

DIVA Automation  
Motion Controls

---

**User Manual for**

**DA1010**

**SuprChipM**

---

**Rev 1.1 30 September 2007**

## DIVA Automation

---



*INNOVATION IN MOTION*

© DIVA Automation

The contents of this manual and the intellectual property described herein are the creation and property of W. H. Speegle dba DIVA Automation.

803 Hermitage Dr., #211, Florence, AL 35630 USA

Tel +1-800-984-DIVA (3482) • Fax +1-877-471-3437

Mail [sales@divaauto.com](mailto:sales@divaauto.com)

# Contents

<b>Key features of the SuprChip M.....</b>	<b>4</b>	Define Integral limit .....	21
<b>Digital / Analog I/O :.....</b>	<b>5</b>	Get Integral Limit .....	21
Digital Inputs:.....	5	Define Following Error .....	22
Digital Outputs:.....	5	Define Baud Rate.....	22
Analog Output: .....	5	Reset Controller.....	22
Analog Input: .....	5	Echo Off.....	23
<b>Communication .....</b>	<b>6</b>	Echo On.....	23
RS-232 Settings .....	6	Motor On.....	23
Reply Termination .....	6	Motor Off .....	24
<b>DiMAC command types.....</b>	<b>7</b>	Abort Motion .....	24
Single Command.....	7	Move Relative.....	25
Compound Commands.....	7	Move Absolute.....	25
Macro Commands.....	7	Repeat.....	25
Sub-Commands.....	8	Find Edge.....	26
Control Characters: .....	8	Find Index .....	26
Address Selection .....	9	Define Home.....	26
<b>DiMAC command Reference.....</b>	<b>10</b>	Go Home.....	27
Tell Position .....	10	Wait Stop .....	27
Tell Target.....	10	Macro Definition.....	27
Tell Status.....	11	Macro Call.....	28
Move Status.....	12	Tell Macro .....	28
Tell Following Error .....	12	Reset Macro .....	28
Tell Board Address.....	13	Set Analog Output .....	29
Define Board Address.....	13	Incremental Analog Output .....	29
Update.....	13	Tell Analog Output.....	29
Define Velocity.....	14	Tell Analog Input.....	30
Get Velocity .....	14	Tell Channel.....	30
Define Acceleration .....	14	Channel Off.....	31
Get Acceleration.....	14	Channel On.....	31
Define P-Term.....	15	<b>Some examples:.....</b>	<b>32</b>
Get P-Term.....	15	<b>Command Short Form.....</b>	<b>34</b>
Define I-Term.....	15	<b>Command in alphabetic order.....</b>	<b>34</b>
Get I-Term.....	16		

## Key features of DA1010 SuprChip M

- A proprietary real-time operating system, DiMAC (DIVA Macro command system) provides a multitasking control environment that translates user-friendly two-letter mnemonic commands into a sequence of codes required to control the velocity, acceleration, position and torque of a DC motor and to integrate them into an overall synchronized control system.
- Supports two limit switches.
- Provides polarity control for index input.
- Provides extensible control language with over 100 commands with in-line syntax and limit checking.
- Adds support for storage and recall of servo parameter setups and user programs.
- Accepts parameter values in engineering units
- Supports multi-drop bus. Up to 32 SuprChip M controllers may share a single RS-232 channel, with individual addressing.
- Drives two-digit LED for command address and error code display.
- Low power operation from a single +5V power supply at 6mA.
- Supports multi-line LCD display.
- Up to eight analog inputs, 0-5 V, 8 bit resolution
- Up to two analog output, 0-5 V, 8 bit resolution, 2 mA.
- Up to 42 general purpose inputs/outputs. All I/O signals are dynamically programmable as input or output.
- All analog and digital I/O signals sampled and controlled by fully integrated commands.
- Motor and sequence synchronization easily achieved with external signals.
- High speed operation. I/O commands execute at speeds up to 100 KHz.
- Provides support for macro commands. User generated commands function as native commands.
- Presence of a defined Macro #0 provides automatic execution on power up.
- Two channels of 5 MHz multi-function counters provided.
- A 32-bit virtual accumulator implements several stack-type commands to support math operations.
- Operates as master, stand-alone or slave.

**Digital / Analog IO :****Digital Inputs:**

Number	Up to 42 (Note 1)
Type	High impedance CMOS
Voltage	5. VDC
I <sub>max</sub>	5 uA

**Digital Outputs:**

Number	Up to 42 (Note 1)
Type	(Note 1)
Voltage	5 VDC
I <sub>max</sub>	10 mA per channel

**Analog Output:**

Number	2
Resolution	8 bit
Voltage	0..5 V
Output impedance	2.5 K ohm

**Analog Input:**

Number	8
Resolution	8 bit 12.5 microsec +/- 1 bit
Voltage	0..5 V
Input current	0.5 micro A

Note 1: There are nine 8-bit ports available. All but two bits of four ports are used as a general purpose microprocessor bus for communication with any external memory that might be desired. The remaining 42 multi-purpose pins are available for general use.

Eight of these pins may be used as digital input or output, or as analog inputs. When interrogated as digital inputs, the threshold is 2.5 V. All other pins have TTL thresholds.

Two pins may be used as digital input or output, or as analog outputs.

Two pins may be used as digital input or output, or as synchronous or asynchronous serial interface.

Three pins may be used as digital input or output, or as a synchronous serial interface.

Two pins may be used as digital input or output, or as multifunction 5 MHz counter inputs.

Five pins may be used as digital input or output, or as external interrupts.

Two pins may be used as digital outputs, only.

Eight pins may be used as digital input or output and have N-channel open drain outputs. All others have CMOS tri-state active outputs. All pins are individually programmable in terms of function and direction.

# Communication

## RS-232 Settings

Baud rates	9600 / 19200 / 38400
Stop bits	1
Parity	None
Protocol	None

## How to communicate

The SuprChip M has a proprietary real-time operating system, DiMAC, (Diva Macro operating system) with an internal command interpreter. All data to or from the controller are ASCII strings.

A command string starts with a 2 letter mnemonic and dependent on the command, one or more parameters!

Over 100 commands are available for programming the SuprChip M.

