

Software Manual

for

SuprMotrV©

Motion Controllers

1.0 DIMAC

It is not necessary to learn how to "program" to be able to use a **SuprMotr**[©]. It does not require "programming" in the traditional sense. It has a built-in capability to convert very simple, direct commands into whatever complex operations are required to accomplish the desired purpose.

There are no files to load, no "booting" process to endure and nothing to learn other than a few simple guidelines on the structure of the commands. The **Diva MACro** language is deceptively simple, yet very powerful. Our goal has been to provide the user with a tool that is intuitively easy to use, yet powerful enough to implement any desired application with ease. If the application requires no more than movement to different positions, then the user need only learn one command, **MA n** (**M**ove to **A**bsolute position **n**). Other commands may be learned as the need arises.

There is no formal structure. Commands are executed immediately after a carriage return character is sent. There are no line numbers, no editor and no "Go To" commands, yet we have never heard of a motion control application that was not practical, using **DIMAC** as the control language. Moreover, **DIMAC** is extremely compact. Very complex machine control programs may be reduced to 20 or 30 steps.

Of course the user may use any preferred language on any host computer to control the SuprMotr. All that is required is to generate ASCII text script files and transmit them through the RS-232 port.

1.1 Terms Used

Throughout this manual certain abbreviations and standards will be used. Commands are shown in **boldface type**. Expected responses are enclosed in quotes. The words "carriage return" are a holdover from the days of the typewriter and refer to the code sent by the PC keyboard key labeled "Enter".

All commands use two-letter abbreviations, indicated in bold type such as **TP** that are either the initial letters of the operation to be performed, or standardized letters for these operations, where possible. **TA** is used for **Tell Analog**, so **Tell accelLeration** has to use **TL**, for example.

For example: since "ON" and "OFF" both begin with the letter "O", we have chosen the letter "N" to mean "ON", and "F" means "OFF", in all commands. Example: **MN** and **MF** are commands to turn the motor servo loop "ON" and "OFF".

For those commands that require a value to be entered, it will follow the command letters. In the following discussion, it will be shown as **n**. The real-time syntax checker monitors the length of the number entered and flags gross errors.

1.2 Command Types

Over 50 commands are available for programming the **SuprMotr**. Moreover, many additional application-specific commands have been developed and can easily be added to the SuprMotr's table-driven command structure. Use these commands for controlling and reporting motion as well as the state of motion control parameters and status.

Wherever possible, the command structure has been designed to minimize the effort required to use the **SuprMotr**. As Everett Long used to say, "Make it as easy to use as a screwdriver."

Commands may be executed in various ways:

Single command --	One function executed immediately
Compound command --	Multiple functions executed immediately
Macro command --	Compound command stored for later execution
Sub-commands --	Single-character commands

1.2.1 Single Commands

A **single command** is executed immediately after the carriage return is received and will be repeated each time the carriage return is received, until a different command is entered. With this feature, it is very easy to continuously monitor the state of an input by simply holding down the "Enter" key.

Examples: **MR2000** (rtn) : Move motor 2,000 steps relative to the present target

MN (rtn)	: Set motor in ON state
GH (rtn)	: Send motor to home position (go home)
TP (rtn)	: Report (tell) position for motor
MA20000 (rtn)	: Send motor to absolute position 20,000
tp (rtn)	: Report (tell) the position of motor

Both uppercase and lowercase characters are valid, and spaces are allowed.

TP (rtn)	: Reports "+000000000"
tp (rtn)	: Reports "+000000000"

1.2.2 Compound Commands

A **compound command** is a series of single commands separated by commas rather than by a carriage return. In this way, it is possible to string together several

commands before terminating with the carriage return. These multiple commands will be executed sequentially.

The syntax for a compound command is:

CMD[*n*], CMD[*n*], ..., (rtn)

Examples: **MR2000,WS,MR-4300** (rtn)

**mr5000,ws,wa500,ma12000,ws,wa800,tp
ma 3500, WS,wa 120,tp**

A compound command such as the following example may be entered as one program line. It instructs the motor to move 1,000 encoder counts in the positive direction, wait in that position 500 milliseconds, return to the original position, wait 1 second, and then repeat the cycle 5 times.

Example: "**MR1000,WS500,MR-1000,WS,RP5**" (rtn)

Once this command is entered, it remains in the buffer until replaced by another command and can be re-executed by transferring a carriage return. Compound commands may contain up to 18 single commands.

1.2.3 Macro Commands

Macros can be a most powerful tool for the programmer. A **macro command** is a grouping of commands to form a short program, implemented by a macro number.

To use macros for programming the **SuprMotr** controller, insert an **MD** (Macro Definition) command as the *first* instruction in the command string. The syntax for macro commands is:

MD(macro#), followed by a compound command string.

Example: **MD3,MR1000,WS500,MR-1000,WS,RP5** (rtn)

In this example, **MD3** defines macro #3. To call up this macro, just issue the command **MC3**.

