PMC

Motion Control Command Language


Revision 3.0
LIMITED WARRANTY

All products manufactured by PRECISION MICROCONTROL CORPORATION are guaranteed to be free from defects in material and workmanship, for a period of five years from the date of shipment. Liability is limited to FOB Factory repair, or replacement, of the product. Other products supplied as part of the system carry the warranty of the manufacturer.

PRECISION MICROCONTROL CORPORATION does not assume any liability for improper use or installation or consequential damage.

© Copyright Precision Micro Control Corporation, 2004-2009. All rights reserved.

Information in this document is subject to change without notice.

Intel is a registered trademark of Intel Corporation.
Microsoft® and Windows® are registered trademarks of Microsoft Corporation.
Acrobat® and Acrobat Reader® are registered trademarks of Adobe Corporation.
# Table of Contents

Table of Contents ................................................................................................................................................... i  
Introduction............................................................................................................................................................ 3  
Who should use MCCL commands? ................................................................................................................... 4  
WinControl – the MCCL command interface utility .............................................................................................. 5  
**Controller Operation Basics** ............................................................................................................................. 9  
MCCL Command Syntax ..................................................................................................................................... 9  
Simple moves with MCCL commands ............................................................................................................... 12  
**MCCL Command Quick Reference Tables** .......................................................... 15  
**Working with MCCL Commands** .................................................................................................................... 19  
MCCL On-line Help ............................................................................................................................................ 19  
Downloading MCCL Text Files ............................................................................................................................ 20  
Building MCCL Macro Sequences ....................................................................................................................... 20  
MCCL Multi-Tasking ........................................................................................................................................... 22  
Outputting Formatted Message Strings ................................................................................................................ 25  
Reading Data from Controller Memory ................................................................................................................ 26  
Single Stepping MCCL Programs ......................................................................................................................... 29  
Controller User Registers .................................................................................................................................... 30  
MultiFlex Scratch Pad Memory ............................................................................................................................ 32  
**Setup Commands** ........................................................................................................................................... 35  
**Mode Commands** ........................................................................................................................................... 57  
**Motion Commands** ......................................................................................................................................... 61  
**Reporting Commands** ..................................................................................................................................... 81  
**I/O Commands** ................................................................................................................................................. 101  
Macro and Multi-tasking Commands ..................................................................................................................... 109  
**Register Commands** ....................................................................................................................................... 115  
Sequence (If/Then) Commands ............................................................................................................................ 125  
**Miscellaneous Commands** ................................................................................................................................ 137  
**Controller Error Codes** .................................................................................................................................. 143  
Index ..................................................................................................................................................................... 147
## 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** (Motion Control Command Language) 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
IMR1000 <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
Introduction

- 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.

![Image of WinControl utility](image1.png)

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.

![Image of Start menu with WinControl selected](image2.png)

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

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 Tell Position (1TP <Enter>).

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

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

![Figure 4: Download a MCCL text file to the controller](image)
Chapter Contents

- MCCL command syntax
- Simple moves with MCCL commands
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:

\[ \text{MA} \ (\text{Move Absolute}). \]

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

\[ 1\text{MA} \ (\text{axis #1 Move Absolute}) \]

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.

\[ 0\text{MA} \ (\text{all axes Move Absolute}) \]

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

\[ 1\text{MA}10.25 \ (\text{axis #1 Move Absolute to position 10.25}) \]

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:** \( \text{aMAn} \) 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.

![WinControl](image)

**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 **Tell Position** (aTP) command to axis number one.
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 (aMN) 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 Tell Target (aTT) command will report the position at which the axis should be and the Tell Position (aTP) command will report the current actual position of the axis. There are several commands that can used to begin motion, including Move Absolute (MA) and Move Relative (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 RePeat (RPn) command at the end causes the previous command to be repeated 6 additional times. The Wait for Stop (aWSn) 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

- Setup Commands
- Motion Commands
- Mode Commands
- Reporting Commands
- Register Commands
- I/O Commands
- Macro Commands
- Sequence Commands
- Miscellaneous Commands
## Setup Commands

<table>
<thead>
<tr>
<th>MCCL</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG</td>
<td>E2h</td>
<td>set Acceleration feed-forward Gain</td>
</tr>
<tr>
<td>AH</td>
<td>EAh</td>
<td>Auxiliary encoder define Home</td>
</tr>
<tr>
<td>BD</td>
<td>D0h</td>
<td>Backlash compensation Distance</td>
</tr>
<tr>
<td>DB</td>
<td>76h</td>
<td>set position DeadBand</td>
</tr>
<tr>
<td>DG</td>
<td>E3h</td>
<td>set Deceleration feed-forward Gain</td>
</tr>
<tr>
<td>DH</td>
<td>23h</td>
<td>Define Home</td>
</tr>
<tr>
<td>DI</td>
<td>24h</td>
<td>Direction</td>
</tr>
<tr>
<td>DS</td>
<td>75h</td>
<td>Deceleration Set</td>
</tr>
<tr>
<td>DT</td>
<td>C5h</td>
<td>Delay at Target</td>
</tr>
<tr>
<td>EI</td>
<td>7Ch</td>
<td>Enable Interrupts</td>
</tr>
<tr>
<td>ES</td>
<td>E5h</td>
<td>Encoder counts / Steps Scale</td>
</tr>
<tr>
<td>FC</td>
<td>40h</td>
<td>Full Current</td>
</tr>
<tr>
<td>FF</td>
<td>33h</td>
<td>amplifier Fault input ofF</td>
</tr>
<tr>
<td>FL</td>
<td>151h</td>
<td>IIR Filter Load coefficients</td>
</tr>
<tr>
<td>FN</td>
<td>32h</td>
<td>amplifier Fault input oN</td>
</tr>
<tr>
<td>FR</td>
<td>27h</td>
<td>set derivative sampling period</td>
</tr>
<tr>
<td>HC</td>
<td>41h</td>
<td>Half Current</td>
</tr>
<tr>
<td>HL</td>
<td>D3h</td>
<td>set motion High Limit</td>
</tr>
<tr>
<td>HS</td>
<td>E8h</td>
<td>set High Speed</td>
</tr>
<tr>
<td>IL</td>
<td>28h</td>
<td>set Integration Limit</td>
</tr>
<tr>
<td>IO</td>
<td>205h</td>
<td>Integral option</td>
</tr>
<tr>
<td>LA</td>
<td>330h</td>
<td>Load commutation phase A</td>
</tr>
<tr>
<td>LB</td>
<td>331h</td>
<td>load commutation phase B</td>
</tr>
<tr>
<td>LD</td>
<td>333h</td>
<td>Load commutation Divisor</td>
</tr>
<tr>
<td>LE</td>
<td>332h</td>
<td>Load commutation encoder prescale</td>
</tr>
<tr>
<td>LF</td>
<td>36h</td>
<td>motion Limits ofF</td>
</tr>
<tr>
<td>LL</td>
<td>D2h</td>
<td>set motion Low Limit</td>
</tr>
<tr>
<td>LM</td>
<td>34h</td>
<td>Limit Mode</td>
</tr>
<tr>
<td>LN</td>
<td>35h</td>
<td>motion Limits oN</td>
</tr>
<tr>
<td>LR</td>
<td>334h</td>
<td>Load commutation repeat count</td>
</tr>
<tr>
<td>LS</td>
<td>36h</td>
<td>set Low Speed</td>
</tr>
<tr>
<td>MS</td>
<td>E7h</td>
<td>set Medium Speed</td>
</tr>
<tr>
<td>MV</td>
<td>C4h</td>
<td>set Minimum Velocity</td>
</tr>
<tr>
<td>NF</td>
<td>150h</td>
<td>No IIR Filter</td>
</tr>
<tr>
<td>OB</td>
<td>C3h</td>
<td>set the Output deadBand</td>
</tr>
<tr>
<td>OM</td>
<td>D8h</td>
<td>set Output Mode</td>
</tr>
<tr>
<td>OO</td>
<td>CBh</td>
<td>set the Output Offset</td>
</tr>
<tr>
<td>PH</td>
<td>73h</td>
<td>set servo output PHase</td>
</tr>
<tr>
<td>SA</td>
<td>28h</td>
<td>Set Acceleration</td>
</tr>
<tr>
<td>SD</td>
<td>2Ch</td>
<td>Set Derivative gain</td>
</tr>
<tr>
<td>SE</td>
<td>19h</td>
<td>Stop on Error</td>
</tr>
<tr>
<td>SG</td>
<td>2Dh</td>
<td>Set prop. Gain of motor</td>
</tr>
<tr>
<td>SI</td>
<td>2Eh</td>
<td>Set Integral gain</td>
</tr>
<tr>
<td>SQ</td>
<td>74h</td>
<td>Set TorQue</td>
</tr>
<tr>
<td>SS</td>
<td>1Dh</td>
<td>Set Slave ratio</td>
</tr>
<tr>
<td>SV</td>
<td>2Fh</td>
<td>Set Velocity</td>
</tr>
<tr>
<td>UA</td>
<td>9Ch</td>
<td>Use as default Axis</td>
</tr>
<tr>
<td>UK</td>
<td>D7h</td>
<td>set User output constant</td>
</tr>
<tr>
<td>UO</td>
<td>B3h</td>
<td>set User Offset</td>
</tr>
<tr>
<td>UP</td>
<td>9Dh</td>
<td>Use Physical axis</td>
</tr>
<tr>
<td>UR</td>
<td>B1h</td>
<td>set User Rate conversion</td>
</tr>
<tr>
<td>US</td>
<td>AFh</td>
<td>set User Scale</td>
</tr>
<tr>
<td>UT</td>
<td>B2h</td>
<td>set User Time conversion</td>
</tr>
<tr>
<td>UZ</td>
<td>B0h</td>
<td>set User Zero</td>
</tr>
<tr>
<td>VA</td>
<td>ADh</td>
<td>set Vector Acceleriation</td>
</tr>
<tr>
<td>VD</td>
<td>AEh</td>
<td>set Vector Deceleriation</td>
</tr>
<tr>
<td>VG</td>
<td>77h</td>
<td>set Velocity Gain</td>
</tr>
<tr>
<td>VO</td>
<td>E0h</td>
<td>set Velocity Override</td>
</tr>
<tr>
<td>VV</td>
<td>ACh</td>
<td>set Vector Velocity</td>
</tr>
<tr>
<td>YF</td>
<td>14Fh</td>
<td>Yes IIR Filter</td>
</tr>
<tr>
<td>ZF</td>
<td>153h</td>
<td>Zero IIR Filter</td>
</tr>
</tbody>
</table>

## Setup Commands continued

<table>
<thead>
<tr>
<th>MCCL</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MS</td>
<td>E7h</td>
<td>set Medium Speed</td>
</tr>
<tr>
<td>MV</td>
<td>C4h</td>
<td>set Minimum Velocity</td>
</tr>
<tr>
<td>NF</td>
<td>150h</td>
<td>No IIR Filter</td>
</tr>
<tr>
<td>OB</td>
<td>C3h</td>
<td>set the Output deadBand</td>
</tr>
<tr>
<td>OM</td>
<td>D8h</td>
<td>set Output Mode</td>
</tr>
<tr>
<td>OO</td>
<td>CBh</td>
<td>set the Output Offset</td>
</tr>
<tr>
<td>PH</td>
<td>73h</td>
<td>set servo output PHase</td>
</tr>
<tr>
<td>SA</td>
<td>28h</td>
<td>Set Acceleration</td>
</tr>
<tr>
<td>SD</td>
<td>2Ch</td>
<td>Set Derivative gain</td>
</tr>
<tr>
<td>SE</td>
<td>19h</td>
<td>Stop on Error</td>
</tr>
<tr>
<td>SG</td>
<td>2Dh</td>
<td>Set prop. Gain of motor</td>
</tr>
<tr>
<td>SI</td>
<td>2Eh</td>
<td>Set Integral gain</td>
</tr>
<tr>
<td>SQ</td>
<td>74h</td>
<td>Set TorQue</td>
</tr>
<tr>
<td>SS</td>
<td>1Dh</td>
<td>Set Slave ratio</td>
</tr>
<tr>
<td>SV</td>
<td>2Fh</td>
<td>Set Velocity</td>
</tr>
<tr>
<td>UA</td>
<td>9Ch</td>
<td>Use as default Axis</td>
</tr>
<tr>
<td>UK</td>
<td>D7h</td>
<td>set User output constant</td>
</tr>
<tr>
<td>UO</td>
<td>B3h</td>
<td>set User Offset</td>
</tr>
<tr>
<td>UP</td>
<td>9Dh</td>
<td>Use Physical axis</td>
</tr>
<tr>
<td>UR</td>
<td>B1h</td>
<td>set User Rate conversion</td>
</tr>
<tr>
<td>US</td>
<td>AFh</td>
<td>set User Scale</td>
</tr>
<tr>
<td>UT</td>
<td>B2h</td>
<td>set User Time conversion</td>
</tr>
<tr>
<td>UZ</td>
<td>B0h</td>
<td>set User Zero</td>
</tr>
<tr>
<td>VA</td>
<td>ADh</td>
<td>set Vector Acceleriation</td>
</tr>
<tr>
<td>VD</td>
<td>AEh</td>
<td>set Vector Deceleriation</td>
</tr>
<tr>
<td>VG</td>
<td>77h</td>
<td>set Velocity Gain</td>
</tr>
<tr>
<td>VO</td>
<td>E0h</td>
<td>set Velocity Override</td>
</tr>
<tr>
<td>VV</td>
<td>ACh</td>
<td>set Vector Velocity</td>
</tr>
<tr>
<td>YF</td>
<td>14Fh</td>
<td>Yes IIR Filter</td>
</tr>
<tr>
<td>ZF</td>
<td>153h</td>
<td>Zero IIR Filter</td>
</tr>
</tbody>
</table>
### Motion Commands