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
- Address command** --

## Command Termination

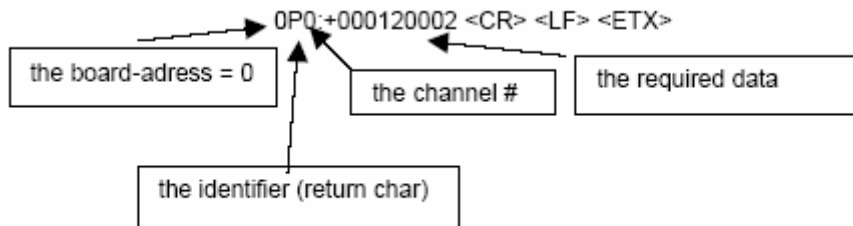
A command must be terminated by the *Carriage Data Identification Char = decimal 13 (<CR>)*  
A <CR> without any other character, repeats the last command!

## Reply Termination

All data sent from the SuprChip M are ASCII Strings. A report string starts with a command identifier which consists of:

- the board address,
- the one character identifier (Data Identification Char),
- the channel number,
- a colon,
- the requested data,
- the reply termination CR LF ETX (decimal 13, 10, 3 or hex 0D, 0A, 03)

Example: TP<CR>



# DiMAC Command Types

## Single Command

A **single command** is executed immediately after the carriage return (cr) 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, using the auto-repeat function of the terminal to send the carriage return continuously.

Both upper case and lower case characters are valid, and spaces are allowed.

## 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)
```

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 15 single commands.

Example:

```
ma0,ws,mr500<CR>  
ma0,ws,mr2000,ws,rp10<CR>
```

## 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, recalled and implemented by a macro number.

To use macros for programming the **SuprChip M** 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 (or EM3).

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 15 single commands.

## 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. If a macro is running, standard commands are not recognized until the macro is stopped.

Sub-commands are one character commands without any command termination.

A space character (h20 32) Stops a running macro.

Character	Hex value	Decimal	Command	Function
space	H20	32		Stops a running macro
'	H27	39	Tp	Reports the current position
?	H3F	63	Tf	Reports the following error
%	H25	37	Ts	Reports system status
!	H21	33	Ab	Stops motion (all controller addresses)
#	H23	35	tc	Reports input port status

## Control Characters:

Character	Hex value	Decimal	Function
CR	H0A	10	Command/reply termination
LF	H0D	13	Reply termination
ETX	H03	3	Reply termination
Ctrl A	H01	1	Address selection

## Address Selection

Up to 16 controllers can share one serial interface. The address selection is done by the command *Ctrl+A* + the controller address, defined by command *<db>*, all others are automatically de-addressed.

**Note: After power-up, the controller is not addressed! The address command must be sent before it will respond to commands!**

Example:	CtrlA5	address Controller #5
	tp	query position
	5p0:+000300	reply of controller #5
	CtrlA2	address Controller #2
	tp	query position
	2p0:+0002300	reply controller #2

There is a interesting feature which enables sending commands to all available controllers.  
The command *Ctrl-a+v+ standard command*  
Please do not use this command with reporting commands, the data will be chaotic!

Example:	CtrlAV DV2000	defines the velocity of all boards
	CtrlAV MR2000	all motors will move 2000 inc relative

There is an interesting feature which enables sending commands to all available controllers.

The command *Ctrl a+v+ standard command*

This command cannot be used with reporting commands, the data will be chaotic!

Example: CtrlAV DV2000 defines the velocity of all boards as 2,000 encoder counts/sec.

CtrlAV MR2000 all motors will move 2000 counts relative to the present position.

# DiMAC Command Reference

Data formats:

l	Long integer (32 bit)
ui	unsigned integer (24 bit)
w	word (16bit)
b	byte (8 bit)



## Tell position

Description

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

Parameter

Tell Position reports the absolute position in encoder increments. **TP** may be used to monitor m

Return

No

Index

Char P

Return Value

always 0

Example tp

l [increments]

0P0:+0000020002



## Tell target

Description

Reports target position in encoder increments. 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 plus or minus sign (+/-), if the position is greater than or less than the position defined as "home."

Parameter

no

Data Identification Char

T

Index

always 0

Return Value

l [increments]

Example tt

0T0:+0000020002



## Tell Status

Description

When the **TS** command is given, the status of the system as well as the motion and limit switches is reported. The answer is a string with 7 Hex Bytes (B\_1 .. B\_7)

If an error occurs (LED Display) the error will be reset by this command!

Parameter  
Data Identification Char  
Index

no  
S  
always 0

b1 b2 b3 b4 b5 b6 b7  
Byte\_1: Motion flags Status Byte

Bit 0	Busy
Bit 1	Command error
Bit 2	Trajectory complete
Bit 3	Index found
Bit 4	Position limit exceeded
Bit 5	Excessive position error
Bit 6	Breakpoint reached
Bit 7	Motor off

Byte\_2: Operational Flags 1

Bit 0	
Bit 1	
Bit 2	Wait in process
Bit 3	Command error
Bit 4	Leading zero suppression active
Bit 5	Macro command called
Bit 6	Leading zero suppression disabled
Bit 7	Echo on

Byte\_3: Operational Flags 2 ( only for factory use)

Bit 0	
Bit 1	
Bit 2	
Bit 3	
Bit 4	
Bit 5	
Bit 6	
Bit 7	

Byte\_4: Motion Flags

Bit 0	
Bit 1	Polarity of dynamic error
Bit 2	Polarity of move direction 1= positive
Bit 3	Move Error
Bit 4	Polarity of move at start of deceleration
Bit 5	Velocity mode
Bit 6	Excessive position error
Bit 7	

Byte 5: Limit switch status

Bit 0	Limit Switch enable flag
Bit 1	
Bit 2	
Bit 3	
Bit 4	Limit Switch active state 1 = high
Bit 5	Find Edge in process
Bit 6	Brake on Flag
Bit 7	Find limit switch flag

Byte 6: Limit Switch inputs

Bit 0	Limit reverse active (typical)
Bit 1	Limit forward active (typical)
Bit 2	Not assigned
Bit 3	Not assigned
Bit 4	Not assigned
Bit 5	Not assigned
Bit 6	Not assigned
Bit 7	Not assigned

Byte 7: Error Codes

01	command not available
02	first command character must be a letter
03	no command
04	negative not allowed
05	character following command must be a number
06	value too large
07	value too small
08	continuation character must be a comma
09	command buffer overflow
0A	macro storage overflow



## Tell following error

Description

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 commanded velocity is higher than that 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.

Parameter  
Data Identification Char  
Index  
Return Value  
Example

no  
F  
always 0  
I [increments]  
Tf 0F0:+0000000022



## Tell board address

Description  
Parameter  
Data Identification Char  
Index  
Return Value  
Example

Reports the board address  
no  
B  
always 0  
number 0.. F  
tb  
0F0:+0000000022



## Define board address = *n*

Description  
Parameter  
Data Identification Char  
Index  
Return Value  
Example

Defines the board address  
the new address  
no  
no  
no  
db5



## Move status

Description  
Parameter  
Data Identification Char  
Return Value  
Index  
Example

Reports only Byte 1 of the **TS** command  
no  
M  
b  
always 1  
ms  
0M1:4C



## Update parameter storage

Description

Stores all parameters like velocity and filter parameters in the nonvolatile memory. When the SuprChip M powers up these parameters will be active!

Parameter  
Return Value  
Example

no  
no  
ud

**dv<sub>n</sub>**

Description

Parameter  
 Parameter range  
 Return Value  
 Example

**Define velocity = *n***

Defines the desired velocity for any movement. Causes the motor to run at velocity *n* in subsequent motion commands. *N* is specified 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 1 to 1 MHz. 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 25,000 to 250,000 counts/second as a maximum velocity.

I [increments/s]  
 0..1 000 000  
 no  
 dv20000

**gv**

Description

Data Identification Char  
 Index  
 Return Value  
 Example

**Get velocity**

Returns the current programmed velocity, (defined with command *dv*)

Y  
 always 0  
 I [increments/s]  
 gv  
 0Y0:+0000020000

**da<sub>n</sub>**

Description

Parameter  
 Parameter range  
 Return Value  
 Example

**Define acceleration = *n***

Defines the desired acceleration *i* for any movement. The maximum value is primarily limited by the mechanical system (the inertia of the system)

I [increments/s<sup>2</sup>]  
 0..10 000 000  
 no  
 da200000

**ga**

Description

Data Identification Char  
 Index  
 Return Value  
 Example

**Get acceleration**

Returns the current programmed acceleration, (defined with command *da*)

A  
 always 0  
 I [increments/s<sup>2</sup>]  
 ga  
 0L0:+0000200000