Macro commands may be stored in any order, but you may prefer to number them sequentially as they are entered, because the system gives no warning if you define (and overwrite) an existing macro. You may wish to do this under many conditions, such as when one macro is called by another. It is sometimes desirable to define a complex

motion in one macro and define key parameters such as torque, gain, or velocity, in another macro which is called by the main macro.

Macro commands can call another macro, without limit. For instance, MC1 could call MC2, and MC2 could then call MC3 and still be able to return to complete the remainder of MC1.

Example: MD1,MC2,MC3,MC4,MC5,MC6

Macro commands may contain up to 17 single commands.

1.2.4 Sub-commands

Sub-commands may be used at any time. They are most useful for interrogating variables without disturbing an operating program. An example would be a situation where a repetitive motion is in process, such as dispensing adhesive in a pattern. The operator would like to know the status of the command without stopping it. The sub-commands can be used to read the number of iterations in a loop, current system status, position, etc. A single character is also provided for emergency stop action.

Sub-commands

!	(ASCII 33)	AB	Abort motion
?	(ASCII 63)	TF	Tell Following error
%	(ASCII 37)	TS	Tell Status
#	(ASCII 35)	TC	Tell I/O Channel status
\	(ASCII 92)	TI	Tell Iteration count
'	(ASCII 39)	TP	Tell Position (single quote or apostrophe)

1.3 Data formats

1.3.1 Reporting Commands

Reporting commands cause the **SuprMotr** controller to emit a string of data, whether a position, servo control parameter, help or other information. These commands are easy to remember as they usually begin with a **Tell** statement. For example, **TT** (Tell Target), or **TP** (Tell Position).

The **Tell** commands query registers which may be of different lengths. Accordingly, the reports have different formats. The following examples assume that **DM** (Decimal Mode) has been selected. Note that commands that return data usually have a format that specifies first the board address to identify the source of the data, then the channel that pertains to the following data, followed by a single letter identifying the command (usually

the same letter used in the command), and then the data in either decimal or hex format, depending on the mode selected.

```
Example:  transmit:  DM,TI ,HM,TI(rtn)
          receive:  "00I+0000000255"
          "00I000000FF"
```

Other formats exist for special purpose commands, such as **TT**, **VE** and **CS**. The format for these commands is as shown in the detailed command description.

2.0 Starting Communications

Note: We strongly urge the use of a terminal emulator for initial testing and familiarity with DIMAC, even if the intended use is through a host program.

The first command to test the communications could be a **VE**rsion command.

Type in a **VE** and "enter". On your terminal monitor a message similar to the following should be displayed:

```
"(c) 2002 DIVA Automation, Inc."
"Ver. 1.0 for SuprMotr, 2 Dec 2002"
```

If you get a response (as above) on your terminal, the **SuprMotr** is ready for use. Now all registers are loaded with their default values. Most tasks may be accomplished using only these default values--other tasks may require value modifications.

If you do not see this response, there are several possible reasons:

- Check the power supply voltage to the **SuprMotr**. It must be within the specified range and must be connected with the proper polarity as shown on the connection diagram.
- Check the cable to your terminal. It should be connected to the TxD and RxD pins from the **SuprMotr**, as well as to ground. Various terminals have differing pin assignments for TxD and RxD, so it may be necessary to reverse pins 2 and 3 on the RS-232 connector. No other connections are required. The **SuprMotr** uses buffering, rather than handshaking, to provide error-free communications, so it may be necessary to disable handshaking on your terminal.
- Check the baud rate on your terminal. The **SuprMotr** is designed to operate at 9600 baud as delivered. The baud rate can be changed by command after communications, but the terminal must be set to the same baud rate as the **SuprMotr**.

- If you are using a PC-compatible computer, the HyperTerm program supplied or available for free download, is an excellent terminal emulator.

If this response does not appear, there is no point in going further until it does. If communications are not possible, please report the problem to us. We will respond as quickly as possible.

3.0 SuprMotr Commands

All **SuprMotr** commands use two-letters to identify the type of operation desired, followed by any pertinent data value. For example, **HE** by itself is adequate to display a list of available commands, but **MR** alone would be useless because the system would not know how far you wished to move. All commands are tested for acceptability as they are entered. In the case above, **HE9** would generate an error because it does not accept data, and **MR** followed immediately by a carriage return or comma would also be rejected. In the case of **MR**, it can be followed by a minimum of 1 and a maximum of 9 digits, to accommodate the allowable range of motion. Single byte quantities such as maximum torque setting with **SQ** accept 1 to 3 digits, and so on.

SuprMotr commands are arranged by group in the following discussion.

3.1 UTILITY COMMANDS

DM **Decimal Mode**

The **DIMAC** program can handle both decimal and hexadecimal numbers. The **DM** command selects decimal mode. After entering this mode, all inputs are interpreted as decimal numbers and the system reports requested values in decimal as well, with the exception of certain commands, such as **CS** (checksum).

The selected number mode is one of the parameters that is restored from EEPROM at reset..

