PMC
Motion Control Command Language
(MCCL) Reference Manual
Revision 3.0

Precision MicroControl Corporation
2075-N Corte del Nogal
Carlsbad, CA 92009 * USA

Tel: +1-760-930-0101
Fax: +1-760-930-0222

www.pmccorp.com

Information: info@pmccorp.com
Technical Support: support@pmccorp.com

Precision MicroControl Corp.
2075-N Corte del Nogal
Carlsbad, CA 92011 • USA

Tel: +1-760-930-0101
Fax: +1-760-930-0222
Web: www.pmccorp.com
Email: info@pmccorp.com
support@pmccorp.com
sales@pmccorp.com

Table of Contents

Chapter 1

Introduction

A motion controller is much more than an I/O card with DAC outputs and encoder inputs. The primary task of a PC based motion controller is to off load control and monitoring duties from the PC processor. While most of today's motion controllers have CPU's powerful enough to handle missile control systems, without a powerful and efficient low level command set the motion controller would be nothing more than an expensive I/O card. Everything that a motion control card does is dependent on the command set. The command set of a state of the art motion controller should include:

- Moving one, some, or all motors simultaneously
- Executing synchronized motion (linear interpolation, circular contouring, helical motion)
- Setting trajectory parameters (maximum velocity, acceleration, deceleration)
- Setting PID filter parameters (proportional gain, derivative gain, derivative sampling period, integral gain, integral limit, allowable following error
- Reporting the status of an axis including: current position of an axis, target of a move, current following error, and indicating when a move is complete
- Electronic gearing of axes
- Homing an axis

The command set for the controller is called **MCCL** (**M**otion **C**ontrol **C**ommand **L**anguage) and it supports well over 200 operations. A complete listing and description of the controller's command set can be found in **Chapters 3 – 13** of this manual.

The MCCL command set is made up of two character alphanumeric mnemonics built with two key characters from the description of the operation (e.g. "MR" for ***Move Relative***). When the controller receives a MCCL command (followed by a carriage return) it will be executed immediately.

```
;Example

1MR1000   <Enter>                        Move axis #1 1000 encoder counts
```

Beginning with Chapter 5 of this manual you will find the MCCL command descriptions. The commands are divided into the following categories:

- Setup commands (set velocity, set proportional gain) – Chapter 5
- Mode commands(position mode, contour mode) – Chapter 6
- Motion commands(move relative, find index) – Chapter 7

- Reporting commands(tell position, tell axis status) – Chapter 8
- I/O commands(turn on digital output, configure input as high true) – Chapter 9
- Macro and Multi-tasking commands(execute a macro, terminate a background task – Chapter 10
- Register commands (copy accumulator to user register, multiply accumulator by) – Chapter 11
- Sequence commands (wait for trajectory complete, if accumulator = execute next command) – Chapter 12
- Miscellaneous commands (reset the controller, parameters in decimal mode) – Chapter 13

Who should use MCCL commands?

The target applications for the MultiFlex family of motion controllers are Windows PC based multi-axis applications, controlled by a user-written host application program written in a high-level language such as C/C++/C#/.NET. In these type of environments, the user's application program will issue function calls to PMC's motion control API (MCAPI) function library. The MCAPI functions will convert the function calls into one or more binary formatted MCCL commands. The Motion Control API device driver then passes the MCCL commands to the motion controller. For additional information on the Motion Control API please refer to the ***Motion Control API Reference Manual***.

For these type of applications, the machine designer does not have to use MCCL commands directly because PMC provides tools for:

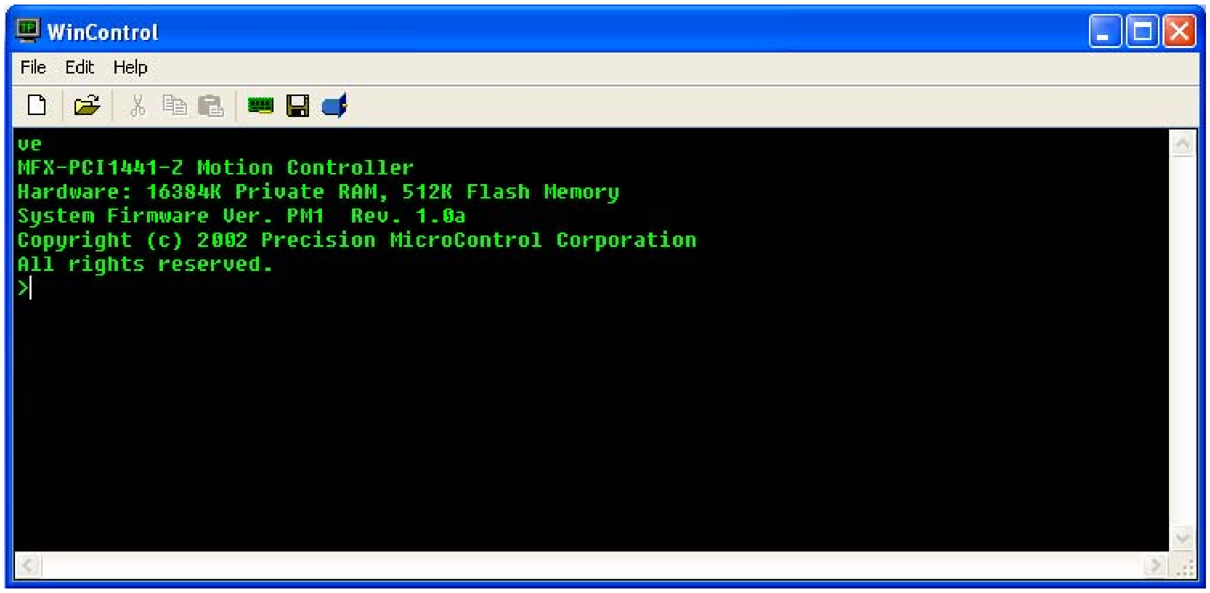
- Integrating the controller and external system components (**Motion Integrator**)
- Tuning the servo (**Servo Tuning program**)
- Exercising the motion control system (**Motor Mover**)

But there are times when the capability to issue MCCL commands directly to the controller can be useful. Some examples are:

- When homing routines are programmed as MCCL command sequences (versus MCAPI functions called from an application program) the PC is free to handle other tasks while the motion system is being initialized.
- Identifying whether unexpected system behavior is the fault of hardware, software, or external devices (amplifiers, sensors, feedback devices, etc.)
- When a non programmer needs to exercise system functions in combinations not supported by PMC tools (Motor Mover, CWDemo, Servo Tuning)
- Some applications require the motion controller to execute embedded subroutines which can run independently of the PC. To provide this functionality, the controller is capable of executing one or more MCCL command sequences simultaneously as background tasks.

WinControl – the MCCL command interface utility

One of the tools included with the MCAPI is the **WinControl** utility. This tool allows the user to issue MCCL commands directly to the controller.



```
ve
MFX-PCI1441-Z Motion Controller
Hardware: 16384K Private RAM, 512K Flash Memory
System Firmware Ver. PM1  Rev. 1.0a
Copyright (c) 2002 Precision MicroControl Corporation
All rights reserved.
>
```

Figure 1: The WinControl MCCL command interface utility

To open the WinControl utility from the Windows Start menu select Programs\Motion Control\Motion Control API\ WinControl.

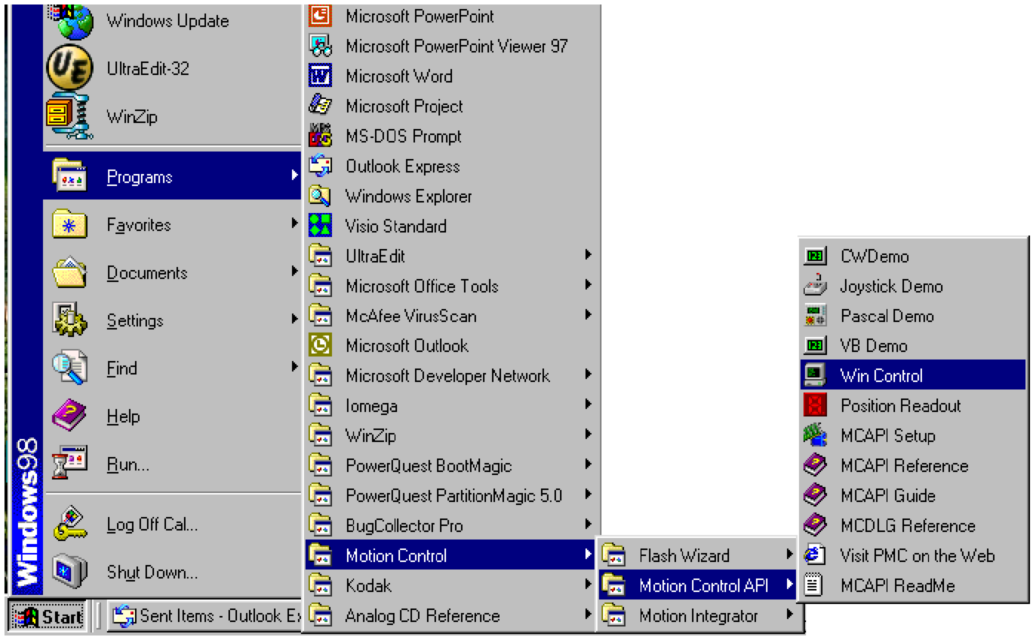
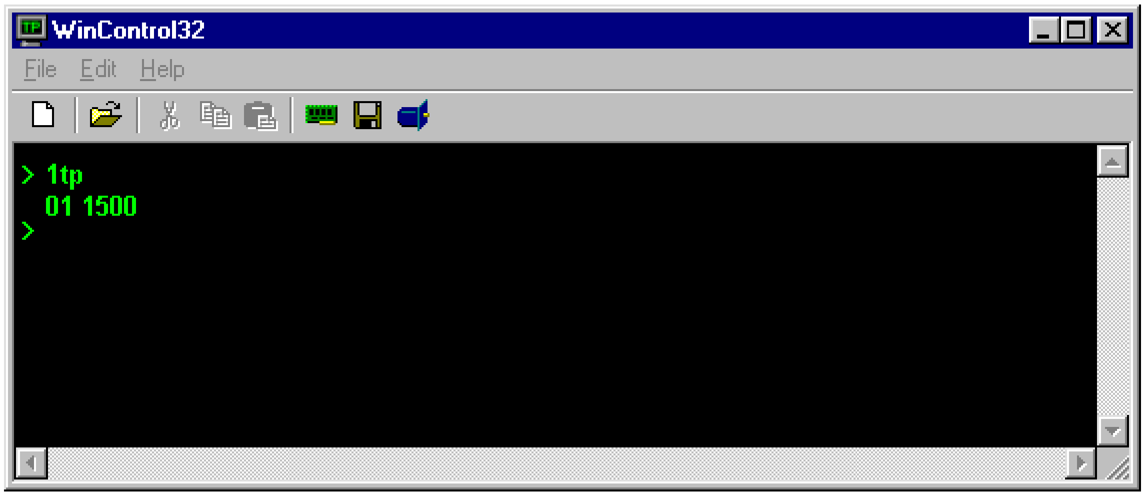


Figure 2: Open WinControl (\Start\Program Files\Motion Control\Motion Control API\WinCtl32)

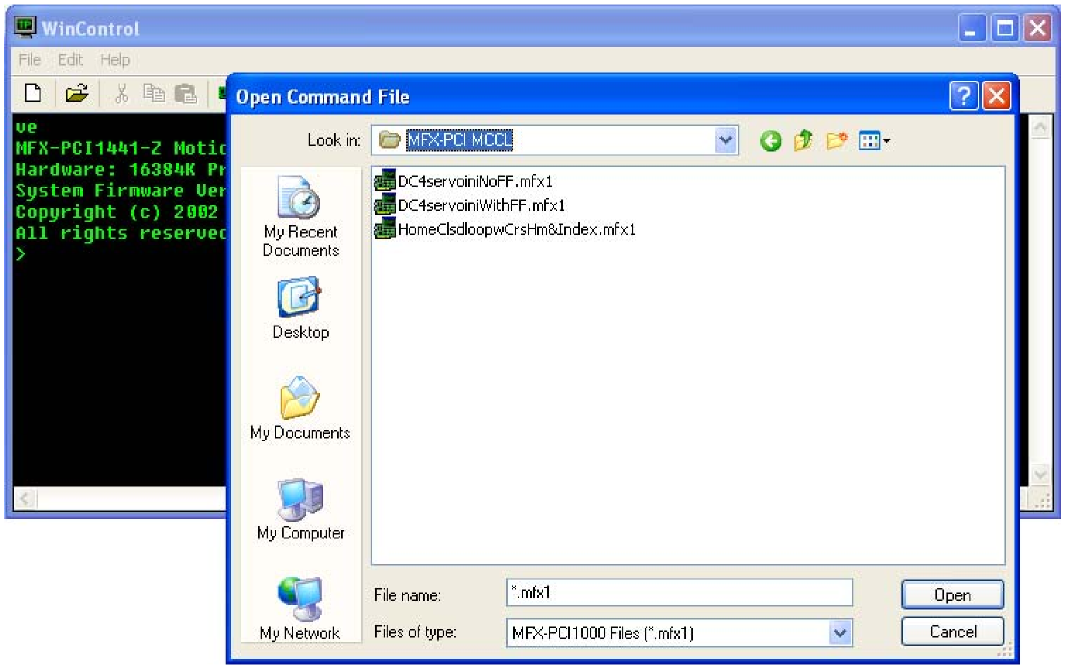
From the PC keyboard, MCCL commands can be entered one character at a time and executed when the user presses the **Enter** key. The user can issue a single MCCL command or multiple MCCL commands separated by commas. In figure #3 the current position of axis #1 is reported by issuing the MCCL command **T**ell **P**osition (***1*TP** <Enter>).



```
> 1tp
  01 1500
>
```

Figure 3: Report the current position of axis #1 by issuing the MCCL command Tell Position (1TP)

By selecting File and Open from the WinControl menu the user can download a text file containing MCCL commands to the controller.



```
ve
MFX-PCI1441-Z Motic
Hardware: 16384K Pr
System Firmware Ver
Copyright (c) 2002
All rights reservec
>
```

Look in: MFX-PCI MCCL

DC4servoiniNoFF.mfx1
DC4servoiniWithFF.mfx1
HomeClsdloopwCrsHm&Index.mfx1

File name: *.mfx1
Files of type: MFX-PCI1000 Files (*.mfx1)

Figure 4: Download a MCCL text file to the controller

Chapter Contents

Chapter
2

Controller Operation Basics

MCCL Command Syntax

All of the controller's MCCL commands are a 2 character mnemonics. The characters that make the mnemonic are selected from the command description so that the command has a direct correlation to the operation to be performed. For example, the MCCL command that is used to move an axis to an absolute position is:

MA	(**M**ove **A**bsolute).

Any MCCL command that references an axis is preceded by a numeric axis specifier. To issue a move absolute to axis #1:

1**MA**	(axis #1 **M**ove **A**bsolute)

By defining the axis specifier = 0 certain MCCL commands support execution on multiple axes. If this capability is supported it will be noted in the command description (later in this manual). For example if the numeric axis specifier equals 0 (0MA) then the command will be executed for all axes.

0**MA**	(all axes **M**ove **A**bsolute)

Most controller commands will also include a parameter value following the mnemonic. To move axis #1 to absolute position 10.25:

1**MA**10.25	(axis #1 **M**ove **A**bsolute to position 10.25)

If no parameter value is given with the command the default value of 0 will be used.

In the command descriptions in this manual, the axis specifier is shown as an italicized '*a*', and the parameter as an italicized '*n*'.

A typical command description is shown below:

Move Absolute

MCCL command:	*a*MA*n* where: *a* = Axis number *n* = integer or real >= 0
applies to:	Analog Command Axis, Pulse Command Axis
see also:	MR, PM

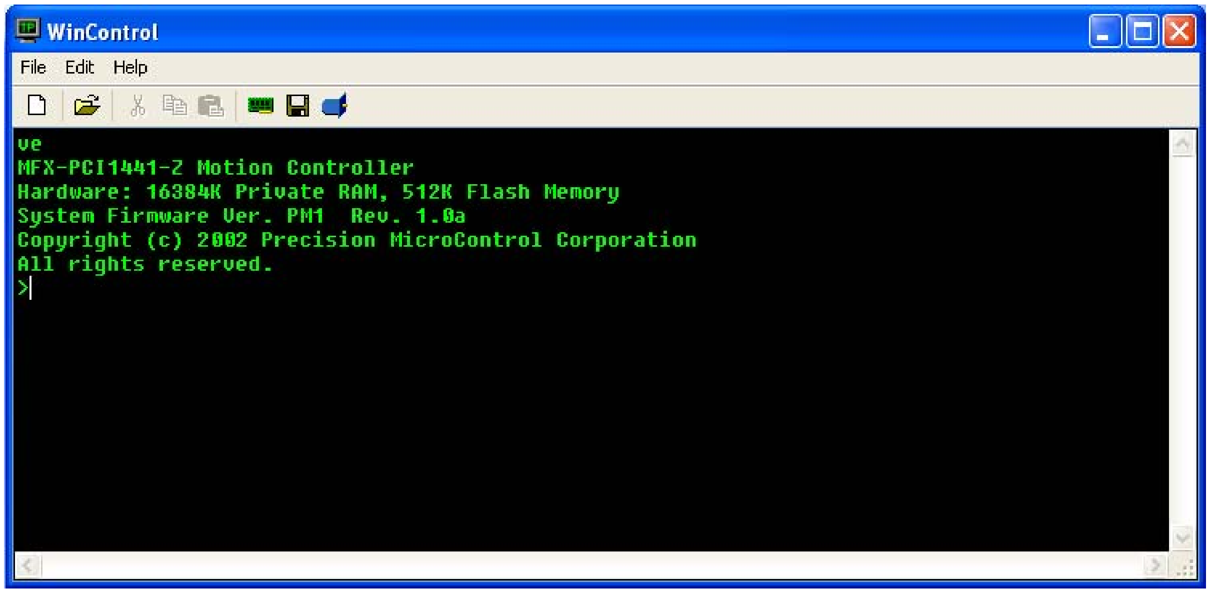
This command generates a motion to an absolute position *n*. A motor number must be specified and that motor must be in the 'on' state for any motion to occur. If the motor is in the off state, only its internal target position will be changed. See the description of **Point to Point Motion** in the **Motion Control** chapter.

The ***MCCL command*** line shows the command syntax and parameter type and/or range

The ***applies to*** line lists the basic controller operation with which the command is associated (Servo, Stepper, I/O, Other)

The ***see also*** line lists associated MCCL commands

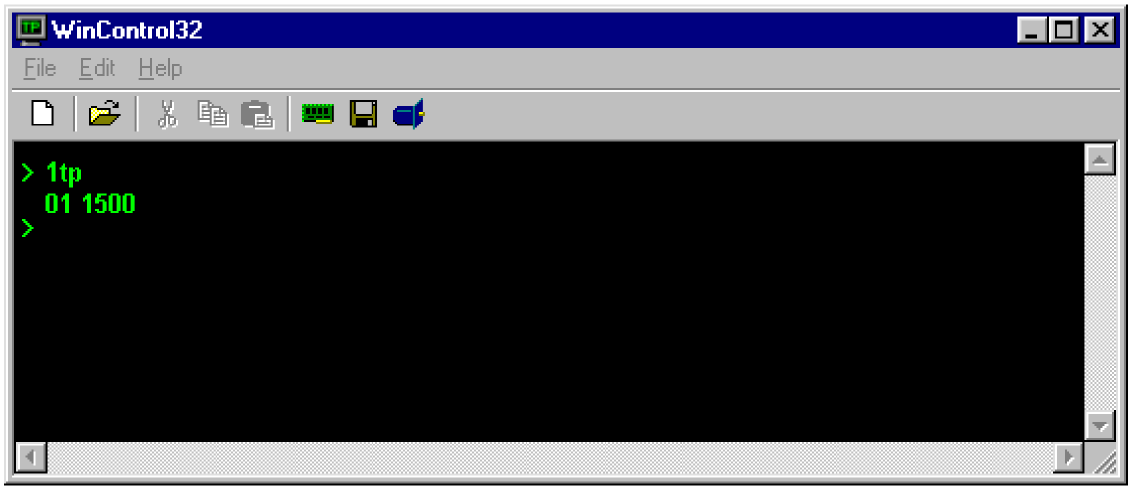
MCCL commands are issued to the controller via the WinControl utility. This tool allows the user to communicate directly with the controller. The graphic in figure #1 shows the result of executing the **VE** command. This command causes the controller to report the controller resources and the firmware version.



```
ve
MFX-PCI1441-2 Motion Controller
Hardware: 16384K Private RAM, 512K Flash Memory
System Firmware Ver. PM1  Rev. 1.0a
Copyright (c) 2002 Precision MicroControl Corporation
All rights reserved.
>
```

Figure 5: Using WinControl to display the firmware version of the controller

All axis related MCCL commands will be preceded by an axis specifier, identifying to which axis the operation is intended. The graphic in figure #2 shows the result of issuing the **T**ell **P**osition (aTP) command to axis number one.



```
> 1tp
  01 1500
>
```

Figure 6: Reporting the position of axis #1

Note that each character typed at the keyboard should be echoed to your display. If you enter an unrecognized or improperly formatted command the controller will echo a question mark character, followed by an error code. The **MCCL Error Code** listing can be found in the **Chapter 14** of this manual. On receiving this response, you should re-enter the corrected command/command string. If you make a mistake in typing, the backspace can be used to correct it; the controller will not begin to execute a command until the **Enter** key is pressed.

Simple moves with MCCL commands

Once you are satisfied that the controller is correctly processing commands issued from the PC host, you are ready to verify the operation of the connected axes. When the controller is powered up or reset, each axis is automatically set to the "motor off" state. In this state, there should be no command signal to the amplifier/drive, which means that no drive current is applied to the motor windings.

Before a motor can be successfully commanded to move certain parameters must be defined by issuing commands to the controller. The minimum required motor setup parameters include:

- PID filter gains (servo or closed loop stepper only)
- Trajectory parameters (maximum velocity, minimum velocity, acceleration, and deceleration)
- Allowable following error (servo and closed loop stepper only)
- Configuring motion limits (hard and/or soft)

For details on setting up these initial parameters please refer to the **Motion Control** chapter of the controller's **User Manual**. There users will find more specific information for each type of motor, including which parameters must be set before a motor should be turned on and how to check the status of the axis.

Assuming that all of the required motor parameters have been defined, the axis is enabled with the Motor oN (***a*MN**) command. The axis specifier '*a*' of the Motor oN command allows the user to turn on a selected axis or all axes. To enable all, enter the Motor oN command with specifier '*a*' = 0. To enable a single axis issue the Motor oN command where 'a' = the axis number to be enabled.

After turning a particular axis on, which will turn on the Amplifier Enable or Driver Enable output, it should hold steady at one position without moving .

If the axis is a servo which has not yet been tuned it may not 'hold position'. For assistance with tuning a servo please refer to the descriptions of **Servo Tuning** in the controller's **User's Manual**.

The **T**ell **T**arget (***a*TT**) command will report the position at which the axis should be and the **T**ell **P**osition (***a*TP**) command will report the current actual position of the axis. There are several commands that can used to begin motion, including **M**ove **A**bsolute (MA) and **M**ove **R**elative (MR). To move axis 2 by 1000 encoder counts, enter 2MR1000 and a carriage return. If the axis is in the "Motor oN" state, it should move in the direction defined as positive for that axis. To move back to the previous position enter 2MR-1000 and a carriage return.

It is possible to group together several MCCL commands. This is not only useful for defining a complex motion that can be repeated by a single keystroke, but is also useful for synchronizing multiple motions. To group commands together, simply place a comma between each command (with no intervening spaces), pressing the Enter key only after the last command.

A repeat cycle can be set up with the following compound command:

```
2MR1000,WS0.5,MR-1000,WS0.5,RP6  <Enter>
```

This command string will cause axis 2 to move from position 1000 to position –1000 a total of 7 times. The **R**e**P**eat (**RP*n***) command at the end causes the previous command to be repeated 6 additional times. The **W**ait for **S**top (***a*WS*n***) commands are required so that the motion will be completed (trajectory complete) before the return motion is started. The number 0.5 following the WS command specifies the number of seconds to wait after the axis has ceased motion to allow some time for the mechanical components to come to rest and reduce the stresses on them that could occur if the motion were reversed instantaneously. Notice that the axis number need be specified only once on a given command line.

A more complex cycle could be set up involving multiple axes. In this case, the axis that a command acts on is assumed to be the last one specified in the command string. Whenever a new command string is entered, the axis is assumed to be 0 (all) until one is specified.

Entering the following command:

```
        2MR1000,3MR-500,0WS0.3,2MR1000,3MR500,0WS0.3,RP4  <Enter>
```

will cause axis 2 to move in the positive direction and axis 3 to move in the negative direction. When both axes have stopped moving, the WS command will cause a 0.3 second delay after which the remainder of the command line will be executed.

After going through this complex motion 5 times, it can be repeated another 5 times by simply pressing the Enter key. All command strings are retained by the controller until some character other than an Enter key is pressed. This comes in handy for observing the position display during a move. If you enter:

```
        1MR1000   <Enter>
        1TP   <Enter>
        <Enter>
        <Enter>
        <Enter>
        <Enter>
```

The controller will respond with a succession of numbers indicating the position of the axis at that time. Many terminals have an "auto-repeat" feature that allows you to track the position of the axis by simply holding down the Enter key.

Another way to monitor the progress of a movement is to use the ***Repeat*** command without a value. If you enter:

```
        1MR10000  <Enter>
        1TP,RP    <Enter>
```

The position will be displayed continuously. These position reports will continue until stopped by the operator pressing the Escape key.

While the controller is executing commands, it will ignore all alphanumeric keys that are pressed. The user can abort a currently executing command or string by pressing the Escape key. If the user wishes only to pause the execution of commands, the user should press the Space bar. In order to restart command execution press the Space bar again. If after pausing command execution, the user decides to abort execution, this can be done by pressing the Escape key.

Chapter Contents

Chapter 3

MCCL Command Quick Reference Tables

Setup Commands

MCCL	Code	Description
AG	E2h	set Acceleration feed-forward Gain
AH	EAh	Auxiliary encoder define Home
BD	DOh	Backlash compensation Distance
DB	76h	set position DeadBand
DG	E3h	set Deceleration feed-forward Gain
DH	23h	Define Home
DI	24h	DIrection
DS	75h	Deceleration Set
DT	C5h	Delay at Target
EI	7Ch	Enable Interrupts
ES	E5h	Encoder counts / Steps Scale
FC	40h	Full Current
FF	33h	amplifier Fault input ofF
FL	151h	IIR Filter Load coefficients
FN	32h	amplifier Fault input oN
FR	27h	set derivative sampling period
HC	41h	Half Current
HL	D3h	set motion High Limit
HS	E8h	set High Speed
IL	28h	set Integration Limit
IO	205h	Integral option
LA	330h	Load commutation phase A
LB	331h	load commutation phase B
LD	333h	Load commutation Divisor
LE	332h	Load commutation encoder prescale
LF	36h	motion Limits ofF
LL	D2h	set motion Low Limit
LM	34h	Limit Mode
LN	35h	motion Limits oN
LR	334h	Load commutation repeat count
LS	36h	set Low Speed

Setup Commands continued

MCCL	Code	Description
MS	E7h	set Medium Speed
MV	C4h	set Minimum Velocity
NF	150h	No IIR Filter
OB	C3h	set the Output deadBand
OM	D8h	set Output Mode
OO	CBh	set the Output Offset
PH	73h	set servo output PHase
SA	2Bh	Set Acceleration
SD	2Ch	Set Derivative gain
SE	19h	Stop on Error
SG	2Dh	Set prop. Gain of motor
SI	2Eh	Set Integral gain
SQ	74h	Set TorQue
SS	1Dh	Set Slave ratio
SV	2Fh	Set Velocity
UA	9Ch	Use as default Axis
UK	D7h	set User output constant
UO	B3h	set User Offset
UP	9Dh	Use Physical axis
UR	B1h	set User Rate conversion
US	AFh	set User Scale
UT	B2h	set User Time conversion
UZ	B0h	set User Zero
VA	ADh	set Vector Acceleration
VD	AEh	set Vector Deceleration
VG	77h	set Velocity Gain
VO	E0h	set Velocity Override
VV	ACh	set Vector Velocity
YF	14Fh	Yes IIR Filter
ZF	153h	Zero IIR Filter coefficients

Motion Commands

MCCL	Code	Description
AB	Ah	ABort
AF	EBh	Auxiliary encoder arm/Find index
BC	144h	Begin position Compare
BF	CFh	Backlash compensation oFf
BN	CEh	Backlash compensation oN
CA	B4h	arc Center Absolute
CB	148h	position Capture Begin
CD	C1h	Contour Distance
CP	C0h	Contour Path
CR	B5h	arc Center Relative
EA	BBh	arc Ending Angle absolute
EL	E4h	home Edge Latch
ER	BCh	arc Ending angle Relative
FE	Bh	Find Edge
FI	Ch	Find Index
GH	Dh	Go Home
GO	Eh	GO
HO	Fh	HOme
IA	5Dh	Index Arm
LC	142h	Load Compare positions
LP	70h	Learn Position
LT	71h	Learn Target
MA	10h	Move Absolute
MF	11h	Motor ofF
MN	13h	Motor oN
MP	14h	Move to Point
MR	15h	Move to Point
NC	143h	Next Compare
NS	ABh	No Synchronization
OC	140h	Output mode for Compare
OP	141h	Output Period
PP	EDh	Profile Parabolic
PR	D4h	Record motion data
PS	EEh	Profile S-curve
PT	EFh	Profile Trapezoidal
RR	B6h	aRc Radius
SN	AAh	Synchronization oN
ST	16h	STop

Mode Commands

MCCL	Code	Description
CM	1Bh	enable Contour Mode (arcs and lines)
GM	8h	enable Gain Mode (no velocity profile)
IM	72h	Input Mode (closed loop stepper)
PM	17h	enable Position Mode
SM	1Ah	enable Master/Slave mode
QM	9H	enable TorQue mode
VM	18h	enable Velocity Mode

Reporting Commands

MCCL	Code	Description
AT	E9h	Auxiliary encoder Tell position
AZ	ECh	Auxiliary encoder tell index
CG	149h	Capture Get count
DA	DBh	Display recorded aux. encoder position
DO	D6h	Display recorded optimal position
DQ	BDh	Display recorded DAC output
DR	D5h	Display recorded actual position
GC	145h	Get the position compare count
TA	49h	Tell Analog to digital converter
ID		Information Display
TB	5Bh	Tell Breakpoint position
TC	4Ah	Tell Channel
TD	4Bh	Tell Derivative gain
TE	4Ch	Tell command interface Error
TF	4Dh	Tell Following error
TG	4Eh	Tell proportional Gain
TI	4Fh	Tell Integral gain
TK	5Ch	Tell velocity gain
TL	50h	Tell integration Limit
TM	51h	Tell stored Macros
TO	59h	Tell Optimal
TP	52h	Tell Position
TQ	D1h	Tell torQue (aSQn)
TR	57h	Tell Register n
TS	53h	Tell Status
TT	54h	Tell Target
TV	55h	Tell Velocity
TX	58h	Tell contouring count
TZ	5Ah	Tell index position
VE	56h	tell VErsion