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

### Mode Commands

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

### Reporting Commands

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

---

Precision MicroControl Corp.
## Register Commands

<table>
<thead>
<tr>
<th>MCCL</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>85h</td>
<td>Accumulator Add</td>
</tr>
<tr>
<td>AC</td>
<td>8Ch</td>
<td>Accumulator Complement</td>
</tr>
<tr>
<td>AD</td>
<td>88h</td>
<td>Accumulator Divide</td>
</tr>
<tr>
<td>AE</td>
<td>8Fh</td>
<td>Accumulator logical Exclusive or</td>
</tr>
<tr>
<td>AL</td>
<td>82h</td>
<td>Accumulator Load</td>
</tr>
<tr>
<td>AM</td>
<td>87h</td>
<td>Accumulator Multiply</td>
</tr>
<tr>
<td>AN</td>
<td>80h</td>
<td>Accumulator logical AND with n,</td>
</tr>
<tr>
<td>AO</td>
<td>83h</td>
<td>Accumulator logical OR with n</td>
</tr>
<tr>
<td>AR</td>
<td>84h</td>
<td>copy Accumulator to Register n</td>
</tr>
<tr>
<td>AS</td>
<td>86h</td>
<td>Accumulator Subtract</td>
</tr>
<tr>
<td>AV</td>
<td>88h</td>
<td>Accumulator eValuate</td>
</tr>
<tr>
<td>AX</td>
<td>E1h</td>
<td>get Aux. Index position</td>
</tr>
<tr>
<td>GA</td>
<td>8Fh</td>
<td>Get Analog value</td>
</tr>
<tr>
<td>GF</td>
<td>152h</td>
<td>Get IIR Filter coefficients</td>
</tr>
<tr>
<td>GU</td>
<td>89h</td>
<td>Get the default axis</td>
</tr>
<tr>
<td>GX</td>
<td>F7h</td>
<td>Get auxiliary encoder position</td>
</tr>
<tr>
<td>LU</td>
<td>81h</td>
<td>Look Up motor table variable</td>
</tr>
<tr>
<td>OA</td>
<td>98h</td>
<td>Output Analog value</td>
</tr>
<tr>
<td>RA</td>
<td>96h</td>
<td>Read Byte into accumulator</td>
</tr>
<tr>
<td>RB</td>
<td>93h</td>
<td>Read Double into accumulator</td>
</tr>
<tr>
<td>RL</td>
<td>98h</td>
<td>Read Long into accumulator</td>
</tr>
<tr>
<td>RV</td>
<td>92h</td>
<td>Read Word into accumulator</td>
</tr>
<tr>
<td>SL</td>
<td>00h</td>
<td>Shift Left accumulator n bits</td>
</tr>
<tr>
<td>SR</td>
<td>91h</td>
<td>Shift Right accumulator n bits</td>
</tr>
<tr>
<td>TR</td>
<td>57h</td>
<td>Tell contents of Register n</td>
</tr>
<tr>
<td>WB</td>
<td>99h</td>
<td>Write accumulator Byte to n</td>
</tr>
<tr>
<td>WD</td>
<td>95h</td>
<td>Write accumulator double to n</td>
</tr>
<tr>
<td>WL</td>
<td>98h</td>
<td>Write accumulator Long to n</td>
</tr>
<tr>
<td>WV</td>
<td>94h</td>
<td>Write accumulator float to n</td>
</tr>
<tr>
<td>WW</td>
<td>9Ah</td>
<td>Write accumulator Word to n</td>
</tr>
</tbody>
</table>

## Macro Commands

<table>
<thead>
<tr>
<th>MCCL</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BK</td>
<td>79h</td>
<td>Break</td>
</tr>
<tr>
<td>ET</td>
<td>F8h</td>
<td>Escape Task</td>
</tr>
<tr>
<td>GT</td>
<td>F9h</td>
<td>Generate Task</td>
</tr>
<tr>
<td>MC</td>
<td>2h</td>
<td>Macro Call</td>
</tr>
<tr>
<td>MD</td>
<td>3h</td>
<td>Macro Definition</td>
</tr>
<tr>
<td>MJ</td>
<td>5h</td>
<td>Macro Jump</td>
</tr>
<tr>
<td>NP</td>
<td>4h</td>
<td>New Priority</td>
</tr>
<tr>
<td>RM</td>
<td>4h</td>
<td>Reset Macros</td>
</tr>
<tr>
<td>TM</td>
<td>51h</td>
<td>Tell Macros</td>
</tr>
</tbody>
</table>

## Sequence Commands

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

## Miscellaneous Commands

<table>
<thead>
<tr>
<th>MCCL</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DM</td>
<td>3Ch</td>
<td>Decimal Mode</td>
</tr>
<tr>
<td>DW</td>
<td>FDh</td>
<td>Disable Watchdog</td>
</tr>
<tr>
<td>EF</td>
<td>25h</td>
<td>Echo off</td>
</tr>
<tr>
<td>EN</td>
<td>26h</td>
<td>Echo on</td>
</tr>
<tr>
<td>FD</td>
<td>CCh</td>
<td>Format text with Doubles</td>
</tr>
<tr>
<td>FM</td>
<td></td>
<td>Free memory</td>
</tr>
<tr>
<td>FT</td>
<td></td>
<td>Format Text with Integers</td>
</tr>
<tr>
<td>HM</td>
<td>3Dh</td>
<td>Hexadecimal Mode</td>
</tr>
<tr>
<td>ME</td>
<td></td>
<td>MEmory allocate</td>
</tr>
<tr>
<td>NO</td>
<td>78h</td>
<td>No Operation</td>
</tr>
<tr>
<td>OD</td>
<td>C3h</td>
<td>Output text with Doubles</td>
</tr>
<tr>
<td>OT</td>
<td></td>
<td>Output Text with integers</td>
</tr>
<tr>
<td>PC</td>
<td>80h</td>
<td>set Prompt Character</td>
</tr>
<tr>
<td>RT</td>
<td>2Ah</td>
<td>Rest system</td>
</tr>
</tbody>
</table>

## I/O Commands

<table>
<thead>
<tr>
<th>MCCL</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CF</td>
<td>1Fh</td>
<td>Channel off</td>
</tr>
<tr>
<td>CH</td>
<td>42h</td>
<td>Channel High true logic</td>
</tr>
<tr>
<td>CL</td>
<td>43h</td>
<td>Channel Low true logic</td>
</tr>
<tr>
<td>CN</td>
<td>21h</td>
<td>Channel on</td>
</tr>
<tr>
<td>GA</td>
<td>F8h</td>
<td>Get Analog</td>
</tr>
<tr>
<td>OA</td>
<td>F9h</td>
<td>Output Analog</td>
</tr>
<tr>
<td>TA</td>
<td>49h</td>
<td>Tell the value of Analog input</td>
</tr>
<tr>
<td>TC</td>
<td>4Ah</td>
<td>Tell state of digital Channel</td>
</tr>
<tr>
<td>WF</td>
<td>67</td>
<td>Wait for channel off</td>
</tr>
<tr>
<td>WN</td>
<td>68</td>
<td>Wait for channel on</td>
</tr>
</tbody>
</table>
Chapter Contents

- MCCL On-line Help
- Downloading MCCL Text Files
- Building MCCL Macro Sequences
- MCCL Multi-Tasking
- Outputting Formatted Messages Strings
- PLC I/O Control using MCCL Sequence Commands
- PLC Control and Analog I/O
- MCCL Command Quick Reference Tables
- Reading Data from Memory
- Single Stepping MCCL Programs
- User Registers
- Scratch Pad Memory
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).

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 (MDn) 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.

![Figure 7: Downloading a MCCL text file via WinControl](image)

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 Tell Macro (Tmn) 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.

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 Reset Macro (RMn) 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 Reset Macro (RMn) 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 Generate Task (GTn) 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,JO-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.
Working with MCCL Commands

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 **BreaK (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,ALL,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 **Escape Task (ETn)** 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 **Information Display** command with parameter n = 7 (ID7) will report the task ID of all active background tasks.

```
;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
```

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 **Look Up (LUs)** 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.

- **aRBn** Read Byte (8 bit) at memory location `n` into accumulator \((\text{ACC} = (n))\)
- **aRDn** Read Double at memory location `n` into accumulator \((\text{ACC} = (n))\)
- **aRLn** Read Long (32 bit) at memory location `n` into accumulator \((\text{ACC} = (n))\)
- **aRVn** Read float at memory location `n` into accumulator \((\text{ACC} = (n))\)
- **aRWn** Read Word (16 bit) at memory location `n` into accumulator \((\text{ACC} = (n))\)

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)

<table>
<thead>
<tr>
<th>Motor Table Variable Description</th>
<th>Variable Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt Mask</td>
<td>INTMASK</td>
</tr>
<tr>
<td>Interrupt Pending</td>
<td>INTPEND</td>
</tr>
<tr>
<td>Motor Status – primary</td>
<td>STATUS</td>
</tr>
<tr>
<td>Motor Status - auxiliary</td>
<td>AUXSTAT</td>
</tr>
<tr>
<td>Position - Adjustment (index + Offset)</td>
<td>CNTADJ</td>
</tr>
<tr>
<td>Position – Current (raw count)</td>
<td>POSCOUNT</td>
</tr>
<tr>
<td>Position - Index Count</td>
<td>IDXCOUNT</td>
</tr>
<tr>
<td>Position - Optimal (raw count)</td>
<td>CMDCOUNT</td>
</tr>
<tr>
<td>Position – Target (raw count)</td>
<td>TGTCOUNT</td>
</tr>
</tbody>
</table>
Working with MCCL Commands

## Motor Table Variables – 64 bit floating point (double)

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

## Motor Table Variables – 32 bit floating point (float)

<table>
<thead>
<tr>
<th>Motor Table Variable Description</th>
<th>Variable Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feed Forward - Velocity Gain</td>
<td>KVEL</td>
</tr>
<tr>
<td>Feed Forward - Acceleration Gain</td>
<td>KACC</td>
</tr>
<tr>
<td>Feed Forward - Deceleration Gain</td>
<td>KDEC</td>
</tr>
<tr>
<td>Output Deadband (volts)</td>
<td>OUTDBAND</td>
</tr>
<tr>
<td>PID - Proportional Gain setting</td>
<td>KPOS</td>
</tr>
<tr>
<td>PID - Derivative Gain setting</td>
<td>KDER</td>
</tr>
<tr>
<td>PID - Integral Gain</td>
<td>KINT</td>
</tr>
<tr>
<td>PID - Integration Limit</td>
<td>ILIM</td>
</tr>
<tr>
<td>Torque Limit – max. output</td>
<td>MAXTRQ</td>
</tr>
<tr>
<td>User Offset (volts)</td>
<td>OUTOFFSET</td>
</tr>
<tr>
<td>Velocity Override</td>
<td>VELOVRD</td>
</tr>
</tbody>
</table>
Motor Table Variables – 16 bit integer (word)

<table>
<thead>
<tr>
<th>Motor Table Variable Description</th>
<th>Variable Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis Number</td>
<td>AXISNUM</td>
</tr>
<tr>
<td>IIR filter coefficients, maximum</td>
<td>COEFMAX</td>
</tr>
<tr>
<td>IIR filter coefficients, qty. defined</td>
<td>COEFCNT</td>
</tr>
<tr>
<td>Master Axis</td>
<td>MASTER</td>
</tr>
<tr>
<td>Input Mode</td>
<td>INPUTMODE</td>
</tr>
<tr>
<td>Integral Term Options</td>
<td>INTEGRALOPT</td>
</tr>
<tr>
<td>Derivative Term Sampling Frequency</td>
<td>SFREQ</td>
</tr>
<tr>
<td>Timer - Wait Stop</td>
<td>WAITSTOP</td>
</tr>
<tr>
<td>Timer - Wait Target</td>
<td>WAITTARGET</td>
</tr>
</tbody>
</table>
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:

\[
\text{MD10, WA1, 1MR1000, 1WS.1, 1TP, 1MR-1000, 1WS.1, 1TP, RP}
\]

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

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
Working with MCCL Commands

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 Accumulator Load (ALn) 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:
<table>
<thead>
<tr>
<th>MCCL Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAn</td>
<td>Accumulator Add ( (ACC = ACC + n) )</td>
</tr>
<tr>
<td>ACn</td>
<td>Accumulator Complement, bit wise ( (ACC = \neg ACC) )</td>
</tr>
<tr>
<td>ADn</td>
<td>Accumulator Divide ( (ACC = ACC/n) )</td>
</tr>
<tr>
<td>AEn</td>
<td>Accumulator logical Exclusive or with ( n ), bit wise ( (ACC = ACC \text{ eor } n) )</td>
</tr>
<tr>
<td>ALn</td>
<td>Accumulator Load with constant ( n ) ( (ACC = n) )</td>
</tr>
<tr>
<td>AMn</td>
<td>Accumulator Multiply ( (ACC = ACC \times n) )</td>
</tr>
<tr>
<td>ANn</td>
<td>Accumulator logical And with ( n ), bit wise ( (ACC = ACC \text{ and } n) )</td>
</tr>
<tr>
<td>ARn</td>
<td>copy Accumulator to Register ( n ) ( (REGn = ACC) )</td>
</tr>
<tr>
<td>ASn</td>
<td>Accumulator Subtract ( (ACC = ACC - n) )</td>
</tr>
<tr>
<td>GAx</td>
<td>Get Analog value ( (ACC = \text{ channel } x) )</td>
</tr>
<tr>
<td>aGX</td>
<td>Get auxiliary encoder position ( (ACC = \text{ axis } a \text{ auxiliary encoder}) )</td>
</tr>
<tr>
<td>IBn</td>
<td>If accumulator is Below ( (&gt; ) ) ( n ), do next command, else skip 2 commands</td>
</tr>
<tr>
<td>ICn</td>
<td>If bit ( n ) of accumulator is Clear, do next command, else skip 2 commands</td>
</tr>
<tr>
<td>IEn</td>
<td>If accumulator Equals constant ( n ), do next command, else skip 2 commands</td>
</tr>
<tr>
<td>IGn</td>
<td>If accumulator is Greater than ( 'n' ), do next command, else skip 2 commands</td>
</tr>
<tr>
<td>OAx</td>
<td>Output Analog value ( (\text{ channel } x = ACC) )</td>
</tr>
<tr>
<td>ISn</td>
<td>If bit ( n ) of accumulator is Set, do next command, else skip 2 commands</td>
</tr>
<tr>
<td>IUn</td>
<td>If accumulator is Unequal to ( 'n' ), do next command, else skip 2 commands</td>
</tr>
<tr>
<td>RAn</td>
<td>copy Register ( n ) to Accumulator ( (ACC = REGn) )</td>
</tr>
<tr>
<td>SLn</td>
<td>Shift Left accumulator ( n ) bits ( (ACC = ACC \ll n) )</td>
</tr>
<tr>
<td>SRn</td>
<td>Shift Right accumulator ( n ) bits ( (ACC = ACC \gg n) )</td>
</tr>
<tr>
<td>TRn.p</td>
<td>Tell contents of Register ( n )</td>
</tr>
<tr>
<td>TR.p</td>
<td>Tell contents of accumulator ( (\text{ register } 0) )</td>
</tr>
</tbody>
</table>
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 `MEmory allocate (ME)` 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 `Free Memory (FMn)` 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.
## Setup Commands

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

## Setup Commands continued

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

**AG**  
**Acceleration Gain**

*MCCL command:* \( aAGn \quad a = \text{Axis number} \quad n = \text{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.

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

**AH**  
**Auxiliary encoder define Home**

*MCCL command:* \( aAHn \quad a = \text{Axis number} \quad n = \text{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:* \( aBDn \quad a = \text{Axis number} \quad n = \text{integer or real} \\
Setup Commands

**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 DeadBand**

**MCCL command:** aDBn  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**

**Deceleration Gain**

**MCCL command:** aDGn  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:** aDHn  \( a = \) 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 D|irection

**MCCL command:** aDIn  \( a = \) 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:** aDSn  \( a = \) 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
**Setup Commands**

- **MCCL command:** `aDTn`  
  - `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:** `aEEn`  
  - `a` = Axis number  
  - `n` = integer  
  - **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 properly terminated to the 1.5V encoder reference.