Example: **DM,TQ,HM,TQ(rtn)**

Returns: "0Q:0085"

"0Q:55"

HM **Hexadecimal Mode**

After transmitting this command, all inputs and outputs are in hexadecimal mode.

Example: **HM,2S7F (rtn)** : Select hex mode and set output channel 2 to 7F (Hex).

EN **Echo oN**

Enables echoing of command characters as they are entered. Each character received is echoed unchanged. This is a very useful feature when the SuprMotr is being controlled manually from a terminal and may be used as a high-level handshake to confirm proper command reception when controlled by a host program.

EF **E**cho of **E**

Disables echoing. When control is from a computer program, it is sometimes easier to program if there is no echo, unless the program uses it for verification of successful transmission.

BR(n) **B**aud **R**ate (0 > *n* > 3)

Sets baud rate according to the following table:

n = 0	4,800
n = 1	9,600
n = 2	19,200
n = 3	38,400

RT **R**ese**I** system

Performs a complete restart of the system, including restoration of all system parameters stored in EEPROM. All axes are set in the **MF** condition. It should be used with care if you have made many changes since powering up the controller, such as setting up gains, velocity or other parameters that you wish to preserve. These values should be stored, using the **UD** command before issuing the **RT** command if they are intended to be permanent changes.

DE **D**Ebugger

Enters on-board monitor. Instructions for the use of this command are contained in a separate appendix. Please contact the factory if there is a need for it.

3.2 MOTION and SEQUENCING COMMANDS

AB **A**Bort motion

This command opens the motor servo loop and dynamically brakes the motor by applying the same voltage to both sides. The motor stops abruptly and any further motion requirements are deleted. The target position is changed to be equal to the present position. **AB**ort motion is used for stopping an undesired motion as well as to retain a position.

Example: **AB** (rtn) : Aborts motion of motor
 AB,MN (rtn) : Aborts motion and holds at the position where issued.

FE_n Find Edge

This command is used to initialize the system at a given position. The motor runs at the last programmed speed until a transition occurs on the reference input line. The direction of motion depends on the value of *n*.

Example: **FE0** (rtn) : Causes motor to move in a positive direction until the reference input changes state. If the reference input is high when the command is issued, the motor runs toward the positive limit until the input changes to low, and vice versa, unless of course a limit switch is encountered first. **FE** is also useful as a simple command to move to the end of motion when there is no reference switch in the direction of motion.

FE1 (rtn) : Causes motor to move in a negative direction until the reference switch changes state.

These commands assume that your system has the reference line connected. Otherwise, motion will continue until the target position of + or - 2,000,000,000 is attained, or until it is stopped manually or by command.

FE can be useful in calibrating the home position on start-up or for simply slewing long distances at the most recent velocity setting.

On startup, a command string such as "**SQ40,FE0,WS,WA500,DH,MN,SQ255**" could be used to find the Home position at a mechanical stop. This command sets the maximum drive to the motor at less than 20% of full torque (40/255) and tells the motor to run until stopped. When the motor encounters the stop, the following error will rise quickly, triggering a halt because of excessive error. (See **SM** command.) This error halt will disable the motor loop, so the **MN** command at the end is required to maintain the new position against any forces that might be attempting to move it, such as gravity on a vertical stage.

The command example given above may also be defined as Macro #0, which will cause it to be executed automatically after every reset or power cycle.

GH Go Home

The Go Home command causes the motor to move to absolute zero position. Equivalent to an **MA0** (Move Absolute axis a to zero) command.

Example: **GH** (rtn) : Moves motor to zero position.

MA0 (rtn) : Equivalent command.

MA n Move Absolute

This command sets the target position to absolute position n . The zero, or home position, may be defined by the **DH** (Define Home) statement. If not otherwise defined, it is the position where the controller was when power was applied, or when an **RT** command was issued.

MA0 (rtn) : Tells motor to go to Home position
MA12 345 (rtn) : Tells the motor to move to the position 12,345 encoder counts from the home position.

MR n Move Relative

This command generates a motion of relative distance of n counts in the specified direction from the actual motor position. n may be either a positive or negative number up to a total target position of approximately 2 billion.

Examples: **MR5000** (rtn) : Motor moves 5,000 counts in positive direction.

MR-330 (rtn) : Motor moves 330 counts in negative direction.

MR2000,WS,MR-1200 (rtn) : Move 2000 counts, wait one second, then move back 1200 counts.

MF Motor ofF

When this command is issued, the motor is no longer held in position control and may be moved freely, although with dynamic braking, so the motion is not as free as if power were removed. The **MF** command is used to prevent unwanted movement or to allow for manual positioning of the unit. When manually positioned, however, the motor position is still monitored and may be reported by the **TP** command. The opposite command is **MN** (Motor oN).

Examples: **MF** (rtn) : Sets motor position control to OFF

For the purpose of implementing joystick capability, the **SuprMotr** allows the motor to be freely driven while in the **MF** state, implementing a sort of continuous **AB** command.

