TP (Trip on position)

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

Mnemonic	TP
Function	Trip on position
Type	Variable
Description	Sets up an event (trip) for the specified position. There are two parameters for the TP variable. The first specifies the position which will cause the event. The second specifies the subroutine that should be executed when the position is detected. 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.
Syntax	TP=<position>,<address/label>
Usage	Program/Immediate, Read/Write
Code Example	`TP=650000,K9  'exe sub K9 when position = 650000` `TE=2  'Re-enable trip`
Notes	Note that TP will always use motor counts regardless of the encoder enabled state (EE=1)
Related	P, TI, PC, S13, CW

TQ (Set torque)

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

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

TR (Trip on relative position)

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

Mnemonic	TR
Function	Trip on relative position
Type	Variable
Description	Sets up an event (trip) for the specified relative position. There are three parameters for the TR variable. The first specifies the position which will cause the event. The second specifies the subroutine that should be executed when the position is detected, if no subroutine address or label is specified then the High Speed Trip Output will activate. The Trip subroutine must use a RETURN (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 after repeating the number of relative trips specified. TE must be re-enabled in the main program prior to the next series of Trip on Relative if it is to be repeated. For exampl, if TR=10000,0,25, the Output (S13) will trip 25 times in succession at 100,000 counts relative to the last position. Following these 25 trips the trip must be re-enabled (TE=16). Trips should be set BEFORE motion commands in the program.
Syntax	TR=<±dist>,<address/label>, <repeat>
Usage	Program/Immediate, Read/Write

Code Example	`TR=650000,0,0  'Activate trip out when at 650000`
	`'counts relative`
	`TE=16  'Re-enable trip`
	
	`TR=512000,K2,27  'Run Sub K2 when at 512000 rel,`
	`'repeat 27 times`
	`TE=16  'Re-enable trip`

Notes	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
Related	P, TI, PC, S13, CW

TS (Set torque speed)

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

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

TT (Trip on position)

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

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

UG (Upgrade firmware)

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

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

UV (Read user variables)

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

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

V (Read velocity)

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

Mnemonic	V
Function	Read axis velocity
Type	Variable
Description	The velocity variable is used in conjunction with the PR (print) instruction to read the current velocity of the axis in counts per second. This variable can also be used with the BR and CL instructions to set a condition based upon a velocity.
Syntax	PR V BR <address/label>, V=<velocity> BR <address/label>, V=<velocity>
Usage	Program/Immediate, Read

Code Example	`PR V  'Read the velocity`
	`CL Ka, V=20000  'Execute sub Ka when velocity is`
	`'20000/steps sec in the + direction`
	`CL Kb, V=-20000 'Execute sub Kb when velocity is`
	`'-20000/steps sec in the - irection`

Notes	Note that V is signed. When using an Hybrid product V will not return an accurate value when Hybrid make up is active. In torque mode V will return 0.
Related	VI, VM, SL, MA, MR

VA (Create user variable)

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

Mnemonic	VA
Function	Create user variable name
Type	Instruction
Description	The VA instruction creates a user variable with a 1 or 2 character name. Can optionally set value assigned to that variable. 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.
Syntax	VA <char><char>=<value>
Usage	Program/Immediate, Read/Write
Code Example	`VA Q3 'Create user Variable Q3` `VA Q3=20000 'Create user Variable Q3, set to 20000`
Related	UV

VC (Velocity changing)

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

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

VF (Torque velocity filter)

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

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

VI (Initial velocity)

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

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

VM (Maximum velocity)

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

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

VR (Firmware version)

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

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

WT (Warning temperature)

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

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

6 Supporting Software

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

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

6.1 IMS Terminal

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

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

6.1.2 Features

General features

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

Program editor window

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

Terminal emulator window

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

6.1.3 Installing IMS Terminal

System requirements

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

Installation

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

6.1.4 Screen and menu overview

IMS Terminal main screen

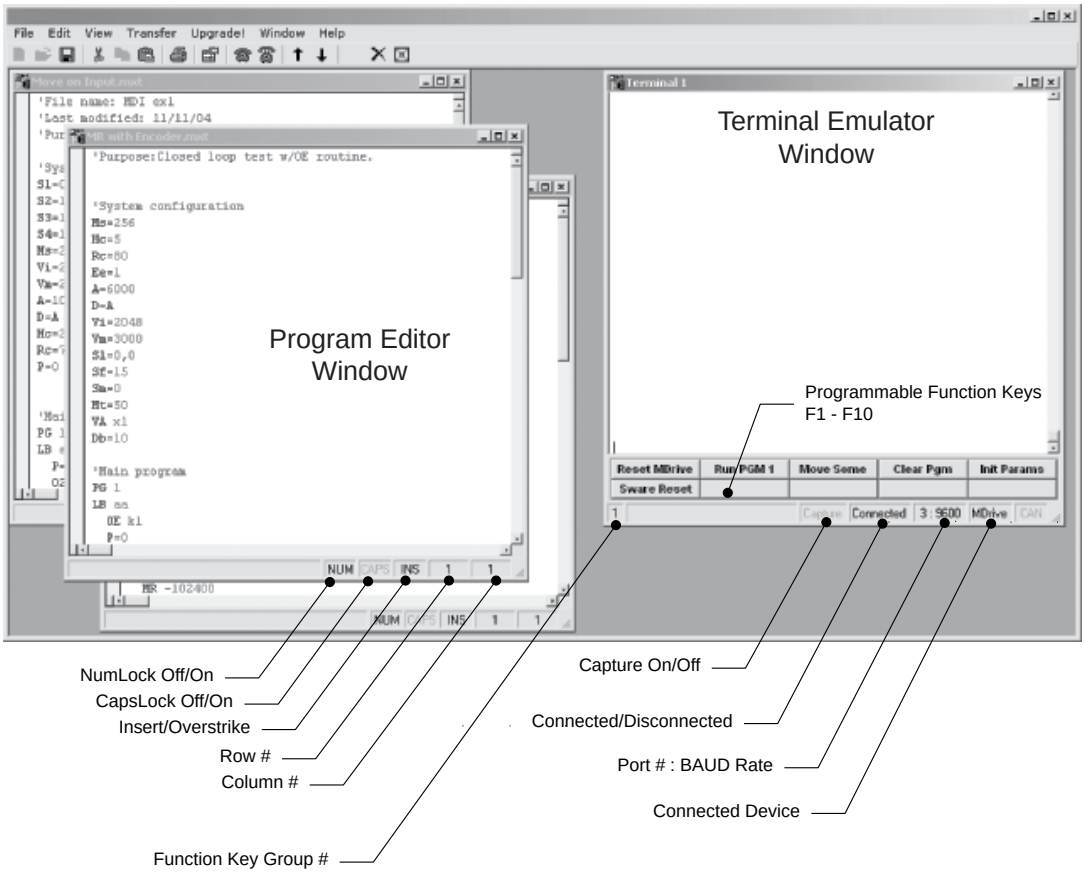


Figure 6.1 IMS Terminal main screen

6.4.2 The menu bar structure and operation

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

File menu functions

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

6.4.3 Edit menu operation

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

Preferences

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

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

Open preferences

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

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

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

Save preferences

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

View menu operation

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

- New edit window
- New terminal window

The operation of these options are covered later in this section

Transfer menu operation

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

Upload

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

Download

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

Capture

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

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

- Pause capture
- Stop capture

These will function as described by their label.

