User Manual

DMC-1300

Manual Rev. 1.3a

By Galil Motion Control, Inc.

Galil Motion Control, Inc.
270 Technology Way
Rocklin, California 95765
Phone: (916) 626-0101
Fax: (916) 626-0102

Internet Address: support@galilmc.com
URL: www.galilmc.com

Rev 12-99

Using This Manual

This user manual provides information for proper operation of the DMC-1300 controller.

Your DMC-1300 motion controller has been designed to work with both servo and stepper type motors. In addition, the DMC-1300 has a daughter board for controllers with more than 4 axes. Installation and system setup will vary depending upon whether the controller will be used with stepper motors or servo motors, and whether the controller has more than 4 axes of control. To make finding the appropriate instructions faster and easier, icons will be next to any information that applies exclusively to one type of system. Otherwise, assume that the instructions apply to all types of systems. The icon legend is shown below.

Attention: Pertains to servo motor use.

Attention: Pertains to stepper motor use.

Attention: Pertains to controllers with more than 4 axes.

Please note that many examples are written for the DMC-1340 four-axis controller or the DMC-1380 eight axes controller. Users of the DMC-1330 3-axis controller, DMC-1320 2-axis controller or DMC-1310 1-axis controller should note that the DMC-1330 uses the axes denoted as XYZ, the DMC-1320 uses the axes denoted as XY, and the DMC-1310 uses the X-axis only.

Examples for the DMC-1380 denote the axes as A,B,C,D,E,F,G,H. Users of the DMC-1350 5-axis controller, DMC-1360 6-axis controller or DMC-1370, 7-axis controller should note that the DMC-1350 denotes the axes as A,B,C,D,E, the DMC-1060 denotes the axes as A,B,C,D,E,F and the DMC-1370 denotes the axes as A,B,C,D,E,F,G. The axes A,B,C,D may be used interchangeably with X,Y,Z,W.

This manual was written for the DMC-1300 firmware revision 2.0 and later. For controllers with firmware previous to revision 2.0, please consult the original manual for your hardware. The later revision firmware was previously specified as DMC-1300-18.

WARNING: Machinery in motion can be dangerous! It is the responsibility of the user to design effective error handling and safety protection as part of the machine. Galil shall not be liable or responsible for any incidental or consequential damages.

Firmware Updates

New feature for Rev 2.0h February 1998:

Feature	Description
1. CMDERR enhanced to support multitasking	If CMDERR occurs on thread 1,2 or 3, thread will be holted. Thread can be re-started with XQ_ED2,_ED1, 1 for retry XQ_ED3,_ED1, 1 for next instruction
2. _VM returns instantaneous commanded vector velocity	
3. FA resolution increased to 0.25.	

New feature for Rev 2.0g November 1997:

Feature	Description
1. CR radius now has range of 16 million	Allows for large circular interpolation radii
2. _AB returns abort input	Allows for monitoring of abort input
3. CW,1 When output FIFO full application program will not pause but data will be lost	Allows for output FIFO buffer to fill up without affecting the execution of a program
4. List Variable (LV), List Array (LA), List app program labels (LL)	Allows for the user to interrogate Ram

New feature for Rev 2.0e May 1997:

Feature	Description
1. ER now accepts argument < 0	Disables error output (LED and Error Output does not turn on for that axis)
2. During a PR decel can now be changed on an unnatural stop	Allows for monitoring of abort input

New feature for Rev 2.0d February 1997:

Feature	Description
1. AP, MF, MR in stepper now uses _DE instead of _RP	Trippoints based on register after buffer
2. \ now terminates QD	Download array no longer requires control sequence to end
3. KS can now be fraction (down to .5)	Allows for smaller stepper motor smoothing delay (due to filter)
4. New arguments for MT of 2.5 and -2.5	Reverses the direction of motion from MT 2 and MT -2
5. MG now can go to 80 characters	Increased message size

New feature for Rev 2.0c October 1996:

Feature	Description
1. MC now works for steppers	More accurate trippoint for stepper motor completion

New feature for Rev 2.0b September 1996:

Feature	Description
1. Operand '&' and '|' for conditional statements	Allows for multiple conditional statements in jump routines IE. (A>=3) & (B<55) | (C=78)

New feature for Rev 2.0 March 1996. (This revision is also designated DMC-1300-18).

DAC resolution increased to 16-bits.
Step motor control method improved.

Command	Description
KS	Step Motor Smoothing

New feature for Rev 1.5 (rev. 1.2 for DMC-1080)

Electronic Cam

New commands:

Command	Description
EA	Choose ECAM master
EM	Cam Cycle Command
EP	Cam table interval and starting point
ET	ECAM table entry
EB	Enable ECAM
EG	Engage ECAM cycle
EQ	Disengage ECAM

New features added Jan 1995:

Allow circular array recording**.**

New commands added July 1994 Rev 1.4:

Command	Description
RI,N	N is a new interrupt mask which allows changing the interrupt mask
QU	Upload array
QD	Download array
MF x,y,z,w	Trippoint for motion - forward direction
MR x,y,z,w	Trippoint for motion - reverse direction
MC XYZW	In position trippoint
TW x,y,z,w	Sets timeout for in position

VR r	Sets speed ratio for VS

New commands added January 1994 Rev 1.3:

Can specify parameters with axis designator. For example:

Command	Description
KPZ=10	Set Z axis gain to 10
KP*=10	Set all axes gains to 10

(KPXZ=10 is invalid. Only one or all axes can be specified at a time).

New commands added July 1993 Rev 1.2:

Command	Description
_UL	Gives available variables
_DL	Give available labels
@COM[n]	2's complement function

New commands added March 1993: Rev 1.2

Command	Description
_CS	Segment counter in LM, VM and CM modes
_AV	Return distance travelled in LM and VM modes
_VPX	Return the coordinate of the last point in a motion sequence, LM or VM
VP x,y<n	Can specify vector speed with each vector segment Where <n sets vector speed

New commands added January 1993:

Command	Description
HX	Halt execution for multitasking
AT	At time trippoint for relative time from reference
ES	Ellipse scale factor
OB n,expression	Defines output n where expression is logical operation, such as I1 & I6, variable or array element
XQ#Label,n	Where n = 0 through 3 and is program thread for multitasking

DV	Dual velocity for Dual Loop

Feature	Description
1.	Allows gearing and coordinated move simultaneously
2.	Multitasking for up to four independent programs
3.	Velocity Damping from auxiliary encoder for dual loop

Contents

Chapter 1 Overview 1

Introduction 1
Overview of Motor Types 2
Standard Servo Motors with +/- 10 Volt Command Signal 2
Stepper Motor with Step and Direction Signals 2
DMC 1300 Functional Elements 2
Microcomputer Section 3
Motor Interface 3
Communication 3
General I/O 3
System Elements 4
Motor 4
Amplifier (Driver) 4
Encoder 4
Watch Dog Timer 5

Chapter 2 Getting Started 7

Elements You Need 7
Installing the DMC 1300 8
Step 1. Determine Overall Motor Configuration 8
Step 2. Configure Address Jumpers on the DMC 1300 9
Step 3. Install the DMC 1300 into VME Host 9
Step 4. Install and Test Communications Software 10
Step 5. Connect Amplifiers and Encoders 10
Step 6a. Connect Standard Servo Motors 12
Step 6b. Connect Step Motors 15
Step 7. Tune the Servo System 16
Design Examples 17
Example 1 - System Set-up 17
Example 2 - Profiled Move 17
Example 3 - Multiple Axes 18
Example 4 - Independent Moves 18
Example 5 - Position Interrogation 18
Example 6 - Absolute Position 19
Example 7 - Velocity Control 19
Example 8 - Operation Under Torque Limit 20
Example 9 - Interrogation 20
Example 10 - Operation in the Buffer Mode 20
Example 11 - Motion Programs 21
Example 12 - Motion Programs with Loops 21
Example 13 - Motion Programs with Trippoints 21
Example 14 - Control Variables 22

Example 15 - Linear Interpolation 23
Example 16 - Circular Interpolation 24

Chapter 3 Connecting Hardware 25

Overview 25
Using Optoisolated Inputs 25
Limit Switch Input 25
Home Switch Input 26
Abort Input 27
Uncommitted Digital Inputs 27
Wiring the Optoisolated Inputs 27
Using an Isolated Power Supply 29
Bypassing the Opto-Isolation: 30
Changing Optoisolated Inputs From Active Low to Active High 31
Amplifier Interface 31
TTL Inputs 32
Analog Inputs 32
TTL Outputs 33
Offset Adjustment 33

Chapter 4 VME Communication 35

Introduction 35
RAM Organization 35
Semaphore Registers 37
General Registers 37
Command Buffer 42
Response Buffer 45
Contour Buffer 47
Program Buffer 48
Axis Buffers 49
Coordinate Axis Buffer 52
Variable Buffer 52
Interrupts 53

Chapter 5 Command Basics 55

Introduction 55
Command Syntax 56
ASCII 56
Binary 57
Coordinated Motion with more than 1 axis 57
Program Syntax 57
Controller Response to DATA 57
Interrogating the Controller 58
Interrogation Commands 58
Additional Interrogation Methods 59
Operands 59
Command Summary 60

Chapter 6 Programming Motion 61

Overview 61
Independent Axis Positioning 61

Command Summary - Independent Axis 62
Operand Summary - Independent Axis 62
Independent Jogging....64
Command Summary - Jogging....64
Operand Summary - Independent Axis 65
Linear Interpolation Mode....65
Specifying Linear Segments 66
Specifying Vector Acceleration, Deceleration and Speed:....66
Additional Commands....67
Command Summary - Linear Interpolation 68
Operand Summary - Linear Interpolation....68
Vector Mode: Linear and Circular Interpolation Motion....71
Specifying Vector Segments....71
Specifying Vector Acceleration, Deceleration and Speed:....72
Additional Commands....72
Command Summary - Vector Mode Motion....74
Operand Summary - Vector Mode Motion....74
Electronic Gearing....76
Command Summary - Electronic Gearing....76
Operand Summary - Electronic Gearing....76
Contour Mode....78
Specifying Contour Segments....78
Additional Commands....80
Command Summary - Contour Mode....80
Operand Summary - Contour Mode....80
Stepper Motor Operation....83
Specifying Stepper Motor Operation....84
Using an Encoder with Stepper Motors 85
Command Summary - Stepper Motor Operation....85
Operand Summary - Stepper Motor Operation....85
Dual Loop (Auxiliary Encoder)....86
Backlash Compensation....87
Command Summary - Using the Auxiliary Encoder....88
Operand Summary - Using the Auxiliary Encoder....88
Motion Smoothing....89
Using the IT and VT Commands (S curve profiling):....89
Using the KS Command (Step Motor Smoothing):....90
Homing....91
High Speed Position Capture (Latch)....94

Chapter 7 Application Programming 97

Overview....97
Using the DMC 1300 Editor to Enter Programs 97
Edit Mode Commands 98
Program Format....100
Using Labels in Programs 100
Special Labels 101
Commenting Programs 101
Executing Programs - Multitasking....103
Debugging Programs 104
Program Flow Commands....106
Event Triggers & Trippoints 106
Event Trigger Examples:....108

Conditional Jumps....111
Subroutines....114
Stack Manipulation....114
Automatic Subroutines for Monitoring Conditions....114
Mathematical and Functional Expressions....117
Mathematical Expressions....117
Bit-Wise Operators....118
Functions....119
Variables....120
Assigning Values to Variables:....120
Operands....122
Special Operands (Keywords)....123
Arrays....123
Defining Arrays....123
Assignment of Array Entries....124
Automatic Data Capture into Arrays....125
Deallocating Array Space....127
Output of Data (Numeric and String)....127
Sending Messages....127
Programmable Hardware I/O....128
Digital Outputs....128
Digital Inputs....129
Input Interrupt Function....130
Analog Inputs....130
Example Applications....131
Wire Cutter....131
X-Y Table Controller....133
Speed Control by Joystick....135
Position Control by Joystick....136
Backlash Compensation by Sampled Dual-Loop....136

Chapter 8 Hardware & Software Protection 139

Introduction....139
Hardware Protection....139
Output Protection Lines....139
Input Protection Lines....139
Software Protection....140
Programmable Position Limits....140
Off-On-Error....141
Automatic Error Routine....141
Limit Switch Routine....141

Chapter 9 Troubleshooting 143

Overview....143
Installation....143
Communication....144
Stability....144
Operation....144

Chapter 10 Theory of Operation 145

Overview....145

Operation of Closed-Loop Systems 147
System Modeling 148
Motor-Amplifier....149
Encoder....151
DAC....152
Digital Filter....152
ZOH....152
System Analysis 153
System Design and Compensation....155
The Analytical Method....155