**dp<sub>n</sub>**

Description

Parameter  
Parameter range  
Return Value  
Example**Define P-Term = *n***

Defines the P-Term of the PID filter. 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 small error value causes a proportionally larger motor current driving the motor towards the target. The default gain value is usually stable. The optimum value depends on friction, inertia, motor power, amplifier gain 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 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 gain must be reduced until the oscillation stops. Further improvement must be accomplished through use of the D and I terms.

w  
0..32767  
no  
dp800

**gp**Description  
Data Identification Char  
Index  
Return Value  
Example**Get P-Term**

Returns the current programmed P-term, (defined with command *dp*)

G  
always 0  
w  
gp  
0G0:+0000000800

**di<sub>n</sub>**

Description

Parameter  
Parameter range  
Return Value  
Example**Define I-Term = *n***

Defines the I-Term of the PID filter.  
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.

w  
0..32767  
no  
di200

**gi****Get I-Term**

Description

Returns the current programmed integral term, (defined with command *di*)

Data Identification Char

I

Index

0

Return Value

w

Example

gi  
010:+0000000400**dd<sub>n</sub>****Define D-Term = *n***

Description

Defines the D-Term of the PID filter.  
Sets the gain to be applied to the derivative term in the PID algorithm. The primary function of this term is to overcome inertia induced dynamic errors.**gd****Get D-Term**

Description

Returns the current programmed derivative term, (defined with command *dd*)**dl<sub>n</sub>****Define integral limit = *n***

Description

Defines the integral limit of the PID filter.

Parameter

n

Parameter range

0..32767

Return Value

no

Example

dl2000

**gl****Get integral limit**

Description

Data Identification Char  
 Return Value  
 Index  
 Example  
 Returns the current  
 programmed integral limit,  
 (defined with command *d*)

L  
 w  
 always 0  
 gl  
 OL0:+0000002000

**de<sub>n</sub>**

**Define following error limit = *n***

Description

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 *DE* command to limit following error to a value slightly above that required for normal operation.

Parameter  
 Parameter range  
 Return Value  
 Example

w  
 0..32767  
 no  
 de2000

**br<sub>n</sub>**

**Define Baud rate**

Description

The SuprChip M is programmed to communicate with 3 standard Baud rates. Other rates are possible with minor changes.

Parameter  
 Parameter range  
 Return Value  
 Example

0 : 9600  
 1: 19200  
 2: 38400  
 b  
 0..2  
 no  
 br0 ; 9600 is active

**rt**

**Reset controller**

Description

Restarts the internal firmware operation as if from a power off condition. All default values are restored. If Macro 0 exists, it will be executed.

Parameter  
 Return Value  
 Example

no  
 no  
 rt

**ef**

## Echo off

Description

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.

Parameter  
Return Value  
Example

no  
no  
ef



## Echo on

Description

Enables echoing of command characters as they are entered. Each character received is echoed unchanged. This is a very useful feature when the SuprChip M is being controlled manually from a terminal..

Parameter  
Return Value  
Example

no  
no  
en



## Motor on

Description

This is the normal system control mode, where the SuprChip M controls the axis position continuously. Any deviation between actual and target position causes the motor to be driven toward the target and possibly with full force, depending on the distance moved during the motor off condition. Use caution when turning the motor back on after a motor off condition. The SuprChip M remembers its position when it received the *MF* command and it will try to return there unless the target position is redefined.

Parameter  
Return Value  
Example

no  
no  
mn



## Motor off

Description

When this command is issued, the motor is no longer held in position control and may be moved freely. 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 in the *MF* status and may be reported by the *TP* command.

Use caution when turning the motor back on (*MN*). The target position is still the same as when the *MF* command was issued and it will try to return there unless the target position is redefined.

Parameter  
Return Value  
Example

no  
no  
mf



## Abort motion immediately

Description	This command stops a motion. The target position is changed to be equal to the present position.
Parameter	n=0 stop abrupt n=1 decelerated stop
Parameter range	n 0 ..1
Return Value	no
Example	ab1 decelerated stop



## Stop motion with deceleration

Description	This command stops a motion. The target position is changed to be equal to the present position.
Parameter	no
Return Value	no
Example	st decelerated stop



## Move relative $n$

Description	This command generates a motion of relative distance of $n$ counts in the specified direction from the actual motor position. $i$ may be either a positive or negative number up to a total target position +/-1 073 741 843
Parameter	$i$
Parameter range	Depends on the absolute target. The cumulative total should not exceed the maximum position +/-1 073 741 843 !
Return Value	no
Example	mr5000



## Move to absolute position $n$

Description	This command generates a motion to the absolute position $i$ . The zero, or home position, may be defined by the <b>DH</b> (Define Home) statement. If not otherwise defined, it is the position where the controller was when powered on.
Parameter	$i$
Parameter range	+/-1 073 741 843
Return Value	no
Example	ma2000

**rp<sub>n</sub>**

Description

Parameter  
Parameter range  
Return Value  
Example

## Repeat *n* times

This command causes the command string to repeat *i* times. If *i* is not specified, the commands are repeated 65,535 times. The repeat loop may be interrupted by transferring the space character. This character may not be the first character of a new command because it will be discarded. (To repeat forever, use two **RP** commands in sequence.)

n  
0..65535  
no  
ma0,ws,ma2000,ws,rp100

**fe<sub>n</sub>**

Description

Parameter  
Parameter range  
Return Value  
Example

## Find edge

This command is used to initialize the system at a given position. The motor runs at the programmed speed until a change of state occurs on the limit input line. The direction of movement is selected by *n*.

Example: **FE0** (rtn) : Causes motor to move in a positive direction until the E2 input is activated.

**FE1** (rtn) : Causes motor to move in a negative direction until the E1 input is activated.

**FE4** (rtn) : Causes motor to move in a positive direction until the E1 input is deactivated.

n  
0,1,4  
no  
fe1 search for E1 in negative direction

**fi**

Description

Parameter  
Parameter Range  
Return Value  
Example

## Find index

This command is used to define a very exact homing position. The motor runs at the programmed speed until the index of the encoder occurs.

Bit 3 indicates if the index occurs (status & \$08)

Normal rotary encoders have only one index per motor revolution.

no  
fi

**dh**Description  
Parameter  
Return Value  
Example

## Define home

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

no  
no  