Upgrade menu operation

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

Window menu operation

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

6.1.5 Configuring IMS Terminal

Configuring program editor format preferences

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

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

To open the program editor preferences dialog:

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

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

Configuring terminal window format preferences

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

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

Configuring communications settings

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

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

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

The sign-on message should appear:

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

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

6.1.6 Troubleshooting communications

Troubleshooting the communications converter

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

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

Check the following connections:

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

6.1.7 Configuring function keys

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

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

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

To open the function key dialog:

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

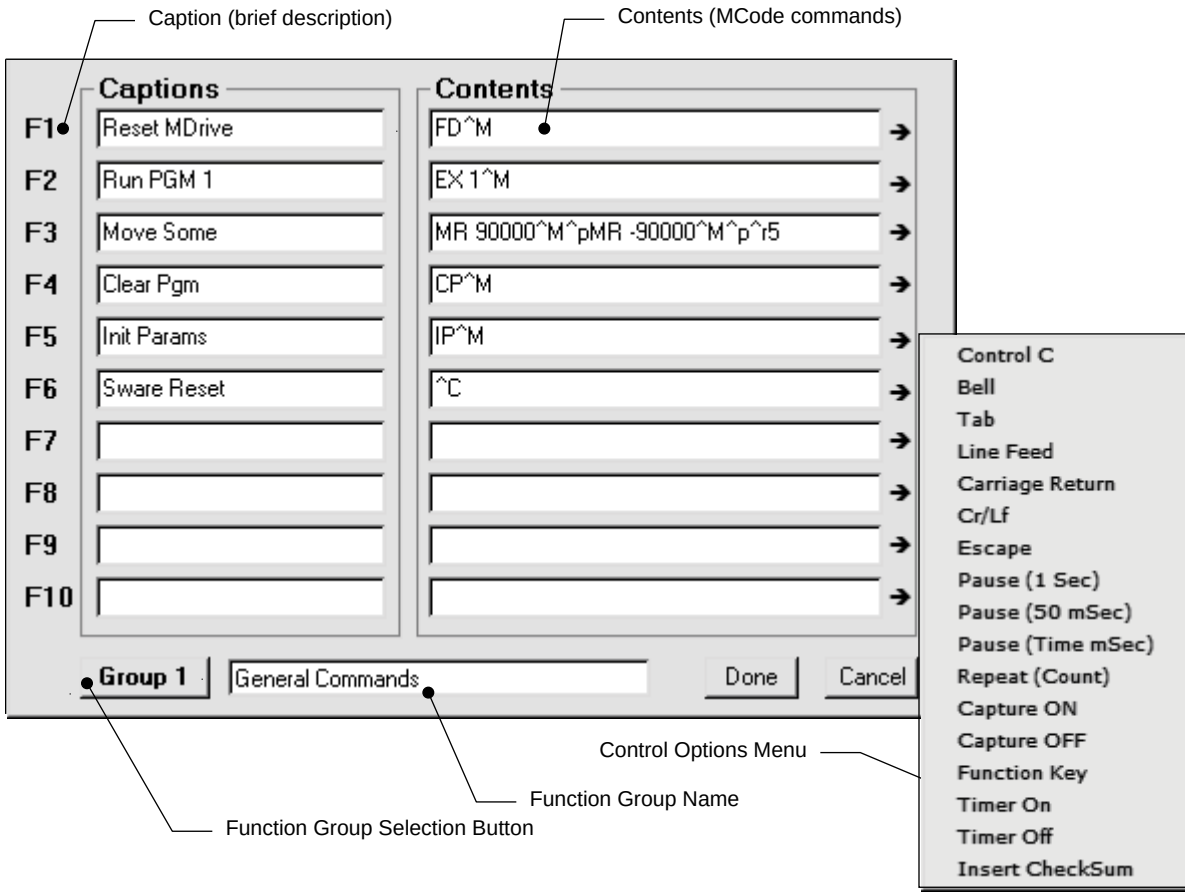


Figure 6.2 Function key setup dialog

Function key control commands

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

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

Table 6.1 Function key control commands

6.1.8 Creating, downloading and uploading programs

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

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

Creating a new program

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

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

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

Downloading a program to the device

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

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

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

To download

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

Save the program. (Always save your programs!)

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

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

Uploading a Program

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

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

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

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

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

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

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

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

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

6.1.9 Program troubleshooting using IMS Terminal

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

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

Single step mode

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

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

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

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

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

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

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

Trace mode

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

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

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

3) Each line can now be analyzed.

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

The capture function

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

Enable the capture function

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

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

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

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

6.1.10 Upgrading firmware

Before upgrading the MCode firmware

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

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

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

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

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

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

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

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

Upgrading the firmware

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

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

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

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

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

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

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

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

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

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

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

- Click “Next”

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

6.2 TCP/IP Configuration Utility

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

6.2.1 Installation

System requirements

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

Installation

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

6.2.2 Screen overview

Configuration SEM Port (Terminal)