<table>
<thead>
<tr>
<th><code>n</code></th>
<th>Encoder fail Enable parameter n function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Encoder fail detection disabled</td>
</tr>
<tr>
<td>1</td>
<td>Enable encoder fail error detection</td>
</tr>
<tr>
<td>2</td>
<td>Enable encoder fail error detection for the auxiliary encoder of a servo axis (not supported at this time)</td>
</tr>
</tbody>
</table>

- **EI**  
  - **Enable bus Interrupts**

- **MCCL command:** `aElna`  
  - `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: \( aESn \quad a = \text{Axis number} \quad n = \text{integer or real} \geq 0 \\
\text{applies to:} \quad \text{Pulse Command Axis (Closed Loop)} \\
\text{see also:} \quad \text{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.

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

**FC**

**Full Current**

MCCL command: \( aFC \quad a = \text{Axis number} \\
\text{applies to:} \quad \text{Pulse Command Axis} \\
\text{see also:} \quad \text{HC}

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

\[
5FC \quad ;\text{turn on axis #5 full current output}
\]

**FF**

**amplifier Fault Off**

MCCL command: \( aFF \quad a = \text{Axis number} \\
\text{applies to:} \quad \text{Analog Command Axis, Pulse Command Axis} \\
\text{see also:} \quad \text{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.
Setup Commands

FN amplifier Fault oN

MCCL command: \( aFN \quad a = \text{Axis number} \)
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: \( aFRn \quad a = \text{Axis number} \quad n = \text{integer} \geq 0 \)
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 \( (n + 1) \times \text{sample period} \). See the loop rate commands (HS, MS, LS) for a discussion of the sample period on servos.

\[
\begin{align*}
1FR0.00025 & \quad ; \text{set derivative sampling rate for low friction} \\
2FR0.0015 & \quad ; \text{set derivative sampling rate for high inertia}
\end{align*}
\]

HC Half Current

MCCL command: \( aHC \quad a = \text{Axis number} \)
applies to: Pulse Command Axis
see also: FC

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

\[
5HC \quad ; \text{turn on axis #5 full current output}
\]

HL High motion soft Limit

MCCL command: \( aHLn \quad a = \text{Axis number} \quad n = \text{integer or real} \)
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 High Speed

**MCCL command:**  
\[ aHS \quad a = \text{Axis number} \]

**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 Integration Limit

**MCCL command:**  
\[ aILn \quad a = \text{Axis number} \quad n = \text{integer or real} \geq 0 \]

**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))
Setup Commands

IO     Integral Option

**MCCL command:** \( \text{aIO} \ n \)  \( a = \text{Axis number} \quad n = \text{integer 0, 1, or 2} \)

**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 **Integral Option** command allows the user to define how the integral term of the PID filter functions while a servo is moving.

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

```plaintext
lsi.0065 ;set integral gain
lio2 ;integral option = zero (wait until
      ;trajectory complete)
lii50.0 ;set integral limit
```

LF     motion Limits oFf

**MCCL command:** \( \text{aLF} \ n \)  \( a = \text{Axis number} \quad n = 0 – 15 \)

**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:** \( \text{aLL} \ n \)  \( a = \text{Axis number} \quad n = \text{integer or real} \)

**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)
1HL1000000              ;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**: \(aLM\_n\)  \(a = \text{Axis number} \quad n = \text{integer 0 - 136}\)

**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.

<table>
<thead>
<tr>
<th>Parameter (n) =</th>
<th>Desired action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,0 *</td>
<td>Turn motor off (disable PID) when hard limit sensor ‘goes’ active or soft motion limit is exceeded</td>
</tr>
<tr>
<td>1,4 *</td>
<td>Stop the motor abruptly (under PID control) when hard limit sensor ‘goes’ active or soft motion limit is exceeded</td>
</tr>
<tr>
<td>2,8 *</td>
<td>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.</td>
</tr>
<tr>
<td>128 **</td>
<td>Invert the active level of the hard limit input. Typically used for normally open hard limit sensors</td>
</tr>
</tbody>
</table>

* 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)
```
**Setup Commands**

**LN**  
Limits oN

*MCCL command:* \( \text{aLNn} \quad a = \text{Axis number} \quad n = 0 - 15 \)

*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 Limit Mode 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 Limit Mode 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.

<table>
<thead>
<tr>
<th>Parameter n</th>
<th>Desired action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0*</td>
<td>Enable both hard limits (+/-) and soft limits (high &amp; low)</td>
</tr>
<tr>
<td>1**</td>
<td>Enable hard limit + error checking</td>
</tr>
<tr>
<td>2**</td>
<td>Enable hard limit – error checking</td>
</tr>
<tr>
<td>3**</td>
<td>Enable hard limit + and hard limit – error checking</td>
</tr>
<tr>
<td>4**</td>
<td>Enable high soft limit error checking</td>
</tr>
<tr>
<td>8**</td>
<td>Enable low soft limit error checking</td>
</tr>
<tr>
<td>12**</td>
<td>Enable high &amp; low soft limit error checking</td>
</tr>
</tbody>
</table>

* 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:* \( \text{aLS} \quad a = \text{Axis number} \)

*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 Medium Speed

**MCCL command:**
aMS 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 Minimum Velocity

**MCCL command:**
aMVn 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 Output dead Band

**MCCL command:**
aOBn 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:*  \( aOMn \)  \( a = \) Axis number  \( n = \) integer 0, 1  

*applies to:*  Analog Command Axis, PWM Command Axis, Pulse Command Axis  

*see also:*  OD

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

<table>
<thead>
<tr>
<th>( n )</th>
<th><strong>Servo Output Mode</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Bipolar Analog output, -10V to +10V (default)</td>
</tr>
<tr>
<td>1</td>
<td>Unipolar Analog output, 0V to +10V</td>
</tr>
<tr>
<td>2</td>
<td>Bipolar PWM output</td>
</tr>
<tr>
<td>3</td>
<td>Unipolar PWM output (requires user to define PWM Direction output channel)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>( n )</th>
<th><strong>Stepper Output Mode</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pulse and Direction outputs (default)</td>
</tr>
<tr>
<td>1</td>
<td>CW and CCW Pulse Outputs</td>
</tr>
</tbody>
</table>

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:*  \( aOOn \)  \( a = \) 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.

100-0.45 ;set command output offset to -0.45 volts

**PH**  **set the servo output PHasing**

*MCCL command:*  \( aPHn \)  \( a = \) 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:** \( PKn \) \( n = \) integer \( \geq -10, \leq +10 \)

**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:** \( aSAN \) \( a = \) Axis number \( n = \) integer or real \( \geq 0 \)

**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.

```
1SV1000000 ;max. velocity = 100,000 counts or pulses / second
1SA2000000 ;accelerate to max. velocity in 0.5 seconds
1DS2000000 ;decelerate to zero velocity in 0.5 seconds
```

SD  

**Set Derivative gain**

**MCCL command:** \( aSDn \) \( a = \) Axis number \( n = \) integer or real \( \geq 0 \)

**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.
Setup Commands

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 Stop on following Error

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

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 Set Proportional gain

MCCL command: aSGn  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 Set the Integral gain

MCCL command: aSIn  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**  
**Set torQue**

**MCCL command:**  
\[ aSQn \quad a = \text{Axis number} \quad \text{Torque Mode } n = \text{integer or real } \geq -10, \leq 10 \]

**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**  
**Set the ratio of Slave axis**

**MCCL command:**  
\[ aSSn \quad a = \text{Axis number} \quad n = \text{integer or real } > 0 \leq 10 \]

**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.
**Setup Commands**

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  

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

---

**SV Set Velocity**

**MCCL command**: aSVn  
**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
  
**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 Get defaUlt 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**

**User Konstant**

*MCCL command:* \( aUK \)
\( a = \text{Axis number} \)
\( n = \text{integer or real} >= 0 \)

*applies to:* Analog Command Axis

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**

**User Offset**

*MCCL command:* \( aUOn \)
\( a = \text{Axis number} \)
\( n = \text{integer or real} \)

*applies to:* Analog Command Axis, Pulse Command Axis

*see also:* UR, US, UT, UZ

The User Offset (aUOn) 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 User Scaling command. The User Offset 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**

**User Rate**

*MCCL command:* \( aURN \)
\( a = \text{Axis number} \)
\( n = \text{integer or real} >= 0 \)

*applies to:* Analog Command Axis, Pulse Command Axis

*see also:* UO, US, UT, UZ

The User Rate (aURN) 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 is enabled / re-enabled).

3UR60
3MN

Typical User Rate values

<table>
<thead>
<tr>
<th>Time Unit</th>
<th>User Rate Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>second</td>
<td>1 (default)</td>
</tr>
<tr>
<td>minute</td>
<td>60</td>
</tr>
<tr>
<td>hour</td>
<td>3600</td>
</tr>
</tbody>
</table>

**US**  
**User Scale**

**MCCL command**:  
`aUSn`  
$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 is 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**:  
`aUTn`  
$a = $Axis number  
$n = $integer or real $\geq 0$

**applies to**:  
Analog Command Axis, Pulse Command Axis  

**see also**:  
UO, UR, UT, 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

52  

*Precision MicroControl Corp.*
**Setup Commands**

*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 User Zero position*

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

**applies to:** Analog Command Axis, Pulse Command Axis

**see also:** UO, UR, US, UT

The User Zero ($aUZn$) 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).

```
3U012.25 ;define offset to 12.25 inches
3U21.25 ;define ‘part zero’ to 1.25 inches
3MN
```

**VA**

*Vector Acceleration*

**MCCL command:** $aVAn$  
*a = Axis number  
$n =$ integer or real $\geq 0$

**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**

*Vector Deceleration*

**MCCL command:** $aVDn$  
*a = Axis number  
$n =$ integer or real $\geq 0$

**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
Setup Commands

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 -0.0009

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:** aVVn  \( a = \) 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

<table>
<thead>
<tr>
<th>MCCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM</td>
<td>enable Contour Mode (arcs and lines)</td>
</tr>
<tr>
<td>GM</td>
<td>enable Gain Mode (no velocity profile)</td>
</tr>
<tr>
<td>IM</td>
<td>Input Mode (closed loop stepper)</td>
</tr>
<tr>
<td>PM</td>
<td>enable Position Mode</td>
</tr>
<tr>
<td>SM</td>
<td>enable Master/Slave mode</td>
</tr>
<tr>
<td>QM</td>
<td>enable TorQue mode</td>
</tr>
<tr>
<td>VM</td>
<td>enable Velocity Mode</td>
</tr>
</tbody>
</table>
Mode Commands

CM

Contour Mode

**MCCL command:** \( aCMn \quad a = \text{Axis number} \quad n = \text{integer} > 0, \leq 8 \)

**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:** \( aGM \quad a = \text{Axis number} \)

**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
```
Mode Commands

**MCCL command:**  alMn  
*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:**  aPM  
*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:**  aSMn  
*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**: aQM  \( a = \) 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 Set torQue 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.

```plaintext
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.

```plaintext
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
```
### Motion Commands

<table>
<thead>
<tr>
<th>MCCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>Abort</td>
</tr>
<tr>
<td>AF</td>
<td>Auxiliary encoder arm/Find index</td>
</tr>
<tr>
<td>BC</td>
<td>Begin position Compare</td>
</tr>
<tr>
<td>BF</td>
<td>Backlash compensation oFf</td>
</tr>
<tr>
<td>BN</td>
<td>Backlash compensation oN</td>
</tr>
<tr>
<td>CA</td>
<td>arc Center Absolute</td>
</tr>
<tr>
<td>CB</td>
<td>position Capture Begin</td>
</tr>
<tr>
<td>CD</td>
<td>Contour Distance</td>
</tr>
<tr>
<td>CP</td>
<td>Contour Path</td>
</tr>
<tr>
<td>CR</td>
<td>arc Center Relative</td>
</tr>
<tr>
<td>EA</td>
<td>arc Ending Angle absolute</td>
</tr>
<tr>
<td>EL</td>
<td>home Edge Latch</td>
</tr>
<tr>
<td>ER</td>
<td>arc Ending angle Relative</td>
</tr>
<tr>
<td>FE</td>
<td>Find Edge</td>
</tr>
<tr>
<td>FI</td>
<td>Find Index</td>
</tr>
<tr>
<td>GH</td>
<td>Go Home</td>
</tr>
<tr>
<td>GO</td>
<td>GO</td>
</tr>
<tr>
<td>HO</td>
<td>HOme</td>
</tr>
<tr>
<td>IA</td>
<td>Index Arm</td>
</tr>
<tr>
<td>LC</td>
<td>Load Compare positions</td>
</tr>
</tbody>
</table>

### Motion Commands continued

<table>
<thead>
<tr>
<th>MCCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP</td>
<td>Learn Position</td>
</tr>
<tr>
<td>LT</td>
<td>Learn Target</td>
</tr>
<tr>
<td>MA</td>
<td>Move Absolute</td>
</tr>
<tr>
<td>MF</td>
<td>Motor oFf</td>
</tr>
<tr>
<td>MN</td>
<td>Motor oN</td>
</tr>
<tr>
<td>MP</td>
<td>Move to Point</td>
</tr>
<tr>
<td>MR</td>
<td>Move to Point</td>
</tr>
<tr>
<td>NC</td>
<td>Next Compare</td>
</tr>
<tr>
<td>NS</td>
<td>No Synchronization</td>
</tr>
<tr>
<td>OC</td>
<td>Output mode for Compare</td>
</tr>
<tr>
<td>OP</td>
<td>Output Period</td>
</tr>
<tr>
<td>PP</td>
<td>Profile Parabolic</td>
</tr>
<tr>
<td>PR</td>
<td>Record motion data</td>
</tr>
<tr>
<td>PS</td>
<td>Profile S-curve</td>
</tr>
<tr>
<td>PT</td>
<td>Profile Trapezoidal</td>
</tr>
<tr>
<td>RR</td>
<td>aRc Radius</td>
</tr>
<tr>
<td>SN</td>
<td>Synchronization oN</td>
</tr>
<tr>
<td>ST</td>
<td>STop</td>
</tr>
</tbody>
</table>
Motion Commands

AB

ABort motion

MCCL command: aAB \( a = \) 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: aAF \( a = \) 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.
Motion Commands

```plaintext
MD1,5MN,5VM,5DI10,5GO,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:**  aBCn  \( a = \text{Axis number} \quad n = \text{integer or real} \geq 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 Load Compare (aLCn) 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 millisecond (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 *Next Compare (aNCn)* command. As with all compare operations the Load Compare command (aLCn) is used to define the first compare position. The Next Compare 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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Output Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0*</td>
<td>Disabled (default)</td>
</tr>
<tr>
<td>1*</td>
<td>Static</td>
</tr>
<tr>
<td>2*</td>
<td>Toggle</td>
</tr>
<tr>
<td>3*</td>
<td>One-shot</td>
</tr>
</tbody>
</table>