Register Commands

MCCL	Code	Description
AA	85h	Accumulator Add
AC	8Ch	Accumulator Complement
AD	88h	Accumulator Divide
AE	8Fh	Accumulator logical Exclusive or
AL	82h	Accumulator Load
AM	87h	Accumulator Multiply
AN	8Dh	Accumulator logical aNd with n,
AO	83h	Accumulator logical Or with n
AR	84h	copy Accumulator to Register n
AS	86h	Accumulator Subtract
AV	8Bh	Accumulator eValuate
AX	E1h	get Aux. indeX position
GA	F8h	Get Analog value
GF	152h	Get IIR Filter coefficients
GU	89h	Get the default axis
GX	F7h	Get auXiliary encoder position
LU	81h	Look Up motor table variable
OA	F9h	Output Analog value
RA	83h	copy Register to Accumulator
RB	96h	Read Byte into accumulator
RD	93h	Read Double into accumulator
RL	98h	Read Long into accumulator
RV	92h	Read float into accumulator
RW	97h	Read Word into accumulator
SL	90h	Shift Left accumulator n bits
SR	91h	Shift Right accumulator n bits
TR	57h	Tell contents of Register n
WB	99h	Write accumulator Byte to n
WD	95h	Write accumulator double to n
WL	9Bh	Write accumulator Long to n
WV	94h	Write accumulator float to n
WW	9Ah	Write accumulator Word to n

Miscellaneous Commands

MCCL	Code	Description
DM	3Ch	Decimal Mode
DW	FDh	Disable Watchdog
EF	25h	Echo ofF
EN	26h	Echo oN
FD	CCh	Format text with Doubles
FM		Free memory
FT		Format Text with Integers
HM	3Dh	Hexadecimal Mode
ME		MEmory allocate
NO	78h	No Operation
OD	C3h	Output text with Doubles
OT		Output Text with integers
PC	80h	set Prompt Character
RT	2Ah	ReseT system

Macro Commands

MCCL	Code	Description
BK	79h	BreaK
ET	FBh	Escape Task
GT	FAh	Generate Task
MC	2h	Macro Call
MD	3h	Macro Definition
MJ	5h	Macro Jump
NP		New Priority
RM	4h	Reset Macros
TM	51h	Tell Macros

Sequence Commands

MCCL	Code	Description
DF	6Bh	Do if channel ofF
DN	6Ah	Do if channel oN
IB	A5h	If Below do next command
IC	A1h	If Clear, do next command
IE	A2h	If Equals do next command
IF	6Dh	If channel ofF do next command
IG	A4h	If accumulator is Greater do next
IN	6Ch	If channel oN do next command
IP	60h	Interrupt on absolute Position
IR	61h	Interrupt on Relative position
IS	A0h	If bit Set do next command
IU	A3h	If Unequal do next command
JP	6h	JumP to command absolute
JR	7h	Jump to command Relative
RP	64h	RePeat
WA	65h	WAit (time)
WE	66h	Wait for Edge
WF	67h	Wait for channel ofF
WI	5Eh	Wait for Index
WN	68h	Wait for channel oN
WP	62h	Wait for absolute Position
WR	63h	Wait for Relative position
WS	69h	Wait for Stop
WT	C6h	Wait for Target

I/O Commands

MCCL	Code	Description
CF	1Fh	Channel ofF
CH	42h	Channel High true logic
CL	43h	Channel Low true logic
CN	21h	Channel oN
GA	F8h	Get Analog
OA	F9h	Output Analog
TA	49h	Tell the value of Analog input
TC	4Ah	Tell state of digital Channel
WF	67	Wait for channel ofF
WN	68	Wait for channel oN

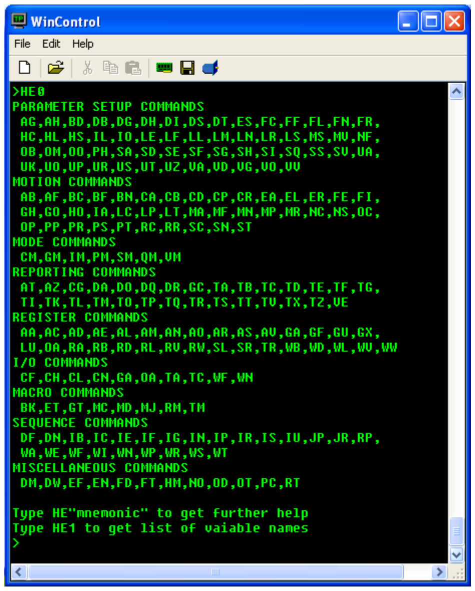
Chapter Contents

Chapter 4

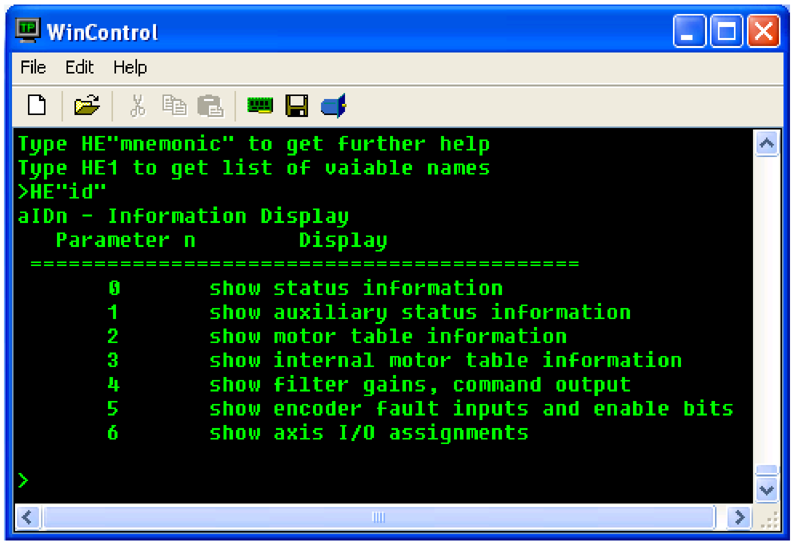
Working with MCCL Commands

MCCL On-line Help

For MCCL command on-line help issue the Help (HE) command from WinControl. When the HE command is issued with a parameter of 0 the controller will report all supported MCCL commands. For a description of a MCCL command issue the HE command with string parameter "mm" (where mm = the command mnemonic).



```
>HE0
PARAMETER SETUP COMMANDS
 AG,AH,BD,DB,DG,DH,DI,DS,DT,ES,FC,FF,FL,FN,FR,
 HC,HL,HS,IL,IO,LE,LF,LL,LM,LN,LR,LS,MS,MV,NF,
 OB,OM,OO,PH,SA,SD,SE,SF,SG,SH,SI,SQ,SS,SV,UA,
 UK,UO,UP,UR,US,UT,UZ,VA,VD,VG,VO,VV
MOTION COMMANDS
 AB,AF,BC,BF,BN,CA,CB,CD,CP,CR,EA,EL,ER,FE,FI,
 GH,GO,HO,IA,LC,LP,LT,MA,MF,MN,MP,MR,NC,NS,OC,
 OP,PP,PR,PS,PT,RC,RR,SC,SN,ST
MODE COMMANDS
 CM,GM,IM,PM,SM,QM,VM
REPORTING COMMANDS
 AT,AZ,CG,DA,DO,DQ,DR,GC,TA,TB,TC,TD,TE,TF,TG,
 TI,TK,TL,TM,TO,TP,TQ,TR,TS,TT,TV,TX,TZ,VE
REGISTER COMMANDS
 AA,AC,AD,AE,AL,AM,AN,AO,AR,AS,AV,GA,GF,GU,GX,
 LU,OA,RA,RB,RD,RL,RV,RW,SL,SR,TR,WB,WD,WL,WV,WW
I/O COMMANDS
 CF,CH,CL,CN,GA,OA,TA,TC,WF,WN
MACRO COMMANDS
 BK,ET,GT,MC,MD,MJ,RM,TM
SEQUENCE COMMANDS
 DF,DN,IB,IC,IE,IF,IG,IN,IP,IR,IS,IU,JP,JR,RP,
 WA,WE,WF,WI,WN,WP,WR,WS,WT
MISCELLANEOUS COMMANDS
 DM,DW,EF,EN,FD,FT,HM,NO,OD,OT,PC,RT

Type HE"mnemonic" to get further help
Type HE1 to get list of vaiable names
>
```



```
Type HE"mnemonic" to get further help
Type HE1 to get list of vaiable names
>HE"id"
aIDn - Information Display
   Parameter n         Display
 ===========================================
       0       show status information
       1       show auxiliary status information
       2       show motor table information
       3       show internal motor table information
       4       show filter gains, command output
       5       show encoder fault inputs and enable bits
       6       show axis I/O assignments

>
```

Reply to 'HE0'

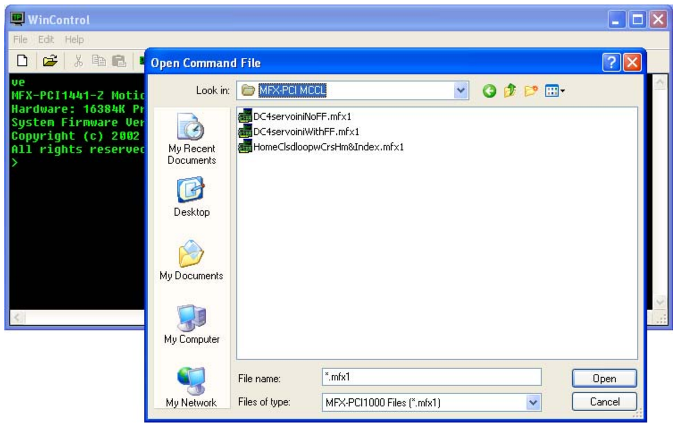
Reply to 'HE"id"' (describe Info Display command)

Downloading MCCL Text Files

Motion Control Command Language (MCCL) command sequences can be downloaded as text files to the controller. If these command sequences are not defined as macro's (MD*n*) then the commands will be executed in real time as they are received by the card. If the command sequences are defined as macro's they will be stored in the memory of the controller for execution at a later time.

While most motion control applications will utilize the high level language (C++, VB, Delphi, LabVIEW, etc..) function calls to program the operation of the machine, downloaded MCCL text files are typically used for initial system integration, defining homing routines, and programming local subroutines (background tasks).

The following graphic is a screen capture of PMC's WinControl . This utility provides the user with a direct interface to the controller. A MCCL text file (init.mfx1) containing servo parameters and a homing routine has been downloaded to the controller using the File – Open menu options.



```
ve
MFX-PCI1441-Z Motic
Hardware: 16384K Pr
System Firmware Ver
Copyright (c) 2002
All rights reservec
>
```

Look in: MFX-PCI MCCL

DC4servoiniNoFF.mfx1
DC4servoiniWithFF.mfx1
HomeClsdloopwCrsHm&Index.mfx1

File name: *.mfx1
Files of type: MFX-PCI1000 Files (*.mfx1)

Figure 7: Downloading a MCCL text file via WinControl

Note: Any characters that are preceded by a semicolon are treated as commands. These comment strings are displayed by WinControl but they are 'stripped' from the file and are not be passed to the controller.

Building MCCL Macro Sequences

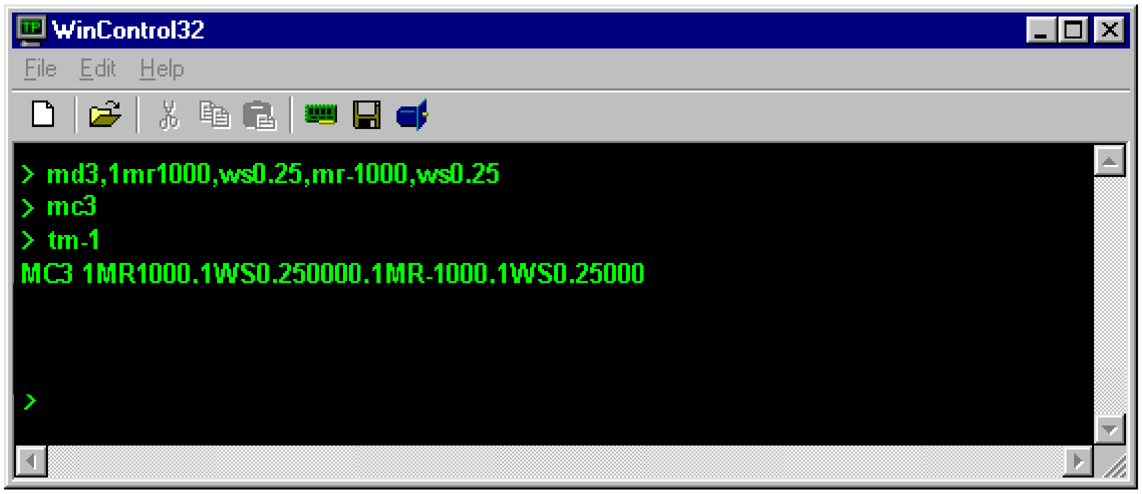
A powerful feature of the controller is the ability to define MCCL (Motion Control Command Language) command sequences as macros. This simply means defining a mnemonic that will execute a user defined sequence of commands. For example:

```
1MR1000,WS0.25,MR-1000,WS0.25
```

will cause the motor attached to axis 1 to move 1000 counts in the positive direction, wait one quarter second after it has reached the destination, then move back to the original position followed by a similar delay. If this sequence were to represent a frequently desired motion for the system, it could be defined as a macro command. This is done by inserting a Macro Define (MD) command as the first command in the command string. For example:

```
MD3,1MR1000,WS0.25,MR-1000,WS0.25
```

will define macro #3. Whenever it is desired to perform this motion sequence, issue the command Macro Call (MC3). To command the controller to display the contents of a macro, issue the **T**ell **M**acro (**TM*n***) command with parameter ‘n’ = the number of the macro to be displayed. To display the contents of all stored macro’s issue the Tell macro command with parameter ‘n’ = -1.



```
> md3,1mr1000,ws0.25,mr-1000,ws0.25
> mc3
> tm-1
MC3 1MR1000.1WS0.250000.1MR-1000.1WS0.25000

>
```

Unless executed as a background task, once a macro operation has begun, the host will not be able to communicate with the controller until the **macro has terminated**. For information on communicating with the controller while executing macro’s please refer to the section titled **MCCL Multi-Tasking**.

The controller can store up to 2000 user defined macros. Each macro can include as many as 255 bytes, resulting in a total macro capacity of 510K bytes. Depending on the type of command and type of parameter, a command can range from 2 bytes (a command with no parameter) to 10 bytes (a command with a 64 bit floating point parameter).

If the amount of available macro memory exceeds 510K bytes the controller will respond with error code - 18

On MultiFlex PCI Series controllers, all controller memory is volatile, which means that the data in memory will be cleared when the controller is reset or power to the board is turned off. The **R**eset **M**acro (**RM*n***) command can be used to erase macros without re-booting the controller.

On the other hand, all MultiFlex ETH Series Ethernet controllers feature non-volatile memory storage which can be used to store MCCL macro routines. This allows completely independent “stand-alone” operation of the controller. MCCL macros stored in non-volatile memory will be cleared when the **R**eset **M**acro (**RM*n***) command is issued.

To terminate the execution of any macro that was started from WinControl press the escape key. To start a macro that runs indefinitely without ‘locking up’ communication with the host, start the macro’s with the ***generate a Background task*** (GT) command instead of the ***Call macro command*** (MC). This will allow the operations called by macro 0 to execute as a background task. Please refer to the next section **Multi-Tasking**.

MCCL Multi-Tasking

The controller's command interpreter is designed to accept commands from the user and execute them immediately. With the addition of sequencing commands, the user is able to create sophisticated command sequences that run continuously, performing repetitive monitoring and control tasks. The drawback of running a continuous command sequence is that the command interpreter is not able to accept other commands from the user.

Unless executed as a background task, once a macro operation has begun, the user will not be able to communicate with the controller until the **macro has terminated**.

The controller supports Multi-tasking, which allows the controller to execute continuous monitoring or control sequences as background tasks while the foreground task communicates with the 'user'.

With the exception of ***reporting commands*** (Tell Position, Tell Status, etc...), any MCCL commands can be executed in a background task. Prior to executing a command sequence/macro as a background task, the ***user should always test the macro by first executing it as a foreground task***. When the user is satisfied with the operation of the macro, it can be run as a background task by issuing the **G**enerate **T**ask (**GT*n***) command, specifying the macro number as the command parameter. After the execution of the Generate Task command, the accumulator (register 0) will contain an identifier for the background task. Within a few milliseconds, the controller will begin running the macro as a background task in parallel with the foreground command interpreter. The controller will be free to accept new commands from the user.

```
;Multitasking example – while axis #1 is moving, monitor the state of digital
;input #4. When the input goes active, stop axis #1 and terminate the
;background task

AL0,AR10                          ;define user register 10 as input #4 active
                                  ;flag register
AL0,AR100                         ;define user register #100 as background task
                                  ;ID register

MD100,IN4,MJ101,NO,1JR-3          ;jump to macro 101 when digital input #4
                                  ;turns on
MD101,1ST,1WS.05,AL1,AR10,ET@100  ;stop axis #1. Terminate background task

GT100,AR@100,1VM,1DI0,1GO         ;spawn macro #100 as background task. Store
                                  ;task ID into register #100. Start axis #1
                                  ;moving in velocity mode,
```

Note: Immediately after 'spawning' the background task (with the GTn command), the value in the accumulator (task identifier) should be stored in a user register. This value will be required to terminate execution of the background task.

Another way to create a background task is to place the Generate Task command as the first command in a command line, using a parameter of 0. This instructs the command interpreter to take all the commands that follow the Generate Task command and cause them to run as a background task. The commands will run identically to commands placed in a macro and generated as a task.

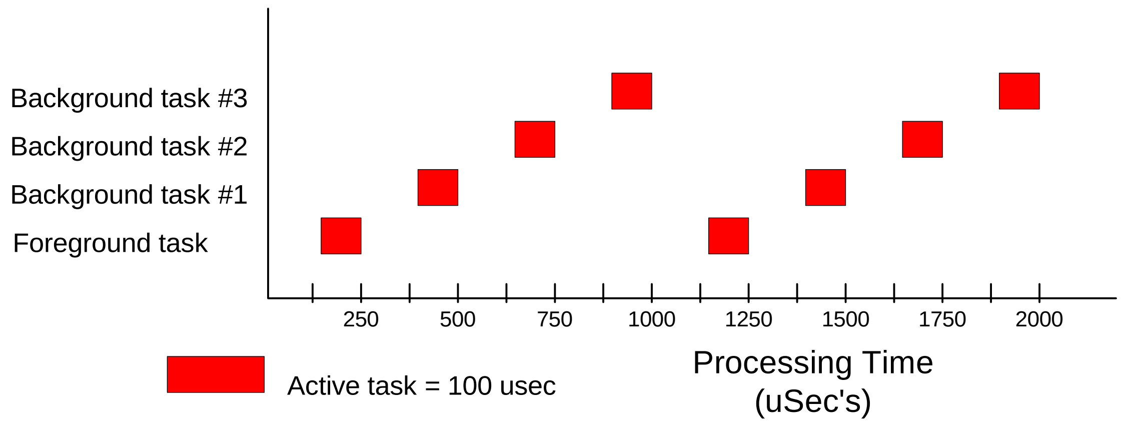
```
;Multitasking example – while axis #1 is moving, monitor the state of the
;motor error status bit (bit 0). If error occurs set bit #1 of user
;register 200

GT0,AR@100,LU”STATUS”,1RL@0,IC0,JR-3,NO,AL1,AR200,ET@100
                                  ;loop on axis #1 status bit 0, if set; set
                                  ;bit #1 of register 200, terminate task using
                                  ;Task ID (in register #100)
```

Within the background task, the commands can move motors, wait for events, or perform operations on the registers, totally independent of any commands issued in the foreground. However, the user must be careful that they do not conflict with each other. For example, if a background task issues a move command to cause a motor to move to absolute position +1000, and the user issues a command at the same time to move the motor to -1000, it is unpredictable whether the motor will go to plus or minus 1000.

In order to prevent conflicts over the registers, the background task has its own set of registers 0 through 9 (register 0 is the accumulator). These are private to the background task and are referred to as its 'local' registers. The balance of the registers, 10 through 999, are shared by the background task and foreground command interpreter, they are referred to as 'global' registers. If the user wishes to pass information to or from the background task, this can be done by placing values in the global register. Note that when a task is created, an identifier for the task is stored in register 0 of both the parent and child tasks.

The controller is able to run multiple background tasks, each with their own set of registers, but can only have one foreground command interpreter for each communication interface (Binary and/or ASCII). The maximum number of background tasks is 21. Each background task that is generated and the foreground command interpreter get an equal share of the controller’s CPU time.

When one or more background tasks are active the controller’s Task Handler will begin issuing local controller interrupts every 250 microseconds. Each time the task handler interrupt is asserted, the controller will switch from executing one task to the next every 250 micro seconds. For example if three background tasks are active, plus the foreground task (always active), each of the four tasks will receive approximately 100 micro seconds of processor time every 1 millisecond.



While a background task executes a **Wait** command, that task no longer receives any processor time. For tasks that perform monitoring functions in an endless loop, the command throughput of the controller can be improved by executing a **Wait** command at the end of the loop until the task needs to run again.

A common way for a background task to be terminated, is when the command sequence of the task finishes execution. This will occur at the end of the macro or if a **Brea K** (**BK**) command is executed. When a task is terminated, the resources it required are made available to run other background tasks.

```
;Multitasking example – this background task will terminate itself if the
;motor error status bit for axis #1 is set. This sequence is similar to the
;previous example except that the task is self terminating, so register #100
is not required.

GT0,LU"STATUS",1RL@0,IC0,JR-3,NO,AL1,AR200,BK
                                  ;loop on axis #1 status bit 0, if set; set
                                  ;bit #1 of register 200, task self terminates
                                  ;(no commands left to execute)
```

Alternatively, the **E**scape **T**ask (**ET*n***) command can be used to force a background task to terminate. When a task is generated by the **GT** command, a value known as the **Task ID** is placed into the accumulator. This value should immediately be copied into a user register. The parameter to this command must be the value that was placed in accumulator (register 0) of the parent task, when the Generate Task command was issued.

Issuing the **I**nformation **D**isplay command with parameter n = 7 (ID7) will report the task ID of all active background tasks.The

```
;Multitasking example – Terminating a background task with the Escape Task
command.

GT100,AR@150                      ;call macro #100 as a background task, copy
                                  ; task ID into user register 150

ET@150                            ;to terminate background task issue escape
                                  ; task command with parameter n = Task ID
```

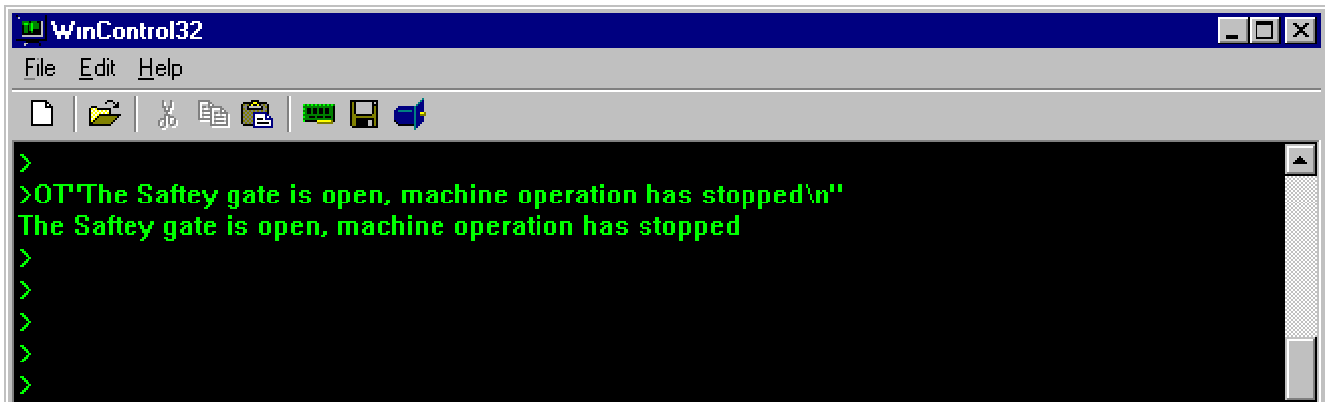
Outputting Formatted Message Strings

The controller supports the outputting of formatted text strings from the ASCII interface using the Output Text commands. The two commands supported are:

Output Text with integer values (OT”   “)
Output text with Double values (OD”   “)

The syntax of these two commands are patterned after standard ‘C’ function ‘printf’. For specific ‘printf’ description please refer to the Microtech Research Inc. MCC960 compiler documentation. The message to be displayed should be delimited by double quotes. Please refer to the examples below:

```
OT”The Safety gate is open, machine operation has stopped \n”
                                        ;output simple text message,
                                        ; \n = line feed
```



```
>
>OT"The Saftey gate is open, machine operation has stopped\n"
The Saftey gate is open, machine operation has stopped
>
>
>
>
>
```

As with typical implementations of ‘C’ print statements, the controller supports variables. Prior to executing the output text command, load the accumulator with the data to be included as a variable. In the following example, the Output Double (OD” “) command is used to report the current position of axis one as a floating point value. The % character indicates that a variable stored in the accumulator will be included in the text message. The ‘f’ indicates that the variable is a floating point value. The ‘\r’ calls for a carriage return at the end of the message, \n calls for a line feed.

```
LU"POSITION",1RD@0,OD”The current position of Axis #1 %f \n”
                                        ;load the accumulator with the
                                        ;position of axis #1. Output a text
                                        ;message displaying the position of
                                        ;axis #1 (floating point value),
                                        ;carriage return
```

To output integer variables use the OT command and %d in the string parameter.

Reading Data from Controller Memory

A group of read commands are available for accessing the internal Data Tables of the controller. These commands provide an easy method of moving motor data in and out of the Accumulator (user register 0).

The **L**ook **U**p (**LU*s***) command is used to load the internal address of a motor table entry into the accumulator. String parameter *s* defines the variable name of the target motor table entry. A Read command (RB, RW, RL, RV, or RD) is then used to load the accumulator with the data from the target motor table entry. The Read command must include the axis specifier ‘a’. The type of command to use (byte, double, long, float or word), is determined by the type of data to be accessed and is listed below.

a*RB*n	Read Byte (8 bit) at memory location n into accumulator (ACC = (n))
a*RD*n	Read Double at memory location n into accumulator (ACC = (n))
a*RL*n	Read Long (32 bit) at memory location n into accumulator (ACC = (n))
a*RV*n	Read float at memory location n into accumulator (ACC = (n))
a*RW*n	Read Word (16 bit) at memory location n into accumulator (ACC = (n))

Examples of using the read commands to access the motor tables are shown below.

To load the 32 bit status of axis 2 into the accumulator, issue the following command sequence:

```
LU”STATUS”,2RL@0                    ;load the motor table address for axis
                                    ;status into the accumulator. Load the
                                    ;32 bit status word of axis #2 into the
                                    ;accumulator
```

To load the 64 bit current position of axis 3 into the accumulator, issue the following command:

```
LU”POSITION”,3RD@0                  ;load the motor table address for current
                                    ;position into the accumulator. Load the
                                    ;accumulator with the 64 bit current
                                    ;position of axis #3
```

Motor Table Variables – 32 bit integer (long)

Motor Table Variable Description	Variable Name
Interrupt Mask	INTMASK
Interrupt Pending	INTPEND
Motor Status – primary	STATUS
Motor Status - auxiliary	AUXSTAT
Position - Adjustment (index + Offset)	CNTADJ
Position – Current (raw count)	POSCOUNT
Position - Index Count	IDXCOUNT
Position - Optimal (raw count)	CMDCOUNT
Position – Target (raw count)	TGTCOUNT

Motor Table Variables – 64 bit floating point (double)

Motor Table Variable Description	**Variable Name**
Backlash count	BACKLASH
Encoder counts / Steps scaling	ENCSCALE
Encoder Fault	ENCODERFLT
Following Error Setting - maximum	MAXERROR
Position - Current	POSITION
Position – Target	TARGET
Position - Optimal	OPTIMAL
Position - Breakpoint	BRKPOS
Position Deadband	POSDBAND
Programmed Acceleration	PGMACC
Programmed Deceleration	PGMDEC
Programmed Minimum Velocity	MINVEL
Programmed Velocity	PGMVEL
Scaling - User	SCALE
Scaling - User Offset	OFFSET
Scaling - User Output Constant	OUTCONST
Scaling - User Rate Conversion	RATE
Scaling - User Zero	ZERO
Soft Motion Limit Setting (low)	LOLIM
Soft Motion Limit Setting (high)	HILIM
Target Delay (seconds)	TGTDELAY
Velocity - Current	CURVEL
Velocity Gain - Scaled	VELGAIN

Motor Table Variables – 32 bit floating point (float)

Motor Table Variable Description	**Variable Name**
Feed Forward - Velocity Gain	KVEL
Feed Forward - Acceleration Gain	KACC
Feed Forward - Deceleration Gain	KDEC
Output Deadband (volts)	OUTDBAND
PID - Proportional Gain setting	KPOS
PID - Derivative Gain setting	KDER
PID - Integral Gain	KINT
PID - Integration Limit	ILIM
Torque Limit – max. output	MAXTRQ
User Offset (volts)	OUTOFFSET
Velocity Override	VELOVRD

Motor Table Variables – 16 bit integer (word)

Motor Table Variable Description	**Variable Name**
Axis Number	AXISNUM
IIR filter coefficients, maximum	COEFMAX
IIR filter coefficients, qty. defined	COEFCNT
Master Axis	MASTER
Input Mode	INPUTMODE
Integral Term Options	INTEGRALOPT
Derivative Term Sampling Frequency	SFREQ
Timer - Wait Stop	WAITSTOP
Timer - Wait Target	WAITTARGET

Single Stepping MCCL Programs

While the controller is executing any Motion Control Command Language (MCCL) macro program via WinControl the user can enable single step mode by entering <ctrl> <B>. Each time this keyboard sequence is entered, the next MCCL command in the program sequence will be executed. The following macro program will be used for this example of single stepping:

```
        MD10,WA1,1MR1000,1WS.1,1TP,1MR-1000,1WS.1,1TP,RP
```

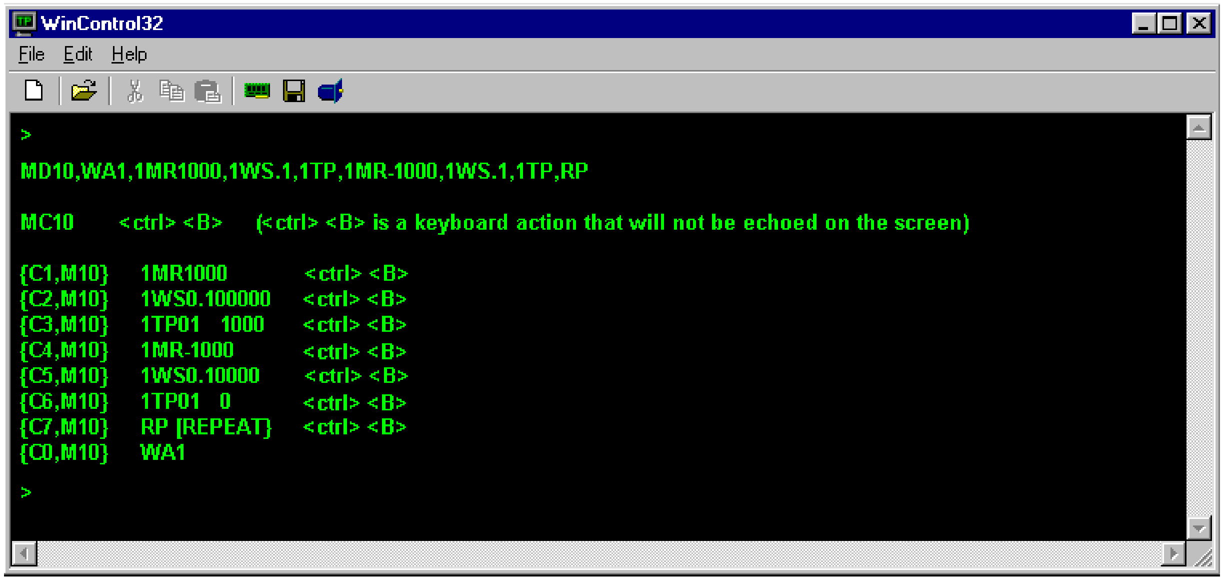
This sample program will: wait for 1 second, move 1000 encoder counts, report the position 100 msec's after the calculated trajectory is complete, move -1000 encoder counts, report the position 100 msec's after the calculated trajectory is complete, repeat the command sequence.