Figure 6.3 TCP/IP Configuration Utility

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

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

6.2.3 Configuration

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

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

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

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

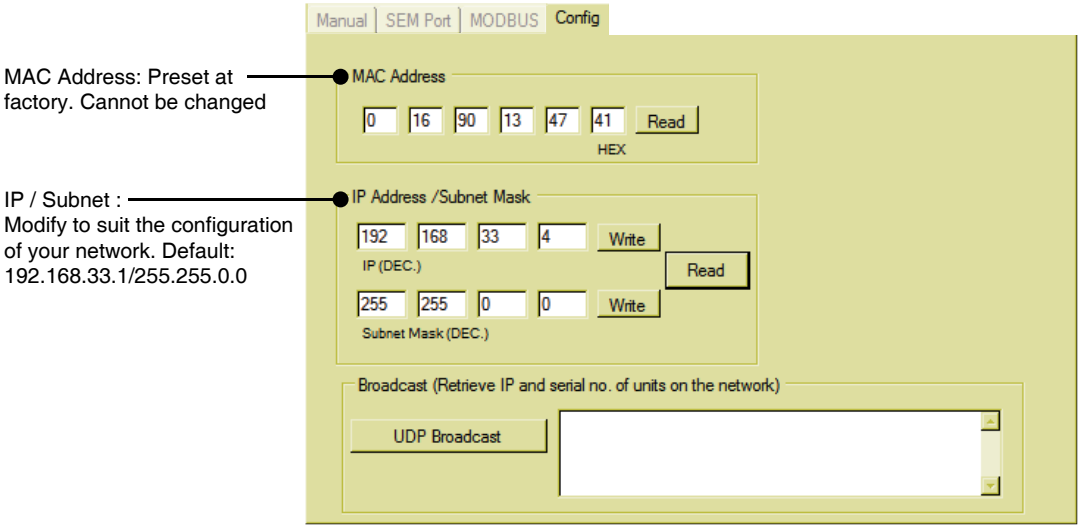


Figure 6.4: Configuration tab

6.2.4 SEM Port tab

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

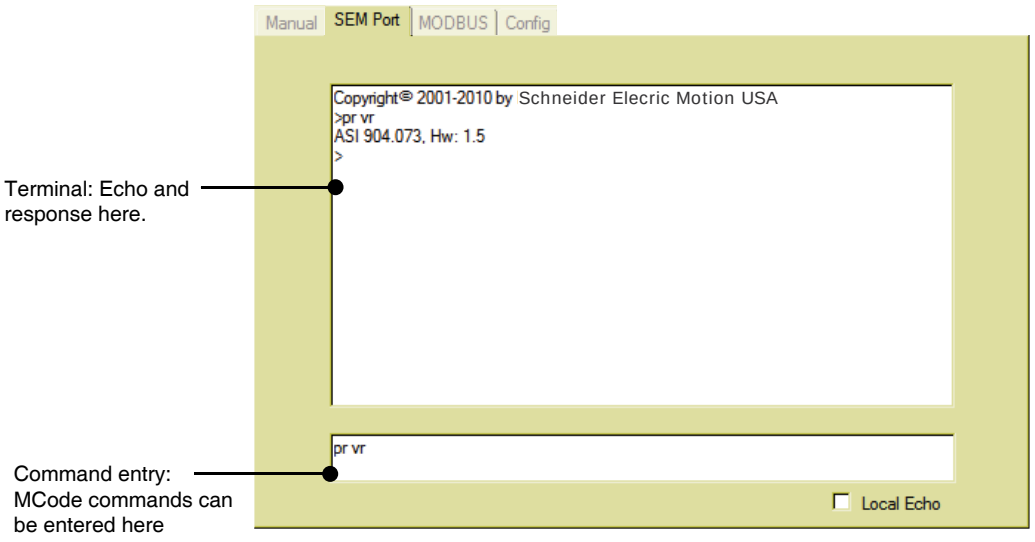


Figure 6.5: SEM Port tab

6.3 Upgrading the Ethernet controller firmware

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

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

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

Requirements

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

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

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

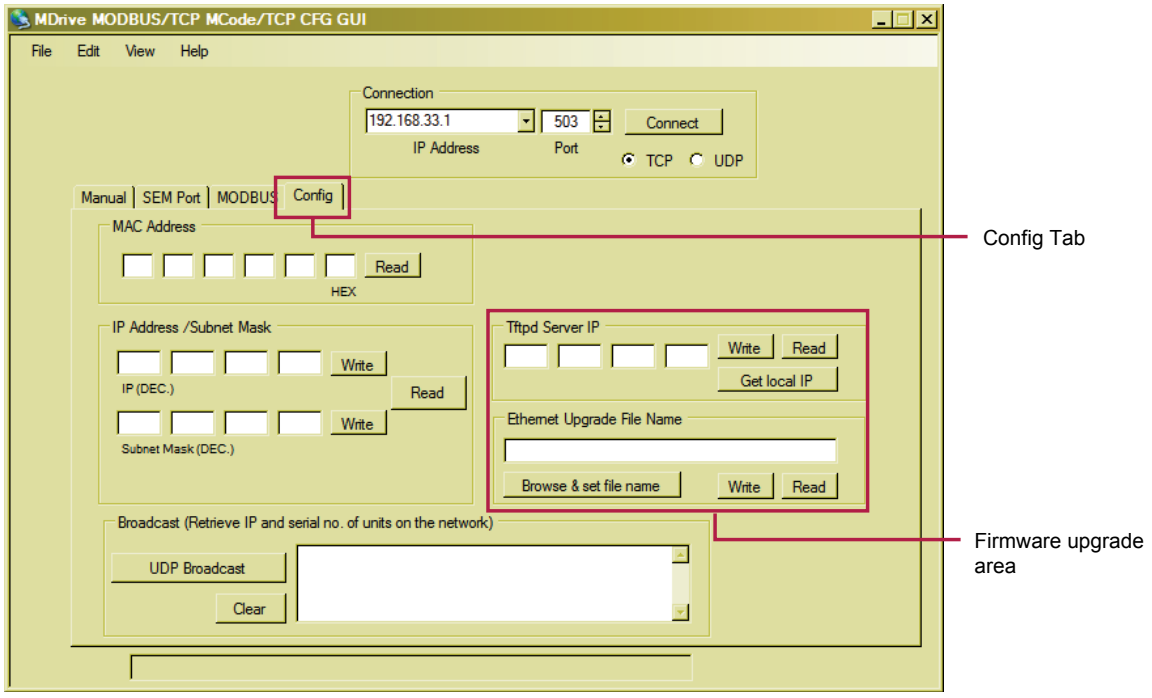


Figure 6.6: TCP/IP Config tab

6.3.1 To begin

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

6.3.2 Set the Tftpd Server IP

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

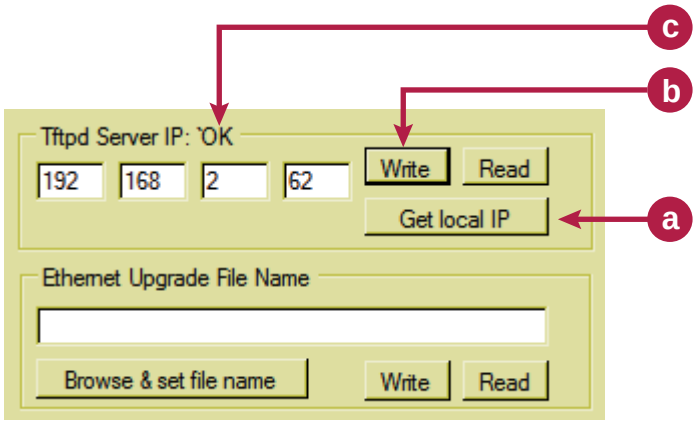


Figure 6.7: Setting the Tftpd Server IP

6.3.3 Set the Ethernet upgrade file name

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

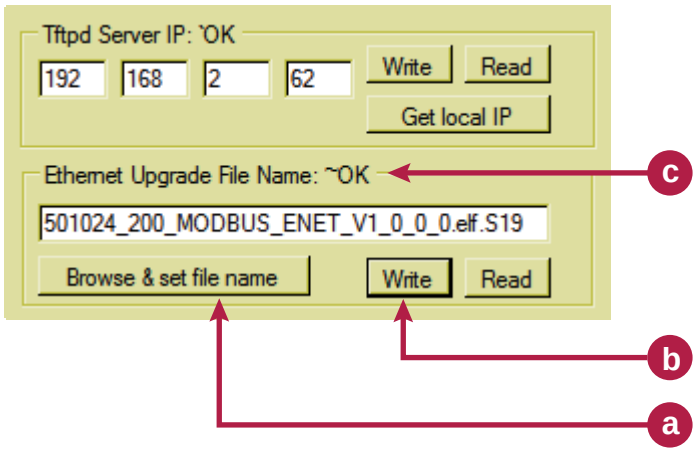


Figure 6.8: Setting the upgrade file name

6.3.4 Enter upgrade mode

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

   2956102

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

6.3.5 Complete upgrade process

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

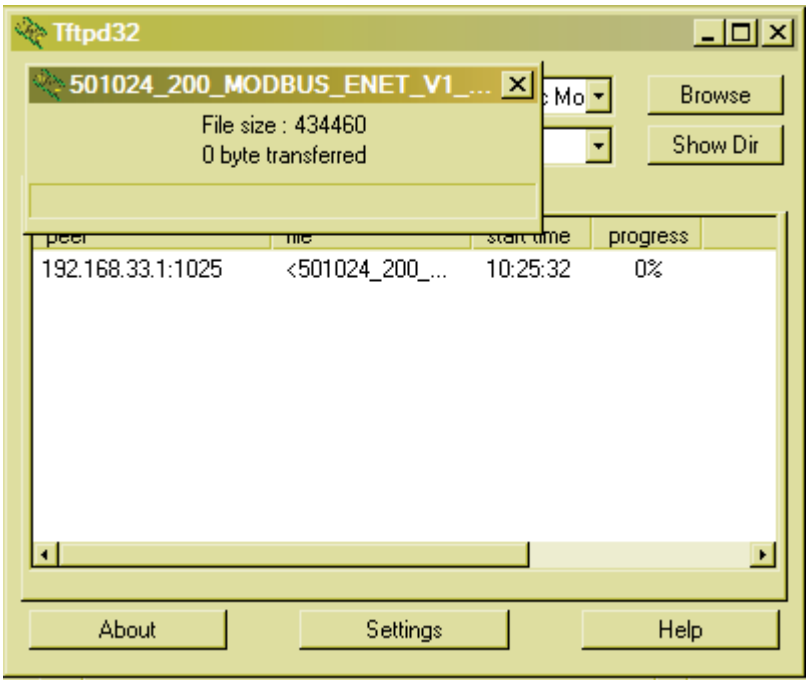


Figure 6.9: Firmware upgrading

7 Programming and application notes

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

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

7.1 Party mode communications

The following communication formats, used by MCode comptible devices.

