of the mnemonics of the MCode programming language.

The maximum amount of user labels is 192.

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
```

Labeling a Program SU will cause that program to execute on power up.

Comments

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

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

Programming Reference

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

Note: the maximum length of a line of code is 64 characters. This includes mnemonics, spaces and commentation.

The MCode will return an error if this limit is exceeded.

SECTION 2

MCode Command Set Summary

Section Overview

This section contains the MCode Command Set for all compatible IMS Products Software commands specific to the Plus² versions are indicated in the command heading. See the table below for a cross-reference of "new" commands.

- New and Expanded Instructions
- Motion Control Command Set Summary
- Motion Control Commands

MCode Programing Language New and Expanded Instructions

Command	Description	Product	
		Plus	Plus²
BP	Break Point	X	X
CE	Control C Software Reset	X	X
CM	Clock Mode Enable		X
CR	Clock Mode Ratio		X
CW	Set Clock Mode Output Step Width		X
D9 - D12	Input 9-12 Switch Debounce time in msec		X
DG	Disable global response	X	X
EL	Encoder Lines Variable		X
ES	Esc flag to switch between ESC and ^E	X	X
FC	Filter Capture input - IO13		X
FM	Filter Motion inputs - I/O 7 & 8 Encoder		X
I9 - I12	Read Input 9-12		X
IL	Read Inputs 1 - 4 as one value	X	X
IH	Read Inputs 9 - 12 as one value		X
IT	Internal Temperature	X	X
O9 - O12	Set Output 9-12		X
OL	Set Binary State of Outputs 1 - 4	X	X
OH	Set Binary State of Outputs 9 - 12		X
MP	Moving to Position Flag	X	X
PC	Position Capture at Trip		X
PN	Part Number, used with Print - Read Only	X	X
S1 - S4	Setup IO 1-4	X	X
S5	Setup IO 5 Analog Input	X	X
S7 - S8	Setup IO 7-8		X
S9 - S12	Setup IO 9-12		X
S13	Setup IO 13 used for position capture		X
SN	Serial Number, used with Print - Read Only	X	X
TC	Trip on Capture		X
TE	Enable Selected Trip	X	X
TR	Trip Relative		X
TT	Trip on Time	X	X
WT	Warning Temperature	X	X

Table 2.1: MCode New and Expanded Instructions

Setup Instructions, Variables and Flags

Mnemonic	Function	Unit	Range	Syntax Example
BD	Communications BAUD Rate	BAUD	48, 96, 19, 38, 11	BD=<baud>
CE	Control C Software Reset	–	–	CE=<0/1>
CK	Check Sum Enable	–	–	CK=<1/0>
CR	Clock Mode Ratio	–	–	CR=1
CW	Clock Mode Output Step Width	–	0 to 255	CW=100
DE	Enable/Disable Drive	–	1/0	DE=<1/0>
DN	Device Name	Character	a–z, A–Z, 0–9	DN=<char>
EM	Echo Mode 0 (def)=Full Duplex, 1=Half Duplex	Mode	<0 to 3>	EM=<mode>
IP	Initial Parameters from NVM	–	–	IP
PY	Enable/Disable Party Mode	Mode	1/0	PY=<mode>
UG	Upgrade Firmware	Code	2956102	IMS Term. Upgrader

Table 2.2: Setup Instructions, Variables and Flags

Miscellaneous Instructions, Variables and Flags

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

Table 2.3: Miscellaneous Instructions, Variables and Flags

Motion Instructions, Variables and Flags

Mnemonic	Function	Unit	Range	Syntax Example
(–)	Do Previously Set Mode to/at This Value	per mode		–<number>
A	Set Acceleration	Steps/Sec^2	1000000000	A=<accel>
D	Set Deceleration	Steps/Sec^2	1000000000	D=<decel>
HC	Set Hold Current	% (Percent)	0 to 100	HC=<percent>

(Continued)

Motion Instructions, Variables and Flags (Continued)

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

Table 2.4: Motion Instructions, Variables and Flags

I/O Instructions, Variables and Flags

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

(Continued)

I/O Instructions, Variables and Flags (Continued)

Mnemonic	Function	Unit	Range	Syntax Example
S7 - S8	Setup I/O Point Type/Active State	–	–	S7=34,0
S9 - S12	Setup I/O Point Type/Active State	–	–	S9=2,0
S13	Setup I/O Point Type/Active State	–	–	S13=60,0
TI	Trip on Input	–	–	TI <input>,<addr>
TE	Trip Enable	See Table	<1–4>	TE=<num>

Table 2.5: I/O Instructions, Variables and Flags

Position Related Instructions, Variables and Flags

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

Table 2.6: Position Related Instructions, Variables and Flags

Encoder Related Instructions, Variables and Flags

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

Table 2.7: Position Related Instructions, Variables and Flags

Program Instructions, Variables and Flags

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

Table 2.8: Program Instructions, Variables and Flags

Mathematical Functions

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

Table 2.9: Mathematical Functions

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background: Applies to Plus^2 Models Only

MNEMONIC **A**	Function: Acceleration	Units: Steps/Sec^2 (EE=0)/Counts/Sec^2 (EE=1)
	Type: Motion Variable	Range: 91 to 1525878997 (Steps), 91 to 61035160 (Encoder Counts)
	Usage: P/I, R/W	Default: 1000000(EE=0), 40000 (EE=1)
	Syntax: A=<value>	Related: D

Description:

The A Variable sets the acceleration rate when changing velocity in steps per second^2. If the A was set at 76800 per second^2 the motor would accelerate at a rate of 76800 counts per second, every second. If the maximum velocity was set at 768000 microsteps per second it would take 10 seconds to reach maximum speed if VI=0.

Usage Example

```
A=20000    'Set Acceleration to 20000 steps/sec2
A=Q1       'Set Acceleration to user variable Q1
```

MNEMONIC **AL**	Function: Retrieve All Parameters
	Type: Variable
	Usage: I, R
	Syntax: PR AL

Description:

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

Usage Example

```
PR AL
```

MNEMONIC **AS**	For Internal IMS Use Only: Will return an error if used. Do Not use for a program label or user variable or flag.

MNEMONIC **AT**	For Internal IMS Use Only: Will return an error if used. Do Not use for a program label or user variable or flag.

MNEMONIC	Function: BAUD Rate	Units: Bits per second
BD	Type: Setup Variable	Range: 48, 96, 19, 38, 11
	Usage: P/I, R/W	Default: 9600 bps
	Syntax:BD=<value>	Related: CK

Description:

This variable sets the baud rate for serial communications with the MCode device. It sets the rate for the RS-422/485 interface. The baud rate is set by indicating the first two digits of the desired rate as shown in the range section below.

In order for the new BAUD rate to take effect, the user must issue the S (SAVE) instruction and then reset the device. When the MCode device is reset, it will communicate at the new BAUD rate.

Range: 48 = 4800 bps, 96 = 9600 bps, 19 = 19200 bps, 38 = 38400 bps, 11 = 115200 bps

Note: If you change the Baud Rate in the MCode device it must be matched in IMS Terminal.

Note: A delay time between the command requests to the device must be considered to allow it time to interpret a command and answer the host before a subsequent command can be sent. The time between requests is dependent on the command and the corresponding response from the device.

Usage Example

```
BD=19    'set communications BAUD Rate to 19200 bps
S        'Save
```

Program Example

Conditional:
LB XX
BR XX, I2=1

Unconditional:
LB XX
BR YY

LB YY

MNEMONIC	Function: Break Point
BP	Type: Instruction
	Usage: I
	Syntax: BP <address/label, count>

Description:

The BP, or Break Point Instruction allows the user to set break points within an MCode program to help in debugging the program.

To use the BP instruction the program must be executed in either trace or single-step mode. The prgram will then run normally the number of times specified by the count, then go into single-step mode at the address or label specified by BP. Press the spacebar to step through the program if in single step mode.

To disable the break point, set BP=0.

Usage Example

```
BP X1, 3       'Break Program at label X1 after 3 cycles

EX P1, 1       'Execute Program P1 in trace mode
or
EX P1,2        'Execute P1 in single step mode
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background: Applies to Plus^2 Models Only

MNEMONIC **BR**	Function: Branch (Conditional or Unconditional)
	Type: Program Instruction
	Usage: P
	Syntax: BR <address/label, condition>

Description:

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

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

Conditional Branch

This type of branch first specifies an address or user label where program execution should continue if the second parameter, the condition, is true. The condition parameter may include flags as well as logical functions that are to be evaluated. Only one condition may exist.

Unconditional Branch

In this type of branch the second parameter is not specified, then the execution will continue at the label or address specified by the first parameter.

Usage Example

```
BR 256, I2=1    'Cond. branch to address 256 if input 1 = ACTIVE
BR G1           'Unconditional branch to program label G1
BR G2, Q4<10    'Cond branch to program G2 if user var Q4 is less than 10
```

MNEMONIC **BY**	Function: Busy Flag	Units: —
	Type: Read Only Flag	Response: 0/1
	Usage: P/I, R	Default: 0
	Syntax: PR BY	Related: PR

Description:

This read only status flag will indicate if a Program is executing (1) or not running (0).

Usage Example

```
PR BY  'No Program Running response: BY=0, Program Running response: BY=1
```

MNEMONIC		
C1	Function: Counter 1	Units: Motor Steps
	Type: Motion Variable	Range: -2147483648 to 2147483647
	Usage: P/I, R/W	Default: 0
	Syntax:C1=<steps>	Related: C2, P

Description:

This variable contains the motor steps representation of the clock pulses generated by the MCode compatible device. Counter 1 may be preset if necessary

Usage Example

```
C1=20000          'Set counter 1 to 20000 motor steps
PR C1             'Print the value of C1 to the terminal screen
CL K5,C1>2100000  'Call subroutine K5 if C1>2100000
```

MNEMONIC		
C2	Function: Counter 2	Units: Encoder Counts
	Type: Motion Variable	Range: -2147483648 to 2147483647
	Usage: P/I, R/W	Default: 0
	Syntax:C2=<counts>	Related: C1, P

Description:

This variable contains the raw count representation of the encoder. Counter 2 may be preset if necessary.

Usage Example

```
C2=512            'Set counter 2 to 512 encoder counts
PR C2             'Print the value of C2 to the terminal screen
CL K2,C2>512000   'Call subroutine K2 if C2>512000
```

MNEMONIC	
CC	For Internal IMS Use Only: Will return an error if used. Do Not use for a program label or user variable or flag.

MNEMONIC		
CE	Function: Software Reset Enable (CTRL+C)	Units: —
	Type: Setup Flag	Range: 0 - 2
	Usage: P/I, R/W	Default: 1
	Syntax: CE=<0-2>	Related:

Description:

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

Usage Example

```
CE=0   'Disables CTRL+C response
CE=1   'Enables CTRL+C response, default
CE=2   'Is addressable in party mode (PY=1) eg A ^C, will respond same as
       'CE=1 when not in party mode (PY=0)
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background: Applies to Plus^2 Models Only

MNEMONIC **CK**	Function: Check Sum Enable	Units: —
	Type: Flag	Range: 0/1/2
	Usage: P/I, R/W	Default: 0
	Syntax: CK=<0/1>	Related: BD

Description:

CK=1 puts the device into Check Sum Mode. When enabled, all communications with the device require a Check Sum to follow thel commands. The Check Sum is the 2's complement of the 7 bit sum of the ASCII value of all the characters in the command "OR"ed with 128 (hex = 0x80). The command will be acknowledged with a NAK (0x15) if the Check Sum is incorrect or an ACK (0x06) when the command is correctly processed (no error).

CK=2 will enable check sum mode, however NAK only sent for bad check sum. "ACK" is not echoed if a program is running. Only a NAK is echoed if an error occurs. In immediate mode both ACK or NAK characters are echoed.

Usage Example

To Send the checksum, in IMS terminal use ALT+ Checksum (In the example ALT+0144) The Response will be 06

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

Program Example

Conditional:
LB XX
CL XX, I2=1

Unconditional:
LB XX
CL YY

LB YY

MNEMONIC **CL**	Function: Call Subroutine	
	Type: Program Instruction	
	Usage: P	
	Syntax: CL <address/label, condition>	Related: RT

Description:

This function can be used to invoke a subroutine within a program. This allows the user to segment code and call a subroutine from a number of places rather than repeating code within a program.

There are two parameters to the CL instruction. The first specifies the program address or label of the subroutine to be invoked if the second parameter, the condition, is true. If the second parameter is not specified, the subroutine specified by the first parameter is always invoked. The condition parameter can include flags as well as logical functions that are to be evaluated. There can only be one condition.

The subroutine should end with a RT (Return) instruction. The RT instruction will cause program execution to return to the line following the CL instruction.

Usage Example

```
CL 256, I5<512    'Call Subroutine at 256, analog input less than 512
CL K5             'Unconditional call to subroutine label K5
CL K8, I4=0       'Call subroutine k8 if input 4 is INACTIVE
```

MNEMONIC PLUS^2 ONLY **CM**	Function: Clock Mode Enable	Units: —
	Type: Flag	Range: 0/1
	Usage: P/I, R/W	Default: 0 (disabled)
	Syntax: CM=<0/1>	Related: S7, S8, CR, CW, FM

Description:

This flag (when CM=1) will enable the clock I/O 7 and 8. If S7 and S8 are set as Input type, the device will be in follower mode. If S7 and S8 are set as output types, signal will be motor step counts out.

Usage Example

```
CM=1    'Enable Clock Mode
```

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

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

MNEMONIC **CP**	Function: Clear Program	
	Type: Program Instruction	
	Usage: I	
	Syntax: CP <address/label>	Related: FD, IP

Description:

This instruction will clear the program space in the NVM as specified by the instruction parameter. Programs are stored directly to the NVM and executed from there. Will clear program addresses only. Will not clear globally declared user variable or flags. FD will clear program memory as well.

Usage Example

```
CP 256    'Clear program space beginning at address 256
CP G3     'Clear program space beginning at label G3
CP        'Clear all of program space
```

MNEMONIC PLUS^2 ONLY **CR**	Function: Clock Ratio	Units: —
	Type: Motion Variable	Range: 0.001 to 2.000
	Usage: P/I, R/W	Default: 1.000
	Syntax:CR=<0.00-2.00>	Related: S7, S8, CM, CW

Description:

Clock Ratio value for electronic gearing. The value selected will set the ratio from the clock input on I/O 7 (Step Clock) and I/O 8 (Direction) to the drive output. A Clock Ratio set to 0.50 will cause the frequency pulses sent to the driver section of the device to be 0.50 of the frequency input to the device, this would cause the device to "follow" at half the velocity of the primary axis.

Note: A value of 1.00 has no acceleration or deceleration. All other values use the value of A and D.

The value of CR will have no impact on a clock output. CR will impact Step/Direction in, Clock Up/ Clock Down in and Quadrature in when I/O 7 and 8 are configured as inputs.

Usage Example

```
CR=1.500    'Ratio set to 1.5
```

MNEMONIC PLUS^2 ONLY **CW**	Function: Clock Width	Units: Nanoseconds
	Type: Motion Variable	Range: 0 - 255
	Usage: P/I, R/W	Default: 10
	Syntax:CW=<0-255>	Related: S7, S8, S13, CM, PC

Description:

When CM=1 and I/O 7 and 8 are configured as outputs, CW sets the pulse width of the output signal. The setting will be a multiplier x 50 nS. CW will set the output clock pulse width for: Step Clock out, Clock Up/Clock Down out, Quadrature out and the Trip output (I/O 13)

Usage Example

```
CW=100    'sets output step width to 5µS (100 x 50nS)
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background: Applies to Plus^2 Models Only

MNEMONIC **D**	Function: Deceleration	Units: Steps/Sec^2 (EE=0)/Counts/Sec^2 (EE=1)
	Type: Motion Variable	Range: 91 to 1525878997 (Steps EE=0), 91 to 61035160 (Counts EE=1)
	Usage: P/I, R/W	Default: 1000000(EE=0), 40000 (EE=1)
	Syntax:D=<value>	Related: C1, C2, P

Description:

The D variable sets the deceleration of the device in steps per second^2. If the D was set at 76800 per second^2 the motor would decelerate at a rate of 76800 per second, every second. If the device was running at a maximum velocity of 768000 microsteps per second it would take 10 seconds to decelerate.

Usage Example

```
D=20000     'set acceleration to 20000 step/sec2
D=A         'set deceleration equal to acceleration
```

MNEMONIC **D1-D4**	Function: Input Switch Debounce	Units: Milliseconds
	Type: I/O Variable	Range: 0 to 255
	Usage: P/I, R/W	Default: 0
	Syntax:D1-4=<time>	Related: I1 - I4

Description:

This variable will set the digital filtering to be applied to the selected input 1 - 4. The input must be stable for "time" amount of milliseconds before a change in state is detected.

Usage Example

```
D1=0     'No debounce
D1=150   'Set filtering to 150 msec
```

MNEMONIC **D5**	Function: Analog Input Filter	Units: counts
	Type: I/O Variable	Range: 0 to 255
	Usage: P/I, R/W	Default: 0
	Syntax:D5=<X>	Related:

Description:

D5 is a continuous filtering process. It does a running average by computing:

(((X-1)/X)*current reading) + (1 / X) If X = 10, then: ((current averaged value * 9)/10) + (new reading / 10) == NEW current averaged value.

Usage Example

```
D5=0     'No debounce
D5=100   'Set filtering to 100 counts
```

MNEMONIC PLUS^2 ONLY **D9-D12**	Function: Input Switch Debounce	Units: Milliseconds
	Type: I/O Variable	Range: 0 to 255
	Usage: P/I, R/W	Default: 0
	Syntax:D9-12=<time>	Related: I9 - I12

Description:

This variable will set the digital filtering to be applied to the selected input 9 - 12. The input must be stable for "time" amount of milliseconds before a change in state is available.

Usage Example

```
D10=0     'No debounce on input 10
D11=150   'Set filtering to 150 msec on input 11
```

MNEMONIC **DB**	Function:Encoder Deadband	Units: Encoder Counts
	Type: Setup Variable	Range: 0 to 65000
	Usage: P/I, R/W	Default: 1
	Syntax:DB=<counts>	Related: EE, C2, SF, SM, ST, PM, EL

Description:

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

When the encoder is enabled, a move is not completed until motion stops within DB. If PM=1 device will correct if pushed outside of DB value once in position.

Usage Example

```
DB=10    'Set encoder deadband to ±10 counts
```

MNEMONIC **DC**	Function: Decrement Variable	
	Type: Program Instruction	
	Usage: P/I	
	Syntax: DC <variable>	Related: IC

Description:

The DC instruction will decrement the specified variable by one.

Usage Example

```
DC R1   'Decrement register r1
DC K5   'Decrement user variable K5
```

MNEMONIC **DE**	Function: Drive Enable	Units: —
	Type: Setup Flag	Range: 0/1
	Usage: P/I, R/W	Default: 1 (Enabled)
	Syntax: DE=<0/1>	Related: —

Description:

The DE flag enables or disables the drive portion of the MCode compatible device.

Usage Example

```
DE=0    'Disable the motor driver section
DE=1    'Enable the motor driver section
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding


Variable


Flag


Instruction


Body Background:
Applies to Plus^2
Models Only

MNEMONIC **DG**	Function: Disable Global	Units: —
	Type: Setup Flag	Range: 0/1
	Usage: P/I, R/W	Default: 1 (Disabled)
	Syntax: DG=<0/1>	Related: DN, PY

Description:

The DG flag enables or disables device response to global commands made while in Party Mode. In the default state (DG=1) the device will not respond but will execute global commands. By setting the the DG flag to 0, that device will respond to global commands.

Note: only one device on the communications bus should be set to DG=0.

Usage Example

```
DG=0    'Enable response to global commands
DG=1    'Disable Response to global commands no echo
```

MNEMONIC **DN**	Function: Device Name	Units: ASCII Characters
	Type: Setup Variable	Range: a-z, A-Z, 0-9
	Usage: P/I, R/W	Default: !
	Syntax:DN=<"ascii char">	Related: PY, S

Description:

DN sets the name of the device for party mode communications. The acceptable range of characters is a-z, A-Z, 0-9. The factory default is "!" Once named, the device name must precede the instruction to that drive. When assigning a device name, the character MUST be within quotation marks.

The name is case sensitive. Refer to Section 2.2 in the Hardware Reference for specific Party Mode configuration and use instructions.

Once the default character has been changed it can not be reset except on a FD (Factory Default Reset)

Usage Example

```
DN="A"    'Set the device name to the character A
DN="65"   'Set the device name to the character A*
```

Note: Must enter S (Save) prior to any reset of the DN will be lost.

*See ASCII Table, Appendix A for Character codes

MNEMONIC	Function: End Program Instruction	
E	Type: Program Instruction	
	Usage: P	
	Syntax: E	Related:PG, EX

Description:

Stops the execution of a program. Used in program mode to designate the end of the program

Usage Example

```
E    'End program
```

MNEMONIC	Function: Encoder Enable	Units: —
EE	Type: Setup Flag	Range: 0/1
	Usage: P/I, R/W	Default: 0 (Disabled)
	Syntax: EE=<0/1>	Related: DB, C2, SF, SM, ST, PM, FM, EL

Description:

The EE flag enables or disables the optional encoder mode of the MCode compatible device. When in Encoder Mode, all moves are done by Encoder Counts. The 512 line Encoder generates counts in a Quadrature format which results in 2048 counts per revolution.

Usage Example

```
EE=0    'Disable the encoder
EE=1    'Enable encoder mode
```

MNEMONIC	Function: Error Flag	Units: —
EF	Type: Status Flag	Response: 0/1
	Usage: P/I, R	Default: —
	Syntax: PR ER	Related: ER, OE

Description:

The Error flag will indicate whether or not an error condition exists. It is automatically cleared when a new program is executed. The only way to manually clear the EF flag is to read the value of the ER variable or set ER=0

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

Usage Example

```
PR EF     'Read the state of the error flag
          'Response = 0: No error exists
          'Response = 1: Error condition exists, Error value exists
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background: Applies to Plus^2 Models Only

MNEMONIC PLUS^2 ONLY **EL**	Function: Encoder Lines	Units: Encoder Lines
	Type: Setup Variable	Range: —
	Usage: P/I, R/W	Default: 512
	Syntax:EL=<lines>	Related: C2, MS, SF, SM, ST, PM, FM

Description:

This variable defines the number of encoder lines that the device will see in a revolution. Counter 2 will read 4 x EL, or 4 counts per line.

Note : The setting for MS (Microstep Resolution) is relative to the EL Setting. To calculate the minimum value for MS use the following equation:

$MS_{minimum}$ = (EL x 8) ÷ 200

Example for 1000 line encoder: 1000 X 8 = 8000, 8000 ÷ 200 = 40

Minumum Microstep Resolution = 50, or 10000 steps/rev. Note: IMS recommends leaving the Microstep Resolution at the default of 256.

Usage Example

```
EL=1000    'Configure EL variable for a 1000 line encoder
MS=50      'Configure microstep resolution to 10000 steps/rev
```

MNEMONIC **EM**	Function: Echo Mode	Units: —
	Type: Setup Flag	Range: 0-3
	Usage: P/I, R/W	Default: 0 (Full Duplex)
	Syntax: EM=<0-3>	Related: BD, CK, PR, L

Description:

The Echo Mode Flag will set the full/half duplex configuration of the RS-485 channel. 0=Full Duplex (default), 1=Half Duplex, 2=Only respond to PR and L, 3=Prints after command is terminated.

Usage Example

```
EM=0   'Echo all information back over communications line. CR/LF
       'Indicates Command Accepted (Full Duplex)
EM=1   'Don't echo the information, only send back prompt. CR/LF
       'Indicates Command Accepted (Half Duplex)
EM=2   'Does not send prompt, only responds to PRINT (PR) and
       'LIST (L) commands
EM=3   'Saves Echo in Print Queue then executes. Prints after command
       'is terminated.
```

MNEMONIC **ER**	Function: Error Number	Units: Numeric Error Code
	Type: Status Variable	Response: Numeric Error Code
	Usage: P/I, R/W	Default: 0
	Syntax: PR ER	Related: EF, OE

Description:

The ER variable indicates the MCode device error code for the most recent error that has occurred. The ER variable must be read or set to zero to clear the EF flag.

A Question Mark <?> in place of the normal cursor indicates an ERROR. A list of Error codes are located at the end of this section.

Usage Example

```
PR ER    'Read the error number, result = <Value> (See Error Codes
         'at the end of this section.
ER 0     'Set the error value to zero
```

MNEMONIC	Function: Escape	Units: —
ES	Type: Setup Flag	Range: 0-3
	Usage: —	Default: 1 (ESC)
	Syntax: ES=<0/1>	Related: —

Description:

ESC flag to switch between ESC and CTRL+E. An Escape will stop both the program and the motion.

Usage Example

```
ES=0    'Escape Flag set to respond to CTRL+E
ES=1    'Escape Flag set to respond to ESC keypress (default)
ES=2    'Escape Flag set to respond addressable CTRL+E (party mode)
ES=3    'Escape Flag set to respond to addressable ESC keypress (party mode)
```

Program Example

```
DN="A"
ES=3

String 0x41 0x1B

Will stop motion and program
```

MNEMONIC	Function: Execute Program	
EX	Type: Program Instruction	
	Usage: I	
	Syntax: EX=<label/address>,<mode>	Related: PG, E

Description:

Execute program at a specified address or label using a selected trace mode. Used in immediate mode.

There are three modes of program execution.

Mode 0 Normal execution, is specified by a mode of 0 (or simply leaving the mode blank).

Mode 1 Trace mode is specified by a mode of 1. This means that the program executes continuously until the program E is encountered, but the instructions are "traced" to the communications port so the user can see what instructions have been executed.

Mode 2 Single step mode is specified by a mode of 2. In this mode, the user can step through the program using the space bar to execute the next line of the program. The program can be resumed at normal speed in this mode by pressing the enter key.