dh

**gh**

Description

Parameter  
Return Value  
Example

## Go home

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

no  
no  
gh

**ws**

Description

Parameter  
Parameter range  
Return Value  
Example

## Wait stop

Wait until the calculated trajectory has reached the end, then wait another  $n$  milliseconds before the execution of the next command. Please note that this command will execute without regard to the actual motor position. The test for completion is based on the theoretical position of the motor. If the motor has been prevented from reaching the desired position for any reason, there is no automatic reporting of the error. Without parameter the wait time is 1000 ms.

n  
0..65535  
no  
mr2000,ws500,ma0

**bf**

Definition

## Brake off

Sets the port assigned to the external brake to the inactive state.

**bn**

Definition

## Brake on

Sets the port assigned to the external brake to the active state.

**cp<sub>n</sub>**

**Channel pattern =  $n$** 

Definition

Sends a complete byte of data to eight output ports.

**Cycle between programmed limits**

Definition

Starts a routine that causes the motor to cycle between programmable position limits until stopped. The selected positions are set by the DX and DY commands and the delay at each position is set by the DW command.

**Define String # $n$** 

Definition

Stores the string of ASCII characters entered after the command until a carriage return character is received. The number  $n$  is assigned to the string, which can be recalled with the PS command. Ds5(cr)This is a test.(cr) Will assign the string "This is a test." as string #5. This string can be recalled with PS5 the command. String storage is limited only by the available system RAM.

**Dump data**

Definition

DU is a composite command provided to speed up typical reporting communications for host programs. The DU command is equivalent to sending separate commands for TT,TP,TD,TS.

**Define delay for CY command =  $n$** 

Definition

Sets the number of ms to dwell at each breakpoint during execution of the CY command.

**dx<sub>n</sub>**

Definition

**Define positive limit for CY = *n***

Sets the value of the most positive breakpoint position during execution of the CY command. May be greater or less than 0, but must be more positive than the value entered with the DY command.

**dy<sub>n</sub>**

Definition

**Define negative limit for CY = *n***

Sets the value of the most negative breakpoint position during execution of the CY command. May be greater or less than 0, but must be more negative than the value entered with the DY command.

**em<sub>n</sub>**

Definition

**Execute macro *n***

Calls macro *n*. This command may be used interchangeably with the MC command.

**gb**

Definition

**Get values of DX and DY**

Reports values set by the DX and DY commands.

**jg**

Definition

**Jog by value in register**

Equivalent to a Move Relative command, with the target position modified by the value stored with the SJ command.

**lf****Limit switch operation off**

Disables limit switch operation.

**ln**

Definition

**Limit switch operation on**

Enables limit switch operation.

**lh**

Definition

**Limit switch active state high**

Sets the active state for limit switch operation to logic high. Typically set during initial system setup and stored with the UD command for automatic setup during subsequent operation.

**ll**

Definition

**Limit switch active state low**

Sets the active state for limit switch operation to logic low. Typically set during initial system setup and stored with the UD command for automatic setup during subsequent operation.

**ps<sub>n</sub>**

Definition

**Print string #*n***Reports the string stored by the DS command at position *n*.**sj<sub>n</sub>**

Definition

**Set jog register = *n***

Stores *n* in the jog register. The JG command uses this value to execute an MR*n* command.

A gray rectangular box containing the white text 'rg'.

## Reset string space

Definition

Clears the entire string space. This is the only method that can be used to remove any string.

A gray rectangular box containing the white text 'sin'.

## Set polarity of the index signal

Definition

Outputs a signal to control an external circuit used for controlling the polarity of the index signal

A gray rectangular box containing the white text 'td'.

## Tell dynamic target

Definition

Reports the output of the trajectory generator, which describes the position at which the motor should be at the current time. The value reported by TD will vary from the start position to the target position during the move.

A gray rectangular box containing the white text 'te'.

## Tell error

Definition

Reports the distance to the target position. This value decreases throughout the move.

A grey square box containing the lowercase letters 'ti' in white.

## Tell iteration number

Definition

Reports the value of the iteration counter used by the RP command. This value decreases with each iteration of the command line. The initial value may be misleading because the command is typically executed before the first RP command which sets the iteration counter. TT,TI,RP5 will report values of ??, 5, 4, 3, 2, 1, 0 as the sequence is performed. The ?? report is the contents of the iteration counter residue from the previous command.

A grey square box containing the lowercase letters 'tj' in white.

## Tell jog value

Definition

Reports the value stored by the SJ command.

A grey square box containing the lowercase letters 'tv' followed by a subscript 'n' in white.

## Tell measured velocity over $n$ ms.

Definition

Reports the difference between two position samples separated by  $n$  ms.

A grey square box containing the lowercase letters 'tz' in white.

## Tell macro zero

Definition

Reports the contents of MC0. If no MC0 has been defined, nothing is returned.

A grey square box containing the lowercase letters 'wa' followed by a subscript 'n' in white.

## Wait $n$ ms

Definition

Delays  $n$  ms before proceeding to the next command.



## Increment output *a* by *n* steps

Definition

Adds the value of *n* to the output register *a*. The value may be positive or negative.



## Stop

Definition

Stops motion using the currently programmed value of acceleration.

# Macro Commands

## md<sub>n</sub>

Description

Parameter  
 Parameter range  
 Return Value  
 Example

### Macro definition #*n*

This command is used to define a new macro command. Any duplication of numbers will simply result in the over writing 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. Macro 0 is an autostart macro, executed automatically after booting the controller  
 macro number + command string  
 0..14 (macro number)  
 no  
 md1,ma0,ws,ma2000,ws

## mc<sub>n</sub>

Description

Parameter  
 Parameter range  
 Return Value  
 Example

### Macro call #*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. A running macro can be stopped by sending the <space> character.  
 dec=32 hex =\$20  
*n*, macro number  
 0.14  
 no  
 mc1

## tm<sub>n</sub>

Description

Parameter  
 Return Value  
 Example

### Tell macros

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 to display all previously stored macro commands. If *n* = 0 or, if *n* is not specified, all macros will be displayed.

*n*, macro number  
 0..14  