*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
```

**Backlash compensation off**

**MCCL command**: `aBF`  
**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
```
Motion Commands

BN
Backlash compensation On

**MCCL command:**  aBN  \( a = \text{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 been set 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).

```plaintext
1BD100 ;define backlash distance
1BN ;enable backlash compensation
1BF ;disable backlash compensation
```

CA
arc Center Absolute

**MCCL command:**  aCAN  \( a = \text{Axis number}  \quad n = \text{integer or real} \geq 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.

```plaintext
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:**  aCBn  \( a = \text{Axis number}  \quad n = \text{integer} \geq 0 \leq 1024

**applies to:**   Servo, Stepper

**see also:**   CG, DR

Precision MicroControl Corp.
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 **Capture Begin (aCBn)** 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 **Capture Begin** command with parameter \( n \) expressed as a negative number. Captured positions are reported by using the **Display Recorded position (aDRn)** command. To load the number of captured positions into the accumulator use the **Capture Get count (aCG)** 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**: \( aCDn \)  \( a = \) Axis number of the controlling axis  \( n = \) integer or real \( \geq 0 \)

**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 **Contour Path (aCPn)** command (\( a = \) the controlling axis) with parameter \( n = 0 \).

2) **Contour Distance (aCDn)** 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.
Motion Commands

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 Contour Path

MCCL command: aCPn 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.

<table>
<thead>
<tr>
<th>Parameter n</th>
<th>Motion Type</th>
<th>Compatible Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>User defined</td>
<td>CD, GH, MA, MR, VV</td>
</tr>
<tr>
<td>1</td>
<td>Linear</td>
<td>GH, MA, MR, VV</td>
</tr>
<tr>
<td>2</td>
<td>Clockwise arc</td>
<td>CA, CR, GH, MA, MR, VV</td>
</tr>
<tr>
<td>3</td>
<td>Counter-Clockwise arc</td>
<td>CA, CR, GH, MA, MR, VV</td>
</tr>
</tbody>
</table>

CR arc Center Relative

MCCL command: aCRn 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.
EA  arc Ending Angle absolute

**MCCL command:**  \[ aEan \]  \[ a = Axis number \]  \[ n = integer or real >= 0 \]

**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 were 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 Edge Latch

**MCCL command:**  \[ aELn \]  \[ a = Axis number \]  \[ n = integer or real >= 0 \]

**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 (\( aWE \)) 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.
**Motion Commands**

```
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
MD11,LU"STATUS",5RL@0,IC24,MJ12,NO,JR-5  ;test for sensors (home and +limit)
MD12,ST,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
MD14,5EL0,MC15,5WE,5ST,5WS.1,5MF,5MN,5PM,5MA-100
MD15,LU"STATUS",5RL@0,IS18,BK,NO,JR-5   ;loop status for Edge found bit set
5MN                      ;enable axis 5
```

**ER**  
**arc Ending angle Relative**

**MCCL command:**  \( a\text{ER}n \)  
\( a = \text{Axis number} \quad n = \text{integer or real} \geq 0 \)

**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**  
**Find Edge**

**MCCL command:**  \( a\text{FE}n \)  
\( a = \text{Axis number} \quad n = \text{integer or real} \)

**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.

```plaintext
; 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",5RL0,IS24,MJ11,NO,IS10,MJ13,NO,JR-8
; test for sensors (home and +limit)
MD11,LU"STATUS",5RL0,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
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",5RL0,IS18,BK,NO,JR-5 ; loop status for Edge found bit set

5MN ; enable axis 5
```

**FI Find Index**

**MCCL command:** aFln

*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 redefining 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 redefining 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.
Motion Commands

MD1,1LM2,1LN3,1VM,1DI0,1GO,MJ10 ;begin positive velocity mode move
MD10,LU"STATUS",1RL@0,IS25,MJ11,NO,IS10,MJ15,NO,JR-9 ;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

Go Home

**MCCL command:** aGH  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:** aGO  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
Motion Commands

HO  HOme

**MCCL command:**  \( aHO \)  \( a = \text{Axis number} \)

**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.

\[
\begin{align*}
1\text{HO} & : \text{home axis #1 by executing macro #1} \\
2\text{HO} & : \text{home axis #2 by executing macro #2}
\end{align*}
\]

IA  Index Arm

**MCCL command:**  \( aIA \) \( a = \text{Axis number} \)  \( n = \text{integer or real} \)

**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 (\( aWI \)) 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.

\[
\begin{align*}
\text{MD1,1SV50000,1SA100000,1DS100000,1LM2,1LN3,1VM,1DI0,1GO,MJ10} & : \text{begin positive velocity mode move} \\
\text{MD10,LU"STATUS",1RL@0,IS25,MJ11,NO,IS10,MJ15,NO,JR-8} & : \text{test for sensors (home and +limit)} \\
\text{MD11,1ST,1WS.01,1DI1,1GO,MC21,1ST,1WS.01,MJ12} & : \text{move out of coarse home} \\
\text{MD12,MC22,1DI0,1GO,MC23,MJ13} & : \text{move back into coarse home sensor} \\
\text{MD13,1IA1000,MC24,1WI,1ST,1WS.01,MJ14} & : \text{capture index (position = 1000) then stop} \\
\text{MD14,1PM,1MN,1MA1000,1WS.1} & : \text{initialize axis, move to index} \\
\text{MD15,1WS.1,1MN,1DI1,1GO,MC23,MC21,1ST,1WS.01,MJ12} & : \text{limit + true, move negative until coarse} \\
& : \text{home true}
\end{align*}
\]
Motion Commands

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

1MN ; enable axis 1

LC Load Compare

**MCCL command:** aL*CN* 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 **Begin Compare** (**aBCn**) in this chapter for a complete description of position compare operations.

LP Learn Position

**MCCL command:** aL*Pn* 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 ; move to point #1, wait for 0.25 seconds
0MP1, 0WS.25
0MP2, 0WS.25 ; move to point #1, wait for 0.25 seconds
LT  Learn Target

**MCCL command:** \( \text{aLT} n \quad a = \text{Axis number} \quad n = \text{integer} \geq 0, \leq 255 \)

**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

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

MA  Move Absolute

**MCCL command:** \( \text{aMA} n \quad a = \text{Axis number} \quad n = \text{integer or real} \)

**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
```
**Motion Commands**

**MF**  
**Motor oFf**

*MCCL command:*  
aMF  
*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:*  
aMPn  
*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.
Motion Commands

MCCL command: \( aMRn \)  
\( a = \) 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.

\[
\begin{align*}
1MR1000 & \quad ; \text{move 1000 counts / steps} \\
2MR-25000 & \quad ; \text{move -25000 counts / steps}
\end{align*}
\]

MCCL command: \( aNCn \)  
\( a = \) 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 Begin Compare (aBCn) in this chapter for a complete description of position compare operations.

MCCL command: \( aNS \)  
\( a = \) 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.
Motion Commands

OC  Output mode for Compare

**MCCL command:**  aOCn  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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Output Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disabled (default)</td>
</tr>
<tr>
<td>1</td>
<td>Static</td>
</tr>
<tr>
<td>2</td>
<td>Toggle</td>
</tr>
<tr>
<td>3</td>
<td>One-shot</td>
</tr>
</tbody>
</table>

See the description of the *Begin Compare (aBCn)* in this chapter for a complete description of position compare operations.

OP  Output compare Period

**MCCL command:**  aOPn  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 than 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 *Begin Compare (aBCn)* in this chapter for a complete description of position compare operations.

PP  Parabolic Profile

**MCCL command:**  aPP  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.
Motion Commands

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

PR Record axis data

**MCCL command:** \( aPRn \) \( a = \) Axis number \( n = \) integer \( \geq 0, \leq 1024 \)

**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

3DR0100,AL0100,AA1,AR100,RP9 ;display the first 10 recorded positions

PS Profile S-curve

**MCCL command:** \( aPS \) \( a = \) 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

---

Motion Control Command Language Reference Manual
Motion Commands

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.

<table>
<thead>
<tr>
<th>DCX Velocity Profiles for Servo Axes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. Velocity 10,000 counts / sec.</td>
</tr>
<tr>
<td>Time</td>
</tr>
<tr>
<td>Trapezoidal Profile</td>
</tr>
<tr>
<td>S curve Profile</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DCX Accel / Decel Profiles for Servo Axes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accel 100,000 counts / sec. / sec.</td>
</tr>
<tr>
<td>Decel 100,000 counts / sec. / sec.</td>
</tr>
<tr>
<td>Time</td>
</tr>
<tr>
<td>Trapezoidal Profile</td>
</tr>
<tr>
<td>S curve Profile</td>
</tr>
</tbody>
</table>

PT  Profile Trapezoidal

**MCCL command:**  aPT  \( a = \) 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:**  aRRn  \( a = \) Axis number  \( n = \) integer or real \( \geq 0 \)

**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.
Motion Commands

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 No Synchronization (aNS) 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:  aST  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

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

**MCCL command:** aAT 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

**MCCL command:** aAZ 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

**MCCL command:** aCG = 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  
**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  
**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  
**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
```
**Reporting Commands**

**DQ**  
**Display the recorded DAC output**

**MCCL command:** \( \text{aDQ}_a \)  
\( a = \) Axis number  
\( p = \) integer \( \geq 0, < 1024 \)

**applies to:** Analog Command Axis

**see also:** DO, DR, PR

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

\[
\begin{align*}
\text{2DH0,2MN} \\
\text{2MA5000,2PR10,2WS.1} \\
\text{AL0,AR100} &; \text{define register #100 as recorded position} \\
&; \text{pointer} \\
\text{2DQ@100,AL@100,AA1,AR100,RP9} &; \text{display the first 10 recorded DAC output} \\
&; \text{levels}
\end{align*}
\]

**DR**  
**Display Recorded position**

**MCCL command:** \( \text{aDR}_a \)  
\( a = \) Axis number  
\( p = \) integer \( \geq 0, < 1024 \)

**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.

\[
\begin{align*}
\text{2DH0,2MN} \\
\text{2MA5000,2PR10,2WS.1} \\
\text{AL0,AR100} &; \text{define register #100 as recorded position} \\
&; \text{pointer} \\
\text{2DO@100,AL@100,AA1,AR100,RP9} &; \text{display the first 10 recorded optimal} \\
&; \text{positions}
\end{align*}
\]

**GC**  
**Get the Compare count**

**MCCL command:** \( \text{aGC} \)  
\( a = \) 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.

\[
\begin{align*}
\text{1GC,TR0} &; \text{report position capture count} \\
\text{01} &; \text{controller reply}
\end{align*}
\]
ID Information Display

**MCCL command:** \( \text{ID} \ n = \text{integer} \geq 0, \leq 6 \)

**applies to:** Analog Command Axis, Pulse Command Axis

**see also:** TS

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

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

Servo axis primary status info

Stepper axis primary status info
Reporting Commands

Motor Table info

Internal Motor Table info

Filter Gains and Command output

Encoder Fault info

Servo axis primary status info

Axis I/O assignments
TA  Tell Analog

MCCL command:  TAx  \( x = \text{Channel number} \quad x = 1, 2, 3, 4, 5, 6, 7, 8 \)

applies to:  A/D inputs

see also:  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  \( a = \text{Axis number} \quad p = 0, .1, .2, .3, .4, .5 \)

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:  TCx  \( x \geq 0 \leq 32 \)

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 \( x = \text{channel number} \). 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** **Tell Derivative gain**

**MCCL command:** \(aTDp\) \(a = \text{Axis number}\) \(p = 0, 1, 2, 3, 4, 5\)

**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** **Tell command interpreter Error**

**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** **Tell Following error**

**MCCL command:** \(aTFp\) \(a = \text{Axis number}\) \(p = 0, 1, 2, 3, 4, 5\)

**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:**  \( a\text{TG}_p \)  \( a = \) 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:**  \( a\text{TI}_p \)  \( a = \) 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:**  \( a\text{TK}_p \)  \( a = \) 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:**  \( a\text{TL}_p \)  \( a = \) 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
```
**Reporting Commands**

**TM**  
**Te**ll **M**acros

**MCCL command:**  \( \text{TMn} \)  
**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,SR-8
MC11,LU"STATUS",1RL@0,IC24,MJ12,NO,SR-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,SR-5
```  

**TO**  
**Te**ll **O**ptimal

**MCCL command:**  \( a\text{TO}p \)  
**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.

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

**TP**  
**Te**ll **P**osition

**MCCL command:**  \( a\text{TP}p \)  
**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  Tell torQuE

**MCCL command:** \( aTq \)  \( a = \) 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.

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

TR  Tell Register ‘n’

**MCCL command:** \( Trn \)  \( n = \) integer \( \geq 0, \leq 255 \)

**applies to:** AL, AR

**see also:**

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.