This command is useful for determining the encoder resolution. For example. Issue the **MF** command, manually position the motor/encoder at some known angle, issue a **DH** command to set the position to zero, then turn the motor/encoder one revolution. Now the **TP** command will report the number of encoder counts/ revolution. High precision is not

required, since encoders are rarely encountered that are other than numbers like 100, 500, 2000, and so on. If the exercise above yields a position of 1997, it may be safely assumed that the encoder is 2000 counts per revolution. An exception are those encoders with binary counts, such as 512, 1024 and so on. An effective technique for determining the difference between a 500-line encoder (2000 counts per revolution) and a 512-line encoder (2048 cpr) is to command a motion of 10 or more revolutions. The difference of 480 counts would be approximately 90 degrees or motion and therefore very obvious.

MN Motor oN

This is the normal system control mode, where the **SuprMotr** controls the axis position continuously. Any deviation between actual and target position causes the motor to be driven toward the target.

Example: **MN** (rtn)

RPn RePeat *n*

This command causes the command string to repeat *n* times. If *n* is not specified, the commands are repeated 65,535 times. The repeat loop may be interrupted by transferring an escape character, which is defined as the backspace. (To repeat forever, use two **RP** commands in sequence.)

Example: **TE,WA500,RP99** (rtn) : Will display the distance to the target every 500 milliseconds (0.5 second) for a total of 100 times.

TV,WA,RP,RP (rtn) : Will display the velocity until interrupted by the escape character.

WAn WAit *n* milliseconds

This command inserts a wait period of *n* milliseconds before going to the next command. If *n* = 0, then it is set to 65,536.

Example: **MR2000,WA3000,MR-2000** (rtn)

This command line will move the motor by 2,000 steps, then wait for 3 seconds after the motor starts, then move back 2,000 steps. The wait period of 3 seconds also includes the time the motor moves. If the actual rest time of the motor should be 3 seconds, an additional command must be inserted to prevent the wait interval from beginning until the motor has completed it's motion.

MR2000,WS3000,MR-2000 (rtn)

WSn Wait *n* milliseconds after **S**top If *n* = 0, then it is set to 65,536.

Wait until axis **a** has reached the end of trajectory before continuing to the next command.

Example: **MR5000,WS,TE** (rtn)

3.3 PARAMETER SETUP COMMANDS

DH Define Home

Defines the current motor position as zero position (home position).

Example: **DH** (rtn) : Sets the current motor position to 0.

DPn Define Proportional Gain ($0 < n < 255$)

This command sets the slope of the proportional relationship between the position error and the motor voltage. The higher the gain value is set, the higher is the stiffness of position coupling, so that a given error value causes a proportionally larger motor current driving the motor towards the target.

The factory-set gain value is usually stable for motors small enough to be driven from the on-board driver. The optimum system value depends on friction, inertia, motor power, and the resolution of the encoder. It must be determined by the user.

If the error reported by an axis after completing its motion is excessive, the proportional gain value may be increased in small increments until the error is within acceptable limits. If the axis becomes unstable and begins to oscillate, the proportional gain must be reduced until the oscillation stops.

Example: **DP20** (rtn) : Sets proportional gain of 20

DIn Define Integral Gain ($0 < n < 255$)

Sets the gain to be applied to the integral term in the **PID** algorithm. The primary function of this term is to overcome friction-induced static errors.

DDn Define Derivative Gain ($0 < n < 255$)

Sets the gain to be applied to the derivative term in the **PID** algorithm. The primary purpose of this term is to increase damping and reduce overshoot at the end of motion.

DLn Define integral "windup" Limit ($0 < n < 255$)

Sets the allowable limit for contributions from the integral gain term. Prevents excessive drive when motor is temporarily prevented from moving.

SAn Set Acceleration

Defines the acceleration rate in encoder counts/second/second. The value for *n* may be within the range from 1 to approximately 900,000.

SVn Set Velocity

Causes the motor to accelerate to and run at velocity *n* in subsequent motion commands. The value is given in encoder counts per second. If the torque load changes on the motor, the controller attempts to maintain the velocity by varying the motor current. The value *n* may be in the range from 1000 to 255,000. However, the usable range of velocity settings is determined by the number of lines in the encoder and the maximum RPM of the motor in use. Many systems will fall into a range of 50,000 to 250,000 counts/second.

(An encoder with 100 line resolution will generate 40,000 counts per second at 6000 RPM, for instance.) (100 lines x 4 counts x 6000 RPM / 60 seconds per minute)

Example: **SV24000** (rtn) : Sets the velocity of motion to 24,000 counts per second.

SQn Set maximum torQue to *n* (0 > *n* > 255)

Sets the maximum drive that will be allowed for any set of motion control situations. The default value is 255, which is the maximum possible output. A typical value for a minimum is around 10-20, dependent on the size of the motor and the load. For maximum system safety, set the **SQ** value slightly above that required for normal operation as determined through experimentation.

SMn Set Maximum following error = *n* counts (1 > *n* > 65535)

Sets the maximum allowable error between the dynamic target and the actual position. May be changed as often as desired to provide maximum protection to the system. The normal following error can be monitored during motion with the **TF** command. For maximum system safety, use the **SM** command to limit following error to a value slightly above that required for normal operation.