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

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

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

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

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

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

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

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

7.1.1 Response to Echo Mode

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

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

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

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

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

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

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

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

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

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

7.1.2 Using Check Sum

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

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

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

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

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

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

7.1.3 Immediate party mode sample codes

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

Move device A, B or C 10000 steps

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

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

7.2 Programming the I/O

7.2.1 I/O availability per device type

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

MDrive, MForce

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

MDrive, MForce, (Plus² expanded features)

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

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

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

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

MDrive Hybrid

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

7.2.2 Active states defined

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

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

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

Examples

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

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

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

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

7.2.3 Digital input functions

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

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

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

Example:

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

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

The following table lists the programmable input functions.

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

Table 7.5 Digital input functions

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

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

Table 7.6 Clock and high speed input functions

7.2.4 Digital output functions

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

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

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

Examples

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

Output Functions

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

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

*Only applies to Hybrid products.

Table 7.7 Digital output functions

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

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

Table 7.8 Clock and high speed output functions

7.2.5 Programmable I/O usage examples

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

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

Input Interface Example - Switch Input (Sinking Input)

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

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

Code Sample

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

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

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

Input interface example - switch input example (sourcing input)

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

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

Code Sample

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

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

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

Output interface example (sinking output)

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

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

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

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

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

Output interface example (sourcing output)

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

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

Code Sample

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

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

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

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

Reading inputs as a group example

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

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

Standard I/O

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

Expanded I/O

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

Interfacing outputs as a group example

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

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

Standard I/O

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

Expanded I/O

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

7.2.6 Dedicated digital I/O usage

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

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

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

Step/Direction

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

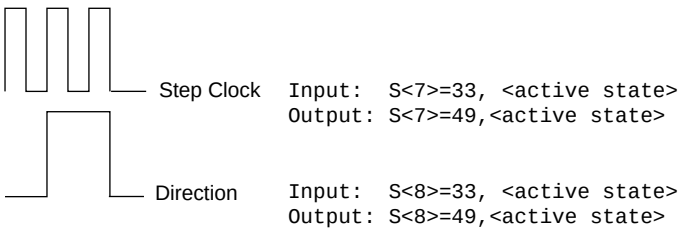
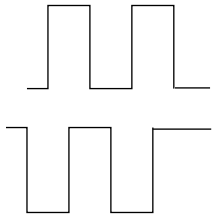


Figure 7.1 Step/direction I/O type & configuration

Quadrature

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



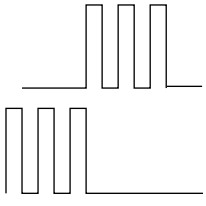

```
Input:  S<7>=34, <active state>
Output: S<7>=50,<active state>

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

Figure 7.2 Quadrature I/O configuration

CW/CCW

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




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

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

Figure 7.3 CW/CCW I/O configuration

7.2.7 Analog input usage

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

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

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

Sample Usage

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

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

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

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

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

7.3 Factors impacting motion commands

7.3.1 Motor steps

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

7.3.2 Microsteps: (MS)

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

7.3.3 Move Command

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

7.3.4 Closed loop control with an encoder

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

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

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

7.3.5 Linear movement

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

7.5 inches divided by 0.1 inches = 75 motor revolutions.

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

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

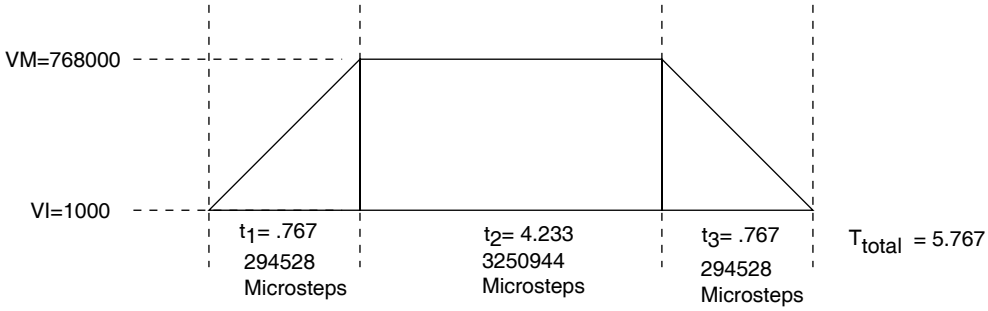


Figure 7.4 Trapezoidal move profile

Calculating axis speed (velocity)

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

Known Values and Parameters:

VM.....768000 Steps/Sec.

VI.....1000 Steps/Sec.

A.....1000000 Steps/Sec2.

D .....1000000 Steps/Sec2.

MA/MR.....3840000 Microsteps

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

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

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

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

= 294911 steps

Determine the t2 time.

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

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

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

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

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

Note that the average speed includes the Acceleration and Deceleration. The maximum axis speed attained is approximately 90 inches/minute.

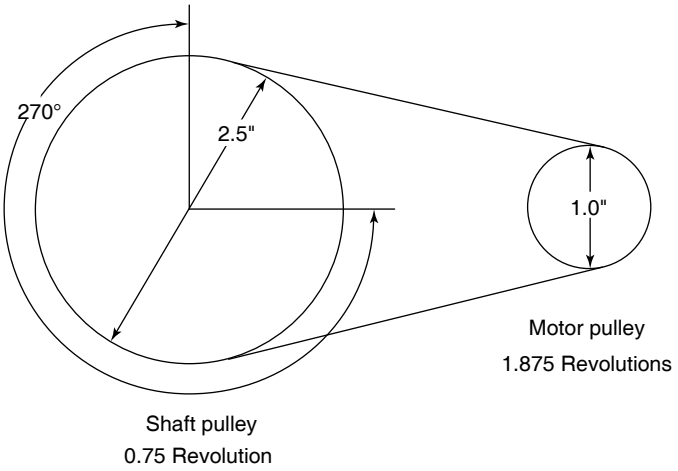
$$\frac{768000}{51200} \times 0.1 \times 60 \textbf{ = 90 IPM}$$

7.3.6 Calculating rotary movement

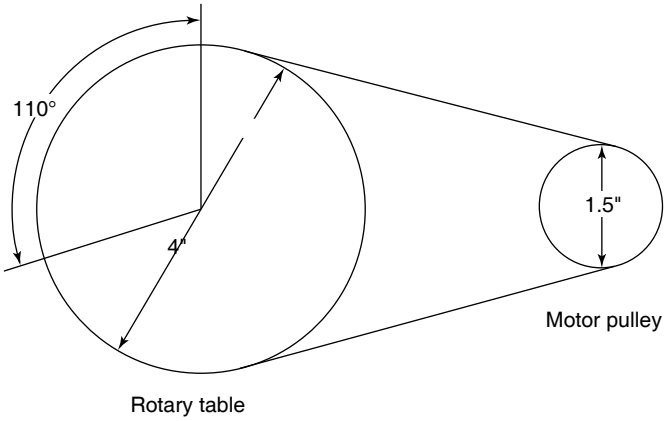
Assume the MS is set to 256. You are using the motor to drive a shaft with a timing belt and pulley arrangement. As shown below, the pulley is 1” in diameter and the shaft pulley is 2.5” in diameter. You must turn the shaft 270°.

- The shaft will rotate 1 full revolution for every 2.5 revolutions of the motor.
- 270° is 0.75 of a revolution.
- 0.75 x 2.5 = 1.875 motor revolutions to turn the shaft 270°.
- If 51200 Microsteps is 1 motor revolution, then the device must be programmed to move 96000 Microsteps (51200 x 1.875).

You may also do many of the calculations in reverse to calculate motor moves to meet a required move of your device. A linear or rotational move as well as speed may be translated into an MCode command.



Rotary drive example 1



Rotary drive example 2

Figure 7.5 Rotary examples

In the example above, the belt driven rotary table must be turned 110° at 3 RPM. How should the device be set up?

Bear in mind that all the numbers are approximate due to rounding.

Mechanical ratio between the motor and the rotary table is 2.666:1. That is, the motor must rotate 2.666 revolutions for the table to rotate 1 revolution and the table will rotate 2.666 times slower than the motor.

In order to move the table 110° the motor must move 293.3°.

`110 x 2.66` **`= 293.3º`**

If 51200 steps = 1 revolution then 1° = 142.222 steps.

$$\frac{51200}{360} = \textbf{142.222 steps}$$

The MCode device must be programmed to move 41713 steps to rotate 293.3°.

`142.222 steps x 293.3º` **`= 41713 steps`**

In order to rotate the table at 3 RPM the motor must turn at 8 RPM.

`3 RPM x 2.666` **`= 8 RPM`**

If you were to set VM at 51200 and MS set at 256 the motor will rotate 1 full revolution (51200 steps) in 1 second or 1 RPS. In order to rotate at 8 RPM, the motor must rotate at 0.13333 RPS.

$$\frac{8}{60} = \textbf{0.133333 RPS}$$

In order to rotate at 0.13333 RPS the VM must be set at 6827 steps/sec.

`51200 x 0.133333` **`= 6827`**

`VM = 6827`

Note: These numbers will vary slightly depending on Acceleration and Deceleration rates.

7.3.7 Programming with the optional encoder enabled

An optional 512 line magnetic encoder is available. When the Encoder is enabled (EE=1) the programming also changes. All motion must now be programmed by the encoder counts. The Encoder operates in the "Quadrature" format. That is, there are four Encoder counts for each Encoder line or 2048 counts per revolution (512 × 4 = 2048). (See Figure below.) If you were to program motion using the MR (Move Relative) or MA (Move Absolute) commands the motor would rotate a distance equal to the encoder counts.

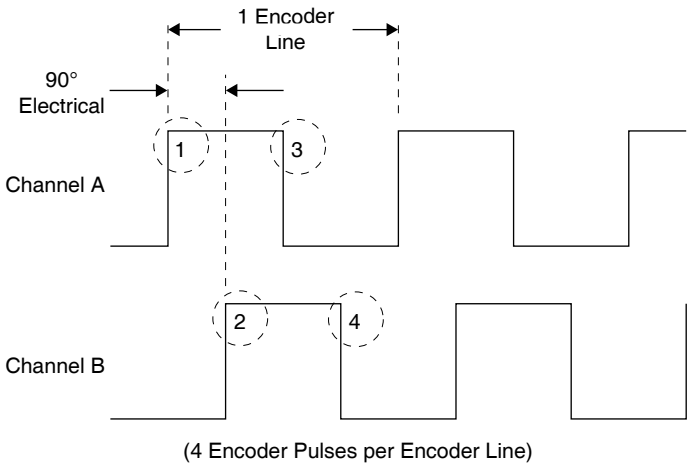


Figure 7.6 Encoder waveform

Example:

A programmed move of 7168 counts would result in the motor rotating 3.5 revolutions at a velocity controlled by VM.