```plaintext
TR0
  01 15 ;controller reply, report value in register 0 ;(accumulator)
```
**TS** | **Tell axis Status**
---|---
**MCCL command**: | aTS\(n\)  \(a = \text{Axis number}\)  \(n = \text{integer}\)
**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:

<table>
<thead>
<tr>
<th>Parameter (n)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Primary Status word (32 bits)</td>
</tr>
<tr>
<td>1 - 31</td>
<td>Primary Status bits 1 - 31</td>
</tr>
<tr>
<td>64</td>
<td>Auxiliary Status word (32 bits)</td>
</tr>
<tr>
<td>65 - 95</td>
<td>Auxiliary Status bits 1 - 31</td>
</tr>
<tr>
<td>96</td>
<td>Profile Generator Status word (32 bits)</td>
</tr>
<tr>
<td>97 - 127</td>
<td>Profile Generator Status bits 1 - 31</td>
</tr>
<tr>
<td>128</td>
<td>Trajectory Status word (32 bits)</td>
</tr>
<tr>
<td>129 - 159</td>
<td>Trajectory Status bits 1 - 31</td>
</tr>
<tr>
<td>160</td>
<td>Encoder Fault Status word (32 bits)</td>
</tr>
<tr>
<td>161 - 191</td>
<td>Encoder Fault Status bit 1 - 31</td>
</tr>
</tbody>
</table>

**Note**: The Tell Status command can only be used to display data. In order to evaluate status data and execute conditional operations the Look Up (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'.

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

\* 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)
Reporting Commands

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 Hex Mode (HM) command. To return to decimal formatted replies issue the Decimal Mode (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'.

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

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

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'.

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

example: \( 1TS96 \)
**Reporting Commands**

**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’.

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

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

*example: \(1TS_{128}\)*

**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’.

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

* 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: \(1TS_{160}\)*

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:** \(aT_p\) \(a = \) 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:** \(aTv_p\) \(a = \) 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:**  
aTZ\(par\)  \(a =\) 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 VErison

**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

<table>
<thead>
<tr>
<th>MCCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CF</td>
<td>Channel off</td>
</tr>
<tr>
<td>CH</td>
<td>Channel High true logic</td>
</tr>
<tr>
<td>CL</td>
<td>Channel Low true logic</td>
</tr>
<tr>
<td>CN</td>
<td>Channel on</td>
</tr>
<tr>
<td>GA</td>
<td>Get Analog</td>
</tr>
<tr>
<td>DF</td>
<td>Do if channel off</td>
</tr>
<tr>
<td>DN</td>
<td>Do if channel on</td>
</tr>
<tr>
<td>IF</td>
<td>If channel off</td>
</tr>
<tr>
<td>IN</td>
<td>If channel on</td>
</tr>
<tr>
<td>TA</td>
<td>Tell the value of Analog input</td>
</tr>
<tr>
<td>TC</td>
<td>Tell state of digital Channel</td>
</tr>
<tr>
<td>WF</td>
<td>Wait for channel off</td>
</tr>
<tr>
<td>WN</td>
<td>Wait for channel on</td>
</tr>
</tbody>
</table>
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.

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

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

**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
I/O Commands

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 (TCx) 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 (TCx) command will report a 1 (channel is on).

**TTL outputs (channels 33 - 48)** - issuing the Channel oN (CNx) 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 (CFx) 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).

**MCCL command:**

\[ \text{CN} \quad \text{CN}x \quad x = \text{Channel number} \]

\text{applies to:} Digital I/O

\text{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).

\[
\text{CN34} \quad \text{; turn on digital I/O channel 34}\n\]

\[
:(\text{TTL output #2})\]
DF | Do if channel off
---|---
**MCCL command:** DFx  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 | Do if channel ‘x’ is on
---|---
**MCCL command:** DNx  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 | Get Analog
---|---
**MCCL command:** GAx  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 | If channel off do next command, else skip 2 commands
---|---
**MCCL command:** IFx  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.
I/O Commands

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
o
f next command, else skip 2 commands

MCCL command: INx  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 Analog

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 Channel

MCCL command: TCx  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:**  
WFx  
\( x = \text{Channel number} \)

**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:**  
WNx  
\( x = \text{Channel number} \)

**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

<table>
<thead>
<tr>
<th>MCCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BK</td>
<td>Break</td>
</tr>
<tr>
<td>ET</td>
<td>Escape Task</td>
</tr>
<tr>
<td>GT</td>
<td>Generate Task</td>
</tr>
<tr>
<td>MC</td>
<td>Macro Call</td>
</tr>
<tr>
<td>MD</td>
<td>Macro Definition</td>
</tr>
<tr>
<td>MJ</td>
<td>Macro Jump</td>
</tr>
<tr>
<td>NO</td>
<td>No Operation</td>
</tr>
<tr>
<td>NP</td>
<td>New Priority</td>
</tr>
<tr>
<td>RM</td>
<td>Reset Macros</td>
</tr>
<tr>
<td>TM</td>
<td>Tell Macros</td>
</tr>
</tbody>
</table>
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:** ETn  
**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:** GTn  
**applies to:** Program flow  
**see also:** GT, TR

```
Motion Control Command Language Reference Manual
```

109
Macro and Multi-tasking Commands

**MC**  
**Macro Call**

**MCCL command:** MC\n  
**applies to:** Program flow
  
**see also:** ET, MC, 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  
**applies to:** Program flow
  
**see also:** ET, GT, MD

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.
**Macro and Multi-tasking Commands**

---

**Macro Jump (MJ)**

**MCCL command:** MJn  \( n = \text{integer} \geq 0, < 1000 \)

**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 Operation (NO)**

**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
```

---

**New Priority (NP)**

**MCCL command:** NPn  \( n = \text{integer} \geq 11, \leq 255 \)

**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.

---

**Reset Macros (RM)**

**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.

---

*Motion Control Command Language Reference Manual*
**Macro and Multi-tasking Commands**

**TM**

**Tell Macros**

**MCCL command:** TMn  \( n = \text{integer} \geq -1, < 1000 \)

**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.

```plaintext
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
```
# Register Commands

## Chapter Contents

### Register Commands

<table>
<thead>
<tr>
<th>MCCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>Accumulator Add</td>
</tr>
<tr>
<td>AC</td>
<td>Accumulator Complement</td>
</tr>
<tr>
<td>AD</td>
<td>Accumulator Divide</td>
</tr>
<tr>
<td>AE</td>
<td>Accumulator logical Exclusive or</td>
</tr>
<tr>
<td>AL</td>
<td>Accumulator Load</td>
</tr>
<tr>
<td>AM</td>
<td>Accumulator Multiply</td>
</tr>
<tr>
<td>AN</td>
<td>Accumulator logical And with n,</td>
</tr>
<tr>
<td>AO</td>
<td>Accumulator logical Or with n</td>
</tr>
<tr>
<td>AR</td>
<td>copy Accumulator to Register n</td>
</tr>
<tr>
<td>AS</td>
<td>Accumulator Subtract</td>
</tr>
<tr>
<td>AV</td>
<td>Accumulator evaluate</td>
</tr>
<tr>
<td>AX</td>
<td>get Aux. index position</td>
</tr>
<tr>
<td>GA</td>
<td>Get Analog value</td>
</tr>
<tr>
<td>GU</td>
<td>Get the default axis</td>
</tr>
<tr>
<td>GX</td>
<td>Get auxiliary encoder position</td>
</tr>
<tr>
<td>LU</td>
<td>Look Up motor table variable</td>
</tr>
<tr>
<td>RA</td>
<td>copy Register to Accumulator</td>
</tr>
<tr>
<td>RB</td>
<td>Read Byte into accumulator</td>
</tr>
<tr>
<td>RD</td>
<td>Read Double into accumulator</td>
</tr>
<tr>
<td>RL</td>
<td>Read Long into accumulator</td>
</tr>
<tr>
<td>RV</td>
<td>Read float into accumulator</td>
</tr>
<tr>
<td>RW</td>
<td>Read Word into accumulator</td>
</tr>
<tr>
<td>SL</td>
<td>Shift Left accumulator n bits</td>
</tr>
<tr>
<td>SR</td>
<td>Shift Right accumulator n bits</td>
</tr>
<tr>
<td>TR</td>
<td>Tell contents of Register n</td>
</tr>
<tr>
<td>WB</td>
<td>Write accumulator Byte to n</td>
</tr>
<tr>
<td>WD</td>
<td>Write accumulator double to n</td>
</tr>
<tr>
<td>WL</td>
<td>Write accumulator Long to n</td>
</tr>
<tr>
<td>WV</td>
<td>Write accumulator float to n</td>
</tr>
<tr>
<td>WW</td>
<td>Write accumulator Word to n</td>
</tr>
</tbody>
</table>
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 \( ACC = \lnot ACC \), 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:**  \( \text{AE} n \)  \( n = \) integer or real

Performs \( \text{ACC} = \text{ACC} \oplus n \), 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:**  \( \text{AL} n \)  \( n = \) 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:**  \( \text{AM} n \)  \( n = \) integer or real

Performs \( \text{ACC} = \text{ACC} \times n \), 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:**  \( \text{AN} n \)  \( n = \) integer or real
Register Commands

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**: \( AR_n \) \( n = \) 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**: \( AV_n \) \( n = \) integer >= 0, <=25
Perform 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 lists the available operations and the respective command parameter to use. The result that is stored in the Accumulator \((1 \leq n \leq 25)\) will be a 64 bit real in all cases except the Convert to ASCII operation that returns an integer.

<table>
<thead>
<tr>
<th>Parameter ( n )</th>
<th>Operation</th>
<th>Return type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Convert to ASCII (Address placed in ACC)</td>
<td>Integer</td>
</tr>
<tr>
<td>2</td>
<td>Change Sign</td>
<td>Double</td>
</tr>
<tr>
<td>3</td>
<td>Absolute Value</td>
<td>Double</td>
</tr>
<tr>
<td>4</td>
<td>Ceiling</td>
<td>Double</td>
</tr>
<tr>
<td>5</td>
<td>Floor</td>
<td>Double</td>
</tr>
<tr>
<td>6</td>
<td>Fraction</td>
<td>Double</td>
</tr>
<tr>
<td>7</td>
<td>Round</td>
<td>Double</td>
</tr>
<tr>
<td>8</td>
<td>Square</td>
<td>Double</td>
</tr>
<tr>
<td>9</td>
<td>Square Root</td>
<td>Double</td>
</tr>
<tr>
<td>10</td>
<td>Sine</td>
<td>Double</td>
</tr>
<tr>
<td>11</td>
<td>Cosine</td>
<td>Double</td>
</tr>
<tr>
<td>12</td>
<td>Tangent</td>
<td>Double</td>
</tr>
<tr>
<td>13</td>
<td>Arc Sine</td>
<td>Double</td>
</tr>
<tr>
<td>14</td>
<td>Arc Cosine</td>
<td>Double</td>
</tr>
<tr>
<td>15</td>
<td>Arc Tangent</td>
<td>Double</td>
</tr>
<tr>
<td>16</td>
<td>Hyperbolic Sine</td>
<td>Double</td>
</tr>
<tr>
<td>17</td>
<td>Hyperbolic Cosine</td>
<td>Double</td>
</tr>
<tr>
<td>18</td>
<td>Hyperbolic Tangent</td>
<td>Double</td>
</tr>
<tr>
<td>19</td>
<td>Exponent</td>
<td>Double</td>
</tr>
<tr>
<td>20</td>
<td>Log</td>
<td>Double</td>
</tr>
<tr>
<td>21</td>
<td>Log10</td>
<td>Double</td>
</tr>
<tr>
<td>22</td>
<td>Load Pi</td>
<td>Double</td>
</tr>
<tr>
<td>23</td>
<td>Load 2 * Pi</td>
<td>Double</td>
</tr>
<tr>
<td>24</td>
<td>Load Pi/2</td>
<td>Double</td>
</tr>
<tr>
<td>25</td>
<td>Convert double register contents to an integer</td>
<td>Integer</td>
</tr>
</tbody>
</table>
**AX**

**get Auxiliary encoder index position**

**MCCL command:** aAXn  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 Auxiliary encoder Find 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**

**Get Analog**

**MCCL command:** GAx  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**

**Get axis Used**

**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 **Use Axis command (UAn).**

```
GU ;load accumulator with the current default axis
```
**Register Commands**

**GX**  
**Get the position of the auxiliary encoder**

*MCCL command:* \( a \text{GX} \)  
*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).

\[ 5\text{GX} ; \text{load accumulator with the position of axis 5} \]
\[ ; \text{auxiliary encoder} \]

**LU**  
**Look Up variable**

*MCCL command:* \( \text{LUs} \) \( s = \text{string parameter ("variable name")} \)

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.

\[ \text{LU"STATUS",2RL@0} ; \text{load the motor table address for axis #2 status} \]
\[ ; \text{into the accumulator. Load the 32 bit status word} \]
\[ ; \text{of axis #2 into the accumulator} \]

**RA**  
**copy Register to Accumulator**

*MCCL command:* \( \text{RAn} \) \( n = \text{integer or real} \)

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.

\[ \text{RA100} ; \text{copy value in register into the accumulator} \]

**RB**  
**Read the Byte at absolute memory location ‘n’ into the accumulator**

*MCCL command:* \( a\text{RB}n \) \( a = \text{Axis number} \) \( n = \text{integer} \)

This is used to load the accumulator with a byte of data. Typically this command is used in conjunction with the Look Up 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          Read the **Double** (64 bit real) value at absolute memory location ‘n’ into the accumulator

**MCCL command:** \( a\text{RD}n \quad a = \text{Axis number} \quad n = \text{real} \)

This is used to load the accumulator with a 64 bit real value. Typically this command is used in conjunction with the **Look Up** 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          Read the **Long** (32 bit integer) value at absolute memory location ‘n’ into the accumulator

**MCCL command:** \( a\text{RL}n \quad a = \text{Axis number} \quad n = \text{integer} \)

This is used to load the accumulator with a 32 bit integer. Typically this command is used in conjunction with the **Look Up** 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          Read the **float** (32 bit real) value at absolute memory location ‘n’ into the accumulator

**MCCL command:** \( a\text{RV}n \quad a = \text{Axis number} \quad n = \text{real} \)

This is used to load the accumulator with a 32 bit real value. Typically this command is used in conjunction with the **Look Up** 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          Read the **Word** (16 bit integer) value at absolute memory location ‘n’ into the accumulator

**MCCL command:** \( a\text{RW}n \quad a = \text{Axis number} \quad n = \text{integer} \)

This is used to load the accumulator with a 16 bit integer. Typically this command is used in conjunction with the **Look Up** 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.
Register Commands

**SL**  **Shift Left accumulator by ‘n’ bits**

*MCCL command*: \(\text{SL}_n\) \(n = \text{integer} > 0, \leq 31\)

Performs \(\text{ACC} = \text{ACC} \ll n\), 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**  **Shift Right accumulator by ‘n’ bits**

*MCCL command*: \(\text{SR}_n\) \(n = \text{integer} > 0, \leq 31\)

Performs \(\text{ACC} = \text{ACC} \gg n\), 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**  **Tell Register ‘n’**

*MCCL command*: \(\text{TR}_n\) \(n = \text{integer} \geq 0, \leq 255\)

*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**  **Write the low Byte in the accumulator to absolute memory location ‘n’**

*MCCL command*: \(\text{WB}_n\) \(n = \text{integer}\)

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  Write the Double (64 bit real) value in the accumulator to absolute memory location ‘n’

**MCCL command:**  \( WDn \quad n = \text{real} \)

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  Write the Long (32 bit integer) value in the accumulator to absolute memory location ‘n’

**MCCL command:**  \( WLn \quad n = \text{integer} \)

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  Write the float (32 bit real) value in the accumulator to absolute memory location ‘n’

**MCCL command:**  \( WVn \quad n = \text{real} \)

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  Write the low Word (16 bit integer) value in the accumulator to absolute memory location ‘n’

**MCCL command:**  \( WWn \quad n = \text{integer} \)

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.*
## Sequence Commands

<table>
<thead>
<tr>
<th>MCCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF</td>
<td>Do if channel off</td>
</tr>
<tr>
<td>DN</td>
<td>Do if channel on</td>
</tr>
<tr>
<td>IB</td>
<td>If Below do next command</td>
</tr>
<tr>
<td>IC</td>
<td>If Clear, do next command</td>
</tr>
<tr>
<td>IE</td>
<td>If Equals do next command</td>
</tr>
<tr>
<td>IF</td>
<td>If channel off do next command</td>
</tr>
<tr>
<td>IG</td>
<td>If accumulator is Greater do next command</td>
</tr>
<tr>
<td>IN</td>
<td>If channel on do next command</td>
</tr>
<tr>
<td>IP</td>
<td>Interrupt on absolute Position</td>
</tr>
<tr>
<td>IR</td>
<td>Interrupt on Relative position</td>
</tr>
<tr>
<td>IS</td>
<td>If bit Set do next command</td>
</tr>
<tr>
<td>IU</td>
<td>If Unequal do next command</td>
</tr>
<tr>
<td>JP</td>
<td>Jump to command absolute</td>
</tr>
<tr>
<td>JR</td>
<td>Jump to command Relative</td>
</tr>
<tr>
<td>RP</td>
<td>RePeat</td>
</tr>
<tr>
<td>WA</td>
<td>WAit (time)</td>
</tr>
<tr>
<td>WE</td>
<td>Wait for Edge</td>
</tr>
<tr>
<td>WF</td>
<td>Wait for channel off</td>
</tr>
<tr>
<td>WI</td>
<td>Wait for Index</td>
</tr>
<tr>
<td>WN</td>
<td>Wait for channel on</td>
</tr>
<tr>
<td>WP</td>
<td>Wait for absolute Position</td>
</tr>
<tr>
<td>WR</td>
<td>Wait for Relative position</td>
</tr>
<tr>
<td>WS</td>
<td>Wait for Stop</td>
</tr>
<tr>
<td>WT</td>
<td>Wait for Target</td>
</tr>
</tbody>
</table>