Chapter 11 Command Reference 159

Command Descriptions 159
Axes Arguments....159
Parameter Arguments 159
Direct Command Arguments 160
Interrogation 160
Operand Usage....160
Usage Description....160
Default Description....161
Servo and Stepper Motor Notation:....161
AB (Binary D3) 162
AC (Binary CC)....163
AD (Binary A2)....164
AI (Binary A1) 166
AL (Binary 90)....167
AM (Binary A4)....168
AP (Binary A3) 169
AR (Binary CF) 170
AS (Binary A5) 171
AT (Binary A7)....172
AV (Binary AB) 173
BG (Binary CE)....174
BL (Binary C7)....176
BN (Binary B0)....177
BP (Binary B2)....178
BV (Binary B2) 179
CB (Binary 8E)....180
CD (No Binary)....181
CE (Binary F2)....182
CM (Binary D4)....183
CN (Binary F3) 184
CP (Binary 9E) 185
CR (Binary E1)....186
CS (Binary E2)....188
CW (No Binary)....189
DA (No Binary)....190
DC (Binary CD)....191
DE (Binary C4) 192
DM (No Binary)....193
DP (Binary C3) 194
DT (No Binary)....195
DV (Binary F4) 196

ED (Binary 98)....197
EI (Binary 8C)....198
EN (Binary 84)....200
ER (Binary 88)....202
ES (Binary EB)....203
FA (Binary C1)....204
FE (Binary D1)....205
FI (Binary D6)....206
FL (Binary C6)....207
FV (Binary C5)....208
GA (No Binary)....209
GN (Binary B8)....210
GR (Binary D7)....211
HM (Binary D0)....212
HX (Binary 97)....214
II (Binary II)....215
IL (Binary B5)....217
IP (Binary CF)....218
IT (Binary BC)....219
JG (Binary CB)....220
JP (No Binary)....221
JS (No Binary)....222
KD (Binary B7)....223
KI (Binary BA)....224
KP (Binary B6)....225
KS (Binary ?)....226
LE (Binary E6)....227
_LF* (No Binary)....228
* This is an Operand - Not a command.....228
LI (Binary E9)....229
LM (Binary E8)....231
_LR* (Binary ?)....233
*Note: This is an Operand - Not a command....233
MC (Binary D8)....234
MF (Binary D9)....236
MG (Binary 81)....237
MO (Binary BD)....238
MR (No Binary)....239
MT (Binary F5)....240
NO (No Binary)....241
OB (Binary 92)....242
OE (Binary C0)....243
OF (Binary C2)....244
OP (Binary 8F)....245
PA (Binary C8)....246
PP (No Binary)....247
PR (Binary C9)....248
RA (No Binary)....249
RC (Binary F0)....250
RD (No Binary)....251
RE (No Binary)....252
RI (No Binary)....253
RL (Binary F1)....254

RM (Binary B1) 255
RP (No Binary) 256
RS (Binary AC) 258
<control>R<control>S 259
<control>R<control>V 260
SB (Binary 8D) 261
SC (No Binary) 262
SH (Binary BB) 263
SP (Binary CA) 264
ST (Binary D2) 265
TB (No Binary) 266
TC (No Binary) 267
TD (No Binary) 270
TE (No Binary) 271
TI (Binary E0) 272
TIME* 274
TL (Binary BE) 275
TM (Binary AE) 276
TN (Binary EC) 277
TP (No Binary) 278
TR (Binary AF) 279
TS (Binary DF) 280
TT (No Binary) 282
TV (No Binary) 283
TW (No Binary) 284
UI (Binary 8B) 285
VA (Binary E3) 286
VD (Binary E5) 287
VE (Binary E6) 288
VM (Binary E7) 289
VP (Binary B2) 291
VS (Binary E4) 293
VT (Binary EA) 294
WC (No Binary) 295
WT (Binary A6) 296
XQ (Binary 82) 297
ZR (Binary B9) 298
ZS (Binary 83) 299

Appendices **301**

Electrical Specifications 301
Servo Control 301
Stepper Control 301
Input/Output 301
Power 302
Performance Specifications 302
Connectors for DMC 1300 Main Board 303
J2 - Main (60 pin IDC) 303
J5 - General I/O (26 pin IDC) 304
J3 - Aux Encoder (20 pin IDC) 304
J4 - Driver (20 pin IDC) 305
J6 - Daughter Board Connector (60 pin) 305
J7 - 10 pin 305

Connectors for Auxiliary Board (Axes E,F,G,H) 306
JD2 - Main (60 pin IDC) 306
JD5 - I/O (26 pin IDC) 308
JD3 - 20 pin IDC - Auxiliary Encoders 308
JD4 - 20 pin IDC - Amplifiers 309
JD6 - Daughterboard Connector (60 pin) 309
Pin-Out Description for DMC 1300 310
Jumper Description for DMC 1300 312
Offset Adjustments for DMC 1300 313
Accessories and Options 313
ICM-1100 Interconnect Module 314
AMP/ICM-1100 CONNECTIONS 314
J2 - Main (60 pin IDC) 318
J3 - Aux Encoder (20 pin IDC) 318
J4 - Driver (20 pin IDC) 318
J5 - General I/O (26 pin IDC) 318
Connectors are the same as described in section entitled “Connectors for DMC 1300 Main Board”. see pg. 303 318
JX6, JY6, JZ6, JW6 - Encoder Input (10 pin IDC) 318
ICM-1100 Drawing 319
AMP-11x0 Mating Power Amplifiers 320
Coordinated Motion - Mathematical Analysis 321
DMC 500/DMC 1300 Comparison 324
DMC 500/DMC 1300 Command Comparison 325
List of Other Publications 329
Contacting Us 329
WARRANTY 330
Using This Manual ii

Index **331**

Chapter 1 Overview

Introduction

The DMC 1300 series motion controller is a state-of-the-art motion controller that plugs into the VME Bus. Performance capability of the DMC 1300 series controllers includes: 8 MHz encoder input frequency, 16-bit motor command output DAC, +/-2 billion counts total travel per move, sample rate at up to 125 usec/axis, 16-bit Dual Port RAM, bus interrupts and non-volatile memory for parameter storage. These controllers provide high performance and flexibility while maintaining ease of use and low cost.

Designed for maximum system flexibility, the DMC 1300 is available for one, two, three or four axes configuration per card. An add-on card is available for control of five, six, seven or eight axes. The DMC 1300 can be interfaced to a variety of motors and drives including step motors, servo motors and hydraulic systems.

Each axis accepts feedback from a quadrature linear or rotary encoder with input frequencies up to 8 million quadrature counts per second. For dual-loop applications in which an encoder is required on both the motor and the load, auxiliary encoder inputs are included for each axis.

The DMC 1300 provides many modes of motion, including jogging, point-to-point positioning, linear and circular interpolation, electronic gearing and user-defined path following. Several motion parameters can be specified including acceleration and deceleration rates and slew speed. The DMC 1300 also provides S-curve acceleration for motion smoothing.

For synchronizing motion with external events, the DMC 1300 includes 8 optoisolated inputs, 8 programmable outputs and 7 analog inputs. 24 inputs and 16 programmable outputs are available for five through eight axes. Event triggers can automatically check for elapsed time, distance and motion complete.

Despite its full range of sophisticated features, the DMC 1300 is easy to program. Commands may be send to the controller in either Binary or ASCII format. ASCII instructions are represented by two letter commands such as BG to begin motion and SP to set motion speed. Conditional Instructions, Jump Statements, and Arithmetic Functions are included for writing self-contained applications programs.

The DMC 1300 provides several error handling features. These include software and hardware limits, automatic shut-off on excessive error, abort input, and user-definable error and limit routines. In addition, the DMC 1300 has a full range of VME Bus interrupts.

Overview of Motor Types

The DMC 1300 can provide the following types of motor control:

1. Standard servo motors with +/- 10 volt command signals
2. Step motors with step and direction signals
3. Other actuators such as hydraulics - For more information, contact Galil.

The user can configure each axis for any combination of motor types, providing maximum flexibility.

Standard Servo Motors with +/- 10 Volt Command Signal

The DMC 1300 achieves superior precision through use of a 16-bit motor command output DAC and a sophisticated PID filter that features velocity and acceleration feedforward, an extra pole filter and integration limits.

The controller is configured by the factory for standard servo motor operation. In this configuration, the controller provides an analog signal (+/- 10Volt) to connect to a servo amplifier. This connection is described in Chapter 2.

Stepper Motor with Step and Direction Signals

The DMC 1300 can control stepper motors. In this mode, the controller provides two signals to connect to the stepper motor: Step and Direction. For stepper motor operation, the controller does not require an encoder and operates the stepper motor in an open loop fashion. Chapter 2 describes the proper connection and procedure for using stepper motors.

DMC 1300 Functional Elements

The DMC 1300 circuitry can be divided into the following functional groups as shown in Figure 1.1 and discussed below.

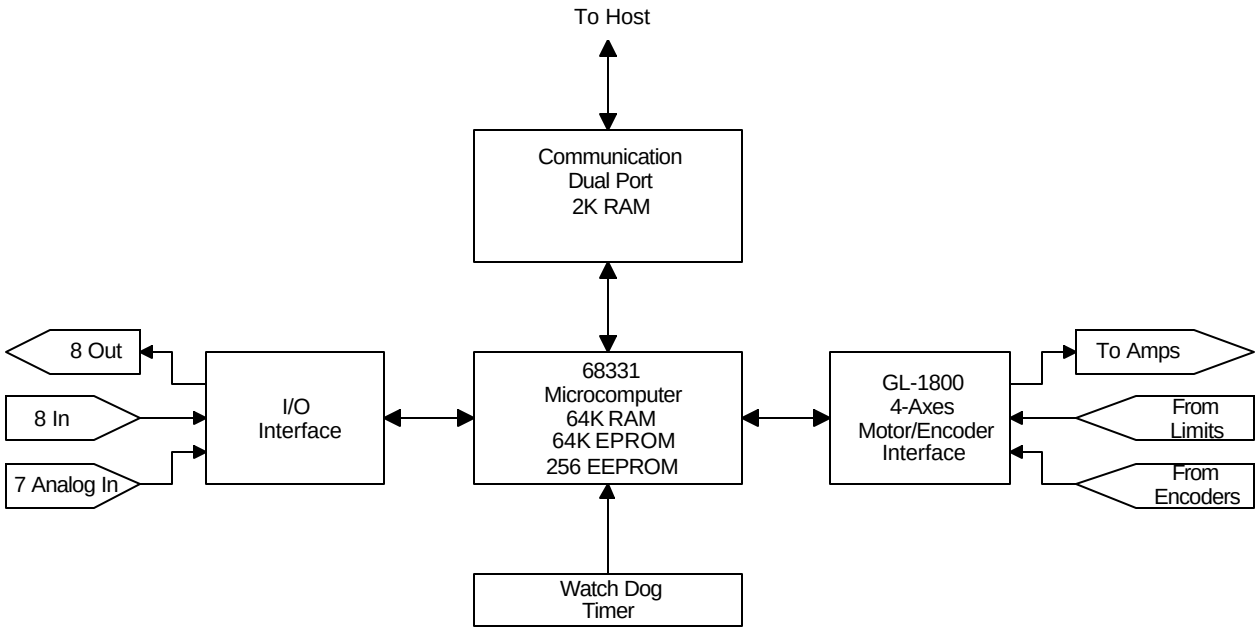


Figure 1.1 - DMC 1300 Functional Elements

Microcomputer Section

The main processing unit of the DMC 1300 is a specialized 32-bit Motorola 68331 Series Microcomputer with 64K RAM (256K available as an option), 64K EPROM and 256 bytes EEPROM. The RAM provides memory for variables, array elements and application programs. The EPROM stores the firmware of the DMC 1300. The EEPROM allows certain parameters and application programs to be saved in non-volatile memory upon power down.

Motor Interface

For each axis, a GL-1800 custom, sub-micron gate array performs quadrature decoding of the encoders at up to 8 MHz, generates a +/-10 Volt analog signal (16 Bit D-to-A) for input to a servo amplifier, and generates step and direction signal for step motor drivers.

Communication

The DMC 1300 uses a Dual Port RAM for communication. This controller resides in the 16-bit VME short I/O space, with 2 byte wide data transfers through the 2K Dual Port RAM (ID77133). The default base address of the controller is F000, with address jumpers A12 - A15 available to select a specific address.

General I/O

The DMC 1300 provides interface circuitry for eight optoisolated inputs, eight general outputs and seven analog inputs (12-Bit ADC).

1380

Controllers with 5 or more axes provide 24 inputs and 16 outputs.

System Elements

As shown in Fig. 1.2, the DMC 1300 is part of a motion control system which includes amplifiers, motors and encoders. These elements are described below.

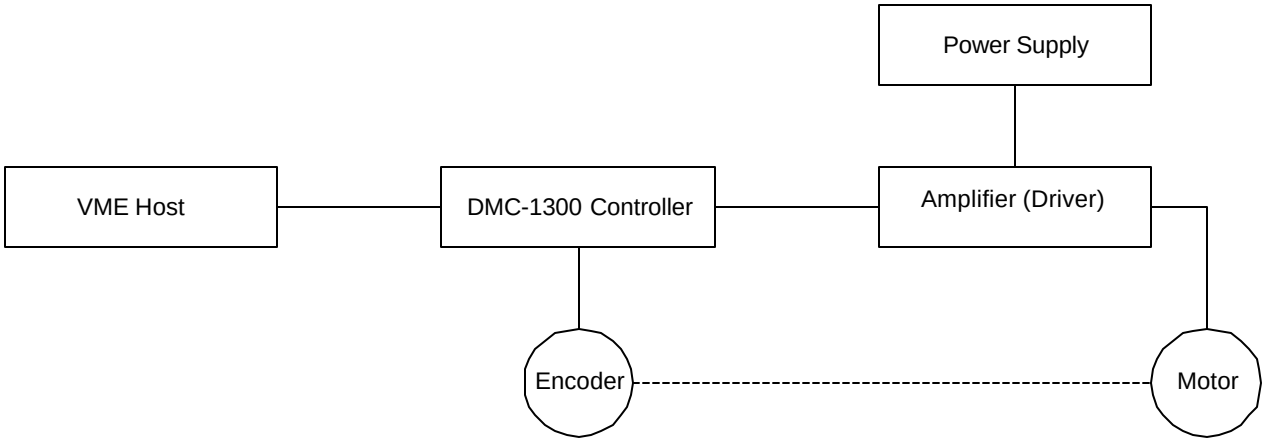


Figure 1.2 - Elements of Servo systems

Motor

A motor converts current into torque which produces motion. Each axis of motion requires a motor sized properly to move the load at the desired speed and acceleration. Galil's Motion Component Selector software can help you calculate motor size and drive size requirements. Contact Galil at 800-377-6329 if you would like this product.