Usage Example

```
EX 1      'Execute program at address 1 normally
EX G2,1   'Execute program G2 in trace mode
EX 200,2  'Execute program at address 200 in single-step mode
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background: Applies to Plus^2 Models Only

MNEMONIC PLUS^2 ONLY **FC**	Function: Filter Capture	Units: Numeric
	Type: I/O Variable	Range: 0 - 9
	Usage: P/I, R/W	Default: 0 (min pulse 50 nS, cutoff of 10 MHz)
	Syntax: FC=<0-9>	Related: I13, TC, S13

Description: This variable will set the digital filtering to be aplied to I/O 13 when configured as a Capture input. The input must be stable for "time" amount of milliseconds before a change in state is detected.

Input 13 Filter Capture Settings		
Range	Min Pulse	Cutoff Frequency
0	50 nS	10 MHz
1	150 nS	3.3 MHz
2	200 nS	2.5 MHz
3	300 nS	1.67 MHz
4	500 nS	1.0 MHz
5	900 nS	555 kHz
6	1.7 µS	294.1 kHz
7	3.3 µS	151 kHz
8	6.5 µS	76.9 kHz
9	12.9 µS	38.8 kHz

Table 3.1: Input 13 Filter Capture Settings

Usage Example

```
FC=3    'Set input filtering for Input 13 min pulse to 200 nS/CO 2.5 MHz
```

MNEMONIC **FD**	Function: Restore Factory Defaults	
	Type: Program Instruction	
	Usage: I	
	Syntax: FD <carriage return>	Related: CP, IP

Description:

FD will clear all program memory and return the MCode compatible device to factory default settings. The response will be the IMS sign on message. FD will clear programs and initialize parameters.

FD will generate an error 73 if the unit is in motion.

Usage Example

```
FD  'Restore device to factory default state
Response "Copyright 2001-2007 by Intelligent Motion Systems, Inc."
```

MNEMONIC PLUS^2 ONLY **FM**	Function: Filter Motion	Units: Numeric
	Type: I/O Variable	Range: 0 - 9
	Usage: P/I, R/W	Default: 2 (min pulse 200 nS, cutoff of 2.5 MHz)
	Syntax: FM=<0-9>	Related: I7, I8, C2, EL, EE, S7, S8

Description:

Filter Motion inputs, I/O 7 & 8 for electronic gearing and optional encoder.

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

Table 3.2: Input 7 & 8 and Capture Filter Settings

Usage Example

```
FM=4    'Set input filtering for Input 7 & 8 max input frequency of 1 MHz
```

MNEMONIC **FT**	For Internal IMS Use Only: Will return an error if used. Do not use for a program label or user variable or flag.

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background: Applies to Plus^2 Models Only

MNEMONIC **H**	Function: Hold Program Execution	Unit: Milliseconds
	Type: Program Instruction	Range: 1 - 65000
	Usage: P/I	Default: 0
	Syntax: H <time>	Related: PG, E, EX

Description:

The hold instruction is used in a program to suspend program execution. If no parameter is specified the execution of the program will be suspended while motion is in progress. This will typically be used following a MA, MR, HI or HM instruction.

A time in milliseconds may be placed as a parameter to the hold instruction, This will suspend program execution for the specified number of milliseconds.

Usage Example

```
MA 1000   ' Move Absolute 1000 steps
H         'Suspend program execution until motion completes
          '(used after a move command)

H 2000    'Suspend program execution for 2 seconds
```

MNEMONIC **HC**	Function: Hold Current	Units: Percent (%)
	Type: Setup Variable	Range: See Table
	Usage: P/I, R/W	Default: 5
	Syntax: HC=<%>	Related: HT, RC, MT

Description:

This variable defines the motor holding current in percent.

Usage Example

```
HC=25    'set the motor holding current to 25%
```

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

Shaded Area Reserved for Future Use

*HC=67 for maximum 2.0 Amp Hold Current

MNEMONIC	Function: Home To Index Mark	Unit: Numeric Type
HI	Type: Program Instruction	Types: 1-4
	Usage:P/I	Default: —
	Syntax: HI<type>	Related: VM, VI, EE, I6, HM

Description:

This instruction will find the the encoder index mark. There are four combinations for this command. (See Use below.)

When HI is executed, the axis moves in the direction specified by the (S) at VM until it reaches the index mark. It then creeps off of the index in the direction specified by the sign of (C) at VI. Motion is stopped as soon as the index changes state. Note that Speed and Creep is set by the VM and VI commands.

1) Speed: Specifies the direction and speed that the axis will move until the switch is activated (VM).
2) Creep: Specifies the direction and speed that the axis will move off the switch until it becomes inactive again (VI).

The diagram on the following page illustrates the different scenarios possible during the Home to Index Mark(HI) sequence. The diagrams represent the four HI types. The four types are listed below

HI 1 Slew at VM in the minus direction and Creep at VI in the plus direction.

HI 2 Slew at VM in the minus direction and Creep at VI in the minus direction.

HI 3 Slew at VM in the plus direction and Creep at VI in the minus direction.

HI 4 Slew at VM in the plus direction and Creep at VI in the plus direction.

Usage Example

```
HI 2    'Slew at VM in the minus direction and Creep at VI in the
        'minus direction
```

MNEMONIC	Function: Find Home Switch	Unit: Numeric Type
HM	Type: Program Instruction	Types: 1-4
	Usage:P/I	Default: —
	Syntax: HM <type>	Related: VM, VI, EE, I6, HI, LM, S<1-4>, <9-12>

Description:

This instruction will find the selected I/O switch assigned to “Home”.

1) Speed: Specifies the direction and speed that the axis will move until the switch is activated (VM).
2) Creep: Specifies the direction and speed that the axis will move off the switch until it becomes inactive again (VI).

When HM is executed, the axis moves at VM in the direction specified by the sign of speed. It then creeps off of the switch at VI in the direction specified by the sign of creep. Motion is stopped as soon as the switch becomes deactivated. Note that Speed and Creep is set by the VM and VI commands.

The diagram on the following page illustrates the different scenarios possible during the Homing (HM) sequence. The diagrams represent the four HM commands. Below are the four combinations of the HM command.

HM 1 Slew at VM in the minus direction and Creep at VI in the plus direction.

HM 2 Slew at VM in the minus direction and Creep at VI in the minus direction.

HM 3 Slew at VM in the plus direction and Creep at VI in the minus direction.

HM 4 Slew at VM in the plus direction and Creep at VI in the plus direction.

The key to the diagrams is as follows.

1 - Slew at VM to find the Index Mark.

2 - Decelerate to zero (0) after finding the Index Mark.

3 - Creep at VI away from the Index Mark.

4 - Stop when at the edge of the Index Mark.

Usage Example

```
HM 2    'Slew at VM in the minus direction and Creep at VI in the
        'minus direction
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

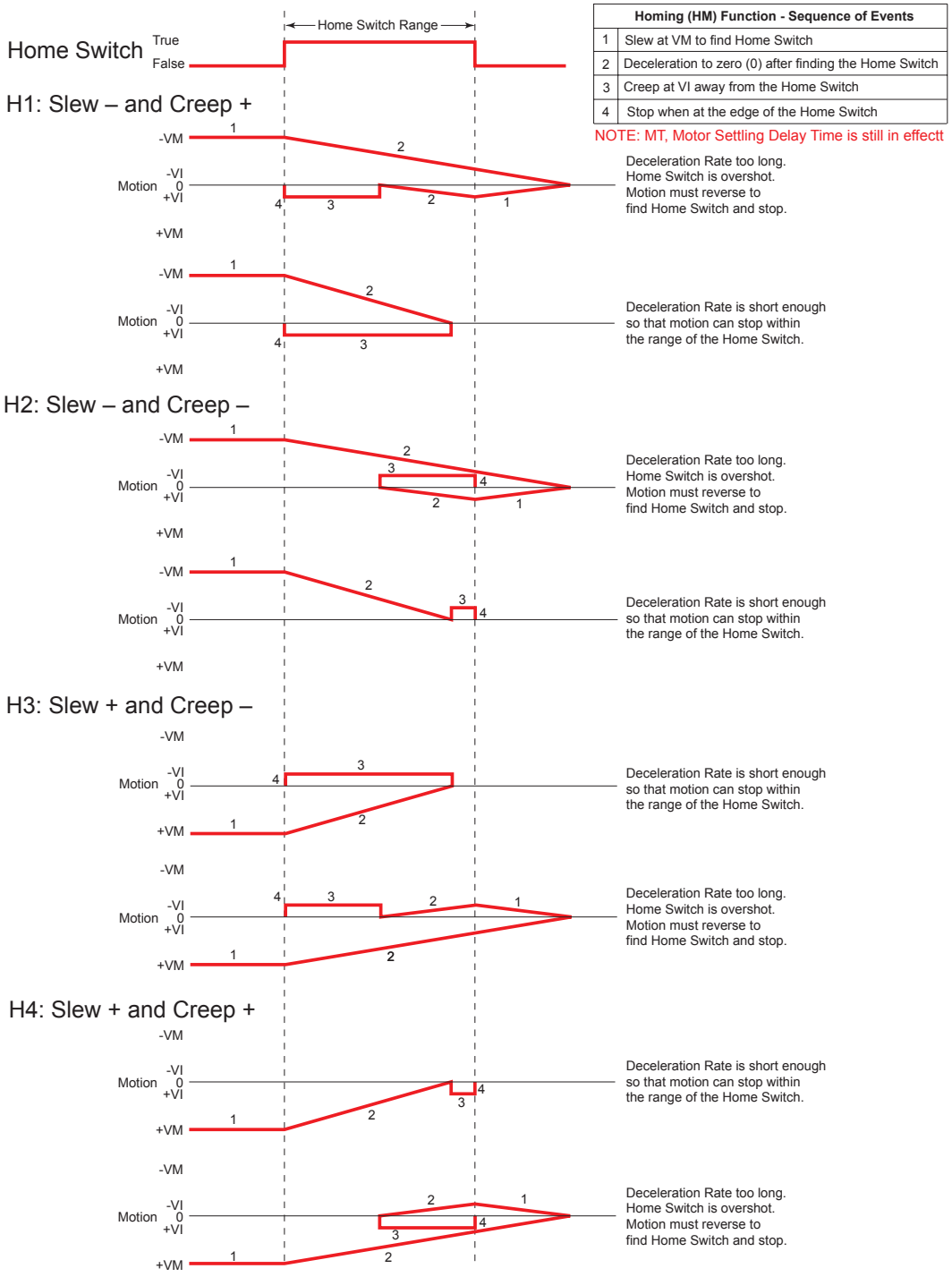
Color Coding

Variable

Flag

Instruction

Body Background:
Applies to Plus^2
Models Only



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

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

Figure 3.1 Homing Functions Sequence of Events

MNEMONIC	Function: Hold Current Delay Time	Units: Milliseconds
HT	Type: Setup Variable	Range: 0 or 2 - 65535
	Usage: P/I, R/W	Default: 500
	Syntax: HT=<time>	Related: HC, MT, RC

Description:

The HT variable sets the delay time in milliseconds between the MV=0 and when the device shifts to the holding current level specified by the HC (Motor Holding Current) variable. The delay time is also effected by the MT (Motor Settling Delay Time) variable in that the total time to current change is represented by the sum of MT + HT. The total of MT+HT cannot add up to more than 65535, thus the value of MT is included in the HT range.

Thus the Maximum setting for HT=(65535-MT). If HT=0, the current will not reduce.

Usage Example

```
HT=1500   'set hold current delay time to 1.5 seconds
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background: Applies to Plus^2 Models Only

MNEMONIC **I1 - I4**	Function: Read Input	Units: Logic State
	Type: I/O Variable	Range: 0/1
	Usage: P/I, R	Default: —
	Syntax: PR I<1-4> BR <addr/lbl>, I<1-4>=<1/0> CL<addr/lbl>, I<1-4>=<1/0>	Related: IL, IN, O1-O4, S1-S4

Description:

This variable will read the state of the specified input 1 - 4. Can be used with PR (Print), BR (Branch) and CL (Call Subroutine) instructions. Can also be used with R1 - R4 and User Variables.

The value of the bit state will be dependant on active (low/high) state of the input, specified by the S<1-4> 2nd parameter.

Usage Example

```
PR I2          'Prints the logic state of input 2
BR 128,I3=1    'Conditional branch to address 125, Input 3 ACTIVE
CL K9, I4=0    'Call subroutine K9, Input 4 INACTIVE
```

MNEMONIC **I5**	Function: Read Analog Input	Units: Numeric Value
	Type: I/O Variable	Range: 0 to 1023
	Usage: P/I, R	Default: —
	Syntax: PR I5 BR <addr/lbl>,I5=<0-1023> CL<addr/lbl>, I5<0-1023>	Related: S5

Description:

This variable will read the value of the correlating bit value seen on the Analog Input. Can be used with PR (Print), BR (Branch) and CL (Call Subroutine) instructions. The value read will between 0 and 1023.

This value represents a voltage or current being seen on the analog input. for example, if in current mode (0 - 20mA range) 1023 would represent 20 mA, 512 would represent 10 mA.

Usage Example

```
PR I5          'Print the value of I5
BR G3,I5>512   'Branch to program G3 if I5 is greater than 512
CL 423,I5<220  'Call subroutine at address 423 if I5 is less than 220
```

MNEMONIC **I6**	Function: Read Encoder Index Mark	Units: Logic State
	Type: I/O Variable	Range: 0/1
	Usage: P/I, R	Default: —
	Syntax: PR I6 BR <addr/lbl>,I6=<0/1> CL<addr/lbl>, I6<0/1>	Related: PR, BR, CL

Description:

This variable will read the on/off state of the Encoder Index Mark. Can be used with PR (Print), BR (Branch) and CL (Call Subroutine) instructions. The value read will be 0 (off mark) or 1 (on mark).

Usage Example

```
PR I6          'Print the on/off state of the encoder index mark
BR 324,I6=1    'Branch to address 324 if I6 is ACTIVE
CL K3,I6=1     'Call subroutine K3 if I6 is ACTIVE
```

MNEMONIC **I7-8, I13**	Reserved. Do not use as a user variable or label.

MNEMONIC	Function: Read Input	Units: Logic State
I9 - I12	Type: I/O Variable	Range: 0/1
	Usage: P/I, R	Default: —
	Syntax:PR <I9-I12> BR <addr/lbl>, I<9-12>=<1/0> CL<addr/lbl>, I<9-12>=<1/0>	Related: IH, IN, O9-O12, S9-S12

Description:

This variable will read the state of the specified input 9 - 12. Can be used with PR (Print), BR (Branch) and CL (Call Subroutine) instructions.

The value of the bit state will be dependent on active (low/high) state of the input, specified by the S<9-12> 2nd Parameter.

Usage Example

```
PR I12          'Prints the logic state of input 12
BR 128,I11=1    'Conditional branch to address 125, Input 11 is ACTIVE
CL K9, I10=0    'Call subroutine K9, Input 10 NOT ACTIVE
```

MNEMONIC	Function: Increment Variable	
IC	Type: Program Instruction	
	Usage: P/I	
	Syntax: IC <variable>	Related: DC

Description:

The IC instruction will increment the specified variable by one.

Usage Example

```
IC R1   'Increment register R1
IC K5   'Increment user variable K5
```

MNEMONIC	Function: Input Variable Pending	Units: —
IF	Type: Setup Flag	Range: 0/1
	Usage: P/I, R	Default: 0
	Syntax: IF=<0/1>	Related: IV

Description:

The IF instruction is automatically set to 1 when IV command is executed. The IF flag reflects an input value from serial port is pending, not that one has been received. IF will be cleared to zero (0) with a carriage return or can be reset manually.

Usage Example

```
No Usage Example, Flag set automatically by IV
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding


Variable


Flag


Instruction


Body Background: Applies to Plus^2 Models Only

MNEMONIC **IL**	Function: Read Inputs 1-4 As One Value	Units: Decimal Value
	Type: I/O Variable	Range: 0-15
	Usage: P/I, R	Default: —
	Syntax: PR IL BR <addr/lbl>,IL=<0-15> CL<addr/lbl>, IL<0-15>	Related: IH, IN, OL, OH, OT, S1-S4

Description:

This keyword will read the binary state of inputs 1-4 and print them as a decimal value. When used thus, Input 1 is the Least Significant Bit (LSb) and Input 4 is the Most Significant Bit (MSb). It may be used in conjunction with the R1-R4 (Registers), PR (Print), BR (Branch) and CL (Call Subroutine) instructions. The value is a function of the actual voltage level of the I/O where 1 = +V and 0 = Ground. (Not a function of the active state defined in S1 to S4 variables).

Usage Example

```
PR IL          'Print the decimal value of IO4-IO1 to the terminal
BR 324,IL=8    'Branch to address 324 if IL=8
CL K3,IL=13    'Call subroutine K3 if IL=13
```

MNEMONIC PLUS^2 ONLY **IH**	Function: Read Inputs 9-12 As One Value	Units: Decimal Value
	Type: I/O Variable	Range: 0-15
	Usage: P/I, R	Default: —
	Syntax: PR IH BR <addr/lbl>,IH=<0-15> CL<addr/lbl>, IH<0-15>	Related: IL, IN, OL, OH, OT, S1-S4, S9-S12

Description:

This keyword will read the binary state of inputs 9-12 and print them as a decimal value. When used thus, Input 9 is the Least Significant Bit (LSb) and Input 12 is the Most Significant Bit (MSb). It may be used in conjunction with the R1-R4 (Registers), PR (Print), BR (Branch) and CL (Call Subroutine) instructions. The value is a function of the actual state of the I/O where 1 = +V and 0 = Ground. (Not a function of the active state defined in S9 to S12 variables).

Usage Example

```
PR IH          'Print the decimal value of IO4-IO1 to the terminal
BR 324,IH=8    'Branch to address 324 if IH=8
CL K3,IH=13    'Call subroutine K3 if IH=13
```

MNEMONIC	Function: Read Inputs 1-4 and 9-12 As One Value	Units: Decimal Value
IN	Type: I/O Variable	Range: 0-255
	Usage: P/I, R	Default: —
	Syntax: PR IN BR <addr/lbl>,IN=<0-255> CL<addr/lbl>, IN<0-255>	Related: IH, IL, OL, OH, OT, S1-S4, S9-S12

Description:

This keyword will read the binary state of inputs 1-4 and 9-12 and print them as a decimal value. When used thus, Input 1 is the Least Significant Bit (LSb) and Input 12 is the Most Significant Bit (MSb). It may be used in conjunction with PR (Print), BR (Branch) and CL (Call Subroutine) instructions. The value is a function of the actual state of the IO where 1 = +V and 0 = Ground. (Not a function of the active state defined in S1 to S4 and S9 to S12 variables). On Standard Plus Models IN will only read the lower group (Inputs 1-4)

Usage Example

```
PR IN            'Print the decimal value of IO4-IO1 and IO9-IO12
                 'to the terminal
BR 324,IN=225    'Branch to address 324 if IN=225
CL K3,IN=113     'Call subroutine K3 if IN=113
```

MNEMONIC	Function: Initiialize Parameters	
IP	Type: Instruction	
	Usage: P/I	
	Syntax: IP	Related: S, FD

Description:

The IP instruction will return all of the device variable and flag parameters to their stored values.

If IP is used while the motor is moving an Error 74: Tried to Initialize Parameters of Clear Program while moving will be issued.

Usage Example

```
IP   'Initialize parameters
```

MNEMONIC	Function: Internal Temperature	Units: Degrees Celsius
IT	Type: Read Only Variable	Range: -55°C to 125°C
	Usage: R	Default: —
	Syntax: PR IT	Related: WT

Description:

Internal Temperature of the MCode compatible driver electronics. Only used on MDrive34Plus and AC input MDrivePlus.

Usage Example

```
PR IT   'Print the internal temperature to the terminal screen
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background: Applies to Plus^2 Models Only

MNEMONIC **IV**	Function: Input Into Variable Instruction	
	Type: Instruction	
	Usage: P/I	
	Syntax: IV <user variable/R1-R4>	Related: IF

Description:

With the IV command, a user may input new variable values. These values must be numeric and will be input into the variable specified in the IV command.

The variable used for the IV may be a system or USER Variable. A USER Variable must be declared prior to the IV command.

When waiting for user input, there must be a conditional program loop based upon the state of IF (Input Pending) until the variable is input by the user.

Usage Example

```
IV R1   'Input data into R1
VA K5   'Create user variable K5
IV K5   'Input data into user variable K5

IV R1            'input value into register 1
     LB k1       'label program loop k1
     BR k1,If=1  'branch to k1 while awaiting user input
```

MNEMONIC **JE**	Function: Jog Enable	Units: —
	Type: Setup Flag	Range: 0/1
	Usage: P/I, R/W	Default: 0 (Disabled)
	Syntax: JE=<0/1>	Related: VM, A, D, VI, S1-S4, S9-S12

Description:

This command will enable Jog Mode if I/O are set for Jog Plus and/or Jog Minus. States are 0=Disabled, 1=Enabled.

Usage Example

```
JE=0    'Disable Jog mode
JE=1    'Enable Jog mode
```

MNEMONIC	Function: List Program Space	
L	Type: Instruction	
	Usage: I	
	Syntax: L <address/label>	Related: CP, FD

Description:

The L instruction will print the contents of program space beginning at the specified address to the end. If no address is specified it will list beginning at address 1.

Usage Example

```
L      'List the contents of program space beginning at address 1
L G5   'List the contents of program space beginning ar label G5
```

MNEMONIC	Function: Label Program or Subroutine	
LB	Type: Instruction	
	Usage: P	
	Syntax: LB <label>	Related: BR, CL, EX, TI, TP, L, CP

Description:

The LB, or Label Instruction, allows the user to assign a 2 character name to a program, (BR) branch process or (CL) call subroutine. There is a limit of 192 labels.

The restrictions for this command are:

1] A label cannot be named after an MCode Instruction, Variable or Flag or Keyword.
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
5] Labels ARE NOT case sensitive.

Usage Example

```
PG 100   'Start Program at address 100
LB G1    'Name program G1

PG 1
LB SU    'Label Program to execute on power up.
```

MNEMONIC	Function:Lock User Program	Units: —
LK	Type: Setup Flag	Range: 0/1
	Usage: I, R/W	Default: 0 (Disabled)
	Syntax:LK=<0/1>	Related: CP, L

Description:

This flag allows the user to lock the program from being listed or modified. It can only be reset by clearing the entire program space: CP (no address). If CP (address,/label), L (address/label) or PG (address/label) are entered, then error 44 (Program Locked) will be set and nothing else will happen.

To clear LK, don't save (S) then do a Ctrl-C or Cycle Power and the LK will be reset to previous unlocked state. (Program is automatically stored in NVM as it is entered.) Or you may clear program (CP). This will clear the program and reset LK to 0.

Usage Example

```
LK=1   'Lock programs from being listed or changed
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background:
Applies to Plus^2
Models Only

MNEMONIC **LM**	Function: Limit Stop Mode	Units: Modes
	Type: I/O Variable	Range: 1-6
	Usage: P/I, R/W	Default: —
	Syntax: LM=<1-6>	Related: H,I, HM, JE, MA, MR, SL

See Description and Examples on Following Page

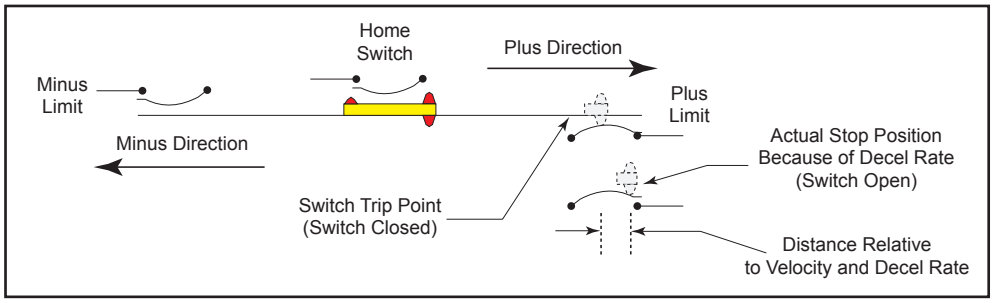


Figure 3.2: Limit Mode Operation

Description:

The LM variable specifies the Limit Stop Mode for the MCode compatible device. There are six LM modes. They are as follows.

LM=1: Normal Limit function with a decel ramp.

The I/O must be set for Limits (S1-S4, S9-S12 Command). If the limit switch in the direction of travel is reached, the motion will decel to a stop. That is, the plus limit works only in the plus direction of travel and the minus limit works only in the minus direction of travel.

In the illustration above, the Limit is activated at a given position but because of the deceleration rate the motion continues for the duration of the deceleration time. This position may be beyond the trip point of the limit and a subsequent move in the same direction will not stop. A crash may be imminent. If the limit is activated and maintained the software will allow motion *only* in the opposite direction. If Homing (HM) is active and a limit is reached, the motion will decel to a stop and then reverse direction and seek the Homing Switch. If the Homing Switch is not activated on the reverse and the opposite limit is reached all motion will stop with a decel ramp. (See HM)

LM=2: A Limit stops all motion with a deceleration ramp but no Homing.

LM=3: A Limit will stop all motion with a deceleration ramp and stop program execution.

LM=4: Functions as LM=1 but with no deceleration ramp.

LM=5: Functions as LM=2 but with no deceleration ramp.

LM=6: Functions as LM=3 but with no deceleration ramp.

Note the MT, Motor Settling Delay Time will be applied upon LM.

Usage Example

```
LM=2    'Set Limit stop with a decel ramp, no homing.
```

MNEMONIC **LR**	For Internal IMS Use Only: Will return an error if used. Do Not use for a program label or user variable or flag.

MNEMONIC	Function: Move to Absolute Position	
MA	Type: Motion Instruction	
	Usage: P/I	
	Syntax: MA <±position>,<parameter 0/1>, <parameter 0/1>	Related: MD, MR, MS, P, SL

Description:

Set mode for absolute move and move to an absolute position relative to (0) zero. MD (Motion Mode) will be set to MA. The time required to calculate the move is 2.5 mSec.

If parameter is true, the DN character will be sent out the serial port when move is complete.

The Third parameter, if true will cause the motor to continue moving after the commanded position is reached.

MA command will not operate during a homing sequence.

Usage Example

```
MA 200000     'Move to absolute position 200000
MA 100000,1   'Move to absolute position 100000, send out Device Name when
              'complete
MA 512000,0,1 'Move to absolute position 512000, continue motion after
               'position is reached.
```

MNEMONIC	Function: Motion Mode	Units: Motor Steps/Encoder Counts
MD	Type: Motion Variable	Range: —
	Usage: P/I, R	Default: —
	Syntax: PR MD	Related: MA, MR, MS, P, PR, SL

Description:

Indicates what the last motion command was. When just a number is entered, then it will execute the move type according to the previous move type entered. This allows the user to apply numeric data to the last motion command without having to enter the command itself.