```
7168 ÷ 2048 = 3.5 revolutions
```

If you were to program motion using the SL (Slew) command the motor would rotate at a "counts per second" rate based on the programmed value.

Example:

An SL (Slew) rate of 7168 counts was programed. The motor will rotate at 7168 counts/sec., 3.5 RPS, or 210 RPM.

```
7168 ÷ 2048 = 3.5 RPS × 60 = 210 RPM
```

When the Encoder is enabled, the parameters are also changed to be compatible with the 2048 counts.

The Encoder Enabled defaults are:

VM.....30720 Counts/Sec.

VI.....40 Counts/Sec.

A.....40000 Counts/Sec

D.....40000 Counts/Sec.

MS.....256 (default for encoder mode.)

To enable the encoder the program syntax is <EE=n> where n is a zero (0) or a one (1). The default is zero (0) which is encoder disabled. To enable the encoder, program EE=1.

Any motion will now be programmed in encoder counts. You can calculate the distance or velocity you need in a similar manner as done previously only with different factors.

Note: The microstep select is defaulted and locked at 256 in the encoder mode to ensure stable, high resolution.

Several Variables work in conjunction with Encoder Enable (EE). They are:

DB .....Encoder Deadband

SF.....The Stall Factor Variable

SM..... The Stall Detection Mode

ST.....Stall Flag

PM..... Position Maintenance

EE .....Encoder Enabled

When the encoder is enabled, all motion is “closed loop”. That is, motion steps are delivered from the MCode device to the motor which turns the encoder. The encoder sends counts back to the drive to complete the motion. If you programmed a move of 2048 counts, the device would output an appropriate number of microsteps provided the stall factor (SF) value or other fault is not encountered. If no faults were encountered, the device would output the full amount of microsteps. Depending on which variables were set, the driver would then wait until the position (plus or minus the encoder deadband) was read and confirmed.

DB - Encoder Deadband

The Encoder Deadband is a Variable that is set in Encoder Counts. Motion will be deemed complete when the Encoder Counts are within ± the Deadband variable. With DB=5 the motion of 2048 counts would be complete between 2043 and 2053 counts.

SF - Stall Factor

The Stall Factor is a Variable which is entered in Encoder Counts. The Stall Factor is active only in the EE=1 mode. The Stall Factor might be compared to the “following error” or “lag error” of a servo drive. The Stall Factor is triggered by the number of steps output from the device to the motor as compared to the number of counts returned by the encoder. The comparison should always be within the value of the Stall Factor, otherwise a fault will occur and the Stall Flag (ST) will be set. If the Stall Detection Mode is active (SM=0), the motion will be stopped.

Example:

A Stall Factor of 30 counts (SF=30) is programmed. A motion command of 2048 counts is programmed. The device reaches a mechanical bind at 2000 counts. The device will keep outputting steps equivalent to 2030 counts (present position plus the SF value) and then the Stall Flag (ST) will be set. The motor will be stopped if the Stall Detection Mode (SM=0) is active.

SM - Stall Detection Mode

The Stall Detection Mode can be programmed to stop the device (SM=0) or to allow the device to continue (SM=1) when the Stall Factor (SF) is reached. Whether SM is active or not, the Stall Flag will always be set when the SF is encountered.

ST - Stall Flag

The Stall Flag will be set any time the SF is reached regardless of the state of the Stall Detection Mode (SM). If the Stall Flag is set, the user must reset it to zero (0).

PM - Position Maintenance

Position maintenance (PM) is active only after the motion has completed. Position maintenance is used to maintain position when there might be an external force on the drive. If position maintenance is enabled (PM=1) and the stall detection mode is enabled (SM=0), the motor will be driven back to its final position if it was forced out of position provided the stall factor (SF) was not reached.

If position maintenance is enabled (PM=1) and the stall detection mode is disabled (SM=1), the motor will be driven back to its final position if it was forced out of position regardless of whether the stall factor (SF) was reached or not.

There are three other variables, although not directly conned to EE, that do affect the overall operation when in encoder mode, they are:

HC..... Motor Hold Current

HT..... Motor Hold Current Delay Time

MT.....Motor Settling Delay Time

HC.....Hold Current

When motion is complete, the device will switch from motor run current (RC) to motor hold current (HC). The hold current is set at a lower percentage than the run current (rc). However, the hold current must be sufficient to overcome an outside force such as driving a vertical slide which maintains a load on the motor at all times. Actual hold current

values will vary depending on the application and the load on the motor when it is at rest.

HT - Motor Hold Current Delay Time

The motor hold current delay time (HT) is a variable that delays the change from run current (RC) to hold current (HC) at the end of a move. The end of the move is triggered by the device when it has completed outputting the correct number of steps. Depending on the application, including velocity, deceleration, load and inertia, the device may lag behind a few counts. The ht will allow the device to finish its move before applying the lower HC.

MT - Motor Settling Delay Time

A stepping motor may ring or oscillate in minuscule amounts at the completion of a move until it satisfies the target position. The amount of this “ringing” is dependent on the application including velocity, deceleration, inertia, friction and load. The motor settling delay time (MT) allows the motor to stop “ringing” before checking the position count. If the device tried to check the position count during this ringing, it would assume a position error and try to correct an already moving motor and possibly cause ringing of a larger magnitude and longevity. Typically, the MT is set between 50 and 100 milliseconds. It is recommended that there is always a Motor Settling Time programmed any time you are in EE=1 mode.

Note: If MT has no value, the motor may hunt and never satisfy the position check.