The motor may be a step or servo motor and can be brush-type or brushless, rotary or linear. For step motors, the controller can control full-step, half-step, or microstep drives.

Amplifier (Driver)

For each axis, the power amplifier converts a +/-10 Volt signal from the controller into current to drive the motor. The amplifier should be sized properly to meet the power requirements of the motor. For brushless motors, an amplifier that provides electronic commutation is required. The amplifiers may be either pulse-width-modulated (PWM) or linear. They may also be configured for operation with or without a tachometer. For current amplifiers, the amplifier gain should be set such that a 10 Volt command generates the maximum required current. For example, if the motor peak current is 10A, the amplifier gain should be 1 A/V. For velocity mode amplifiers, 10 Volts should run the motor at the maximum speed.

For stepper motors, the amplifier converts step and direction signals into current.

Encoder

An encoder translates motion into electrical pulses which are fed back into the controller. The DMC 1300 accepts feedback from either a rotary or linear encoder. Typical encoders provide two channels in quadrature, known as CHA and CHB. This type of encoder is known as a quadrature encoder. Quadrature encoders may be either single-ended (CHA and CHB) or differential (CHA,CHA-,CHB,CHB-). The DMC 1300 decodes either type into quadrature states or four times the number of cycles. Encoders may also have a third channel (or index) for synchronization.

The DMC 1300 can also interface to encoders with pulse and direction signals.

There is no limit on encoder line density, however, the input frequency to the controller must not exceed 2,000,000 full encoder cycles/second or 8,000,000 quadrature counts/sec. For example, if the encoder line density is 10,000 cycles per inch, the maximum speed is 200 inches/second.

The standard voltage level is TTL (zero to five volts), however, voltage levels up to 12 Volts are acceptable. If using differential signals, 12 Volts can be input directly to the DMC 1300. Single-ended 12 Volt signals require a bias voltage input to the complementary inputs.

To interface with other types of position sensors such as resolvers or absolute encoders, Galil can customize an expanded I/O board and DMC 1300 command set. Please contact Galil to talk to one of our applications engineers about your particular system requirements.

Watch Dog Timer

The DMC 1300 provides an internal watch dog timer which checks for proper microprocessor operation. The timer toggles the Amplifier Enable Output (AEN) which can be used to switch the amplifiers off in the event of a serious DMC 1300 failure. The AEN output is normally high. During power-up and if the microprocessor ceases to function properly, the AEN output will go low. The error light for each axis will also turn on at this stage. A reset is required to restore the DMC 1300 to normal operation. A hardware interrupt may also be configured to notify the VME host of a watch dog timer occurrence. Hardware interrupts are discussed in more detail in Chapter 4. Consult the factory for a Return Materials Authorization (RMA) Number if your DMC 1300 is damaged.

Chapter 2 Getting Started

Elements You Need

Before you start, you will need the following system elements:

1. DMC 1300 Motion Controller and included 60-pin ribbon cable. Also included is a 26-pin ribbon cable for general I/O.

1a. For stepper motor operation, you will need an additional 20-pin ribbon cable for J4.

2. Servo motors with Optical Encoder (one per axis) or step motors
3. Power Amplifiers
4. Power Supply for Amplifiers
5. VME Bus host system with VME interface software
6. BIT 3's "PC to VME Adapter System" with PC and Galil Comm-1300 software (Optional, but strongly recommended).
7. An Interface Module (Optional, but strongly recommended). The Galil ICM-1100 is an interconnect module with screw type terminals that directly interfaces to the DMC 1300 controller. Note: An additional ICM-1100 is required for the DMC-1350 through DMC-1380.

The motors may be servo (brush type or brushless) or steppers. The amplifiers should be suitable for the motor and may be linear or pulse-width-modulated. An amplifier may have current feedback or voltage feedback.

For servo motors, the amplifiers should accept an analog signal in the +/-10 Volt range as a command. The amplifier gain should be set so that a +10V command will generate the maximum required current. For example, if the motor peak current is 10A, the amplifier gain should be 1 A/V. For velocity mode amplifiers, a command signal of 10 Volts should run the motor at the maximum required speed.

For step motors, the amplifiers should accept step and direction signals. For start-up of a step motor system refer to "Connecting Step Motors" on page **Error! Bookmark not defined.**.

Installing the DMC 1300

The DMC 1300 is a VME card that requires users to be familiar with VME system protocol and/or programming. The following section describes the steps for installing, communicating with and developing the DMC 1300 system.

There are two options available for interfacing to a VME system:

1. Write custom interface software for the VME host. C-drivers are available for the Galil controller to help in this development. Chapter 4 of this manual describes in detail all the DMC 1300 address registers needed for a custom host program. This approach requires familiarity with both the VME system protocol and programming.

OR

2. Use a PC to VME adapter system, such as the Model 406-202 from BIT 3 (Phone 612-881-6955). This system will substitute a PC for the VME host, allowing for quick and easy development. The Galil Comm-1300 software may be used with this setup, which includes the basic terminal emulator, interface access to the Dual Port RAM and development tools for tuning servo motors. This approach allows for a faster system setup, and is useful in prototyping applications.

Installation of a complete, operational DMC 1300 system can be described in 9 steps.

Step 1. Determine overall motor configuration.

Step 2. Configure address jumpers on the DMC 1300.

Step 3. Install the DMC 1300 into the VME host.

Step 4. Install and test communications software.

Step 5. Connect amplifiers and Encoders.

Step 6a. Connect standard servo motors.

Step 6b. Connect step motors.

Step 7. Tune the servo system

Step 1. Determine Overall Motor Configuration

Before setting up the motion control system, the user must determine the desired motor configuration. The DMC 1300 can control any combination of standard servo motors, and stepper motors. Other types of actuators, such as hydraulics, can also be controlled. Please consult Galil for more information.

The following configuration information is necessary to determine the proper motor configuration:

Standard Servo Motor Operation:

The DMC 1300 has been setup by the factory for standard servo motor operation providing an analog command signal of +/- 10V. No hardware or software configuration is required for standard servo motor operation.

Stepper Motor Operation:

To configure the DMC 1300 for stepper motor operation, the controller requires a jumper for each stepper motor and the command, MT, must be given.

The DMC 1300 has jumpers on the board which may need to be installed for stepper motor operation. The following describes each of the jumpers.

For each axis that will be driving a stepper motor, a stepper mode (SM) jumper must be connected.

1380

If you using a controller with more than 4 axis, you will have two VME cards residing on the backplane. In this case, you will have 2 sets of stepper motor jumpers, one on each card. The jumpers on the first card will be for axes X,Y,Z and W (or A,B,C, and D) and the second will be E,F,G and H.

The stepper mode jumpers are located next to the GL-1800 which is the largest IC on the board. The jumper set is labeled JP20 and the individual stepper mode jumpers are labeled SMX, SMY, SMZ, SMW. The fifth jumper of the set, OPT, is for use by Galil technicians only. Further instruction for stepper motor connections are discussed in Step 8b.

The jumper set, JP9, can be used to connect the controllers internal power supply to the optoisolation inputs. This may be desirable if your system will be using limit switches, home inputs digital inputs, or hardware abort **and** optoisolation is not necessary for your system. For a further explanation, see section *Bypassing the Opto-Isolation* in Chapter 3.

Step 2. Configure Address Jumpers on the DMC 1300

The DMC 1300 is installed directly into the VME backplane. The address jumpers of the controller must be set for proper communication with the host. If using the BIT 3 system, address jumpers must also be set on both the PC card and VME card. The procedures for both setups are outlined below.

BIT 3 System Interface

In order to communicate with the DMC 1300 using the Bit 3 system, jumpers must be installed on the controller, Bit 3 VME card and Bit 3 PC card. Setting the address jumpers of the Galil controller is identical to the set-up for the custom VME interface, with the default at F0 00. Once this has been accomplished, the Bit 3 VME and PC card are configured as shown on page XX of the appendix.

Custom VME Interface

The first step in communicating with the Galil controller is to set the address jumpers to the proper configuration. These address jumpers can be found at location JP11, labeled as A12 through A15. The default address of the board with no jumpers installed is F0 00. Placing a jumper will make the corresponding bit a zero, while no jumper corresponds to a one. For example, to set the base address to E0 00 hex, the following jumpers would be installed.

A15	**A14**	**A13**	**A12**
1	1	1	0
N	N	N	J

where N means no jumper and J means a jumper is present.

Step 3. Install the DMC 1300 into VME Host

With the address jumpers properly configured, the DMC 1300 may be installed into the VME host. With the custom VME system, this is simply a matter of installing the controller into an available slot and powering up the system. With the Bit 3 system, the process is more in depth. First, the Bit 3 PC adapter card, once properly address, must be inserted into an ISA slot on the host PC. The Bit 3 VME card, also properly addressed, is then installed into the master VME slot (usually the first slot). The Bit 3 cable is then used to connect the PC to the VME host. Finally, the DMC 1300 may be installed into any VME slot. Please refer to the Bit 3 documentation for more specific information.

Step 4. Install and Test Communications Software

The communication software used will depend on the type of VME system being used, Bit 3 or custom system. Both procedures are outlined below.

BIT 3 System Interface

Communication with the Bit 3 system can be established using the Galil Comm 1300 software. This software provides a terminal emulator to send commands to the controller as well as a display of axis and Dual Port RAM status information.

To install this software, insert the Comm 1300 communication disks. In the DOS command prompt, type A:INSTALL <enter> and follow the instructions on the screen. Upon installation, execute the Comm 1300 software by running COMM1300.exe. The screen should now show the terminal emulator as well as status information for the controller axes and the Dual Port RAM.

To test the communication with the controller, type TP <enter> at the command prompt. The position of the corresponding axes should be displayed. There are also various special functions that can be used in this terminal screen such as:

	!UL <file name>	Uploads file to PC from 1300
	!DO <file name>	Downloads file from PC into 1300
	!?	Reports available screen and configuration options
	!BI	Selects binary mode of communication
	!AS	Selects ASCII mode of communication
	!DE	Selects decimal display option
	!HE	Selects hex display option
	!W m,n	Displays contents of address m, m+1, m+2, m+3 as a four byte value. n is the watch number 1, 2 or 3. You can watch up to three groups of data.
Example:	!W 20,1	Watches address 20; #1
	!W 30,2	Watches address 30; #2
	Q	Quits the COMM1300

Custom VME Interface

Communication with a custom VME system will depend on the type of host software being used. Upon powering up the DMC 1300, the first step should be to test communication with the controller. To test this, read the data at address 241 hex above the base address. This should return a 00 hex. Next, write a byte to that address and read the data again. If this was successful, the controller has been properly addressed.

Sending commands to the controller is a fairly detailed process. The procedure for sending commands can be found in Chapter 4.

Step 5. Connect Amplifiers and Encoders.

Once you have established communications between the host and the DMC 1300, you are ready to connect the rest of the motion control system. The motion control system typically consists of an ICM-1100 Interface Module, an amplifier for each axis of motion, and a motor to transform the current from the amplifier into torque for motion. Galil also offers the AMP-11X0 series Interface Modules which are ICM-1100's equipped with servo amplifiers for brush type DC motors.

If you are using an ICM-1100, connect the 100-pin ribbon cable to the DMC 1300 and to the connector located on the AMP-11X0 or ICM-1100 board. The ICM-1100 provides screw terminals for access to the connections described in the following discussion.

1380

Motion Controllers with more than 4 axes require a second ICM-1100 or AMP-11X0 and second 100-pin cable.

System connection procedures will depend on system components and motor types. Any combination of motor types can be used with the DMC 1300.

Here are the first steps for connecting a motion control system:

Step A. Connect the motor to the amplifier *with no connection to the controller*. Consult the amplifier documentation for instructions regarding proper connections. Connect and turn-on the amplifier power supply. If the amplifiers are operating properly, the motor should stand still even when the amplifiers are powered up.

Step B. Connect the amplifier enable signal.

Before making any connections from the amplifier to the controller, you need to verify that the ground level of the amplifier is either floating or at the same potential as earth.

WARNING: When the amplifier ground is not isolated from the power line or when it has a different potential than that of the computer ground, serious damage may result to the computer controller and amplifier.

If you are not sure about the potential of the ground levels, connect the two ground signals (amplifier ground and earth) by a 10 KΩ resistor and measure the voltage across the resistor. Only if the voltage is zero, connect the two ground signals directly.