Note that if the IF flag is pending, numeric entry will be applied to the IV (Input Variable).

Usage Example

```
PR MD    'Return the last motion command used to the terminal screen
          'Response will be the last motion command ie. MR

MR 10000  'Move Relative 10000 steps
5000      'Motor will move relative 5000 steps
```

MNEMONIC	Function: Moving to Position	Units: —
MP	Type: Read Only Status Flag	Range: 0/1
	Usage: P/I, R	Default: 0 (Not Moving)
	Syntax: PR MP	Related: S1-S4, S9-S12

Description:

This flag will=1 when the axis is moving to a position following a MA or MR instruction until MT expires..

Usage Example

```
PR MP  'read the state of the MP flag to terminal 1=moving, '0= not
moving to position
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background: Applies to Plus^2 Models Only

MNEMONIC **MR**	Function: Move to Relative Position	
	Type: Motion Instruction	
	Usage: P/I	
	Syntax: MR <±distance>,<parameter 0/1>	Related: MD, MA, MS, P, SL

Description:

Set mode for relative move and move a relative distance. MD (Motion Mode) will be set to MR. The time required to calculate the move is 2.5 mSec.

If parameter is true, then DN will be sent out when move is complete. The Third parameter, if true will cause the motor to continue moving after the commanded position is reached.

MR will have no effect during a homing sequence.

Usage Example

```
MR 200000    'Move 200000 motor steps positive direction
MR -500000,1  'Move 500000 motor steps negative direction
              'Send Device Name when motion complete.
MR 512000,0,1 'Move to relative position 512000, continue motion after
              'position is reached.
```

MNEMONIC **MS**	Function: Microstep Resolution	Units: Microsteps/Step
	Type: Motion Variable	Range: 1 - 256
	Usage: P/I, R/W	Default: 256
	Syntax: MS=<parameter>	Related: MA, MR, MS, P, PR, SL, C1

Description:

The MS variable controls the microstep resolution of the MCode compatible device. There are 20 different microstep resolutions that can be used with the device. The table below illustrates the parameter settings and their associated resolutions for the 1.8° stepping motor used with the MCode compatible device.

The MS parameters given in the table below are the only valid parameters that will be accepted by the device.

NOTE: If the Encoder is enabled (EE=1) the lowest Microstep Resolution that can be used is 25 (5000 steps/rev) with a 512 line encoder. If using an encoder, IMS recommends leaving MS at the default MS=256.

Usage Example

```
MS=50    'set µStep resolution to 50 µSteps/Step (10000 Steps/Rev)
```

Microstep Resolution Settings			
Binary µStep Resolution Settings		Decimal µStep Resolution Settings	
MS=<µSteps/Full Step>	µSteps/Revolution	MS=<µSteps/Full Step>	µSteps/Revolution
1	200	5	1000
2	400	10	2000
4	800	25	5000
8	1600	50	10000
16	3200	100	20000
32	6400	125	25000
64	12800	200	40000
128	25600	250	50000
256	51200		
Additional Resolution Settings			
180	36000 (0.01°/µStep)		
108	21600 (1 Arc Minute/µStep)		
127	25400 (0.001mm/µStep)		

Table 3.3: Microstep Resolution Settings

MNEMONIC **MT**	Function: Motor Settling Delay Time	Units: Milliseconds
	Type: Motion Variable	Range: 0 to 65000
	Usage: P/I, R/W	Default: 0
	Syntax: MT=<time>	Related: HC, HT, RC

Description:

Specifies the motor settling delay time in milliseconds. MT allows the motor to settle following a move. This is the time between moves if consecutive motions are executed. the MV flag will be active during this time.

Note: MT is added into HT (Hold Current Delay Time). The total of the two cannot exceed 65535. Thus the maximum setting for MT=(65535-HT).

MT should be at least 50 mS when Encoder function is enabled (EE=1)

Usage Example

```
MT=50    'Set motor settling delay time to 50 milliseconds
```

MNEMONIC **MV**	Function: Moving	Units: —
	Type: Read Only Status Flag	Range: 0/1
	Usage: P/I, R	Default: 0 (Not Moving)
	Syntax: PR MV	Related: VC, S1-S4, S9-S12

Description:

Moving flag will be in a logic 1 state when a motion is occurring. This flag will set an output ACTIVE if S<1-4, 9-12>=17

Usage Example

```
PR MV   'read the state of the moving flag to terminal, 1=moving,
        '0= stopped
```

MNEMONIC **NE**	Function: Numeric Enable	Units: —
	Type: R/W Status Flag	Range: 0/1
	Usage: P/I, R	Default: 1 (Enabled)
	Syntax: NE=<1/0>	Related: MA, MR, SL

Description:

Numeirc enable flag will enable (default) or disable the ability of the device to execute the last motion command upon the entry of a numeric value without the preceding command being entered. By default, if a motion command is executed i.e. MR 20000, when the user enters -20000, the device will move relative -20000.

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

Usage Example

```
NE=0   'disable motion mode on numeric entry
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background: Applies to Plus^2 Models Only

MNEMONIC	Function: Set Output Logic State	Units: Logic State
O1 - O4	Type: I/O Variable	Range: 0/1
	Usage: P/I, W	Default: —
	Syntax:O<1-4>=<0/1>	Related: OL, OH, OT, S1-S4

Description:

This variable will set the logic state of the specified output to 1 or 0.

The voltage level will be dependant on the active (low/high) state of the output, specified by the S<1-4> variable.

Usage Example

```
O2=1     'Set output 2 ACTIVE
```

MNEMONIC	Function: Set Output Logic State	Units: Logic State
O9 - O12	Type: I/O Variable	Range: 0/1
	Usage: P/I, W	Default: —
	Syntax:O<9-12>=<0/1>	Related: OL, OH, OT, S9-S12

Description:

This variable will set the logic state of the specified output to 1 or 0.

The voltage level will be dependant on the active (low/high) state of the output, specified by the S<1-4,9-12> variable.

Usage Example

```
O10=1     'Set output 10 ACTIVE
```

MNEMONIC	Function: On Error Handler	
OE	Type: Program Instruction	
	Usage: P	
	Syntax: OE <address/label>	Related: EF, ER

Description:

When an error occurs, the specified subroutine is called. If a program was running when the fault occurs, once the error routine completes, program execution continues with the instruction after the one that caused the error. After OE, the command is executed. A program need not be running for the subroutine specified by OE to run.

The ON ERROR function is disabled by setting the address parameter to 0 or resetting the device with an FD or CP OE Subroutine MUST have an RT at the end.

OE will not execute during programming.

Usage Example

```
OE K1     'run subroutine K1 on an error
```

MNEMONIC **OL**	Function: Set Outputs 1-4 As One Value	Units: 4 Bit Binary Number
	Type: I/O Variable	Range: 0-15
	Usage: P/I, W	Default: —
	Syntax: OL=<0-15>	Related: IH, IN, IL, OH, OT, S1-S4

Description:

The OL variable allows the user to set Outputs 1-4 as one 4 bit binary value. The value is entered in decimal, with a range of 0-15 in binary where Output 1 will be the LSb and Output 4 will be the MSb.

Usage Example

```
OL=13  'set output group 1-4 to 1101
```

MNEMONIC PLUS^2 ONLY **OH**	Function: Set Outputs 9-12 As One Value	Units: 4 Bit Binary Number
	Type: I/O Variable	Range: 0-15
	Usage: P/I, W	Default: —
	Syntax: OH=<0-15>	Related: IH, IN, OL, OH, OT, S9-S12

Description:

The OH variable allows the user to set Outputs 9-12 as one 4 bit binary value. The value is entered in decimal, with a range of 0-15 in binary where Output 9 will be the LSb and Output 12 will be the MSb.

Usage Example

```
OH=13  'set output group 9-12 to 1101
```

MNEMONIC **OT**	Function: Set All Outputs As One Value	Units: 8 Bit Binary Number (4 bit on Plus devices)
	Type: I/O Variable	Range: 0-255
	Usage: P/I, W	Default: —
	Syntax: OT=<0-255>	Related: IL, IH, IN, OL, OH, S1-S4, S9-S12

Description:

The OT variable allows the user to set Outputs 1-4 and 9-12 as one 8 bit binary value. The value is entered in decimal, with a range of 0-255 in binary where Output 1 will be the LSb and Output 12 will be the MSb.

NOTE: On Standard Plus Models OT will only set the lower group (Outputs 1-4)

Usage Example

```
OT=214  'set the standard output group to 11010110
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background: Applies to Plus^2 Models Only

MNEMONIC **P**	Function: Position Counter	Units: Motor Steps (EE=0)/Encoder Counts (EE=1)
	Type: Motion Variable	Range: -2147483648 to 2147483647
	Usage: P/I, R/W	Default: 0
	Syntax: P=<±position>, PR P	Related: C1, C2

Description:

This instruction is used to set or print the value of the MCode compatible device position counter. The position will read in Motor Steps from C1 (Counter 1) by default, if encoder functions are enabled, the position counter will read in Encoder Counts from C2 (Counter 2).

Modifying P in essence changes the frame of reference for the axis for Move Absolute (MA) instructions. P will probably be set once during system set up to reference or "home" the system.

Usage Example

```
P=0    'Set the position counter to zero
PR P   'Read the position counter to the terminal window
```

MNEMONIC PLUS^2 ONLY **PC**	Function: Position Capture At Trip	
	Type: Instruction	
	Usage: I	
	Syntax: PR PC	Related: TE, TI, TC, TT, S13

Description:

Captures motor or encoder position during a trip event. Activation will occur upon any trip function EXCEPT a position trip (TP). Will display in either motor steps (EE=0) or encoder counts (EE=1)

Usage Example

```
PR PC     'Display captured position
```

MNEMONIC **PG**	Function: Enter/Exit Program Mode	
	Type: Instruction	
	Usage: P/I	
	Syntax: PG <address>	Related: E

Description:

When starting program mode, you must specify the starting address to begin entering the program. Simply type "PG" again when you have finished entering your program commands to go back to immediate mode. The device will determine the addresses following the one specified by the starting PG.

While in program mode, leading tabs, spaces and blank lines are ignored. This allows the user to format a text file for readability, and then download the program to the device by transferring the text file in a program such as IMS Terminal or Hyper terminal. See Appendix B of this document for more information on the IMS Terminal software. The example given below could be stored in a text file and downloaded. The lines preceded by an apostrophe (') are comments and will be ignored by the MCode compatible device.

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

Note that the code between Pgs is automatically saved to NVM after download.

Usage Example

```
PG     'Enter program mode
'****PROGRAM****
E          'End Program
PG         'Exit program mode
```

MNEMONIC	Function: Position Maintenance Enable	Units: —
PM	Type: Setup Flag	Range: 0/1
	Usage: P/I, R/W	Default: 0 (Disabled)
	Syntax:PM=<0/1>	Related: VI, EE, SM, DB, C2, SF

Description:

This flag will enable the position maintenance functions of an MCode compatible device with encoder. The position maintenance velocity will be at the setting for VI (Initial Velocity). If moved beyond the value of DB (DeadBand), unit will correct

If SM = 0 and PM = 1, Position Maintenance will take place provided the position does not exceed the Stall Factor (SF).

If SM = 1 and PM = 1, Position Maintenance will take place even if the Stall Factor (SF) is exceeded, unless VI is set too high causing the motor to stall.

Usage Example

```
PM=1     'Enable position maintenance
```

MNEMONIC	Function: Part Number	Units: IMS Part Number
PN	Type: Variable	Range: —
	Usage: R	Default: —
	Syntax: PR PN	Related: —

Description:

Reads Products Part Number

Usage Example

```
PR PN    'read theMCode part number
```

MNEMONIC	Function: Print Selected Data/Text	
PR	Type: Instruction	
	Usage: P/I	
	Syntax: PR <"text">, <data>	Related: —

Description:

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

It is important to note that the receive buffer for the MCode device is 64 characters, this includes the PR instruction itself, any spaces, text characters, etc. If the buffer length is exceeded a CR/LF will ocur and set error 20.

Note: A delay time between the print requests to the device must be considered to allow the device time to interpret a command and answer the host before a subsequent command can be sent.

Usage Example

```
PR  "Position =", P   'print axis position
PR MS                 'Print the µStep Resolution setting
```

MNEMONIC	Function: Pause Program	
PS	Type: Program Instruction	
	Usage: I	
	Syntax: PS	Related: RS, SL, MA, MR, HI, HM

Description:

This instruction is used to pause an executing program and invoke normal deceleration of any motion being executed to Zero. Immediate mode instructions are allowed while a program is in a paused state. To resume the program the RS instruction is used.

Usage Example

```
PS      'Pause running program
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background: Applies to Plus^2 Models Only

MNEMONIC **PW**	Function: PWM Configuration	Units: vary
	Type: Variable	Range: See tables
	Usage: I, R/W	Default: 204,95,170
	Syntax: PW=<mask>,<period>,<sfreq>, <chksum> PR PW (Response = <mask>,<period>,<sfreq>,<boot_writes_remaining>	

Note: The PW variable is only used on the MForce product line. It is not a reserved word on the MDrive product line and may be used as a User Variable or Label.

This variable is used to set the PWM Current Control settings of the MForce ONLY! It does not apply in any function to the MDrive series and may be used as a label or user variable or flag.

See Appendix G of this document for parameter settings and usage. It is recommended that these settings not be modified from the factory default unless the motor exhibits rough motion, audible noise or poor zero-crossing performance. If these symptoms exist, please check for other issues such as electrical noise coupled onto logic signals or ground loops before modifying these settings.

⚠ CAUTION

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

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

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

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

Note: the Party Mode Enable flag setting must be saved (S) prior to any reset or the setting will be lost.

MNEMONIC **PY**	Function: Party Mode Enable	Units: —
	Type: Setup Flag	Range: 0/1
	Usage: I, R/W	Default: 0 (Disabled)
	Syntax:PY=<0/1>	Related: DN, DG

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 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.

NOTE: Party Mode Configuration and Details are covered at length in of the Hardware Manual specific to the device you purchased.

Usage Example

```
PY=1     'Enable Party Mode Communications
```

MNEMONIC **QD**	Function: Queued	Units: —
	Type: Setup Flag	Range: 0/1
	Usage: I, R/W	Default: 0 (Disabled)
	Syntax:QD=<0/1>	Related: PY, DN

Description:

Function is to queue drives on party lines. Works similar to uLynx QUED flag.

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.

Usage Example

```
QD=1     'Queue Drive
```

MNEMONIC **R1 - R4**	Function: User Registers	Units: Numeric Value
	Type: I/O Variable	Range: -2147483647 to 2147483647
	Usage: P/I, R/W	Default: —
	Syntax: R<1-4>=<number>	Related: —

Description:

The MCode compatible device has four 32 bit user registers to contain numerical data. These registers may contain up to 11 digits including the sign and may be used to store and retrieve data to set variables, perform math functions, store and retrieve moves and set conditions for branches and call subroutine.

Usage Example

```
R1=50000    'Set Register 1 to 50000
R2=Q2       'Set Register 2 to the value of User Variable Q2
```

MNEMONIC **RC**	Function: Run Current	Units: Percent
	Type: Variable	Range: See table
	Usage: P/I, R/W	Default: 25
	Syntax: RC=<%>	Related: HC

Description:

This variable defines the motor run current in percent.

Usage Example

```
RC=75    'Set motor run current to 75%
```

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

Shaded Area Reserved for Future Use

*RC=67 for maximum 2.0 Amp Run Current

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background: Applies to Plus^2 Models Only

MNEMONIC **RS**	Function: Resume Program	
	Type: Program Instruction	
	Usage: I	
	Syntax: RS	Related: PS, SL, MA, MR, HI, HM

Description:

This instruction is used to resume a program that has been paused using the PS instruction. Any move that was paused will resume as well. Motion will resume using the normal acceleration profiles.

Usage Example

```
RS      'Resume paused program
```

MNEMONIC **RT**	Function: Return From Called or On Error Subroutine	
	Type: Program Instruction	
	Usage: P	
	Syntax: RT	Related: CL, OE

Description:

This instruction defines the end of a subroutine. This instruction is required and will be the final instruction in the subroutine executed by the CL or OE instruction. When used, it will return to the program address immediately following the instruction which executed the subroutine.

Usage Example

```
CL K8   'Call Subroutine K8
LB K8
*****SUBROUTINE K8*****
RT      'Go back to main program
```

MNEMONIC **S**	Function: Save to NVM	
	Type: Instruction	
	Usage: P/I	
	Syntax: S	Related: —

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: Communications 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.

Usage Example

```
S     'Save all variable and flag states to NVM
```

MNEMONIC	Function: Setup I/O Points 1 - 4	
S1-S4	Type: I/O Instruction	
	Usage: P/I, R/W	
	Syntax: S<1-4>=<type>,<active>,<sink/source>	Related: I1-4, IN, O1-4, OT, D1-D4

Description:

This instruction is used to setup the I/O type and active states, and sink/source setting for I/O points 1 - 4. Each of the device I/O points 1- 4 may be programmed as either general purpose inputs and outputs, or to one of nine dedicated input functions or one of two dedicated output functions.

When programmed as inputs, these points can be sinking or sourcing, and may be programmed such that they are active when pulled to ground, or active when left floating. By default each point is configured as a general purpose input, active when LOW.

There are three parameters attached to this instruction:

1) The type specifies the function of the I/O point (see tables - following page).
2) The second parameter sets the active state, which defines the point as (0 - Default) LOW or (1) HIGH ACTIVE.
3) The third parameter specifies whether the point will be (0 - Default) sinking or sourcing (1).

Please see the tables on the following page for a definition of all of the I/O type parameters.

Usage Example

```
S1=1,0,0     'Set IO1 to homing input, active when LOW, sinking
S2=4,1,1     'Set IO2 to be a G0 input active when HIGH, sourcing
S3=17,1,0    'Set IO3 to be a moving output, active when HIGH, Sinking
```

NOTE: Output Types are SINKING ONLY on Plus devices.

NOTE: Once set wait debounce time before using.

Refer to the Hardware Reference Section dealing with the I/O for the device purchased for more examples.

Input Functions

Function	Description	Parameter (S1-S4, S9-S12)	Active	Sink/ Source
General Purpose	General Purpose Input function used to control program branches, subroutine calls or BCD functions when input bank is used as a group	0	0/1	0/1
Home	Homing input. Will function as specified by the Home (HM) command.	1	0/1	0/1
Limit +	Positive Limit Input. Will function as specified by the Limit (LM) Command.	2	0/1	0/1
Limit –	Negative Limit Input. Will function as specified by the Limit (LM) Command.	3	0/1	0/1
G0	G0 Input. Will run program located at address 1 on activation.	4	0/1	0/1
Soft Stop	Soft Stop input. Stops motion with deceleration and stops program execution. Note that Soft Stop is Ignored if Pause is enabled.	5	0/1	0/1
Pause	Pause/Resume program with motion.	6	0/1	0/1
Jog +	Will Jog motor in the positive direction at Max. Velocity (VM). The Jog Enable (JE) Flag must be set for this to function.	7	0/1	0/1
Jog –	Will Jog motor in the negative direction at Max. Velocity (VM). The Jog Enable (JE) Flag must be set for this to function.	8	0/1	0/1
Reset	When set as RESET input, then the action is equivalent to a ^C entered into a terminal. Note: If setting the input to sourcing, active true, ground the input first or a reset will occur.	11	0/1	0/1

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background: Applies to Plus^2 Models Only

Output Functions				
Function	**Description**	**Parameter (S1-S4, S9-S12)**	**Active**	**Sink/ Source**
General Purpose User	A general purpose output can be set in a program or in immediate mode to trigger external events. When used as a group they can be a BCD output.	16	0/1	0/1
Moving	Will be in the Active State when the motor is moving.	17	0/1	0/1
Fault	Will be in the Active State when a error occurs. See Software Manual for error code listing.	18	0/1	0/1
Stall	Will be in the Active State when a stall is detected. Encoder Required, Stall Detect Mode (SM) must be enabled.	19	0/1	0/1
Velocity Changing	Will be in the Active State when the velocity is changing. Example: during acceleration and deceleration.	20	0/1	0/1
Moving To Position	Will be in an active state while moving to an absolute position.	23	0/1	0/1

Table 3.4: I/O Types and Settings

Output Circuit Conditions

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

Table 3.5: Output Circuit Conditions

MNEMONIC	Function: Setup I/O Point 5 (Analog Input)	
S5	Type: Instruction	
	Usage: P/I, R/W	
	Syntax: S5=<type>,<0/1>	Related: I5, JE

Description:

This I/O point configures the analog input reference as either a current or voltage source. The value of this input will be read using the I5 instruction, which has a range of 0 to 1023, where 0 = 0 volts and 1023 = 5.0 volts. The device may also be configured for a 4 to 20 mA or 0 to 20 mA Analog Input (S5 = 10).

The second parameter specifies the voltage/current range:

0 (default) = 0-5 VDC/0-20 mA

1= 0-10 VDC/4-20 mA

Ranges if setting S5 as 0 - 20 mA: Resolution 0 - 1023

Ranges if Setting S5 as 4 - 20 mA: Resolution 0 - 800

Usage Example

```
S5=9,0     'Analog Input set to voltage reference, 0-5 VDC (Default)
S5=9,1     'Analog Input set to voltage reference, 0-10 VDC
S5=10,0    'Analog Input set to current reference, 0-20 mA
S5=10,1    'Analog Input set to current reference, 4-20 mA
```

MNEMONIC PLUS^2 ONLY **S7-S8**	Function: Setup I/O Points 7 & 8	
	Type: I/O Instruction	
	Usage: P/I, R/W	
	Syntax: S<7/8>=<type>,<active>	Related: FM, CW, CM

Description:

Sets up I/O 7 and I/O 8 clock type. Can be set as inputs or outputs, I/O 7 and I/O 8 are setup in pairs. The clock types are step clock/direction, up/down, and quadrature.

Clock Input Functions

Function	Description	Parameter (S7, S8)	Active
Step/Direction	Sets I/O 7 and 8 to receive step and direction inputs from an external source. The motion will occur based on the input frequency seen at I/O 7 in the Direction relative to the logic state of I/O 8. The step rate will be based upon the ratio set by Clock Ratio (CR)	33	0/1
Quadrature	Sets I/O 7 and 8 to receive Channel A and Channel B Quadrature inputs from an external source such as an encoder. The motion will follow the Quadrature Input. The clock rate will be based upon the ratio set by Clock Ratio (CR)	34	0/1
Up/Down	Sets I/O 7 and 8 to receive Clock Up/Clock Down inputs from an external source. The motion will occur based upon the input clock frequency in the direction relative to the input being clocked. The step rate will be based upon the ratio set by Clock Ratio (CR)	35	0/1

Clock Output Functions

Function	Description	Parameter (S7, S8)	Active
Step/Direction	Step clock pulses will be output from Point 7, Direction from Point 8. The step clock output rate will be based upon the Pulse Width set by Clock Width (CW). The logic state of the Direction output will be with respect to the direction of the motor.	49	0/1
Quadrature	Will output Quadrature signals.	50	0/1
Up/Down	Will output Clock Up/Clock Down signals. The step clock output rate will be based upon the Pulse Width set by Clock Width (CW). The Active output will be based on the motor direction.	51	0/1

Table 3.6: Clock Output Functions

Usage Example

```
S7=33,0   'Set IO7 to be a step/direction input, active LOW
S8=33,0   'Set IO8 to be a step/direction input, active LOW

S7=50,1   'Set IO7 to be a quadrature output, active HIGH
S8=50,1   'Set IO8 to be a quadrature output, active HIGH
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background: Applies to Plus^2 Models Only

MNEMONIC PLUS^2 ONLY

S9-S12

Function: Setup I/O Points 9 - 12

Type: I/O Instruction

Usage: P/I, R/W

Syntax: S<9-12>=<type>,<active>,<sink/source>

Related: I9-12, IN, O9-12, OT, D1-D4, D9-D12

Description:

This instruction is used to setup the I/O type, active states and sink/source configuration for I/O points 9 - 12. Each of device I/O points 9 - 12 may be programmed as either general purpose inputs and outputs, or to one of nine dedicated input functions or one of two dedicated output functions.

When programmed as inputs, these points can be sinking or sourcing, and may be programmed such that they are active when pulled to ground, or active when left floating. By default each point is configured as a general purpose input, active when LOW.

There are three parameters attached to this instruction:

1) The type specifies the function of the I/O point (see tables - previous page).
2) The second parameter sets the active state, which defines the point as (0 - Default) LOW or (1) HIGH ACTIVE.
3) The third parameter specifies whether the point will be (0 - Default) sinking or sourcing (1).

Please Reference S1 - S4 Command for a definition of all of the I/O type parameters and output circuit conditions.

Usage Example

```
S11=1,0,0     'Set IO1 to homing input, active when LOW, sinking
S12=4,1,1     'Set IO12 to be a G0 input active when HIGH, sourcing
S9=17,1,0    'Set IO9 to be a moving output, active when HIGH, Sinking
```

MNEMONIC PLUS^2 ONLY

S13

Function: Setup I/O Point 13 (Capture/Trip)

Type: I/O Instruction

Usage: P/I, R/W

Syntax: S13=<type>,<0/1>

Related: FC, CW, TP, PC

Description:

Sets up high speed position capture input and postion trip output.

Note that this I/O Point is for position capture and trip ONLY. It is not effected by any other trips.

Function	Description	Parameter (S13)	Active
High Speed Capture Input	The Capture input is a momentary high speed input that operates with the Trip Capture (TC) variable to run a subroutine upon the trip. It feature variable input filtering ranging from 50 nS to 12.9 µS	60	0/1

Function	Description	Parameter (S13)	Active
High Speed Trip Output	The trip output will activate on Position Trips (TP) only. The output will pulse out at the trip point. The pulse width will be determined by Clock Width (CW)	61	0/1

Table 3.7: High Speed I/O Types

Usage Example

```
S13=60,0    'Set up IO13 as High Speed Capture Input, active LOW(default)
```

MNEMONIC	Function: Stall Factor	Units: Encoder Counts
SF	Type: Variable	Range: 0 to 65000
	Usage: P/I, R/W	Default: 15
	Syntax: SF=<counts>	Related: EE, SM, ST, PM

Description:

If the encoder is enabled (EE = 1) and the encoder differs from the commanded position by more than the specified factor, a STALL is indicated. If SM is set to 0, then the motor will be stopped when a STALL is detected. If SM=1, the motor will not be stopped upon detection of a stall. ST will return an ER=86 on stall.

Usage Example

```
SF=20    'Set the stall factor for 20 counts
```

MNEMONIC	Function: Slew Axis	Units: Motor Steps (EE=0)/Encoder Counts (EE=1)
SL	Type: Motion Instruction	Range: ±5000000 (EE=0)/±200000 (EE=1)
	Usage: P/I	
	Syntax: SL <velocity>	Related: MA, MR, VI

Description:

The SL instruction will slew the axis at the specified velocity in steps per second. The axis will accelerate at the rate specified by the A (Acceleration) variable.

Note that the maximum slew velocity is independent of the maximum velocity specified by the VM variable. If a slew is commanded at a velocity greater than the setting of VM, the axis will accelerate to the SL velocity regardless of the setting of VM.

App. Note: 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.

Usage Example

```
SL 20000    'Slew at a rate of 20000 steps/sec
```

MNEMONIC	Function: Stall Detection Mode	Units: 0/1
SM	Type: Variable	Range: 0/1
	Usage: P/I, R/W	Default: 0
	Syntax: SM=<1/0>	Related: EE, SM, ST, PM, SF

Description:

The SM variable specifies the action which will be taken by the device when a stall is detected. When set to 0 (default) the motion will be stopped upon a stall detection. When SM=1, the motor will try to continue the move. In either case ST (Stall Flag) will be set.

The functionality of SM when used with Position Maintenance (PM) is listed below:

If SM = 0 and PM = 1, Position Maintenance will take place provided the position does not exceed the Stall Factor (SF).

If SM = 1 and PM = 1, Position Maintenance will take place even if the Stall Factor (SF) is exceeded, unless VI is set too high causing the motor to stall.

Usage Example

```
SM=1    'Change Stall mode that the motor doesn't stop on stall detect
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background: Applies to Plus^2 Models Only

MNEMONIC **SN**	Function: Serial Number	Units: IMS Serial Number
	Type: Variable	Range: —
	Usage: R	Default: —
	Syntax: PR SN	Related: —

Description:

Reads Products Serial Number

Usage Example

```
PR SN    'read the device serial number
```

MNEMONIC **ST**	Function: Stall Flag	Units: —
	Type: Encoder Flag	Range: 0/1
	Usage: P/I, R/W	Default: 0 (Not Active)
	Syntax: PR ST BR <addr>, ST=1 CL <addr>, ST=1	Related: EE, SF, OE

Description:

The ST flag will be set to 1 when a stall is detected. It is the responsibility of the user to reset it to zero (0). Encoder function must be enabled (EE=1) in order for a stall flag to set.