7.4 MForce PWM configuration

⚠ CAUTION

This variable is only applicable to the mforce product line and is used to tune the PWM Settiings to optimize the current control of the MForce driver. It should not be used unless erratic motion or positional accuracy problems are being experienced.

Note that there are other factors that could contribute to these problems. Ensure that all wiring conforms to the guidelines in the mforce hardware manual.

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

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

7.4.1 Description:

This variable defines the parameter settings for the PWM and should not be used unless motion problems such as smoothness and positional accuracy are experienced.

The PW variable consists of four components: <mask>, <period>, <sfrq> and either <checksum>, if writing, or <boot_writes_remaining> if reading using the PR keyword.

7.4.2 PWM Mask <mask> Parameter

The PWM mask signal prevents the premature end of the forward period caused by switching transients when the motor phase current is at low levels. Adjusting this value can impact the zero-crossing performance of the motor. If experiencing the “tick” which is inherit in stepper motor systems, this may be minimized or eliminated by adjusting this value. The range of this value is 0 to 255d and will be entered as a decimal value.

The Mask will act as a filter on the PWM signal to allow time for any ringing in the output circuitry to settle.

This range represents a 8-bit Hex value that specifies the Bridge Reverse Measure Time (REVTM) and the Minimum Bridge Forward On Time (FORTM) ranging from 600 nS to 3.4 µS each (see table and diagram below). Typically these values would be balanced. The table below shows the decimal value for each time.

Note that these are typical values and the currents may be unbalanced to fine tune the motor performance.

The default value for this parameter is 204 (0xCC), which represents a Reverse Measure Time and Minimum Forward On Time of 2.5 µS.

Hex	Time	Hex	Time	Hex	Time	Hex	Time
0x0	600 ns	0x4	1.0 µs	0x8	1.6 µs	0xC	2.5 µs
0x1	700 ns	0x5	1.1 µs	0x9	1.8 µs	0xD	2.8 µs
0x2	800 ns	0x6	1.2 µs	0xA	2.0 µs	0xE	3.1 µs
0x3	900 ns	0x7	1.4 µs	0xB	2.2 µs	0xF	3.4 µs

Table 7.8 PWM Mask Settings

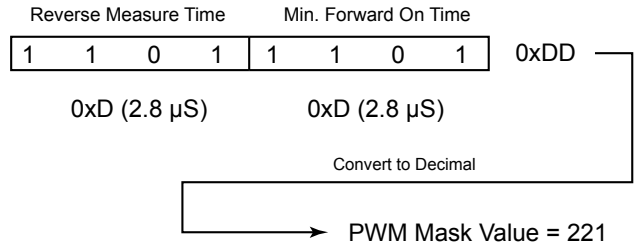


Figure 7.7 PWM mask bits

Mask (hex)	Mask (dec)	REVTM	FORTM	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 7.9 Typical PWM Mask Settings

7.4.3 Maximum PWM duty cycle (%) <period> parameter

This parameter sets the maximum duty cycle as a percentage of the bridge PWM oscillator period. The range for this parameter is 0 to 95%. Entries above 95% will generate an out of range error (Error 21) and will not allow the setting to be written to the boot.

The default value for this parameter is 95%.

7.4.4 PWM frequency <sfreq> parameter

The PWM Frequency Parameter sets the initial and maximum frequencies for the PWM. As with the MASK parameter, the PWM Frequency is a two part 8-bit hex number which is entered as a decimal value ranging from 0 to 255.

The default for this 170 (0xAA) with an initial PWM Frequency of 20 kHz and a Maximum of 60 kHz.

Hex	Freq.	Hex	Freq	Hex	Freq	Hex	Freq
0x0	40	0x4	48	0x8	56	0xC	64
0x1	42	0x5	50	0x9	58	0xD	66
0x2	44	0x6	52	0xA	60	0xE	68
0x3	46	0x7	54	0xB	62	0xF	70

Table 7.10 Maximum PWM frequency

Hex	Freq.	Hex	Freq	Hex	Freq	Hex	Freq
0x0	10	0x4	14	0x8	18	0xC	22
0x1	11	0x5	15	0x9	19	0xD	23
0x2	12	0x6	16	0xA	20	0xE	24
0x3	13	0x7	17	0xB	21	0xF	25

Table 7.11 Initial PWM frequency

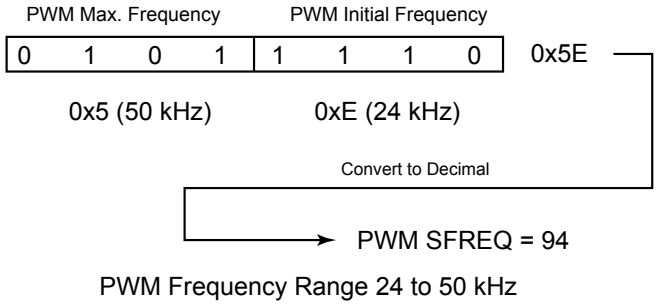


Figure 7.8 PWM frequency range

7.4.5 PWM checksum <chksum> parameter (boot write only)

⚠ CAUTION

This parameter should be left blank for testing parameters. Only insert a checksum if you have verified that these parameters cause the motor to perform as desired as the presence of the checksum WILL write the PWM settings to the boot, Limited Writes (8) are available.

The PWM checksum parameter is only used on the write cycle to write the PWM parameters to the boot. To write the PWM parameters to an Mforce a checksum must be sent along with the parameters.

PW = mask, period, frequency, checksum

To compute the checksum use the following steps:

1. Add the decimal values for mask, period, and frequency. Total = mask + period + frequency
2. Use A, B, or C to calculate checksum
   A. If the total is less than or equal to 256:
   (Total <= 256) checksum = 256 - Total
   B. If the total is greater than 256 but less than or equal to 512:
   (256 < Total <= 512) checksum = 512 - Total
   C. If the total is greater than 512, then:
   (Total > 512)checksum = 768 - Total

Examples

Add 102 + 90 + 10 = 202. The total is less than 256 so use A. to calculate checksum.

Checksum = 256 - 202
Checksum = 54
PW = 102,90,10,54

Add 238 + 95 + 170 = 503. The total is greater than 256 but less than or equal to 512 so use B. to calculate checksum.

Checksum = 512 - 503
Checksum = 9
PW = 238,95,170,9

Add 255 + 90 + 170 = 515. The total is greater than 512 so use C. to calculate checksum.

Checksum = 768 - 515
Checksum = 253
PW = 255,90,170,253

7.4.6 Boot writes remaining <boot_writes_remaining> parameter (read only)

This parameter will be the fourth parameter shown during a read of the PWM settings (PR PW).

The range is from 8 to 0 and represents the number of times remaining that the PWM settings can be written to the boot of the drive.

7.4.7 Example PWM settings by motor specifications

The following settings are based upon IMS settings per motor specifications and should serve as a baseline to work from with regard to the manufacturer specifications of the motor being utilized. Note that these are example settings ONLY!

Frame Size	Stack Size	Phase Current (A_{RMS})	Phase Resistance (Ω)	Phase Inductance (mH)	MASK <mask>	Duty Cycle <period>	Frequency <sfreq>	Checksum <chksum>
14	Single	0.75	4.30	4	102	90	170	250
17	Single	1.5	1.30	2.1	136	90	170	116
	Double	1.5	2.10	5.0	136	90	170	116
	Triple	1.5	2.00	3.85	136	90	170	116
23	Single	2.4	0.95	2.4	136	90	170	116
	Double	2.4	1.20	4.0	136	90	170	116
	Triple	2.4	1.50	5.4	136	90	170	116
34	Single	6.3	0.25	1.6	168	95	170	79
	Double	6.3	0.35	3.3	220	95	170	27
	Triple	6.3	0.50	6.6	253	95	170	250
MForce Default		—	—	—	204	95	170	—

Table 7.9 Example PWM settings

Usage Example