This command sequence can be entered directly into the memory of the controller by typing the command sequence in the terminal interface program WinCtl32.exe or by downloading a text file via WinControl's file menu.

To begin single step execution of the above example macro enter MC10 (call macro #10) then <ctrl> <B> the following will be displayed:

```
    {C1,MC10} 1MR1000 <
```

The display format of single step mode is: {Command #,Macro #} Next command to be executed



```
>
MD10,WA1,1MR1000,1WS.1,1TP,1MR-1000,1WS.1,1TP,RP

MC10     <ctrl> <B>    (<ctrl> <B> is a keyboard action that will not be echoed on the screen)

{C1,M10}    1MR1000         <ctrl> <B>
{C2,M10}    1WS0.100000     <ctrl> <B>
{C3,M10}    1TP01   1000    <ctrl> <B>
{C4,M10}    1MR-1000        <ctrl> <B>
{C5,M10}    1WS0.10000      <ctrl> <B>
{C6,M10}    1TP01   0       <ctrl> <B>
{C7,M10}    RP [REPEAT]     <ctrl> <B>
{C0,M10}    WA1

>
```

To end single stepping and return to immediate MCCL command execution press <Enter>. To abort the MCCL program enter <Escape>. Single step mode is not supported for a MCCL sequence that is executing as a background task.

Single stepping can also be enabled from within a MCCL program by using the break command immediately followed by a "string" parameter. When the break command is executed the controller will display the characters in the string (inside the quotation marks) and then delay additional command execution until the space bar (execute next command and then delay) or the enter key (terminate single stepping and resume program execution) is pressed. In the following example axis one will move 1000 counts, report the position, move –1000 counts, report the position, halt command execution until the enter key is pressed, repeat one time.

```
MC10 1MR1000,1WS0.100000,1TP,1MR-1000,1WS0.100000,1TP,BK"wait",RP1
```

```
>mc10
01 997
01 0
BREAK AT COMMAND 6, MACRO 10
wait
 {C7,M10} RP10 [REPEAT] <
   <enter>
01 997
01 0
BREAK AT COMMAND 6, MACRO 10
wait
 {C7,M10} RP10 [REPEAT] <
>  <enter>
```

Controller User Registers

The controller contains 1000 general purpose global registers that can be used for; storing command parameters, performing math computations and controlling command execution. The registers are numbered 0 through 999, with register 0 being the 'accumulator'. The accumulator (register 0) is used by all commands that manipulate register data.

Each register can hold a 32 bit integer, a 32 bit single precision floating point number, or a 64 bit double precision floating point number. A register will be loaded with the double precision floating point number if the **A**ccumulator **L**oad (**AL*n***) command is issued with a parameter containing a decimal point. Otherwise, the register will be loaded with a 32 bit integer. When executing commands that perform math operations on the accumulator (AA, AD, AM, ...), the result will have the same precision as the command parameter or the accumulator (prior to the command), whichever is more precise. Since the 32 bit integer is considered to be the least precise, multiplying an integer by a floating point number will always result in a floating point number. If a floating point indirect parameter is used for a command that does not support floating point parameters (e.g. CN, LM, PC,...), the register contents will be rounded to the nearest integer prior to use.

Typically the user issues commands with 'immediate' parameters (i.e.: the parameter ‘n’ is a constant). The user can also issue commands, specifying that the parameter is the contents of a register. This is done by replacing the command parameter with the register number preceded with an '@' sign. For example, the command "1MR@10" will cause the controller to move axis 1 by the number stored in register 10. The use of a register specifier can be used in any command as the parameter. The controller ***does not*** support the use of the '@' sign in front of an axis number. The following commands are available for working with the registers:

MCCL Command	Description
AAn	Accumulator Add (ACC = ACC + n)
ACn	Accumulator Complement, bit wise (ACC = !ACC)
ADn	Accumulator Divide (ACC = ACC/n)
AEn	Accumulator logical Exclusive or with n, bit wise (ACC = ACC eor n)
ALn	Accumulator Load with constant n (ACC = n)
AMn	Accumulator Multiply(ACC = ACC x n)
ANn	Accumulator logical aNd with n, bit wise (ACC = ACC and n)
AOn	Accumulator logical Or with n, bit wise (ACC = ACC or n)
ARn	copy Accumulator to Register n (REGn = ACC)
ASn	Accumulator Subtract (ACC = ACC - n)
GAx	Get Analog value (ACC = channel x)
aGX	Get auXiliary encoder position (ACC = axis a auxiliary encoder)
IBn	If accumulator is Below (>) n, do next command, else skip 2 commands
ICn	If bit n of accumulator is Clear, do next command, else skip 2 commands
IEn	If accumulator Equals constant n, do next command, else skip 2 commands
IGn	If accumulator is Greater than ‘n’, do next command, else skip 2 commands
OAx	Output Analog value (channel x = ACC)
ISn	If bit n of accumulator is Set, do next command, else skip 2 commands
IUn	If accumulator is Unequal to ‘n’, do next command, else skip 2 commands
RAn	copy Register n to Accumulator (ACC = REGn)
SLn	Shift Left accumulator n bits (ACC = ACC << n)
SRn	Shift Right accumulator n bits (ACC = ACC >> n)
TRn.p	Tell contents of Register n
TR.p	Tell contents of accumulator (register 0)

MultiFlex Scratch Pad Memory

Over and above what is available by using the User Registers, the controller also provides user scratch pad memory. By default this scratch pad memory area is 64KB, but it may be reduced by user defined macros or other memory operations. The **ME**mory allocate (**ME*n***) command is used to format local memory space for user operations. Parameter *n* of the ME command defines the number of bytes to be allocated for scratch pad memory.

Upon executing the ME command, the accumulator will be loaded with the address of the first byte of allocated memory. If the requested amount of allocated memory is not available a zero will be loaded into the accumulator. The following commands are used to write data from the accumulator into the allocated memory locations:

WBn	Write accumulator low Byte (8 bit) to memory location n ((n) = ACC)
WDn	Write accumulator Double to absolute memory location n ((n) = ACC)
WLn	Write accumulator Long (32 bit) to memory location n ((n) = ACC)
WVn	Write accumulator float to absolute memory location n ((n) = ACC)
WWn	Write accumulator low Word (16 bit) to memory location n ((n) = ACC)

The following commands are used to read data from the allocated memory into the accumulator:

RBn	Read Byte (8 bit) at memory location n into accumulator (ACC = (n))
RDn	Read Double at memory location n into accumulator (ACC = (n))
RLn	Read Long (32 bit) at memory location n into accumulator (ACC = (n))
RVn	Read float at memory location n into accumulator (ACC = (n))
RWn	Read Word (16 bit) at memory location n into accumulator (ACC = (n))

The **F**ree **M**emory (**FM*n***) command returns previously allocated memory to the ‘heap’ from which it was allocated. Parameter *n* of this command must be the same as the value that was loaded into the accumulator upon issuing the MEmory allocate (ME) command.

Chapter Contents

Setup Commands

MCCL	Description
AG	set Acceleration feed-forward Gain
AH	Auxiliary encoder define Home
BD	Backlash compensation Distance
DB	set position DeadBand
DG	set Deceleration feed-forward Gain
DH	Define Home
DI	DIrection
DS	Deceleration Set
DT	Delay at Target
ES	Encoder counts / Steps Scale
EE	Encoder fail Enable
FC	Full Current
FF	amplifier Fault input ofF
FN	amplifier Fault input oN
FR	set derivative sampling period
HC	Half Current
HL	set motion High Limit
HS	set High Speed
IL	set Integration Limit
IO	Integral option
LF	motion Limits ofF
LL	set motion Low Limit
LM	Limit Mode
LN	motion Limits oN
LS	set Low Speed
MS	set Medium Speed
MV	set Minimum Velocity

Setup Commands continued

MCCL	Description
OB	set the Output deadBand
OM	set Output Mode
OO	set the Output Offset
PH	set servo output PHase
PK	Pwm time Konstant
SA	Set Acceleration
SD	Set Derivative gain
SE	Stop on Error
SG	Set prop. Gain of motor
SI	Set Integral gain
SQ	Set TorQue
SS	Set Slave ratio
SV	Set Velocity
UA	Use as default Axis
UK	set User output constant
UO	set User Offset
UR	set User Rate conversion
US	set User Scale
UT	set User Time conversion
UZ	set User Zero
VA	set Vector Acceleration
VD	set Vector Deceleration
VG	set Velocity Gain
VO	set Velocity Override
VV	set Vector Velocity

Chapter 5

Setup Commands

AG Acceleration Gain

MCCL command: *a*AG*n* *a* = Axis number *n* = integer or real
applies to: Analog Command Axis
see also: DG, VG

This command sets the acceleration feed-forward gain for a servo. The product of this gain and the motor's calculated acceleration will be summed into the controller's DAC output. The acceleration gain is only applied while a motor is accelerating, when it decelerates the deceleration gain term is used (see DG command). Typically acceleration gain parameters are negative values.

comment: Acceleration and deceleration feed-forwards are not calculated when a motor is in contour mode.

```
      1AG-0.00002                              ;Sets Acceleration gain for axis 1 to
                                               ;-0.00002
```

AH Auxiliary encoder define Home

MCCL command: *a*AH*n* *a* = Axis number *n* = integer or real
applies to: Pulse Command Axis
see also: AF, DH

This command causes axis *a* auxiliary encoder position to be set to *n*. For defining the home position of the Analog Command Axis, see the Define Home (aDH) command.

BD Backlash compensation Distance

MCCL command: *a*BD*n* *a* = Axis number *n* = integer or real

applies to: Analog Command Axis
see also: BF, BN

Use this command to set the distance required to nullify the effects of mechanical backlash in the system. The command parameter should be equal to half the amount the motor must move to take up backlash when it changes direction. The units for this command parameter are encoder counts, or the units established by the User Scale command for the axis.

Once the backlash compensation distance is set, issuing the Backlash compensation oN command will cause the controller to add or subtract the distance from the motor's commanded position during all subsequent moves. If the motor moves in a positive direction, the distance will be added; if the motor moves in a negative direction, it will be subtracted. When the motor finishes a move, it will remain in the compensated position until the next move.

```
1BD100                          ;define backlash distance
1BN                             ;enable backlash compensation

1BF                             ;disable backlash compensation
```

DB set position **D**ead**B**and

MCCL command: *a*DB*n*   *a* = Axis number   *n* = integer or real >= 0
applies to: Analog Command Axis
see also: DT, WT

This command sets the position dead band that is used by the controller to determine when a servo axis is 'At Target'. In order for the At Target flag in the motor status to be set, a servo must remain within the specified dead band of the current target position for a period of time specified with the Delay at Target (aDTn) command.

```
1DB5                            ;deadband range = +/- 5 encoder counts
1DT0.0025                       ;at target time = 25 msec's
1MR1000,1WT                     ;move axis #1 1000 counts, wait until 'at
                               ;target conditions are met
```

DG **D**eceleration **G**ain

MCCL command: *a*DG*n*   *a* = Axis number   *n* = integer or real
applies to: Analog Command Axis
see also: AG, VG

This command sets the deceleration feed-forward gain for a servo. The product of this gain and the motor's calculated deceleration will be summed into the controller's DAC output. The deceleration gain is only applied while a motor is decelerating. When it accelerates the acceleration gain term is used (see AG command). Typically deceleration gain parameters are negative values.

comment: Acceleration and deceleration feed-forwards are not calculated when a motor is in contour mode.

```
      1DG-0.00002                              ;Sets Deceleration gain for axis 1 to
                                               ;-0.00002
```

DH Define Home

MCCL command:	*a*DH*n* *a* = Axis number *n* = integer or real
applies to:	Analog Command Axis, Pulse Command Axis
see also:	EL, FE, FI, IA, WE, WI

Defines the current position of a motor to be *n*. From then on, all positions reported for that motor will be relative to that point.

```
      1DH500                                   ;define the current location of axis
                                               ;#1 as position 500
```

DI DIrection

MCCL command:	*a*DI*n* *a* = Axis number *n* = integer 0 or 1
applies to:	Analog Command Axis, Pulse Command Axis
see also:	GO, VM

Sets the move direction of a motor when in velocity mode. A parameter value of 0 results in motion in the positive direction, a value of 1 causes motion in the negative direction.

```
      1VM,2VM                                  ;config. axes 1 & 2 for Velocity mode
      1DI0                                     ;set positive direction for axis 1
      2DI1                                     ;set positive direction for axis 2
      1GO,2GO                                  ;begin axes 1 & 2 velocity mode motion
```

DS Deceleration Set

MCCL command:	*a*DS*n* *a* = Axis number *n* = integer or real > 0
applies to:	Analog Command Axis, Pulse Command Axis
see also:	SA, SV

Defines the deceleration rate for an axis. The default units for the command parameter are encoder counts (or steps) per second per second.

```
      1SV100000                  ;set max. velocity = 100,000 counts or steps / second
      1SA200000                  ;accelerate to max. velocity in 0.5 seconds
      1DS200000                  ;decelerate to zero velocity in 0.5 seconds
```

DT Delay at Target

MCCL command: *a*DT*n* *a* = Axis number *n* = integer or real >= 0
applies to: Analog Command Axis
see also: DB, WT

This command sets the time period during which a servo must remain within the position dead band of the target for the 'At Target' flag in the motor status to be set.

```
        1DB5                                 ;deadband range = +/- 5 encoder counts
        1DT0.0025                            ;at target time = 25 msec's
        1MR1000,1WT                          ;move axis #1 1000 counts, wait until 'at
                                            ;target conditions are met
```

EE Encoder fail Enable

MCCL command: *a*EE*n* *a* = Axis number *n* = 1 - 8

applies to: Analog Command Axis, Pulse Command Axis
see also: TS

The differential receivers used by the controller to interface to an encoder provide encoder fail detection circuitry. By issuing the Encoder fail Enable command hard coded error checking of the A+/A-, B+/B-, & Z+/Z- signals will be performed. Once encoder fail detection has been enabled (aEE1) and the axis has been turned on (aMN), if an encoder failure is detected status bit 14 (Primary Encoder Fault Tripped) will be set. To clear an encoder fail error issue the Motor oN command. Note: encoder fail detection for single ended encoders requires that the A-, B-, and Z+/Z- inputs be preperly terminated to the 1.5V encoder reference.

n =	Encoder fail Enable parameter n function
0	**Encoder fail detection disabled**
1	Enable encoder fail error detection
2	Enable encoder fail error detection for the auxiliary encoder of a servo axis (not supported at this time)

EI Enable bus Interrupts

MCCL command: *a*EI*n* *a* = Axis number *n* = integer >= 0
applies to: Analog Command Axis, Pulse Command Axis
see also: TS

This command is used to enable host interrupts. Parameter *n* defines the status bits for which host interrupts will be enabled. To disable interrupts issue the EI command with *n* = 0.
Note: The Enable Interrupt (EI) command should only be issued by the MCAPI function ***MCEnableInterrupt( )***. Issuing this MCCL command directly to the controller via WinControl will cause a malfunction.

```
        1EI9                              ;Enable host interrupts for Motor Error (bit 0)
                                           ;and Trajectory Complete (bit 3)
```

ES Encoder Scale

MCCL command:	*a*ES*n* *a* = Axis number *n* = integer or real >= 0
applies to:	Pulse Command Axis (Closed Loop)
see also:	IM

Defines the steps to encoder count ratio for a closed loop stepper axis. This parameter must be set before commanding closed loop stepper motion. The default value is 1.

```
5MF                                     ;turn motor off
5HS                                     ;select high speed mode
5ES25.6                                 ;divide steps per rotation (51200)
                                        ; by encoder counts per rotation
                                        ;(2000)scaling
5IM1                                    ;enable closed loop stepper mode
5SV35000,5SA170000,5DS170000,5MV3500
                                        ;set trajectory parameters (in encoder
                                        ;units)
5MF,5MN                                 ;turn motor on to initialize
                                        ;position registers
```

FC **F**ull **C**urrent

MCCL command :	*a*FC *a* = Axis number
applies to:	Pulse Command Axis
see also:	HC

Causes Full/Half Current output signal of a stepper to go low.

```
5FC                                     ;turn on axis #5 full current output
```

FF amplifier **F**ault o**F**f

MCCL command:	*a*FF *a* = Axis number
applies to:	Analog Command Axis, Pulse Command Axis
see also:	FN

Disables the error checking of the Amplifier Fault input of a servo control axis or the Driver Fault input of a stepper control axis. See description of amplifier Fault input oN command (FN), for further details.

FN amplifier **F**ault o**N**

MCCL command : aFN *a* = Axis number
applies to: Analog Command Axis, Pulse Command Axis
see also: FF, TS

Enables the error checking of the Amplifier Fault input of a servo axis or the Driver Fault input of a stepper axis. If the Amp Fault / Driver Fault input goes active after this command is executed, the motor will be turned off and the amplifier fault tripped flag (bit 9) in the status word will be set. The tripped flag will remain set until the motor is turned back on with the MN command.

FR set the derivative sampling period

MCCL command: a*FRn* *a* = Axis number *n* = integer >= 0
applies to: Analog Command Axis
see also: SD, SG

Helps tune servo loop to the inertial characteristics of system. High inertial loads normally require a longer period and low inertial loads a shorter period. The default value is 0 (0.00025 seconds). For a value of *n*, the derivative sampling period will be (n +1) * (sample period). See the loop rate commands (HS, MS, LS) for a discussion of the sample period on servos.

```
1FR0.00025                ;set derivative sampling rate for low friction
                           ;servo system
2FR0.0015                 ;set derivative sampling rate for high inertia
                           ;servo system
```

HC **H**alf **C**urrent

MCCL command: *a*HC *a* = Axis number
applies to: Pulse Command Axis
see also: FC

Causes Full/Half Current output signal of a stepper axis to go high.

```
5HC                                    ;turn on axis #5 full current output
```

HL **H**igh motion soft **L**imit

MCCL command: *a*HL*n* *a* = Axis number *n* = integer or real
applies to: Analog Command Axis, Pulse Command Axis
see also: LF, LL, LM, LN

This command sets the high limit for motion. After this command is issued, and the motion limit is enabled with the Limit oN (aLNn) command, the command parameter is used as a 'soft' limit for all motion of the axis. If the desired or true position of the axis is greater than this limit, and the axis is being commanded to move in the positive direction, the Soft Motion Limit High and the Motor Error flags in the motor status will be set. The axis will be turned off, stopped abruptly, or stopped smoothly, depending upon the mode set by the Limit Mode command.

comment: When the axis is in contouring mode, this limit will be tripped anytime the desired or true position is greater than the limit regardless of commanded direction. Thus, to move an axis out of the limit region, it must be placed in a non-contour mode. When one or more axes are moving in contour mode, and one of the axes experiences a limit trip, all axes associated with the motion will be turned off or stopped.

```
1LM10                                  ;when hard or soft limit is tripped
                                       ;decelerate to a stop (PID remain
                                       ;active)
1HL100000                             ;high soft limit = 100,000
1LL-100000                             ;high soft limit = -100,000
1LN0                                    ;enable both hard and soft limits
                                       ;error checking
```

HS **H**igh **S**peed

MCCL command: *a*HS *a* = Axis number
applies to: Analog Command Axis, Pulse Command Axis
see also: LS, MS

This command has a different effect depending on whether it is issued to a servo or stepper motor axis. For an Analog Command Axis, it sets the feedback loop to 4 KHz update rate. For a Pulse Command Axis, it selects the low speed pulse rate range. In this mode the controller can generate pulse rates from 500 pulses/sec. to 5.0 Million pulses/sec.

```
5LS                                    ;set axis #5 (stepper) pulse rate range
                                       ;(500 pulses/Sec. – 5M pulses/Sec.)
```

IL **I**ntegration **L**imit

MCCL command: *a*IL*n* *a* = Axis number *n* = integer or real >= 0
applies to: Analog Command Axis, Pulse Command Axis (Closed Loop)
see also: IO, SI, SD, SG

Limits level of power (maximum command voltage) that integral gain can use to reduce the position error of a closed loop axis. The default units for the command parameter are (encoder counts) * (PID update period).

```
1IL50                                  ;set Integration Limit
```

```
Integral Term Output Voltage =
     IntegralGain * IntegralLimit(sum(FollowingError))
```

IO Integral Option

MCCL command : a**IO***n* *a* = Axis number *n* = integer 0, 1, or 2
applies to: Analog Command Axis, Pulse Command Axis (Closed Loop)
see also: IL, SI, WS

The integral term accumulates the position error for servos (and closed loop stepper) and generates an output signal to reduce the position error to zero. The integral gain determines the magnitude of this term. The default value is zero. The **I**ntegral **O**ption command allows the user to define how the integral term of the PID filter functions while a servo is moving.

n =	**Integral term function**
0	**Integral term always on (default)**
1	Freezes accumulation of integration term during movement. Integration will continued once the calculated trajectory (trajectory complete, status bit 3 = 1) has been completed.
2	Zero integration term when motion begins. When the calculated trajectory (trajectory complete, status bit 3 = 1) has completed, enable the integration term

```
1si.0065                        ;set integral gain
1io2                            ;integral option = zero (wait until
                                ;trajectory complete)
1il50.0                         ;set integral limit
```

LF motion Limits oFf

MCCL command: *a*LF*n* *a* = Axis number *n* = 0 – 15
applies to: Analog Command Axis, Pulse Command Axis
see also: LN, LM

Disables one or more 'hard' limit switch inputs or 'soft' position limits for an axis. The parameter to this command determines which limits will be disabled. The coding of the parameter is the same as for the motion Limits oN command (LN).

LL Low motion soft Limit

MCCL command: *a*LL*n* *a* = Axis number *n* = integer or real
applies to: Analog Command Axis, Pulse Command Axis
see also: LF, LH, LM, LN

This command sets the low limit for motion. After this command is issued, and the motion limit is enabled with the Limit oN (aLNn) command, the command parameter is used as a 'soft' limit for all motion of the axis. If the desired or true position of the axis is less than this limit, and the axis is being commanded to move in the negative direction, the Soft Motion Limit Low and the Motor Error flags in the motor status will be set. The axis will also be turned off, stopped abruptly, or stopped smoothly, depending upon the mode set by the Limit Mode command.

comment: When the axis is in contouring mode, this limit will be tripped anytime the desired or true position is less than the limit regardless of commanded direction. Thus, to move an axis out of the limit region, it must be placed in a non-contour mode. When one or more axes are moving in contour mode, and one of the axes experiences a limit trip, all axes associated with the motion will be turned off or stopped.

```
1LM10                                   ;when hard or soft limit is tripped
                                        ;decelerate to a stop (PID remain
                                        ;active)
1HL100000                              ;high soft limit = 100,000
1LL-100000                              ;high soft limit = -100,000
1LN0                                     ;enable both hard and soft limits
                                        ;error checking
```

LM Limit Mode

MCCL command: *a*LM*n*   *a* = Axis number   *n* = integer 0 - 136
applies to: Analog Command Axis, Pulse Command Axis
see also: LF, LN

This command is used to select how the controller will react when a 'hard' limit switch or a 'soft' position limit is tripped by an axis. The command parameter should be formed by adding a value of 0, 1, or 2 for the hard limit switch mode, to a value of 0, 4, or 8 for the soft position limit mode. In all cases the Motor Error and one of Limit Tripped flags in the status word will be set when a limit event occurs. This will prevent the controller from moving the motor until a Motor oN command is issued.

Parameter n =	**Desired action**
0,0 *	Turn motor off (disable PID) when **hard limit sensor ‘goes’ active** or soft motion limit is exceeded
1,4 *	Stop the motor abruptly (under PID control) when **hard limit sensor ‘goes’ active** or soft motion limit is exceeded
2,8 *	Decelerate and stop the motor (under PID control) when **hard limit sensor ‘goes’ active** or the soft motion limit is exceeded. Use the current deceleration setting.
128 **	Invert the active level of the hard limit input. Typically used for normally open hard limit sensors

* Values in red are for defining the Limit Mode for hard limits. Values in black are for defining the mode for soft motion limits. When using both hard and soft limits, parameter n should equal hard limit parameter n + soft limit parameter n.
** For inverted active level hard limits, parameter n = 128 plus desired mode

```
1LM130                     ;Axis #1 Limit mode = decelerate & stop when hard
                           ;limit activated (n=2) + invert active level(n=128)
```

LN Limits oN

MCCL command: *a*LN*n* *a* = Axis number *n* = 0 - 15
applies to: Analog Command Axis, Pulse Command Axis
see also: LF, LM

This command is used to enable the 'hard' limit switch inputs and/or the 'soft' position limits of an axis. If a limit switch input goes active after it has been enabled by this command, and the motor has been commanded to move in the direction of that switch, the Motor Error and one of the Hard Limit Tripped Flags will be set in the motor status. At the same time the motor will be turned off or stopped (depending on the value of parameter *n* of the **L**imit **M**ode command). If a soft motion limit is enabled, and the respective axis is commanded to move beyond the motion limits set by the High motion Limit and the Low motion Limit commands, the Motor Error and one of the Soft Limit Tripped Flags will be set. At the same time the motor will be turned off or stopped (depending on the value of parameter *n* of the **L**imit **M**ode command). The flags will remain set until the motor is turned back on with the MN command. Once the motor is turned back on, it can be moved out of the limit region with any of the standard motion commands. The parameter to this command determines which of the hard and soft limits will be enabled. See the description of **Motion Limits** in the **Motion Control** chapter.

The **LN** command enables hard coded limit error checking.

Parameter n	Desired action
0*	Enable both hard limits (+/-) and soft limits (high & low)
1**	Enable hard limit + error checking
2**	Enable hard limit – error checking
3**	Enable hard limit + and hard limit – error checking
4**	Enable high soft limit error checking
8**	Enable low soft limit error checking
12**	Enable high & low soft limit error checking

* If parameter n = 0 both hard and soft limit error checking will be enabled.

** Values in red are for enabling limit error checking for hard limits. if both hard and soft limits are to be used the parameter ***n*** should equal hard limit parameter ***n*** + soft limit parameter ***n***.

```
1LN0                         ;Axis #1 - enable hard and soft limits

2LN7                         ;Axis #2 – enable both hard limits (n=3) and
                             ;high soft motion limit (n=4)
```

LS Low Speed

MCCL command: *a*LS *a* = Axis number
applies to: Analog Command Axis, Pulse Command Axis
see also: HS, MS

This command has a different effect depending on whether it is issued to a servo or stepper motor axis. For an Analog Command Axis, it sets the feedback loop to 1 KHz update rate. For a Pulse Command Axis, it selects the low speed pulse rate range. In this mode the controller can generate pulse rates from 0.1 pulses/sec. to 78,000 pulses/sec.

```
       5LS                                     ;set axis #5 (stepper) pulse rate range
                                               ;(0.1 Steps/Sec. – 78K Steps/Sec.)
```

MS   **M**edium **S**peed

MCCL command:	*a*MS *a* = Axis number
applies to:	Analog Command Axis, Pulse Command Axis
see also:	HS, LS

This command has a different effect depending on whether it is issued to a servo or stepper motor axis. For an Analog Command Axis, it sets the feedback loop to 2 KHz update rate. For a Pulse Command Axis, it selects the low speed pulse rate range. In this mode the controller can generate pulse rates from 500 pulses/sec. to 650,000 pulses/sec.

```
       5MS                                     ;set axis #5 (stepper) pulse rate range
                                               ;(500 Steps/Sec. – 625K Steps/Sec.)
```

MV   set **M**inimum **V**elocity

MCCL command:	*a*MV*n* *a* = Axis number *n* = integer or real >= 0
applies to:	Pulse Command Axis
see also:	SV

Sets the minimum velocity for a given Pulse Command Axis. The purpose of this command is to set an initial and final velocity for motion of stepper motors. Below this velocity a full stepping motor is 'cogging' between steps. The default units for the command parameter are steps per second. This command will have no effect on servos. The Minimum Velocity is typically set to between **1% and 10% of the maximum velocity**

```
       5SV250000                  ;set max. velocity of axis #5 (pulse command)
       5MV2500                    ;minimum velocity = 1% of max. velocity
```

OB   **O**utput dead **B**and

MCCL command:	*a*OB*n* *a* = Axis number *n* = integer or real > 0, <=10
applies to:	Analog Command Axis, Pulse Command Axis
see also:	OO

This command can be used to simulate a ‘frictionless servo system’. Parameter ***n*** must be a positive real value between 0 and 10. Parameter ***n*** modifies the commanded analog output to a servo. This value will be added to or subtracted from the analog output. If the calculated command output voltage is positive, the voltage is increased by the output deadband voltage (n). If the calculated output voltage is negative, the command voltage is decreased by the deadband amount (n) . And if the calculated output is zero, the output will be 0 volts.

```
        2OB1.5                         ;set Output deadBand to a value (voltage level =
                                       ;1.5V) that is sufficient to overcome mechanical
                                       ;friction of a servo
```

OM Output Mode

MCCL command: *a*OM*n* *a* = Axis number *n* = integer 0, 1
applies to: Analog Command Axis, PWM Command Axis, Pulse Command Axis
see also:

This command is used to set a servo or stepper output mode. The available modes are listed in the following tables.

n	Servo Output Mode
0	**Bipolar Analog output, -10V to +10V (default)**
1	Unipolar Analog output, 0V to +10V *
2	Bipolar PWM output
3	Unipolar PWM output (requires user to define PWM Direction output channel)

* The direction of the servo is indicated by the Stepper Direction output (J1-3, J2-3, J3-3, J4-3)

n	Stepper Output Mode
0	**Pulse and Direction outputs (default)**
1	CW and CCW Pulse Outputs

```
        5OM0                           ;axis #1 (pulse) requires Pulse / Dir. outputs
        6OM0                           ;axis #2 (pulse) requires Pulse / Dir. outputs
        7OM1                           ;axis #3 (pulse) requires CW / CCW outputs
```

OO Output Offset

MCCL command: *a*OO*n* *a* = Axis number *n* = integer or real >= -10, <= +10
applies to: Analog Command Axis
see also: OD

This command is used to provide software programmability of the zero point of a servo output. Similar to adjusting an offset potentiometer, the parameter *n* will change the ‘zero’ output level.

```
        1OO-0.45                       ;set command output offset to -0.45 volts
```

PH set the servo output PHasing

MCCL command: *a*PH*n* *a* = Axis number *n* = 0 or 1
applies to: Analog Command Axis, Pulse Command Axis
see also: OM

This command is used to set a servo or closed loop stepper output phasing. The phase of the output will determine whether the axis drives the motor in a direction that reduces position error, or increases it. The axis defaults to standard phasing, which is the same as issuing this command with a parameter of 0. The output can be set to reverse phase by issuing this command with a parameter of 1. This command has no effect on the command output polarity when operating in Torque Mode (aQM).

```
1PH0                      ;servo axis #1 = standard phasing
1PH1                      ;servo axis #2 = reverse phasing (instead of
                          ;changing the wiring from the encoder)
```

PK PWM time Konstant

MCCL command:	PK*n* *n* = integer >= -10, <= +10
applies to:	PWM Command Servo Axis
see also:	OM

This command is used to set the PWM frequency of a PWM Command servo. The default value is 19.53 KHz (n = 32). To change the default frequency set parameter n to the desired frequency divided by 610. Parameter is an integer, so the PWM frequency is limited to increments of 610 Hz.