Usage Example

```
CL K5,ST=1     'Call subroutine K5 if motor stalls
ST=0           'Clear stall flag
```

LABEL **SU**	Function: Startup Label	Units: —
	Type: Predefined Label	
	Usage: P	
	Syntax: LB SU	Related: LB

Description:

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

Usage Example

```
PG 1      'Start Program at address 1
  LB SU   'Label program so that it executes on power up.
```

MNEMONIC PLUS^2 ONLY **TC**	Function: Trip Capture	Units: Program Address/Label
	Type: Variable	Range: —
	Usage: P/I, R/W	Default: —
	Syntax: TC=<address/label>	Related: TE, S13

Description:

Sets the Capture input trip for I/O 13. Sets one parameter for trip address. The TE command (Trip Enable/Disable TC) is reset when trip occurs. TE must be re-enabled in the main program prior to the next trip if it is to be repeated. The Trip subroutine must use a RETURN (RET) to exit the subroutine, use of a BRANCH will cause stack errors.

Usage Example

```
TC=K1    'Run subroutine K1 on Input Trip
TE=4     'Re-enable Trip
```

MNEMONIC **TE**	Function: Trip Enable	Units: —
	Type: Setup Flag	Range: 0-16
	Usage: P/I, R/W	Default: 0 (Disabled)
	Syntax: TE=<0-4,8>	Related: I1-I4, I9-I12, P, S1-S4, S9-S12, TI, TP, TC, TT

Description:

This flag will enable or disable specified trip functions.

TE=0.....Disabled

TE=1.....Trip On Input Enabled

TE=2.....Trip On Position Enabled

TE=4.....Trip On Capture (I/O 13) Enabled

TE=8.....Trip On Time Enabled

TE=16.....Trip On Relative Position

The trip functions may be combined by using binary settings to enable multiple trips where time is the MSb and input is the LSb. For example TE=3 will trip on input or on position, TE=15 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.

Usage Example

```
TE=1    'Enable trip on input function
TE=8    'Enable trip on time function
TE=6    'Trip on Position or Capture input.
```

MNEMONIC **TI**	Function: Trip on Input	Units: —
	Type: Variable	Range: —
	Usage: P/I, R/W	Default: —
	Syntax: TI=<input>,<address/label>	Related: I1-4, S1-4, TE, TP, TT

Description:

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

The TE is reset when a Trip occurs. TE must be re-enabled prior to the next Trip if it is to be repeated.

Usage Example

```
TI=2,K3   'execute subroutine K3 when input 2 active
TE=1      'Re-enable Trip
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding


Variable


Flag


Instruction


Body Background:
Applies to Plus^2
Models Only

MNEMONIC **TP**	Function: Trip on Position	Units: —
	Type: Variable	Range: —
	Usage: P/I, R/W	Default: —
	Syntax: TP=<position>,<address/label>	Related: P, TI, PC, S13, CW

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 (RET) to exit the subroutine, use of a BRANCH will cause stack errors

Trips should be set BEFORE motion commands in the program.

Note that TP will always use motor counts regardless of the encoder enabled state (EE=1)

Usage Example

```
TP=650000,K9   'execute subroutine K9 when motor position 650000
TE=2           'Re-enable trip
```

MNEMONIC **TR**	Function: Trip on Relative Position	
	Type: Variable	Range: —
	Usage: P/I, R/W	Default: —
	Syntax: TR=<±dist>,<address/label>, <repeat>	Related: P, TI, PC, S13, CW

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. The third sets the number of times the trip will repeat. This is optional, if no subroutine address or label is specified then the High Speed Trip Output will activate. The Trip subroutine must use a RETURN (RET) to exit the subroutine, use of a BRANCH will cause stack errors

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 when follwing the repeat of trips. TE must be re-enabled in the main program prior to the next series of Trip on Relative if it is to be repeated. For exampl, if TR=10000,0,25, the Output (S13) will trip 25 times in succession at 100,000 counts relative to the last position. Following these 25 trips the trip must be re-enabled (TE=16).

Trips should be set BEFORE motion commands in the program.

Note: Output S13 must be configured as a trip output (S13=61,1/0)

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.

Usage Example

```
TR=650000,0,0  'Activate the trip output when the motor is at 650000
               'counts relative
TE=16          'Re-enable trip

TR=512000,K2,27  'Run Subroutine K2 when at 512000 rel, repeat 27 times
TE=16            'Re-enable trip
```

MNEMONIC	Function: Trip on Time	Units: Milliseconds
TT	Type: Variable	Range: 1 to 65535
	Usage: P/I, R/W	Default: —
	Syntax: TT=<time>,<address/label>	Related: TE, TP, TI

Description:

Sets up a trip based on time. The first parameter is time in mSec. The second parameter specifies the subroutine that should be executed when the time is expired. The Trip subroutine must use a RETURN (RET) to exit the subroutine, use of a BRANCH will cause stack errors

TE must be re-enabled in the main program prior to the next Trip if it is to be repeated.

Usage Example

```
TT=2000,K3  'Trip on time 2000 mS, execute subroutine K3
TE=8        'Re-enable trip
```

MNEMONIC	Function: Upgrade Firmware	
UG	Type: Instruction	
	Usage: I	
	Syntax: UG 2956102	Related: —

Description:

Upgrade Firmware Instruction. Upgrade code is 2956102. This will put the device in Upgrade Mode. Once set, the firmware Upgrade MUST be completed.

Usage Example

```
UG 2956102
```

MNEMONIC	Function: Read User Variables	Units: —
UV	Type: Variable	Range: —
	Usage: P/I, R	Default: —
	Syntax: PR UV	Related: PR, VA

Description:

Read User Variables is used with the PR (Print) Instruction to read the value of all user variables.

Usage Example

```
PR UV    'Read the value of all user variables
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background: Applies to Plus^2 Models Only

MNEMONIC **V**	Function: Read Only Velocity Variable	Units: Motor Steps (EE=0)/Encoder Counts (EE=1)
	Type: Motion Variable	Range: 0-5M/0-200k
	Usage: P/I, R	Default: —
	Syntax: PR V BR <address/label>, V=<velocity> BR <address/label>, V=<velocity>	Related: VI, VM, SL, MA, MR

Description:

The velocity variable is used in conjunction with the PR (print) instruction to read the current velocity of the axis in counts per second. This variable can also be used with the BR and CL instructions to set a condition based upon a velocity.

Note that V is signed.

Usage Example

```
PR V             'Read the velocity
CL Ka, V=20000   'Execute sub Ka when velocity is 20000/steps sec in the
                 ' positive direction
CL Kb, V=-20000  'Execute sub Kb when velocity is -20000/steps sec in the
                 'negative direction
```

MNEMONIC **VA**	Function: Create User Variable Name	
	Type: Instruction	Range: Signed 32 Bit Integer
	Usage: P/I, R/W	
	Syntax: VA <char><char>=<value>	Related: UV

Description:

The VA instruction allows the user to assign a 2 character name to a user defined variable.

The restrictions for this command are:

1] A variable cannot be named after a 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.

Note: Local variables can be re-declared in program. Labels and global variables can not be re-declared. This change will be ignored resulting in error 28.

Usage Example

```
VA Q3=20000   'Create user Variable Q3, set value to 20000
```

MNEMONIC **VC**	Function: Velocity Changing Flag	Units: —
	Type: Motion Flag	Range: 0/1
	Usage: P/I, R	Default: 0 (Not Active)
	Syntax: PR VC BR <addr>, VC=1 CL <addr>, VC=1	Related: MV, VI, VM, S1-S4, S9-S12

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.

States: VC=0: Motor stopped or constant velocity, VC=1: motor velocity is changing

Usage Example

```
CL K5,VC=1     'Call subroutine K5 if velocity is changing
PR VC          'Print the state of the VC Flag
```

MNEMONIC	Function: Initial Velocity	Units: Motor Steps (EE=0)/Encoder Counts (EE=1)
VI	Type: Motion Variable	Range: 5000000 (EE=0)/200000 (EE=1)
	Usage: P/I, R/W	Default:1000/40
	Syntax: VI=<velocity>	Related: VM, MR, MA, HI, HM

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).

Usage Example

```
VI=10000    'Set initial velocity to 10000 steps/sec.
VI=Q1       'Set initial velocity to the value of User Var Q1
```

MNEMONIC	Function: Max Velocity	Units: Motor Steps (EE=0)/Encoder Counts (EE=1)
VM	Type: Motion Variable	Range: 5000000 (EE=0)/200000 (EE=1)
	Usage: P/I, R/W	Default:768000/30720
	Syntax: VM=<velocity>	Related: VI, MR, MA, HI, HM, JE

Description:

The VM variable specifies the maximum velocity in steps/counts per second that the axis will reach during a move command.

VM must be greater than VI.

Usage Example

```
VM=500000   'Set max velocity to 500000 steps/counts sec.
VM=Q1       'Set initial velocity to the value of User Var Q1
```

MNEMONIC	Function: Firmware Version	Units: IMS Version Number
VR	Type: Variable	Range: —
	Usage: R	Default: —
	Syntax: PR VR	Related: UG

Description:

This variable is used in conjunction with the PR instruction to read the version of the firmware installed at the factory.

Usage Example

```
PR VR     'Read the firmware version installed
```

USAGE ABBREVIATIONS

Program = P
For use within a user program

Immediate = I
Not for use within user program

Read = R
Use in print statement

Write = W
Write to a Variable

Color Coding

Variable

Flag

Instruction

Body Background:
Applies to Plus2
Models Only

MNEMONIC		
WT	Function: Setup Variable	Units: Degrees C
	Type: Variable	Range: 0 to 80
	Usage: P/I, R/W	Default: 80
	Syntax: WT=<temp>	Related: IT

Description:

The Warning Temperature variable allows the user to set a threshold temperature at which the device will print an error 71 to the terminal screen if the set temperature is exceeded.

NOTE: This functionality is only standard on 34 (DC and AC) and 42 (AC) Frame Devices.

Usage Example

```
WT=75     'set the warning temperature to 75 deg. C
```

Error Codes

A question mark <?> displayed as a cursor indicates an ERROR. To determine what the ERROR is, type <PR ER> in the IMS 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.

Error Code	Fault
0	No Error
I/O Errors	
6	An I/O is already set to this type. Applies to non-General Purpose I/O.
8	Tried to set an I/O to an incorrect I/O type.
9	Tried to write to I/O set as Input or is “TYPED”.
10	Illegal I/O number.
11	Incorrect CLOCK type.
12	Illegal Trip / Capture
Data Errors	
20	Tried to set unknown variable or flag. Trying to set an undefined variable of flag. Also could be a typo.
21	Tried to set an incorrect value. Many variables have a range such as the Run Current (RC) which is 1 to 100%. As an example, you cannot set the RC to 110%.
22	VI is set greater than or equal to VM. The Initial Velocity is set equal to, or higher than the Maximum Velocity. VI must be less than VM.
23	VM is set less than or equal to VI. The Maximum Velocity is set equal to, or lower than the Initial Velocity. VM must be greater than VI.
24	Illegal data entered. Data has been entered that the device does not understand.
25	Variable or flag is read only. Read only flags and variables cannot be set.
26	Variable or flag is not allowed to be incremented or decremented. IC and DC cannot be used on variables or flags such as Baud and Version.
27	Trip not defined.Trying to enable a trip that has not yet been defined.
28	WARNING! Trying to redefine a program label or variable. This can be caused when you download a program over a program already saved. Before downloading a new or edited program, type <FD> and press ENTER to return the device to the Factory Defaults. You may also type <CP> and press ENTER to Clear the Program.
29	Trying to redefine a built in command, variable or flag.
30	Unknown label or user variable. Trying to Call or Branch to a Label or Variable that has not yet been defined.
31	Program label or user variable table is full. The table has a maximum capacity of 22 labels and/or user variables.
32	Trying to set a label (LB). You cannot name a label and then try to set it to a value. Example: Lable P1 (LB P1). The P1 cannot be used to set a variable such as P1=1000.
33	Trying to SET an Instruction.
34	Trying to Execute a Variable or Flag
35	Trying to Print Illegal Variable or Flag
36	Illegal Motor Count to Encoder Count Ratio
37	Command, Variable or Flag Not Available in Drive
38	Missing parameter separator
39	Trip on Position and Trip on Relative Distance not allowed together

Program Errors	
40	Program not running. If HOLD (H) is entered in Immediate Mode and a program is not running.
41	Not Used.
42	Illegal program address. Tried to Clear, List, Execute, etc. an incorrect Program address.
43	Tried to overflow program stack. Calling a Sub-Routine or Trip Routine with no Return.
44	Program locked. User Programs can be Locked with the <LK> command. Once Locked, the program cannot be listed or edited in any way.
45	Trying to Overflow Program Space.
46	Not in Program Mode.
47	Tried to Write to Illegal Flash Address
48	Program Execution stopped by I/O set as Stop.
Communications Errors	
60	Not used
61	Trying to set illegal BAUD rate. The only Baud Rates accepted are those listed on the Properties Page of IMS Terminal. (4,800, 9,600, 19,200, 38,400, 115,200)
62	IV already pending or IF Flag already TRUE.
System Errors	
70	FLASH Check Sum Fault
71	Internal Temperature Warning, 10C to Shutdown
72	Internal Over TEMP Fault, Disabling Drive
73	Tried to SAVE while moving
74	Tried to Initialize Parameters (IP) or Clear Program (CP) while Moving
75	Linear Overtemperature Error (For units without Internal Over Temp)
Motion Errors	
80	HOME switch not defined. Attempting to do a HOME (H) sequence but the Home Switch has not yet been defined.
81	HOME type not defined. The HOME (HM or HI) Command has been programmed but with no type or an illegal type. (Types = 1, 2, 3, or 4)
82	Went to both LIMITS and did not find home. The motion encroached both limits but did not trip the Home switch. Indicates a possible bad switch or a bad circuit.
83	Reached plus LIMIT switch. The LIMIT switch in the plus direction was tripped.
84	Reached minus LIMIT switch. The LIMIT switch in the minus direction was tripped.
85	MA or MR isn't allowed during a HOME and a HOME isn't allowed while the device is in motion.
86	Stall detected. The Stall Flag (ST) has been set to 1.
87	In Clock Mode, JOG not allowed
88	Following Error
89	Reserved
90	Motion Variables are too low switching to EE=1
91	Motion stopped by I/O set as Stop.
92	Position Error in Closed loop. motor will attempt tp position the shaft within the deadband, After failing 3 attempts Error 92 will be generated. Axis will continue to function normally.
93	MR or MA not allowed while correcting position at end of previous MR or MA.

APPENDICES

Appendix A: ASCII Table

Appendix B: Installing and Using IMS Terminal

Appendix C: Upgrading Firmware in the MCode Compatible Device

Appendix D: Commonly Used Instructions, Variables and Flags

Appendix E: Sample Programs

Appendix F: Factors Impacting Motion Instructions

Appendix G: MForce PWM Configuration (PW)

Page Intentionally Left Blank

APPENDIX A

ASCII Table

Dec	Hex	Char
0	0	NUL
1	1	SOH
2	2	STX
3	3	ETX
4	4	EOT
5	5	ENQ
6	6	ACK
7	7	BEL
8	8	BS
9	9	TAB
10	A	LF
11	B	VT
12	C	FF
13	D	CR
14	E	SO
15	F	SI
16	10	DLE
17	11	DC1
18	12	DC2
19	13	DC3
20	14	DC4
21	15	NAK
22	16	SYN
23	17	ETB
24	18	CAN
25	19	EM
26	1A	SUB
27	1B	ESC
28	1C	FS
29	1D	GS
30	1E	RS
31	1F	US

Dec	Hex	Char
32	20	<Space>
33	21	!
34	22	“
35	23	#
36	24	$
37	25	%
38	26	&
39	27	‘
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	´
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	DEL

NOTE: IMS Terminal is available from two locations, the Product CD that shipped with your product, and the IMS Web Site at http://www.imshome.com. Please check the web site for updates to IMS Terminal and MCode Firmware.

APPENDIX B

IMS Terminal

Section Overview

This section will acquaint the user with the IMS Terminal Software.

- Installing IMS Terminal Software.
- Configuring IMS Terminal for use with your product.
- Downloading and Uploading Programs to and from your MCode Compatible Device.

Introduction

NOTE: An Interactive Tutorial detailing the installation process is available online at http://www.imshome.com/tutorials.html

The IMS Terminal Software is a programming/communications interface. This program was created by IMS to simplify programming and upgrading MCode compatible devices: Motion Control MDrivePlus and MForce. The IMS Terminal Software is also necessary to upgrade the firmware in your device. These updates are posted to the IMS web site at www.imshome.com as they are made available.

Installation

- IBM Compatible PC.
- Windows XP Service Pack 2.
- A free USB or Serial Communications Port.

To install the IMS Terminal Software onto your hard drive, insert the Product CD into your CD-ROM Drive. The CD should autostart to the Product CD Main Index Page if you have the Macromedia Flash player installed. If the CD does not autostart, right click on the drive icon of your CD drive in My Computer and select "Explore". The IMS Terminal Software is located in the "Software" folder. The Product CD Main Index Page will be displayed.

1) Click the button in the upper right navigation area labeled "Software".
2) In the software dialog, click the IMS Terminal link and the "Setup" dialog box will be displayed.
4) Click the option "Open" on the dialog box, this will initiate the setup program for IMS Terminal.
5) Follow the on-screen prompts to complete the installation of IMS Terminal.

Figure B.1: Product CD Main Index Page

Figure B.2: Product CD Software Selection Screen

NOTE: The device may be programmed using any ASCII text editor in combination with a terminal emulator such as Hyperterminal. Please be aware that upgrading MCode firmware in your device **REQUIRES** IMS Terminal.

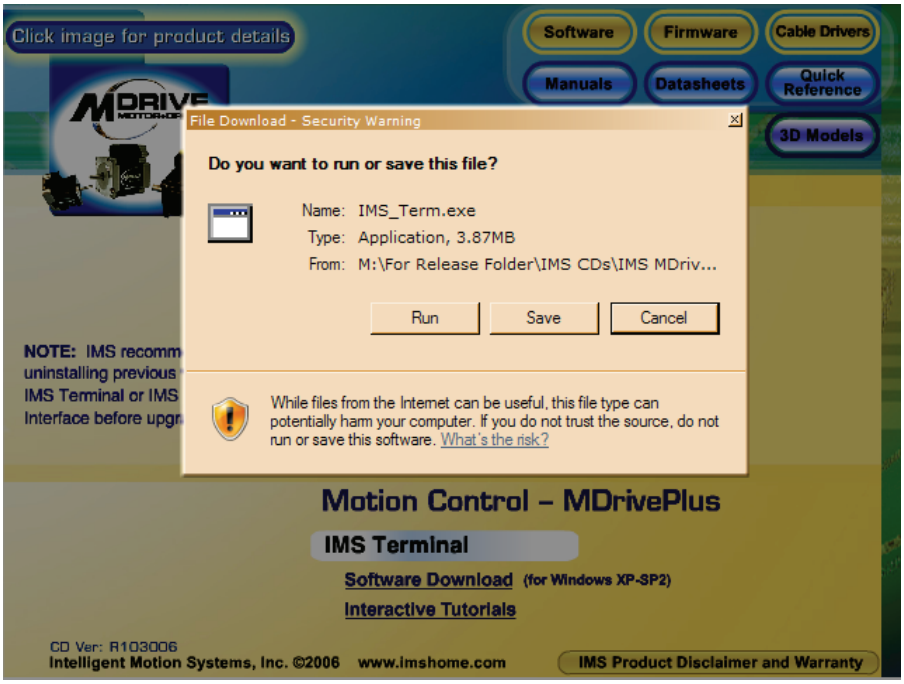


Figure B.3: Product CD Software Setup Command

TIP: To get your IMS Terminal look to match that shown in the screen captures of this document, select Window>Tile Horizontally on the menu bar.

Introduction To IMS Terminal

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

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

Features

General Features

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

Program Editor Window

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

Terminal Emulator Window

- Programmable Function Keys set in Groups of Ten.
- Multiple Function Key Groupings.
- Communications Settings defaulted to MCode device settings.
- Special “Capture Mode” allows the capture of all terminal communications to a text file.
- Terminal Window Status, i.e. Connected/Disconnected, Port #, BAUD Rate and etc. displayed on a clickable bar across the bottom of the Terminal Window.

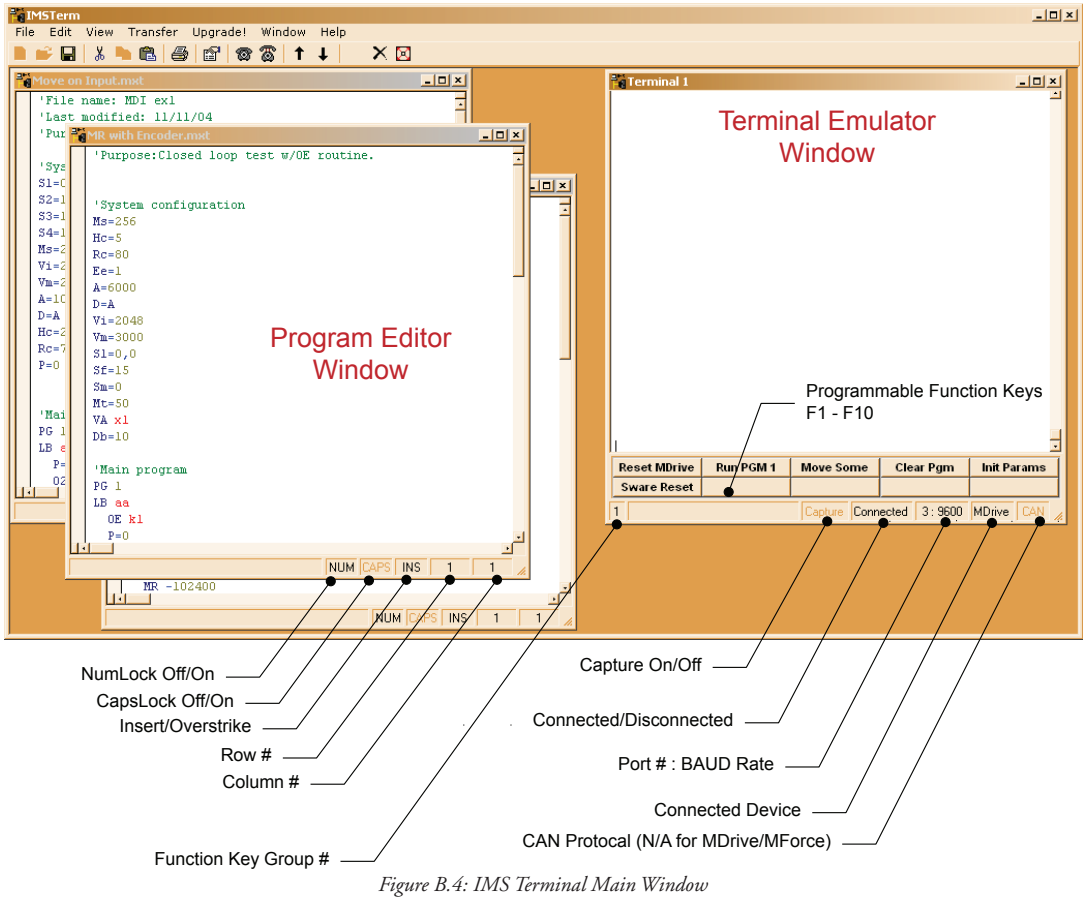


Figure B.4: IMS Terminal Main Window

The Menu Bar Structure and Operation

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

NOTE: While you may have multiple Program Editor and terminal Windows open at the same time, only one window can be active at any one time. Be aware that the menu items and button bar buttons may function differently based upon whether or not a Program Editor or Terminal Window is active!