 the contend of the macro(s)  
 tm1  
 mc1,mr555,ws,gh

**rm<sub>n</sub>**

Description  
 Parameter  
 Example

**Reset macro #*n***

Resets macro memory  
 0..14(macro number)  
 rm reset all macros  
 rm 1 reset only macro 1

**so<sub>n</sub>**

Description  
 Parameter  
 Example

**Set analog output = *n***

Sets the analog output to a value. The output is a 0 to 5 V 8 bit output. One bit is 19.5 mV b -> value of the output 8 bit  
 0..255  
 so255  
 the output is 5 V

**io<sub>n</sub>**

Description  
 Parameter  
 Parameter range  
 Example

**Increment analog output by *n***

Increments the analog output . The output is a 0 to 5 V 8 bit output. One bit is 19.5 mV.  
 l -> incremental value  
 -255..255  
 io2 ; new analog value = old value + 2  
 mc2,io1,ws1,rp ; generates an analog sawtooth

**to**

Description  
 Parameter  
 Data Identification Char  
 Index  
 Return Value  
 Example

**Tell analog output setting**

Returns the value of the currently commanded output  
 no  
 o  
 always 0  
 b [0..255]  
 to  
 0o0:255



Description  
Parameter

Data Identification Char  
Index  
Return Value  
Example

## Tell analog input #*n*

Returns the actual value of the analog inputs.  
b channel [0 ..2]  
0 = report both inputs  
1..2 = reports channel #  
A  
b [0..2]  
n m -> value of the analog inputs 8 bit  
n = channel 1  
m = channel 2  
0..255  
ta  
0A0: 127 230  
ta1  
0A1 :127  
ta2  
0A2 :230



Description  
Parameter

Data Identification Char

Index  
Return Value

Example

## Tell channel #*n*

Returns the actual status of the digital inputs and outputs.  
b [ 0..4]  
0 = report all outputs and inputs  
1..4 = report input channel #  
H  
0 ..4  
n m -> value of the inputs /outputs 8 bit  
n = outputs  
m = inputs  
tc  
0H0: 0 255 ; all outputs and inputs not active  
tc1  
0H1: 1 ; reports channel 1



Description

Parameter

Return Value  
Example

## Channel off #*n*

Reset digital output(s)  
b, 0..4  
0 = all outputs  
1..4 = only channel #  
no  
cf 0 or cf resets all 4 digital outputs  
cf1 resets channel 1



Description  
Parameter

Return Value  
Example

## Channel on #*n*

Set digital output (s)  
b, 0..4  
0 = all outputs  
1..4 =only channel #  
no  
cn0 or cn sets all 4 digital outputs  
cn1 sets channel 1



Description  
Parameter

Return value  
Example

## Execute if port *n* is on

Conditional execution of remainder of compound command is port *n* is active.  
b, 0..4  
0 = all outputs  
1..4 =only channel #  
no  
xn3,mc12,wa250,rp  
(Execute MC12 every 250 ms if channel 3 is high.)



Description  
Parameter

Return value  
Example

## Execute if port *n* is off

Conditional execution of remainder of compound command is port *n* is active  
b, 0..4  
0 = all outputs  
1..4 =only channel #  
no  
xf3,mc12,wa250,rp  
(Execute MC12 every 250 ms if channel 3 is low.)



Description

Parameter  
Return value  
Example

## Execute routine at address *n*

Executes routine from memory, starting at the address pointed to by the contents of address *n*.  
b, 0..4  
no  
(.)

## Breakpoint commands

**bd<sub>n</sub>**

### Breakpoint definition = *n*

Definition

Sets the position breakpoint register.

**bi<sub>n</sub>**

### Breakpoint increment = *n*

Definition

Sets the position breakpoint increment. Each time the previous breakpoint is achieved, the increment is automatically added to the breakpoint register.

**pw<sub>n</sub>**

### Breakpoint pulse width = *n*

Definition

Sets the width of the pulse that will be output on the selected port when the breakpoint position is achieved.

**sb<sub>n</sub>**

### Set breakpoint port = *n*

Definition

Selects the position breakpoint output port.

## Housekeeping commands

**CS**

### Calculate checksum of program area

Definition

Adds all program data in the firmware to a running total and reports the least significant byte..

**he**

### Help

Definition

Reports the mnemonic of all installed commands..

**hm**

### Hexadecimal mode

Definition

Selects hexadecimal mode for all subsequent input and output.

**dm**

### Decimal mode

Definition

Selects decimal mode for all subsequent input and output. This is the default mode.

**mo**

### Enter monitor/debugger

Definition

Enters the DiMAC monitor/debugger. Provides direct access to all registers and memory.

**SC<sub>*n*</sub>**

### Selects board address *n*

Definition

Selects the board address. This command is equivalent to the ctrl-A sequence used through the interface to select a board address. The primary purpose of this command is to perform and internal address selection when MC0 is used for automatic operation. Without selecting the address, no output operations can be performed.

**sn<sub>n</sub>****Store serial number *n***

Definition

Stores a user generated serial number. This command is useful for tracking embedded products in system applications.

**rn****Report serial number**

Definition

Reports the serial number stored by the SN command.

**ve****Report firmware version number**

Definition

Reports the firmware revision level and copyright notice.

**xr<sub>n</sub>****Execute routine at address *n***

Definition

Transfers control to the firmware routine located at address *n*.

## Virtual accumulator commands

The SuprChip M provides an on-board math capability for ease of calculation of related variables and parameters. For example, using these commands, it is very easy to construct a macro command that reads a potentiometer input, subtracts an offset, multiplies by a scale factor and applies the result of the calculation as a gain, a velocity or a distance to be moved.

The code for this operation might be: MD15,GG3,AS123,AM432,

In another example, the position might be multiplied by a scale factor that converts encoder counts into engineering units, such as mm.

The virtual accumulator is a set of 32-bit registers in RAM that can be modified by various commands to load and perform arithmetic operations. Commands such as AAn (Add n to accumulator) leave the modified result on the accumulator. This location might be referred to as TOS (top of stack) in other languages. If the original quantity should be preserved, then it must be stored prior to the calculation.



### Add *n* to Accumulator

Description

Adds the value of *n* to the contents of the virtual accumulator and stores the sum in the virtual accumulator. The original contents of the virtual accumulator are lost.

Parameter  
Return value  
Example

I, 0..2,000,000,000  
no  
aa12345



### Load Accumulator from address *n*

Definition

Loads the virtual accumulator with a byte at address *n*.



### Multiply Accumulator by *n*

Definition

Multiplies the contents of the virtual accumulator by *n*. The result is stored in the virtual accumulator.

**as<sub>n</sub>**

Definition

**Subtract *n* from Accumulator**

Subtracts the value of *n* from the virtual accumulator and stores the result in the virtual accumulator.

**av**

Definition

**Take absolute value of Accumulator**

Extracts the absolute value of the virtual accumulator and stores the result in the virtual accumulator.

**gg<sub>n</sub>**

Definition

**Get A/D channel *n* into Accumulator**

Reads the value of the analog input at channel *n* and stores the result in the virtual accumulator.

**la<sub>n</sub>**

Definition

**Load *n* into Accumulator**

Loads the value of *n* into the virtual accumulator.