**Chapter Contents**
Sequence (If/Then) Commands

**DF**

**Do if channel oFf**

**MCCL command:** DF\textsubscript{x} \quad x = \text{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\textsubscript{x} \quad x = \text{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
Sequence (If/Then) Commands

### IB
**If the accumulator is Below ‘n’, execute the next command, else skip 2 commands**

**MCCL command:** IB \( n \)  
**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
**If bit ‘n’ of the accumulator is Clear (equal to 0), execute the next command, else skip 2 commands**

**MCCL command:** IC \( n \)  
**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
**If the accumulator Equals “n”, execute the next command, else skip 2 commands**

**MCCL command:** IE \( n \)  
**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
```
Sequence (If/Then) Commands

**IF**

If channel off 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**

If the accumulator is Greater 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**

If channel 'on' 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
```
Sequence (If/Then) Commands

**IP**  
**Interrupt on absolute Position**

- **MCCL command:** allPn  
  - 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:** allRn  
  - 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 | If bit ‘n’ of the accumulator is Set execute the next command, else skip 2 commands

**MCCL command:** IS\_n \hspace{1em} n = integer \geq 0, \leq 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 | If the accumulator is Unequal to “n” execute the next command, else skip 2 commands

**MCCL command:** IU\_n \hspace{1em} 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 | Jump to command absolute

**MCCL command:** JP\_n \hspace{1em} 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
```
Sequence (If/Then) Commands

**JR**  
**Jump to command Relative**

**MCCL command:** JR$n \ n = \text{integer}$  
**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**  
**RePeat**

**MCCL command:** RP$n \ n = \text{integer} \geq 0, \leq 2,147,483,647$  
**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**  
**WAit**

**MCCL command:** WAn \ $n = \text{integer or real} \geq 0$  
**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:  aWE  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 Edge Latch (EL) command.

WF  Wait for digital channel oFf

MCCL command:  WFx  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:  aWln  a = Axis number  n = integer or real \( \geq 0 \)
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)
Sequence (If/Then) Commands

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 Index Arm (IA) command.

WN  
**Wait for digital channel oN**

*MCCL command:* \( \text{WN}x \quad x = \text{Channel number} \)
*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  
**Wait for absolute Position**

*MCCL command:* \( \text{aWP}n \quad a = \text{Axis number} \quad n = \text{integer or real} \)
*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  
**Wait for Relative position**

*MCCL command:* \( \text{aWR}n \quad n = \text{integer or real} \)
*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.


dd

```
W S
```

## WS

**MCCL command:**  
*WSn*  

**applies to:** Analog Command Axis, Pulse Command Axis

**see also:** WA, WT

Wait for Stop

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.

```
W T
```

## WT

**MCCL command:**  
*WTn*  

**applies to:** Analog Command Axis, Pulse Command Axis

**see also:** DB, DT, WS

Wait for Target

This command will delay additional command execution until the 'At Target' status bit has been set.
Sequence (If/Then) Commands

Servo axis - The conditions for a closed loop axis servo to have reached its' target, are that it remains within the position DeadBand range $n$ for the time period specified by the Delay at Target 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.

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.
## Miscellaneous Commands

<table>
<thead>
<tr>
<th>MCCL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DM</td>
<td>Decimal Mode</td>
</tr>
<tr>
<td>DW</td>
<td>Disable Watchdog</td>
</tr>
<tr>
<td>EF</td>
<td>Echo off</td>
</tr>
<tr>
<td>EN</td>
<td>Echo on</td>
</tr>
<tr>
<td>FD</td>
<td>Format text with Doubles</td>
</tr>
<tr>
<td>FM</td>
<td>Free Memory</td>
</tr>
<tr>
<td>FT</td>
<td>Format Text with Integers</td>
</tr>
<tr>
<td>HE</td>
<td>HELP</td>
</tr>
<tr>
<td>HM</td>
<td>Hexadecimal Mode</td>
</tr>
<tr>
<td>ME</td>
<td>MEMory allocate</td>
</tr>
<tr>
<td>NO</td>
<td>No Operation</td>
</tr>
<tr>
<td>OD</td>
<td>Output text with Doubles</td>
</tr>
<tr>
<td>OT</td>
<td>Output Text with integers</td>
</tr>
<tr>
<td>PC</td>
<td>set Prompt Character</td>
</tr>
<tr>
<td>RT</td>
<td>Rest system</td>
</tr>
</tbody>
</table>
### 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  

Disable the processor watchdog circuit.  
**comment:** This command is reserved for factory use only.

### EF

**Echo oFf**

**MCCL command:** EFn  
$n = 0, 1, 2, 3$
Miscellaneous Commands

**EN**

**Echo on**

*MCCL command:*  
EN\textit{n} \quad 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:*  
FM\textit{n} \quad n = \text{integer} >0, \leq 65536

*applies to:*  
Other

*see also:*  
ME

Returns the memory space allocated by the ME command to the ‘heap’. Parameter \textit{n} must equal the value (memory address) that was loaded into the accumulator after successful execution of the ME memory (ME) allocate command.

**FD**

**Format text with Doubles**

*MCCL command:*  
OD\textit{s} \quad s = \text{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:*  
OT\textit{s} \quad s = \text{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

**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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Report supported MCCL commands</td>
</tr>
<tr>
<td>1</td>
<td>Report Data Table variable names</td>
</tr>
<tr>
<td>&quot;mm&quot;</td>
<td>Report the description of the MCCL command (where mm = valid MCCL command mnemonic)</td>
</tr>
</tbody>
</table>

HM  

**Hexadecimal Mode**

**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
1TS
01 10 ;controller reply, axis status = 10d

HM
1TS
01 000000a ;hexadecimal mode
             ;controller reply, axis status = Ah
```
Miscellaneous Commands

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.

```plaintext
;register 200 stores the part count
a10,ar200

MC10,MC20,MC30,MC100

MD100,a1@200,OT"The part count = %d \n"
```

;load the accumulator with the
;part count (register 200).

---

**PC**

**Prompt Character**

**MCCL command:** $\text{PCn}$

**applies to:** Other

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:** $\text{aRT}$

**applies to:** Analog Command Axis, Pulse Command Axis, Digital I/O

Performing 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

- MCCL Error codes
Controller Error Codes

When executing MCCL (Motion Control Command Language) command sequences the command interpreter will report the following error code when appropriate:

<table>
<thead>
<tr>
<th>Description</th>
<th>Error code</th>
</tr>
</thead>
<tbody>
<tr>
<td>No error</td>
<td>0</td>
</tr>
<tr>
<td>Unrecognized command</td>
<td>1</td>
</tr>
<tr>
<td>Bad command format</td>
<td>2</td>
</tr>
<tr>
<td>I/O error</td>
<td>3</td>
</tr>
<tr>
<td>Command string too long</td>
<td>4</td>
</tr>
<tr>
<td>Command Parameter Error</td>
<td>-1</td>
</tr>
<tr>
<td>Command Code Invalid</td>
<td>-2</td>
</tr>
<tr>
<td>Negative Repeat Count</td>
<td>-3</td>
</tr>
<tr>
<td>Macro Define Command Not First</td>
<td>-4</td>
</tr>
<tr>
<td>Macro Number Out of Range</td>
<td>-5</td>
</tr>
<tr>
<td>Macro Doesn't Exist</td>
<td>-6</td>
</tr>
<tr>
<td>Command Canceled by User</td>
<td>-7</td>
</tr>
<tr>
<td>Contour Path Command Not First</td>
<td>-8</td>
</tr>
<tr>
<td>Contour Path Command Parameter Invalid</td>
<td>-9</td>
</tr>
<tr>
<td>Contour Path Command Doesn't Specify an AXIS</td>
<td>-10</td>
</tr>
<tr>
<td>Axis error (over travel error, max. following error exceeded)</td>
<td>-13</td>
</tr>
<tr>
<td>No axis specified</td>
<td>-14</td>
</tr>
<tr>
<td>Axis not assigned</td>
<td>-15</td>
</tr>
<tr>
<td>Axis already assigned</td>
<td>-16</td>
</tr>
<tr>
<td>Axis duplicate assigned</td>
<td>-17</td>
</tr>
<tr>
<td>Insufficient memory</td>
<td>-18</td>
</tr>
<tr>
<td>Unrecognized variable name</td>
<td>-19</td>
</tr>
<tr>
<td>Invalid background task ID</td>
<td>-20</td>
</tr>
<tr>
<td>Command not supported</td>
<td>-21</td>
</tr>
</tbody>
</table>
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.
Controller Error Codes
Index
## Index

### A

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abort move</td>
<td>61</td>
</tr>
<tr>
<td>Acceleration set</td>
<td>47</td>
</tr>
<tr>
<td>Acceleration gain set</td>
<td>35</td>
</tr>
<tr>
<td>Accumulator copy from register</td>
<td>120</td>
</tr>
<tr>
<td>Accumulator copy to register</td>
<td>117</td>
</tr>
<tr>
<td>Accumulator if below</td>
<td>126</td>
</tr>
<tr>
<td>Accumulator if bit n clear</td>
<td>126</td>
</tr>
<tr>
<td>Accumulator if bit n set</td>
<td>129</td>
</tr>
<tr>
<td>Accumulator if equal</td>
<td>129</td>
</tr>
<tr>
<td>Accumulator if greater</td>
<td>127</td>
</tr>
<tr>
<td>Accumulator if unequal</td>
<td>129</td>
</tr>
<tr>
<td>Accumulator load</td>
<td>116</td>
</tr>
<tr>
<td>Accumulator load with byte</td>
<td>120</td>
</tr>
<tr>
<td>Accumulator load with double</td>
<td>121</td>
</tr>
<tr>
<td>Accumulator load with long</td>
<td>121</td>
</tr>
<tr>
<td>Accumulator load with word</td>
<td>121</td>
</tr>
<tr>
<td>Accumulator shift left</td>
<td>122</td>
</tr>
<tr>
<td>Accumulator shift right</td>
<td>122</td>
</tr>
<tr>
<td>Accumulator write byte</td>
<td>122</td>
</tr>
<tr>
<td>Accumulator write double</td>
<td>123</td>
</tr>
<tr>
<td>Accumulator write float</td>
<td>123</td>
</tr>
<tr>
<td>Accumulator write long</td>
<td>123</td>
</tr>
<tr>
<td>Accumulator write word</td>
<td>123</td>
</tr>
<tr>
<td>Amplifier fault disable</td>
<td>39</td>
</tr>
<tr>
<td>Amplifier fault enable</td>
<td>40</td>
</tr>
<tr>
<td>Analog input reporting</td>
<td>87, 105, 106, 119</td>
</tr>
<tr>
<td>Arc center</td>
<td>64, 66</td>
</tr>
<tr>
<td>Arc radius</td>
<td>78</td>
</tr>
<tr>
<td>At target commanding</td>
<td>133</td>
</tr>
<tr>
<td>Auxiliary encoder find index</td>
<td>61</td>
</tr>
<tr>
<td>Auxiliary encoder home</td>
<td>35</td>
</tr>
<tr>
<td>Auxiliary encoder load position in accumulator</td>
<td>120</td>
</tr>
<tr>
<td>Auxiliary encoder report position</td>
<td>82, 120</td>
</tr>
<tr>
<td>Auxiliary encoder report position in accumulator</td>
<td>119</td>
</tr>
</tbody>
</table>

### B

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background task generating</td>
<td>109</td>
</tr>
<tr>
<td>Background task terminating</td>
<td>109</td>
</tr>
<tr>
<td>Backlash compensation set</td>
<td>35</td>
</tr>
<tr>
<td>Breakpoint on absolute position</td>
<td>128</td>
</tr>
</tbody>
</table>

### C

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closed loop encoder scaling</td>
<td>39</td>
</tr>
<tr>
<td>Closed loop stepper disable</td>
<td>58</td>
</tr>
<tr>
<td>Closed loop stepper enable</td>
<td>58</td>
</tr>
<tr>
<td>Command format (MCCL)</td>
<td>10</td>
</tr>
<tr>
<td>Command interpreter error report</td>
<td>88</td>
</tr>
<tr>
<td>Compare output mode period</td>
<td>76</td>
</tr>
<tr>
<td>Contour count report</td>
<td>97</td>
</tr>
<tr>
<td>Contour mode set</td>
<td>57</td>
</tr>
<tr>
<td>Cubic spline interpolation</td>
<td>75, 79</td>
</tr>
<tr>
<td>Current position report</td>
<td>10, 90</td>
</tr>
</tbody>
</table>
## Index

### D

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deceleration</td>
<td>37</td>
</tr>
<tr>
<td>Deceleration gain</td>
<td>36</td>
</tr>
<tr>
<td>Decimal mode</td>
<td>137</td>
</tr>
<tr>
<td>Delay at target</td>
<td>38</td>
</tr>
<tr>
<td>Derivative gain</td>
<td>47, 88</td>
</tr>
<tr>
<td>Derivative sampling period</td>
<td>40</td>
</tr>
<tr>
<td>Digital I/O</td>
<td>103, 106, 127, 102, 87, 107, 131, 132</td>
</tr>
<tr>
<td>Direction</td>
<td>37</td>
</tr>
<tr>
<td>Backlash compensation</td>
<td>63</td>
</tr>
<tr>
<td>Display controller information</td>
<td>85</td>
</tr>
<tr>
<td>Display memory</td>
<td>83</td>
</tr>
<tr>
<td>Download</td>
<td>20</td>
</tr>
<tr>
<td>Driver fault</td>
<td>39, 40</td>
</tr>
</tbody>
</table>

### E

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electronic gearing</td>
<td>49</td>
</tr>
<tr>
<td>Backlash compensation</td>
<td>64</td>
</tr>
<tr>
<td>Enable the axis</td>
<td>12</td>
</tr>
<tr>
<td>Encoder fail disable</td>
<td>38</td>
</tr>
<tr>
<td>Encoder fail enable</td>
<td>38</td>
</tr>
<tr>
<td>Encoder fault</td>
<td>96</td>
</tr>
<tr>
<td>Encoder index</td>
<td>69, 71, 98</td>
</tr>
<tr>
<td>Error codes</td>
<td>143, 88</td>
</tr>
<tr>
<td>Example</td>
<td>52, 50, 25</td>
</tr>
</tbody>
</table>