File Menu Functions

The illustration in Figure B.5 details the functions of the file menu functions. These functions, when used in the Program Editor, are common to most Windows Programs. When the Terminal Window is in an active state, the New, Open and Save items will disable. Save As will allow the user to save the contents of the Terminal Window, including the Scroll Back Buffer, if desired to a text file.

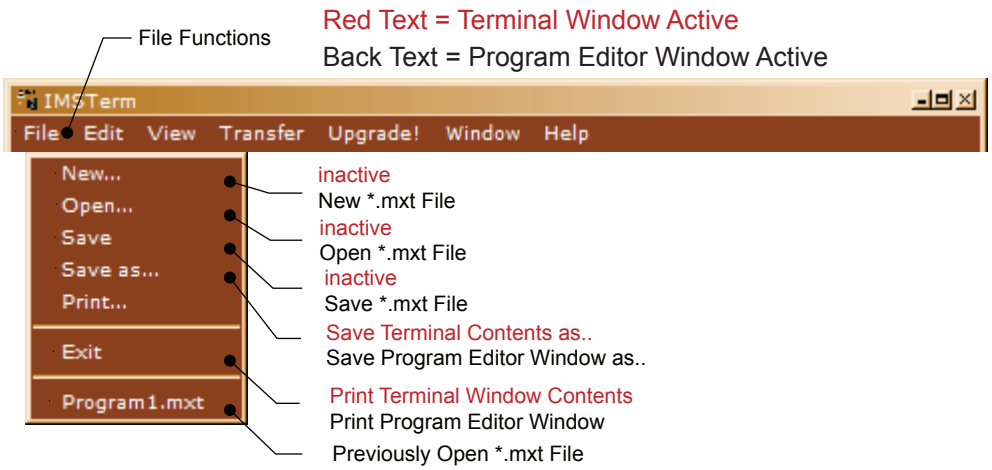


Figure B.5: IMS Terminal File Menu Functions

Edit Menu Operation

As with the File Menu, many of the Edit Menu functions are familiar to Windows users such as Redo, Undo, Copy, Cut, Paste, Delete, Select All, Find, Find Next and Replace. These will function as they do with other Windows-based software programs. Of these, only Copy and Paste will be available if a Terminal Window is active.

Preferences

The Preferences item will open a tabbed dialog with the various window configuration settings. If selected with the Program Editor Window Active, it will display the font, color, style selections for the Program Window.

If selected with the Terminal Window Active, it will display the visual configurations settings for the Terminal Window. These Configurations will be discussed in detail later in this Appendix.

Open Preferences

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

1. Text Editor Preferences
2. Program Editor Preferences
3. Terminal Format Preferences
4. Terminal Window Communications Settings
5. Open Program or Text Files
6. Open Terminal Windows
7. Function Key and Group Configuration

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

NOTE: IMS Terminal is available from two locations, the Product CD that shipped with your product, and the IMS Web Site at http://www.imshome.com/software_interfaces.html.
Please check the web site for updates to IMS Terminal and Firmware.

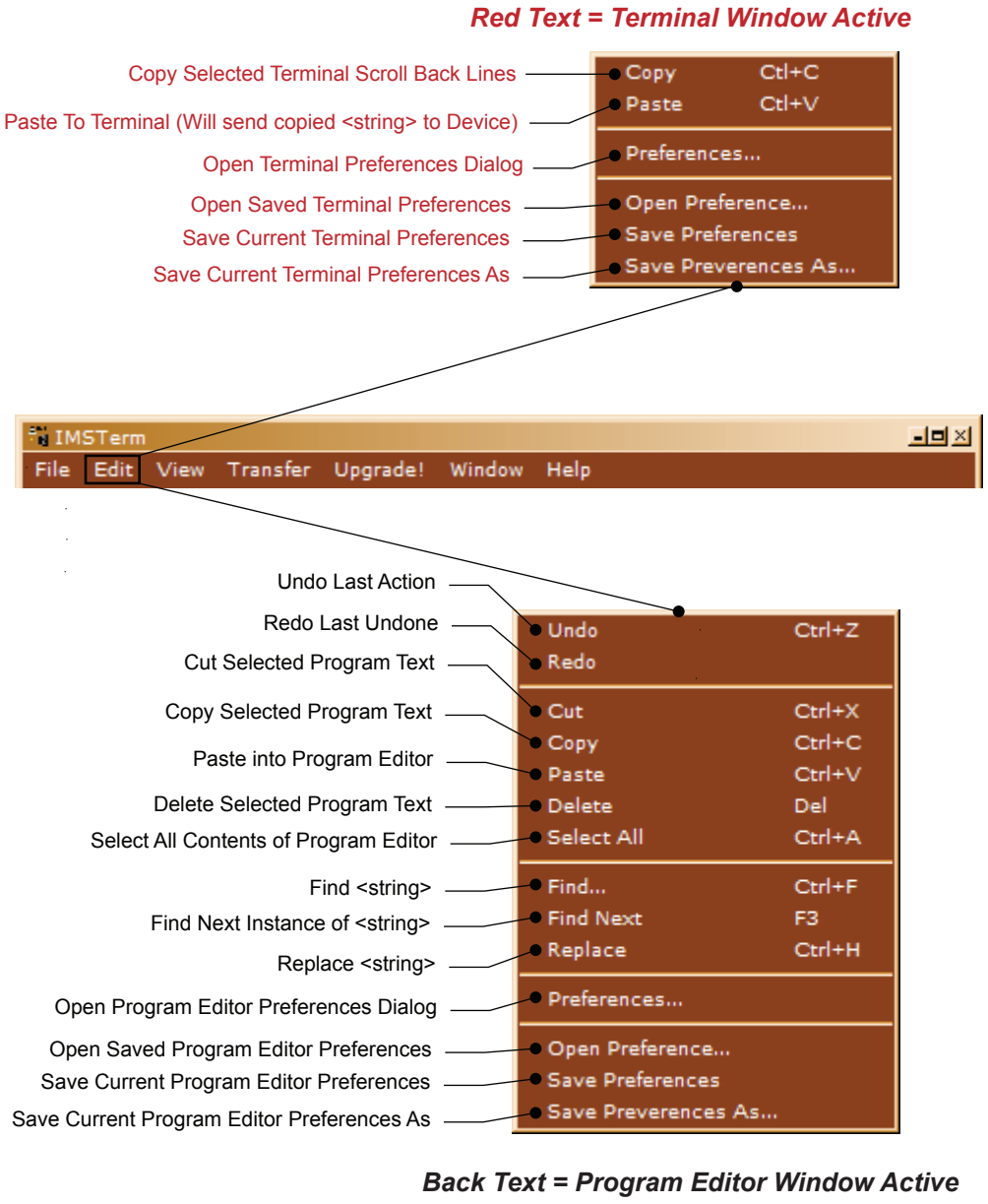


Figure B.6: IMS Terminal Edit Menu Functions

Save Preferences

Save Preferences will save the current preferences to the loaded file.

Save Preferences As

Save Preferences as different *.ltm filename.

View Menu Operation

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

- New Edit Window
- New Terminal Window

The operation of these options are covered in *Using the IMS Terminal.*

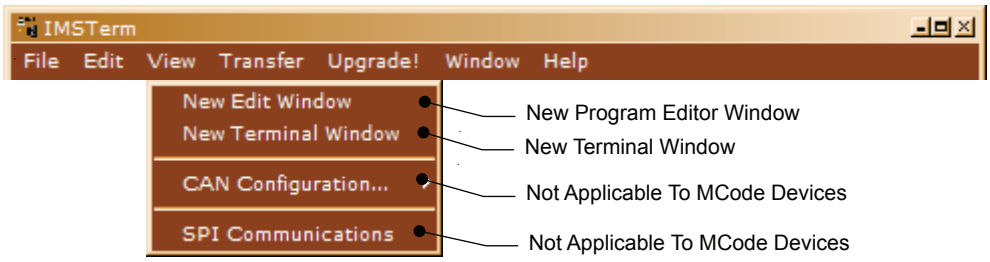


Figure B.7:IMS Terminal View Menu Functions

Transfer Menu Operation

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

Upload

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

Download

The Download Menu item will open a dialog allowing the user to selectively download Variable and Flag declarations and Programs to the MCode compatible device from either an open Program Editor window or a saved *.mxt file.

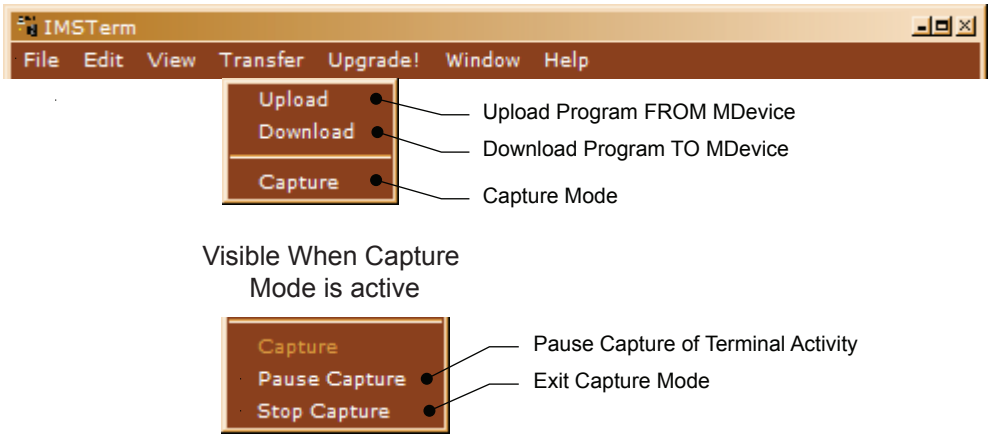


Figure B.8: IMS Terminal Transfer Menu Functions

Capture

The Capture menu item will place the Active Terminal window into a special “Capture Mode”. When in Capture Mode all Terminal activity is captured to a text file in real time. The On/Off state of Capture Mode is displayed on the bottom indicator bar of the active Terminal window.

When in Capture Mode, two additional menu items appear on the Transfer menu:

1. Pause Capture
2. Stop Capture

These will function as described by their label.

NOTE: Please read Appendix C: Upgrading Firmware in its entirety prior to attempting to upgrade the firmware in your device!

Upgrade Menu Operation

The Upgrade! menu item will open the IMS Terminal Upgrader Utility for upgrading the firmware in your MCode Device. It will not place your device in the upgrade mode for upgrading.

Before attempting to upgrade your firmware, please read Appendix C: Upgrading Firmware in it's entirety.

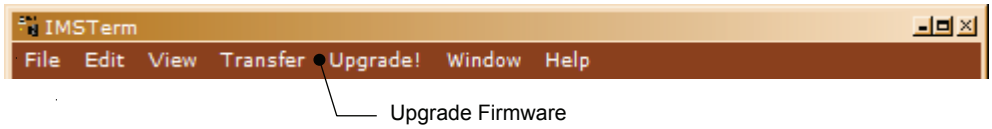


Figure B.9: IMS Terminal Upgrade Menu Functions

Window Menu Operation

The Window menu offers the options common to most Microsoft Windows™ programs. The illustration in Figure B.10 lists details of these options.

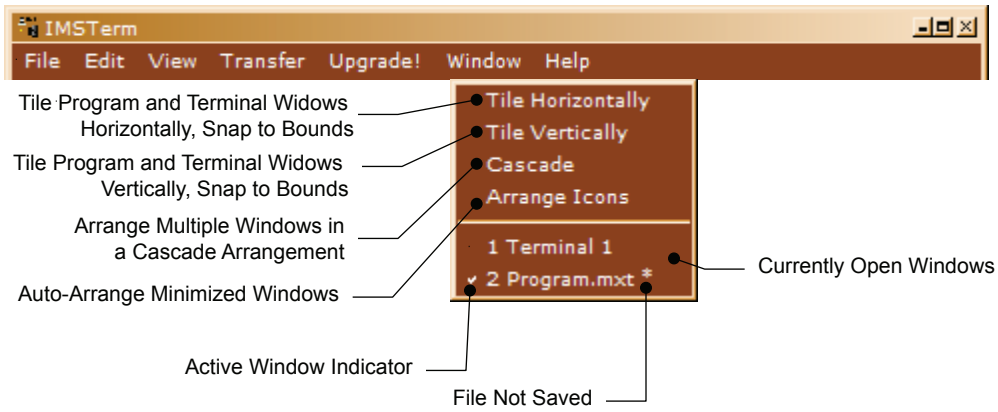


Figure B.10: IMS Terminal Edit Menu Functions

Help Menu Operation

The Help menu features a direct Web-link to the IMS Interactive Tutorials.

These Tutorials are in Adobe Flash format and are an excellent companion to this document. These are available online at http://www.imshome.com/tutorials.html or on the product CD which shipped with your device.

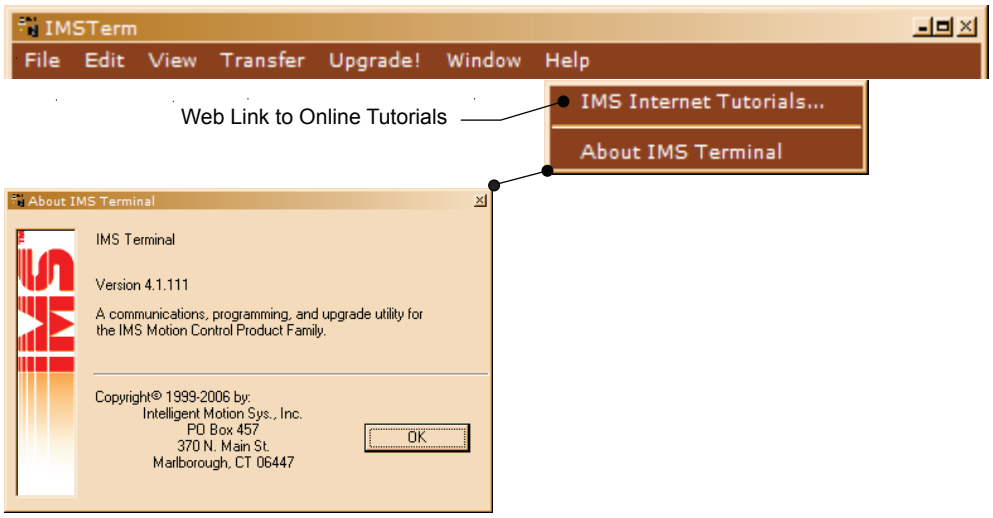


Figure B.11 IMS Terminal Help Menu Functions

Figure B.12: IMS Terminal Interactive Tutorials

Available Tutorials

1. Installing the IMS Terminal

This tutorial covers the installation of IMS Terminal.

2. Configuring Communications

This tutorial covers the communications configuration of the IMS Terminal to operate with the MDrivePlus Motion Control.

3. Configuring Function Keys

The Function Keys are a powerful part of the IMS Terminal. They can be programmed to a number of functions. This tutorial will step through setting up some basic functions.

4. Downloading a Program to the MDrivePlus

This tutorial covers downloading a program from the Program Editor Window to the MDrivePlus Motion Control.

5. Upgrading the MDrivePlus Motion Control Firmware

This tutorial covers firmware upgrades to the MDrivePlus Motion Control.

The Button Bar Structure and Operation

Program Editor Active

The Button Bar offers shortcuts to standard Windows functions when the program editor is open. These functions are defined below.

Terminal Window Active

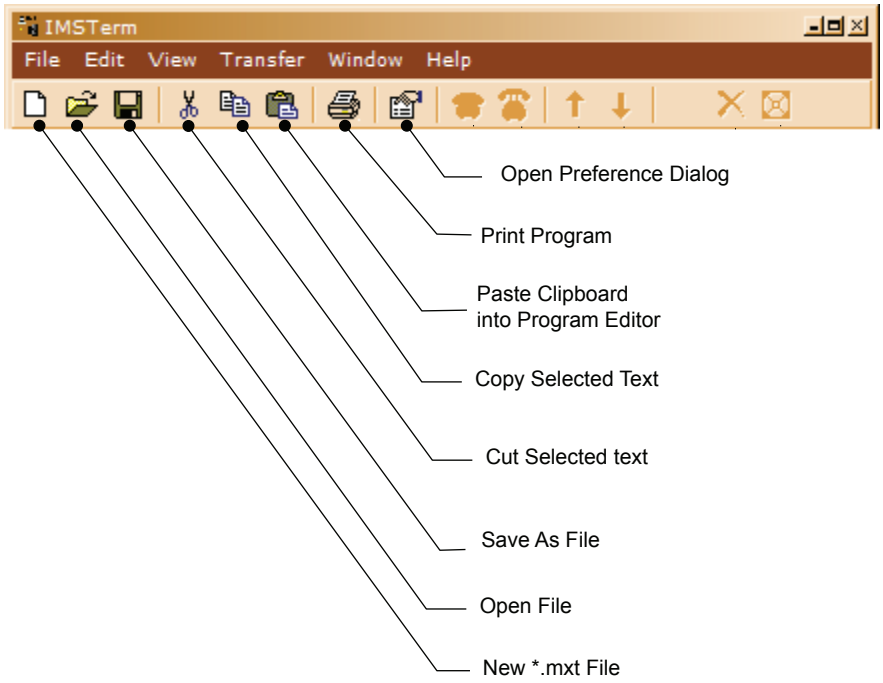


Figure B.13: IMS Terminal Button Bar (Program Editor Active)

The Button Bar offers shortcuts to Communications and text manipulation functions when the program editor is active, these functions are identified below.

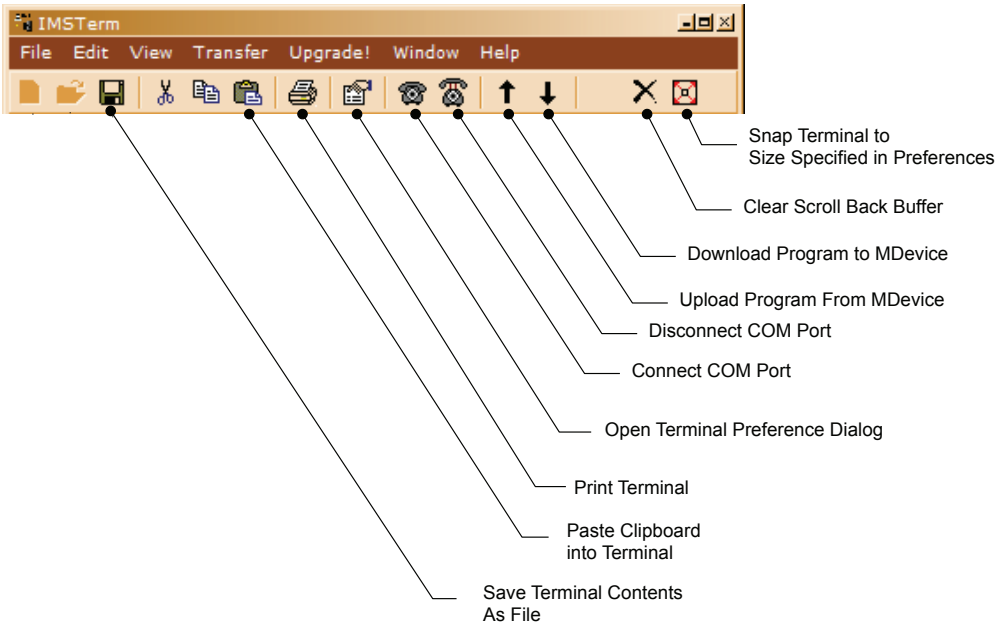


Figure B.14: IMS Terminal Button Bar (Terminal Active)

Configuring Program Editor Format Preferences

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

While the default format of the Program Editing Window is sufficient for most users, you may configure the format to your preference.

To open the Program Editor Preferences Dialog:

1. Right-Click into the Program Editor Window.
2. Select Preferences.
3. The Program Editor Format tab of the Preferences dialog will open.
4. Using the Diagram shown in below as a reference, set the format to that which you desire.
5. Note that the use of a mono-spaced font such as the default, Courier New, makes code editing and debugging much easier than using a variable spaced font.

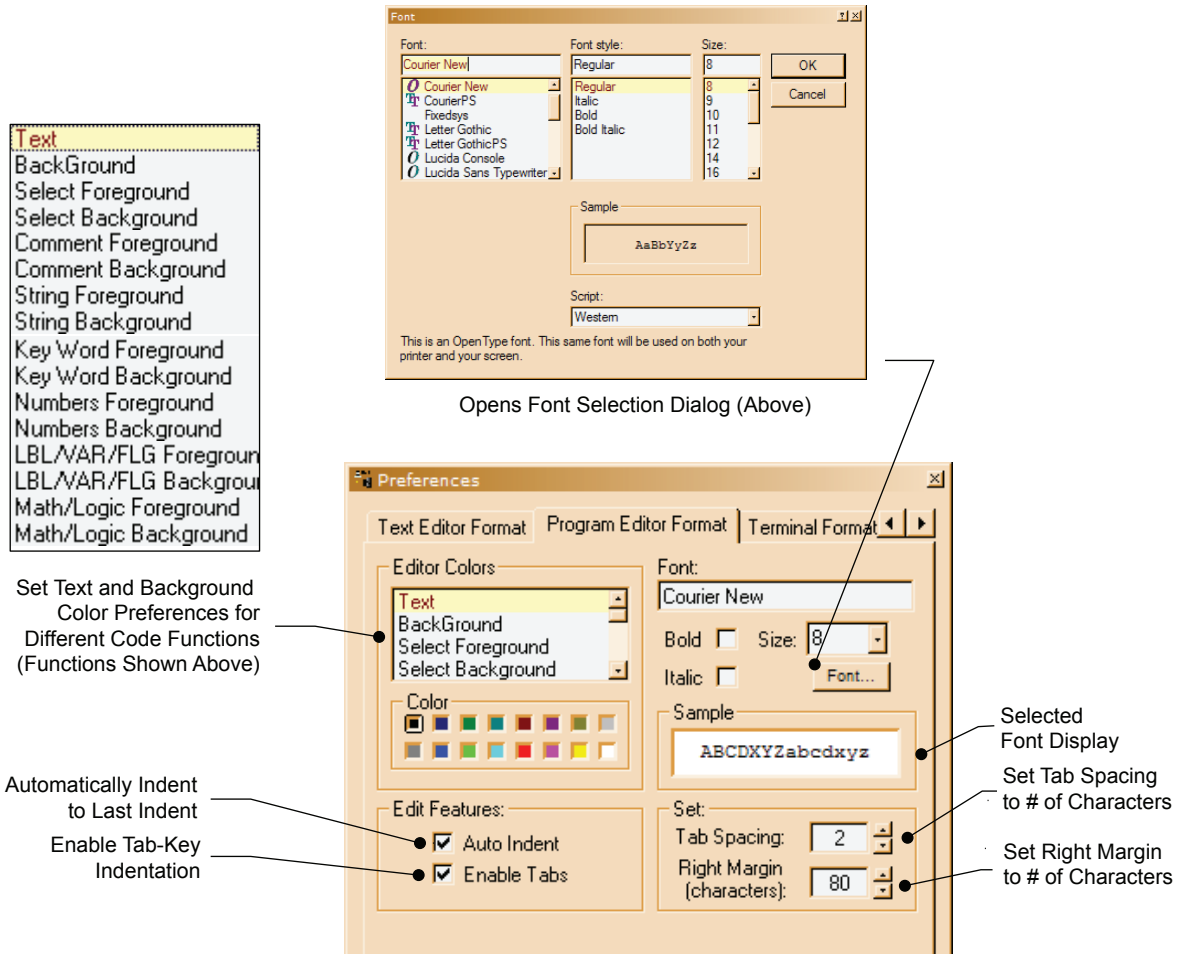


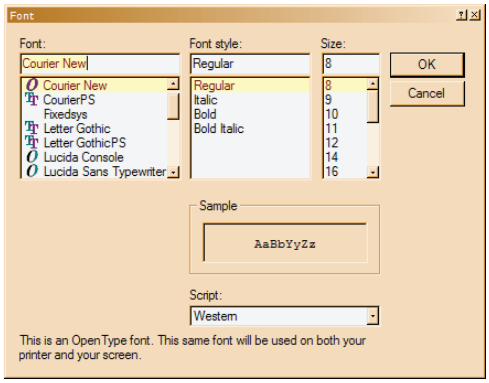
Figure B.15: IMS Terminal Program Editor Preferences

Configuring Terminal Window Format Preferences

The Terminal Window features similar formatting preferences as the Program Editor.

To open the Terminal Format Preferences Dialog:

1. Right-Click into the Terminal Window.
2. Select Preferences.
3. The Terminal Format tab of the Preferences dialog will open.
4. Using the Diagram below as a reference, set the format to that which you desire.



Opens Font Selection Dialog (Above)

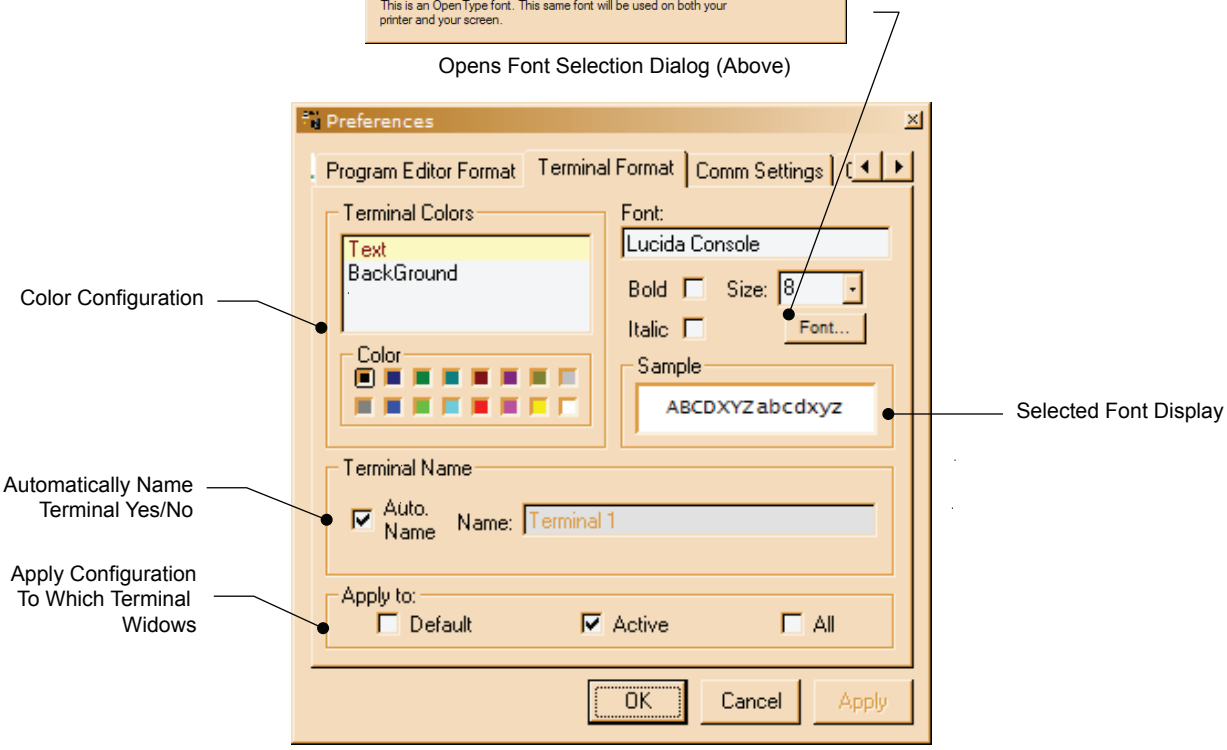


Figure B.16: Terminal Window Preferences

Configuring Communications Settings

Click the Image above to launch an interactive tutorial on Configuring Communications!