The amplifier enable signal is used by the controller to disable the motor. It will disable the motor when the watchdog timer activates, the motor-off command, MO, is given, or the position error exceeds the error limit with the "Off-On-Error" function enabled (see the command OE for further information).

The standard configuration of the AEN signal is TTL active high. In other words, the AEN signal will be high when the controller expects the amplifier to be enabled. The polarity and the amplitude can be changed if you are using the ICM-1100 interface board. To change the polarity from active high (5 volts = enable, zero volts = disable) to active low (zero volts = enable, 5 volts = disable), replace the 7407 IC with a 7406. Note that many amplifiers designate the enable input as 'inhibit'.

To change the voltage level of the AEN signal, note the state of the resistor pack on the ICM-1100. When Pin 1 is on the 5V mark, the output voltage is 0-5V. To change to 12 volts, pull the resistor pack and rotate it so that Pin 1 is on the 12 volt side. If you remove the resistor pack, the output signal is an open collector, allowing the user to connect an external supply with voltages up to 24V.

On the ICM-1100, the amplifier enable signal is labeled AENX for the X axis. Connect this signal to the amplifier (figure 2.3) and issue the command, MO, to disable the motor amplifiers - often this is indicated by an LED on the amplifier.

Step C. Connect the encoders

For stepper motor operation, an encoder is optional.

For servo motor operation, if you have a preferred definition of the forward and reverse directions, make sure that the encoder wiring is consistent with that definition.

The DMC 1300 accepts single-ended or differential encoder feedback with or without an index pulse. If you are not using the AMP-11X0 or the ICM-1100 you will need to consult the appendix for the encoder pinouts for connection to the motion controller. The AMP-11X0 and the ICM-1100 can accept encoder feedback from a 10-pin ribbon cable or individual signal leads. For a 10-pin ribbon cable encoder, connect the cable to the protected header connector labeled X ENCODER (repeat for each axis necessary). For individual wires, simply match the leads from the encoder you are using to the encoder feedback inputs on the interconnect board. The signal leads are labeled XA+ (channel A), XB+ (channel B), and XI+. For differential encoders, the complement signals are labeled XA-, XB-, and XI-.

Note: When using pulse and direction encoders, the pulse signal is connected to XA+ and the direction signal is connected to XB+. The controller must be configured for pulse and direction with the command CE. See the command summary for further information on the command CE.

Step D. Verify proper encoder operation.

Start with the X encoder first. Once it is connected, turn the motor shaft and interrogate the position with the instruction TPX <return>. The controller response will vary as the motor is turned.

At this point, if TPX does not vary with encoder rotation, there are three possibilities:

1. The encoder connections are incorrect - check the wiring as necessary.
2. The encoder has failed - using an oscilloscope, observe the encoder signals. Verify that both channels A and B have a peak magnitude between 5 and 12 volts. Note that if only one encoder channel fails, the position reporting varies by one count only. If the encoder failed, replace the encoder. If you cannot observe the encoder signals, try a different encoder.
3. There is a hardware failure in the controller- connect the same encoder to a different axis. If the problem disappears, you probably have a hardware failure. Consult the factory for help.

Step 6a. Connect Standard Servo Motors

The following discussion applies to connecting the DMC 1300 controller to standard servo motor amplifiers:

The motor and the amplifier may be configured in the torque or the velocity mode. In the torque mode, the amplifier gain should be such that a 10 Volt signal generates the maximum required current. In the velocity mode, a command signal of 10 Volts should run the motor at the maximum required speed.

Step by step directions on servo system setup are also included on the WSDK (Windows Servo Design Kit) software offered by Galil. See section on WSDK for more details.

Step A. Check the Polarity of the Feedback Loop

It is assumed that the motor and amplifier are connected together and that the encoder is operating correctly (Step B). Before connecting the motor amplifiers

to the controller, read the following discussion on the setting Error Limits and Torque Limits. Note that this discussion only uses the X axis for the examples.

Step B. Set the Error Limit as a Safety Precaution

Usually, there is uncertainty about the correct polarity of the feedback. The wrong polarity causes the motor to run away from the starting position. Using a terminal program, such as DMCTERM, the following parameters can be given to avoid system damage:

Input the commands:

ER 2000 <CR> Sets error limit on the X axis to be 2000 encoder counts

OE 1 <CR> Disables X axis amplifier when a excess position error exists

If the motor runs away and creates a position error of 2000 counts, the motor amplifier will be disabled. **Note:** This function requires the AEN signal to be connected from the controller to the amplifier.

Step C. Set Torque Limit as a Safety Precaution

To limit the maximum voltage signal to your amplifier, the DMC 1300 controller has a torque limit command, TL. This command sets the maximum voltage output of the controller and can be used to avoid excessive torque or speed when initially setting up a servo system.

When operating an amplifier in torque mode, the voltage output of the controller will be directly related to the torque output of the motor. The user is responsible for determining this relationship using the documentation of the motor and amplifier. The torque limit can be set to a value that will limit the motors output torque.

When operating an amplifier in velocity or voltage mode, the voltage output of the controller will be directly related to the velocity of the motor. The user is responsible for determining this relationship using the documentation of the motor and amplifier. The torque limit can be set to a value that will limit the speed of the motor.

For example, the following command will limit the output of the controller to 1 volt on the X axis:

TL 1 <CR>

Note: Once the correct polarity of the feedback loop has been determined, the torque limit should, in general, be increased to the default value of 9.99. The servo will not operate properly if the torque limit is below the normal operating range. See description of TL in the command reference.

Step D. Connect the Motor

Once the parameters have been set, connect the analog motor command signal (ACMD) to the amplifier input.

To test the polarity of the feedback, command a move with the instruction:

PR 1000 <CR> Position relative 1000 counts

BGX <CR> Begin motion on X axis

When the polarity of the feedback is wrong, the motor will attempt to run away. The controller should disable the motor when the position error exceeds 2000 counts. If the motor runs away, the polarity of the loop must be inverted.

Note: Inverting the Loop Polarity

When the polarity of the feedback is incorrect, the user must invert the loop polarity and this may be accomplished by several methods. If you are driving a brush-type DC motor, the simplest way is to invert the two motor wires (typically red and black). For example, switch the M1 and M2 connections going from your amplifier to the motor. When driving a brushless motor, the polarity reversal may be done with the encoder. If you are using a single-ended encoder, interchange the signal CHA and CHB. If, on the other hand, you are using a differential encoder, interchange only CHA+ and CHA-. The loop polarity and encoder polarity can also be affected through software with the MT, and CE commands. For more details on the MT command or the CE command, see the Command Reference section.

Note: Reversing the Direction of Motion

If the feedback polarity is correct but the direction of motion is opposite to the desired direction of motion, reverse the motor leads AND the encoder signals.

When the position loop has been closed with the correct polarity, the next step is to adjust the PID filter parameters, KP, KD and KI. It is necessary to accurately tune your servo system to ensure fidelity of position and minimize motion oscillation as described in the next section.

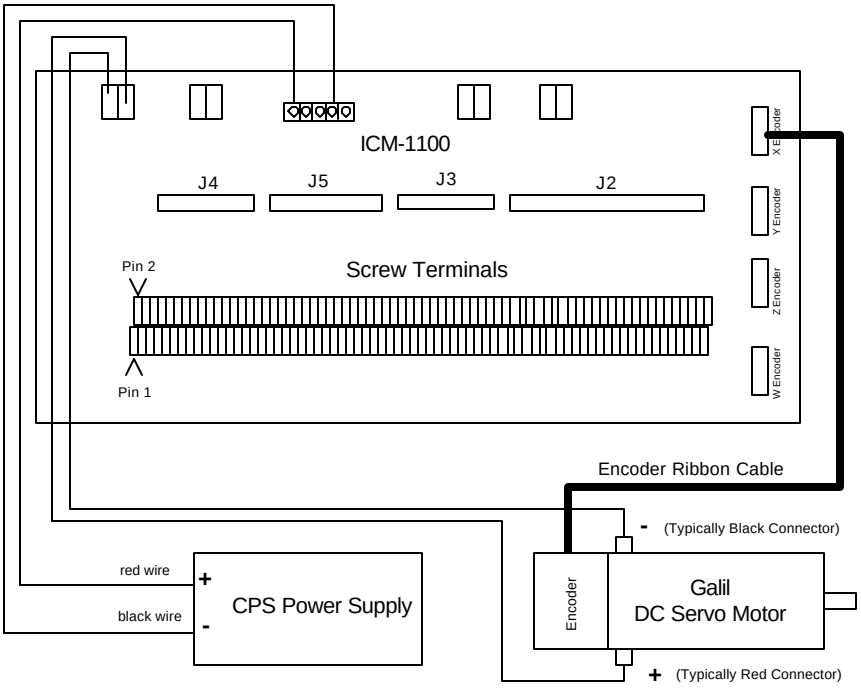


Figure 2-2 - System Connections with the AMP-1100Amplifier. Note: this figure shows a Galil Motor and Encoder which uses a flat ribbon cable to connect to the AMP-1100 unit.

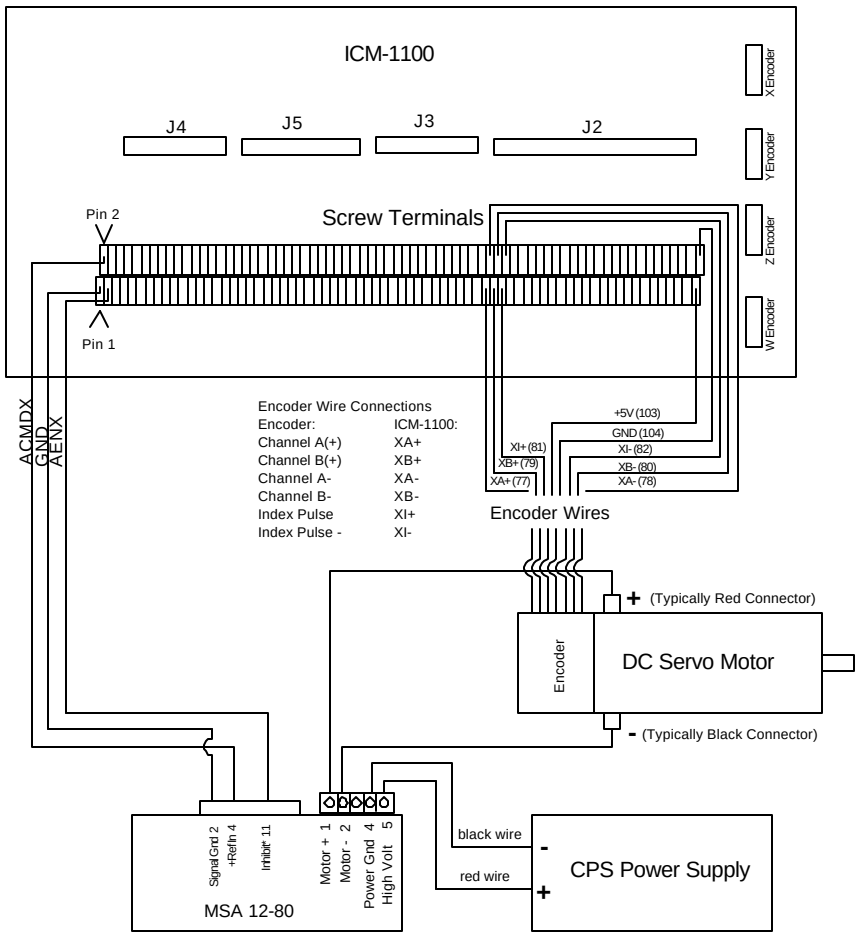


Figure 2-3 System Connections with a separate amplifier (MSA 12-80). This diagram shows the connections for a standard DC Servo Motor and encoder.

Step 6b. Connect Step Motors

In Stepper Motor operation, the pulse output signal has a 50% duty cycle. Step motors operate open loop and do not require encoder feedback. When a stepper is used, the auxiliary encoder for the corresponding axis is unavailable for an external connection. If an encoder is used for position feedback, connect the encoder to the main encoder input corresponding to that axis. The commanded position of the stepper can be interrogated with RP or DE. The encoder position can be interrogated with TP.

The frequency of the step motor pulses can be smoothed with the filter parameter, KS. The KS parameter has a range between 0.5 and 8, where 8 implies the largest amount of smoothing. *See Command Reference regarding KS.*

The DMC 1300 profiler commands the step motor amplifier. All DMC 1300 motion commands apply such as PR, PA, VP, CR and JG. The acceleration, acceleration, slew speed and smoothing are also

used. Since step motors run open-loop, the PID filter does not function and the position error is not generated.

To connect step motors with the DMC 1300 you must follow this procedure:

Step A. Install SM jumpers

Each axis of the DMC 1300 that will operate a stepper motor must have the corresponding stepper motor jumper installed. For a discussion of SM jumpers, see step 2.

Step B. Connect step and direction signals.

Make connections from controller to motor amplifiers. (These signals are labeled PULSX and DIRX for the x-axis on the ICM-1100). Consult the documentation for your step motor amplifier.