SA Set Acceleration

MCCL command:	*a*SA*n* *a* = Axis number *n* = integer or real >= 0
applies to:	Analog Command Axis, Pulse Command Axis
see also:	DS, SV

Set the maximum acceleration rate for a given axis. The default units for the command parameter are encoder counts (or steps) per second per second.

```
1SV100000                 ;max. velocity = 100,000 counts or pulses / second
1SA200000                 ;accelerate to max. velocity in 0.5 seconds
1DS200000                 ;decelerate to zero velocity in 0.5 seconds
```

SD Set Derivative gain

MCCL command:	*a*SD*n* *a* = Axis number *n* = integer or real >= 0
applies to:	Analog Command Axis, Pulse Command Axis (Closed Loop)
see also:	FR, IL, SI, SG

This command is used to set the derivative gain of a servo's or closed loop stepper feedback loop. Increasing the derivative gain has the effect of dampening oscillations.

```
        1sg.04                                  ;set proportional gain
        1sd.2                                   ;set derivative gain
        1fr7                                    ;set sampling period (2 msec.)
        1si.0065                                ;set integral gain
        1io2                                    ;integral option = zero (wait until
                                                ;trajectory complete)
        1il50.0                                 ;set integral limit
```

SE **S**top on following **E**rror

MCCL command : *a*SE*n* *a* = Axis number *n* = integer or real <= 0
applies to: Analog Command Axis, Pulse Command Axis (Closed Loop)
see also:

Used to set the maximum following or position error for a servo or closed loop stepper. Once this command is issued and the motor is on, if the position error exceeds the specified value the motor error status flag will be set, and the axis will be turned off. The error flag will remain set until the motor is turned back on with the MN command. Issuing this command with parameter *n* = 0 will disable errors on excessive following errors.

```
        1SE1024                                 :axis #1 (servo) following error =
                                                ;1024 encoder counts
        1SE100                                  :axis #2 (servo) following error = 100
                                                ;encoder counts
```

SG **S**et **P**roportional gain

MCCL command : *a*SG*n* *a* = Axis number *n* = integer or real >= 0.000153, <=10
applies to: Analog Command Axis, Pulse Command Axis (Closed Loop)
see also: IL, SI, SD

This command is used to set the proportional gain of a servo or closed loop stepper feedback loop. Increasing the proportional gain has the effect of stiffening the force holding a servo in position. The parameter to this command has default units of volts per encoder count. This command should not be used for open loop stepper axes.

```
        1sg.04                                  ;set proportional gain
        1sd.2                                   ;set derivative gain
        1fr7                                    ;set sampling period (2 msec.)
        1si.0065                                ;set integral gain
        1io2                                    ;integral option = zero (wait until
                                                ;trajectory complete)
        1il50.0                                 ;set integral limit
```

SI **S**et the **I**ntegral gain

MCCL command : *a*SI*n* *a* = Axis number *n* = integer or real >= 0
applies to: Analog Command Axis, Pulse Command Axis

see also: IL, IO, SI, SD, SG

The integral term accumulates the position error for a servo or closed loop stepper and generates an output signal to reduce the position error to zero. The integral gain determines the magnitude of this term. The default value is zero. Note that Integration Limit (IL) command must be set to a nonzero value before integral gain will have any effect.

```
      1si.0065                                ;set integral gain
      1io2                                    ;integral option = zero (wait until
                                              ;trajectory complete)
      1il50.0                                 ;set integral limit


Integral Term Output Voltage =
     IntegralGain * IntegralLimit(sum(FollowingError))
```

SQ **S**et tor**Q**ue

MCCL command: *a*SQ*n* *a* = Axis number *Torque Mode n* = integer or real >= -10, <= 10
Position/Velocity Mode n = integer or real >= 0, <= 10
applies to: Analog Command Axis
see also: QM, PM, VM

In Position Mode the SQ command will define (limit) the maximum voltage level of the analog command output for a servo. When in torQue Mode, this command sets a fixed output level, essentially turning the DAC output circuitry into a programmable power supply. The default units for the command parameter are volts.

comment: When the torQue Mode (aQM) command is issued the command output of the servo axis will be set to the value last defined by the SQ command (default = 10.0 volts).

```
      1SQ6.25                                 ;limit the voltage command of servo
                                              ;axis #1 to a range of +/- 6.25 volts

      2SQ0                                    ;set SQ=0 before entering torQue Mode
      2QM                                     ;enable torQue Mode for axis 2
      2SQ-3.85                                ;set axis #2 analog command output
                                              ;to -3.85 volts
      2MN                                     ;turn axis on
```

SS **S**et the ratio of **S**lave axis

MCCL command : *a*SS*n* *a* = Axis number *n* = integer or real > 0 <= 10 $^{-208}$
applies to: Analog Command Axis, Pulse Command Axis
see also: SM

This command specifies the ratio at which the slave axis (designated by *a*) will move relative to a changed in encoder counts (or steps) of the master axis. As soon as the Set Master command is issued, the slave axis will begin tracking the master axis with the programmed ratio. The controller makes the position calculations using the optimal positions of the master and slave axes when the Set Master command was issued as the starting point.

```
2SS5                                    ;axis #2 slave ration = 5:1
3SS-.1                                  ;axis #2 slave ration = 0.1:1
4SS-1000                                ;axis #2 slave ration = -1000:1

2SM1,3SM1,4SM1                          ;Enable gearing of axis 2, 3, and 4
                                        ;with axis #1 as the master

;terminate gearing
2SM0,3SM0,4SM0
2PM,3PM,4VM
1MN,2MN,3MN,4MN
```

SV Set Velocity

MCCL command : *a*SV*n* *a* = Axis number *n* = integer or real >= 0
applies to: Analog Command Axis, Pulse Command Axis
see also: DS, MV, SA

Set the maximum velocity for a given axis. The default units for the command parameter are encoder counts (or steps) per second.

```
1SV100000                 ;set max. velocity = 100,000 counts or steps /second
1SA200000                 ;accelerate to max. velocity in 0.5 seconds
1DS200000                 ;decelerate to zero velocity in 0.5 seconds
```

Note: When issued to a pulse command axis, if the Minimum Velocity (MV) setting is greater than Set Velocity parameter *n*, the Minimum Velocity value will be redefined to be equal to the Set Velocity parameter *n*.

UA Use Axis as default

MCCL command: UA*n* *n* = integer > 0, <= 4
applies to: Analog Command Axis, Pulse Command Axis
see also:

The controller defaults to setting the default axis to zero. If the user executes a motion or setup command with the axis specifier missing, the default axis will be used. In some cases a motion or setup command issued to axis zero commands that operation to all axes. By defining a non-zero default axis, the user can execute 'generic' macro's (no axis number specified) to any axis.

This command is used to define a default axis. After issuing this command, any commanded move, setup, etc. command that utilizes an axis designator (*a*) will execute the command to the axis specified by parameter n. To query the controller as to the current default axis use the **G**et defa**Ul**t axis (**GU**) command.

```
        MD10,MR1000                     ;Macro 10 will execute a relative move
                                        ;of 1000 counts to the default axis
                                        ;(defined by the User Axis command).
                                        ;Note that the move command does not
                                        ;include the axis designator a.
        UA1,MC10                        ;Define axis #1 as the default axis,
                                        ;call macro ten to move 1000 counts
        UA2,MC10                        ;Define axis #2 as the default axis,
                                        ;call macro ten to move 1000 counts
```

UK **U**ser **K**onstant

MCCL command: *a*UK*n* *a* = Axis number *n* = integer or real >= 0
applies to: Analog Command Axis
see also:

This command is used to define the units to be used for setting the feed forward (velocity, acceleration, & deceleration), output deadband, and output offset parameters. The default setting is 1.0. This command should be issued before issuing the Feed-forward (AG, DG, VG), Output Deadband, or Output Offset commands.

UO **U**ser **O**ffset

MCCL command: aUOn *a* = Axis number *n* = integer or real
applies to: Analog Command Axis, Pulse Command Axis
see also: UR, US, UT, UZ

The User Offset (*a*UO*n*) command allows the user to define a ‘work area’ zero position of the axis. Use parameter ***n*** to define the distance from the servo or stepper motor home position, to the machine zero position. This offset distance must use the same units as currently defined by set **U**ser **S**caling command. The **U**ser **O**ffset command does not change the index or home position of the servo or stepper motor, it only establishes an arbitrary zero position for the axis. Beginning with firmware rev. 2.8a an updated User Offset value will take affect immediately if the axis is disabled (otherwise the new value will not be incorporated until the axis in enabled / re-enabled).

```
        3UO12.25                    ;define user = 12.25 inches
        3MN
```

UR **U**ser **R**ate

MCCL command: *a*UR*n* *a* = Axis number *n* = integer or real >= 0
applies to: Analog Command Axis, Pulse Command Axis
see also: UO, US, UT, UZ

The User Rate (*a*UR*n*) command sets the time unit for velocity, acceleration and deceleration values. The controller defaults to counts or steps per second. If velocities are to be in units of inches per minute, the user time unit is a minute. Parameter ***n*** of the **UR** command is the number of seconds per 'user time unit'. If the velocity, acceleration and deceleration are to be specified in units of inches per minute (and inches per minute per minute) for axis 1, then parameter ***n*** should be set to 60 seconds/1 minute = 60 (1UR60). Beginning with firmware rev. 2.8a an updated User Rate value will take affect immediately if the axis is disabled (otherwise the new value will not be incorporated until the axis in enabled / re-enabled).

```
3UR60
3MN
```

Typical User Rate values

Time Unit	User Rate Conversion
second	1 (default)
minute	60
hour	3600

US User Scale

MCCL command : *a*US*n* *a* = Axis number *n* = integer or real
applies to: Analog Command Axis, Pulse Command Axis
see also: UO, UR, UT, UZ

The User Scale (aUSn) command is used to define the number of encoder counts or steps per user unit. The default setting is 1.0. Beginning with firmware rev. 2.8a an updated User Scale value will take affect immediately if the axis is disabled (otherwise the new value will not be incorporated until the axis in enabled / re-enabled).

For example, if the servo encoder on axis 1 has 1000 quadrature counts per rotation, and the mechanics move 1 inch per rotation of the servo, then to setup the controller for user units of inches:

```
3US1000                    ;axis #3 user scale = 1000 (1000 encoder counts per
                            ;inch of travel)
3MN                        ;turn on axis

3SV1.0,1SA10,1DS10         ;define trajectory parameters for new scaling
                            ;parameters

3MR2.5                     ;move 2.5 inches
```

UT User Time

MCCL command: *a*UT*n* *a* = Axis number *n* = integer or real >= 0
applies to: Analog Command Axis, Pulse Command Axis
see also: UO, UR, US, UZ

The default unit for dwell commands (WAit, Wait for Stop, Wait for Target) is seconds. The User Time (UTn) command allows the user to change the units for dwell commands. Parameter **n** is the number of 1 second periods in the user's unit of time. If the user prefers time parameters in units of minutes, parameter n = 60 should be issued.

```
UT60
```

comment: When The UT command only effects the dwell time in the **task or command interface** from which it was issued. Each command interface (Binary, ASCII) and tasks maintain separate dwell time settings.

UZ set the **U**ser **Z**ero position

MCCL command: *a*UZ*n*   *a* = Axis number   *n* = integer or real
applies to: Analog Command Axis, Pulse Command Axis
see also: UO, UR, US, UT

The User Zero (*a*UZ*n*) command would typically be used in conjunction with the User Offset to define a ‘part zero’ position. A PCB (Printed Circuit Board) pick and place operation is a good example of how this function would be used. After a new PCB is loaded and clamped into place the X and Y axes would be homed. The **UO** command is used to define the ‘work area’ zero of the PCB. The **UZ** command is used to define the ‘part program’ or ‘local’ zero position. This way a single ‘part placement program’ can be developed for the PCB type, and a ‘step and repeat’ operation can be used to assemble multiple part assemblies. Beginning with firmware rev. 2.8a an updated User Zero value will take affect immediately if the axis is disabled (otherwise the new value will not be incorporated until the axis in enabled / re-enabled).

```
3UO12.25                         ;define offset to 12.25 inches
3UZ1.25                          ;define ‘part zero’ to 1.25 inches
3MN
```

VA **V**ector **A**cceleration

MCCL command: *a*VA*n*   *a* = Axis number   *n* = integer or real >= 0
applies to: Analog Command Axis, Pulse Command Axis
see also: CP, VD, VV

This command specifies the acceleration rate for motion along a contour path. It should be issued to the controlling axis prior to the first Contour Path command. If issued to the controlling axis while motion is in progress, the command will take effect immediately, and will be used for all succeeding motion.

```
   1VA50000                      ;Set the vector acceleration to 50,000
                                 ;counts or steps / second / second
```

VD **V**ector **D**eceleration

MCCL command: *a*VD*n*   *a* = Axis number   *n* = integer or real >= 0
applies to: Analog Command Axis, Pulse Command Axis
see also: CP, VA, VV

This command specifies the deceleration rate for motion along a contour path. It should be issued to the controlling axis prior to the first Contour Path command. If issued to the controlling axis while motion is in progress, the command will take effect immediately, and will be used for all succeeding motion.

```
        1VD50000                                ;Set the vector deceleration to 50,000
                                                ;counts or steps / second / second
```

VG Velocity Gain

MCCL command : aVGn *a* = Axis number *n* = integer or real
applies to: Analog Command Axis
see also: AG, DG

Sets the feed forward gain of the servo PID-FF loop. The default units for the parameter to this command are volts per encoder counts per second. For example, if the Velocity gain of a servo is set to -0.0001, the feed forward component of the output will be -1 volt at a speed of 10000 encoder counts per second (-0.0001 * 10000 = -1). Typically velocity gain parameters are expressed as negative numbers.

```
        1VG-0.00009                                     ;Sets Velocity gain for axis 1 to
                                                        ;-0.00009
```

VO Velocity Override

MCCL command : aVOn *a* = Axis number *n* = integer or real >= 0
applies to: Analog Command Axis, Pulse Command Axis
see also: CP, VV

Sets a multiplying factor that will be applied to the velocity of a servo or stepper. The default setting (multiplying factor) for this command is 1. For contour moves (linear, circular) the axis identifier ‘a’ should be the axis number of the ‘controlling’ axis of the contour group.

```
        1VO0.5                    ;reduce max. velocity for axis #1 by 50%
        2VO1.25                   ;increase axis #2 max. velocity by 25%
```

VV Vector Velocity

MCCL command: aVV*n* *a* = Axis number *n* = integer or real >= 0
applies to: Analog Command Axis, Pulse Command Axis
see also: CP, VA, VD

This command specifies the maximum velocity for motion along a contour path. It should be issued to the controlling axis prior to the first Contour Path command. When a Contour Path command is issued, the current vector velocity will be stored with the move in the motion table. The Vector Velocity command can also be issued to the controlling axis while motion is in progress, but it won't have any effect on the contour path motions already issued (loaded into the contouring buffer). To adjust the velocity of motions already in progress, use the Velocity Override command

```
1CM1                      ;Axis #1 is controlling axis of a contour group
2CM1                      ;Axis #2 is a member of the axis #1 contour group
1VA50000,1VD50000         ;set accel & decel for contour group


1CP1,1MA10000,2MA20000,1VV25000         ;execute linear interpolated move with
vector velocity of 25000 counts / steps per second
```

Chapter Contents

Mode Commands

MCCL	Description
CM	enable Contour Mode (arcs and lines)
GM	enable Gain Mode (no velocity profile)
IM	Input Mode (closed loop stepper)
PM	enable Position Mode
SM	enable Master/Slave mode
QM	enable TorQue mode
VM	enable Velocity Mode

Chapter 6

Mode Commands

CM Contour Mode

MCCL command: *a*CM*n* *a* = Axis number *n* = integer > 0, <= 8
applies to: Analog Command Axis, Pulse Command Axis
see also: CP

This command places a servo or stepper motor in the Contour Mode of operation. The parameter *n* to this command specifies the controlling axis for a group of axes to be included in contour path commands. The controlling axis should be the lowest numbered axis in the group. Contour Mode is terminated by issuing a Position (aPM) or Velocity Mode (aVMn) command to all axes in the group. The controlling axis should be taken out of contour mode last.

```
1CM1                        ;Axis #1 is controlling axis of a contour group
2CM1                        ;Axis #2 is a member of the axis #1 contour group

2PM                         ;Terminate contour mode
1PM                         ;Terminate contour mode
```

GM enable Gain Mode

MCCL command: *a*GM *a* = Axis number
applies to: Analog Command Axis
see also: IL, SD, SG, SI

This command places a servo in the Gain Mode of operation. In this mode, the servo can be commanded to execute moves to specific positions. However, no velocity profile (maximum velocity, acceleration, or deceleration) will be calculated. The servo will be driven to the new target based **only upon** the output of the PID loop.

comment: Typically this mode of operation is only used to tune a servo axis.

```
1GM                                    ;configure axis #1 for gain mode
1MR1000                                   ;move 1000 encoder counts
```

IM Input Mode

MCCL command:	*a*IM*n* *a* = Axis number *n* = 0 or 1 *n* = 0, 1, 128
applies to:	Pulse Command Axis
see also:	SE

The Input Mode command issued with a parameter of 1 enables closed loop stepper motion. Issued with *n* = 0 disables closed loop motion. See the description of **controller Stepper Basics** in the **Motion Control** chapter. If parameter *n* is set to 128 the user can change pulse command axis PID parameters

PM Position Mode

MCCL command :	*a*PM *a* = Axis number
applies to:	Analog Command Axis, Pulse Command Axis
see also:	MA, MR

This command places a servo or stepper motor in the Position Mode of operation. In this mode, it can be commanded to execute moves to specific positions. The moves will be carried out using a trapezoidal, parabolic or S-curve velocity profile. Upon start up, or after a Reset, motors will be placed in the Position Mode.

```
1PM                         ;configure axis #1 for position mode operation
1MA500                      ;move axis #1 to position 500
```

SM Set Master/Slave mode

MCCL command:	*a*SM*n* *a* = Axis number *n* = integer > 0, <= 135
applies to:	Analog Command Axis, Pulse Command Axis
see also:	SS

This command will cause axis ‘a’ to be "slaved" to a "master" axis n with a ratio specified by the Set Slave ratio command. Alternatively, this command can slave one axis to two master axes for tangential knife control in cutter applications. In this case the command parameter is determined by the following algorithm:

parameter = master 1 axis number + (master 2 axis number x 16)

Note: For tangential knife control the axis controlling the knife must be a Analog Command Axis.

Issuing the SM command with parameter *n*=0 the slave axis, will terminate the connection to the master axis.

```
2SS5                        ;slave ratio for axis 2 = 5.0:1
3SS-0.1                     ;slave ratio for axis 3 = -0.1
2SM1,3SM1                   ;slave axes 2 & 3 to axis #1 (master)

2SM0,3SM0                   ;terminate Master/Slave mode
2MN,3MN                     ;issue motor on to reinitialize position registers
```

QM torQue Mode

MCCL command: *a*QM *a* = Axis number
applies to: Analog Command Axis
see also: SQ

This command places a servo (not valid for open or closed loop steppers) in the Torque Mode of operation. This command does **not imply** that the torque generated by or current across the motor is monitored or controlled by the controller. In this mode, the analog command output will be set to the voltage level as specified with the **S**et tor**Q**ue command. The parameter to this command has default units of volts. In this mode of operation, the servo command output is 'turned into' a programmable voltage source. As such, the change of position as indicated by the encoder of an axis, while still recorded by the controller, will have no affect on the operation of the controller.

```
2SQ0                                ;set SQ=0 before entering torQue Mode
2QM                                 ;enable torQue Mode for axis 2
2SQ-3.85                            ;set axis #2 analog command output
                                    ;to -3.85 volts
```

VM Velocity Mode

MCCL command: aVM *a* = Axis number
applies to: Analog Command Axis, Pulse Command Axis
see also: DI, GO

This command places a motor in the Velocity Mode of operation. In this mode, the motor can be commanded to move in either direction at a given velocity. The motor will move in that direction until commanded to stop. In Velocity Mode the user can specify the direction for the motor to move using the DIrection (DI) command. While a motor is moving, the user can issue new direction or velocity commands. The acceleration or deceleration rate at which the motor velocity will change is determined by the Set Acceleration (SA) and Deceleration Set (DS) commands.

```
1VM,2VM                             ;axes 1 & 2 = Velocity mode
1DI0                                ;axis 1 = positive direction
2DI1                                ;axis 2 = negative direction
0GO                                 ;begin motion of axes 1 & 2

0ST                                 ;stop velocity mode motion
```

Chapter Contents

Chapter 7

Motion Commands

AB ABort motion

MCCL command: *a*AB *a* = Axis number (0 = Abort motion on all axes)
applies to: Analog Command Axis, Pulse Command Axis
see also: ST

This command serves as an emergency stop. For a servo, motion stops abruptly but leaves the position feedback loop (PID) and the amplifier enabled. For a stepper motor, the pulses will be disabled immediately. For both servos and stepper motors, the target position of the axis is set equal to the present position. This command can be issued to a specific axis, or can be issued to all axes simultaneously by using an axis specifier of 0.

comment: Issuing the Abort command to a moving servo will very likely trip the Following error and/or cause significant 'jerk' in the system.

```
2AB                          ;causes the motion of axis 2 to be aborted

AB                           ;stop motion of all axes
```

AF Auxiliary encoder Find index mark

MCCL command : *a*AF *a* = Axis number *n* = integer or real
applies to: Pulse Command Axis
see also: AH, AX

This command is used to initialize the reported position of an open loop steppers' auxiliary encoder . Upon issuing the AF command status bit 19 (Looking for Aux. Encoder Index) will be set. When the auxiliary encoder index mark is next asserted status bit 20 (Aux. Encoder Index Found) will be set and status bit 19 (Looking for Aux. Encoder Index) will be cleared. The reported position of the auxiliary encoder (at the location of the index mark) will equal the value set by parameter *n*. Status bit 20 (Aux. Encoder Index Found) will remain set until the AF command is reissued or the controller is reset.

```
MD1,5MN,5VM,5DI0,5GO,WA.1,MJ10          ;start velocity mode move
MD10,5AF0,WA0.1,LU"STATUS",5RL@0,IS20,MJ11,NO,JR-5
                                        ;Enable aux. encoder index mark
                                        ;capture, loop until Aux. Index
                                        ;Found = True
MD11,5ST,5WS.1,5PM,5MN                  ;stop the move, enable position mode
```

BC Begin Compare

MCCL command: *a*BC*n* *a* = Axis number *n* = integer or real >= 0 <=1024
applies to: Analog Command Axis, Pulse Command Axis
see also: CG, LC, NC, OC, OP

The ***MultiFlex family of controllers*** provide two (one for the four Analog command axes and one for the four Pulse Command axes) high speed TTL outputs to indicate that a position compare event has occurred. The Begin Compare command is used to start or terminate a position compare event. As many as 1024 compare positions can be stored for an axis. Begin Compare parameter *n* defines the number of position compare events. Parameter ***n*** of the **L**oad **C**ompare (*a*LC*n*) command defines the initial compare position. If Begin Compare parameter *n* is expressed as a negative number the position compare operation will be terminated.

The compare position update frequency is based on the trajectory generator, which executes every milisecond (1KHz). Therefore the distance between compare positions cannot be such that the time from one compare event to the next is less than the position update frequency. The maximum latency between the an axis reaching the compare position and the activation of the compare output is 100 nano seconds.

For compare events at fixed distances of travel use the **N**ext **C**ompare (***a*NC*n***) command. As with all compare operations the **L**oad **C**ompare command (***a*LC*n***) is used to define the first compare position. The **N**ext **C**ompare command is used to define the increment between compare events. To start fixed interval compare issue the Begin Compare command with parameter ***n* = 0**. After the motors position equals the first compare position, the parameter to the Next Compare command is added to (the parameter can be a positive or negative number) the previous compare position. At each compare position the output will be activated. To disable fixed interval position compare issue the Begin Compare command with parameter ***n* = -1**.

When the compare output is activated (latency = 100 nano seconds) as the result of a compare or breakpoint occurrence, the compare output signal will react according to the which mode has been selected with the Output mode for Compare command (OC, code=320). The parameter to this command selects one of the following modes:

Parameter	Output Mode
0*	**Disabled (default)**
1*	Static
2*	Toggle
3*	One-shot

*The active level of the output can be switched by adding a value of 128 to the OC command parameter.

In the static mode, when a breakpoint command is issued (WP, WR, IP or IR), the output will go to the in-active level and remain there until the breakpoint position is reached. When the position is reached the output will go to the active level, and remain there until a succeeding breakpoint command is issued. This output mode should not be used for the compare from memory or compare incremental functions.

Selecting the toggle output mode will cause the output to switch between in-active and active levels with each compare or breakpoint position reached. The initial state of the output can be set to in-active by selecting the Disabled mode momentarily, and then setting it back to toggle mode. Reaching the first compare or breakpoint position will cause the output to go to the active level. At the second position or breakpoint, it will switch back to the in-active level, and so on.

Selecting the one-shot output mode will cause the compare output to generate a pulse of a fixed period at each compare or breakpoint position. The period of the pulse is set by issuing the Output Period command (OP, code=321). The parameter to this command is in units of seconds. The programmable one-shot timer can generate a pulse period of from one milisecond to 2147 seconds (with 1 milisecond resolution). The output signal is guaranteed to go active within 100 nano seconds of the compare event.

To determine how many compares have occurred, either from memory or incremental, the Get Compare count command (GC, code=325) is issued to the axis. This will cause the number to be stored in the accumulator (user register 0). Once in the accumulator, it can be displayed with the Tell Register command (TR) or used for conditional testing.

```
1OC3,1OP.01                             ;one shot compare output of 10 msec's
1LC1000                                 ;first compare position = 1000
1NC250                                  ;position compare increment = 250
1BC0                                    ;enable position compare

1MA3000                                 ;move to position 3000
1WP2100                                 ;wait for position 2100
1BC-1                                   ;disable position compare
```

BF **B**acklash compensation o**F**f

MCCL command : *a*BF *a* = Axis number
applies to: Analog Command Axis
see also: BD, BN

Use this command to disable backlash compensation. As soon as this command is executed, the motor will move to its uncompensated position.

```
1BD100                           ;define backlash distance
1BN                              ;enable backlash compensation

1BF                              ;disable backlash compensation
```

BN Backlash compensation oN

MCCL command:	*a*BN *a* = Axis number
applies to:	Analog Command Axis
see also:	BD, BF

Use this command to enable backlash compensation. It should be issued after the backlash compensation distance has set been with the BD command. Prior to issuing the Backlash Compensation On command, the motor should be positioned halfway between the two positions where it makes contact with the mechanical gearing. This will allow the controller to take up the backlash (when the first move in either direction is made) without 'bumping' the mechanical position.

While backlash compensation is enabled, the response to the Tell Position, Tell Target and Tell Optimal commands will be adjusted to reflect the ideal positions (as if no mechanical backlash were present).

```
1BD100                              ;define backlash distance
1BN                                 ;enable backlash compensation

1BF                                 ;disable backlash compensation
```

CA arc Center Absolute

MCCL command:	*a*CA*n* *a* = Axis number *n* = integer or real >= 0
applies to:	Analog Command Axis, Pulse Command Axis
see also:	CM, CP

This command is used to specify the center of an arc for a Contour Path motion. Since the arc motion is performed by two axes, this command (or the arc Center Relative command) should occur twice in a Contour Path command that initiates the arc motion. The parameter to this command specifies the center of the arc for the selected axis in absolute user units.

```
1CM1                       ;Axis #1 is controlling axis
2CM1                       ;Axis #2 is a member of the contour group


1CP2,1CA10000,2CA0,1MA20000,2MA0
                            ;180 degree clockwise arc (defined by arc center
                            ;absolute coordinates)
```

CB Capture Begin

MCCL command:	*a*CB*n* *a* = Axis number *n* = integer >= 0 <=1024
applies to:	Servo, Stepper
see also:	CG, DR

This command is used to store the position of axis *a* when the capture input transitions to active. As many as 1024 captured position points can be stored for an axis . When parameter *n* is an integer between 1 and 1024 position capture will begin. The maximum frequency of position captures is 1 KHz. If parameter *n* is expressed as a negative number position capture will be terminated.