The communications settings are configured by means of the Preferences Dialog.

The optimum communications settings for the MCode compatible device are set by default. The only thing that will need to be set to begin communicating with your MCode device is to set the COM Port to which your RS-422/485 Communications converter is connected.

1. Open the Communications Preferences Dialog by either opening the preferences from the menu bar and selecting the Comm Settings Tab, or double clicking the Port: BAUD Rate field on the status bar at the bottom of the Terminal Window.
2. The Communications Preferences are already configured for your MCode device. The only setting you will need to change is the COM Port to the Port your RS-422/485 Converter is connected to.
3. The Window Size and Function Key Settings are optional.

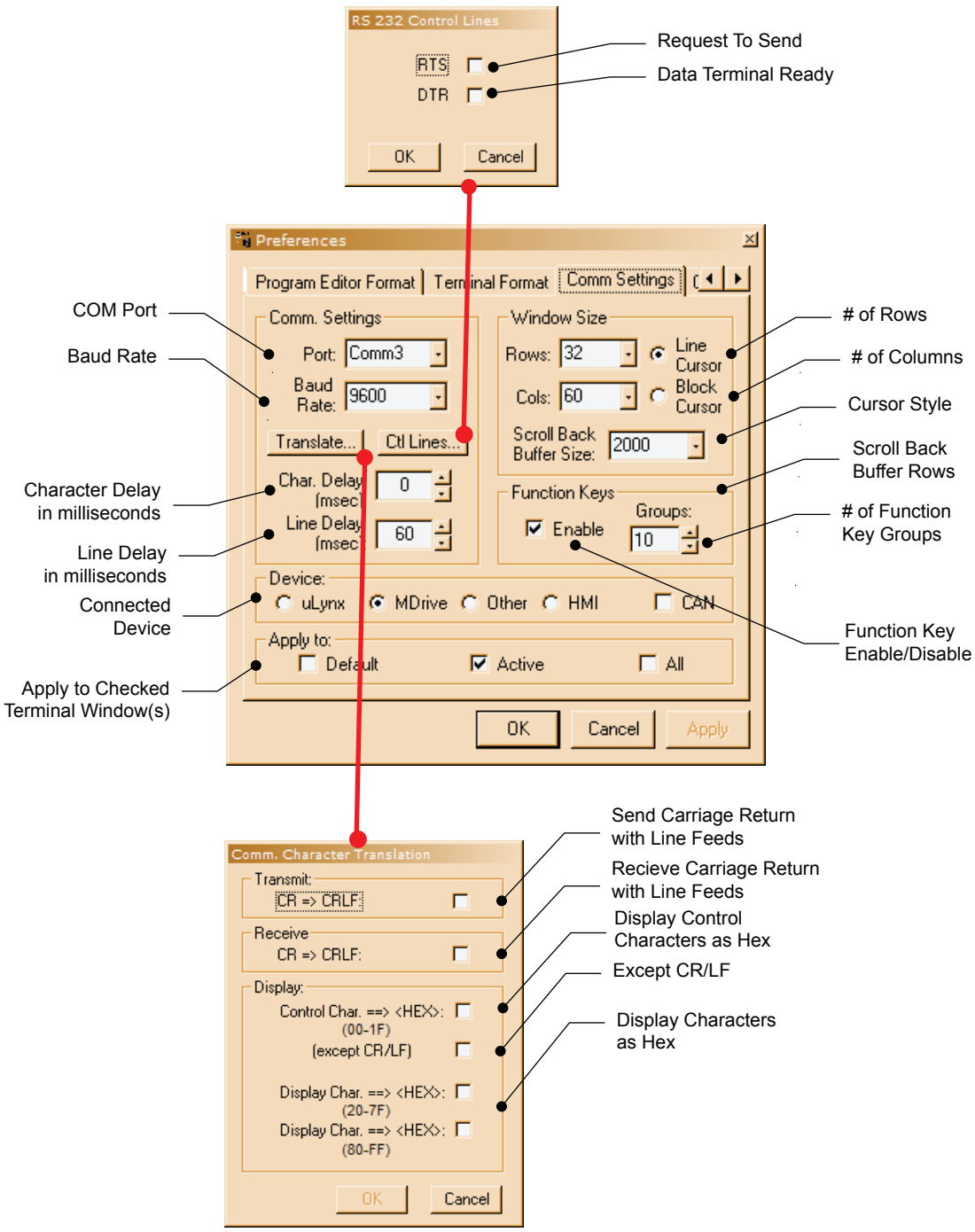


Figure B.17: Communications Preferences

4. Once you have selected the correct COM Port, click OK.
5. Verify hardware connections and apply power to the MCode Device.
6. If not already connected, connect to the device by clicking the "Connect" Icon on the button bar, or by double clicking the Disconnected field on the status bar of the Terminal Window.
7. Key in CTRL+C.
8. The sign-on message shown in Figure B.18 should appear.

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

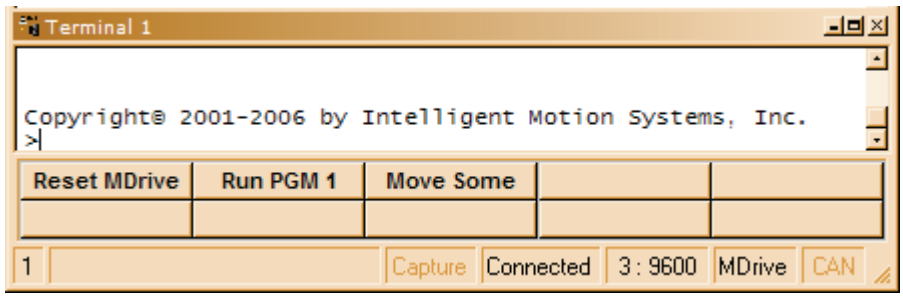


Figure B.18: IMS Sign-On Message

Troubleshooting Communications

Troubleshooting the Communications Converter

1. Go to the Start Menu on Windows XP.
2. Right Click “My Computer”, select “Properties”.
3. On the System Properties Dialog, click the Tab labeled “Hardware”.
4. Click the Device Manager button.
5. On the Device Manager window, click the “+” sign next to Ports (COM and LPT) to expand the category.
6. The RS-422 Converter should be listed there with its Port number (See Figure B.19).
7. Verify that the Port is the one configured in the Communications settings for IMS Terminal.

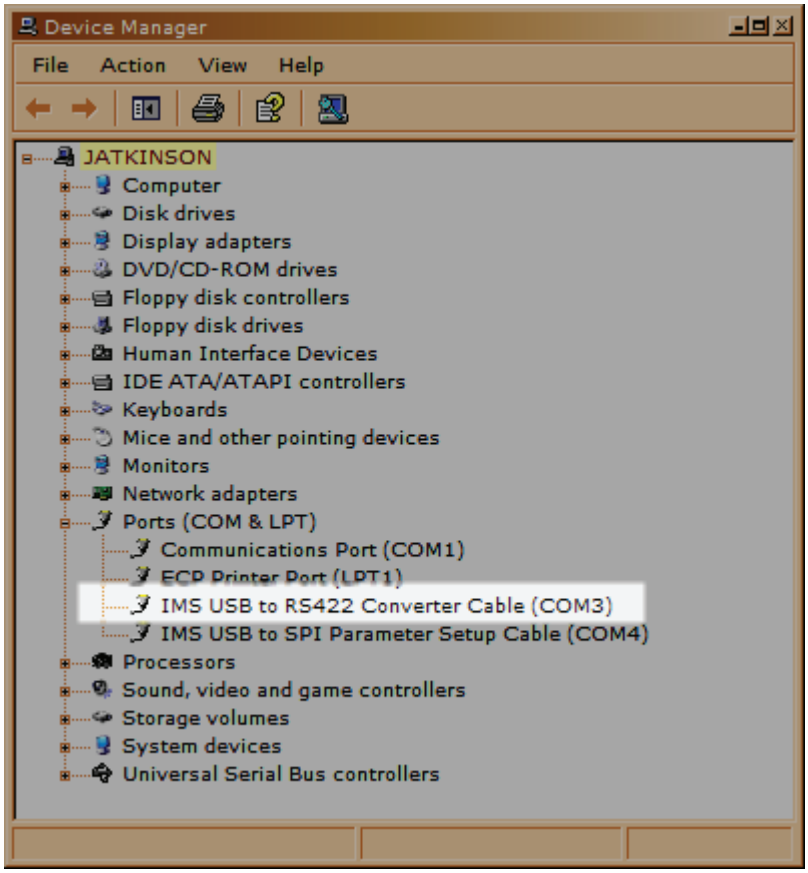


Figure B.19: Windows Device Manager showing COM Port

8. If the Communications converter DOES NOT show in the Device Manager verify that the drivers for the converter are installed per the Converter Manufacturer Documentation.
9. If the Converter will not install, contact the device manufacturer support desk.

Communications Converter Installed and Functioning

1. Verify that all mating connectors are firmly seated.
2. If not using an IMS MD-CC40x-000 and appropriate adapter, verify the wiring.
3. Check the following connections:

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

Configuring Function Keys

Click the Image above to launch an interactive tutorial on Configuring Function Keys!

The ability to Program MCode Functions and Code Strings and assign a Function Key to them is one of the most powerful features of IMS Terminal.

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

The illustration below shows the Function Key Dialog with example Functions programmed in.

To open the Function Key Dialog:

1. Right-Click any of the Function Keys on the bottom of the Terminal Window or key-in CTRL+[F1-F10].

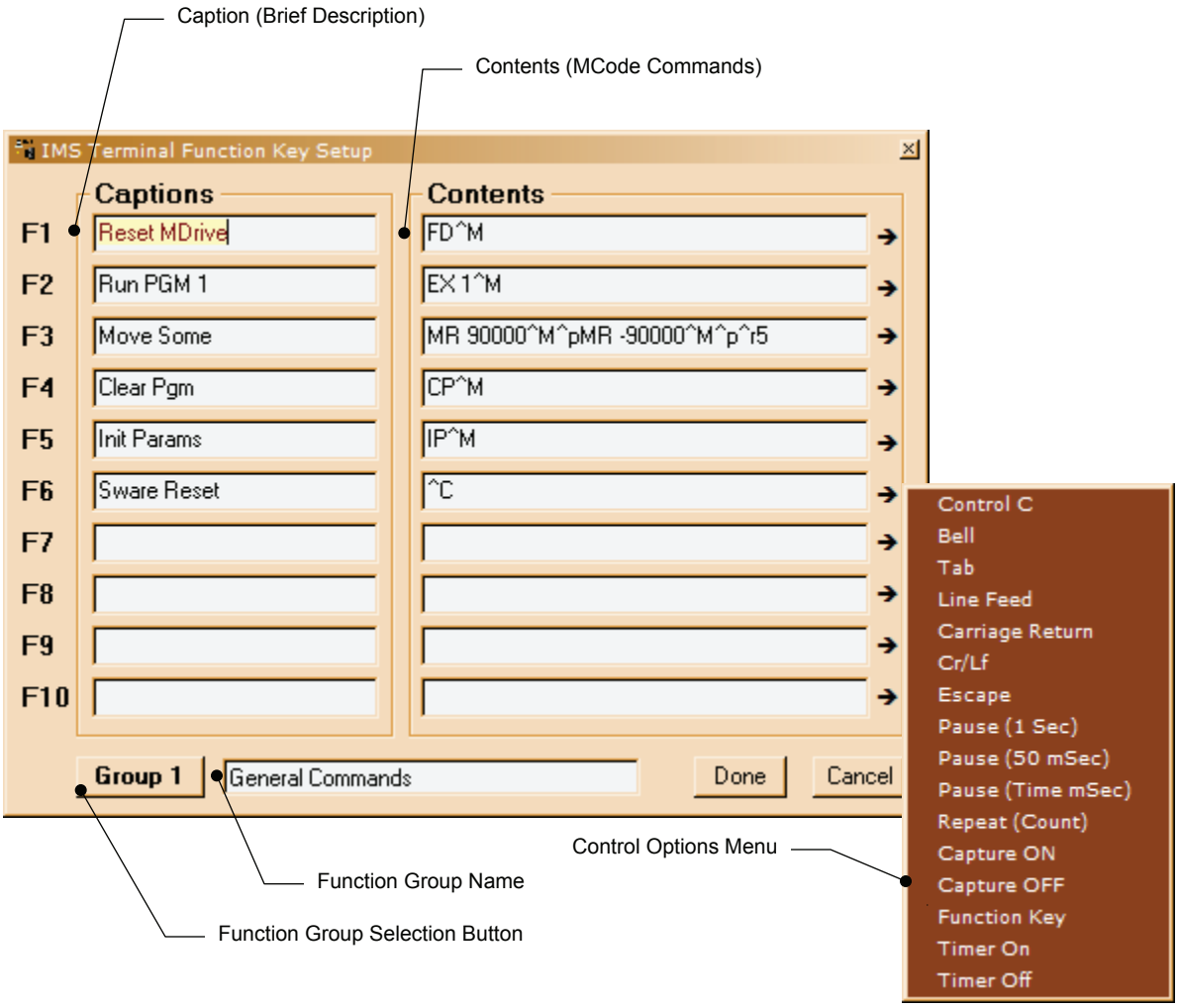


Figure B.20: Function Key Dialog

Function Key Control Commands

The IMS Terminal Function Keys have a number of Control Commands that are used to input or impact MCode strings sent to the Terminal on Function Key press.

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

Table B.1: Function Key Control Commands

Setting Up A Function Key

The Function Keys you see in this document have already been set up for you in the default Preferences file.

Access the Function Key dialog by right-clicking a Function button or by keying in CTRL+<F1-10>.

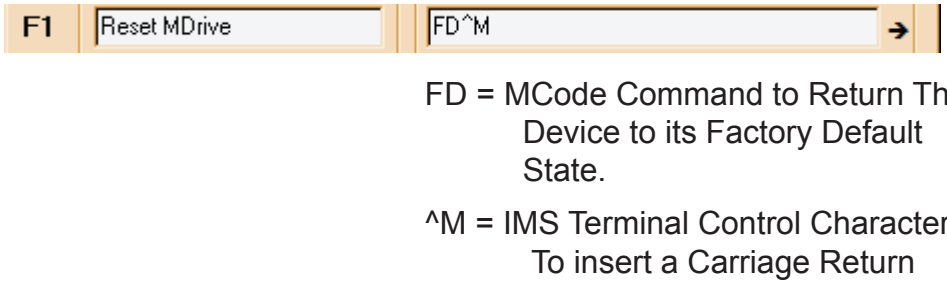
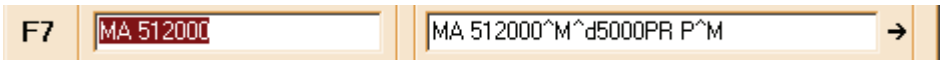


Figure B.21: F1 Setup

Study how the Function keys have been set up to send various immediate mode commands to the terminal

Following the examples shown in the Functions F1-F6 in Function Group 1, set up Function Key F7 to perform an Absolute Move to Position 512000 (MA 512000), wait 5 seconds and Print the Position (PR P). We will need to include the appropriate delays.

1. In the Caption field for F7 Enter MA 512000.
2. Begin entering code into the Contents field by keying in MA 512000.
3. Set a Carriage Return by selecting “Carriage Return” on the fly out menu, or key-in ^M.
4. Select “Delay (Time) from the fly out menu or key-in ^d to set a delay.
5. Set the delay to 5000, or 5 seconds.
6. Key-in PR P to print the position to the Terminal.
7. Set a Carriage Return. Your F7 fields should appear thus:



Creating, Downloading and Uploading Programs

Click the Image above to launch an Interactive tutorial on Downloading Programs!

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

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

Creating a New Program

Before you create a program you must have a new Program Editor Window open. Follow these steps:

1) Click on the Drop-Down Menu "View". The following dialog box will be displayed:
2) Click on "New Edit Window". The dialog box in Figure B-22 will be displayed.
3) You must assign a file name in order to open the new window. If there is no file name the "OK" button will not be highlighted. Name this file <My Program.mxt>. The <mxt> extension designates MCode programs.
4) Click "OK" and the new Program Editor Window will be displayed.

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

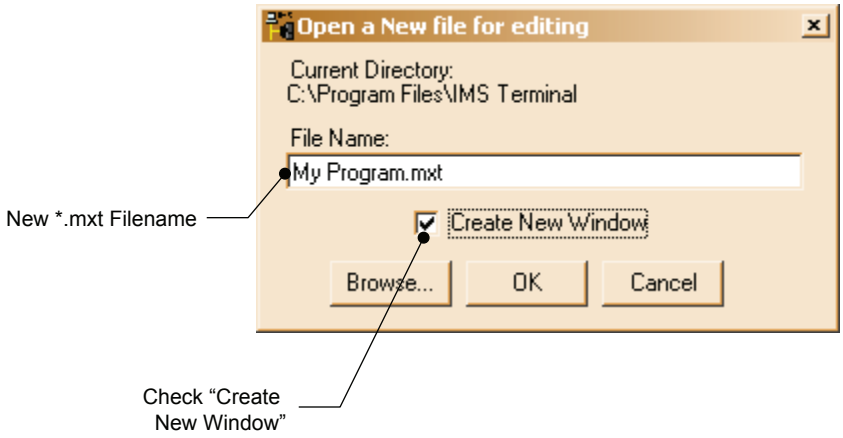


Figure B.22: New Editor Window

Downloading a Program

NOTE: Before downloading any programs type FD into the Terminal Window and press ENTER to set the device to the Factory Defaults.

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

1) Directly from the Program Editor Window of the IMS Terminal.
2) From a file folder located on a hard drive or removable disk.

To download:

1. Activate the Program Editor Window containing the Program MR.mxt.
2. Click the menu item "Transfer > Download". The Download Dialog Box will open (Figure B.23).
3. Click the Download Button on the Main Tool Bar. The Download Dialog Box will open. Select the "Source Type > Edit Window" option, and click download. The program will transfer to the device. If a Program has been previously created and stored, it may be downloaded to the device from the *.mxt file.
4. Once the program is downloaded, type S and press ENTER to Save the program. (Always save your programs!)
5. Now type EX 1 and press Enter or Click the Function button Run PGM 1. (EX=Execute and 100 is the Program Number.) The motor should move a short distance back and forth.
6. NOTE: The program can be stopped by pressing the Escape Button or by pressing <Ctrl+C>.

TIP: Naming the MCode program using the extension *.mxt will allow the IMS Terminal to automatically format and color-code the program code for ease of visual editing!

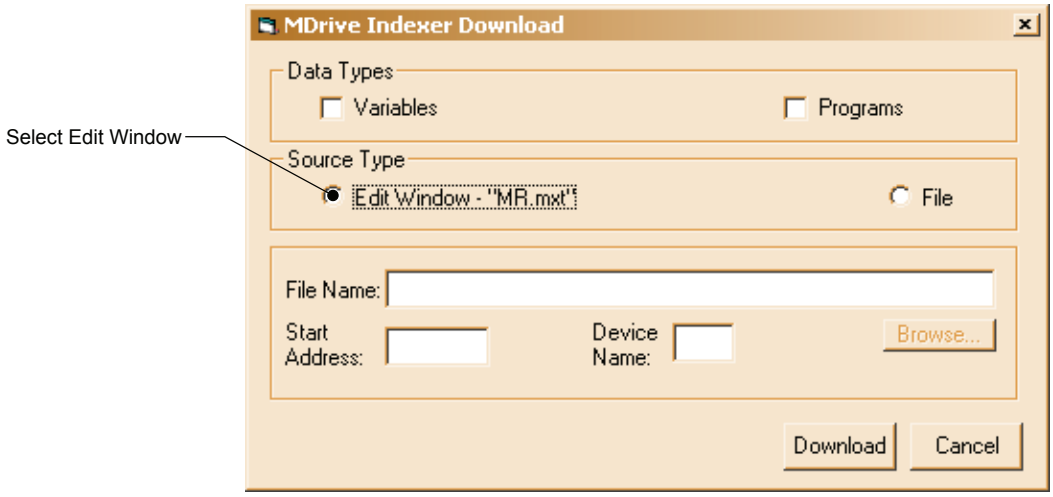


Figure B.23: Download Program To MCode Device

NOTE: The program is not downloaded to the Terminal Window. It is downloaded directly to the device. What is shown in the Terminal Window is an echo of the downloaded program.

NOTE: Because the program is downloaded directly to the device, the unit must be powered up and the sign-on message must be displayed (communicating).

NOTE: When the program is downloaded, the color of all characters will be changed to black and line numbers will be added.

NOTE: After the program is downloaded it must be saved. Type an <s> next to the cursor and press Enter to save the program.

Uploading a Program

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

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

1) Directly to the Program Editor Window of the IMS Terminal.
2) To a file folder located on a hard drive or removable disk.

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

1) Click the menu item "Transfer > Upload". The Upload Dialog Box will open.
2) Click the Upload Button on the Main Tool Bar. The Upload Dialog Box will open. The Upload Dialog box is similar in appearance to the Download Dialog box.

With the Upload Dialog Box open, select the "Destination Type > Edit Window" option, click "Upload". The program will transfer from the device.

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

NOTE: When uploading MCode Program Files they will be slightly changed from the original. The device will upload the Program only with the data within the Program. That is, the data between the two Program Modes (PG). Data such as Variables entered outside the PG Modes will not be uploaded. The uploaded program

TIP: In the IMS Terminal, Downloading sends the program to the MCode Device, Uploading is bringing the contents of Non-volatile memory into the text editor window or to a file.

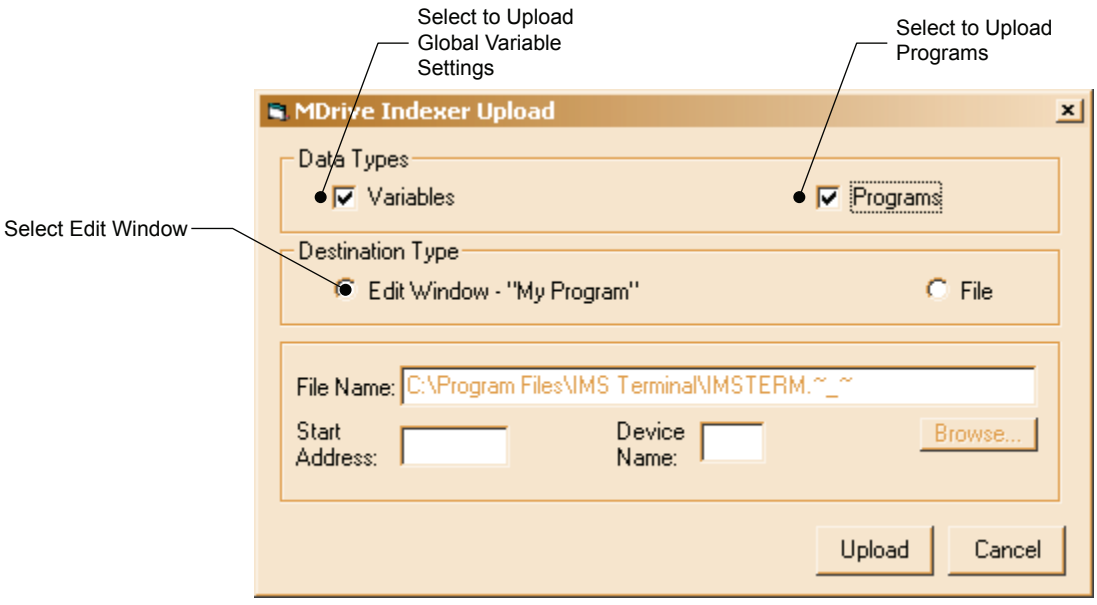


Figure B.24: Upload Programs to Program Editor Window

will also have a header '[PROGRAMS] and a footer '[END]. These will not affect your program as they are remarked with the apostrophe (') or they can be removed during editing.

You may Upload the Program Variables by clicking "Variables" in the Upload Dialog Box. However, this will upload all of the current Variables, not just those associated with the Program.

Program Troubleshooting Using IMS Terminal

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

- Execute in Single Step Mode
- Execute in Trace Mode
- The Scroll Back Function
- The Capture Function

Single Step Mode

The Single Step Mode allows the user to execute a program in the Immediate Mode one line at a time. This will help the user to define problem areas by process of elimination. To use Single Step Mode, do the following:

It is recommended that you List (L) the program in the Terminal Window and either print it on paper or cut and paste it to another Program Edit Window. This will allow you to look ahead and see what line is coming up next.

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

   Example: The Program Label is <aa>. Type EX aa,2. The program will run one line at a time.
3) Each line will be executed and listed in the Terminal Window and the Program will stop.
4) To execute and list the next line, press the Space Bar.
5) Press the Space Bar for each successive line until the program has completed.

While the program is executing, it will stop after each line is listed. At this time you may enter immediate commands such as velocity variables or actual moves as tests within the program. After entering immediate commands you may continue running in Single Step Mode by pressing the Space Bar again.

If you decide to cancel the Single Step Mode press the "Enter" key and the program will run in normal mod and finish or press Escape (Esc) to abort the program.

NOTE: The Capture Function may also be enabled through the Fly-Out menu on the Function Key configuration page by inserting it into the command string in the "Contents" line. However, the Capture Function can not be programmed with the Repeat command.