### F

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find encoder index</td>
<td>69</td>
</tr>
<tr>
<td>Find home sensor edge</td>
<td>68</td>
</tr>
<tr>
<td>Firmware version</td>
<td>98</td>
</tr>
<tr>
<td>Following error</td>
<td>48</td>
</tr>
<tr>
<td>Frictionless servo</td>
<td>45</td>
</tr>
<tr>
<td>Full current</td>
<td>39</td>
</tr>
</tbody>
</table>

### G

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gain mode</td>
<td>57</td>
</tr>
</tbody>
</table>

### H

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Half current</td>
<td>40</td>
</tr>
<tr>
<td>Help</td>
<td>19</td>
</tr>
<tr>
<td>Hexadecimal mode</td>
<td>139</td>
</tr>
<tr>
<td>Home sensor</td>
<td>68</td>
</tr>
</tbody>
</table>

### I

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integral gain</td>
<td>42, 89, 48</td>
</tr>
<tr>
<td>Interrupts, enable</td>
<td>38</td>
</tr>
</tbody>
</table>

### L

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning points</td>
<td>72, 73</td>
</tr>
<tr>
<td>Limits</td>
<td>103, 104, 42, 44, 43, 40, 42</td>
</tr>
<tr>
<td>Loading MFX-PCI1000 data</td>
<td>30</td>
</tr>
<tr>
<td>Logical operations</td>
<td>26</td>
</tr>
</tbody>
</table>
and.................................................................116
complement..................................................115
evaluate..........................................................117
or 116, 117

M

Macro command
as background task ........................................22
changing priority ............................................111
defining ........................................................21, 110
described ......................................................20
jump to absolute command ..........................129
jump to relative .............................................130
memory size .................................................21
reporting ......................................................21
resetting (deleting) ......................................21, 111
starting .....................................................110, 111
volatile .....................................................21
Macro command reporting ........................90, 112
Master/slave mode
set ..............................................................58
Math operations
add..............................................................115
divide ............................................................115
multiply .....................................................116
subtract ....................................................117
MCCL command
1 KHz servo loop .......................................44
2 KHz servo loop .......................................45
4 KHz servo loop .......................................41
abort move..................................................61
acceleration ...............................................47
Acceleration gain ....................................35
accumulator add .......................................115
accumulator and .......................................116
accumulator complement ........................115
accumulator copy ......................................117
accumulator divide ...................................115
accumulator evaluate ................................117
accumulator load ......................................116
accumulator logical or .................................116
accumulator multiply ................................116
accumulator or ..........................................117
accumulator subtract .................................117
amplifier fault, disable ..............................39
amplifier fault, enable ..................................40
arc center ..................................................64, 66
arc radius ...................................................78
aux. encoder find index ..................................61
axis status ...................................................92
backlash compensation ................................35
backlash compensation, off .........................35
backlash compensation, on ..........................64
break ..........................................................109
channel as high active .................................103
channel as low active .................................103
compare position .........................................75
contour distance ..........................................65
contour mode .............................................57
contour path ..............................................66
deceleration ...............................................37
deceleration gain .........................................36
decimal mode ..............................................137
default axis ...............................................137
define home ...............................................37
delay at target .............................................38
derivative gain ..............................................47
derivative output mode ..................................46
Derivative sampling period ..........................40
direction .....................................................37
disable watchdog .........................................137
display controller information ......................85
display memory ..........................................83
driver fault, disable .....................................39, 40
echo off ......................................................137
echo on .......................................................138
eedge latch ..................................................67
electronic gearing .........................................49
enable bus interrupts ....................................38
encoder fail disable .....................................38
encoder fail enable .....................................38
encoder scaling .........................................39
ending angle .............................................67, 68
escape background task ................................109
Find edge (home sensor) ..............................68
Find encoder index ......................................69
following error, allowable .........................48
format text ................................................138
free memory .............................................138
gain mode ..................................................57
generate background task ..........................109
go ..............................................................70
go home .....................................................70
half current, stepper ....................................40
hexadecimal mode .....................................139
high motion limit .......................................40
home .........................................................71
home aux. encoder .....................................35
if below .....................................................126
if bit n clear .............................................126
if bit n set ................................................129
if equal .....................................................126
if greater ..................................................127
if input off .............................................105, 125, 127
if input on .............................................105, 106, 125, 127
if unequal ................................................129
index arm ...................................................71
integral gain .............................................48
integral limit ...............................................41
integral option ..........................................42
jump to command, absolute .......................129
jump to command, relative .........................130
learn current position .................................72
Index

learn target position ................................................... 73
limit enable ............................................................... 44
limit mode ............................................................... 43
limits disable ............................................................ 42
load aux. enc. index position .............................. 119
load compare ........................................................... 72
look up variable ....................................................... 26, 120
low motion limit ........................................................ 42
macro call ............................................................... 110
macro define ........................................................... 110
macro jump ............................................................. 111
macro reset ............................................................. 111
master/slave mode .................................................. 58
memory allocate ........................................................ 139
minimum velocity, stepper ....................................... 45
motor off ................................................................. 74
motor on ................................................................. 74
move absolute .......................................................... 73
move relative ........................................................... 75
move to point ........................................................... 74
new priority ............................................................. 111
no operation ............................................................ 111, 140
no synchronization ................................................... 75
on-line help ............................................................. 19
output offset ............................................................. 46
output compare mode ............................................... 76
output compare period ............................................... 76
output deadband ..................................................... 45
output text .............................................................. 140
parabolic profile ...................................................... 76
phasing, output/encoder .......................................... 46
position capture ...................................................... 64
position compare .................................................... 62
position deadband ................................................... 36
position mode ........................................................ 58
prompt character ..................................................... 141
proportional gain .................................................... 48
PWM time constant ................................................ 47
read byte ............................................................... 120
read double ............................................................ 121
read long ............................................................... 121
read word .............................................................. 121
record axis data ....................................................... 77
register copy .......................................................... 120
repeat ................................................................. 130
report actual position ............................................. 90
report analog input ............................................... 87, 106
report analog input value ...................................... 105, 119
report aux. encoder index position ....................... 82
report aux. encoder position .................................... 82, 120
report breakpoint ................................................... 87
report command interpreter error ......................... 88
report contouring count .......................................... 97
report default axis .................................................. 119
report derivative gain ............................................. 88
report digital input state ........................................ 87, 106
report firmware version ........................................ 98
report following error ............................................. 88
report index position ............................................ 98
report integral gain ................................................. 89
report macro ........................................................ 90, 112
report optimal position ........................................... 90
report position capture count .................................. 82
report position compare count ................................ 84
report proportional gain ....................................... 89
report recorded actual position ................................ 84
report recorded aux. encoder position .................... 83
report recorded DAC output ................................... 84
report recorded optimal position.......................... 83
report register value ............................................. 91, 122
report target ........................................................ 97
report torque ........................................................ 91
report velocity ....................................................... 97
report velocity constant ......................................... 89
reset controller ...................................................... 141
s-curve ............................................................... 77
set breakpoint on absolute position ...................... 128
set breakpoint on relative position ....................... 128
shift left .............................................................. 122
shift right ............................................................ 122
slave ratio ........................................................... 49
status ................................................................. 92
Stop ................................................................. 79
synchronization on ............................................... 79
torque mode ........................................................ 59
torque, DAC output ............................................... 49
trapezoidal profile .................................................. 78
turn off digital output ............................................ 103
turn on digital output ............................................ 104
user units ............................................................. 51, 52, 53
vector acceleration ............................................... 53
vector deceleration ............................................... 53
vector velocity, max .............................................. 54
velocity gain ........................................................ 54
velocity mode ....................................................... 59
velocity override ................................................... 54
velocity, max ......................................................... 50
wait, absolute position ......................................... 132
wait, at target ......................................................... 133
wait, digital input .................................................. 107, 131, 132
wait, home edge ..................................................... 131
wait, index ............................................................. 131
wait, relative position .......................................... 132
wait, time ............................................................. 130
wait, trajectory complete ....................................... 133
write byte ............................................................ 122
write double ........................................................ 123
write float ........................................................... 123
write long ............................................................ 123
write word ........................................................... 123
MCCL commands

MCCL mnemonic
AA ............................................................... 115
AB ................................................................. 61
AC ................................................................. 115
<table>
<thead>
<tr>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD ...</td>
</tr>
<tr>
<td>AF ...</td>
</tr>
<tr>
<td>AG ...</td>
</tr>
<tr>
<td>AH ...</td>
</tr>
<tr>
<td>AL ...</td>
</tr>
<tr>
<td>AM ...</td>
</tr>
<tr>
<td>AN ...</td>
</tr>
<tr>
<td>AO ...</td>
</tr>
<tr>
<td>AR ...</td>
</tr>
<tr>
<td>AS ...</td>
</tr>
<tr>
<td>AT ...</td>
</tr>
<tr>
<td>AV ...</td>
</tr>
<tr>
<td>AX ...</td>
</tr>
<tr>
<td>AZ ...</td>
</tr>
<tr>
<td>BC ...</td>
</tr>
<tr>
<td>BD ...</td>
</tr>
<tr>
<td>BF ...</td>
</tr>
<tr>
<td>BK ...</td>
</tr>
<tr>
<td>BN ...</td>
</tr>
<tr>
<td>CA ...</td>
</tr>
<tr>
<td>CB ...</td>
</tr>
<tr>
<td>CD ...</td>
</tr>
<tr>
<td>CF ...</td>
</tr>
<tr>
<td>CG ...</td>
</tr>
<tr>
<td>CH ...</td>
</tr>
<tr>
<td>CL ...</td>
</tr>
<tr>
<td>CM ...</td>
</tr>
<tr>
<td>CN ...</td>
</tr>
<tr>
<td>CP ...</td>
</tr>
<tr>
<td>CR ...</td>
</tr>
<tr>
<td>DA ...</td>
</tr>
<tr>
<td>DB ...</td>
</tr>
<tr>
<td>DE ...</td>
</tr>
<tr>
<td>DF ...</td>
</tr>
<tr>
<td>DG ...</td>
</tr>
<tr>
<td>DH ...</td>
</tr>
<tr>
<td>DI ...</td>
</tr>
<tr>
<td>DM ...</td>
</tr>
<tr>
<td>DN ...</td>
</tr>
<tr>
<td>DO ...</td>
</tr>
<tr>
<td>DQ ...</td>
</tr>
<tr>
<td>DR ...</td>
</tr>
<tr>
<td>DS ...</td>
</tr>
<tr>
<td>DT ...</td>
</tr>
<tr>
<td>DW ...</td>
</tr>
<tr>
<td>EA ...</td>
</tr>
<tr>
<td>EE ...</td>
</tr>
<tr>
<td>EF ...</td>
</tr>
<tr>
<td>EI38</td>
</tr>
<tr>
<td>EL ...</td>
</tr>
<tr>
<td>EN ...</td>
</tr>
<tr>
<td>ER ...</td>
</tr>
<tr>
<td>ES ...</td>
</tr>
<tr>
<td>ET ...</td>
</tr>
<tr>
<td>FC ...</td>
</tr>
<tr>
<td>FD ...</td>
</tr>
<tr>
<td>FE ...</td>
</tr>
<tr>
<td>FF ...</td>
</tr>
<tr>
<td>FI69</td>
</tr>
<tr>
<td>FM ...</td>
</tr>
<tr>
<td>FN ...</td>
</tr>
<tr>
<td>FR ...</td>
</tr>
<tr>
<td>GA ...</td>
</tr>
<tr>
<td>GC ...</td>
</tr>
<tr>
<td>GH ...</td>
</tr>
<tr>
<td>GM ...</td>
</tr>
<tr>
<td>GO ...</td>
</tr>
<tr>
<td>GT ...</td>
</tr>
<tr>
<td>GU ...</td>
</tr>
<tr>
<td>GX ...</td>
</tr>
<tr>
<td>HC ...</td>
</tr>
<tr>
<td>HE ...</td>
</tr>
<tr>
<td>HL ...</td>
</tr>
<tr>
<td>HM ...</td>
</tr>
<tr>
<td>HO ...</td>
</tr>
<tr>
<td>HS ...</td>
</tr>
<tr>
<td>IA ...</td>
</tr>
<tr>
<td>IB ...</td>
</tr>
<tr>
<td>IC ...</td>
</tr>
<tr>
<td>ID ...</td>
</tr>
<tr>
<td>IE126</td>
</tr>
<tr>
<td>IG ...</td>
</tr>
<tr>
<td>IL41</td>
</tr>
<tr>
<td>IM ...</td>
</tr>
<tr>
<td>IN ...</td>
</tr>
<tr>
<td>IO ...</td>
</tr>
<tr>
<td>IP128</td>
</tr>
<tr>
<td>IR ...</td>
</tr>
<tr>
<td>IS129</td>
</tr>
<tr>
<td>IU ...</td>
</tr>
<tr>
<td>JP129</td>
</tr>
<tr>
<td>JR ...</td>
</tr>
<tr>
<td>LC ...</td>
</tr>
<tr>
<td>LF ...</td>
</tr>
<tr>
<td>LL ...</td>
</tr>
<tr>
<td>LM ...</td>
</tr>
<tr>
<td>LN ...</td>
</tr>
<tr>
<td>LP ...</td>
</tr>
<tr>
<td>LS ...</td>
</tr>
<tr>
<td>LT ...</td>
</tr>
<tr>
<td>LU ...</td>
</tr>
<tr>
<td>MA ...</td>
</tr>
<tr>
<td>MC ...</td>
</tr>
<tr>
<td>MD ...</td>
</tr>
<tr>
<td>ME ...</td>
</tr>
<tr>
<td>MF ...</td>
</tr>
<tr>
<td>MJ ...</td>
</tr>
<tr>
<td>MN ...</td>
</tr>
<tr>
<td>MP ...</td>
</tr>
<tr>
<td>MR ...</td>
</tr>
<tr>
<td>MS ...</td>
</tr>
<tr>
<td>MV ...</td>
</tr>
<tr>
<td>NC ...</td>
</tr>
</tbody>
</table>
Index