```
PW=102,90,150      'set pwm, DO NOT WRITE TO BOOT!
PW=136,90,170,116 'set pwm settings, write to boot

PR PW                        'read PWM
Response = 136,90,170,7    'last integer notes 7 boot
                             'writes remaining
```

This page intentionally left blank

8 Sample programs

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

8.1 Move on an input

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

'System configuration
S1=0,0     'set IO1 to gen. purpose input, active LOW,
           'sinking
Ms=256     'set µstep resolution to 256 µsteps/step
Vi=200000  'set initial velocity to 200000 steps/sec
Vm=2500000 'set max velocity to 2500000 steps/sec
A=1000000  'set acceleration to 1000000 steps/sec2
D=A                'set deceleration equal to acceleration
Hc=2               'set motor holding current to 2%
Rc=75              'set motor run current to 75%
P=0                'set position counter to 0


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

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

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

8.2 Change velocity during a move

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

```
'System configuration
Ms=256      'set µstep resolution to 256 µsteps/step

Hc=20       'set motor holding current to 20%
Rc=100      'set motor run current to 75%

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

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

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

PG
```

8.3 Binary mask

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

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

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

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

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

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

```
  RT

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

LB k4                    'Declare sub K4 executed if R1=4
  PR "Logic 100"
  MR 4*51200
  H
  H 200
  RT

LB k5                    'Declare sub K5 executed if R1=5
  PR "Logic 101"
  MR 5*51200
  H
  H 200
  RT

LB k6                    'Declare sub K6 executed if R1=6
  PR "Logic 110"
  MR 6*51200
  H
  H 200
  RT

LB k7                    'Declare sub K7 executed if R1=7
  PR "Logic 111"
  MR 7*51200
  H
  H 200
  RT

E
PG
```

8.4 Closed Loop

This program illustrates closed loop control with an On Error (OE) routine which will perform math functions on the counters to display the position error.

```
'System configuration
Ms=256     'declare global system variables and flags
Hc=5
Rc=80
Ee=1               'encoder enabled
A=60000
D=A
Vi=2048
Vm=30000
S1=0,0
Sf=15              'encoder stall variables declared
Sm=0
Mt=50
VA Q1              'user variable Q1 declared

'Main program
PG 1
LB Ga              'Ga declares the error call and locally
                   'sets the position
  OE k1    'counter to 0 encoder counts
  P=0
LB Gb              'Gb performs ± motions and increments
                   'Q1 after each
  MR 51200 'until Q1 reaches 100
  H
  H 500
  MR -51200
  H
  H 500
  IC Q1
  BR Gb,Q1<100
E

'Subroutines
LB k1              'Sub K1 calculates the position error
                   'by dividing
  R3=C1/25         'actual motor steps moved by 25 and
                   'subtracts
  R1=R3 - C2       'the number of encoder counts in C2 to
                   'determine
  PR "Counts error = ",R1 'the counts error
  PR "Error = ",Er
  Er=0
  H 20
  RT

E
PG
```

8.5 User input into variables

This program demonstrates the ability to hold up program execution while the user enters multiple variables. Uses registers R1-R3 and a user declared flag for program control.

```
'System configuration
Ms=256     'Global variable declarations
Vi=10000
Vm=50000
A=10000
D=A
Hc=5
Rc=70
P=0
R1=0               'Registers set to 0
R2=0
R3=0
VA X1=0    'User flag X1 declared and set to 0

'Main program
PG 1
LB G1              'Local var-flg settings
  P=0
  X1=0
LB G2              'label for program hold loop
  H 20
  BR G1,X1=0 'command for pg hold loop
LB G3
  X1=0     'reset x1 to 0.
  A=R1     'set A to R1 value
  D=A
  Vm=R2
  MR R3
  H
  H 500
  BR G2
PG
```

8.6 Closed loop with homing

This program demonstrates the use of the home to home switch instruction (HM) in closed loop, also there is a move on input routine.

```
Ee=1                'Global variable and flag declarations
Vm=4096
Vi=Vm/50
A=20480
D=A
Hc=50
Rc=50
Mt=50
Sf=20
Sm=0
Db=5
S1=1,0     'Home input
S2=0,0     'Move on input
S3=17,0    'Moving output
S4=19,0    'Fault output

D1=100

'Program
PG 1
LB G1
 H 1000
 PR "C1 ",C1
 PR "C2 ",C2
 Pm=1
 PR "C1 ",C1
 PR "C2 ",C2
 H 5000
 HM 1
 H
 P=0
 LB G2
  BR G2,I2=0
 MR 7186
 H
 PR "p=",P
 BR G2
E
PG
```

8.7 Input trip

This program demonstrates the use input trips

```
'System configuration
Ms=256
Hc=0
Rc=100
D=800000000
Vi=10000
Vm=50000
S1=0,0
S2=16,0
S3=16,0
S4=16,0
O2=1
O3=1
O4=1

'Main program
PG 1
LB Ga
  CL K1  'call to configure 1st input trip
  SL 50000
LB Gb
  H 10
  BR Gb,Mv>0
  R3=R2-R1
  PR "Distance between inputs = ",R3
  H 1000
  PR " "
  BR Ga
  E

'Subroutines
LB K1         'Config for 1st input trip
  S1=0,0
  Ti=1,K2
  Te=1
  RT