SJn Set the number of steps to be moved by the Jog switch to *n*.

When the jog switch is closed, *n* will be added to the target position.

SNn Set the miNimum output to the motor to *n* D/A counts.

This command is for factory setup and is not normally used in the field. It is used to trim the output for maximum performance.

LN Limit switch operation oN

Enable limit switch operation. When a limit switch is encountered during motion, operation is halted and is no longer possible in that direction as long as the switch remains closed. Movement in the reverse direction is not affected.

LF Limit switch operation ofF

Disables all limit switch operation.

LL Limit switch active Low

Sets the active state of both limit switch inputs to low. When this input is less than 1 volt and limits are enabled, motion in that direction will be terminated.

LH Limit switch active High

Sets the active state of both limit switch inputs to high. When this input is greater than 3 volts and limits are enabled, motion in that direction will be terminated.

3.4 REPORTING COMMANDS

Reporting commands cause the **SuprMotr** controller to emit a string of data, whether a position, target, help or other information. These commands are easy to remember as they usually begin with a **Tell** statement. For example, **TT** (Tell Target), or **TP** (Tell Position).

The **Tell** commands have been structured to display only the maximum number of allowable digits. Accordingly, the reports have different formats. The following examples assume that **DM** (Decimal Mode) has been selected. In the first example, if **HM** (Hex Mode) had been selected, there would be only 8 digits after the letter 'T.'

Example for 32-bit register query:

```
transmit: TT (rtn)
receive: "00T:+0000000000"
```

Example for 8 and 16-bit register query:

transmit: **TQ** (rtn)
receive: "00Q0255" (in DM)
 "00QFF" (in HM)

VE Version report

Reports the copyright notice and revision level of the installed firmware. This is the default command loaded into the command buffer on startup. If the first character received is a carriage return, this command should be executed.

TE Tell Error

Reports the position error of the motor as determined by subtracting the actual position from the target position. This command may be given while the motor is moving. It can be used to determine if the motor is actually moving, if it is moving in the proper direction, and if it is maintaining the position without oscillation after stopping.

Example: **TE,WA150,RP** (rtn)

This example will track the calculated positional error of the motor during and after the move. Every 150 ms the actual distance from the target is reported. The output may be stopped by entering any character. This command string can be very useful for monitoring the settling of the motor at end of travel, to detect overshoot or oscillation.

Report: "+0000000000"

TG Tell Gain

A special command that reports the values defined in the **DP** (Set Proportional Gain), **DI** (Set integral gain) and **DD** (Set derivative gain) commands, as well as the **DL** (Define Limit) command.

Report: 0P:00000080
 0I:00000030
 0D:00000070
 0L:00000000

GP Get Proportional gain value

Reports the value of only the proportional gain term.

GI Get Integral gain value

Reports the value of only the integral gain term.

GD Get Derivative gain value

Reports the value of only the derivative gain term.

GL Get integral term Limit value

Reports the value of only the proportional gain term.

TI Tell Iterations

This command reports the state of the repeat counter. It is useful for determining the number of times a repetitive action has taken place.

Example: **MR100,WS250,TI,RP99**. The motor will make repetitive moves of 100 steps, with a delay of .25 seconds between steps, for a total of 100 times. It will report the number of iterations remaining to be performed after each iteration.

```
Report:      "00I+0000000000" (First TI is executed before number of loops is
              specified)
              "00I+0000000099"
              "00I+0000000098"
              (97-03
              "00I+0000000002"
              "00I+0000000001"
```

TA Tell Analog

Reports the value of all eight analog input channels as 0-255. The normal reference value is +5 volts, so the conversion from reported value to volts is the ratio of the reported value to 256, multiplied by 5. A report of 237 relates to **4.63** volts. $(237/256*5)$

TO Tell Output

Reports the value of the present analog output setting. Normally used when the output is controlled manually with the **SO** and **IO** commands. The same scale factor calculation pertains, as described for the **TA** command.

TCn Tell Channel *n*

Reports the logic state of the digital input channels. When *n* = 0, all 16 channels are reported in two bytes. See the **CP** command description for the method used to interpret the reported value by individual channels.

When *n* > 0, then only that channel is reported, and the format is **0Cn:0** or **0Cn:1**.

DU DUmp

A special command to report the current value of several variables. It is equivalent to sending **TT, TP, TD, TS**.

TS Tell Status

When the **TS** command is given, the status of the system as well as the motion and limit switches is reported. The format is "0S:OO MM LL SS VV EE ", where OO is the operating system status consisting of the flags listed below, MM are the motion control status flags as listed below, LL are the limit switch status flags as listed below, SS are the live status of the limits and reference switch, VV is the move mode and EE is the error number.

The data is presented in either decimal or hex form, depending on the mode selected. For those not familiar with hexadecimal, it will require some practice to make use of this command. A tutorial on the use of this type of data will be included in a planned appendix to this manual.