Step C. Configure DMC 1300 for motor type using MT command. You can configure the DMC 1300 for active high or active low pulses. Use the command MT 2 for active high step motor pulses and MT -2 for active low step motor pulses. *See description of the MT command in the Command Reference.*

Step 7. Tune the Servo System

Adjusting the tuning parameters for the servo motors is required when using servo motors. The system compensation provides fast and accurate response by adjusting the filter parameters. The following presentation suggests a simple and easy way for compensation.

The filter has three parameters: the damping, KD; the proportional gain, KP; and the integrator, KI. The parameters should be selected in this order.

To start, set the integrator to zero with the instruction

KI 0 (CR) Integrator gain

and set the proportional gain to a low value, such as

KP 1 (CR) Proportional gain

KD 100 (CR) Derivative gain

For more damping, you can increase KD (maximum is 4095). Increase gradually and stop after the motor vibrates. A vibration is noticed by audible sound or by interrogation. If you send the command

TE X (CR) Tell error

a few times, and get varying responses, especially with reversing polarity, it indicates system vibration. When this happens, simply reduce KD.

Next you need to increase the value of KP gradually (maximum allowed is 1023). You can monitor the improvement in the response with the Tell Error instruction

KP 10 (CR) Proportion gain

TE X (CR) Tell error

As the proportional gain is increased, the error decreases.

Again, the system may vibrate if the gain is too high. In this case, reduce KP. Typically, KP should not be greater than KD/4. (Only when the amplifier is configured in the current mode).

Finally, to select KI, start with zero value and increase it gradually. The integrator eliminates the position error, resulting in improved accuracy. Therefore, the response to the instruction

TE X (CR)

becomes zero. As KI is increased, its effect is amplified and it may lead to vibrations. If this occurs, simply reduce KI. Repeat tuning for the Y, Z and W axes.

For a more detailed description of the operation of the PID filter and/or servo system theory, see Chapter 10 - Theory of Operation.

Design Examples

Here are a few examples for tuning and using your controller. These examples have remarks next to each command - these remarks must not be included in the actual program.

Example 1 - System Set-up

This example assigns the system filter parameters, error limits and enables the automatic error shut-off.

INSTRUCTION	INTERPRETATION
KP10,10,10,10,10,10,10,10	Set gains for a,b,c,d,e,f,g,and h axes
KP10,10,10,10,10,10,10,10	Set gains for a,b,c,d,e,f,g,and h axes
KP*=10	Alternate method for setting gain on all axes
KPX=10	Alternate method for setting X (or A) axis gain
KPA=10	Alternate method for setting A (or X) axis gain

1380

When using controllers with 5 or more axes, the X,Y,Z and W axes can also be referred to as the A,B,C,D axes.

INSTRUCTION	INTERPRETATION
OE 1,1,1,1,1,1,1,1	Enable automatic Off on Error function for all axes
ER*=1000	Set error limit for all axes to 1000 counts
KP10,10,10,10,10,10,10,10	Set gains for a,b,c,d,e,f,g,and h axes
KP*=10	Alternate method for setting gain on all axes
KPX=10	Alternate method for setting X (or A) axis gain
KPA=10	Alternate method for setting A (or X) axis gain
KPZ=10	Alternate method for setting Z axis gain
KPD=10	Alternate method for setting D axis gain
KPH=10	Alternate method for setting H axis gain

Example 2 - Profiled Move

Objective: Rotate the X axis a distance of 10,000 counts at a slew speed of 20,000 counts/sec and an acceleration and deceleration rates of 100,000 counts/s2. In this example, the motor turns and stops:

INSTRUCTION	INTERPRETATION
PR 10000	Distance

SP 20000	Speed
DC 100000	Deceleration
AC 100000	Acceleration
BG X	Start Motion

Example 3 - Multiple Axes

Objective: Move the four axes independently.

INSTRUCTION	INTERPRETATION
PR 500,1000,600,-400	Distances of X,Y,Z,W
SP 10000,12000,20000,10000	Slew speeds of X,Y,Z,W
AC 100000,10000,100000,100000	Accelerations of X,Y,Z,W
DC 80000,40000,30000,50000	Decelerations of X,Y,Z,W
BG XZ	Start X and Z motion
BG YW	Start Y and W motion

Example 4 - Independent Moves

The motion parameters may be specified independently as illustrated below.

INSTRUCTION	INTERPRETATION
PR ,300,-600	Distances of Y and Z
SP ,2000	Slew speed of Y
DC ,80000	Deceleration of Y
AC, 100000	Acceleration of Y
SP ,,40000	Slew speed of Z
AC ,,100000	Acceleration of Z
DC ,,150000	Deceleration of Z
BG Z	Start Z motion
BG Y	Start Y motion

Example 5 - Position Interrogation

The position of the four axes may be interrogated with the instruction, TP.

INSTRUCTION	INTERPRETATION
TP	Tell position all four axes
TP X	Tell position - X axis only
TP Y	Tell position - Y axis only
TP Z	Tell position - Z axis only

TP W	Tell position - W axis only

The position error, which is the difference between the commanded position and the actual position can be interrogated with the instruction TE.

INSTRUCTION	INTERPRETATION
TE	Tell error - all axes
TE X	Tell error - X axis only
TE Y	Tell error - Y axis only
TE Z	Tell error - Z axis only
TE W	Tell error - W axis only

Example 6 - Absolute Position

Objective: Command motion by specifying the absolute position.

INSTRUCTION	INTERPRETATION
DP 0,2000	Define the current positions of X,Y as 0 and 2000
PA 7000,4000	Sets the desired absolute positions
BG X	Start X motion
BG Y	Start Y motion

After both motions are complete, the X and Y axes can be command back to zero:

PA 0,0	Move to 0,0
BG XY	Start both motions

Example 7 - Velocity Control

Objective: Drive the X and Y motors at specified speeds.

INSTRUCTION	INTERPRETATION
JG 10000,-20000	Set Jog Speeds and Directions
AC 100000, 40000	Set accelerations
DC 50000,50000	Set decelerations
BG XY	Start motion

after a few seconds, send the following command:

JG -40000	New X speed and Direction
TV X	Returns X speed

and then

JG ,20000	New Y speed

TV Y	Returns Y speed

These cause velocity changes including direction reversal. The motion can be stopped with the instruction

ST	Stop

Example 8 - Operation Under Torque Limit

The magnitude of the motor command may be limited independently by the instruction TL.

INSTRUCTION	INTERPRETATION
TL 0.2	Set output limit of X axis to 0.2 volts
JG 10000	Set X speed
BG X	Start X motion

In this example, the X motor will probably not move since the output signal will not be sufficient to overcome the friction. If the motion starts, it can be stopped easily by a touch of a finger.

Increase the torque level gradually by instructions such as

INSTRUCTION	INTERPRETATION
TL 1.0	Increase torque limit to 1 volt.
TL 9.98	Increase torque limit to maximum, 9.98 Volts.

The maximum level of 10 volts provides the full output torque.

Example 9 - Interrogation

The values of the parameters may be interrogated. Some examples …

INSTRUCTION	INTERPRETATION
KP ?	Return gain of X axis.
KP ,,?	Return gain of Z axis.
KP ?,?,?,?	Return gains of all axes.

Many other parameters such as KI, KD, FA, can also be interrogated. The command reference denotes all commands which can be interrogated.

Example 10 - Operation in the Buffer Mode

The instructions may be buffered before execution as shown below.

INSTRUCTION	INTERPRETATION
PR 600000	Distance
SP 10000	Speed
WT 10000	Wait 10000 milliseconds before reading the next instruction
BG X	Start the motion

Example 11 - Motion Programs

Motion programs may be edited and stored in the controllers on-board memory.

The instruction

ED	Edit mode

moves the operation to the editor mode where the program may be written and edited. The editor provides the line number. For example, in response to the first ED command, the first line is zero.

LINE #	INSTRUCTION	INTERPRETATION
000	#A	Define label
001	PR 700	Distance
002	SP 2000	Speed
003	BGX	Start X motion
004	EN	End program

To exit the editor mode, input <cntrl>Q. The program may be executed with the command.

XQ #A	Start the program running

Example 12 - Motion Programs with Loops

Motion programs may include conditional jumps as shown below.

INSTRUCTION	INTERPRETATION
#A	Label
DP 0	Define current position as zero
V1=1000	Set initial value of V1
#Loop	Label for loop
PA V1	Move X motor V1 counts
BG X	Start X motion
AM X	After X motion is complete
WT 500	Wait 500 ms
TP X	Tell position X
V1=V1+1000	Increase the value of V1
JP #Loop,V1<10001	Repeat if V1<10001
EN	End

After the above program is entered, quit the Editor Mode, <cntrl>Q. To start the motion, command:

XQ #A	Execute Program #A

Example 13 - Motion Programs with Trippoints

The motion programs may include trippoints as shown below.

INSTRUCTION	INTERPRETATION

#B	Label
DP 0,0	Define initial positions
PR 30000,60000	Set targets
SP 5000,5000	Set speeds
BGX	Start X motion
AD 4000	Wait until X moved 4000
BGY	Start Y motion
AP 6000	Wait until position X=6000
SP 2000,50000	Change speeds
AP ,50000	Wait until position Y=50000
SP ,10000	Change speed of Y
EN	End program

To start the program, command:

XQ #B	Execute Program #B

Example 14 - Control Variables

Objective: To show how control variables may be utilized.

INSTRUCTION	INTERPRETATION
#A;DP0	Label; Define current position as zero
PR 4000	Initial position
SP 2000	Set speed
BGX	Move X
AMX	Wait until move is complete
WT 500	Wait 500 ms
#B	
V1 = _TPX	Determine distance to zero
PR -V1/2	Command X move 1/2 the distance
BGX	Start X motion
AMX	After X moved
WT 500	Wait 500 ms
V1=	Report the value of V1
JP #C, V1=0	Exit if position=0
JP #B	Repeat otherwise
#C	Label #C
EN	End of Program

To start the program, command

XQ #A	Execute Program #A

This program moves X to an initial position of 1000 and returns it to zero on increments of half the distance. Note, _TPX is an internal variable which returns the value of the X position. Internal variables may be created by preceding a DMC 1300 instruction with an underscore, _.

Example 15 - Linear Interpolation

Objective: Move X,Y,Z motors distance of 7000,3000,6000, respectively, along linear trajectory. Namely, motors start and stop together.

INSTRUCTION	INTERPRETATION
LM XYZ	Specify linear interpolation axes
LI 7000,3000,6000	Relative distances for linear interpolation
LE	Linear End
VS 6000	Vector speed
VA 20000	Vector acceleration
VD 20000	Vector deceleration
BGS	Start motion

Example 16 - Circular Interpolation

Objective: Move the XY axes in circular mode to form the path shown on Fig. 2-4. Note that the vector motion starts at a local position (0,0) which is defined at the beginning of any vector motion sequence. See application programming for further information.

INSTRUCTION	INTERPRETATION
VM XY	Select XY axes for circular interpolation
VP -4000,0	Linear segment
CR 2000,270,-180	Circular segment
VP 0,4000	Linear segment
CR 2000,90,-180	Circular segment
VS 1000	Vector speed
VA 50000	Vector acceleration
VD 50000	Vector deceleration
VE	End vector sequence
BGS	Start motion

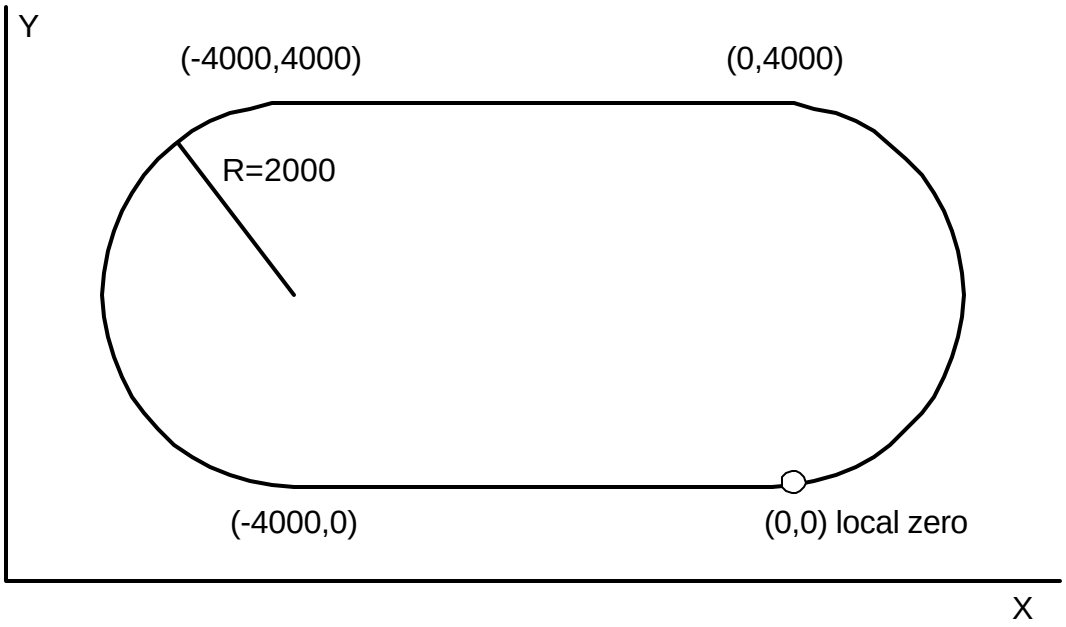


Figure 2-4 Motion Path for Example 16

Chapter 3 Connecting Hardware

Overview

The DMC 1300 provides optoisolated digital inputs for forward limit, reverse limit, home, and abort signals. The controller also has 8 optoisolated, uncommitted inputs (for general use) as well as 8 TTL outputs and 7 analog inputs configured for voltages between +/- 10 volts.