**li<sub>n</sub>**

Definition

**Load Accumulator indirect through *n***

Loads the virtual accumulator with the data found at the address pointed to by *n*.

**ra**

Definition

**Report Accumulator contents**

Reports the contents of the virtual accumulator.



Definition

## **Divide Accumulator by $n$**

Divides the contents of the virtual accumulator by  $n$  and stores the result in the virtual accumulator.

## Some examples:

### Homing

DV1000  
FE1  
WS  
DV500  
FE4  
WS  
DH

Set relatively slow velocity to seek the E1 switch  
Search in reverse direction  
Wait until after move complete for 1000 ms  
Set very slow velocity  
Search for toggle of the E1 switch in forward direction  
Wait until after move complete for 1000 ms  
Define home, actual position = 0

### Demo

MA0  
WS250  
MR500  
WS  
RP50

Move absolute to 0 position (equivalent to GH command)  
Wait until after move complete for 250 ms  
Move relative 500 increments  
Wait until after move complete for 1000 ms (default value)  
Repeat the line 50 times

## Some compound commands

DV1000,FE1,WS,DV500,FE4,WS,DH (Performs homing sequence as shown above, as a single command.)

## Using macro commands

MD17,DV10000,FE1,WS,DV500,FE4,WS,DH (Creates a macro that performs the homing sequence.) To perform this operation automatically at power on, the number MD0 could be used, or MD0 could be defined with a call to MC17 plus other functions.

MD0,SC4,VE,CS,MC17 will run automatically at power on. It will first address itself (assuming address 4 in this example), then display the firmware version, followed by a calculation and display of the firmware checksum and the homing sequence of MC17. Without the SC command, the board would not be addressed and would not display any data, but the checksum would be calculated and the homing sequence would run.

## Some more macros

**Macro 2:** This macro calls macros 10, 11 and 12, then returns to the home position, waits 500 ms after the end of the trajectory and repeats 9 times (for a total of 10 executions)

MD2, EM10,EM11,EM12,GH,WS500,RP9

**Macro 10:** The motor moves 20000 increments relative to the current position, when the trajectory has been complete for 1 ms, output 1 goes high for 200 ms, then goes low.

MD10,MR20000,WS1,CN1,WA200,CF1

**Macro 11:** The motor moves 500 increments relative, wait for 100 ms after end of trajectory and repeat the macro 10 times (for a total of 11 executions).

MD11, MR500,WS100,RP10

**Macro 12:** Similar to macro 11 with only 6 executions

MD12, MR500,WS100,RP5

(Please note that the TM command uses a fixed format for easier handling by a program, but may be confusing for a human. Also note that both MC and EM may be used to call a macro, but TM and TZ report EM regardless of which command was entered.)

MD12, MR-0000000500,WS+0000000100,RP+0000000005

# Command Short Form

Read command	Data ID	Set command
TS Tell Status	S:	
HE Print list of commands (Help)		
TE Tell Error (distance from target)	E:	
TP Tell Position	P:	
TT Tell Target	T:	
TF Tell Following error (distance between position and dynamic target)	F:	
TD Tell Dynamic target	N:	
TV Tell actual Velocity	V:	
GV Get Velocity	Y:	DVn Define Velocity
GA Get Acceleration	A:	Dan Define Acceleration
GP Get Proportional gain	G:	DPn Define Proportional gain
GI Get Integral gain	I:	DIn Define Integral gain
GD Get Derivative gain	D:	DDn Define Derivative gain
GL Get Integral Limit	L:	DLn Define integral Limit
		DEn Define maximum following Error
VE Display Version number		
TI Tell the Iteration number	X:	
TB Tell Board address	B:	DBn Define board address
CS Perform self-test Checksum	C:	
		LN Limit switch operation on
		LF Limit switch operation off
		LL Limits switches active low
		LH Limit switches active high
		UD Update
		DM Input and output in Decimal format mode
		HM Input and output in Hex format mode
		RT Reset all parameters to default and do power-up start
		EF Turn Echo off to RS-232 port
		EN Turn Echo on to RS-232 port
		AB Abort Motion
		MN Motor On
		MF Motor off
		Wan Wait specified time
		MRn Move Relative
		Man Move Absolute
		DH Define Home
		GH Go Home
		RPn Repeat from beginning of line
		WSn Wait until after end of trajectory
		FEn Find Edge
		FI Find Index
TM Tell Macro (1..31)		MDn Macro Definition
TZ Tell Macro Zero		
		MCn Macro Command
		EMn Execute Macro (alias for MC)
		RMn Reset Macro
TO Tell Analog Output	O:	SOn Set Analog Output
TA Tell Analog Input	A	
TC Tell Channel	H	CFn Channel Off
		CNn Channel on
RN Reports the serial number.		SIn Sets polarity of the index signal.

DU Dumps suite of data, equivalent to TT,TP,TD,TS.	T,P,D,S	
RA Report value of virtual accumulator.		AA <sub>n</sub> Add n to virtual accumulator.
		LA <sub>n</sub> Load value of n into virtual accumulator.
		AM <sub>n</sub> Multiply virtual accumulator by n.
		AL <sub>n</sub> Load virtual accumulator from memory address n.
		AS <sub>n</sub> Subtract n from virtual accumulator.
		AD <sub>n</sub> Divide virtual accumulator by n.
		GG <sub>n</sub> Get value of A/D channel n into virtual accumulator.
		PW <sub>n</sub> Set width of output pulse to n when breakpoint is reached.
		BL <sub>n</sub> Set position breakpoint increment = n.
		BD <sub>n</sub> Set position breakpoint at position n.
		SB <sub>n</sub> Set position breakpoint output port = n.
		CP <sub>n</sub> Byte wide loading of output ports.
		XF <sub>n</sub> Execute remainder of command line if port n is inactive.
		XN <sub>n</sub> Execute remainder of command line if port n is active.
		CY Cycle between programmed position limits.
		MO Enters debug/monitor.
		LI <sub>n</sub> Load virtual accumulator with data pointed to by address n.
		XI <sub>n</sub> Executes an internal sub-routine pointed to by address n.
		SN <sub>n</sub> Stores a serial number in external memory.
		SC <sub>n</sub> Selects the board address.
		BF Sets the brake signal to inactive.
		BN Sets the brake signal to active.
		PM Sets position mode.
		VM Sets velocity mode.
		AV Take absolute value of virtual accumulator.
		AX <sub>n</sub> Execute routine at address n.
		DN Enable continuous position display.
		DFD Disable continuous position display.
TJ Report value of jog register.		SJ <sub>n</sub> Set jog register = n.
		JG Jog by value of register.
GW Report value stored by DW.		DW <sub>n</sub> Set value of delay used in CY command = n.
GB Report DX and DY values.		DX <sub>n</sub> Set positive limit of travel used by CY command = n.
		DY <sub>n</sub> Set negative limit of travel used by CY command = n.
		RG Reset string storage.
PS <sub>n</sub> Print string #n.		DS <sub>a</sub> Define string = all characters received before next carriage return.
		BR <sub>n</sub> Baud rate
		IO <sub>n</sub> Increment output
		ST Stop motion

## DiSOS Stand-alone Operating System

It is possible to completely control the SuprChip M through the use of only two analog input port pins. One input decodes a multi-level signal from a resistor array and the other is digitized as a value selector. Both inputs can be used simultaneously to signal a third function.

There are 65 operating modes that can be selected by use of these two inputs. The currently selected operating mode is displayed on the two-digit LED that is used to display board address and error codes when used under control of a host device.