The **C**apture **B**egin (***a*CB*n***) command is used to initiate position capture. When this feature is enabled the position of the axis will be recorded on the rising edge of the capture input. If parameter *n* equals 0 or 1 only one position will be captured. If parameter *n* equals 2 two positions will be captured, and so on. When the number of positions captured = *n*, bit 5 of the axis status (Position Captured Flag) will be set. To disable position capture issue the **C**apture **B**egin command with parameter *n* expressed as a negative number. Captured positions are reported by using the **D**isplay **R**ecorded position (***a*DR*n***) command. To load the number of captured positions into the accumulator use the **C**apture **G**et count (***a*CG**) command

```
;Use the MCCL command Capture Begin to capture 10 positions

1CB10                                   ;capture 10 positions
1MR10000                                ;start moving
LU”STATUS”,1RL@0,IS5,BK,NO,JR-5
                                        ;loop until Position Capture Flag =1
```

CD Contour Distance

MCCL command : *a*CD*n* *a* = Axis number of the controlling axis *n* = integer or real >= 0
applies to: Analog Command Axis, Pulse Command Axis
see also: CM, CP

For user defined contour moves, this command is used to specify the distance as measured along the path from the contour path starting point to the end of the next motion. For typical orthogonal (X, Y, Z) geometry, the controller calculates the contour move distance ($\sqrt{}X^2+Y^2+Z^2$) based on the target positions specified by the move absolute and/or move relative commands. For applications where orthogonal geometry is not applicable, the controller allows the user to define a custom contour distance. This is accomplished by:

1) The command sequence must be preceded by the **C**ontour **P**ath (***a*CP*n***) command (***a*** = the controlling axis) with parameter ***n*** = 0.
2) **C**ontour **D**istance (***a*CD*n***) must be the last command in the compound command sequence, with parameter ***n*** = the Calculated Contour Distance of the move

The controller will use the current settings for vector velocity, vector acceleration, and vector deceleration to calculate the period of the motion.

```
        1CM1,2CM1,3CM1                          ;configure axes 1, 2, & 3 as a contour
                                                ;group with axis #1 as the controlling
                                                ;axis

        1CP0,1MA1000,2MA1000,3MA1000,1CD10000
                                                ;execute a user defined contour path with
                                                ;a contour distance of 10,000

        1CP0,1MA0,2MA0,3MA0,1CD20000            ;execute a second user defined contour
                                                ;path with a contour distance of 10,000.
                                                ;The CD command parameter n is 10,000 +
                                                ;10,000 = 20,000
```

CP define the **C**ontour **P**ath

MCCL command: *a*CP*n*   *a* = Axis number   *n* = integer 0, 1, 2, or 3
applies to: Analog Command Axis, Pulse Command Axis
see also: CM

This command is used to form a 'compound' command that specifies a multi axis motion. The compound command must begin with the Contour Path command, followed by a variable number of other motion commands. The axis number used with the Contour Path command must be the controlling axis in a group of motors that have been previously placed in Contour Mode. The parameter to the Contour Path command selects either a linear, arc or 'user defined' motion. The table below list the type of motion each parameter value specifies, and the acceptable commands that can be include in the compound command.

```
        1CP1,1GH,2GH,3GH,1VV60000
        1CP1,1MA10000,2MA20000,3MR-5000,1VV30000
        1CP1,1GH,2GH,3GH
        1CP2,1CA20000,2CA0,1MA40000,2MA0
        1CP3,1CR-20000,2CR0,1MR-40000,2MR0
```

Parameter n	Motion Type	Compatible Commands
0	User defined	CD,GH,MA,MR,VV
1	Linear	GH,MA,MR,VV
2	Clockwise arc	CA,CR,GH,MA,MR,VV
3	Counter-Clockwise arc	CA,CR,GH,MA,MR,VV

CR arc **C**enter **R**elative

MCCL command: *a*CR*n*   *a* = Axis number   *n* = integer or real >= 0
applies to: Analog Command Axis, Pulse Command Axis
see also: CM, CP

This command is used to specify the center of an arc for a Contour Path motion. Since the arc motion is performed by two axes, this command (or the arc Center Absolute command) should occur twice in a Contour Path command that initiates the arc motion. The parameter to this command specifies the center of the arc for the selected axis in user units, relative to its' target position prior to beginning the arc motion.

```
      1CP2,1CR-10000,2CR0,1MR-20000,2MR0
                                              ;180 degree clockwise arc (defined by
                                              ;arc center relative coordinates)
                                              ;starting at X=20000, Y=0 and moving
                                              ;to X=0, Y=0
```

EA arc **E**nding **A**ngle absolute

MCCL command: *a*EA*n* *a* = Axis number *n* = integer or real >= 0
applies to: Analog Command Axis, Pulse Command Axis
see also: CM, CP

This command is used to specify the ending angle (end point) of a contour arc move. The parameter *n* is expressed as an absolute angle relative to when the axes where last homed. This command would be used in conjunction with the Center Absolute, Center Relative, or aRc Radius commands.

```
      1CP2,1CA10000,2CA0,1EA0                 ;Clockwise arc motion in X & Y
```

EL home **E**dge **L**atch

MCCL command : *a*EL*n* *a* = Axis number *n* = integer or real >= 0
applies to: Pulse Command Axis (Open Loop)
see also: FE, WE

This command is used to arm the capturing of the home sensor position of an open loop stepper. When combined with Wait for Edge (***a*WE**) it performs the same operation as the Find Edge command. After the Edge Latch command is issued, when the home sensor is activated, the step count position of the home sensor will be captured and the Home found status bit (status bit 18) will be set. The Wait for Edge and Motor oN commands are then issued to define the location of the home sensor as position *n*. The difference between using the Edge Latch and Wait for Edge commands versus the Find Edge command is that Find Edge will stall the command interpreter (no additional command execution) until the home sensor is captured. The Edge Latch command arms the capture of the home sensor but does not stop additional command execution.
Note: To home (re-initialize) the reported position of an open loop stepper the controller uses a 32 bit position capture register. For this reason the Stepper Home input function cannot be reassigned to a different optically isolated input circuit or TTL digital input circuit.

```
MD5,5LM2,5LN3,MJ10                      ;enable limits, call homing macro
MD10,5VM,5DI0,5SV10000,5GO,LU"STATUS",5RL@0,IS24,MJ11,NO,IS10,MJ13,NO,JR-8
                                        ;test for sensors (home and +limit)
MD11,LU"STATUS",5RL@0,IC24,MJ12,NO,JR-5 ;continue moving until home sensor
                                        ;is off
MD12,5ST,5WS.1,5DI1,5SV5000,5GO,MJ14    ;move back to the home sensor
MD13,5WS0.01,5MN,5DI1,5SV5000,5GO,MJ14  ;move out of limit sensor range
                                        ;back toward the home sensor
MD14,5EL0,MC15,5WE,5ST,5WS.1,5MF,5MN,5PM,5MA-100
                                        ;capture the active edge of the
                                        ;home sensor. Stop axis and
                                        ;define a position 0, ;move to
                                        ;position -100
MD15,LU"STATUS",5RL@0,IS18,BK,NO,JR-5   ;loop status for Edge found bit set

5MN                                     ;enable axis 5
```

ER arc **E**nding angle **R**elative

MCCL command: *a*ER*n* *a* = Axis number *n* = integer or real >= 0
applies to: Analog Command Axis, Pulse Command Axis
see also: CM, CP

This command is used to specify the ending angle (end point) of a contour arc move. The parameter *n* is expressed as an angle relative to when the axes where last homed. This command would be used in conjunction with the Center Absolute, Center Relative, or aRc Radius commands.

```
1CP2,1CR-10000,2CR0,1ER180              ;Clockwise arc motion in X & Y
```

FE **F**ind **E**dge

MCCL command: *a*FE*n* *a* = Axis number *n* = integer or real
applies to: Pulse Command Axis (Open Loop)
see also: DH, EL, WE

This command is used to reference (home) the reported position of an open loop stepper to the home sensor. This is a conditional command and will not complete execution until the activated home sensor has been captured. The FE command performs two operations, capturing of the home sensor event, and storing the step count position of the home sensor in preparation for redefining the reported position of the sensor. Upon completion of this command (and after stopping the motor), issuing the Motor oN command will cause the step count position of the sensor to be redefined to position *n.* The current position will be redefined to *n* plus or minus the distance in step counts from the sensor.

If for any reason the activation of the home sensor is not captured the controller will stall and no additional commands will be executed. In this case to clear the FE and unlock the command interpreter enter the escape key. To home an open loop servo without the possibility of locking up the controller use the Edge Latch (aELn) command to enable capturing of the sensor location and the Wait for Edge (aWE) command to store the step count position of the sensor in preparation for redefining the reported position of the sensor. To avoid stalling the command interpreter it is recommended that the FE command only be issued from a macro running as a background task.

Note: The FE command does not start or stop motion, it is up to the user to initiate motion prior to issuing the find edge command.
Note: To home (re-initialize) the reported position of an open loop stepper the controller uses a 32 bit position capture register. For this reason the Stepper Home input function cannot be reassigned to a different optically isolated input circuit or TTL digital input circuit.

```
; MCCL Stepper linear stage homing sequence using Home & positive limit
;sensors
MD5,5LM2,5LN3,MJ10                                ;enable limits, call homing macro
MD10,5VM,5DI0,5SV10000,5GO,LU"STATUS",5RL@0,IS24,MJ11,NO,IS10,MJ13,NO,JR-8
                                                  ;test for sensors (home and +limit)
MD11,LU"STATUS",5RL@0,IC24,MJ12,NO,JR-5           ;continue moving until home sensor
                                                 ;is off
MD12,5ST,5WS.1,5DI1,5SV5000,5GO,MJ14              ;move back to the home sensor
MD13,5WS0.01,5MN,5DI1,5SV5000,5GO,MJ14            ;move out of limit sensor range
                                                  ;back toward the home sensor
MD14,5FE0,5ST,5WS.1,5MF,5MN,5PM,5MA-100
                                                   ;capture the active edge of the
                                                   ;home sensor. Stop axis and
                                                   ;define a position 0, ;move to
                                                   ;position -100
MD15,LU"STATUS",5RL@0,IS18,BK,NO,JR-5             ;loop status for Edge found bit set

5MN                                    ;enable axis 5
```

FI Find Index

MCCL command: a FI*n*   *a* = Axis number   *n* = integer or real
applies to: Analog Command Axis, Pulse Command Axis (Closed Loop)
see also: DH, IA, WI

This command is used to reference (home) the reported position of a closed loop system to the index mark of an encoder. This is a conditional command and will not complete execution until the index mark of the encoder has been captured. The FI command performs two operations, capturing of the encoder index event, and storing the encoder position of the index in preparation for redefing the reported position of the encoder. Upon completion of this command (and after stopping the motor), issuing the Motor oN command will cause the encoder position of the index to be redefined to position *n.* The current position will be redefined to *n* plus or minus the distance in encoder counts from the index mark.

If for any reason the encoder index mark is not captured the controller will stall and no additional commands will be executed. In this case to clear the FI and unlock the command interpreter enter the escape key. To home a closed loop system without the possibility of locking up the controller use the Index Arm (aIAn) command to enable capturing of the index mark location and the Wait for Index (aWI) command to store the encoder position of the index in preparation for redefing the reported position of the encoder. To avoid stalling the command interpreter it is recommended that the FI command only be issued from a macro running as a background task. Note: The FI command does not start or stop motion, it is up to the user to initiate motion prior to issuing the find index command.

Since an index pulse may occur at numerous points of a servo's travel (once per revolution of a rotary encoder), typical closed loop systems will require a coarse home signal to "qualify" the index pulse. The encoder and / or coarse home sensor would then be adjusted so that the index pulse occurs while the sensor is active. See the description of **Homing Axes** in the **Motion Control** chapter.

```
MD1,1LM2,1LN3,1VM,1DI0,1GO,MJ10
                                  ;begin positive velocity mode move
MD10,LU"STATUS",1RL@0,IS25,MJ11,NO,IS10,MJ15,NO,JR-8
                                  ;test for sensors (home and +limit)
MD11,1ST,1WS.01,1DI1,1GO,MC21,1ST,1WS.01,MJ12
                                  ;move out of coarse home
MD12,MC22,1DI0,1GO,MC23,MJ13      ;move back into coarse home sensor
MD13,1FI1000,1ST,1WS.01,MJ14      ;capture index (position = 1000) then
                                  ;stop
MD14,1PM,1MN,1MA1000,1WS.1        ;initialize axis, move to index
MD15,1WS.1,1MN,1DI1,1GO,MC23,MC21,1ST,1WS.01,MJ12
                                  ;limit + true, move negative until coarse
                                  ;home true

;homing sub routines
MD21,LU"STATUS",1RL@0,IC25,BK,NO,JR-5   ;test for coarse home false
MD22,1SV10000,1SA100000,1DS100000       ;reduce trajectory parameters
MD23,LU"STATUS",1RL@0,IS25,BK,NO,JR-5   ;test for coarse home true

1MN                                     ;enable axis 1
```

## GH		**G**o **H**ome

MCCL command: *a*GH *a* = Axis number
applies to: Analog Command Axis, Pulse Command Axis
see also: MA, MC, MD

Causes the specified axis or axes to move to the offset position that was specified when the last DH, IA & WI, EL & WE, FI, or FE command was issued. This is equivalent to a Move Absolute command, where the destination is 0 or the offset of the home position.

## GO		GO

MCCL command: *a*GO*n* *a* = Axis number *n* = integer 0 or 1
applies to: Analog Command Axis, Pulse Command Axis
see also: CM, SN< VM

Causes one or all axes to begin motion in velocity or contour mode. In contour mode, synchronization must be on. The parameter to this command is only used for contour mode, and determines whether the motions will be linearly interpolated (n = 0), or a cubic spline (n = 1).

```
1VM,1DI0,1GO                            ;begin axis #1 velocity mode move
```

HO **HO**me

MCCL command: *a*HO *a* = Axis number
applies to: Analog Command Axis, Pulse Command Axis
see also: EL, FE, FI, IA, MC, MD, WE, WI

This command will cause a user defined macro to be executed. It is up to the user to define the macro to carry out the appropriate homing sequence for that motor. Issuing 1HO will cause macro 1 to be executed, issuing 2HO will cause macro 2 to be executed, and so on. Issuing this command with no motor specified will cause macro 9 to be executed.

```
1HO                                   ;home axis #1 by executing macro #1
2HO                                   ;home axis #2 by executing macro #2
```

IA **I**ndex **A**rm

MCCL command : *a*IA*n* *a* = Axis number *n* = integer or real
applies to: Analog Command Axis, Pulse Command Axis (Closed Loop)
see also: FI, WI

This command is used to arm the index capture function of a closed loop axis. When combined with Wait for Index (***a*WI**) it performs the same operation as the Find Index command. After the Index Arm command is issued, when the index pulse occurs, the encoder position of the index mark will be captured and the Encoder Index status bit (status bit 18) will be set. The Wait for Index and Motor oN commands are then issued to define the location of the index pulse as position *n*. The difference between using the Index Arm and Wait for Index commands versus the Find Index command is that Find Index will stall the command interpreter (no additional command execution) until the index is captured. The Index Arm command arms the capture of the index but does not stop additional command execution.

```
MD1,1SV50000,1SA100000,1DS100000,1LM2,1LN3,1VM,1DI0,1GO,MJ10
                                 ;begin positive velocity mode move

MD10,LU"STATUS",1RL@0,IS25,MJ11,NO,IS10,MJ15,NO,JR-8
                                 ;test for sensors (home and +limit)

MD11,1ST,1WS.01,1DI1,1GO,MC21,1ST,1WS.01,MJ12
                                 ;move out of coarse home
MD12,MC22,1DI0,1GO,MC23,MJ13      ;move back into coarse home sensor
MD13,1IA1000,MC24,1WI,1ST,1WS.01,MJ14
                                 ;capture index (position = 1000) then stop
MD14,1PM,1MN,1MA1000,1WS.1        ;initialize axis, move to index
MD15,1WS.1,1MN,1DI1,1GO,MC23,MC21,1ST,1WS.01,MJ12
                                 ;limit + true, move negative until coarse
                                 ;home true
```

```
;homing sub routines
MD21,LU"STATUS",1RL@0,IC25,BK,NO,JR-5   ;test for coarse home false
MD22,1SV10000,1SA100000,1DS100000       ;reduce trajectory parameters
MD23,LU"STATUS",1RL@0,IS25,BK,NO,JR-5   ;test for coarse home true
MD24,LU"STATUS",1RL@0,IS24,BK,NO,JR-5   ;test for Index Found

1MN                                     ;enable axis 1
```

LC **L**oad **C**ompare

MCCL command: *a*LC*n* *a* = Axis number *n* = integer or real
applies to: Analog Command Axis, Pulse Command Axis
see also: CG, LC, NC, OC, OP

This command is used to store the compare positions. As many as 1024 position points can be stored for each axis. See the description of the **B**egin **C**ompare (aBCn) in this chapter for a complete description of position compare operations.

LP **L**earn **P**osition

MCCL command: *a*LP*n* *a* = Axis number *n* = integer >= 0, <=255
applies to: Analog Command Axis, Pulse Command Axis
see also: LT, MP

Used for storing the current position of one or more axes in the controller's point memory. Positions stored in the point memory can be used by the Move to Point command to repeat a stored motion pattern. The command parameter n specifies the entry in the point memory where the position will be stored.

If the LP command is issued with an axis specifier of 0, the positions of all axes on the controller board will be stored in the point memory. If the command is issued with a non-zero axis specifier, only the position of that axis will be stored in the point memory. No other positions in the point memory will be changed.

```
1DH0,1MN,2DH0,2MN          ;define current position as 0
1MR250,2MR-100,0WS0.02     ;move and wait
1LP1,2LP1                  ;store current position as learned point #1
1MR500,2MR300,0WS0.01      ;move and wait
1LP2,2LP2                  ;store current position as learned point #2


1MA0,2MA0,0ws.25
0MP1,0WS.25                ;move to point #1, wait for 0.25 seconds
0MP2,0WS.25                ;move to point #1, wait for 0.25 seconds
```

LT **L**earn **T**arget

MCCL command: *a*LT*n* *a* = Axis number *n* = integer >= 0, <=255
applies to: Analog Command Axis, Pulse Command Axis
see also: LP, MP

Similar to the LP command, but stores the axes' target position (versus actual position). Motion of an axis is not required for storing target positions. This makes it possible to download coordinates from a host computer or CAD system.

Turn off the motor drive outputs with the MF command, then send motion commands prior to the LT command. Targets stored in the point memory can be used by the Move to Point command to repeat a stored motion pattern. The command parameter n specifies the entry in the point memory where the position will be stored. If the LT command is issued with an axis specifier of 0, the targets of all axes on the controller board will be stored in the point memory. If the command is issued with a non-zero axis specifier, only the target of that axis will be stored in the point memory. No other targets in the point memory will be changed.

```
1DH0,1MF,2DH0,2MF       ;define current position as 0
1MR250,2MR-100          ;issue move
1LP1,2LP1               ;store current position as learned point #1
1MR500,2MR300           ;issue move
1LP2,2LP2                  ;store current position as learned point #2


1MN,2MN
1MA0,2MA0,0ws.25
0MP1,0WS.25                ;move to point #1, wait for 0.25 seconds
0MP2,0WS.25                ;move to point #1, wait for 0.25 seconds
```

MA **M**ove **A**bsolute

MCCL command : *a*MA*n* *a* = Axis number *n* = integer or real
applies to: Analog Command Axis, Pulse Command Axis
see also: MR, PM

This command generates a motion to absolute position *n*. A motor number must be specified and that motor must be in the ‘on’ state for any motion to occur. If the motor is in the off state, only its' internal target position will be changed.

```
1MA1000                                 ;move to position 1000
2MA-25000                               ;move to position -25000
```

MF Motor oFf

MCCL command :	*a*MF *a* = Axis number
applies to:	Analog Command Axis, Pulse Command Axis
see also:	MN

Issuing this command will place one or all servos and stepper motors in the "off" state. For servos, the Analog Signal will go to the null level, the servo loop (PID) will terminate, and the Amplifier Enable output will go inactive. For stepper motors, the Driver Enable output will go inactive. This command can be used to prevent unwanted motion or to allow manual positioning of the servo or stepper motor.

```
      1MF                                      ;turn off axis #1
```

MN Motor oN

MCCL command:	aMN *a* = Axis number
applies to:	Analog Command Axis, Pulse Command Axis
see also:	MF

Use this command to place one or all servos and stepper motors in the on state. If an axis is off when this command is issued, the target and optimal (commanded) positions will be set to the motor's current position. This can cause a change in the axis' reported position based on new user units. At the same time, a servo Amplifier Enable or a stepper motor Drive Enable output signal will go active. This has the effect of causing servo and stepper motors to hold their current position. If an axis is already on when this command is issued, the position values will be set for the current user units, but the commanded encoder or pulse position will not be changed.

```
      1MN                                      ;turn on axis #1
```

MP Move to Point

MCCL command:	*a*MP*n* *a* = Axis number *n* = integer >= 0, <=255
applies to:	Analog Command Axis, Pulse Command Axis
see also:	LP, LT

Used for moving one or more axes to a previously stored point. The command parameter n specifies which entry in the controller's point memory is to be used as the destination of the move. If the MP command is issued with an axis specifier of 0, all axes will move to the positions stored in the point memory for that point. If the command is issued with a non-zero axis specifier, only that axis will move to the position in the point memory. No other axes will be commanded to move. Points can be stored in the point memory with the Learn Point (LP) and Learn Target LT) commands.

```
        1DH0,2DH0,1MN,2MN           ;define current position as 0
        1MR250,2MR-100,0WS0.02      ;move and wait
        1LP1,2LP1                   ;store current position as learned point #1
        1MR500,1WS0.01              ;move and wait
        1LP2,2LP2                   ;store current position as learned point #2


        1MN,2MN
        1MA0,2MA0,0ws.25
        0MP1,0WS.25                 ;move to point #1, wait for 0.25 seconds
        0MP2,0WS.25                 ;move to point #1, wait for 0.25 seconds
```

MR **M**ove **R**elative

MCCL command :	a*MR*n *a* = Axis number *n* = integer or real
applies to:	Analog Command Axis, Pulse Command Axis
see also:	MA, PM

This command generates a motion of relative distance *n*. A motor number must be specified and that motor must be in the ‘on’ state for any motion to occur. If the motor is in the off state, only its' internal target position will be changed.

```
        1MR1000                                  ;move 1000 counts / steps
        2MR-25000                                ;move -25000 counts / steps
```

NC **N**ext **C**ompare position

MCCL command:	*a*NC*n* *a* = Axis number *n* = integer or real
applies to:	Analog Command Axis, Pulse Command Axis
see also:	BG, CG, DR, HS, LS, MS

This command is used to define a fixed increment distance that will be used for position compare. See the description of the **B**egin **C**ompare (aBCn) in this chapter for a complete description of position compare operations.

NS **N**o **S**ynchronization

MCCL command:	*a*NS *a* = Axis number
applies to:	Analog Command Axis, Pulse Command Axis
see also:	SN

This command turns synchronization off in contour path motions. It should be issued to the controlling axis of the contour group.

OC **O**utput mode for **C**ompare

MCCL command: *a*OC*n* *a* = Axis number *n* = integer 0, 1, 2, 3
applies to: Analog Command Axis, Pulse Command Axis
see also: BG, NC, OP

This command is used to define the mode of operation for the Position Compare output. The possible values for parameter n are:

Parameter	Output Mode
0	**Disabled (default)**
1	Static
2	Toggle
3	One-shot

See the description of the **B**egin **C**ompare (aBCn) in this chapter for a complete description of position compare operations.

OP **O**utput compare **P**eriod

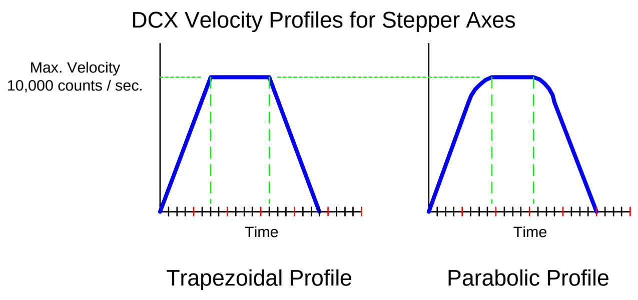
MCCL command: *a*OP*n* *a* = Axis number *n* = integer or real > 0
applies to: Analog Command Axis, Pulse Command Axis
see also: BG, NC,OC

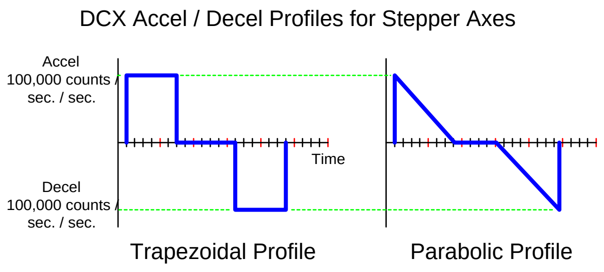
This command is used to define the period of the Position Compare output when configured for one-shot mode. The parameter to this command is in units of seconds. A one-shot timer is used to generate a pulse period of from 1 microsecond to 1 second. For time periods less then 50 milliseconds the timer has 1 microsecond resolution. For time periods greater than or equal to 50 milliseconds, it has 50 microsecond resolution. With either short or long duration pulses, the output signal is guaranteed to go active within the 1/2 microsecond latency of the compare function (not counting the delay for the optical isolator). See the description of the **B**egin **C**ompare (aBCn) in this chapter for a complete description of position compare operations.

PP **P**arabolic **P**rofile

MCCL command: *a*PP *a* = Axis number
applies to: Analog Command Axis, Pulse Command Axis
see also: PS, PT

This command causes the respective servo or stepper motor to perform ***point to point*** motions with a triangular acceleration profile. The resulting velocity profile is parabolic. Motion with this profile is limited to ***position and contour*** mode moves, where the acceleration, deceleration, velocity, and destination ***don't change during the move***.





```
1PT               ;Axis #1 moves using trapezoidal profile
2PP               ;Axis #2 moves using parabolic profile
```

PR Record axis data

MCCL command : *a*PR*n* *a* = Axis number *n* = integer >= 0, <=1024
applies to: Analog Command Axis, Pulse Command Axis
see also: DR,DO, DQ

This command is used to begin the recording of motion data (actual position, optimal position, DAC output, aux. encoder position) for an axis.

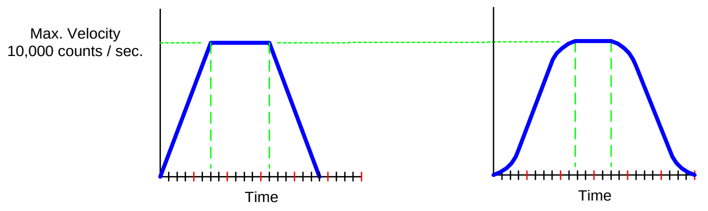
```
3DH0,3MN
3PR10,3MA5000,3WS.1

AL0,AR100                       ;define register #100 as recorded position
                                ;pointer

3DR@100,AL@100,AA1,AR100,RP9    ;display the first 10 recorded positions
```

PS Profile S-curve

MCCL command: *a*PS *a* = Axis number
applies to: Analog Command Axis, Pulse Command Axis
see also: PP, PT

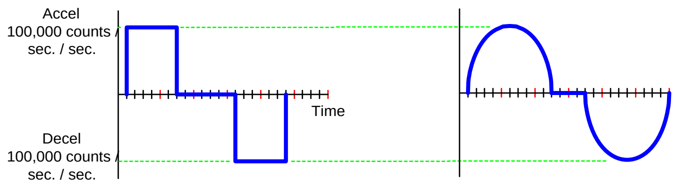
This command causes the respective servo or stepper motor to perform ***point to point*** motions with a sinusoidal acceleration profile. The resulting velocity profile is trapezoidal with rounded corners, thus the name S-curve. Motion with this profile is limited to ***position and contour*** mode moves, where the acceleration, deceleration, velocity, and destination ***don't change during the move***.

DCX Velocity Profiles for Servo Axes



Trapezoidal Profile S curve Profile

DCX Accel / Decel Profiles for Servo Axes



Trapezoidal Profile S curve Profile

```
1PT             ;Axis #1 moves using trapezoidal profile
2PS             ;Axis #2 moves using S-curve profile
```

PT Profile Trapezoidal

MCCL command:	*a*PT *a* = Axis number
applies to:	Analog Command Axis, Pulse Command Axis
see also:	PS, PT

This command causes the respective servo or stepper motor to perform point to point motions with a constant acceleration profile. The resulting velocity profile is trapezoidal. When motion is being performed with this profile, the acceleration, velocity, and destination can be changed at any time during the move.

RR Radius of aRc

MCCL command:	*a*RR*n* *a* = Axis number *n* = integer or real >= 0
applies to:	Analog Command Axis, Pulse Command Axis
see also:	CM, CP

This command specifies the radius of a contour mode arc. For an arc of less than 180 degrees the parameter n should be a positive value equal to the radius of the arc. For an arc of greater than 180 degrees the parameter n should be a negative value equal to the radius of the arc.

```
        1CM1,2CM1                               ;define axis 1 as controlling axis
        1CP2,1MR10000,2MR10000,1RR10000         ;90°  clockwise arc, radius = 10000
        1CP2,1MR-10000,2MR-10000,1RR-10000      ;270° degree arc, radius = 10000,
                                                ;negative radius parameter indicates
                                                ;arc greater than 180°
```

SN Synchronization oN

MCCL command:	aSN *a* = Axis number
applies to:	Analog Command Axis, Pulse Command Axis
see also:	GO, CP, NS

This command allows the contour buffer to be pre-loaded prior to beginning contour motion. This capability is required for cubic spline interpolation of a contour path motion. The SN command should be issued to the controlling axis prior to Contour Path commands. With synchronization on, no motion will occur when a Contour Path command is issued, until the GO command is issued to the controlling axis. Motion will continue from the first to the last point in the contour buffer. To return to normal operation, issue the **N**o **S**ynchronization (***a*NS**) command with ***a*** = the controlling axis.

comment: Note that when performing cubic spline interpolation, only **128 motions** can be queued up in the contour buffer.

ST STop

MCCL command:	*a*ST *a* = Axis number (0 = Stop motion on all axes)
applies to:	Analog Command Axis, Pulse Command Axis
see also:	AB, MF

This command is used to stop one or all motors. It differs from the Abort command in that motors will decelerate at their preset rate, instead of stopping abruptly. This command can be issued to a specific axis, or can be issued to all axes simultaneously by using an axis specifier of 0.

```
        1ST,1WS.1                   ;stop axis #1, wait 0.1 seconds after calculated
                                     ;trajectory has completed


        0ST,0WS.1                   ;Stop all axes, wait 0.1 seconds after calculated
                                     ;trajectory has completed
```

Chapter Contents

Reporting Commands

MCCL	Description
AT	Auxiliary encoder Tell position
AZ	Auxiliary encoder tell index
CG	Capture Get count
DA	Display recorded aux. encoder position
DE	Display mEmory
DO	Display recorded optimal position
DQ	Display recorded DAC output
DR	Display recorded actual position
GC	Get the position compare count
TA	Tell Analog to digital converter
ID	Information Display
TB	Tell Breakpoint position
TC	Tell Channel
TD	Tell Derivative gain
TE	Tell command interface Error
TF	Tell Following error
TG	Tell proportional Gain
TI	Tell Integral gain
TK	Tell velocity gain
TL	Tell integration Limit
TM	Tell stored Macros
TO	Tell Optimal
TP	Tell Position
TQ	Tell torQue (aSQn)
TR	Tell Register n
TS	Tell Status
TT	Tell Target
TV	Tell Velocity
TX	Tell contouring count
TZ	Tell index position
VE	tell VErsion

Chapter 8

Reporting Commands

The commands in this section are used to display the current values of internal controller data. Some of these values are 'real' numbers that must be displayed with fractional parts. To allow the user flexibility in defining the displayed value of a controller response certain reporting commands accept a parameter that sets the number of digits displayed to the right of the decimal point. These commands will show a 'p' as a parameter in their descriptions.

For ASCII command interfaces, *p* can be replaced with a number between 0 and 1 and the tenths digit will be interpreted as the number of decimal digits to display to the right of the decimal point. If no parameter is used with the command, or a parameter of 0 is used, the reply to the command will be an integer with no decimal point. Example:

```
  ;If axis 1 position is 123.4567
    1TP;        controller replies 123
    1TP0;       controller replies 123
    1TP.1;      controller replies 123.4
    1TP.3;      controller replies 123.456
```

For the Binary command interface, the reporting commands that have a 'p' listed as their parameter will accept an integer value of 0, 1 or 2 in place of *p*. A value of 0 will generate an integer reply, a value of 1 will generate a 64 bit floating point reply, and a value of 2 will generate a 32 bit floating point reply. See the appendix describing the controller’s Binary Command Interface for more details on these reply formats.