	<u>Operating system</u>	<u>Motion flags</u>	<u>Limit switch flags</u>	<u>Bit value</u>
Bit 0	Echo	Motor on	Lim ena	1
Bit 1	Wait flag	Error pol	Lim state	2
Bit 2	Error	Move pol	FE flag	4
Bit 3	Leading 0	Move error	FE state	8
Bit 4	Macro in process	Decel pol	Lim left live	10
Bit 5	LZ disable	Encoder phase	Lim right live	20
Bit 6	Number mode	Excess foll err	-----	40
Bit 7	Addressed	End of trajectory	-----	80

Limit/reference switch

Bit 0	Reference switch live status	1	
Bit 1	Positive limit live status		2
Bit 2	Negative limit live status	4	
Bit 3-7	-----		8

Move mode

Error

0 = Stopped	1 = Illegal first character	6 = Number too long
1 = Accelerate	2 = Letter required here	7 = Number too short
2 = Constant velocity	3 = Command not found	8 = Comma required
3 = Decelerate	4 = Negative value not allowed	9 = Too many commands
4 = Homing	5 = Number required here	A =
		B = Macro number > max

TP Tell Position

Tell Position reports the absolute position of the motor. **TP** may be used to monitor motion during both motor on and motor off status.

Example: **TP,WA100,RP** (rtn)

This command string causes the controller to report the current position every 100 ms.

Report: "+0000005555"
 "+0000005555"
 "+0000005555" ...(until stopped, or 65536 times)

TT Tell Target

Reports target position. This is the absolute position to which the servo loop will try to drive the motor any time the **MN** (Motor ON) state is in effect. The target position may be specified directly with the **MA** (Move Absolute) and several other commands, or indirectly with the **MR** (Move Relative) command.

If the system is in decimal mode, ten digits will be reported with a leading minus sign (-), if the position is less than the position defined as "home." When the hex mode is in effect, eight digits are reported in two's-complement notation. In both cases, the data is preceded by a "T" character.

Example: **TT** (rtn) "0T:+0000000000"

TY Tell programmed velocitY

Reports the value programmed with the **SV** command. The values reported with the **TV** and **TY** commands should differ by only a few counts.

Example: **TY** (rtn)

Report: "0Y:+00020000"

TV Tell actual Velocity

Measures and reports the number of encoder counts moved during the previous second. This number will ordinarily be very nearly identical to the programmed velocity reported with the **TY** command, during motion. (During the constant velocity portion of the move.)

TJ Tell Jog value

Reports the number of steps to be moved in response to the jog switch actuation.

HE HElp

This command reports the valid mnemonics for the installed firmware version. It lists all two-letter commands supported by your software version.

TM[n] Tell Macros (0 < n < ??)

Displays all previously stored macro commands. If $n = 0$ or, if n is not specified, all macros will be displayed. Since macros may be defined in any sequence, the **TM** command is useful for confirming the existence of, as well as the contents of, macro commands.

TZ Tell macro Zero

Special case of the **TM** command for Macro command zero, which is handled differently from other macros. If there is a macro zero loaded, it will be executed automatically when power is applied. This is the mechanism by which totally turn-key operation is accomplished.

Macro zero may be used only to set up custom parameters before manual control is begun or it may then transfer control to other macros for stand-alone operation.

TB Tell Board address

Reports address of enabled board. If more than one board is accidentally enabled, this command will help detect it. It is useful in confirming both the presence of a **SuprMotr** having the most recent address, and for confirming successful reception of an address change. Please see discussion of multi-drop addressing for the **SuprMotr** in the appendix.

Report: "A0000"

CS CheckSum

Calculates and reports the sum of the entire contents of the **SuprMotr** operating system program in memory. It does not include the macro storage space. Useful for conducting a confidence test of the **SuprMotr**. The reported value should remain constant. If not,

there is a problem and the **SuprMotr** should not be used for motion control until it is resolved.

Report: "C53A"

TL **T**ell acceLeration

Reports programmed acceleration value.

TF **T**ell Following error

Reports the difference between the dynamic target and the actual position. During motion, it is normal for the actual position to lag behind the target position by some amount, usually dependent on the programmed velocity. If the velocity is higher than physically possible for the system, or if it has encountered an impediment, the following error will increase. If the obstruction is temporary, the servo action will attempt to restore the error to zero when it is removed.

TD **T**ell Dynamic target

Reports the instantaneous value of the dynamic target. As the motor is moved along the programmed path to the final target, the dynamic target controls the position at each instant along the way to define the trajectory.

TQ **T**ell torQue

Reports the setting for maximum allowable output to the motor driver. The maximum value is 255. The minimum usable value will depend on the motor used as well as the load. For maximum safety, this value should be set slightly above that required for normal operation. This value can be found by experimentation. Set **TQ** to a trial value and command a normal move. During the move, monitor the following error with the **TF** command. When a value for **TQ** has been found that causes the following error to increase, set **TQ** somewhat above this value.