1380

Controllers with 5 or more axes have an additional 16 TTL level inputs and 8 TTL level outputs.

This chapter describes the inputs and outputs and their proper connection.

To access the analog inputs or general inputs 5-8 or all outputs except OUT1, connect the 26-pin ribbon cable to the 26-pin J5 IDC connector from the DMC 1300 to the AMP-11X0 or ICM-1100 board.

If you plan to use the auxiliary encoder feature of the DMC 1300, you must also connect a 20-pin ribbon cable from the 20-pin J3 header connector on the DMC 1300 to the 26-pin J3 header connector on the AMP-11X0 or ICM-1100. This cable is not shipped unless requested when ordering.

Using Optoisolated Inputs

Limit Switch Input

The forward limit switch (FLSx) inhibits motion in the forward direction immediately upon activation of the switch. The reverse limit switch (RLSx) inhibits motion in the reverse direction immediately upon activation of the switch. If a limit switch is activated during motion, the controller will make a decelerated stop using the deceleration rate previously set with the DC command. The motor will remain in a servo state after the limit switch has been activated and will hold motor position.

When a forward or reverse limit switch is activated, the current application program that is running will be interrupted and the controller will automatically jump to the #LIMSWI subroutine if one exists. This is a subroutine which the user can include in any motion control program and is useful for executing specific instructions upon activation of a limit switch.

After a limit switch has been activated, further motion in the direction of the limit switch will not be possible until the logic state of the switch returns back to an inactive state. This usually involves physically opening the tripped switch. Any attempt at further motion before the logic state has been reset will result in the following error: “022 - Begin not possible due to limit switch” error.

The operands, _LFx and _LRx, return the state of the forward and reverse limit switches, respectively (x represents the axis, X,Y,Z,W etc.). The value of the operand is either a '0' or '1' corresponding to the logic state of the limit switch. Using a terminal program, the state of a limit switch can be printed to the screen with the command, MG _LFx or MG _LFx. This prints the value of the limit switch operands for the 'x' axis. The logic state of the limit switches can also be interrogated with the TS command. For more details on TS see the Command Reference.

The state of the forward and reverse limit switches can also be read directly through the Dual Port RAM. Bit 3 of the Switches address in the Axis Buffer indicates the status of the forward limit switch on an axis, while Bit 2 of that address indicates the status of the reverse limit switch. For example, the forward limit switch for the DMC 1340 X-axis is read at Bit 3 of address 105, while the reverse limit switch for the DMC 1380 X-Axis is read at Bit 2 of address 205.

Home Switch Input

The Home inputs are designed to provide mechanical reference points for a motion control application. A transition in the state of a Home input alerts the controller that a particular reference point has been reached by a moving part in the motion control system. A reference point can be a point in space or an encoder index pulse.

The Home input detects any transition in the state of the switch and toggles between logic states 0 and 1 at every transition. A transition in the logic state of the Home input will cause the controller to execute a homing routine specified by the user.

There are three homing routines supported by the DMC 1300: Find Edge (FE), Find Index (FI), and Standard Home (HM).

The Find Edge routine is initiated by the command sequence: FEX <return>, BGX <return>. The Find Edge routine will cause the motor to accelerate, then slew at constant speed until a transition is detected in the logic state of the Home input. The motor will then decelerate to a stop. The acceleration rate, deceleration rate and slew speed are specified by the user, prior to the movement, using the commands AC, DC, and SP. *It is recommended that a high deceleration value be used so the motor will decelerate rapidly after sensing the Home switch.*

The Find Index routine is initiated by the command sequence: FIX <return>, BGX <return>. Find Index will cause the motor to accelerate to the user-defined slew speed (SP) at a rate specified by the user with the AC command and slew until the controller senses a change in the index pulse signal from low to high. The motor then decelerates to a stop at the rate previously specified by the user with the DC command. *Although Find Index is an option for homing, it is not dependent upon a transition in the logic state of the Home input, but instead is dependent upon a transition in the level of the index pulse signal.*

The Standard Homing routine is initiated by the sequence of commands HMX <return>, BGX <return>. Standard Homing is a combination of Find Edge and Find Index homing. Initiating the standard homing routine will cause the motor to slew until a transition is detected in the logic state of the Home input. The motor will accelerate at the rate specified by the command, AC, up to the slew speed. After detecting the transition in the logic state on the Home Input, the motor will decelerate to a stop at the rate specified by the command, DC. After the motor has decelerated to a stop, it switches direction and approaches the transition point at the speed of 256 counts/sec. When the logic state changes again, the motor moves forward (in the direction of increasing encoder count) at the same speed, until the controller senses the index pulse. After detection, it decelerates to a stop and defines this position as 0. The logic state of the Home input can be interrogated with the command MG _HMX. This command returns a 0 or 1 if the logic state is low or high, respectively. The state of the Home input can also be interrogated indirectly with the TS command.

The status of the Home Switch can also be read through the Dual Port RAM. Bits 1, 2 and 3 of the Status #1 address in the Axis Buffer gives the state of the HM command. Bit 1 shows when home has been found, Bit 2 shows when the 1st phase of the homing routine has completed, and Bit 3 shows when the 2nd phase of the homing routine has completed. For example, a 1 at Bit 2 of address 240 on a DMC 1380 indicates that the 1st phase of homing on the Y-axis has completed.

For examples and further information about Homing, see command HM, FI, FE of the Command Reference and the section entitled 'Homing' in the Programming Motion Section of this manual.

Abort Input

The function of the Abort input is to immediately stop the controller upon transition of the logic state.

NOTE: The response of the abort input is significantly different from the response of an activated limit switch. When the abort input is activated, the controller stops generating motion commands immediately, whereas the limit switch response causes the controller to make a decelerated stop.

NOTE: The effect of an Abort input is dependent on the state of the off-on-error function for each axis. If the Off-On-Error function is enabled for any given axis, the motor for that axis will be turned off when the abort signal is generated. This could cause the motor to 'coast' to a stop since it is no longer under servo control. If the Off-On-Error function is disabled, the motor will decelerate to a stop as fast as mechanically possible and the motor will remain in a servo state.

All motion programs that are currently running are terminated when a transition in the Abort input is detected. For information on setting the Off-On-Error function, see the Command Reference, OE.

NOTE: The error LED does not light up when the Abort Input is active.

Uncommitted Digital Inputs

The DMC 1300 has 8 uncommitted opto-isolated inputs. These inputs are specified as INx where x specifies the input number, 1 through 24. These inputs allow the user to monitor events external to the controller. For example, the user may wish to have the x-axis motor move 1000 counts in the positive direction when the logic state of IN1 goes high.

1380

Controllers with 5 or more axes have 16 opto-isolated inputs and 8 TTL level inputs. .

The inputs 9-16 and the limit switch inputs for the additional axes are accessed through the second 26-pin connector, JD 5.

The status of the general purpose inputs can be read in the General Registers of the Dual Port RAM. Address 02A on the DMC 1310/1340 shows the status of the 8 general purpose inputs, while addresses 02A - 02C of the DMC 1350/1380 show the status of the 24 general purpose inputs.

Wiring the Optoisolated Inputs

The default state of the controller configures all inputs to be interpreted as a logic one without any connection. The inputs must be brought low to be interpreted as a zero. With regard to limit switches, a limit switch is considered to be activated when the input is brought low (or a switch is closed to ground). Some inputs can be configured to be active when the input is high - see section *Changing Optoisolated Inputs from Active High to Active Low*.

The optoisolated inputs are organized into groups. For example, the general inputs, IN1-IN8, and the ABORT input are one group. Each group has a common signal which supplies current for the inputs in the group. In order to use an input, the associated common signal must be connected to voltage between +5 and +28 volts, see discussion below.

The optoisolated inputs are connected in the following groups (these inputs are accessed through the 26-pin J5 header).

Group	Common Signal
IN1-IN8, ABORT	INCOM
FLX,RLX,HOMEX	LSCOM
FLY,RLY,HOMEY	
FLZ,RLZ,HOMEZ	
FLW,RLW,HOMEW	

1380

For controllers with more than 4 axes, the inputs 9-16 and the limit switch inputs for the additional axes are accessed through a separate connector, JD5.

Group	Common Signal
IN9-IN16	INCOM
FLE,RLE,HOMEE	LSCOM
FLF,RLF,HOMEF	
FLG,RLG,HOMEG	
FLH,RLH,HOMEH	

A logic zero is generated when at least 1mA of current flows from the common signal to the input. A positive voltage (with respect to the input) must be supplied at the common. This can be accomplished by connecting a voltage in the range of +5V to +28V into INCOM of the input circuitry from a separate power supply

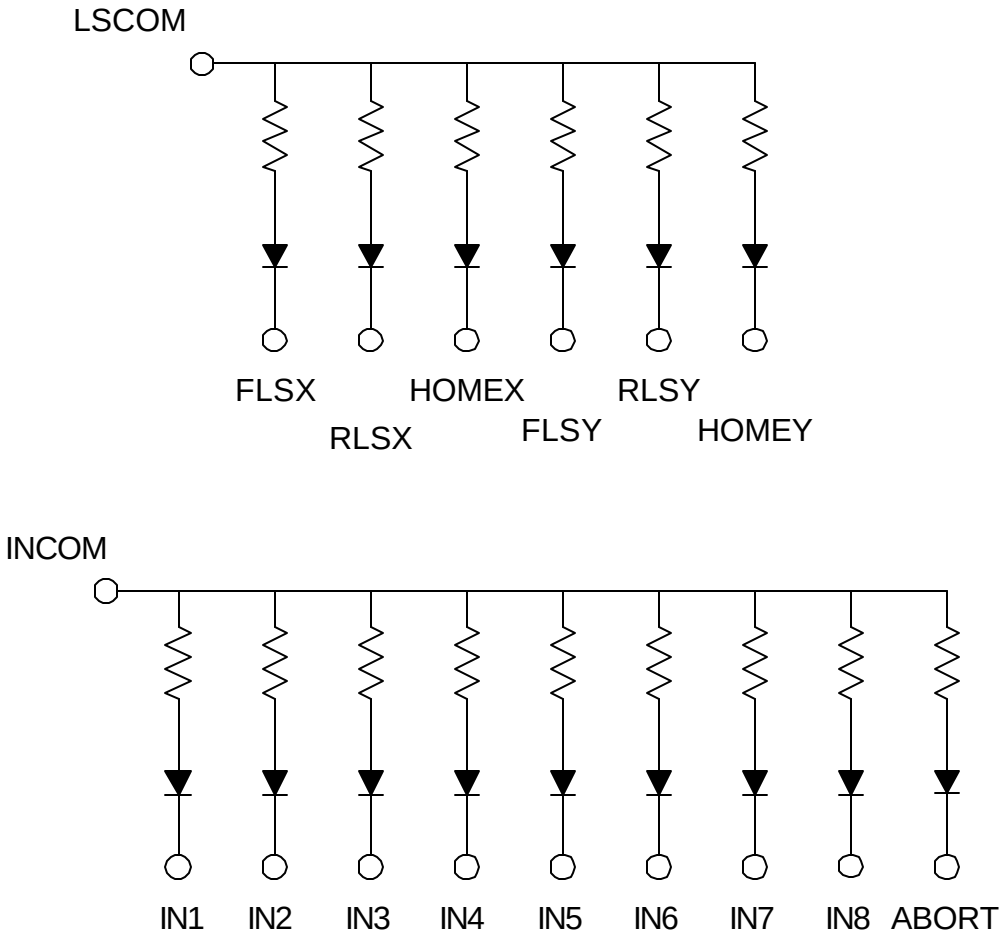


Figure 3-1. The Optoisolated Inputs

Using an Isolated Power Supply

To take full advantage of opto-isolation, an isolated power supply should be used to provide the voltage at the input common connection. When using an isolated power supply, do not connect the ground of the isolated power to the ground of the controller. A power supply in the voltage range between 5 to 28 Volts may be applied directly (see Figure 3-2). For voltages greater than 28 Volts, a resistor, R, is needed in series with the input such that

$$1\ mA < V\ supply/(R + 2.2K\Omega) < 15\ mA$$

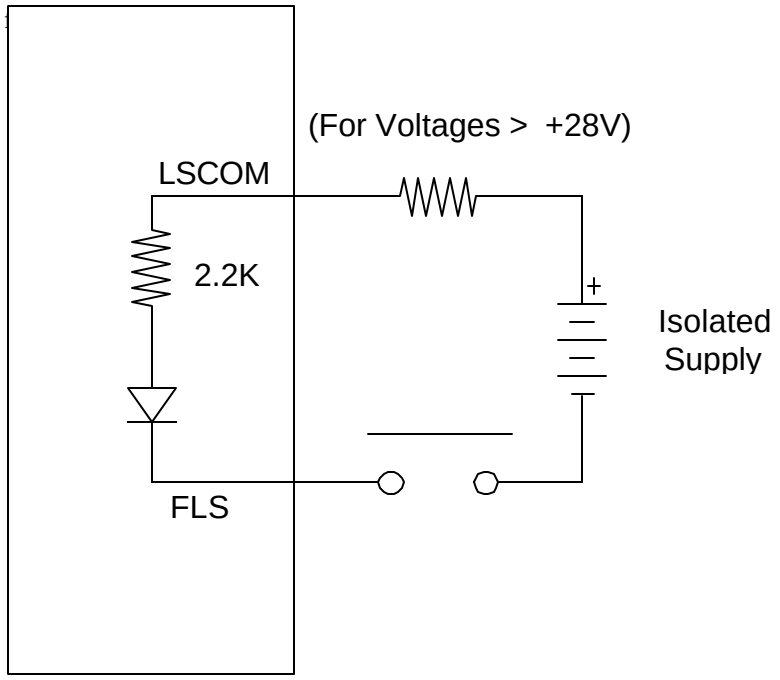


Figure 3-2. Connecting a single Limit or Home Switch to an Isolated Supply

NOTE: As stated in Chapter 2, the wiring is simplified when using the ICM-1100 or AMP-11x0 interface board. This board accepts the signals from the ribbon cables of the DMC 1300 and provides phoenix-type screw terminals. A picture of the ICM-1100 can be seen on pg. 2-14. The user must wire the system directly off the ribbon cable if the ICM-1100 or equivalent breakout board is not available.