AT Auxiliary encoder Tell position

MCCL command: *a*AT a = Axis number
applies to: Pulse Command Axis
see also: AF, AH

Reports the absolute position of the auxiliary encoder of an axis. To read the primary encoder or stepper position, see the Tell Position command.

```
1AT.3                  ;report aux. encoder position with 3 digit precision
   01   9.762          ;controller reply
```

AZ Auxiliary encoder tell index

MCCL command: *a*AZ a = Axis number
applies to: Pulse Command Axis
see also: AF, AH

Reports the position where the auxiliary encoder's index pulse was observed. This position is relative to the encoder's position when the controller was reset or an Auxiliary encoder define Home command was issued to the axis.

```
2AZ.1                  ;report aux. index position with 1 digit precision
   02   21.7           ;controller reply
```

CG Capture Get count

MCCL command: *a*CG = Axis number
applies to: Analog Command Axis, Pulse Command Axis
see also: CB, TR

Loads the accumulator with the number of positions that have been captured since the Capture Begin command was last issued with a positive parameter n. The Tell Register command is then used to report the value.

```
1CG,TR0                ;report position capture count
   01   25             ;controller reply
```

DA Display the recorded Auxiliary encoder position

MCCL command: aDAp *a* = Axis number *p* = integer >= 0, < 1024
applies to: Pulse Command Axis
see also: DO, DR, DQ, PR

This command is used to report the captured auxiliary encoder position of an axis. Typically this command is used for tuning a closed loop stepper axis or measuring the performance of a pulse command servo axis.

```
2DH0,2MN
2MA5000,2PR10,2WS.1

AL0,AR100                        ;define register #100 as recorded position
                                 ;pointer

2DA@100,AL@100,AA1,AR100,RP9     ;display the first 10 recorded aux. encoder
                                  ;positions
```

DE Display mEmory

MCCL command: DAn *n* = integer >= 0, < 65536
applies to: Other
see also: ME

Similar to a 'debug memory dump' this command will allow the user to display 256 bytes of the controller’s internal memory.

DO Display the recorded Optimal position

MCCL command: aDOp *a* = Axis number *p* = integer >= 0, < 1024
applies to: Analog Command Axis, Pulse Command Axis
see also: DR, DQ, PR

This command is used to report the captured optimal position of an axis.

```
2DH0,2MN
2MA5000,2PR10,2WS.1

AL0,AR100                        ;define register #100 as recorded position
                                 ;pointer

2DO@100,AL@100,AA1,AR100,RP9     ;display the first 10 recorded optimal
                                  ;positions
```

DQ — **D**isplay the recorded DAC output

MCCL command: aDQp *a* = Axis number *p* = integer >= 0, < 1024
applies to: Analog Command Axis
see also: DO, DR, PR

This command is used to report the captured DAC output of an axis.

```
2DH0,2MN
2MA5000,2PR10,2WS.1

AL0,AR100                        ;define register #100 as recorded position
                                 ;pointer

2DQ@100,AL@100,AA1,AR100,RP9     ;display the first 10 recorded DAC output
                                  ;levels
```

DR — **D**isplay **R**ecorded position

MCCL command: aDRp *a* = Axis number *p* = integer >= 0, < 1024
applies to: Analog Command Axis, Pulse Command Axis
see also: DO, DQ, PR

This command is used to report the captured actual position of an axis.

```
2DH0,2MN
2MA5000,2PR10,2WS.1

AL0,AR100                        ;define register #100 as recorded position
                                 ;pointer

2DO@100,AL@100,AA1,AR100,RP9     ;display the first 10 recorded optimal
                                  ;positions
```

GC — **G**et the **C**ompare count

MCCL command : *aGC* *a* = Axis number
applies to: Analog Command Axis, Pulse Command Axis
see also: BG, DR, HS, LS, MS

Loads the number of executed position compare events into the accumulator (user register 0). The Tell Register command is then used to report the value. For a detailed description of position compare please refer to the Begin Compare command.

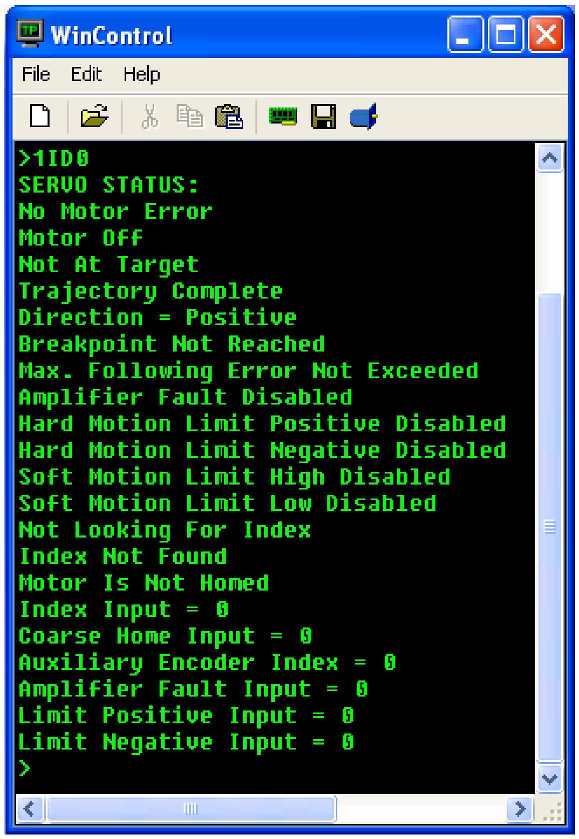
```
1GC,TR0               ;report position capture count
   01   25            ;controller reply
```

ID Information Display

MCCL command: IDn n = integer >= 0, <= 6
applies to: Analog Command Axis, Pulse Command Axis
see also: TS

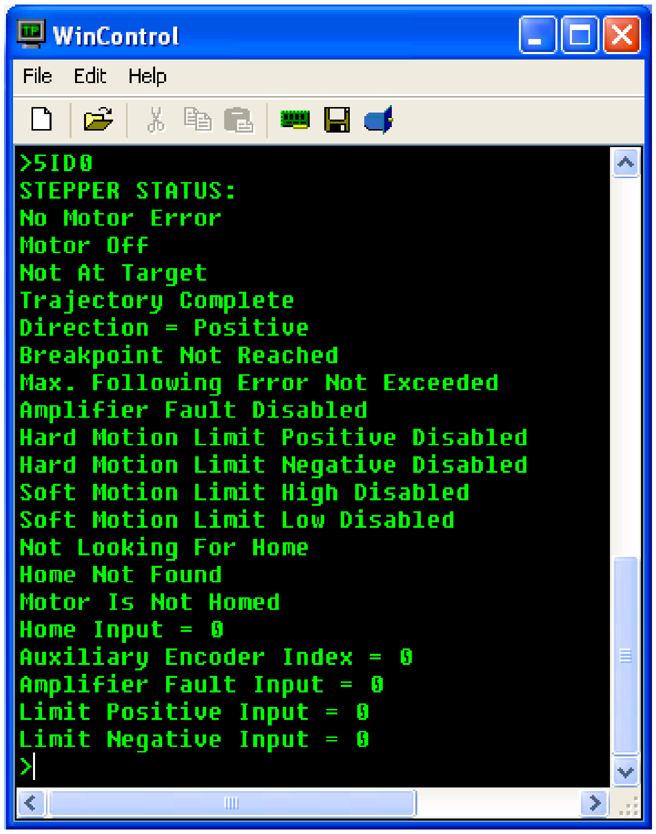
This command is used to display formatted controller settings and status.

Parameter n	Description
0	Display Primary Status information
1	Display Auxiliary Status information
2	Display Motor Table information
3	Display Internal Motor Table information
4	Display Filter Gains, Command output
5	Display Encoder Fault inputs and Enable bits
6	Display Axis I/O assignments
7	Active background task ID's



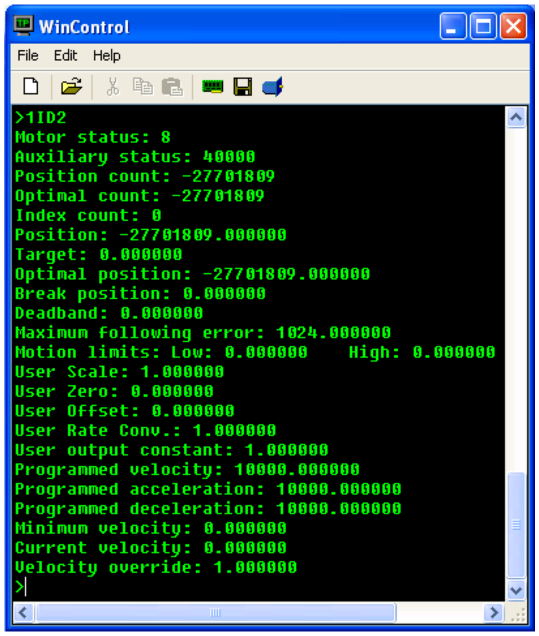
```
>1ID0
SERVO STATUS:
No Motor Error
Motor Off
Not At Target
Trajectory Complete
Direction = Positive
Breakpoint Not Reached
Max. Following Error Not Exceeded
Amplifier Fault Disabled
Hard Motion Limit Positive Disabled
Hard Motion Limit Negative Disabled
Soft Motion Limit High Disabled
Soft Motion Limit Low Disabled
Not Looking For Index
Index Not Found
Motor Is Not Homed
Index Input = 0
Coarse Home Input = 0
Auxiliary Encoder Index = 0
Amplifier Fault Input = 0
Limit Positive Input = 0
Limit Negative Input = 0
>
```

Servo axis primary status info



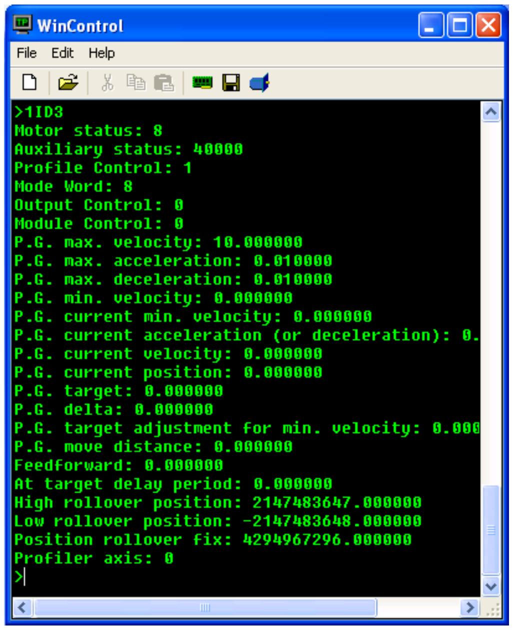
```
>5ID0
STEPPER STATUS:
No Motor Error
Motor Off
Not At Target
Trajectory Complete
Direction = Positive
Breakpoint Not Reached
Max. Following Error Not Exceeded
Amplifier Fault Disabled
Hard Motion Limit Positive Disabled
Hard Motion Limit Negative Disabled
Soft Motion Limit High Disabled
Soft Motion Limit Low Disabled
Not Looking For Home
Home Not Found
Motor Is Not Homed
Home Input = 0
Auxiliary Encoder Index = 0
Amplifier Fault Input = 0
Limit Positive Input = 0
Limit Negative Input = 0
>
```

Stepper axis primary status info



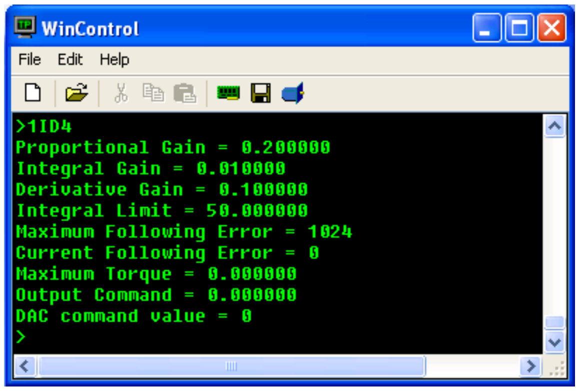
```
>1ID2
Motor status: 8
Auxiliary status: 40000
Position count: -27701809
Optimal count: -27701809
Index count: 0
Position: -27701809.000000
Target: 0.000000
Optimal position: -27701809.000000
Break position: 0.000000
Deadband: 0.000000
Maximum following error: 1024.000000
Motion limits: Low: 0.000000    High: 0.000000
User Scale: 1.000000
User Zero: 0.000000
User Offset: 0.000000
User Rate Conv.: 1.000000
User output constant: 1.000000
Programmed velocity: 10000.000000
Programmed acceleration: 10000.000000
Programmed deceleration: 10000.000000
Minimum velocity: 0.000000
Current velocity: 0.000000
Velocity override: 1.000000
>
```

Motor Table info



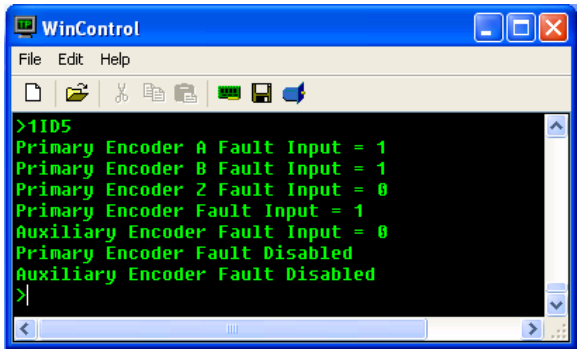
```
>1ID3
Motor status: 8
Auxiliary status: 40000
Profile Control: 1
Mode Word: 8
Output Control: 0
Module Control: 0
P.G. max. velocity: 10.000000
P.G. max. acceleration: 0.010000
P.G. max. deceleration: 0.010000
P.G. min. velocity: 0.000000
P.G. current min. velocity: 0.000000
P.G. current acceleration (or deceleration): 0.
P.G. current velocity: 0.000000
P.G. current position: 0.000000
P.G. target: 0.000000
P.G. delta: 0.000000
P.G. target adjustment for min. velocity: 0.000
P.G. move distance: 0.000000
Feedforward: 0.000000
At target delay period: 0.000000
High rollover position: 2147483647.000000
Low rollover position: -2147483648.000000
Position rollover fix: 4294967296.000000
Profiler axis: 0
>
```

Internal Motor Table info



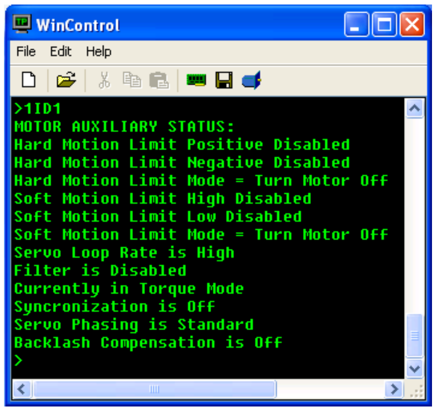
```
>1ID4
Proportional Gain = 0.200000
Integral Gain = 0.010000
Derivative Gain = 0.100000
Integral Limit = 50.000000
Maximum Following Error = 1024
Current Following Error = 0
Maximum Torque = 0.000000
Output Command = 0.000000
DAC command value = 0
>
```

Filter Gains and Command output



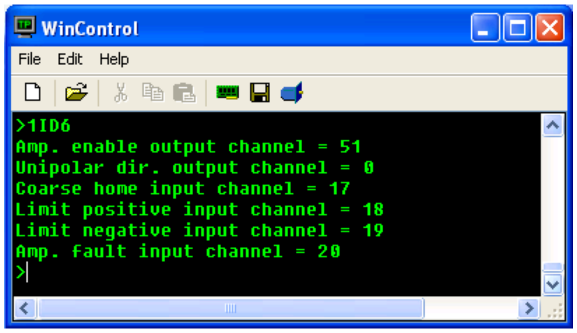
```
>1ID5
Primary Encoder A Fault Input = 1
Primary Encoder B Fault Input = 1
Primary Encoder Z Fault Input = 0
Primary Encoder Fault Input = 1
Auxiliary Encoder Fault Input = 0
Primary Encoder Fault Disabled
Auxiliary Encoder Fault Disabled
>
```

Encoder Fault info



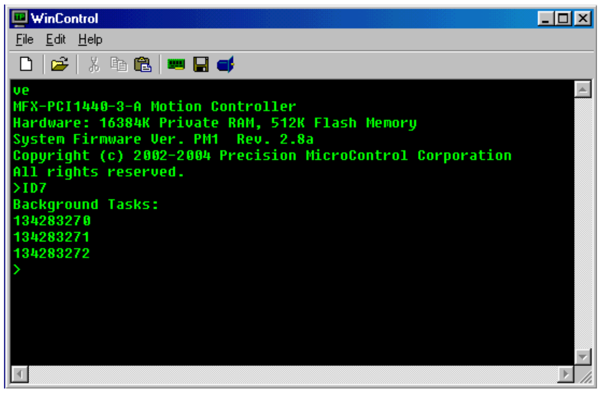
```
>1ID1
MOTOR AUXILIARY STATUS:
Hard Motion Limit Positive Disabled
Hard Motion Limit Negative Disabled
Hard Motion Limit Mode = Turn Motor Off
Soft Motion Limit High Disabled
Soft Motion Limit Low Disabled
Soft Motion Limit Mode = Turn Motor Off
Servo Loop Rate is High
Filter is Disabled
Currently in Torque Mode
Syncronization is Off
Servo Phasing is Standard
Backlash Compensation is Off
>
```

Servo axis primary status info



```
>1ID6
Amp. enable output channel = 51
Unipolar dir. output channel = 0
Coarse home input channel = 17
Limit positive input channel = 18
Limit negative input channel = 19
Amp. fault input channel = 20
>
```

Axis I/O assignments



```
ve
MFX-PCI1440-3-A Motion Controller
Hardware: 16384K Private RAM, 512K Flash Memory
System Firmware Ver. PM1  Rev. 2.8a
Copyright (c) 2002-2004 Precision MicroControl Corporation
All rights reserved.
>ID7
Background Tasks:
134283270
134283271
134283272
>
```

Active background task ID info

TA Tell Analog

MCCL command: TAx *x* = Channel number *x* = 1, 2, 3, 4, 5, 6, 7, 8
applies to: A/D inputs
see also:

Reports the digitized analog input values for a specific analog input channel. For each analog input channel, the TA command will display a number between 0 and 65536, corresponding to the entire input voltage range. For example, if the input voltage range is -10V to +10V; then -10.0V=0, 0.0V=32768 and +10.0V=65536. If the input voltage range is 0.0 to +4.0V; then 0.0V=0, 2.0V=32768 and 4.0V=65536.

TB Tell Breakpoint

MCCL command: aTBp a = Axis number *p* = 0, .1, .2, .3, .4, .5
applies to: Analog Command Axis, Pulse Command Axis
see also: IP, IR

Reports the position where the breakpoint for a motor is placed. Breakpoints are placed with the IP, IR, WP and WR commands. The interpretation of the command parameter *p* is explained at the beginning of this section.

```
2IP21.68                 ;define axis #2 breakpoint position = 21.68

2TB.2
    02    21.68          ;controller reply
```

TC Tell digital Channel

MCCL command: TC*x* *x* >= 0 <= 32
applies to: Digital I/O
see also: CL, CH, DF, DN, IF, IN

Reports the on/off status the digital input channels. To report the state of an individual digital input channel issue the **TC** command with *x* = channel number. To report the state of all 32 digital inputs channels as a 32 bit value issue the **TC** command with *x* = 0.

The controller responds by displaying the channel number and a "1" if the channel is "on", or a "0" if the channel is "off". **TTL inputs** - By default a TTL input (channels 1 - 16) is on if the input is a TTL high. **Optically Isolated inputs** - By default an optically isolated input (channels 1 - 16) is on if the device is conducting.

```
TC3
   03   1              ;controller reply, digital input #3 is on
```

TD   **T**ell **D**erivative gain

MCCL command:	aTDp a = Axis number *p* = 0, .1, .2, .3, .4, .5
applies to:	Analog Command Axis, Pulse Command Axis (Closed Loop)
see also:	SD

Reports the derivative gain setting of a closed loop axis.

```
1SD.4
   01   0.0032         ;controller reply
```

TE   **T**ell command interpreter **E**rror

MCCL command:	TE
applies to:	N/A
see also:	

Reports the last command interpreter error (syntax error, invalid character, etc.). For a listing of error codes please refer to the **MCCL Error Codes** chapter.

```
TE
   -2                  ;controller reply, command parameter error
```

TF   **T**ell **F**ollowing error

MCCL command:	*a*TF*p* a = Axis number *p* = 0, .1, .2, .3, .4, .5
applies to:	Analog Command Axis, Pulse Command Axis (Closed Loop)
see also:	SE

Reports the current following error of a closed loop axis. This error is the difference between the commanded position (calculated by the trajectory generator) and the current position.

```
1TF
   01   512            ;controller reply, following error = 512
```

TG Tell proportional Gain

MCCL command: *a*TG*p* a = Axis number *p* = 0, .1, .2, .3, .4, .5
applies to: Analog Command Axis, Pulse Command Axis (Closed Loop)
see also: SG

Reports the proportional gain setting for a closed loop axis.

```
      1TG.5
         01   0.00032        ;controller reply, proportional gain = 0.0032
```

TI Tell Integral gain

MCCL command: *a*TI*p* a = Axis number *p* = 0, .1, .2, .3, .4, .5
applies to: Analog Command Axis, Pulse Command Axis (Closed Loop)
see also: SI

Reports the integral gain setting for a closed loop axis.

```
      1TI.5
         01   0.00015        ;controller reply, integral gain = 0.00015
```

TK Tell velocity Konstant

MCCL command: aTKp a = Axis number *p* = 0, .1, .2, .3, .4, .5
applies to: Analog Command Axis, Pulse Command Axis (Closed Loop)
see also: VG

Reports the velocity constant for a servo. This is the value that was set with the Velocity Gain command. For a closed loop stepper axis this command reports the Velocity Gain that was set during initialization.

```
      1TK.5
         01   -0.00009       ;controller reply, velocity gain = -0.00009
```

TL Tell integral Limit setting

MCCL command: *a*TL*p* a = Axis number *p* = 0, .1, .2, .3, .4, .5
applies to: Analog Command Axis, Pulse Command Axis (Closed Loop)
see also: IL

Reports the integral limit setting of a closed loop axis.

```
      1TL.1
         01   5.0            ;controller reply, integral limit 5.0
```

TM Tell Macros

MCCL command: TM*n* *n* = integer >= -1, <= 1000
applies to: N/A
see also: MD, RM

Displays the commands that make up any macros that have been defined. If n = -1, all macros will be displayed. Since macros may be defined in any sequence, the TM command is useful for confirming the existence and/or contents of macro commands.

```
       1TM-1
                               ;controller reply, integral limit 5.0
          MC1,1LM2,1LN3,MJ10
          MC10,1VM,1DI0,1SV10000,1GO,LU"STATUS",1RL@0,IS24,MJ11,NO,IS17,MJ13,NO,JR-8
          MC11,LU"STATUS",1RL@0,IC24,MJ12,NO,JR-5
          MC12,1ST,1WS.1,1DI1,1SV5000,1GO,MJ14
          MC13,1WS0.01,1MN,1DI1,1SV5000,1GO,MJ14
          MC14,1FE0,1ST,1WS.1,1MF,1MN,1PM,1MA-100
          MC15,LU"STATUS",1RL@0,IS10,BK,NO,JR-5
```

TO Tell Optimal

MCCL command: *a*TO*p* a = Axis number *p* = 0, .1, .2, .3, .4, .5
applies to: Analog Command Axis, Pulse Command Axis (Closed Loop)
see also: TP, TT

Reports the current desired position of an axis. For a closed loop axis, the reported value will be different than the position reported by the TP command if a following error is present.

```
       1T0.3
          01   121.025        ;controller reply, optimal position = 121.025
```

TP Tell Position

MCCL command: *a*TP*p* a = Axis number *p* = 0, .1, .2, .3, .4, .5
applies to: Analog Command Axis, Pulse Command Axis (Closed Loop)
see also: DH, EL, FE, FI, IA,WE, WI

Reports the current position/encoder count of a closed loop axis or the 'step pulse count' of a stepper axis. It may be used to monitor motion during both Motor oN (MN) and Motor ofF (MF) states. The interpretation of the command parameter p is explained at the beginning of this section.

```
       1TP.3
          01   121.025        ;controller reply, current position = 121.025
```

TQ **T**ell tor**Q**ue

MCCL command:	*a*TQ*p* a = Axis number *p* = 0, .1, .2, .3, .4, .5
applies to:	Analog Command Axis
see also:	QM, SQ

Reports the current setting for the Set torQue command.

```
1TQ.1
   01   10.0           ;controller reply, current torque setting
```

TR **T**ell **R**egister '*n*'

MCCL command:	TR*n* *n* = integer >= 0, <= 255
applies to:	
see also:	AL, AR

Displays the contents of User Register *n*. When the command parameter is set to 0 (or not specified), this command reports the contents of User Register zero, which is the accumulator.

```
TR0
   01   15         ;controller reply, report value in register 0
                   ;(accumulator)
```

TS **T**ell axis **S**tatus

MCCL command : *a*TS*n* a = Axis number *n*= integer
applies to: Analog Command Axis, Pulse Command Axis
see also: ID, LU, RD, RL,

The Tell Status command is used to report the controller status. The following data can be reported by the Tell Status command:

Parameter n	Description
0	Primary Status word (32 bits)
1 - 31	Primary Status bits 1 - 31
64	Auxiliary Status word (32 bits)
65 - 95	Auxiliary Status bits 1 - 31
96	Profile Generator Status word (32 bits)
97 - 127	Profile Generator Status bits 1 - 31
128	Trajectory Status word (32 bits)
129 - 159	Trajectory Status bits 1 - 31
160	Encoder Fault Status word (32 bits)
161 - 191	Encoder Fault Status bit 1 - 31

Note: The Tell Status command can only be used to display data. In order to evaluate status data and execute conditional operations the **L**ook **U**p (**LU**) command should be used to load controller data into the accumulator. For additional information please refer to the description of **Reading Data from controller Memory** on page 26.

Tell Status description continued on the next 4 pages.

Primary Status Word (*n* = 0) - Reports the primary status word of axis *a*. If a bit is set its value will be reported as a '1', if a bit is cleared its value will be reported as a '0'.

Bit number	Description	Supports PCI-bus Interrupts? (Applies to MultiFlex PCI only)
0	Motor Error (Limit +/- tripped, max. following error exceeded)	Yes
1	Motor On	Yes
2	At Target	Yes
3	Trajectory Complete (Optimal = Target)	Yes
4	Direction (0 = positive, 1 = negative)	Yes
5	Position Captured	Yes
6	Breakpoint Reached (IP, IR)	Yes
7	Reserved	
8	Exceeded Max. Following Error *	Yes
9	Servo Amplifier / Stepper Driver Fault Tripped	Yes
10	Hard Limit Positive Tripped	Yes
11	Hard Limit Negative Tripped	Yes
12	Soft Motion Limit High Tripped	Yes
13	Soft Motion Limit Low Tripped	Yes
14	Primary Encoder Fault Tripped	Yes
15	Auxiliary Encoder Fault Tripped (not supported at this time)	
16	Reserved	
17	Looking For Index (FI, WI) ***/ Looking For Edge (FE, WE) ***	Yes
18	Index found (closed loop) ****/ Edge found (open loop stepper) ****	Yes
19	Looking For Aux. Index (open loop stepper)	Yes
20	Aux. Encoder Index found (open loop stepper)	Yes
21	Motor homed *****	Yes
22	Reserved	
23	Reserved	
24	Encoder Index (closed loop) / Stepper Home (current state) **	Consult factory
25	Encoder Coarse Home (current state)	Consult factory
26	Auxiliary Encoder Index (current state)	Consult factory
27	Servo Amplifier / Stepper Driver Fault (current state)	Consult factory
28	Limit Positive Input Active (current state)	Yes
29	Limit Negative Input Active (current state)	Yes
30	Reserved	
31	Reserved	

* Not supported by open loop stepper axes

** Defaults to displaying the current state of Index / Home input. Issuing Index Arm (Index) or Edge Latch (Home) will latch bit 24 upon activation of input. Motor off / Motor On commands return to reporting the current state of the input.

*** Set by initialization of homing (FI, IA, WI, EL, FE, WE). After event captured, cleared by Wait for Index (WI) or Wait for Edge (WE).

**** Set after armed (IA/FI or EL/FI) and event captured. Cleared by Motor oN (MN) after Wait for Index (WI) or Wait for Edge (WE).

***** Set after completed homing procedure (closed loop FI / IA & WI + MN or open loop FE / EL & WE + MN)

```
example:
   1TS0                         ;report the status of axis #1
      01  268436489             ;
```

The controller reply: 01 268436489 indicates:

;bit 28 Limit + Input Active set
;Bit 10 Hard Limit + Tripped set
;bit 3 Trajectory Complete set
;bit 0 Motor Error set

The controller defaults to formatting all replies as decimal values. For hexadecimal formatted replies issue the **H**ex **M**ode (**HM**) command. To return to decimal formatted replies issue the **D**ecimal **M**ode (**DM**) command.

```
example:
   HM                       ;Place controller in Hexadecimal Output Mode
    1TS                     ;report the status of axis #1
       01  100000409        ;
```

For a formatted display that reports the state of all primary status bits use the Information Display (ID) command.

Auxiliary Status Word (*n* = 64) - Reports the auxiliary status word of axis *a*. If a bit is set its value will be reported as a '1', if a bit is cleared its value will be reported as a '0'.

Bit number	Description	Supports PCI-bus Interrupts? (Applies to MultiFlex PCI only)
0	Hard Motion Limit Positive Enabled	No
1	Hard Motion Limit Negative Enabled	No
2	Hard Motion Limit Mode Abrupt	No
3	Hard Motion Limit Mode Smooth	No
4	Soft Motion Limit High Enabled	No
5	Soft Motion Limit Low Enabled	No
6	Soft Motion Limit Mode Abrupt	No
7	Soft Motion Limit Mode Smooth	No
8	Positive Limit Invert	No
9	Negative Limit Invert	No
10	Amplifier / Driver Fault Enabled	No
11	Phasing (0 = Standard, 1 = Reverse)	No
12	Backlash Compensation = On	No
13	Backlash Compensation Positive Direction	No
14	Backlash Compensation Negative Direction	No
15	Reserved	No
16	Low Speed	No
17	Medium Speed	No
18	High Speed	No
19	Reserved	No
20	Waiting For Primary Compare	No
21	Waiting For Primary Capture	No

22	Synchronization On	No
23	Ready for Synchronized move	No
24	Reserved	No
25	Reserved	No
26	Full Step (stepper only)	No
27	Full Current (stepper only)	No
28	Open Loop Enable PID changes (stepper only)	No
29	Reserved	No
30	Reserved	No
31	Reserved	No

To report the state of all Auxiliary Status bits issued the Tell Status command with parameter *n* = 64:

```
example:       1TS64
```

For a formatted display that reports the state of all auxiliary status bits use the Information Display (ID) command.

Profile Generator Status Word (*n* = 96) - Reports the profile generator status word of axis *a*. If a bit is set its value will be reported as a '1', if a bit is cleared its value will be reported as a '0'.

Bit number	Description	Supports PCI-bus Interrupts? (Applies to MultiFlex PCI only)
0	Trajectory Complete	No
1	Direction (0 = positive, 1 = negative)	No
2	Desired Direction	No
3	Acceleration Mode	No
4	Reserved	No
5	Motor Stopping	No
6	Reserved	No
7	Reserved	No
8	Reserved	No
9	Reserved	No
10	Reserved	No
11	Reserved	No
12	Trapezoidal profile	No
13	S-curve profile	No
14	Parabolic profile	No

To report the state of all Profile Generator Status bits issued the Tell Status command with parameter *n* = 96:

```
example:       1TS96
```

Trajectory Mode Status Word (*n* = 128) - Reports the trajectory mode status word of axis *a*. If a bit is set its value will be reported as a '1', if a bit is cleared its value will be reported as a '0'.