<table>
<thead>
<tr>
<th>Index</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>111, 140</td>
</tr>
<tr>
<td>NP</td>
<td>111</td>
</tr>
<tr>
<td>NS</td>
<td>.75</td>
</tr>
<tr>
<td>OB</td>
<td>.45</td>
</tr>
<tr>
<td>OC</td>
<td>.76</td>
</tr>
<tr>
<td>OD</td>
<td>140</td>
</tr>
<tr>
<td>OM</td>
<td>.46</td>
</tr>
<tr>
<td>OO</td>
<td>.46</td>
</tr>
<tr>
<td>OP</td>
<td>.76</td>
</tr>
<tr>
<td>OT</td>
<td>140</td>
</tr>
<tr>
<td>PC</td>
<td>141</td>
</tr>
<tr>
<td>PH</td>
<td>.46</td>
</tr>
<tr>
<td>PK</td>
<td>.47</td>
</tr>
<tr>
<td>PM</td>
<td>.58</td>
</tr>
<tr>
<td>PF</td>
<td>.76</td>
</tr>
<tr>
<td>PR</td>
<td>.77</td>
</tr>
<tr>
<td>PS</td>
<td>.78</td>
</tr>
<tr>
<td>PT</td>
<td>.58</td>
</tr>
<tr>
<td>QM</td>
<td>.58</td>
</tr>
<tr>
<td>RA</td>
<td>120</td>
</tr>
<tr>
<td>RB</td>
<td>120</td>
</tr>
<tr>
<td>RD</td>
<td>121</td>
</tr>
<tr>
<td>RL</td>
<td>121</td>
</tr>
<tr>
<td>RM</td>
<td>111</td>
</tr>
<tr>
<td>RP</td>
<td>130</td>
</tr>
<tr>
<td>RR</td>
<td>.78</td>
</tr>
<tr>
<td>RT</td>
<td>141</td>
</tr>
<tr>
<td>RV</td>
<td>121</td>
</tr>
<tr>
<td>RW</td>
<td>121</td>
</tr>
<tr>
<td>SA</td>
<td>.47</td>
</tr>
<tr>
<td>SD</td>
<td>.47</td>
</tr>
<tr>
<td>SE</td>
<td>.48</td>
</tr>
<tr>
<td>SG</td>
<td>.48</td>
</tr>
<tr>
<td>SI48</td>
<td></td>
</tr>
<tr>
<td>SL</td>
<td>122</td>
</tr>
<tr>
<td>SM</td>
<td>.58</td>
</tr>
<tr>
<td>SN</td>
<td>.79</td>
</tr>
<tr>
<td>SQ</td>
<td>.49</td>
</tr>
<tr>
<td>SR</td>
<td>122</td>
</tr>
<tr>
<td>SS</td>
<td>.49</td>
</tr>
<tr>
<td>ST</td>
<td>.79</td>
</tr>
<tr>
<td>SV</td>
<td>.50</td>
</tr>
<tr>
<td>TA</td>
<td>.87, 106</td>
</tr>
<tr>
<td>TB</td>
<td>.87</td>
</tr>
<tr>
<td>TC</td>
<td>.87, 106</td>
</tr>
<tr>
<td>TD</td>
<td>.88</td>
</tr>
<tr>
<td>TE</td>
<td>.88</td>
</tr>
<tr>
<td>TF</td>
<td>.88</td>
</tr>
<tr>
<td>TG</td>
<td>.89</td>
</tr>
<tr>
<td>TI89</td>
<td></td>
</tr>
<tr>
<td>TK</td>
<td>.89</td>
</tr>
<tr>
<td>TL</td>
<td>.89</td>
</tr>
<tr>
<td>TM</td>
<td>90, 112</td>
</tr>
<tr>
<td>TO</td>
<td>.90</td>
</tr>
<tr>
<td>TP</td>
<td>.90</td>
</tr>
<tr>
<td>TQ</td>
<td>.91</td>
</tr>
<tr>
<td>TR</td>
<td>.91, 122</td>
</tr>
<tr>
<td>TS</td>
<td>.92</td>
</tr>
<tr>
<td>TT</td>
<td>.97</td>
</tr>
<tr>
<td>TV</td>
<td>.97</td>
</tr>
<tr>
<td>TX</td>
<td>.97</td>
</tr>
<tr>
<td>TZ</td>
<td>.98</td>
</tr>
<tr>
<td>U</td>
<td>.50</td>
</tr>
<tr>
<td>UK</td>
<td>.51</td>
</tr>
<tr>
<td>UO</td>
<td>.51</td>
</tr>
<tr>
<td>UR</td>
<td>.51</td>
</tr>
<tr>
<td>US</td>
<td>.52</td>
</tr>
<tr>
<td>UT</td>
<td>.52</td>
</tr>
<tr>
<td>UZ</td>
<td>.53</td>
</tr>
<tr>
<td>VA</td>
<td>.53</td>
</tr>
<tr>
<td>VD</td>
<td>.53</td>
</tr>
<tr>
<td>VE</td>
<td>.98</td>
</tr>
<tr>
<td>VG</td>
<td>.54</td>
</tr>
<tr>
<td>VM</td>
<td>.59</td>
</tr>
<tr>
<td>VO</td>
<td>.54</td>
</tr>
<tr>
<td>VV</td>
<td>.54</td>
</tr>
<tr>
<td>WA</td>
<td>.130</td>
</tr>
<tr>
<td>WB</td>
<td>.122</td>
</tr>
<tr>
<td>WD</td>
<td>.123</td>
</tr>
<tr>
<td>WE</td>
<td>.131</td>
</tr>
<tr>
<td>WF</td>
<td>107, 131</td>
</tr>
<tr>
<td>WI</td>
<td>131</td>
</tr>
<tr>
<td>WL</td>
<td>123</td>
</tr>
<tr>
<td>WN</td>
<td>107, 132</td>
</tr>
<tr>
<td>WP</td>
<td>132</td>
</tr>
<tr>
<td>WR</td>
<td>132</td>
</tr>
<tr>
<td>WS</td>
<td>133</td>
</tr>
<tr>
<td>WT</td>
<td>133</td>
</tr>
<tr>
<td>WV</td>
<td>123</td>
</tr>
<tr>
<td>WV</td>
<td>123</td>
</tr>
<tr>
<td>WV</td>
<td>123</td>
</tr>
<tr>
<td>WW</td>
<td>123</td>
</tr>
<tr>
<td>Memory display</td>
<td>83</td>
</tr>
<tr>
<td>MFX-PCI1000 command (MCCL) single stepping</td>
<td>29</td>
</tr>
<tr>
<td>MFX-PCI11000 command (MCCL) description</td>
<td>3, 10</td>
</tr>
<tr>
<td>pausing a command / sequence</td>
<td>13</td>
</tr>
<tr>
<td>terminating a command / sequence</td>
<td>13</td>
</tr>
<tr>
<td>Minimum velocity, stepper set</td>
<td>45</td>
</tr>
<tr>
<td>Motion required settings</td>
<td>12</td>
</tr>
<tr>
<td>Motor off</td>
<td>74</td>
</tr>
<tr>
<td>Motor on</td>
<td>74</td>
</tr>
<tr>
<td>enable the axis</td>
<td>12</td>
</tr>
<tr>
<td>Motor status</td>
<td>92</td>
</tr>
<tr>
<td>loading</td>
<td>26</td>
</tr>
<tr>
<td>Motor table data look up</td>
<td>26, 120</td>
</tr>
<tr>
<td>Move abort</td>
<td>61</td>
</tr>
<tr>
<td>absolute</td>
<td>73</td>
</tr>
<tr>
<td>arc</td>
<td>66</td>
</tr>
<tr>
<td>arc center</td>
<td>64, 66</td>
</tr>
<tr>
<td>arc radius</td>
<td>78</td>
</tr>
<tr>
<td>contour distance</td>
<td>65</td>
</tr>
</tbody>
</table>

152 Precision MicroControl Corp.
Index

| Multi-tasking | report status ...............................................................95  
|              | set ..............................................................................58  
|              | set ..............................................................................36  
|              | report compare count...................................................84  
|              | report settings ......................................................88, 89  
|              | enable ........................................................................74  
|              | disable .......................................................................74  
|              | set ..............................................................................46  
|              | report ........................................................................92  
|              | active level of input channel...........................103, 104  
|              | encoder fault, report................................................96  
|              | motor .................................................................92  
|              | report .................................................................92  
|              | trajectory mode, report.............................................96  
|              | velocity profile, report.............................................95  
|              | step position capture .............................................67  
|              | step rate max. .........................................................47  
|              | .78KHz ......................................................................44  
|              | 5 MHz .......................................................................41  
|              | 625 KHz ....................................................................45  
|              | stepper motor control .............................................46  
|              | CW / CCW ...............................................................46  
|              | pulse & dir. .............................................................46  
|              | stop move...............................................................79  
|              | syntax .....................................................................141  
|              | MCCL commands.......................................................9  
|              | Optimal position  
|              | report ........................................................................90  
|              | Output deadband  
|              | set ..............................................................................45  
|              | Output offset  
|              | set ..............................................................................46  
|              | Parabolic velocity profile ............................................76  
|              | Pausing  
|              | MCCL command / sequence ........................................13  
|              | Phasing, output/encoder  
|              | set ..............................................................................46  
|              | PID filter  
|              | disable .................................................................74  
|              | enable ........................................................................74  
|              | report settings ......................................................88, 89  
|              | Position capture ....................................................64, 65, 72, 75, 84  
|              | report capture count................................................82  
|              | Position compare ....................................................62, 76  
|              | enable ........................................................................75  
|              | load points ............................................................72  
|              | mode .........................................................................76  
|              | period .........................................................................76  
|              | report compare count .............................................84  
|              | Position deadband  
|              | set ..............................................................................36  
|              | Position mode  
|              | set ..............................................................................58  
|              | Profile generator  
|              | report status ..............................................................95  
|              | Prompt character ........................................................141  
|              | Proportional gain ........................................................9  
|              | Report  
|              | current position .......................................................12  
|              | current position of axis ...........................................10  
|              | motor status ............................................................92  
|              | Required settings for motion ....................................12  
|              | Reset  
|              | the controller ..........................................................141  
|              | System  
|              | S-curve velocity profile .............................................77  
|              | Single stepping a program ........................................29  
|              | Slave ratio  
|              | set..............................................................................49  
|              | Software  
|              | WinControl .............................................................20, 29, 144  
|              | Status  
|              | active level of input channel .....................................103, 104  
|              | encoder fault, report ...............................................96  
|              | motor .................................................................92  
|              | report .................................................................92  
|              | trajectory mode, report .............................................96  
|              | velocity profile, report .............................................95  
|              | Step position capture .............................................67  
|              | Step rate max. .........................................................47  
|              | .78KHz ......................................................................44  
|              | 5 MHz .......................................................................41  
|              | 625 KHz ....................................................................45  
|              | stepper motor control .............................................46  
|              | CW / CCW ...............................................................46  
|              | pulse & dir. .............................................................46  
|              | stop move ...............................................................79  
|              | Syntax  
|              | MCCL commands.......................................................9  
|              | PWM frequency  
|              | set ..............................................................................48  
|              | R  
|              | Record  
|              | actual position .......................................................77  
|              | aux. encoder position ..............................................77  
|              | DAC output .............................................................77  
|              | following error .......................................................77  
|              | optimal position ......................................................84  
|              | report actual position .............................................84  
|              | report aux. encoder position ....................................83  
|              | report DAC output ....................................................83  
|              | report optimal position ............................................83  
|              | Register  
|              | report value .............................................................91, 122  
|              | Repeat MCCL command sequence ................................130  
|              | Repeating a command or sequence .........................12  
|              | O  
|              | Optimal position  
|              | report ........................................................................90  
|              | Output deadband  
|              | set ..............................................................................45  
|              | Output offset  
|              | set ..............................................................................46  
|              | P  
|              | Parabolic velocity profile ............................................76  
|              | Pausing  
|              | MCCL command / sequence ........................................13  
|              | Phasing, output/encoder  
|              | set ..............................................................................46  
|              | PID filter  
|              | disable .................................................................74  
|              | enable ........................................................................74  
|              | report settings ......................................................88, 89  
|              | Position capture ....................................................64, 65, 72, 75, 84  
|              | report capture count................................................82  
|              | Position compare ....................................................62, 76  
|              | enable ........................................................................75  
|              | load points ............................................................72  
|              | mode .........................................................................76  
|              | period .........................................................................76  
|              | report compare count .............................................84  
|              | Position deadband  
|              | set ..............................................................................36  
|              | Position mode  
|              | set ..............................................................................58  
|              | Profile generator  
|              | report status ..............................................................95  
|              | Prompt character ........................................................141  
|              | Proportional gain ........................................................9  
|              | ending angle ..............................................................67, 68  
|              | go (velocity mode) ......................................................70  
|              | go home ......................................................................70  
|              | homing routine ............................................................71  
|              | linear interpolation ....................................................66  
|              | relative ......................................................................75  
|              | stop ..............................................................................79  
|              | to learned point ........................................................74  
|              | Multi-tasking  
|              | commands not supported ...........................................22  
|              | CPU utilization ............................................................23  
|              | described ...................................................................22  
|              | example .....................................................................22, 23, 24  
|              | global data registers ...................................................23  
|              | passing data between .................................................23  
|              | private data registers ...................................................23  
|              | quantity supported ....................................................23  
|              | testing .......................................................................22  
|              | O  
|              | Optimal position  
|              | report ........................................................................90  
|              | Output deadband  
|              | set ..............................................................................45  
|              | Output offset  
|              | set ..............................................................................46  
|              | P  
|              | Parabolic velocity profile ............................................76  
|              | Pausing  
|              | MCCL command / sequence ........................................13  
|              | Phasing, output/encoder  
|              | set ..............................................................................46  
|              | PID filter  
|              | disable .................................................................74  
|              | enable ........................................................................74  
|              | report settings ......................................................88, 89  
|              | Position capture ....................................................64, 65, 72, 75, 84  
|              | report capture count................................................82  
|              | Position compare ....................................................62, 76  
|              | enable ........................................................................75  
|              | load points ............................................................72  
|              | mode .........................................................................76  
|              | period .........................................................................76  
|              | report compare count .............................................84  
|              | Position deadband  
|              | set ..............................................................................36  
|              | Position mode  
|              | set ..............................................................................58  
|              | Profile generator  
|              | report status ..............................................................95  
|              | Prompt character ........................................................141  
|              | Proportional gain ........................................................9  
|              | report .........................................................................89  
|              | set ..............................................................................48  
|              | PWM frequency  
|              | set ..............................................................................47  

Motion Control Command Language Reference Manual
Index

T

Target position report .................................................97
Terminating background task ........................................109
command sequence .............................................109
MCCL command / sequence ................................13
Text formatting .........................................................138
Text file download ....................................................20
Text message outputting ...........................................140
Torque report ...........................................................91
Torque mode set ......................................................59
Trajectory mode report status ....................................96
Trapezoidal velocity profile ........................................78

U

User memory ..........................................................32
User registers description ........................................30
User units set ..........................................................51, 52, 53

V

Vector acceleration set ................................................53
Vector deceleration set ................................................53
Vector velocity, max. set ............................................54
Velocity report setting ...............................................97
Velocity gain set ......................................................54
Velocity mode set ....................................................59
Velocity override set .................................................54
Velocity profile report status .....................................95
Velocity profiles parabolic ..........................................76
s-curve ................................................................77, 78
Velocity, max. set .....................................................50

W

Wait absolute position .............................................132
at target .................................................................133
digital input ..........................................................107, 131, 132
home edge ..............................................................131
index .................................................................131
relative position .....................................................132
time .................................................................130
trajectory complete ...............................................133
Watchdog disable ....................................................137
WinControl opening ...............................................6