3.5 MACRO COMMANDS

MCn execute Macro Command *n*

This command may be used to implement a previously defined macro command. If there is no macro defined by the number *n*, no action will be taken.

Example: **MC3** (rtn) : Will execute macro #3

Before calling MC3, that macro must have been defined with the **MD** command.

MDn Macro Definition

This command is used to define a new macro command. Any duplication of numbers will simply result in the loss of any previously defined macro using that number. To define a macro, choose any number in the allowable range for the new macro and enter MD followed by this number and a comma before entering the function you wish the macro to perform, in the normal manner.

Example: **TT,TP** (rtn) :Tells target followed by position
MD4,TT,TP (rtn) :Creates a macro command to Tell Target, then
:Tell Position and assigns it to number 4.
MC4 (rtn) : Executes macro #4
:(Same as entering **TT,TP** (rtn))

Important note: If there is a Macro #0 defined, it will be automatically executed one second after power is applied or a reset (**RT**)command is issued. Although this is a very powerful and useful capability, it may also create a problem if for some reason it is no longer desired, or has become corrupted. In order to allow the user the opportunity to modify an existing MC0 without executing it, any keyboard character may be used to interrupt operation during the one second delay. Control simply bypasses MC0 and passes to the normal command loop. From this condition, MC0 may be reset or redefined as desired.

RMn Reset Macro *n* space (*n* = 0 resets all macros)

Used only to initialize the memory reserved for macro commands. It clears the macro storage.

Example: **RM** (rtn) : Removes all stored macros

RZ Reset macro zero space

A separate command is required for macro zero because *n* = 0 is used as a special case for **RM**.

3.6 I/O Commands

BN **Brake oN**

Sets dedicated brake output to high level.

BF **Brake ofF**

Sets dedicated brake output to low level.

CIn **Channel *n* is an Input**

Sets the direction of channel *n* as an input.

CT*n* **Channel *n* is an ouTput**

Sets the direction of channel *n* as an output. Unless this command is used to define the channel direction, the CN and CF commands will have no effect on that channel.

It is always possible to read any channel, whether set as an input or an output. However, setting it as an output can interfere with the ability to read an external signal, depending on the relative source impedances if one is high and the other low.

Important Note: If a channel is defined as an output, do not connect it to a low impedance source. The I/O drivers are active pull-up/pull-down and may be damaged if there is no current limiting to an external source.

CN*n* **Channel *n* oN**

Sets channel *n* to high level. Also sets direction of channel *n* to output. The state may be read with the **TC** command, regardless of the direction.

CF*n* **Channel *n* ofF**

Sets channel *n* to low level. Also sets direction of channel *n* to output. The state may be read with the **TC** command, regardless of the direction.

CP*n* **Set Channel Pattern to *n***

Sets all channels previously defined as outputs to the level described in hex format by *n*. Does not set direction of channels to output. All sixteen channels may be set simultaneously with this command, or any subset. It is the responsibility of the user of this command to prevent accidental change of a previously set output.

For example, if channel 3 had been set **oN** by a previous command and it is to remain **oN**, then the user must make certain that the bit for channel 3 is high in all subsequent **CP** commands. If it is not practical or desirable to do so, then the use of this command should be avoided. It provides a very fast means of setting all outputs with a single command, but carries a responsibility with its use.

The format of **n** is as follows:

<u>Channel</u>	<u>Decimal value</u>	<u>Hex value</u>
1	1	1
2	2	2
3	4	4
4	8	8
5	16	10
6	32	20
7	64	40
8	128	80
9	256	100
10	512	200
11	1024	400
12	2048	800
13	4096	1000
14	8192	2000
15	16384	4000
16	32768	8000

To set channels 3,6,8 and 14 **oN**, add the values corresponding to those channels. In this case, 4+32+128+8192=8356 for DM and 4+20+80+2000=20A4 for HM. Any output channel not included will be set to **oF**. Channels defined as inputs will not be affected.

Example: **CP8356** (Normal mode after reset is **DM**)
 HM,CP20A4 (Mode remains selected until reset)

aSO_n Set analog Output **a** to **n** (1 > **a** > 2) (0 > **n** > 255)

The output voltage is determined by the ration of **n/256** to 5 volts.

Example: **2SO128** sets the voltage at output #2 to 2.50 volts.

This command may not be used when the servo loop is in use. The servo uses both analog outputs to drive the motor in two directions. This command is provided for applications that do not use the servo loop or for special situations where it is desirable to drive the motor with a manual control or to simply slew it at an open-loop velocity.

aIO n Increment analog Output **a** by **n** ($-255 > n > 255$)

The same considerations apply to use of this command as to the **SO** command. The servo loop updates constantly when enabled, overwriting any effect of this command.

IO is useful for adjusting the output in variable size steps or for creating a low frequency waveform. A simple method of testing the output stage is to use the command string: **1IO16,RP,RP**. This creates a 16-step sawtooth wave on D/A output #1 that repeats forever. If only one RP command were used, the command would run to completion very quickly because of the 65,536 limit of a single RP command. The repeat count of the second RP command is overwritten by the first command when it repeats, which results in continuous operation until reset or *interrupted by a backspace character*.