Bit number	Description	Supports PCI-bus Interrupts? (Applies to MultiFlex PCI only)
0	Position Mode	No
1	Velocity Mode	No
2	Gain Mode	No
3	Torque Mode	No
4	Contour Mode	No
16	Slave to Master Axis	No
17	Slave to Master Contour	No
18	Slave to Master Encoder	No
19	Slave to Master Index Mark	No

To report the state of all Trajectory Mode Status bits issued the Tell Status command with parameter *n* = 128:

```
example:      1TS128
```

Encoder Fault Status Word (*n* = 128) - Reports the encoder fault status word of axis *a*. If a bit is set its value will be reported as a '1', if a bit is cleared its value will be reported as a '0'.

Bit number	Description	Supports PCI-bus Interrupts? (Applies to MultiFlex PCI only)
0	Primary Encoder Fault *	No
1	Primary Encoder Phase A Fault *	No
2	Primary Encoder Phase B Fault *	No
3	Primary Encoder Phase Z Fault *	No
4	Auxiliary Encoder Fault (not supported at this time)	No
12	Primary Encoder Fault Enabled *	No
16	Auxiliary Encoder Fault Enabled (not supported at this time)	No

* Supports axes 1, 3, 5, and 7
** Supports axes 1 - 8

To report the state of all Trajectory Mode Status bits issued the Tell Status command with parameter *n* = 160:

```
example:      1TS160
```

For a formatted display that reports the state of all encoder fault status bits use the Information Display (ID) command.

TT Tell Target

MCCL command: *a*TT*p* a = Axis number *p* = 0, .1, .2, .3, .4, .5
applies to: Analog Command Axis, Pulse Command Axis
see also: MA, MR

Reports target position. This is the position to which the servo or stepper motor was last commanded to move. It may be specified directly with the Move Absolute (MA) command or indirectly with the Move Relative (MR) command. The interpretation of parameter *p* is explained at the beginning of this section.

```
1TT.3
   01   121.025        ;controller reply, target position = 121.025
```

TV Tell Velocity

MCCL command : aTVp a = Axis number *p* = 0, .1, .2, .3, .4, .5
applies to: Analog Command Axis, Pulse Command Axis
see also: DS, SA, SV, UR, US

Reports the current commanded velocity of a servo or stepper motor. The reported value is scaled by the current setting for User Scaling and User Rate.

```
1TV.2
   01   10.00          ;controller reply current velocity
```

TX Tell contouring count

MCCL command: aTX *a* = Axis number
applies to: Analog Command Axis, Pulse Command Axis
see also: CP

Reports the current contour path motion that an axis is performing. The value that the controller replies is only valid for the controlling axis in a group of axes performing contoured path motion. After the Contour Mode command is issued to an axis, the TX command will have a reply value of 0. For each Linear or User Defined Contour Path motion that the controller completes, the contouring count will be incremented by one. For Arc Contour Path motions, the count will be incremented by 2. By counting the number of Contour Path commands that have been issued to the controller (1 for linear, 2 for arc), and comparing it to the response from the TX command, the user can determine on what segment of a continuous path motion the motors are on. The contour count is stored as a 32 bit value (0 - 2,147,483,647). To reset the contour count value and avoid ‘wrap around’, the user should stop motion and issue the Contour Mode command.

```
1TX
   01   12             ;controller reply, current contour count = 12
```

TZ Tell index position

MCCL command:	*a*TZ*p* a = Axis number *p* = 0, .1, .2, .3, .4, .5
applies to:	Analog Command Axis, Pulse Command Axis (Closed Loop)
see also:	

Reports the position where the index pulse was observed. This position is relative to the encoder's position when the controller was reset or a Define Home command was issued to the axis.

```
1TZ.3
   01   1990.200       ;controller reply, index position = 1990.200
```

VE tell firmware VErsion

MCCL command:	VE
applies to:	N/A
see also:	

Reports the revision level of the firmware running on the controller. This command also displays the amount of memory installed on the motion control card.

example: VE

Returns the following reply with a MultiFlex PCI 1440 controller:

```
MFX-PCI1440-3-A Motion Controller
Hardware: 16384K Private RAM, 512K Flash Memory System Firmware Ver. PM1
Rev. 4.5a Copyright (c) 2002-2006 Precision MicroControl Corporation All
rights reserved.
```

Chapter Contents

I/O Commands

MCCL	Description
CF	Channel ofF
CH	Channel High true logic
CL	Channel Low true logic
CN	Channel oN
GA	Get Analog
DF	Do if channel oFf
DN	Do if channel oN
IF	If channel oFf
IN	If channel oN
TA	Tell the value of Analog input
TC	Tell state of digital Channel
WF	Wait for channel ofF
WN	Wait for channel oN

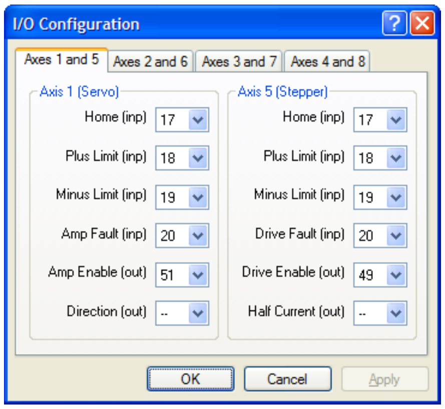
Chapter 9

I/O Commands

The basic controller provides:

> 16 TTL inputs (digital I/O channels 1 - 16)
> 16 optically isolated inputs (digital I/O channels 17 - 32)
> 16 TTL outputs (digital I/O channels 33 - 48)
> 12 open collector outputs (digital I/O channels 49 - 64) (channel #'s 52, 56, 60, 64 reserved)
> 8 optional analog inputs (analog channels 1 - 8)

By default the optically isolated inputs and open collector outputs are associated with 'hard coded' motion control functions (Limits, Homing, Amp/Drive fault, Amp/Drive Enable). For maximum application flexibility the controller allows the user to reassign most of the default digital I/O assignments. The Windows I/O Configuration dialog is used to change the default digital I/O configuration.



Axis 1 (Servo)

Home (inp) 17
Plus Limit (inp) 18
Minus Limit (inp) 19
Amp Fault (inp) 20
Amp Enable (out) 51
Direction (out) --

Axis 5 (Stepper)

Home (inp) 17
Plus Limit (inp) 18
Minus Limit (inp) 19
Drive Fault (inp) 20
Drive Enable (out) 49
Half Current (out) --

Figure 8: MultiFlex Digital I/O configuration panel

The configuration dialog is launched from the Motion Control Panel (\Properties\Advanced\Configure)

The Stepper Home function **cannot be reassigned** to a different digital input channel. An open loop stepper axis can only be 'homed' by applying an active level on VHDCI connector pin #27. Capture and compare functions cannot be reassigned to different digital I/O channels.

Table 1. Default I/O configuration

Ch. #	Description	Ch. #	Description
1	TTL input 1 (Capture #1, axes 1 & 2)	33	TTL output 1 (Compare #1, axes 1 - 4)
2	TTL input 2	34	TTL output 2
3	TTL input 3	35	TTL output 3
4	TTL input 4	36	TTL output 4
5	TTL input 5 (Capture #2, axes 3 & 4)	37	TTL output 5
6	TTL input 6	38	TTL output 6
7	TTL input 7	39	TTL output 7
8	TTL input 8	40	TTL output 8
9	TTL input 9 (Capture #3, axes 5 & 6)	41	TTL output 9 (Compare #2, axes 5 - 8)
10	TTL input 10	42	TTL output 10
11	TTL input 11	43	TTL output 11
12	TTL input 12	44	TTL output 12
13	TTL input 13 (Capture #4, axes 7 & 8)	45	TTL output 13
14	TTL input 14	46	TTL output 14
15	TTL input 15	47	TTL output 15
16	TTL input 16	48	TTL output 16
17	Opto isolated (5V - 24V) Axis 1 Coarse Home Axis 5 Home	49	To be determined
18	Opto isolated (5V - 24V) Axis 1/5 Limit +	50	To be determined
19	Opto isolated (5V - 24V) Axis 1/5 Limit -	51	To be determined
20	Opto isolated (5V - 24V) Axis 1/5 Amp Fault	52	To be determined
21	Opto isolated (5V - 24V) Axis 2 Coarse Home Axis 3 Home	53	To be determined
22	Opto isolated (5V - 24V) Axis 2/6 Limit +	54	To be determined
23	Opto isolated (5V - 24V) Axis 2/6 Limit -	55	To be determined
24	Opto isolated (5V - 24V) Axis 2/6 Amp Fault	56	To be determined
25	Opto isolated (5V - 24V) Axis 3 Coarse Home Axis 4 Home	57	To be determined
26	Opto isolated (5V - 24V) Axis 3/7 Limit +	58	To be determined
27	Opto isolated (5V - 24V) Axis 3/7 Limit -	59	To be determined
28	Opto isolated (5V - 24V) Axis 3/7 Amp Fault	60	To be determined
29	Opto isolated (5V - 24V) Axis 4 Coarse Home Axis 8 Home	61	To be determined
30	Opto isolated (5V - 24V) Axis 4/8 Limit +	62	To be determined
31	Opto isolated (5V - 24V) Axis 4/8 Limit -	63	To be determined
32	Opto isolated (5V - 24V) Axis 4/8 Amp Fault	64	To be determined

CF Channel oFf

MCCL command:	CF*x* *x* = Channel number
applies to:	Digital I/O
see also:	CN

Causes digital output x to go to "off" state. If the channel has been configured for "high true", the channel will be at a logic low (less that 0.4 volts DC) after this command is executed. If it has been configured for "low true", the channel will be at a logic high (greater than 2.4 volts DC).

```
        CF33                                    ;turn off digital I/O channel 33
                                                ;(TTL output #1)
```

CH Channel High

MCCL command :	CH*x* *x* = Channel number
applies to:	Digital I/O
see also:	CF, CL, CN, TS

Causes digital I/O channel x to be configured for "high true" or positive logic. By default all I/O channels are set to 'high true' logic.

TTL inputs (channels 1 - 16) - if the input signal is greater than 2.4 volts DC the Tell Channel (TC*x*) command will report a 1 (channel is on).

Opto Isolated inputs (channels 17 - 32) - if the levels of the two inputs cause the opto device to 'conduct' the Tell Channel (TC*x*) command will report a 1 (channel is on).

TTL outputs (channels 33 - 48) - issuing the Channel oN (CN*x*) command will set the output to a TTL 'high' (greater than 2.4 volts DC).

Open collector outputs (channels 49 - 64) - issuing the Channel oN (CN*x*) command will cause the open collector to 'sink' current.

Notes:
1) Issuing this command will not cause an output channel to change its current state.
2) Issuing this command without specifying a channel (or *x* = 0) will cause all channels present on the controller to be configured as "high true".
3) Changing the active logic level of a digital I/O channel that also is mapped into the Primary Axis Status word (aTS) will cause the associated status bit to change state. For example, if digital I/O channel #17 (Axis 1/5 opto isolated input) is wired for a normally closed switch the opto should be wired to be conducting unless the switch is opened or a wire is broken. The TC and TS command will indicate that the input is on while the device is conducting (which the controller will interpret as an error condition). To cause the TC and TS commands to indicate that the 'limit' is active when the switch is opened issue the Channel Low command with X = 17 (CL17).

CL Channel Low

MCCL command :	CL*x* *x* = Channel number
applies to:	Digital I/O

see also:	CF, CH, CN, TS

Causes digital I/O channel x to be configured for "low true" or negative logic. By default all I/O channels are set to 'high true' logic. After issuing the CL command:

TTL inputs (channels 1 - 16) - if the input signal is less than 0.4 volts DC the Tell Channel (TC*x*) command will report a 1 (channel is on).

Opto Isolated inputs (channels 17 - 32) - if the levels of the two inputs do not cause the opto device to 'conduct' the Tell Channel (TC*x*) command will report a 1 (channel is on).

TTL outputs (channels 33 - 48) - issuing the Channel oN (CN*x*) command will set the output to a TTL 'low' (less than 0.4 volts DC).

Open collector outputs (channels 49 - 64) - issuing the Channel oFf (CF*x*) command will cause the open collector to 'sink' current.

Notes:
1) Issuing this command will not cause an output channel to change its current state.
2) Issuing this command without specifying a channel (or *x* = 0) will cause all channels present on the controller to be configured as "low true".
3) Changing the active logic level of a digital I/O channel that also is mapped into the Primary Axis Status word (aTS) will cause the associated status bit to change state. For example, if digital I/O channel #17 (Axis 1/5 opto isolated input) is wired for a normally closed switch the opto should be wired to be conducting unless the switch is opened or a wire is broken. The TC and TS command will indicate that the input is on while the device is conducting (which the controller will interpret as an error condition). To cause the TC and TS commands to indicate that the limit' is active when the switch is opened issue the Channel Low command with X = 17 (CL17).

CN Channel oN

MCCL command:	CN*x* *x* = Channel number
applies to:	Digital I/O
see also:	CF, CH, CL

Causes digital output channel x to go to "on" state. If the channel has been configured for "high true", the channel will be at a logic high (greater than 2.4 volts DC) after this command is executed. If it has been configured for "low true", the channel will be at a logic low (less that 0.4 volts DC).

```
      CN34                                  ;turn on digital I/O channel 34
                                            :(TTL output #2)
```

DF **D**o if channel o**F**f

MCCL command:	DF*x* x = Channel number
applies to:	Digital I/O
see also:	CH, CL, DN, IF, IN

Used for conditional execution of commands. If digital I/O channel *x* is "off", commands that follow on the command line or in the macro will be executed. Otherwise the rest of the command line or macro will be skipped.

```
        DF2,1MR1000,1WS0.1                          ;If channel 2 is off move 1000
```

DN **D**o if channel ‘x’ is o**N**

MCCL command:	DN*x* x = Channel number
applies to:	Digital I/O
see also:	CH, CL, DF, IF, IN

Used for conditional execution of commands. If digital I/O channel *x* is "on", commands that follow on the command line or in the macro will be executed. Otherwise the rest of the command line or macro will be skipped.

```
        DN3,1MR1000,1WS0.1                          ;If channel 2 is off move 1000
```

GA **G**et **A**nalog

MCCL command:	GA*x* x = Channel number
applies to:	A/D inputs (if A/D option is installed)
see also:	TA

Performs analog to digital conversion on the specified input channel and places the result into the Accumulator (User Register 0). Analog channels are numbered from 1 - 8.
Note: requires the A/D option.

```
        GA2,IG2048,BK,NO,JR-4                       ;loop until A/D reading > 2048
```

IF **I**f channel o**F**f do next command, else skip 2 commands

MCCL command:	IF*x* x = Channel number
applies to:	Digital I/O
see also:	CH, CL, DF, DN, IN

Used for conditional execution of commands. If digital I/O channel *x* is "off", command execution will continue with the command following the IF command. Otherwise the two commands following the IF command will be skipped, and command execution will continue from the third command.

```
      IF5,MJ10,NO,MJ11                        ;If digital input #5 is off jump to
                                              ;macro 10, otherwise jump to macro 11
```

IN — **I**f channel ' o**N** do next command, else skip 2 commands

MCCL command: IN*x*   x = Channel number
applies to: Digital I/O
see also: CH, CL, DF, DN, IF

Used for conditional execution of commands. If digital I/O channel *x* is "on", command execution will continue with the command following the IN command. Otherwise the two commands following the IN command will be skipped, and command execution will continue from the third command.

```
      IN5,MJ10,NO,MJ11                        ;If digital input #5 is on jump to
                                              ;macro 10, otherwise jump to macro 11
```

TA — **T**ell **A**nalog

MCCL command: TAx   *x* = Channel number
applies to: A/D inputs (if A/D option is installed)
see also: TA

Displays the digitized analog input values. For each channel, the TA command will display a number between 0 and 65536. The displayed value is the ratio of the analog input voltage to the reference voltage (+4V or +10V) multiplied by 65536.
Note: requires the A/D option.

TC — **T**ell digital **C**hannel

MCCL command: TC*x*   *x* = Channel number
applies to: Digital I/O
see also: CH, CL, DF, DN, IF, IN

Reports the on/off status of each digital I/O line. This data is reported separately for each channel. The controller responds by displaying the channel number and a "1" if the channel is "on", or a "0" if the channel is "off".

```
      TC3
         03   1              ;controller reply, digital input #3 is on
```

WF Wait for digital channel oFf

MCCL command: WF*x* x = Channel number
applies to: Digital I/O
see also: CH, CL, WN

Wait until digital I/O channel x is "off" before continuing to the next command on the command line or in the macro. If this command was issued from an ASCII interface, it can be aborted by sending an Escape character.

```
WF2,1MR1000
```

WN Wait for digital channel oN

MCCL command: WN*x* x = Channel number
applies to: Digital I/O
see also: CH, CL, WF

Wait until digital I/O channel x is "on" before continuing to the next command on the command line or in the macro. If this command was issued from an ASCII interface, it can be aborted by sending an Escape character.

```
WN2,1MR1000
```

Chapter Contents

Macro Commands

MCCL	Description
BK	BreaK
ET	Escape Task
GT	Generate Task
MC	Macro Call
MD	Macro Definition
MJ	Macro Jump
NO	No Operation
NP	New Priority
RM	Reset Macros
TM	Tell Macros

Chapter 10

Macro and Multi-tasking Commands

BK BreaK

MCCL command: BK
applies to: Program flow
see also: GT, IC, IF, IN, IS, TR

Execution of this command will cause the rest of the command line or macro to be skipped. This command is typically used in conjunction with the If oN and If ofF commands to implement conditional execution.

To enable single stepping of a MCCL program use the break command immediately followed by a “string” parameter. When the break command is executed the controller will display the characters in the string (inside the quotation marks) and then delay additional command execution until the space bar (execute next command and then delay) or the enter key (terminate single stepping and resume program execution) are selected.

```
IN5,BK,NO,MJ11                          ;If digital input #5 is on jump to
                                        ;macro 10, otherwise jump to macro 11
```

ET Escape Task

MCCL command: ET*n* *n* = integer
applies to: Program flow
see also: GT, TR

This command is used to terminate a 'background task' that was created with the Generate Task command. The parameter to this command must be the task identifier that was placed in the accumulator (user register 0) of the task that issued the Generate Task command. A background task can use this command to terminate itself, but it must first acquire its identifier from the 'parent' task through a global register. Note that the task that interprets and executes commands received from the command interfaces cannot be terminated. See the description of **Multi-Tasking** in **chapter 4**.

GT Generate Task

MCCL command: GT*n* *n* = integer > = 0, < 1000

applies to:	Program flow
see also:	ET, MC, MD, TR

This command will cause macro *n* to be executed as a background task. Alternatively, this command can precede a sequence of commands. In this case, the commands following the Generate Task command will be executed as a background task. After this command is issued, an identifier for the background task will be placed in the accumulator (register 0) of the task that issued the command. This identifier can be used as the parameter to the Escape Task command to terminate the background task. See the description of **Multi-Tasking** in **chapter 4**.

MC Macro Call

MCCL command:	MC*n* *n* = integer > = 0, < 1000
applies to:	Program flow
see also:	ET, MD

This command may be used to execute a previously defined macro command. If there is no macro defined by the number n, an error message will be displayed. Macro Call Commands can also be used in compound commands with other commands in the instruction set. In addition, a macro command can call another macro command, which in turn can call another macro command, and so on. If macros are 'nested' (one macro calls another) when the nested macro has completed execution the command interpreter will return to the previously called macro. See the description of **Building MCCL Macro Sequences** in **chapter 4**.

MD Macro Define

MCCL command:	MD*n* *n* = integer > = 0, < 1000
applies to:	Program flow
see also:	ET, GT, MD, RM

Used to define a new macro. This is done by placing the Macro Define command as the first command in a sequence of commands. All commands following the Macro Define command will be included in the macro. See the description of **Building MCCL Macro Sequences** in **chapter 4**.

Macros will erased if power to the board is turned off. A macro can be redefined but the memory space occupied by the previous version of the macro will not be reused until a Reset Macro command is issued. Thus, if macro *n* already exists when a Macro Define command for that macro is issued, the previously defined macro will be replaced by the new macro definition.

MJ **M**acro **J**ump

MCCL command: MJ*n* *n* = integer > = 0, < 1000
applies to: Program flow
see also: ET, GT, MD

Jumps to a previously defined macro. This command differs from the Macro Call command in that execution will not return to the command following the MJ command.

```
IN5,BK,NO,MJ11                      ;If digital input #5 is on jump to
                                    ;macro 10, otherwise jump to macro 11
```

NO **N**o **O**peration

MCCL command: NO
applies to: Program flow
see also: MC, MD, BK

This command does nothing. It can be used to cause short delays in command line executions or as a filler in sequence commands.

```
IN5,BK,NO,MJ11                      ;If digital input #5 is on jump to
                                    ;macro 10, otherwise jump to macro 11
```

NP **N**ew **P**riority

MCCL command: NP*n* *n* = integer > = 11, <= 255
applies to: Program flow
see also: ET, GT, MC

This command is used to change the priority of a MCCL command sequence. This command changes only the priority of the task in which the command was issued. By default all MultiFlex tasks are issued with priority of 100. The highest priority that can be defined is 11 and the lowest priority is 255.

RM **Reset Macros**

MCCL command: RM
applies to: Program flow
see also: MC, MD

This command will initialize the memory space used for storage of macro commands. It has the effect of erasing currently defined macros from memory. It is also the only way in which macro commands can be removed from memory after they are defined. It is always a good idea to use the Reset Macro command (RM) before setting up a new set of macro commands. See the description of **Macro Commands** in the **Working with MCCL Commands** chapter.

TM Tell Macros

MCCL command: TM*n* *n* = integer >= -1, < 1000
applies to: Program flow
see also: MD, RM

Displays the commands that make up any macros that have been defined. If n = -1, all macros will be displayed. Since macros may be defined in any sequence, the TM command is useful for confirming the existence and/or contents of macro commands.

```
1TM-1                  ;controller reply

   MC1,1LM2,1LN3,MJ10
   MC10,1VM,1DI0,1SV10000,1GO,LU"STATUS",1RL@0,IS24,MJ11,NO,IS17,MJ13,NO,JR-8
   MC11,LU"STATUS",1RL@0,IC24,MJ12,NO,JR-5
   MC12,1ST,1WS.1,1DI1,1SV5000,1GO,MJ14
   MC13,1WS0.01,1MN,1DI1,1SV5000,1GO,MJ14
   MC14,1FE0,1ST,1WS.1,1MF,1MN,1PM,1MA-100
   MC15,LU"STATUS",1RL@0,IS10,BK,NO,JR-5
```

Chapter Contents

Register Commands

MCCL	Description
AA	Accumulator Add
AC	Accumulator Complement
AD	Accumulator Divide
AE	Accumulator logical Exclusive or
AL	Accumulator Load
AM	Accumulator Multiply
AN	Accumulator logical aNd with n,
AO	Accumulator logical Or with n
AR	copy Accumulator to Register n
AS	Accumulator Subtract
AV	Accumulator eValuate
AX	get Aux. indeX position
GA	Get Analog value
GU	Get the default axis
GX	Get auXiliary encoder position
LU	Look Up motor table variable
RA	copy Register to Accumulator
RB	Read Byte into accumulator
RD	Read Double into accumulator
RL	Read Long into accumulator
RV	Read float into accumulator
RW	Read Word into accumulator
SL	Shift Left accumulator n bits
SR	Shift Right accumulator n bits
TR	Tell contents of Register n
WB	Write accumulator Byte to n
WD	Write accumulator double to n
WL	Write accumulator Long to n
WV	Write accumulator float to n
WW	Write accumulator Word to n

Chapter 11

Register Commands

AA Accumulator Add

MCCL command: AAn *n* = integer or real

Performs ACC = ACC + *n*, the addition of the command parameter ***n*** to the Accumulator (User Register 0). If the command parameter is in integer format, the result is stored in the Accumulator as a 32 bit integer. If the command parameter is in real format, the result is stored in the Accumulator (User Register 0 and 1) as a 64 bit real value.

```
AL@100,AA5,AR101          ;load accumulator with the value in register 100,
                          ;add 5, store result in register 101
```

AC Accumulator Complement, bit wise

MCCL command: AC

Performs ACC = !ACC, the bit wise logical complement of the Accumulator (User Register 0). The result is stored in the Accumulator as a 32 bit integer.

```
al@100,AC,AR101           ;load accumulator with the value in register 100,
                          ;bit wise complement, store result in register 101
```

AD Accumulator Divide

MCCL command: ADn *n* = integer or real

Performs ACC = ACC/*n*, the division of the Accumulator (User Register 0) by the command parameter. If the command parameter is in integer format, the result is stored in the Accumulator as a 32 bit integer. If the command parameter is in real format, the result is stored in the Accumulator (User Register 0 and 1) as a 64 bit real value. No operation is done if the command parameter is zero.

```
al@100,AD@101,AR102       ;load accumulator with the value in register 100,
                          ;divide by value in register 101, store result in
                          ;register 102
```

AE Accumulator logical Exclusive or with '*n*' , bit wise

MCCL command: AE*n* *n* = integer or real

Performs ACC = ACC ^ *n*, the bit wise logical exclusive or'ing of the Accumulator (User Register 0) with the command parameter. The result is stored in the Accumulator as a 32 bit integer.

```
al@100,AE@101,AR102        ;load accumulator with the value in register 100,
                           ;perform exclusive or with value in register 101,
                           ;store result in register 102
```

AL Accumulator load

MCCL command: AL*n* *n* = integer or real

Loads the Accumulator (User Register 0) with *n* . If the command parameter is an integer (no decimal point or exponent label) the Accumulator will be marked as containing a 32 bit integer, otherwise it will be marked as containing a 64 bit real value.

```
AL12345                                  ;Load 12345 into the accumulator
AL1234.5                                 ;Load 1234.5 into the accumulator
AL0.12345                                ;Load 0.12345 into the accumulator
```

AM Accumulator Multiply

MCCL command: AM*n* *n* = integer or real

Performs ACC = ACC * *n*, the multiplication of the Accumulator (User Register 0) by the command parameter. If the command parameter is in integer format, the result is stored in the Accumulator as a 32 bit integer. If the command parameter is in real format, the result is stored in the Accumulator (User Register 0 and 1) as a 64 bit real value.

```
al@100,AM@101,AR102        ;load accumulator with the value in register 100,
                           ;multiply by value in register 101, store result in
                           ;register 102
```

AN Accumulator logical 'aNd' the '*n*' , bit wise

MCCL command: AN*n* *n* = integer or real

Performs ACC = ACC & *n*, the bit wise logical AND of the Accumulator (User Register 0) with the command parameter. The result is stored in the Accumulator as a 32 bit integer.

```
al@100,AN@101,AR102        ;load accumulator with the value in register 100,
                            ;and with value in register 101, store result in
                            ;register 102
```

AO Accumulator logical ‘Or’ with ‘*n*’ , bit wise

MCCL command : AO*n* *n* = integer or real

Performs ACC = ACC | *n*, the bit wise logical OR of the Accumulator (User Register 0) with the command parameter. The result is stored in the Accumulator as a 32 bit integer.

```
al@100,AO@101,AR102        ;load accumulator with the value in register 100,
                            ;or with value in register 101, store result in
                            ;register 102
```

AR copy Accumulator to Register

MCCL command: ARn *n* = integer or real

Copies the contents of the Accumulator (User Register 0) to the User Register specified by *n*. The contents of the Accumulator are unaffected by this command.

```
al@100,AD@101,AR102        ;load accumulator with the value in register 100,
                            ;divide by value in register 101, store result in
                            ;register 102
```

AS Accumulator Subtract

MCCL command: AS*n* *n* = integer or real

Performs ACC = ACC - *n*, the subtraction of the command parameter from the Accumulator (User Register 0). If the command parameter is in integer format, the result is stored in the Accumulator as a 32 bit integer. If the command parameter is in real format, the result is stored in the Accumulator (User Register 0 and 1) as a 64 bit real value.

```
al@100,AS@101,AR102        ;load accumulator with the value in register 100,
                            ;subtract by value in register 101, store result in
                            ;register 102
```

AV Accumulator eValuate

MCCL command: AVn *n* = integer >= 0, <=25

Performs a unary operation on the contents of the Accumulator (User Register 0), placing the result in the Accumulator, overwriting the original contents. Parameter *n* specifies the desired operation. The table below list the available operations and the respective command parameter to use. The result that is stored in the Accumulator (1<= *n* <= 25) will be a 64 bit real in all cases except the Convert to ASCII operation that returns an integer.

Parameter n =	Operation	Return type
1	Convert to ASCII (Address placed in ACC)	Integer
2	Change Sign	Double
3	Absolute Value	Double
4	Ceiling	Double
5	Floor	Double
6	Fraction	Double
7	Round	Double
8	Square	Double
9	Square Root	Double
10	Sine	Double
11	Cosine	Double
12	Tangent	Double
13	Arc Sine	Double
14	Arc Cosine	Double
15	Arc Tangent	Double
16	Hyperbolic Sine	Double
17	Hyperbolic Cosine	Double
18	Hyperbolic Tangent	Double
19	Exponent	Double
20	Log	Double
21	Log10	Double
22	Load Pi	Double
23	Load 2 * Pi	Double
24	Load Pi/2	Double
25	Convert double register contents to an integer	Integer

AX get **A**uxiliary encoder inde**X** position

MCCL command: *a*AXn a = Axis number
applies to: Pulse Command Axis
see also: GX

Loads the accumulator with the position of the auxiliary encoder when the index pulse was last captured (using the **A**uxiliary encoder **F**ind command). The value is relative to the auxiliary encoder's position when the controller was reset or an Auxiliary encoder Home command was last issued to the axis.

```
5AX                         ;load accumulator with last aux. encoder index
                            ;position
```

GA **G**et **A**nalog

MCCL command: GA*x* x = Channel number
applies to: A/D inputs (if A/D option is installed)
see also:

Performs analog to digital conversion on the specified input channel and places the result into the Accumulator (User Register 0). Analog channels are numbered starting with 1.
Note: requires the A/D option.

```
GA2,IG2048,BK,NO,JR-4                 ;loop until A/D reading > 2048
```

GU **G**et axis **U**sed

MCCL command: GU

The controller defaults to setting the default axis to zero. If the user executes a motion or setup command with the axis specifier missing, the default axis will be used. In most cases a motion or setup command issued to axis zero commands that operation to all axes. By defining a non-zero default axis, the user can execute ‘generic’ macro’s (no axis number specified) to any axis.
This Get axis Used command is used to report the current default axis by placing the current setting into the accumulator. The default axis is defined by using the setup command set the **U**se **A**xis command (**UA*n***).

```
GU                          ;load accumulator with the current default axis
```

GX — **G**et the position of the auxiliary encoder

MCCL command: *a*GX *a* = Axis number
applies to: Pulse Command Axis
see also: AT, AX

This command reads the auxiliary encoder associated with axis *a* and places the value into the Accumulator (User Register 0).

```
        5GX                         ;load accumulator with the position of axis 5
                                     ;auxiliary encoder
```

LU — **L**ook **U**p variable

MCCL command: LU*s* s= *string parameter ("variable name")*

Loads the accumulator with the memory location for a motor table data entry. For additional information including a complete listing of variable names please refer to the description of **Reading Data from controller Memory** in **Chapter 4** of this manual.

```
        LU"STATUS",2RL@0            ;load the motor table address for axis #2 status
                                     ;into the accumulator. Load the 32 bit status word
                                     ;of axis #2 into the accumulator
```

RA — copy **R**egister to **A**ccumulator

MCCL command: RAn *n* = integer or real

Copies the contents of the User Register *n* into the Accumulator (User Register 0). The original contents of the accumulator is overwritten, while the contents of the source User Register are unaffected.

```
        RA100                           ;copy value in register into the accumulator
```

RB — **R**ead the **B**yte at absolute memory location '*n*' into the accumulator

MCCL command: *a*RB*n* *a* = Axis number *n* = integer

This is used to load the accumulator with a byte of data. Typically this command is used in conjunction with the **L**ook **U**p command to reference a Motor Table variable. The **Reading Data from controller Memory** section of Chapter 4 provides a detailed description of the Motor Tables.