Trace Mode

The Trace Mode allows the user to run a program and list each line as it is executed. Running Trace Mode in conjunction with the Scroll Back Function or the Capture Function will enhance your program troubleshooting tasks. To run Trace Mode:

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

   Example: The Program Label is <aa>. Type EX aa,1. The program will run in Trace Mode and each line will be executed and listed in the Terminal Window.
3) Each line can now be analyzed.

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

TIP: Function Keys may also be assigned to "Capture" and "Stop Capture" from the flyout menu to the right of each function.

The Capture Function

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

Enable the Capture Function

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

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

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

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

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

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

Before Upgrading the MCode Firmware

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

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

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

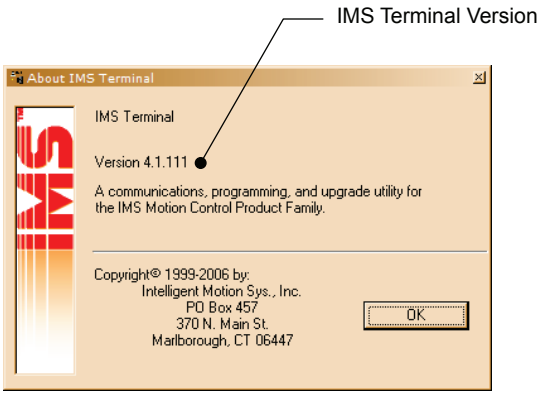


Figure C.1: IMS Terminal Information Page

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

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

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

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

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

Upgrading the Firmware

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

NOTE: Your device is configured with the most recent firmware at the time of shipment.

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

NOTES: An isolated communications system free of electrical noise and interference is essential for trouble free communication.

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

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

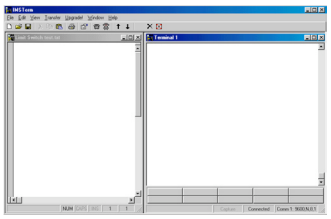
If it is determined that a firmware upgrade is necessary, download the most recent version into a known folder from www.imshome.com/flash_code.html.

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

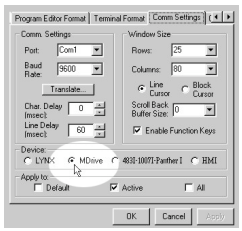
When using a laptop PC it is recommended that you power the RS-232 to RS-485 cable with an external +5 VDC power supply. This will fortify communications.

The Mdevice remains in the Upgrade Mode until the upgrade is complete. Cycling power will not clear the Upgrade Mode.

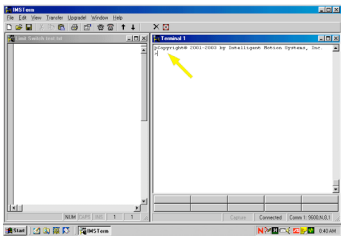
It is recommended that you use this procedure as it is tailored for the device while the on-screen instructions are designed for several different products.



1) Open “IMS Terminal”. The following screen should be displayed. The left panel is the Program Edit Window and the right panel is the Terminal Window. The Firmware Upgrade will superimpose several dialog boxes and instructions over these two windows.



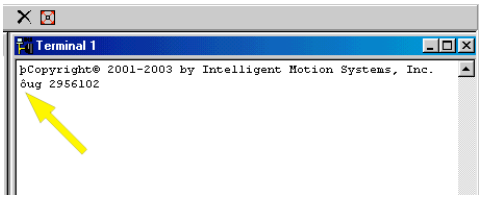
2) Check to see that the terminal window is set for MDrive communication.
   - Right click in the Terminal Window.
   - Click “Preferences” near the bottom of the pop-up menu.
   - A “Preferences” dialog box will be displayed.
   - Click on the “Comm Settings” tab at the top of the box.
   - Confirm that MDrive is selected in the “Devices” block.



WARNING! The only way to clear the upgrade mode is to complete the procedure. Cycling power will not clear the upgrade mode.

3) Power up the MDrive Motion Control.
   - The sign on message will appear.

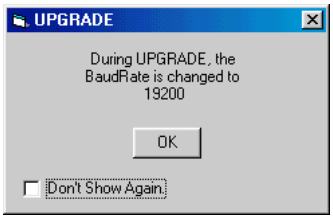
`“Copyright 2001-2006 by Intelligent Motion Systems, Inc.”`



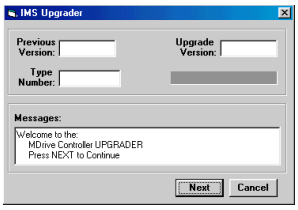
4) Check and/or reestablish communications if the sign on message does not appear.

5) Type UG 2956102 in the Terminal Window and then press <enter>. Include the space between the G and the 2.

   The MDrive will return a random symbol character (ô or ö) when it is in the upgrade mode.



6) Click the “Upgrade” menu item on the IMS Terminal menu bar.



7) Message appears: “During upgrade, the baud rate is changed to 19,200.”
   - Click “OK”

9) The Windows Explorer page “Select MDrive upgrade file” opens.
   - Browse and select the desired version of the upgrade file.
   - Click “Open” or double click the file.

10) Message appears: Step 2 Select upgrade file.
   - The Upgrade Version will now appear in the Upgrade Version

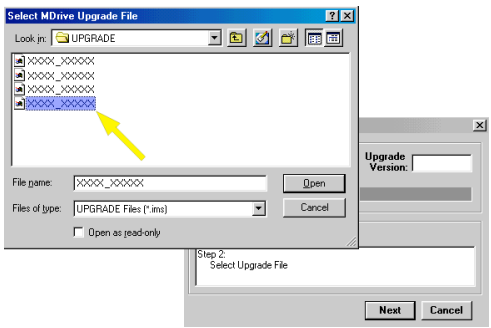


Figure C.2: MDrive Firmware Upgrade Dialog Progression 1

window.
- Click “Next”

11) Message appears: Step 3 Reminder Press cancel if you need to setup COMM port.
    - The COMM port has been setup previously. This is just a reminder.
    - Click “Next”

12) Message appears: Step 4 Connect RS-422 cable to the MDrive Controller.
    - THE RS-422 HAS BEEN CONNECTED PREVIOUSLY. DO NOT PERFORM THIS STEP.
    - Click “Next”

13) Message appears: Step 5 If MDrive Controller is not in the Upgrade mode, press cancel then type ‘UG 2956102’ in the terminal window.
    - THE MDRIVE CONTROLLER WAS PLACED IN THE UPGRADE MODE PREVIOUSLY. DO NOT ENTER CODE AGAIN.
    - Click “Next”

14) Message: Step 6 Power up or cycle power to MDrive Controller.
    - THE UNIT HAS BEEN PREVIOUSLY POWERED UP. DO NOT CYCLE POWER.
    - Click “Next”

15) Message: Step 7 Establishing COMM with MDrive Controller.
    - Wait for step 8 to appear.
    - The previous version of firmware will now be displayed in the “Previous Version” window.

16) Message: Step 8 Press upgrade button to start.
    - Click the upgrade button.

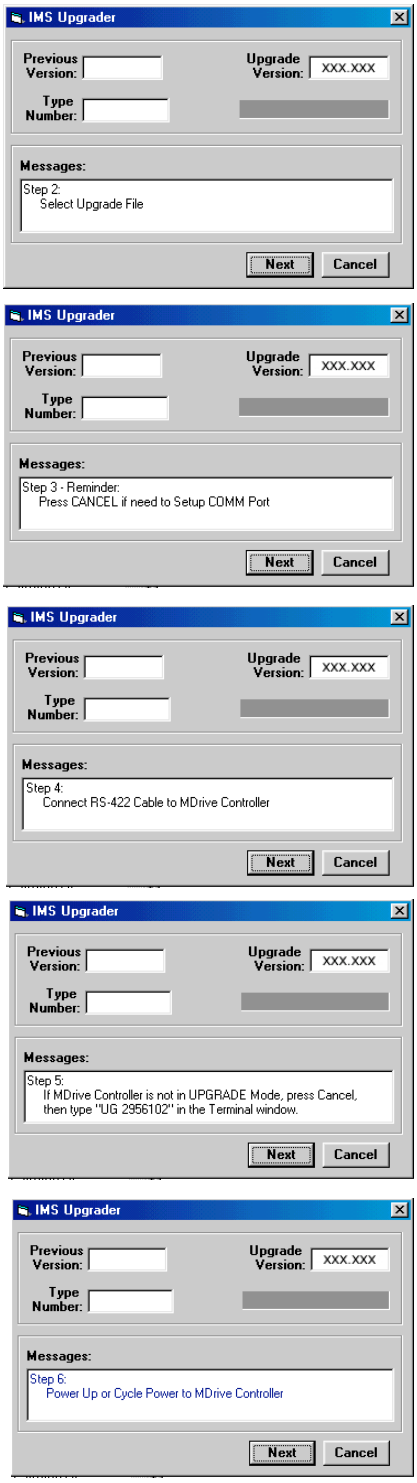


Figure C.3: MDrive Firmware Upgrade Dialog Progression 2

Note: The IMS Terminal automatically shifts to a 19,200 Baud Rate upon clicking the “Upgrade” command.

Note: In the event of loss of power or disconnection of the RS-232 cable, the unit will maintain the “Upgrade” mode on Power Up. The Upgrade must be completed. DO NOT retype “UG 2956102”!

- NOTE: An upper case E will be displayed in the “Type Number” window. This confirms the upgrade is functioning properly.

17) Message: Step 9 Press ABORT to abort upgrade.
   - DO NOT ABORT THE UPGRADE. THE MDRIVE REMAINS IN THE UPGRADE MODE AND THE UPGRADE MUST BE COMPLETED.
   - Monitor the progress in the “Upgrading…%” window.
   - Step 10 will appear when DONE

18) Message: Step 10 Resetting MDrive Controller. Then Press DONE.
   - Click “DONE”
   - Upgrade window will close.

19) Press “Control + C” <Ctrl + C> while the Terminal Window is active to reset the MDrive Controller and exit the upgrade mode.
   - The sign on message will appear. “Copyright 2001-2003 by Intelligent Motion Systems, Inc.”
   - The > cursor will appear.

20) The MDrive Motion Controller firmware has been upgraded.

21) Optional confirmation of the upgrade: Type “PR VR” in the terminal window and press <enter>.
   - The new firmware version is displayed.

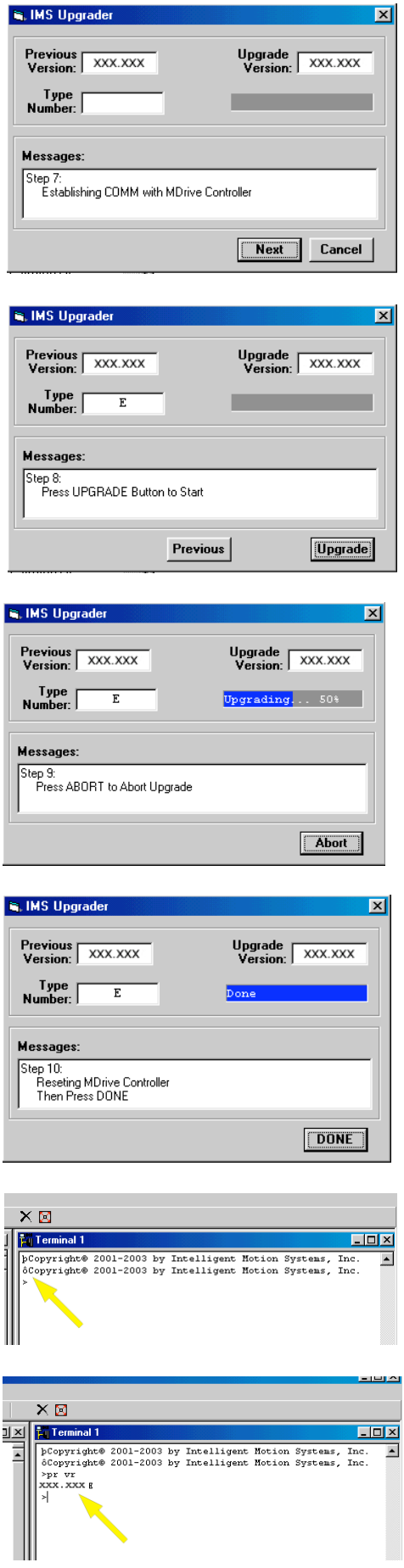


Figure C.4: MDrive Firmware Upgrade Dialog Progression 3

APPENDIX D

Most Commonly Used Variables and Instructions

Variables

MS

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

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

NOTE: There are a number of factors that impact how the various motion instructions will perform. Please see Appendix C: Factors Impacting Motion Instructions.

P

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

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

VI

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

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

VM

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

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

A

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

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

D

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

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

Motion Instructions

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

All motion is programmed either Microsteps Per Second or (when the Encoder is enabled) Encoder Counts (Pulses) Per Second.

All Motion is directly affected by the Motion Command and the Program Variables.

NOTE: There are circumstances where you may not want to hold up program execution.

There are a number of factors impacting Motion Instructions. These are addressed in detail in ***Appendix C: Factors Impacting Motion Instructions.***

MA

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 to 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

NOTE: All instructions referencing I/O Points 7-8, 9-12 and I/O13 are applicable to the MDrivePlus^2 Motion Control models with the enhanced I/O set.

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

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

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.)

I/O Instructions

IMPORTANT! All I/O Instructions relating to I/O points 7-8, 9-12 and I/O13 are applicable to the Plus^2 Motion Control Models ONLY!

S<1-4> <9-12>†

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

Using the PR command to read I/O parameters

```
Read I/O1 Setup - "PR S1"
Read I/O2 Setup - "PR S2"
```

Setting the I/O parameters

```
Set IO 3  parameters - "S3=0,1" Sets IO3 as a General Purpose Input, Active
High
```

For example: To set I/O4 as a Jog+ Input/Active Low

```
S4 =7,0
```

† S<9-12> are applicable to the MDrivePlus Motion Control Enhanced I/O models only!

NOTE: All instructions referencing I/O Points 7-8, 9-12 and I/O13 are applicable to the MDrivePlus^2 Motion Control models with the enhanced I/O set.

I<1-4><9-12>†

Used to read the state of an individual input.

```
PR I1 will read the state of input 1 and display it to the terminal window.
BR K5, I2=0 will branch to the program address labled K5 when Input 2 is
LOW
```

† I<9-12> are applicable to the MDrivePlus Motion Control Enhanced I/O models only!

IN†

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

```
PR IN will print the decimal value of the inputs.
```

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

IL

Used to read the decimal equivalent of the 4 bit binary number represented by inputs 1 - 4 collectively. Note the Input 4 is the Most Significant Bit.

```
PR IL will print the decimal value of the standard input set.
```

IH

Used to read the decimal equivalent of the 4 bit binary number represented by inputs 9 - 12 collectively. Note the Input 12 is the Most Significant Bit.

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

O<1-4><9-12>†

Used to set the state of an output.

```
O2=1 will set Output 2 TRUE
```

† O<9-12> are applicable to the MDrivePlus Motion Control Enhanced I/O models only!

OT

Used to set the 8 bit binary equivalent of the decimal number represented by all 8 outputs collectively. Note the Output 12 is the Most Significant Bit.

```
OT=214 will set the outputs to 11010110
```

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

OL

Used to set the 4 bit binary equivalent of the decimal number represented by outputs 1 - 4 collectively. Note the Output 4 is the Most Significant Bit.

```
OT=10 will set the standard outputs to 1010.
```

OH

Used to set the 4 bit binary equivalent of the decimal number represented by outputs 1 - 4 collectively. Note the Output 12 is the Most Significant Bit.

```
OT=13 will set the enhanced outputs to 1101.
```

System Instructions

The following System Instructions will be used frequently.

CP

NOTE: Any program labeled SU will execute upon the power up of the MDrive Motion Control.

The CP Instruction is used to clear Program memory space.

FD

The FD Instruction is used to return the MDrive Motion Control to its factory default state.

<esc>	The ESCAPE key will stop the user program and stop the motor with no decel rate.
<CTRL+C>	CTRL+C will reboot the unit. This includes reloading of the programs stored in non-volatile memory into RAM and executing any programs residing at label SU (Start Up).

Program Instructions

PG

This instruction toggles the MDrive Motion Control 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

The MDrive Motion Control 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 a MDrive Motion Control Instruction, Variable or Flag.
2] The first character must be alpha, the second character may be alpha-numeric.
3] A label is limited to to characters.
4] A program labeled SU will run on power-up

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

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

BR

Used to branch conditionally or unconditionally to a routine.

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

E

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

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 rexecution of program
 BR K1          'Unconditional branch to Program Label K1
E               'End Program
PG              'Switch out of program mode
```

PR

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 rexecution of program
 PR "Position =", P     'Print position
 BR K1                  'Unconditional branch to Program Label K1
E                       'End Program
PG                      'Switch out of program mode
```

VA

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

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

APPENDIX E

MCode Program Samples

This Appendix is made up of several example programs designed to aid the user in discovering the MCode Programming language.

Sample Programs

Move on an Input

```
'Last modified: 01/26/2006
'Purpose: Demonstrate move on input.

'System configuration
S1=0,0          'set IO1 to gen. purpose input, active LOW, sinking
Ms=256          'set µstep resolution to 256 µsteps/step
Vi=200000       'set initial velocity to 200000 steps/sec
Vm=2500000      'set max velocity to 2500000 steps/sec
A=1000000       'set acceleration to 1000000 steps/sec^2
D=A             'set deceleration equal to acceleration
Hc=2            'set motor holding current to 2%
Rc=75           'set motor run current to 75%
P=0             'set position counter to 0


'Main program
PG 1            'enter program mode at address 1
LB Ga           'label program Ga
  P=0           'initialize position counter
LB G1           'label program G1
  CL Kb,I1=1    'call subroutine Kb on input HIGH state
  H 10          'hold program execution 10 msec
  BR G1         'loop to G1

'Subroutine from trigger event
LB Kb           'declare subroutine Kb
  MA 51200      'move to absolute motor position 51200
  H             'suspend pogram execution until motion completes
  MA 0          'move to absolute motor position 0
  H             'suspend pogram execution until motion completes
  RT            'return from subroutine

E               'designate end of program
PG              'exit to immediate mode
```

Change Velocity During A Move

This program will demonstrate ability to change speed during move. MDI does not have abiltity to change speed during point to point move, so we use the Slew command with position trips. End position trip, decel and slew speed determine actual ending position. Program is written to print ending position to serial port 100 times for averaging, expected end position = 102400.

```
'System configuration
Ms=256        'set µstep resolution to 256 µsteps/step
Hc=20         'set motor holding current to 20%
Rc=100        'set motor run current to 75%

'Main program
PG 1
LB Ga         'Program Label Ga sets up local variables and register values
  Vi=20000
  Vm=500000
  A=500000
  D=800000000
  R1=0
  R2=0
LB Gx         'Program label Gx sets up position trips and math functions
  P=0
  Tp=51200,Kb 'set position trip at P=51200
  Te=2
  SL 101200
  H
  H 250
  R1=R1+1            'increment R1
  R2=R2+P            'add position to r2 to set up position calculation
  BR Gx,R1<100       'loop to Gx if R1 indicates less than 100 moves
  R2=R2/100          'after 100 moves have completed, r2 is divided by 100
  PR "Average end pos = ",R2 to obtain and print final Pos.
  E

'Subroutines
LB Kb         'Subroutine called by position trip in Gx which doubles
  SL 202400   'the motor speed
  Tp=102290,Kc
  Te=2
  RT

LB Kc         'Subroutine executed by position trip in Kb
  SL 0
  H
  RT

PG
```

Binary Mask

This program will demonstrate ability to execute various subroutines depending on the binary value of inputs 1-3 while masking all I/O above Input 3.

```
'System configuration
S1=0,1        'setup IO points 1-4, 9-11 as General purpose user
S2=0,1
S3=0,1
S4=0,1
S9=0,1
S10=0,1
S11=0,1
S12=16,0      'set up IO point 12 as a gen. purp. output
Ms=256        'global system variable declarations
Vi=20000
Vm=1000000
A=500000
D=A
Hc=20
Rc=75

'Main program
PG 1
LB Ga
  P=0
LB G1
  R1=In               'capture input combined value to register 1
  R1=R1 & 7           'bits 00000111=7
  CL k0,R1 = 0        'Subroutine calls based on the decimal value of IO
  CL k1,R1 = 1        'points 1-3
  CL k2,R1 = 2
  CL k3,R1 = 3
  CL k4,R1 = 4
  CL k5,R1 = 5
  CL k6,R1 = 6
  CL k7,R1 = 7
  H 10
  BR G1
  E

'Subroutines
LB k0                 'Declare sub K0 executed if R1=0
  PR "Logic 000"
  MR 0*51200
  H
  H 200
  RT

LB k1                 'Declare sub K1 executed if R1=1
  PR "Logic 001"
  MR 1*51200
  H
  H 200
  RT

LB k2                 'Declare sub K2 executed if R1=2
  PR "logic 010"
  MR 2*51200
  H
  H 200
  RT

LB k3                 'Declare sub K3 executed if R1=3
  PR "Logic 011"
  MR 3*51200
  H
  H 200
  RT

LB k4                 'Declare sub K4 executed if R1=4
```

```
  PR "Logic 100"
  MR 4*51200
  H
  H 200
  RT

LB k5                'Declare sub K5 executed if R1=5
  PR "Logic 101"
  MR 5*51200
  H
  H 200
  RT

LB k6                'Declare sub K6 executed if R1=6
  PR "Logic 110"
  MR 6*51200
  H
  H 200
  RT

LB k7                'Declare sub K7 executed if R1=7
  PR "Logic 111"
  MR 7*51200
  H
  H 200
  RT

E
PG
```

Closed Loop

This program illustrates closed loop control with an On Error (OE) routine which will perform math functions on the counters to display the position error.

```
'System configuration
Ms=256          'declare global system variables and flags
Hc=5
Rc=80
Ee=1            'encoder enabled
A=60000
D=A
Vi=2048
Vm=30000
S1=0,0
Sf=15           'encoder stall variables declared
Sm=0
Mt=50
VA Q1           'user variable Q1 declared

'Main program
PG 1
LB Ga           'Ga declares the error call and locally sets the position
  OE k1         'counter to 0 encoder counts
  P=0
LB Gb           'Gb performs ± motions and increments Q1 after each
  MR 51200      'until Q1 reaches 100
  H
  H 500
  MR -51200
  H
  H 500
  IC Q1
  BR Gb,Q1<100
E

'Subroutines
LB k1                   'Sub K1 calculates the position error by dividing
  R3=C1/25              'actual motor steps moved by 25 and subtracts
  R1=R3 - C2            'the number of encoder counts in C2 to determine
  PR "Counts error = ",R1    'the counts error
  PR "Error = ",Er
  Er=0
  H 20
  RT

E
PG
```

User Input into Variables

This program demonstrates the ability to hold up program execution while the user enters multiple variables. Uses registers R1-R3 and a User declared flag for program control.

```
'System configuration
Ms=256        'Global variable declarations
Vi=10000
Vm=50000
A=10000
D=A
Hc=5
Rc=70
P=0
R1=0          'Registers set to 0
R2=0
R3=0
VA X1=0       'User flag X1 declared and set to 0

'Main program
PG 1
LB G1         'Local var-flg settings
  P=0
  X1=0
LB G2         'label for program hold loop
  H 20
  BR G1,X1=0 'command for pg hold loop
LB G3
  X1=0        'reset x1 to 0.
  A=R1        'set A to R1 value
  D=A
  Vm=R2
  MR R3
  H
  H 500
  BR G2
PG
```

Closed Loop with Homing

This program demonstrates the use of the Home to Home Switch Instruction (HM) in closed loop, also there is a move on input routine.

```
Ee=1          'Global variable and flag declarations
Vm=4096
Vi=Vm/50
A=20480
D=A
Hc=50
Rc=50
Mt=50
Sf=20
Sm=0
Db=5
S1=1,0     'Home input
S2=0,0     'Move on input
S3=17,0    'Moving output
S4=19,0    'Fault output

D1=100

'Program
PG 1
LB G1
 H 1000
 PR "C1 ",C1
 PR "C2 ",C2
 Pm=1
 PR "C1 ",C1
 PR "C2 ",C2
 H 5000
 HM 1
 H
 P=0
 LB G2
  BR G2,I2=0
 MR 7186
 H
 PR "p=",P
 BR G2
E
PG
```

Input Trip

This program demonstrates the use input trips

```
'System configuration
Ms=256
Hc=0
Rc=100
D=800000000
Vi=10000
Vm=50000
S1=0,0
S2=16,0
S3=16,0
S4=16,0
O2=1
O3=1
O4=1

'Main program
PG 1
LB Ga
  CL K1  'call to configure 1st input trip
  SL 50000
LB Gb
  H 10
  BR Gb,Mv>0
  R3=R2-R1
  PR "Distance between inputs = ",R3
  H 1000
  PR " "
  BR Ga
  E

'Subroutines
LB K1        'Config for 1st input trip
  S1=0,0
  Ti=1,K2
  Te=1
  RT

LB K2         'Config for 2nd input trip
  R1=Pc
  S1=0,1
  Ti=1,K3
  Te=1
  RT

LB K3         'Sub for 2nd input trip
  R2=Pc
  SL 0

LB K4
  BR K4,Vc=1
  RT

E
PG
```

APPENDIX F

Factors Impacting Motion Commands

Motor Steps

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

Microsteps: (MS)

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

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.

Closed Loop Control With an Encoder

If the Encoder is enabled the Move Commands use different values. The Encoder has 512 lines and yields 2048 counts or counts per revolution. Therefore, the MR and MA Command values are programmed in Encoder counts. One full revolution would be programmed as MR or MA 2048.

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

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

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

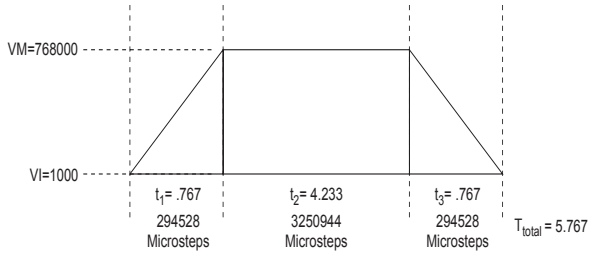


Figure F.1: Trapezoidal Move Profile

move the axis the required distance. This is done with a Trapezoidal Profile as shown below.