To easily adjust an output voltage, use the SO command to set an approximate voltage, then use the IO command with repeated carriage returns to adjust the output as desired. Example: **1IO-2** will decrement channel 1 by 2/256 (approx. 39 mv) each time the ENTER key is pressed.

Note: The output value is not clamped at maximum output. If the commanded value exceeds 255, the output rolls over modulo 256. In other words, **1SO250,IO10** is equivalent to **1SO4** because $250+10-256=4$.

The **TO** command may be used to report the level of the output.

WN n Wait until channel **n** is **oN**

Command execution is paused until the specified channel is high. If already high, there is no delay.

WF n Wait until channel **n** is **ofF**

Command execution is paused until the specified channel is low. If already low, there is no delay. **WF** and **WN** may be used together to synchronize execution of a loop to the actuation of a switch.

Example: **WF3,WN3,MR12345,RP** This will cause the program to wait until channel 3 is low (perhaps from a switch closure), then continue waiting until it is high (the switch is released) before moving the target. Each time the switch is pressed and released, the

target will be advanced. Note that without the **WN** command, the loop would be executed as fast as the program could loop. The value of 12345 would be added to the target position a thousand or more times each second, as long as the input is low. There may be applications where it is desired to have a simple gating action such as this, but it is highly unlikely that a motion command would fall into this category.

Another sequence option would be to insert short delays to guard against multiple executions caused by "switch bounce". Many mechanical switches bounce away from the contact on actuation. An effective means of preventing response to the extra indications is to wait until the switch has had time to settle against the contact. Typical switches will settle within 10-15 milliseconds. We can insert a 20 ms delay after each of the wait commands to insure a stable condition before continuing.

WF3,WA20,WN3,WA20,MR12345,RP will accomplish this function. These delays are short in comparison with manual operation.

XNn eXecute command only if channel *n* is o**N**

When it is desirable to perform an action only if certain conditions are met, then the **XN** command is very useful.

Example: **XN12,MC15** If we assume that macro 15 performs an operation that would be unnecessary, undesirable or dangerous if it were performed when channel 12 is low, then this sequence will prevent it from occurring. An example might be if there is an input from a limit switch of some kind or from a system logic controller that indicates when a container is available for filling. If the task is to fill bottles with a liquid, it is undesirable to dispense the

XF n eXecute command only if channel *n* is of**F**

Command Summary

(Note: Not all commands are implemented in the first release. Non-implemented commands are denoted by ~~strikethrough-text~~.)

Reporting commands

VE Display version number
TI Report the iteration number
TS Report status
HE Print list of commands
TB Report board address
TM Report macro contents
~~TZ Tell Macro Zero~~
TY Tell programmed velocity
TL Tell programmed acceleration
TE Tell error (distance between position and final target)
GP Get Pgain
GI Get Igain
GD Get Dgain
GL Get Iterm limit
TG Tell gains (P, I, D, L)
TP Tell position
TT Tell target
TF Tell following error (distance between position and dynamic target)
TD Tell dynamic target
TV Tell actual velocity
TQ Tell torque setting
TA Tell Analog input value
TC Tell I/O channel
TJ Tell jog value
TO Tell D/A output setting
DU Dump command (TT, TP, TD, TS)
CS Perform self-test checksum

Utility commands

DM Input and output in decimal format
HM Input and output in hex format
RT Reset all parameters to default and do power-up start
EF Turn off echo to RS-232 port
EN Turn on echo to RS-232 port
BR Set baud rate
DE Enter debugger
~~UD Update system parameters in EEPROM~~

DA Define controller address

Motion and sequencing commands

AB Abort motion
ST ~~Stop with deceleration~~
MN Motor on
MF Motor off
WA Wait specified time
MR Move relative
MA Move absolute
DH Define home
GH Go home
RP Repeat from beginning of line
WS Wait until end of trajectory (motor stopped)
FE Find edge

I/O Commands

BN ~~Brake output on~~
BF ~~Brake output off~~
CF Channel off
CN Channel on
CI Channel In
CT Channel out
CP ~~Channel pattern~~
SQ ~~Set analog output voltage~~
IO ~~Increment analog output setting~~
WN Wait until specified I/O channel = 1
WF Wait until specified I/O channel = 0
XN ~~Execute only if specified I/O channel = 1~~
XF ~~Execute only if specified I/O channel = 0~~

Parameter setup commands

SV Set Velocity
SA Set acceleration
SQ Set maximum torque
DP Define proportional gain
DI Define integral gain
DD Define derivative gain
DL Define integral "windup" limit
SM Set maximum following error
SJ Set jog value
SN Set minimum motor output

Limit Switches

LN Limit switch operation on
LF Limit switch operation off
LL Limits switches active low
LH Limit switches active high

Macro commands

MD Macro definition
MC Macro command
RM Reset macro
RZ Reset macro zero
TM Report macro contents
TZ Tell Macro Zero