RD — **R**ead the **Double** (64 bit real) value at absolute memory location '*n*' into the accumulator

MCCL command: *a*RD*n* *a* = Axis number *n* = real

This is used to load the accumulator with a 64 bit real value. Typically this command is used in conjunction with the **L**ook **U**p command to reference a Motor Table variable. The **Reading Data from controller Memory** section of Chapter 4 provides a detailed description of the Motor Tables.

RL — **R**ead the **L**ong (32 bit integer) value at absolute memory location '*n*' into the accumulator

MCCL command: *a*RL*n* *a* = Axis number *n* = integer

This is used to load the accumulator with a 32 bit integer. Typically this command is used in conjunction with the **L**ook **U**p command to reference a Motor Table variable. The **Reading Data from controller Memory** section of Chapter 4 provides a detailed description of the Motor Tables.

RV — **R**ead the float (32 bit real) value at absolute memory location '*n*' into the accumulator

MCCL command: *a*RV*n* *a* = Axis number *n* = real

This is used to load the accumulator with a 32 bit real value. Typically this command is used in conjunction with the **L**ook **U**p command to reference a Motor Table variable. The **Reading Data from controller Memory** section of Chapter 4 provides a detailed description of the Motor Tables.

RW — **R**ead the **W**ord (16 bit integer) value at absolute memory location '*n*' into the accumulator

MCCL command: *a*RW*n* *a* = Axis number *n* = integer

This is used to load the accumulator with a 16 bit integer. Typically this command is used in conjunction with the **L**ook **U**p command to reference a Motor Table variable. The **Reading Data from controller Memory** section of Chapter 4 provides a detailed description of the Motor Tables.

SL **S**hift **L**eft accumulator by ‘*n*’ bits

MCCL command: SL*n* *n* = integer > 0, < = 31

Performs ACC = ACC << *n*, the logical shift of the Accumulator (User Register 0) to the left. The command parameter specifies the number of bits to shift the accumulator. Zero bits will be shifted in on the right. The result is stored in the Accumulator as a 32 bit integer.

```
        al@100,SL@101,AR102        ;load accumulator with the value in register 100,
                                    ;shift left number of bits in register 101, store
                                    ;result in register 102
```

SR **S**hift **R**ight accumulator by ‘*n*’ bits

MCCL command: SR*n* *n* = integer > 0, < = 31

Performs ACC = ACC >> *n* , the logical shift of the Accumulator (User Register 0) to the right. The command parameter specifies the number of bits to shift the accumulator. Zero bits will be shifted in on the left. The result is stored in the Accumulator as a 32 bit integer.

```
        al@100,SR@101,AR102        ;load accumulator with the value in register 100,
                                    ;shift right number of bits in register 101, store
                                    ;result in register 102
```

TR **T**ell **R**egister ‘*n*’

MCCL command: TR*n* *n* = integer >= 0, <= 255
see also: AL, AR

Displays the contents of User Register *n*. When the command parameter is set to 0 (or not specified), this command reports the contents of User Register zero, which is the accumulator.

```
        TR0
           01   15          ;controller reply, report value in register 0 (accumulator)
```

WB **W**rite the low **B**yte in the accumulator to absolute memory location ‘*n*’

MCCL command: WB*n* *n* = integer

This command will copy the low byte of the accumulator (User Register 0) to the byte located at absolute memory address *n*.
Note: recommended for factory use only.

WD — **W**rite the **D**ouble (64 bit real) value in the accumulator to absolute memory location ‘*n*’

MCCL command: WD*n* *n* = real

This command will copy a Double (64 bit real) from the accumulator (User Register 0 and 1) to absolute memory address *n*.
Note: recommended for factory use only.

WL — **W**rite the **L**ong (32 bit integer) value in the accumulator to absolute memory location ‘*n*’

MCCL command: WL*n* *n* = integer

This command will copy a Long (32 bit integer) from the accumulator (User Register 0) to absolute memory address *n*.
Note: recommended for factory use only.

WV — **W**rite the float (32 bit real) value in the accumulator to absolute memory location ‘*n*’

MCCL command: WV*n* *n* = real

This command will copy a float (32 bit real) from the accumulator (User Register 0) to absolute memory address *n*.
Note: recommended for factory use only.

WW — **W**rite the low **W**ord (16 bit integer) value in the accumulator to absolute memory location ‘*n*’

MCCL command: WW*n* *n* = integer

This command will copy the low Word (16 bit integer) of the accumulator (User Register 0) to absolute memory address *n*.
Note: recommended for factory use only.

Chapter Contents

Sequence Commands

MCCL	Description
DF	Do if channel ofF
DN	Do if channel oN
IB	If Below do next command
IC	If Clear, do next command
IE	If Equals do next command
IF	If channel ofF do next command
IG	If accumulator is Greater do next
IN	If channel oN do next command
IP	Interrupt on absolute Position
IR	Interrupt on Relative position
IS	If bit Set do next command
IU	If Unequal do next command
JP	JumP to command absolute
JR	Jump to command Relative
RP	RePeat
WA	WAit (time)
WE	Wait for Edge
WF	Wait for channel ofF
WI	Wait for Index
WN	Wait for channel oN
WP	Wait for absolute Position
WR	Wait for Relative position
WS	Wait for Stop
WT	Wait for Target

Chapter

12

Sequence (If/Then) Commands

DF Do if channel oFf

MCCL command: DF*x* x = Channel number
applies to: Program flow
see also: DN

Used for conditional execution of commands. If digital I/O channel *x* is "off", commands that follow on the command line or in the macro will be executed. Otherwise the rest of the command line or macro will be skipped. See the description of **Digital I/O** in the **General Purpose I/O** chapter.

```
      DF2,1MR1000                          ;If channel 2 is off move 1000
```

DN Do if channel ‘x’ is oN

MCCL command: DN*x* x = Channel number
applies to: Program flow
see also: DF

Used for conditional execution of commands. If digital I/O channel *x* is "on", commands that follow on the command line or in the macro will be executed. Otherwise the rest of the command line or macro will be skipped. See the description of **Digital I/O** in the **General Purpose I/O** chapter.

```
      DN2,1MR1000                          ;If channel 2 is off move 1000
```

IB **I**f the accumulator is **B**elow 'n', execute the next command, else skip 2 commands

MCCL command:	IBn *n* = integer or real
applies to:	Program flow
see also:	IE, IG, IU

Used for conditional execution of commands. If the contents of the accumulator (User Register 0) is less than *n*, command execution will continue with the command following the IB command. Otherwise the two commands following the IB command will be skipped, and command execution will continue from the third command.

```
IB0,MJ10,NO,MJ11                    ;If the accumulator contents is less
                                    ;than 10 jump to macro 10, otherwise
                                    ;jump to macro 11
```

IC **I**f bit 'n' of the accumulator is **C**lear (equal to 0), execute the next command, else skip 2 commands

MCCL command:	IC*n* *n* = integer >= 0, <= 31
applies to:	Program flow
see also:	IS

Used for conditional execution of commands. If the contents of the accumulator (User Register 0) has bit *n* reset, command execution will continue with the command following the IC command. Otherwise the two commands following the IC command will be skipped, and command execution will continue from the third command.

```
IC3,MJ10,NO,MJ11                    ;If accumulator bit 3 is cleared jump
                                    ;to macro 10, otherwise jump to macro
                                    ;11
```

IE **I**f the accumulator **E**quals "n", execute the next command, else skip 2 commands

MCCL command:	IE*n* *n* = integer or real
applies to:	Program flow
see also:	IB, IG, IU

Used for conditional execution of commands. If the contents of the accumulator (User Register 0) equals *n*, command execution will continue with the command following the IE command. Otherwise the two commands following the IE command will be skipped, and command execution will continue from the third command.

```
IE0,MJ10,NO,MJ11                       ;If accumulator contents equals 0 jump
                                       ;to macro 10, otherwise jump to macro
                                       ;11
```

IF — **I**f channel o**F**f do next command, else skip 2 commands

MCCL command: IF*x* x = Channel number
applies to: Program flow
see also: IN

Used for conditional execution of commands. If digital I/O channel *x* is "off", command execution will continue with the command following the IF command. Otherwise the two commands following the IF command will be skipped, and command execution will continue from the third command.

```
IF5,MJ10,NO,MJ11                   ;If digital input #5 is off jump to
                                   ;macro 10, otherwise jump to macro 11
```

IG — **I**f the accumulator is **G**reater than ‘n’ execute the next command, else skip 2 commands

MCCL command: IG*n* *n* = integer or real
applies to: Program flow
see also: IB, IE, IU

Used for conditional execution of commands. If the contents of the accumulator (User Register 0) is greater than *n*, command execution will continue with the command following the IG command. Otherwise the two commands following the IG command will be skipped, and command execution will continue from the third command. See the description of **Digital I/O** in the **General Purpose I/O** chapter.

```
IG25,MJ10,NO,MJ11                     ;If the accumulator contents is
                                      ;greater than 25 jump to macro 10,
                                      ;otherwise jump to macro 11
```

IN — **I**f channel ’ o**N** do next command, else skip 2 commands

MCCL command: IN*x* x = Channel number
applies to: Program flow
see also: IF

Used for conditional execution of commands. If digital I/O channel *x* is "on", command execution will continue with the command following the IN command. Otherwise the two commands following the IN command will be skipped, and command execution will continue from the third command.

```
IN5,MJ10,NO,MJ11                   ;If digital input #5 is on jump to
                                   ;macro 10, otherwise jump to macro 11
```

IP Interrupt on absolute Position

MCCL command: *a*IP*n* a = Axis number *n* = integer or real
applies to: Analog Command Axis, Pulse Command Axis
see also: IR, TS, WP, WR

This command is used to indicate when an axis has reached a specific position. The position is specified by parameter *n* as a relative distance from the axis home position. When the specified position has been reached, the controller will set the "breakpoint reached" flag in the motor status for that axis (which can then be used by the MCAPI to interrupt the host PC). The IP command can be issued to an axis before or after it has been commanded to move.

Note:
Open loop stepper - the setting of the Breakpoint Reached Flag is based on the number of step pulses issued to the stepper driver.
Closed loop stepper - the setting of the Breakpoint Reached Flag is based on the encoder feedback.

```
1IP1000                             ;breakpoint position = 1000
LU"STATUS",1RL@0,IC6,NO,JR-4        ;loop until breakpoint reached
```

IR Interrupt upon reaching Relative position

MCCL command: *a*IR*n* a = Axis number *n* = integer or real
applies to: Analog Command Axis, Pulse Command Axis
see also: IR, TS, WP, WR

This command is used to indicate when an axis has reached a specific position. The position is specified by parameter *n* as a relative distance from the target position established by the last motion command. When the specified position has been reached, the controller will set the "breakpoint reached" flag in the status for that axis (which can then be used by the MCAPI to interrupt the host PC). The IR command can be issued to an axis before or after it has been commanded to move.

Note:
Open loop stepper - the setting of the Breakpoint Reached Flag is based on the number of step pulses issued to the stepper driver.
Closed loop stepper - the setting of the Breakpoint Reached Flag is based on the encoder feedback.

```
1IR-1000                            ;breakpoint position = -1000 from
                                    ;target
LU"STATUS",1RL@0,IC6,NO,JR-4        ;loop until breakpoint reached
```

IS **I**f bit ‘n’ of the accumulator is **S**et execute the next command, else skip 2 commands

MCCL command:	IS*n* *n* = integer >= 0, <= 31
applies to:	Program flow
see also:	IC

Used for conditional execution of commands. If the contents of the accumulator (User Register 0) has bit *n* set, command execution will continue with the command following the IS command. Otherwise the two commands following the IS command will be skipped, and command execution will continue from the third command.

```
IS3,MJ10,NO,MJ11                        ;If accumulator bit 3 is set jump to
                                        ;macro 10, otherwise jump to macro 11
```

IU **I**f the accumulator is **U**nequal to “n” execute the next command, else skip 2 commands

MCCL command:	IU*n* *n* = integer or real
applies to:	Program flow
see also:	IB, IE, IG

Used for conditional execution of commands. If the contents of the accumulator (User Register 0) does not equal *n*, command execution will continue with the command following the IU command. Otherwise the two commands following the IU command will be skipped, and command execution will continue from the third command.

```
IU0,MJ10,NO,MJ11                        ;If accumulator contents is unequal to
                                        ;0 jump to macro 10, otherwise jump to
                                        ;macro 11
```

JP **J**um**P** to command absolute

MCCL command:	JP*n* *n* = integer
applies to:	Program flow
see also:	JR

Jumps to the specified command in the current command string or macro. Commands are numbered consecutively starting with 0.

```
IE0,JP5,NO,1MR1000,1WS,1MR2000,1WS      ;If accumulator equals 0 jump to
                                        ;1MR2000 command
```

JR **J**ump to command **R**elative

MCCL command: JRn *n* = integer
applies to: Program flow
see also: JP

Jumps forward or backward by *n* commands in the current command string or macro. Specifying a positive value will cause a forward jump in the command string or macro. Specifying a negative value will cause a backward jump. A jump of relative 0 will cause the command to jump to itself.

```
1MR1000,LU"STATUS",1RL@0,IC3,JR-3,BK       ;If trajectory not complete (bit 3)
                                           ;loop back to LU"STATUS" command
```

RP **R**e**P**eat

MCCL command: RPn *n* = integer > = 0, < = 2,147,483,647
applies to: Program flow
see also:

This command causes all the commands preceding the RP command to be executed *n* + 1 times. If *n* is not specified or is 0 then the commands are repeated indefinitely. Note - There can be only one RP command in a command string or macro.

```
TP,RP999                                ;Display the position of axis #1, 1000
                                        ;times
```

WA **WA**it

MCCL command: WAn *n* = integer or real >= 0
applies to: Program flow
see also: WS, WT

Insert a wait period of *n* seconds before going on to the next command. If this command was issued from an ASCII interface, it can be aborted by sending an Escape character.

```
1TP,WA0.1,RP9                           ;Display the position of axis #1, 10
                                        ;times with a delay of one tenth of a
                                        ;second between displays
```

WE Wait for Edge

MCCL command:	*a*WE	*a* = Axis number
applies to:	Pulse Command Axis (Open Loop)	
see also:	EL, FE	

This command is used in conjunction with the Edge Latch command to home an open loop stepper. When combined, the EL and WE commands perform the same operation as Find Edge (FE) without the negative side effect of possibly stalling the command interpreter (in the event that the edge is never captured) . The WE command should not be issued until:

1) The motor is moving towards the home sensor mark
2) The EL command has been issued
3) The home sensor has been captured (status bit 18 - Home found is set)

After the Edge Latch command is issued, when the home sensor activates, the step count position of the sensor will be captured and the Home found status bit (status bit 18) will be set. The Wait for Edge and Motor oN commands are then issued to define the location of the home sensor as position *n*. See the description of the **E**dge **L**atch (EL) command.

WF Wait for digital channel oFf

MCCL command:	WF*x*	x = Channel number
applies to:	Digital I/O	
see also:	WN	

Wait until digital I/O channel x is "off" before continuing to the next command on the command line or in the macro. If this command was issued from an ASCII interface, it can be aborted by sending an Escape character.

```
WF2,1MR1000
```

WI Wait for encoder Index mark

MCCL command:	*a*WI*n*	*a* = Axis number	*n* = integer or real >= 0
applies to:	Analog Command Axis, Pulse Command Axis (Closed Loop)		
see also:	FI, IA		

This command is used in conjunction with the Index Arm (IA) command to home a closed loop axis. When combined, the IA and WI commands perform the same operation as Find Index (FI) without the negative side effect of possibly stalling the command interpreter (in the event that the edge is never captured). The WI command should not be issued until:

1) The motor is moving towards the index mark
2) The IA command has been issued
3) The Index mark has been captured (status bit 18 - Index found is set)

After the Index Arm command is issued, when the index pulse occurs, the encoder position of the index mark will be captured and the Encoder Index status bit (status bit 18) will be set. The Wait for Index and Motor oN commands are then issued to define the location of the index pulse as position *n*. See the description of the **I**ndex **A**rm (IA) command.

WN **W**ait for digital channel o**N**

MCCL command:	WN*x* x = Channel number
applies to:	Digital I/O
see also:	WF

Wait until digital I/O channel x is "on" before continuing to the next command on the command line or in the macro. If this command was issued from an ASCII interface, it can be aborted by sending an Escape character.

```
WN2,1MR1000
```

WP **W**ait for absolute **P**osition

MCCL command:	*a*WP*n* a = Axis number *n* = integer or real
applies to:	Analog Command Axis, Pulse Command Axis
see also:	IP, IR, WR

This command is used to delay command execution until axis *a* has reached a specific position. The position is specified by the command parameter as a relative distance from the home position of the axis. When the specified position has been reached, the controller will set the "breakpoint reached" flag in the status for that axis, and then continue execution of commands following WP. The WP command will typically be issued to an axis after it has been commanded to move. If this command was issued from an ASCII interface, it can be aborted by sending an Escape character.

Note:
Open loop stepper - the setting of the Breakpoint Reached Flag is based on the number of step pulses issued to the stepper driver.
Closed loop stepper - the setting of the Breakpoint Reached Flag is based on the encoder feedback.

```
1MA1000,1WP500,CN2
```

WR **W**ait for **R**elative position

MCCL command:	*a*WR*n* *n* = integer or real
applies to:	Analog Command Axis, Pulse Command Axis
see also:	IP, IR, WP

This command is used to delay command execution until axis *a* has reached a specific position. The position is specified by the command parameter as a relative distance from the target position established by the last motion command. When the specified position has been reached, the controller will set the "breakpoint reached" flag in the status for that axis, and then continue execution of commands following WR. The WR command will typically be issued to an axis after it has been commanded to move. If this command was issued from an ASCII interface, it can be aborted by sending an Escape character.

Note:
Open loop stepper - the setting of the Breakpoint Reached Flag is based on the number of step pulses issued to the stepper driver.
Closed loop stepper - the setting of the Breakpoint Reached Flag is based on the encoder feedback.

```
        1MR1000,1WP-500,CN2
```

WS **W**ait for **S**top

MCCL command:	*a*WS*n* *n* = integer or real
applies to:	Analog Command Axis, Pulse Command Axis
see also:	WA, WT

Will delay execution of the next command in the sequence until the Trajectory Complete status bit (bit 3) for axis *a* (or all axes if axis specifier a = 0) has been set. The command parameter *n* specifies an additional time period (in seconds) that the controller will wait before continuing execution of the commands following WS (allowing for settling time of a closed loop servo).

Closed Loop Servo axis - The trajectory complete bit of a closed loop servo will be set when the **calculated (Optimal) position = the Target position**.

Closed Loop Stepper axis - The trajectory complete bit of a closed loop stepper will be set when the **Encoder position = the Target position**.

Open Loop Stepper axis - The trajectory complete bit of an open loop stepper will be set when the **Step count position = the Target position**.

```
        3MR1000,WS0.1,MR-1000                 ;Perform a forward then backward
                                              ;motion sequence
```

comment: If the WS command was not used in the above example, there would be no motion of the axis. The reason being that the target position would simply be changed twice. The computer would add 1000 counts to the target position then subtract the same amount. This would take place far quicker than the axis could begin moving.

WT **W**ait for **T**arget

MCCL command:	*a*WT*n* *n* = integer or real
applies to:	Analog Command Axis, Pulse Command Axis
see also:	DB, DT, WS

This command will delay additional command execution until the 'At Target' status bit has been set.

Servo axis - The conditions for a closed loop axis servo to have reached its' target, are that it remains within the position **D**ead**B**and range ***n*** for the time period specified by the **D**elay at **T**arget parameter '***n***'. If parameter *n* is non zero an additional dwell time = *n* will delay the setting of the 'At Target' status bit. This additional time (or dwell) period is typically used to allow the for additional settling of the servo.

Stepper axis - The 'At Target' status bit will be set when the position of the axis is equal to the move target. The DeadBand (DB) and Delay at Target (DT) are not supported for a pulse command axis. the

Note: The Wait for Target command should not be used for axes in contour mode.

```
        2DB5,2DT0.01                            ;deadband range = +/- 5, at target
                                                ;timer = 0.01 seconds
        3MR1000,WT0.1,MR-1000                   ;Perform a forward then backward
                                                ;motion sequence
```

comment: If the WT command was not used in the above example, there would be no motion of the axis. The reason being that the target position would simply be changed twice. The computer would add 1000 counts to the target position then subtract the same amount. This would take place far quicker than the axis could begin moving.

Chapter Contents

Miscellaneous Commands

MCCL	Description
DM	Decimal Mode
DW	Disable Watchdog
EF	Echo ofF
EN	Echo oN
FD	Format text with Doubles
FM	Free Memory
FT	Format Text with Integers
HE	HElp
HM	Hexadecimal Mode
ME	MEmory allocate
NO	No Operation
OD	Output text with Doubles
OT	Output Text with integers
PC	set Prompt Character
RT	ReseT system

Chapter 13

Miscellaneous Commands

DM Decimal Mode

MCCL command: `DM`
applies to: Other
see also: HM

Input and output numbers in decimal format.
comment: The Decimal Mode command must be "executed" by the controller before commands can be issued with decimal formatted parameters. The Decimal Mode (DM) and Hexadecimal Mode (HM) commands cannot be in the same command string.

```
;Axis #1 status = Trajectory complete (bit 3) = 1
;                 Motor on (bit 1) = 1
DM                                   ;decimal mode
1TS
   01   10                           ;controller reply, axis status = 10d

HM                                   ;hexadecimal mode
1TS
   01   0000000a                     ;controller reply, axis status = Ah
```

DW Disable Watchdog

MCCL command: `DM`
applies to: Other
see also:

Disable the processor watchdog circuit.
comment: This command is reserved for factory use only.

EF Echo oFf

MCCL command: EFn   *n* = 0, 1, 2, or 3

applies to:	Other
see also:	EN

comment: This command is reserved for factory use only.

EN Echo oN

MCCL command:	EN*n* *n* = 0, 1, 2, or 3
applies to:	Other
see also:	EF

comment: This command is reserved for factory use only.

FM Free Memory

MCCL command:	FMn *n* = integer >0, <= 65536
applies to:	Other
see also:	ME

Returns the memory space allocated by the ME command to the ‘heap’. Parameter *n* **must equal** the value (memory address) that was loaded into the accumulator after successful execution of the MEmory (ME) allocate command.

FD Format text with Doubles

MCCL command:	ODs s = *string parameter*
applies to:	Other
see also:	FT, OD, OT

This command outputs a message string and double precision values to the communication interface that issued the OD command. For additional information please refer to the description of **Outputting Formatted Message Strings** in the **chapter 4**.

FT Format Text with integers

MCCL command:	OTs s = *string parameter*
applies to:	Other
see also:	FD, OD, OT

This command places a formatted message string and integer values into controller memory. Upon completion of this command the memory address where the formatted message is stored is available in the accumulator (register 0). For additional information please refer to the description of **Outputting Formatted Message Strings** in the **Working with MCCL Commands chapter**

HE display the supported MCCL commands

MCCL command: HE__ 0, 1, *mm* = *string parameter*
applies to: Other
see also:

Reports the valid controller command mnemonics for the installed software version. Issuing the HElp command with string parameter *"mm"* equal to a valid command mnemonic will cause the controller to display additional information about that MCCL command.

Parameter	**Description**
0	Report supported MCCL commands
1	Report Data Table variable names
"mm"	Report the description of the MCCL command (where *mm* = valid MCCL command mnemonic)

HM **H**exadecimal **M**ode

MCCL command: HM
applies to: Other
see also: DM

Input and output numbers in hexadecimal format.
comment: The Hexadecimal Mode command must be executed by the controller before commands can be issued with hexadecimal formatted parameters. The Hexadecimal Mode (HM) and Decimal Mode (DM) and commands cannot be in the same command string. If a command parameter is to be entered in hexadecimal format, and the number starts with either A, B, C, D, E, or F, it must be preceded by a '0' (zero).

```
;Axis #1 status = Trajectory complete (bit 3) = 1
;                 Motor on (bit 1) = 1
DM                                  ;decimal mode
1TS
   01   10                          ;controller reply, axis status = 10d

HM                                  ;hexadecimal mode
1TS
   01   0000000a                    ;controller reply, axis status = Ah
```

ME **ME**mory allocate

MCCL command:MEn *n* = integer >0, <= 65536

applies to: Other
see also: FM

Formats and allocates scratch pad memory. Parameter *n* equals the requested amount of memory to be allocated. Upon successful execution the first allocated memory address will be loaded into the accumulator. If the requested amount of memory is not available the accumulator will be loaded with a zero.

NO No Operation

MCCL command: NO
applies to: Program flow
see also: IB, IC, IG, IS, IU

This command does nothing. It can be used to cause short delays in command line executions or as a filler in sequence commands.

```
IN5,BK,NO,MJ11                          ;If digital input #5 is on jump to
                                        ;macro 10, otherwise jump to macro 11
```

OD Output text with Doubles

MCCL command: ODs s = *string parameter*
applies to: Other
see also: FT, OT

This command outputs a message string and double precision values to the communication interface that issued the OD command. For additional information please refer to the description of **Outputting Formatted Message Strings** in the **chapter 4**.

```
LU"POSITION"1RD@0,OD”The current position of Axis #1 %f \n”
                                        ;load the accumulator with the
                                        ;position of axis #1. Output a text
                                        ;message displaying the position of
                                        ;axis #1 (floating point value),
                                        ;carriage return
```

OT Output Text with integers

MCCL command: OTs s = *string parameter*
applies to: Other
see also: FD, OD

This command outputs a message string and integer values to the communication interface that issued the OT command. For additional information please refer to the description of **Outputting Formatted Message Strings** in the **chapter 4**.

This command places a formatted message string and integer values into controller memory. Upon completion of this command the memory address where the formatted message is stored is available in the accumulator (register 0). For additional information please refer to the description of **Outputting Formatted Message Strings** in the **Working with MCCL Commands chapter**.

```
;register 200 stores the part count
al0,ar200                               ;initialize register 200 to 0

MC10,MC20,MC30,MC100                    ;run part program

MD100,al@200,OT”The part count = %d \n”
                                        ;load the accumulator with the
                                        ;part count (register 200).
```

PC Prompt Character

MCCL command: PCn *n* = integer >0, <= 255
applies to: Other
see also:

This command sets the character that will be sent out an ASCII command port when the controller completes execution of a command issued to that port. The parameter to this command is the ASCII code for the character. Issuing the command with a parameter of 0 will inhibit any character from being sent. The default prompt character is '>' (ASCII 62 decimal).

RT ReseT

MCCL command: *a*RT *a* = Axis number (0 resets the entire controller)
applies to: Analog Command Axis, Pulse Command Axis, Digital I/O
see also:

Performs a reset of the entire controller or reinitializes a specific axis. If an axis number is specified when the command is issued, just that axis will be reset. If no axis is specified, the entire controller will be reset. When an axis is reset, the default conditions such as acceleration and velocity will be restored, and the axes will be placed in the "off" state.

Chapter Contents

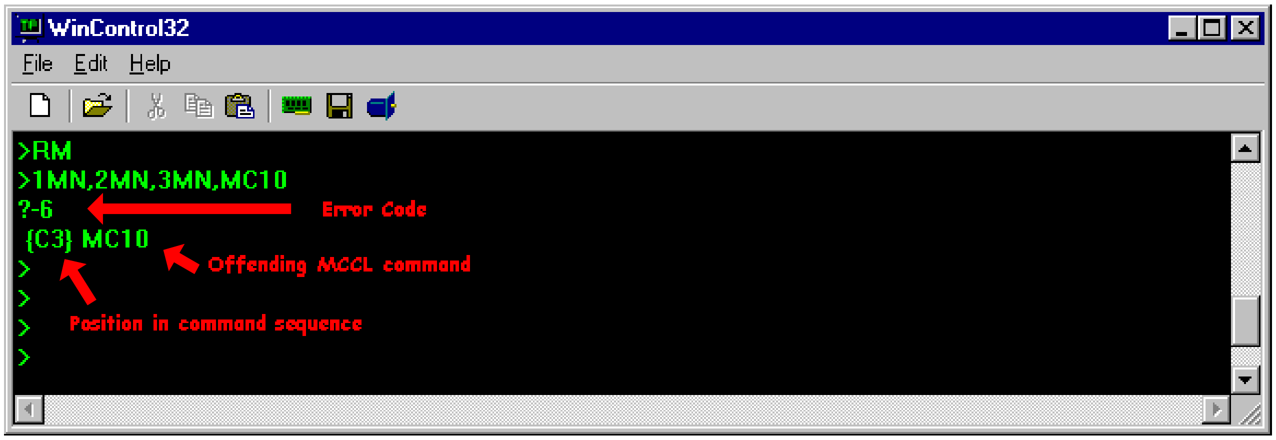
Chapter 14

Controller Error Codes

When executing MCCL (Motion Control Command Language) command sequences the command interpreter will report the following error code when appropriate:

Description	**Error code**
No error	0
Unrecognized command	1
Bad command format	2
I/O error	3
Command string to long	4
Command Parameter Error	-1
Command Code Invalid	-2
Negative Repeat Count	-3
Macro Define Command Not First	-4
Macro Number Out of Range	-5
Macro Doesn't Exist	-6
Command Canceled by User	-7
Contour Path Command Not First	-8
Contour Path Command Parameter Invalid	-9
Contour Path Command Doesn't Specify an AXIS	-10
Axis error (over travel error, max. following error exceeded	-13
No axis specified	-14
Axis not assigned	-15
Axis already assigned	-16
Axis duplicate assigned	-17
Insufficient memory	-18
Unrecognized variable name	-19
Invalid background task ID	-20
Command not supported	-21

Many error code reports will not only include the error code but also the offending command. In the following example the Reset Macro command was issued. This command clears all macro's from memory. The next command sequence turns on 3 motors and then calls macro 10. The command MC10 is a valid command but with no macros in memory error code –6 is displayed.



```
>RM
>1MN,2MN,3MN,MC10
?-6
 {C3} MC10
>
>
>
>
```

Index

Precision MicroControl Corporation
2075-N Corte del Nogal
Carlsbad, CA 92009 * USA

Tel: +1-760-930-0101
Fax: +1-760-930-0222

www.pmccorp.com

Information: info@pmccorp.com
Technical Support: support@pmccorp.com