Both the SOS and the DiMAC operating systems can be active simultaneously without conflict. Values set with the external switch array can be reported through the DiMAC serial interface and register values for gains, velocity, jog size, etc., can be set more easily through this interface, with subsequent operations performed completely off line.

### DiSOS Stand-alone Operating Modes

The DiSOS Stand-alone Operating System program has been defined with the modes of operation listed below.

On power up, the operating mode goes to 64. To select other modes, hold both S1 and S2 for 500 mS. The LED blinks rapidly to indicate that the operating mode can now be selected by using S1 and S2 to increment and decrement the mode number. When the desired mode is displayed, press both S1 and S2 again to enter that mode. If it is a single operation such as GH, no other action is required.

Press both S1 and S2 again to select other operating modes. If it is a continuous type of operation, such as incrementing and decrementing the velocity, then control remains in the same op mode until S1 and S2 are pressed again simultaneously. Until this is done, S1 will cause an increment and S2 a decrement.

All switches require an actuation duration of several milliseconds in order to reject accidental multiple actuations.

Operating mode	Operation	S1	S2	S1&S2	Potentiometer
0	Jog by pot setting	Pos	Neg	Change op mode	Logarithmic table
1	MN/MF	MN	MF	Change op mode	
2	GH Go Home			Change op mode	
3	DH Define Home			Change op mode	
4	Jog by jog register	Pos	Neg	Change op mode	
5	Inc/dec jog register by 1	Inc	Dec	Change op mode	
6	Inc/dec jog register by 10	Inc	Dec	Change op mode	
7	Inc/dec jog register by 100	Inc	Dec	Change op mode	
8	Inc/dec jog register by 1000	Inc	Dec	Change op mode	
9	Inc/dec jog register by 10000	Inc	Dec	Change op mode	
10	Set velocity by pot			Set value	% of factory setting
11	Set acceleration by pot			Set value	% of factory setting
12	UD Update permanent storage			Activate	
13	Set jog register = 100			Activate	
14	Cycle between limit switches			Activate	
15	Cycle between software limits			Activate	
16	Set positive software limit = position			Activate	
17	Set negative software limit = position			Activate	
18	Set WS by pot for op modes 14, 15			Activate	% of 9.9 seconds
19	Restore velocity to EEPROM			Activate	
20	Inc/dec velocity by 1	Inc	Dec	Activate	
21	Inc/dec velocity by 10	Inc	Dec	Activate	
22	Inc/dec velocity by 100	Inc	Dec	Activate	
23	Inc/dec velocity by 1000	Inc	Dec	Activate	
24	Inc/dec velocity by 10000	Inc	Dec	Activate	
25	Toggle external display on/off			Activate	
26	Inc/dec acceleration by 10	Inc	Dec	Activate	

27	Inc/dec acceleration by 100	Inc	Dec	Activate	
28	Inc/dec acceleration by 1000	Inc	Dec	Activate	
29	Inc/dec acceleration by 10000	Inc	Dec	Activate	
30	DP Set Pgain by pot			Activate	% of factory setting
31	DI Set Igain by pot			Activate	% of factory setting
32	DD Set Dgain by pot			Activate	% of factory setting
33	DL Set Ilimit by pot			Activate	% of factory setting
34	RT Reset to factory settings/clear MC0			Activate	
35	Inc/dec Pgain by 1	Inc	Dec	Activate	
36	Inc/dec Pgain by 10	Inc	Dec	Activate	
37	Inc/dec Pgain by 100	Inc	Dec	Activate	
38	Inc/dec Igain by 1	Inc	Dec	Activate	
39	Inc/dec Igain by 10	Inc	Dec	Activate	
40	Inc/dec Igain by 100	Inc	Dec	Activate	
41	Inc/dec Dgain by 1	Inc	Dec	Activate	
42	Inc/dec Dgain by 10	Inc	Dec	Activate	
43	Inc/dec Dgain by 100	Inc	Dec	Activate	
44	Inc/dec Ilimit by 1	Inc	Dec	Activate	
45	Inc/dec Ilimit by 10	Inc	Dec	Activate	
46	Inc/dec Ilimit by 100	Inc	Dec	Activate	
47	FE0/FE1	FE0	FE1	Change op mode	
48	FI			Activate	
49	MC0 <b>Execute macro #0</b>			Activate	
50	MC1 <b>Execute macro #1</b>			Activate	
51	MC2			Activate	
52	MC3			Activate	
53	MC4			Activate	
54	MC5			Activate	
55	MC6			Activate	
56	MC7			Activate	
57	MC8			Activate	
58	MC9			Activate	
59	MC10			Activate	
60	MC11			Activate	
61	MC12			Activate	
62	MC13			Activate	
63	MC14			Activate	
64	Slew with velocity control by pot	Pos	Neg	Change op mode	% of setting

## Alphabetical order

AAn	Add n to virtual accumulator.	JG	Jog by value of register.
AB	Abort Motion	LAn	Load value of n into virtual accumulator.
ADn	Divide virtual accumulator by n.	LF	Limit switch operation off
ALn	Load virtual accumulator from memory address n.	LH	Limit switches active high
AMn	Multiply virtual accumulator by n.	LIn	Load virtual accumulator with data pointed to by address n.
ASn	Subtract n from virtual accumulator.	LL	Limit switches active low
AV	Take absolute value of virtual accumulator.	LN	Limit switch operation on
AXn	Execute routine at address n.	MA	Move Absolute
BDn	Set position breakpoint at position n.	MC	Macro Command
BF	Sets the brake signal to inactive.	MD	Macro Definition
Bin.	Set position breakpoint increment = n	MF	Motor off
BN	Sets the brake signal to active.	MN	Motor On
BR	Baud rate		
CF	Channel OfF	MO	Enters debug/monitor.
CN	Channel oN	MS	Move Status
CPn	Byte wide loading of output ports.	MR	Move Relative
CS	Perform self-test CheckSum	PM	Sets position mode.
CY	Cycle between programmed position limits.	PSn	Print string #n.
DA	Define Acceleration	PWn	Set width of output pulse when breakpoint is reached to n.
DB	Define board address	RA	Report value of virtual accumulator.
DD	Define Derivative gain	RG	Reset string storage.
DE	Define maximum following Error	RM	Reset Macro
DF	Disable continuous position display.	RNn	Reports the serial number.
DH	Define Home	RP	Repeat from beginning of line
DI	Define Integral gain	RT	Reset all parameters to default and do power-up start
DL	Define integral Limit	SBn	Set position breakpoint output port = n.
DM	Input and output in Decimal format mode	SCn	Selects the board address.
DN	Enable continuous position display.	SJn	Set jog register = n
DP	Define Proportional gain	SIn	Sets polarity of the index signal.
DSa	Define string = all characters received before next carriage return.	SNn	Stores a serial number in external memory.
DV	Define Velocity	SO	Set Analog Output
		ST	Stop
DU	Dumps suite of data, equivalent to TT,TP,TD,TS.	TA	Tell Analog Input
DWn	Set value of delay used in CY command = n.	TB	Tell Board address
DXn	Set positive limit of travel used by CY command = n.	TC	Tell Channel
DYn	Set negative limit of travel used by CY command = n.	TD	Tell Dynamic target
EF	Turn Echo off to RS-232 port	TE	Tell Error (distance from target)
EM	Execute Macro (alias for MC)	TF	Tell Following error (distance between position and dynamic target)
EN	Turn Echo on to RS-232 port	TI	Tell the Iteration number
FE	Find Edge	TJ	Report value of jog register.
FI	Find Index	TM	Tell Macro (1..31)
GA	Get Acceleration	TO	Tell Analog Output
GB.	Report DX and DY values	TP	Tell Position
GD	Get Derivative gain	TS	Tell Status
GGn	Get value of A/D channel n into virtual accumulator	TT	Tell Target
GH	Go Home	TV	Tell actual Velocity
GI	Get Integral gain	TZ	Tell Macro Zero
GL	Get Integral Limit	UD	Update
GP	Get Proportional gain	VE	Display VErsion number
GV	Get Velocity	VM	Sets velocity mode.
GW	Report value stored by DW.	WA	Wait specified time
HE	Print list of commands (HElp)	WS	Wait until end of trajectory
HM	Input and output in Hex format mode	XNn	Execute remainder of command line if port n is active.
IOn	Increment output	XFn	Execute remainder of command line if port n is inactive.
		XIn	Executes an internal sub- routine pointed to by address n.