LB K2          'Config for 2nd input trip
  R1=Pc
  S1=0,1
  Ti=1,K3
  Te=1
  RT

LB K3          'Sub for 2nd input trip
  R2=Pc
  SL 0

LB K4
  BR K4,Vc=1
  RT

E
PG
```

9 Error codes

A question mark <?> Displayed as a cursor indicates an error. To determine what the error is, type <pr er> in the terminal window. The device will respond with an error number displayed in the terminal window. The error number may then be referenced to this list.

0	No Error

9.1 I/O errors

6	An I/O is already set to this type. Applies to non-General Purpose I/O.
8	Tried to set an I/O to an incorrect I/O type.
9	Tried to write to I/O set as Input or is “TYPED”.
10	Illegal I/O number.
11	Incorrect CLOCK type.
12	Illegal Trip / Capture

9.2 Data errors

20	Tried to set unknown variable or flag. Trying to set an undefined variable of flag. Also could be a typo.
21	Tried to set an incorrect value. Many variables have a range such as the Run Current (RC) which is 1 to 100%. As an example, you cannot set the RC to 110%.
22	VI is set greater than or equal to VM. The Initial Velocity is set equal to, or higher than the Maximum Velocity. VI must be less than VM.
23	VM is set less than or equal to VI. The Maximum Velocity is set equal to, or lower than the Initial Velocity. VM must be greater than VI.
24	Illegal data entered. Data has been entered that the device does not understand.
25	Variable or flag is read only. Read only flags and variables cannot be set.
26	Variable or flag is not allowed to be incremented or decremented. IC and DC cannot be used on variables or flags such as Baud and Version.
27	Trip not defined.Trying to enable a trip that has not yet been defined.
28	WARNING! Trying to redefine a program label or variable. This can be caused when you download a program over a program already saved. Before downloading a new or edited program, type <FD> and press ENTER to return the device to the Factory Defaults. You may also type <CP> and press ENTER to Clear the Program.
29	Trying to redefine a built in command, variable or flag.
30	Unknown label or user variable. Trying to Call or Branch to a Label or Variable that has not yet been defined.

31	Program label or user variable table is full. The table has a maximum capacity of 22 labels and/or user variables.
32	Trying to set a label (LB). You cannot name a label and then try to set it to a value. Example: Lable P1 (LB P1). The P1 cannot be used to set a variable such as P1=1000.
33	Trying to SET an Instruction.
34	Trying to Execute a Variable or Flag
35	Trying to Print Illegal Variable or Flag
36	Illegal Motor Count to Encoder Count Ratio
37	Command, Variable or Flag Not Available in Drive
38	Missing parameter separator
39	Trip on Position and Trip on Relative Distance not allowed together

9.3 Program errors

40	Program not running. If HOLD (H) is entered in Immediate Mode and a program is not running.
41	Stack overflow
42	Illegal program address. Tried to Clear, List, Execute, etc. an incorrect Program address.
43	Tried to overflow program stack. Calling a Sub-Routine or Trip Routine with no Return.
44	Program locked. User Programs can be Locked with the <LK> command. Once Locked, the program cannot be listed or edited in any way.
45	Trying to Overflow Program Space.
46	Not in Program Mode.
47	Tried to Write to Illegal Flash Address
48	Program Execution stopped by I/O set as Stop.

9.4 Communications errors

60	Not used
61	Trying to set illegal BAUD rate. The only Baud Rates accepted are those listed on the Properties Page of IMS Terminal. (4,800, 9,600, 19,200, 38,400, 115,200)
62	IV already pending or IF Flag already TRUE.
63	Character over-run. Character was received. Processor did not have time to process it and it was over-written by the next character.
64	Startup Calibration failed (Hybrid only)

9.5 System errors

70	FLASH Check Sum Fault
71	Internal Temperature Warning, 10C to Shutdown
72	Internal Over TEMP Fault, Disabling Drive
73	Tried to SAVE while moving
74	Tried to Initialize Parameters (IP) or Clear Program (CP) while Moving
75	Linear Overtemperature Error (For units without Internal Over Temp)

9.6 Motion errors

80		HOME switch not defined. Attempting to do a HOME (H) sequence but the Home Switch has not yet been defined.
81		HOME type not defined. The HOME (HM or HI) Command has been programmed but with no type or an illegal type. (Types = 1, 2, 3, or 4)
82		Went to both LIMITS and did not find home. The motion encroached both limits but did not trip the Home switch. Indicates a possible bad switch or a bad circuit.
83		Reached plus LIMIT switch. The LIMIT switch in the plus direction was tripped.
84		Reached minus LIMIT switch. The LIMIT switch in the minus direction was tripped.
85		MA or MR isn't allowed during a HOME and a HOME isn't allowed while the device is in motion.
86		Stall detected. The Stall Flag (ST) has been set to 1.
87	**MDrive**	In Clock Mode, JOG not allowed
	Hybrid	Not allowed to change AS mode while calibrating
88	**MDrive**	Following error
	Hybrid	Moves not allowed while calibration is in progress.
89	**MDrive**	Reserved
	Hybrid	Calibration not allowed while motion is in progress.
90		Motion Variables are too low switching to EE=1
91		Motion stopped by I/O set as Stop.
92		Position Error in Closed loop. motor will attempt tp position the shaft within the deadband, After failing 3 attempts Error 92 will be generated. Axis will continue to function normally.
93		MR or MA not allowed while correcting position at end of previous MR or MA.

9.7 Hybrid errors

100	Configuration test done, encoder resolution mismatch
101	Configuration test done, encoder direction incorrect
102	Configuration test done, encoder resolution and direction incorrect
103	Configuration not done, drive not enabled
104	Locked rotor
105	Maximum position count reached
106	Lead limit reached
107	Lag limit reached
108	Reserved
109	Calibration failed

WARRANTY

TWENTY-FOUR (24) MONTH LIMITED WARRANTY

Schneider Electric Motion USA warrants only to the purchaser of the Product from Schneider Electric Motion USA (the “Customer”) that the product purchased from Schneider Electric Motion USA (the “Product”) will be free from defects in materials and workmanship under the normal use and service for which the Product was designed for a period of 24 months from the date of purchase of the Product by the Customer. Customer’s exclusive remedy under this Limited Warranty shall be the repair or replacement, at Company’s sole option, of the Product, or any part of the Product, determined by Schneider Electric Motion USA to be defective. In order to exercise its warranty rights, Customer must notify Company in accordance with the instructions described under the heading “Obtaining Warranty Service”.

NOTE: MDrive Motion Control electronics are not removable from the motor in the field. The entire unit must be returned to the factory for repair.

This Limited Warranty does not extend to any Product damaged by reason of alteration, accident, abuse, neglect or misuse or improper or inadequate handling; improper or inadequate wiring utilized or installed in connection with the Product; installation, operation or use of the Product not made in strict accordance with the specifications and written instructions provided by IMS; use of the Product for any purpose other than those for which it was designed; ordinary wear and tear; disasters or Acts of God; unauthorized attachments, alterations or modifications to the Product; the misuse or failure of any item or equipment connected to the Product not supplied by Schneider Electric Motion USA; improper maintenance or repair of the Product; or any other reason or event not caused by Schneider Electric Motion USA.

Schneider Electric Motion USA HEREBY DISCLAIMS ALL OTHER WARRANTIES, WHETHER WRITTEN OR ORAL, EXPRESS OR IMPLIED BY LAW OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. CUSTOMER’S SOLE REMEDY FOR ANY DEFECTIVE PRODUCT WILL BE AS STATED ABOVE, AND IN NO EVENT WILL IMS BE LIABLE FOR INCIDENTAL, CONSEQUENTIAL, SPECIAL OR INDIRECT DAMAGES IN CONNECTION WITH THE PRODUCT.

This Limited Warranty shall be void if the Customer fails to comply with all of the terms set forth in this Limited Warranty. This Limited Warranty is the sole warranty offered by Schneider Electric Motion USA with respect to the Product. Schneider Electric Motion USA does not assume any other liability in connection with the sale of the Product. No representative of Schneider Electric Motion USA is authorized to extend this Limited Warranty or to change it in any manner whatsoever. No warranty applies to any party other than the original Customer.

Schneider Electric Motion USA and its directors, officers, employees, subsidiaries and affiliates shall not be liable for any damages arising from any loss of equipment, loss or distortion of data, loss of time, loss or destruction of software or other property, loss of production or profits, overhead costs, claims of third parties, labor or materials, penalties or liquidated damages or punitive damages, whatsoever, whether based upon breach of warranty, breach of contract, negligence, strict liability or any other legal theory, or other losses or expenses incurred by the Customer or any third party.

OBTAINING WARRANTY SERVICE

If the Product was purchased from an Schneider Electric Motion USA Distributor, please contact that Distributor to obtain a Returned Material Authorization (RMA). If the Product was purchased directly from Schneider Electric Motion USA, please contact Customer Service at info@motion.schneider-electric.com or 860-295-6102 (Eastern Time Zone).

Customer shall prepay shipping charges for Products returned to Schneider Electric Motion USA for warranty service and Schneider Electric Motion USA shall pay for return of Products to Customer by ground transportation. However, Customer shall pay all shipping charges, duties and taxes for Products returned to Schneider Electric Motion USA from outside the United States.

Schneider Electric Motion USA
370 North Main Street, P.O. Box 457
Marlborough, CT 06447 - U.S.A.
Tel. +00 (1) 860 295-6102 - Fax +00 (1) 860 295-6107
e-mail: info@imshome.com
http://motion.schneider-electric.com

REV062812
Product Disclaimer and most recent product information at motion.schneider-electric.com.