Calculating Axis Speed (Velocity)

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

Known Values and Parameters:

VM	768000 Steps/Sec.
VI	1000 Steps/Sec.
A	1000000 Steps/Sec^2.
D	1000000 Steps/Sec^2.
MA/MR	3840000 Microsteps

Determine the Acceleration (A) and Deceleration (D) times (t_1 and t_3). Since the Deceleration (D) value is also 1000000 Steps/Sec. the Deceleration time (t_3) will be the same as the Acceleration time (t_1).

$$(t_1 \text{ and } t_3) = \frac{VM-VI}{A} \text{ or } \frac{768000 - 1000}{1000000} = 0.767 \text{ Seconds}$$

Determine the distance (Steps) traveled in t_1 or t_3.

$$\text{Distance} = \frac{VM+VI}{2} \times t_1 \text{ or } \frac{768000+1000}{2} \times 0.767 = 294911 \text{ Steps}$$

Determine the t_2 time.

The t_2 time is calculated by dividing the remainder of MA/MR by VM.

The remainder of MA/MR = MA/MR - (t_1 steps + t_3 steps) or 3840000 - 589056 = 3250944.

$$t_2 = \frac{3250944}{768000} = 4.233 \text{ Seconds}$$

Determine the total time. ($t_1 + t_2 + t_3$) or (0.767 + 4.233 + 0.767) = 5.767 Seconds

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

Note that the average speed includes the Acceleration and Deceleration. The maximum axis speed attained is approximately 90 inches/minute.

$$\frac{768000}{51200} \times 0.1 \times 60 = 90 \text{ IPM}$$

Calculating Rotary Movement

Assume the MS is set to 256. You are using the motor to drive a shaft with a timing belt and pulley arrangement. As shown below, the pulley is 1" in diameter and the shaft pulley is 2.5" in diameter. You must turn the shaft 270°.

- The shaft will rotate 1 full revolution for every 2.5 revolutions of the motor.
- 270° is 0.75 of a revolution.
- 0.75 x 2.5 = 1.875 motor revolutions to turn the shaft 270°.
- If 51200 Microsteps is 1 motor revolution, then the device must be programmed to move 96000 Microsteps (51200 x 1.875).

You may also do many of the calculations in reverse to calculate motor moves to meet a required move of your device. A linear or rotational move as well as speed may be translated into an MCode command.

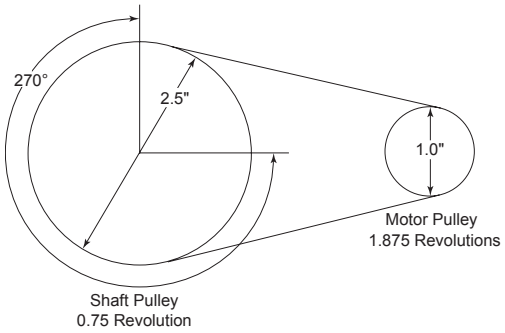


Figure F.2: Rotary Drive Example 1

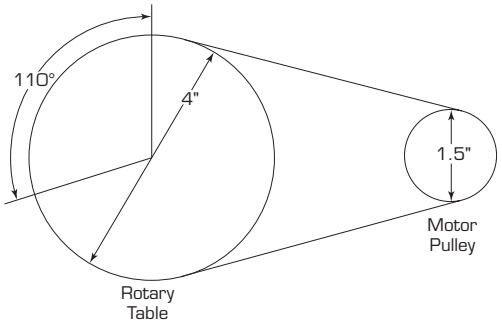


Figure F.3: Rotary Drive Example 2

In the example above, the belt driven Rotary Table must be turned 110° at 3 RPM. How should the device be set up?

Bear in mind that all the numbers are approximate due to rounding.

Mechanical ratio between the motor and the rotary table is 2.666:1. That is, the motor must rotate 2.666 revolutions for the table to rotate 1 revolution and the table will rotate 2.666 times slower than the motor.

- In order to move the table 110° the motor must move 293.3°.

  110 × 2.666 = 293.3°

- If 51200 steps = 1 revolution then 1° = 142.222 steps.

$$\frac{51200}{360} = 142.222 \text{ steps}$$

- The MCode device must be programmed to move 41713 steps to rotate 293.3°

  142.222 steps × 293.3° = 41713 steps

- In order to rotate the table at 3 RPM the motor must turn at 8 RPM.

  3 RPM × 2.666 = 8 RPM

- If you were to set VM at 51200 and MS set at 256 the motor will rotate 1 full revolution (51200 steps) in 1 second or 1 RPS. In order to rotate at 8 RPM, the motor must rotate at 0.13333 RPS.

$$\frac{8}{60} = 0.133333 \text{ RPS}$$

- In order to rotate at 0.13333 RPS the VM must be set at 6827 steps/sec.

  51200 × 0.133333 = VM 6827

Note: These numbers will vary slightly depending on Acceleration and Deceleration rates.

Programming with the Optional Encoder Enabled

An optional 512 line magnetic encoder is available. When the Encoder is enabled (EE=1) the programming also changes. All motion must now be programmed by the encoder counts. The Encoder operates in the "Quadrature" format. That is, there are four Encoder counts for each Encoder line or 2048 counts per revolution (512 × 4 = 2048). (See Figure below.) If you were to program motion using the MR (Move Relative) or MA (Move Absolute) commands the motor would rotate a distance equal to the encoder counts.

Example:

A programmed move of 7168 counts would result in the motor rotating 3.5 revolutions at a velocity controlled by VM.

(7168 ÷ 2048 = 3.5 revolutions)

If you were to program motion using the SL (Slew) command the motor would rotate at a "counts per second" rate based on the programmed value.

Example:

An SL (Slew) rate of 7168 counts was programed. The motor will rotate at 7168 counts/sec., 3.5 RPS, or 210 RPM.

(7168 ÷ 2048 = 3.5 RPS × 60 = 210 RPM)

When the Encoder is enabled, the parameters are also changed to be compatible with the 2048 counts.

The Encoder Enabled defaults are:

VM	30720 Counts/Sec.
VI	40 Counts/Sec.
A	40000 Counts/Sec
D	40000 Counts/Sec.
MS	256 (Default for Encoder Mode. Cannot be changed.)

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.

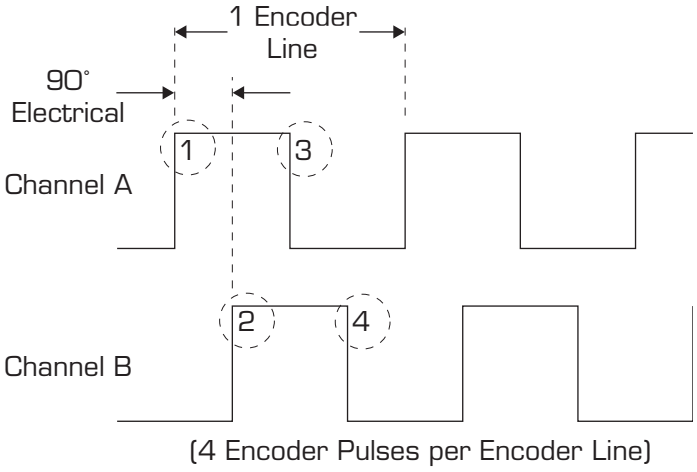


Figure F.4: Quadrature Encoder Counts

Several Variables work in conjunction with Encoder Enable (EE). They are:

DB	Encoder Deadband
SF	The Stall Factor Variable
SM	The Stall Detection Mode
ST	Stall Flag
PM	Position Maintenance

EE - Encoder Enabled

When the Encoder is enabled, all motion is “closed loop”. That is, motion steps are delivered from the MCode device to the motor which turns the encoder. The encoder sends counts back to the drive to complete the motion. If you programmed a move of 2048 counts, the device would output an appropriate number of Microsteps provided the Stall Factor (SF) value or other fault is not encountered. If no faults were encountered, the device would output the full amount of Microsteps. Depending on which variables were set, the driver would then wait until the position (plus or minus the Encoder Deadband) was read and confirmed.

DB - Encoder Deadband

The Encoder Deadband is a Variable that is set in Encoder Counts. Motion will be deemed complete when the Encoder Counts are within ± the Deadband variable. With DB=5 the motion of 2048 counts would be complete between 2043 and 2053 counts.

SF - Stall Factor

The Stall Factor is a Variable which is entered in Encoder Counts. The Stall Factor is active only in the EE=1 mode. The Stall Factor might be compared to the “following error” or “lag error” of a servo drive. The Stall Factor is triggered by the number of steps output from the device to the motor as compared to the number of counts returned by the encoder. The comparison should always be within the value of the Stall Factor, otherwise a fault will occur and the Stall Flag (ST) will be set. If the Stall Detection Mode is active (SM=0), the motion will be stopped.

Example:

A Stall Factor of 30 counts (SF=30) is programmed. A motion command of 2048 counts is programmed. The device reaches a mechanical bind at 2000 counts. The device will keep outputting steps equivalent to 2030 counts (present position plus the SF value) and then the Stall Flag (ST) will be set. The motor will be stopped if the Stall Detection Mode (SM=0) is active.

SM - Stall Detection Mode

The Stall Detection Mode can be programmed to stop the device (SM=0) or to allow the device to continue (SM=1) when the Stall Factor (SF) is reached. Whether SM is active or not, the Stall Flag will always be set when the SF is encountered.

ST - Stall Flag

The Stall Flag will be set any time the SF is reached regardless of the state of the Stall Detection Mode (SM). If the Stall Flag is set, the user must reset it to zero (0).

PM - Position Maintenance

Position Maintenance (PM) is active only after the motion has completed. Position Maintenance is used to maintain position when there might be an external force on the drive. If Position Maintenance is enabled (PM=1) and the Stall Detection Mode is enabled (SM=0), the motor will be driven back to its final position if it was forced out of position provided the Stall Factor (SF) was not reached.

If Position Maintenance is enabled (PM=1) and the Stall Detection Mode is disabled (SM=1), the motor will be driven back to its final position if it was forced out of position regardless of whether the Stall Factor (SF) was reached or not.

There are three other variables, although not directly connected 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 value, the motor may hunt and never satisfy Position Check.

n o

t h e

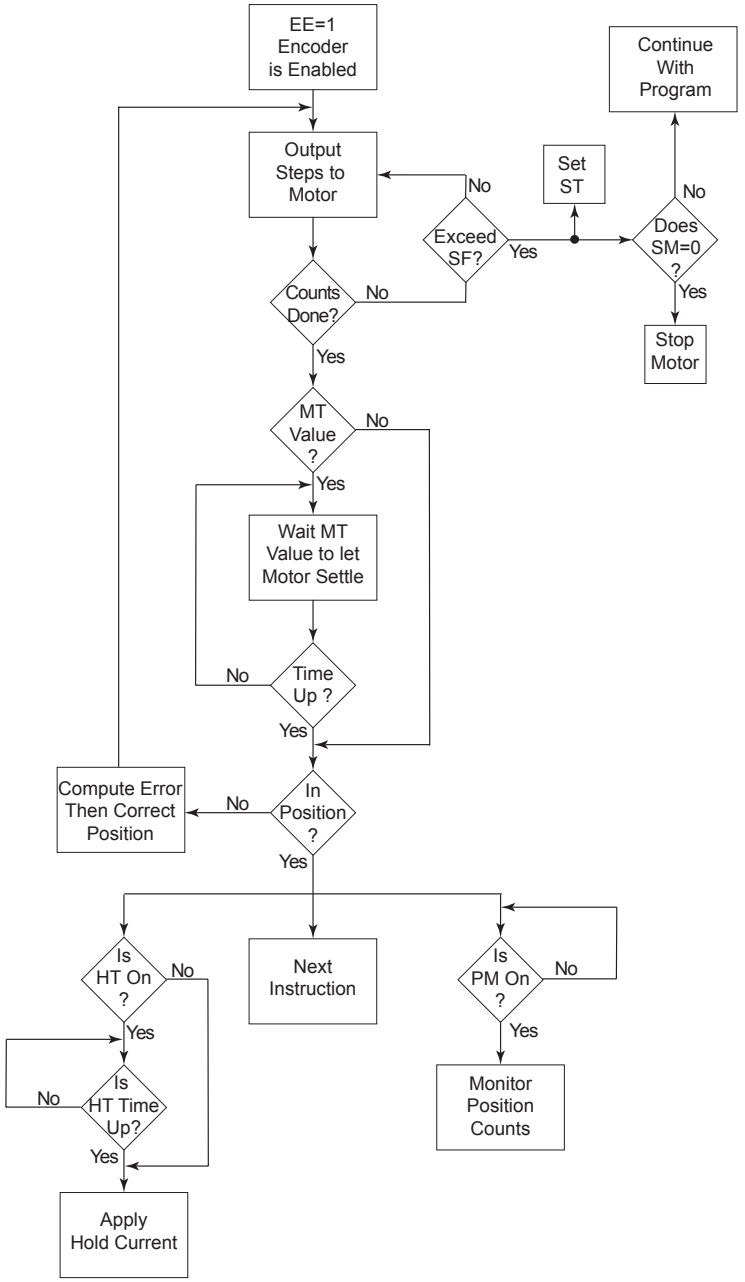


Figure F.5: EE=1 Flowchart

Page Intentionally Left Blank

APPENDIX G

MForce PWM Configuration (PW)

⚠ CAUTION

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

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

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

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

Description:

This variable defines the parameter settings for the PWM and should not be used unless motion problems such as smoothness and positional accuracy are experienced.

The PW variable consists of four components: <mask>, <period>, <sfrq> and either <checksum>, if writing, or <boot_writes_remaining> if reading using the PR keyword.

PWM Mask <mask> Parameter

The PWM mask signal prevents the premature end of the forward period caused by switching transients when the motor phase current is at low levels. Adjusting this value can impact the zero-crossing performance of the motor. If experiencing the "tick" which is inherit in stepper motor systems, this may be minimized or eliminated by adjusting this value. The range of this value is 0 to 255d and will be entered as a decimal value.

The Mask will act as a filter on the PWM signal to allow time for any ringing in the output circuitry to settle.

This range represents a 8-bit Hex value that specifies the Bridge Reverse Measure Time (REVTM) and the Minimum Bridge Forward On Time (FORTM) ranging from 600 nS to 3.4 µS each (see table and diagram below). Typically these values would be balanced. The table below shows the decimal value for each time.

Note that these are typical values and the currents may be unbalanced to fine tune the motor performance.

The default value for this parameter is 204 (0xCC), which represents a Reverse Measure Time and Minimum Forward On Time of 2.5 µS.

Reverse Measure Time/Minimum Forward On Time							
Hex	Time	Hex	Time	Hex	Time	Hex	Time
0x0	600 ns	0x4	1.0 µs	0x8	1.6 µs	0xC	2.5 µs
0x1	700 ns	0x5	1.1 µs	0x9	1.8 µs	0xD	2.8 µs
0x2	800 ns	0x6	1.2 µs	0xA	2.0 µs	0xE	3.1 µs
0x3	900 ns	0x7	1.4 µs	0xB	2.2 µs	0xF	3.4 µs

Table G.1: PWM Mask Settings

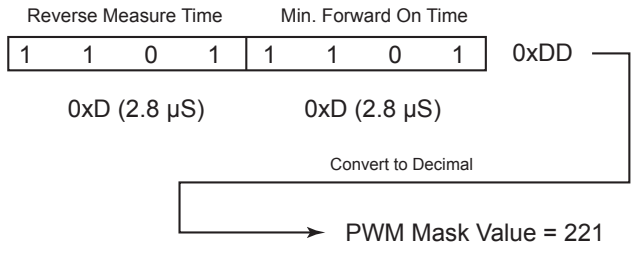


Figure G.1: PWM Mask Bits

Typical PWM Mask Settings (Currents Balanced)							
Mask (hex)	Mask (dec)	REVTM	FORTM	Mask (hex)	Mask (dec)	REVTM	FORTM
0x00	0	600 ns	600 ns	0x88	135	1.6 µs	1.6 µs
0x11	17	700 ns	700 ns	0x99	153	1.8 µs	1.8 µs
0x22	34	800 ns	800 ns	0xAA	170	2.0 µs	2.0 µs
0x33	51	900 ns	900 ns	0xBB	187	2.2 µs	2.2 µs
0x44	68	1.0 µs	1.0 µs	0xCC	204	2.5 µs	2.5 µs
0x55	85	1.1 µs	1.1 µs	0xDD	221	2.8 µs	2.8 µs
0x66	102	1.2 µs	1.2 µs	0xEE	238	3.1 µs	3.1 µs
0x77	119	1.4 µs	1.4 µs	0xFF	255	3.4 µs	3.4 µs

Table G.2: Typical PWM Mask Settings

Maximum PWM Duty Cycle (%) <period> Parameter

This parameter sets the maximum duty cycle as a percentage of the bridge PWM oscillator period. The range for this parameter is 0 to 95%. Entries above 95% will generate an out of range error (Error 21) and will not allow the setting to be written to the boot.

The default value for this parameter is 95%.

PWM Frequency <sfreq> Parameter

The PWM Frequency Parameter sets the initial and maximum frequencies for the PWM. As with the MASK parameter, the PWM Frequency is a two part 8-bit hex number which is entered as a decimal value ranging from 0 to 255.

The default for this 170 (0xAA) with an initial PWM Frequency of 20 kHz and a Maximum of 60 kHz.

Maximum PWM Frequency (kHz)							
Hex	Freq.	Hex	Freq	Hex	Freq	Hex	Freq
0x0	40	0x4	48	0x8	56	0xC	64
0x1	42	0x5	50	0x9	58	0xD	66
0x2	44	0x6	52	0xA	60	0xE	68
0x3	46	0x7	54	0xB	62	0xF	70

Initial PWM Frequency (kHz)							
Hex	Freq.	Hex	Freq	Hex	Freq	Hex	Freq
0x0	10	0x4	14	0x8	18	0xC	22
0x1	11	0x5	15	0x9	19	0xD	23
0x2	12	0x6	16	0xA	20	0xE	24
0x3	13	0x7	17	0xB	21	0xF	25

Table G.3: Maximum and Initial PWM Frequency

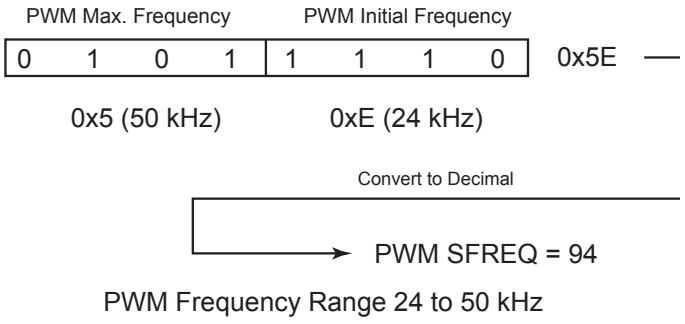


Figure G.2 PWM Frequency Range

PWM Checksum <chksum> Parameter (Boot Write Only)

> **⚠ CAUTION**
>
> This parameter should be left blank for testing parameters. Only insert a checksum if you have verified that these parameters cause the motor to perform as desired as the presence of the checksum WILL write the PWM settings to the boot, Limited Writes (8) are available.

The PWM Checksum parameter is only used on the write cycle to write the PWM parameters to the boot. To calculate the checksum (a number from 0 to 255), add the Mask, Period and Frequency

If the sum is ≥ 512, subtract 512, the remainder is the checksum.

If the sum is < 512 subtract 255, the remainder is the checksum.

Example:

Mask 136

Period 90

Freq. 170

Sum = 396

396 - 256 = 140

Checksum = 140

Boot Writes Remaining <boot_writes_remaining> Parameter (Read Only)

This parameter will be the fourth parameter shown during a read of the PWM settings (PR PW).

The range is from 8 to 0 and represents the number of times remaining that the PWM settings can be written to the boot of the drive.

Example PWM Settings By Motor Specifications

The following settings are based upon IMS settings per motor specifications and should serve as a baseline to work from with regard to the manufacturer specifications of the motor being utilized. Note that these are example settings ONLY!

Example PW Settings								
Frame Size	Stack Size	Phase Current (A_{RMS})	Phase Resistance (Ω)	Phase Inductance (mH)	MASK <mask>	Duty Cycle <period>	Frequency <sfreq>	Checksum <chksum>
14	Single	0.75	4.30	4	102	90	170	106
17	Single	1.5	1.30	2.1	136	90	170	140
	Double	1.5	2.10	5.0	136	90	170	140
	Triple	1.5	2.00	3.85	136	90	170	140
23	Single	2.4	0.95	2.4	136	90	170	140
	Double	2.4	1.20	4.0	136	90	170	140
	Triple	2.4	1.50	5.4	136	90	170	140
34	Single	6.3	0.25	1.6	168	95	170	177
	Double	6.3	0.35	3.3	220	95	170	229
	Triple	6.3	0.50	6.6	253	95	170	6
MForce Default		—	—	—	204	95	170	—

Usage Example

```
        PW=102,90,150          'set pwm settings, DO NOT WRITE TO BOOT!
        PW=136,90,170,140      'set pwm settings, write to boot

        PR PW                  'read PWM
Response = 136,90,170,7        'last integer notes 7 boot writes remaining
```

Page Intentionally Left Blank

WARRANTY

TWENTY-FOUR (24) MONTH LIMITED WARRANTY

Intelligent Motion Systems, Inc. (“IMS”), warrants only to the purchaser of the Product from IMS (the “Customer”) that the product purchased from IMS (the “Product”) will be free from defects in materials and workmanship under the normal use and service for which the Product was designed for a period of 24 months from the date of purchase of the Product by the Customer. Customer’s exclusive remedy under this Limited Warranty shall be the repair or replacement, at Company’s sole option, of the Product, or any part of the Product, determined by IMS to be defective. In order to exercise its warranty rights, Customer must notify Company in accordance with the instructions described under the heading “Obtaining Warranty Service.”

This Limited Warranty does not extend to any Product damaged by reason of alteration, accident, abuse, neglect or misuse or improper or inadequate handling; improper or inadequate wiring utilized or installed in connection with the Product; installation, operation or use of the Product not made in strict accordance with the specifications and written instructions provided by IMS; use of the Product for any purpose other than those for which it was designed; ordinary wear and tear; disasters or Acts of God; unauthorized attachments, alterations or modifications to the Product; the misuse or failure of any item or equipment connected to the Product not supplied by IMS; improper maintenance or repair of the Product; or any other reason or event not caused by IMS.

IMS HEREBY DISCLAIMS ALL OTHER WARRANTIES, WHETHER WRITTEN OR ORAL, EXPRESS OR IMPLIED BY LAW OR OTHERWISE, INCLUDING WITHOUT LIMITATION, **ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE**. CUSTOMER’S SOLE REMEDY FOR ANY DEFECTIVE PRODUCT WILL BE AS STATED ABOVE, AND IN NO EVENT WILL THE IMS BE LIABLE FOR INCIDENTAL, CONSEQUENTIAL, SPECIAL OR INDIRECT DAMAGES IN CONNECTION WITH THE PRODUCT.

This Limited Warranty shall be void if the Customer fails to comply with all of the terms set forth in this Limited Warranty. This Limited Warranty is the sole warranty offered by IMS with respect to the Product. IMS does not assume any other liability in connection with the sale of the Product. No representative of IMS is authorized to extend this Limited Warranty or to change it in any manner whatsoever. No warranty applies to any party other than the original Customer.

IMS and its directors, officers, employees, subsidiaries and affiliates shall not be liable for any damages arising from any loss of equipment, loss or distortion of data, loss of time, loss or destruction of software or other property, loss of production or profits, overhead costs, claims of third parties, labor or materials, penalties or liquidated damages or punitive damages, whatsoever, whether based upon breach of warranty, breach of contract, negligence, strict liability or any other legal theory, or other losses or expenses incurred by the Customer or any third party.

OBTAINING WARRANTY SERVICE

Warranty service may obtained by a distributor, if the Product was purchased from IMS by a distributor, or by the Customer directly from IMS, if the Product was purchased directly from IMS. Prior to returning the Product for service, a Returned Material Authorization (RMA) number must be obtained. Complete the form at http://www.imshome.com/rma.html after which an RMA Authorization Form with RMA number will then be faxed to you. Any questions, contact IMS Customer Service (860) 295-6102.

Include a copy of the RMA Authorization Form, contact name and address, and any additional notes regarding the Product failure with shipment. Return Product in its original packaging, or packaged so it is protected against electrostatic discharge or physical damage in transit. The RMA number MUST appear on the box or packing slip. Send Product to: Intelligent Motion Systems, Inc., 370 N. Main Street, Marlborough, CT 06447.

Customer shall prepay shipping changes for Products returned to IMS for warranty service and IMS shall pay for return of Products to Customer by ground transportation. However, Customer shall pay all shipping charges, duties and taxes for Products returned to IMS from outside the United States.

U.S.A. SALES OFFICES
Eastern Region
Tel. 862 208-9742 - Fax 973 661-1275
e-mail: jroake@imshome.com
Central Region
Tel. 260 402-6016 - Fax 419 858-0375
e-mail: dwaksman@imshome.com
Western Region
Tel. 602 578-7201
e-mail: dweisenberger@imshome.com

IMS ASIA PACIFIC OFFICE
30 Raffles Pl., 23-00 Chevron House, Singapore 048622
Tel. +65/6233/6846 - Fax +65/6233/5044
e-mail: wllee@imshome.com

IMS EUROPEAN SALES MANAGEMENT
4 Quai Des Etroits
69005 Lyon, France
Tel. +33/4 7256 5113 - Fax +33/4 7838 1537
e-mail: bmartinez@imshome.com

IMS UK LTD.
Sanderson Centre, 15 Lees Lane
Gosport, Hampshire PO12 3UL
Tel. +44/0 2392-520775 - Fax +44/0 2392-502559
e-mail: mcheckley@imshome.com

TECHNICAL SUPPORT
Tel. +00 (1) 860 295-6102 - Fax +00 (1) 860 295-6107
e-mail: etech@imshome.com

Intelligent Motion Systems, Inc.
370 North Main Street, P.O. Box 457
Marlborough, CT 06447 - U.S.A.
Tel. +00 (1) 860 295-6102 - Fax +00 (1) 860 295-6107
e-mail: info@imshome.com
http: //www.imshome.com

IMS™ INTELLIGENT MOTION SYSTEMS, INC.

by Schneider Electric