Bypassing the Opto-Isolation:

If no isolation is needed, the internal 5 Volt supply may be used to power the switches, as shown in Figure 3-3. This can be done by connecting a jumper between the pins LSCOM or INCOM and 5V, labeled J9. These jumpers can be added on either the ICM-1100 or the DMC 1300. This can also be done by connecting wires between the 5V supply and common signals using the screw terminals on the ICM-1100 or AMP-11x0.

To close the circuit, wire the desired input to any ground (GND) terminal.

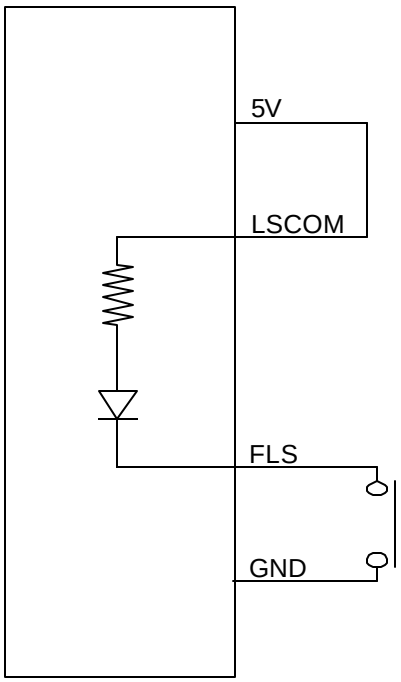


Figure 3-3 - Connecting Limit switches to the internal 5V supply

Changing Optoisolated Inputs From Active Low to Active High

Some users may prefer that the optoisolated inputs be active high. For example, the user may wish to have the inputs be activated with a logic one signal. The limit, home and latch inputs can be configured through software to be active high or low with the CN command. For more details on the CN see Command Reference manual.

The Abort input *cannot* be configured in this manner.

Amplifier Interface

The DMC 1300 analog command voltage, ACMD, ranges between +/-10V. This signal, along with GND, provides the input to the power amplifiers. The power amplifiers must be sized to drive the motors and load. For best performance, the amplifiers should be configured for a current mode of operation with no additional compensation. The gain should be set such that a 10 Volt input results in the maximum required current.

The DMC 1300 also provides an amplifier enable signal, AEN. This signal changes under the following conditions: the watchdog timer activates, the motor-off command, MO, is given, or the OE1command (Enable Off-On-Error) is given and the position error exceeds the error limit. As shown in Figure 3-4, AEN can be used to disable the amplifier for these conditions.

The standard configuration of the AEN signal is TTL active high. In other words, the AEN signal will be high when the controller expects the amplifier to be enabled. The polarity and the amplitude can be changed if you are using the ICM-1100 interface board. To change the polarity from active high (5 volts

= enable, zero volts = disable) to active low (zero volts = enable, 5 volts= disable), replace the 7407 IC with a 7406. Note that many amplifiers designate the enable input as 'inhibit'.

To change the voltage level of the AEN signal, note the state of the resistor pack on the ICM-1100. When Pin 1 is on the 5V mark, the output voltage is 0-5V. To change to 12 volts, pull the resistor pack and rotate it so that Pin 1 is on the 12 volt side. If you remove the resistor pack, the output signal is an open collector, allowing the user to connect an external supply with voltages up to 24V.

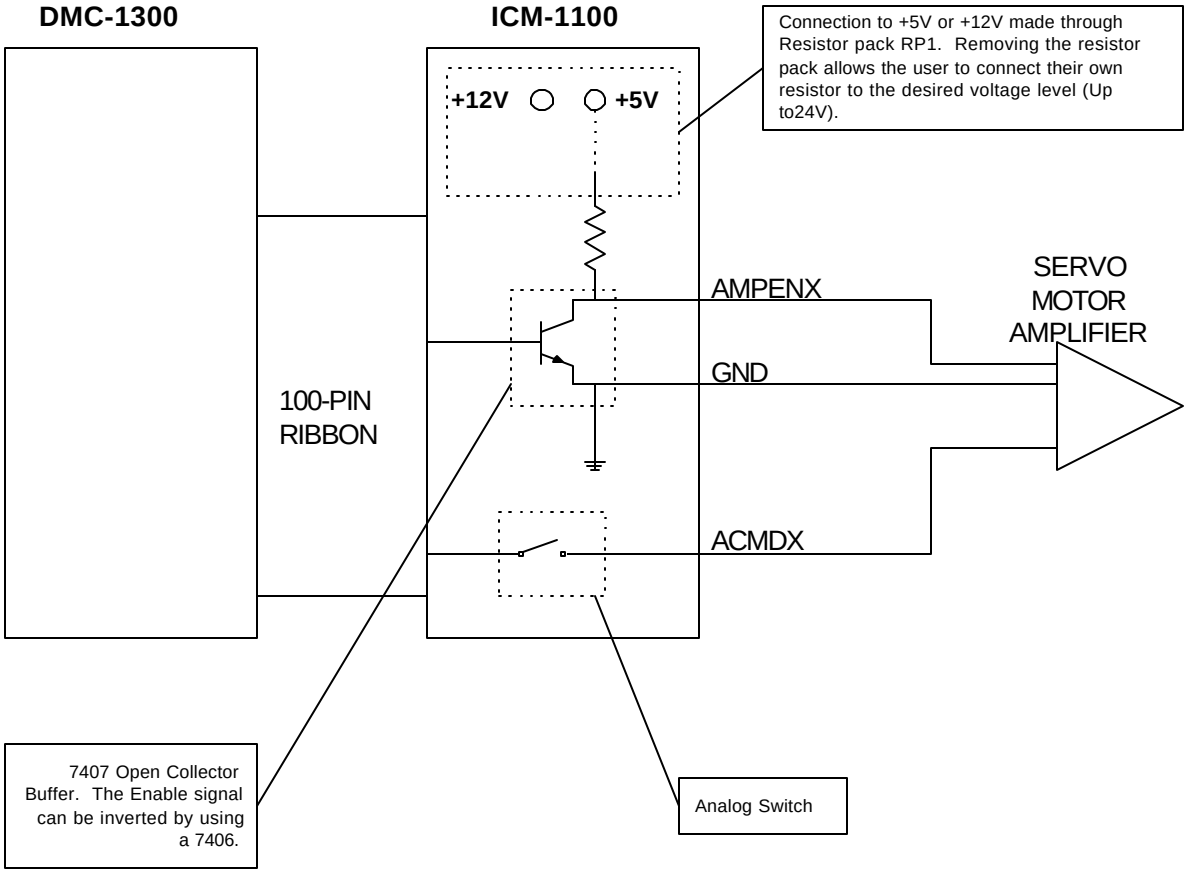


Figure 3-4 - Connecting AEN to the motor amplifier

TTL Inputs

1380

As previously mentioned, the DMC 1300 has 16 additional uncommitted TTL level inputs for controllers with 5 or more axes. These are specified as INx where x ranges from 9 thru 24. The reset input is also a TTL level, non-isolated signal and is used to locally reset the DMC 1300 without resetting the PC.

Analog Inputs

The DMC 1300 has seven analog inputs configured for the range between -10V and 10V. The inputs are decoded by a 12-bit A/D converter giving a voltage resolution of approximately .005V. The impedance of these inputs is 10 KΩ. The analog inputs are specified as AN[x] where x is a number 1 thru 7. Galil can supply the DMC 1300 with a 16-bit A/D converter as an option.

TTL Outputs

The DMC 1300 provides eight general use outputs and an error signal output.

The general use outputs are TTL and are accessible by connections to OUT1 thru OUT8. These outputs can be turned On and Off with the commands, SB (Set Bit), CB (Clear Bit), OB (Output Bit), and OP (Output Port). For more information about these commands, see the Command Summary. The value of the outputs can be checked with the operand _OP and the function @OUT[] (see Chapter 7, Mathematical Functions and Expressions).

1380

Controllers with 5 or more axes have an additional eight general use TTL outputs (connector JD5).

The status of the general purpose outputs can be read in the General Registers of the Dual Port RAM. Address 02B on the DMC 1310/1340 shows the status of the 8 general purpose outputs, while addresses 02E and 02F of the DMC 1350/1380 show the status of the 16 general purpose outputs.

The error signal output is available on the main connector (J2, pin 3). This is a TTL signal which is low when the controller has an error. This signal is not available through the phoenix connectors of the ICM-1100.

Note: When the error signal is active, the LED on the controller will be on. An error condition indicates one of the following conditions:

1. At least one axis has a position error greater than the error limit. The error limit is set by using the command ER.
2. The reset line on the controller is held low or is being affected by noise.
3. There is a failure on the controller and the processor is resetting itself.
4. There is a failure with the output IC which drives the error signal.

Offset Adjustment

For each axis, the DMC 1300 provides offset correction potentiometers to compensate for any offset in the analog output. These potentiometers have been adjusted at the factory to produce 0 Volts output for a zero digital motor command. Before making any adjustment to the offset, send the motor off command, MO, to the DMC 1300. This causes a zero digital motor command. Connect an oscilloscope or voltmeter to the motor command pin. You should measure zero volts. If not, adjust the offset potentiometer on the DMC 1300 until zero volts is observed.

Chapter 4 VME Communication

Introduction

The DMC-1300 utilizes a Dual Port RAM communication system. The DMC-1300 occupies 2K of the 65K available in the short I/O space. Either supervisory or user modes are permitted. To address this space, the address modifier lines of the VME Bus must be set to the following:

AM5	AM4	AM3	AM2	AM1	AM0
1	0	1	X	0	1

Please consult your VME CPU's user manual for more specific information on the proper configuration of address modifiers.

The DMC-1300 provides 4 address jumpers, labeled A15 through A12, where A15 represents the MSB of the address or 2^{15}. Bits 2^{0} through 2^{11} are all zero. The address jumpers A15 through A12 are configured for the desired address. A jumper present is a zero, a jumper missing sets the bit to a one. For the following example, R = Jumper removed and J = Jumper present.

A15	A14	A13	A12
R	J	J	J
1	0	0	0

This results in a base address of 8000 hex. The default address for the DMC-1300 is no jumpers present, or F000 hex.

RAM Organization

All addresses in the communication section will be in hex and be an offset from the base address (set by jumpers). This section will also show address locations for two versions of the controller, one for the DMC-1310/1340 and the other for the DMC-1350/1380.

The dual-port RAM of the DMC-1300 is organized into 12 buffers. Those buffer locations are listed below:

DMC-1310/1340

Address	Description
000 - 00F	Semaphore Registers
010 - 03F	General Registers
040 - 06F	Command Registers
070 - 09F	Response Buffer
0A0 - 0B1	Contour Buffer
0BE - 0E7	Program Buffer
100 - 13F	X axis
140 - 17F	Y axis
180 - 1BF	Z axis
1C0 - 1FF	W axis
200 - 23F	Coordinated Axis - S
240 - 3BF	Variables

DMC-1350/1380

Address	Description
000 - 00F	Semaphore Registers
010 - 03F	General Registers
040 - 073	Command Buffer
090 - 0C3	Response Buffer
0E0 - 101	Contour Buffer
10E - 15F	Program Buffer
200 - 23F	X axis
240 - 27F	Y axis
280 - 2BF	Z axis
2C0 - 2FF	W axis
300 - 33F	E axis
340 - 37F	F axis
380 - 3BF	G axis
3C0 - 3FF	H axis
400 - 43F	Coordinated Axis - S
440 - 5BF	Variables

Each of these registers and buffers will be described in detail in the following sections.

Semaphore Registers

DMC 1310/1380 Address 000 - 00F

The semaphore registers control signals for communication timing between the host CPU and the DMC-1300. These semaphore registers are set and cleared by either the host CPU or the DMC-1300. They are addressed only by their most significant bit (Bit 7), with the exception of the Program Buffer semaphore. Bit 6 of that semaphore is also set if an application program line is in the program buffer, and is cleared if the buffer contains communication from the application program. Below are the addresses and functions of each of the semaphore registers. These are identical for both the DMC-1310/1340 and the DMC-1350/1380.

Address	Function	Bit 7 Set by	Bit 7 Cleared by
001	Command Buffer	Host	DMC-1300
003	Response Buffer	DMC-1300	Host
005	Contour Buffer	Host	DMC-1300
007	Freeze Updates	Host	Host
009	Updating	DMC-1300	DMC-1300
00A	Clear Trippoint	Host	DMC-1300
00B	Program Buffer	DMC-1300	Host
00C	Thread 1 Paused	DMC-1300 or Host	Host
00D	Thread 2 Paused	DMC-1300 or Host	Host
00E	Thread 3 Paused	DMC-1300 or Host	Host
00F	Thread 4 Paused	DMC-1300 or Host	Host

General Registers

DMC 1310/1340 Address 010 - 036

DMC 1350/1380 Address 010 - 03B

The General Registers contain information about the controller such as motion status, error status, general I/O and interrupt status. Some of these registers are copies of internal DMC-1300 registers and writing to them will have no effect. Other registers are the only representation, and writing to them affects the internal status.

DMC-1310/1340

Address	Register
010	**General Status** Bit 7 = Application Strand Executing Bit 6 = Trace On Bit 5 = Contour Mode Bit 4 = Edit Mode Bit 3 = Overflow in Program Buffer Bit 2 = Contour Error Bit 1 = Error in Application Program Command Bit 0 = Error in Command from Command Buffer
012	**Command Buffer Error Code and Contour Mode Error Code**

	This byte contains the error code of the last error from a command buffer command. The error code will remain valid until cleared by the host or another error occurs. A list of error codes is listed in the TC command.
013	**Application Program Error Code** This byte contains the error code of the last error from an application program command. The error code will remain valid until cleared by the host or another error occurs.
014 - 017	**Sample Time** This 4-byte value contains a count of the samples since reset. It is the last item to be updated during an update cycle and can therefore be used to determine whether new axis data has been updated. NOTE: Writing in these locations has no effect.
018 - 019	**Coordinated Move Segment Count** For coordinated moves, the 2-byte value shows which coordinated segment is being run.
020 - 025	**Firmware Revision** This 6-byte value shows the firmware revision of the controller.
026	**Axis Number** This register contains the number of axis of the controller (1 - 4).
027	**Analog Inputs** Contains 1 if Analog; 0 if No Analog
028	**Program Buffer Control** This register chooses between three communication modes for the Application Program Buffer. To select the mode, write its number to the register. **Mode 0** - If the Program Buffer is full and an application program needs to write to the buffer, the new data will be lost. **Mode 1 -** If the Program Buffer is full and an application program needs to write to the buffer, application program execution will be held up until the buffer is clear and no data will be lost. **Mode 2 -** If the Program Buffer is full and an application program needs to write to the buffer, the old data will be lost.
029	**Number of Samples between Updates (divided by 2)** The default is 1 sample (2 msec). This register can be used to help a host create a position history at a particular time interval.
02A	**Uncommitted Input Port** This is a copy of the uncommitted inputs I8 - I1, with I8 being Bit 7.
02B	**Uncommitted Output Port** This is a copy of the uncommitted outputs O8 - O1. Writing to this address will change the state of the outputs on the following sample.
030 - 031	**Interrupt Status** These registers state which event has caused the VME Bus interrupt. These

	interrupts are set by the controller, and need to be cleared by the host after the interrupt has been processed. 030 Bit 7 = Inputs Bit 6 = Command Done Bit 5 = Application program stopped Bit 4 = User Interrupt Bit 3 = Watchdog timer Bit 2 = Limit switch occurred Bit 1 = Excess position error Bit 0 = All axes motion complete 031 Bit 7 = Application program paused Bit 6 = Contour interrupt Bit 5 = Bit 4 = Bit 3 = W Axis Motion Complete Bit 2 = Z Axis Motion Complete Bit 1 = Y Axis Motion Complete Bit 0 = X Axis Motion Complete
032	**Input Number** This register states which of the digital inputs caused an interrupt.
033	**User Interrupt Number** This register states which user interrupt has been sent using the UI command.
034 - 035	**Interrupt Mask** These two registers state which events will cause the VME bus to interrupt. The conditions that cause the interrupt are selected with the EI command. 034 Bit 7 = Inputs Bit 6 = Command Done Bit 5 = Application program stopped Bit 4 = Bit 3 = Watchdog timer Bit 2 = Limit Switch occurred Bit 1 = Excess position error Bit 0 = Motion complete on all axes 035 Bit 7 = Bit 6 = Contour interrupt Bit 5 = Bit 4 = Bit 3 = W axis motion complete Bit 2 = Z axis Motion Complete Bit 1 = Y axis Motion Complete Bit 0 = X axis Motion Complete
036	**Input Mask** This register shows which general inputs will cause a bus interrupt.

DMC 1350/1380

Address	Register
010	**General Status** Bit 7 = Application Strand Executing Bit 6 = Trace On

	Bit 5 = Contour Mode Bit 4 = Edit Mode Bit 3 = Overflow in Program Buffer Bit 2 = Contour Error Bit 1 = Error in Application Program Command Bit 0 = Error in Command from Command Buffer
012	**Command Buffer Error Code and Contour Mode Error Code** This byte contains the error code of the last error from a command buffer command. The error code will remain valid until cleared by the host or another error occurs. A list of error codes is listed in the TC command.
013	**Application Program Error Code** This byte contains the error code of the last error from an application program command. The error code will remain valid until cleared by the host or another error occurs.
014 - 017	**Sample Time** This 4-byte value contains a count of the samples since reset. It is the last item to be updated during an update cycle and can therefore be used to determine whether new axis data has been updated. NOTE: Writing in these locations has no effect.
018 - 019	**Coordinated Move Segment Count** For coordinated moves, the 2-byte value shows which coordinated segment is being run.
020 - 025	**Firmware Revision** This 6-byte value shows the firmware revision of the controller.
026	**Axis Number** This register contains the number of axis of the controller (1 - 8).
027	**Analog Inputs** Contains 1 if Analog; 0 if No Analog
028	**Program Buffer Control** This register chooses between three communication modes for the Application Program Buffer. To select the mode, write its number to the register. **Mode 0** - If the Program Buffer is full and an application program needs to write to the buffer, the new data will be lost. **Mode 1 -** If the Program Buffer is full and an application program needs to write to the buffer, application program execution will be held up until the buffer is clear and no data will be lost. **Mode 2 -** If the Program Buffer is full and an application program needs to write to the buffer, the old data will be lost.
029	**Number of Samples between Updates (divided by 2)** The default is 1 sample (2 msec). This register can be used to help a host create a position history at a particular time interval.
02A - 02C	**Uncommitted Input Port**

	This is a copy of the uncommitted inputs I24 - I1. The locations are I8 - I1 at 02A, I16 - I9 at 02B and I24 - I17 at 02C.
02E - 02F	**Uncommitted Output Port** This is a copy of the uncommitted outputs O16 - O1. Writing to this register will change the outputs on the next sample. The locations are 016 - 09 at 02E and 08 - 01 at 02F.
030 - 033	**Interrupt Status** These registers state which event has caused the VME Bus interrupt. These interrupts are set by the controller, and need to be cleared by the host after the interrupt has been processed. 030 Bit 6 = Command done Bit 5 = Application program stopped Bit 4 = User Interrupt Bit 3 = Watchdog timer Bit 2 = Limit switch occurred Bit 1 = Excess position error Bit 0 = Inputs 031 Bit 7 = Application program paused Bit 6 = Contour interrupt 032 Bit 0 = All axes motion complete 033 Bit 7 = H axis motion complete Bit 6 = G axis motion complete Bit 5 = F axis motion complete Bit 4 = E axis motion complete Bit 3 = W axis motion complete Bit 2 = Z axis motion complete Bit 1 = Y axis motion complete Bit 0 = X axis motion complete
034	**Input Number** This address shows which general purpose input caused the interrupt.
035	**User Interrupt Number** This address shows which user interrupt, sent by the UI command, caused the VME interrupt.
036 - 039	**Interrupt Mask** These two registers state which events will cause the VME bus to interrupt. The conditions that cause the interrupt are selected with the EI command. 036 Bit 6 = Command done Bit 5 = Application program stopped Bit 4 = Bit 3 = Watchdog timer Bit 2 = Limit switch occurred Bit 1 = Excess position error Bit 0 = Inputs

	037	Bit 7 = Bit 6 = Contour interrupt
	038	Bit 0 = All axes motion complete
	039	Bit 7 = H axis motion complete Bit 6 = G axis motion complete Bit 5 = F axis motion complete Bit 4 = E axis motion complete Bit 3 = W axis motion complete Bit 2 = Z axis motion complete Bit 1 = Y axis motion complete Bit 0 = X axis motion complete
03A - 03B	**Input Mask** This address shows which general purpose input will cause a bus interrupt.	

Command Buffer

DMC 1310/1340 Addresses 040 - 059

DMC 1350/1380 Addresses 040 - 073

The command buffer is used by the host to send commands to the DMC-1300. These commands can be sent in either Binary or ASCII format. A complete list of DMC commands in both Binary and ASCII format can be found in Chapter 12. If the Bit 3 system is being used, commands may be sent directly from the DMC terminal. Otherwise, commands will be written directly to the command buffer.

Sending commands using the Bit 3 System

Loading the Galil COMM1300 software gives the user a basic terminal emulator and status screen. All the basic commands of the controller can be sent to the command buffer from this screen. The communication options available through this screen are accessed as follows:

!UL <file name>		Uploads file to PC from 1300
!DO <file name>		Downloads file from PC into 1300
!?		Reports available screen and configuration options
!BI		Selects binary mode of communication
!AS		Selects ASCII mode of communication
!DE		Selects decimal display option
!HE		Selects hex display option
!W m,n		Displays contents of address m, m+1, m+2, m+3 as a four byte value. n is the watch number 1, 2 or 3. You can watch up to three groups of data.
Example:	!W 20,1	Watches address 20; #1
	!W 30,2	Watches address 30; #2
Q		Quits the COMM1300

Sending commands to the Command Buffer

The procedure for sending a command to the DMC 1300, whether Binary or ASCII, is as follows:

1. Check that Bit 7 of the Command Semaphore register (001) is clear. This means the last command has been completed.
2. Load the command into the command buffer in either Binary or ASCII.
3. Set Bit 7 (80 hex) of the command semaphore to start the command being processed.
4. Check that Bit 7 of the command semaphore is clear. Then check Bit 0 of the General Status (010). If the value is a one, then the command was not accepted. The command buffer error code will help find the cause of the problem.
5. If an interrogation command was sent, read the response buffer and clear the response buffer semaphore register.

ASCII Commands

The DMC 1300 instructions are represented by two ASCII upper case characters followed by applicable arguments. These arguments are of the form X, Y, Z and W for 1 through 4 axes and A, B, C, D, E, F, G and H for 5 through 8 axes. The host loads the buffer with the proper ASCII values starting at Address 040. Every ASCII command must be terminated with a carriage return (0D hex). Only one command can be sent at a time. Axis parameters in ASCII mode are separated by commas. If no data is specified for an axis, a comma is still needed as shown in the examples below.

KP12,8,,10,23	Set the proportional gain of the X axis to 12, Y axis to 8, W axis to 10 and E axis to 23.
OF,23	Set the Y axis offset to 23.
KD,,,,,100	Set the F axis derivative gain to 100.

Instead of data, some commands request action to occur on an axis or group of axes. For example, STXY stops motion on both the X and Y axes. Commas are not required in this case since the particular axis is specified by the appropriate letter. If no parameter follow the instruction, action will take place on all axes. Here are some examples of syntax for requesting action.

SHXW	Perform the Servo Here function on the X and W axes.
MO	Turn the motors off on all axes.
STG	Stop motion on the G axis.
BGAE	Begin motion on the A and E axes.

When requesting action for coordinated motion, the letter S is used to specify the coordinated motion. For example:

BGS	Begin coordinated sequence
BGSW	Begin coordinated

Below are two examples of sending ASCII commands to the DMC 1300, and their corresponding addresses.

Example: Send the command STX in ASCII format.

Address	Value (hex)	Characters
40	53	S
41	54	T
42	58	X
43	0D	Return

Example: Send the command PR 1024,,2048 in ASCII format.

Address	Value (hex)	Characters
40	50	P
41	52	R
42	31	1
43	30	0
44	32	2
45	34	4
46	2C	,
47	2C	,
48	32	2
49	30	0
4A	34	4
4B	38	8
4C	0D	Return

Binary Commands

Commands may also be sent to the DMC 1300 controller in Binary format. The Binary command format is in the form of a fixed format record. The first byte is always the command number which is between 138 and 255. The second byte is used to define whether the command is an interrogation and which axis or fields are valid for the command. Four fields of six bytes each follow for the data for each axis where 4 bytes are integer and 2 bytes are fraction. Numbers in these fields are represented in 2's complement.

DMC 1310/1340

040	Command
041	Format Bit 7 = 1 for interrogation, 0 for otherwise Bit 6 = Reserved Bit 5 = Reserved Bit 4 = Coordinated axis - S Bit 3 = W axis or field 4 data valid Bit 2 = Z axis or field 3 data valid Bit 1 = Y axis or field 2 data valid Bit 0 = X axis or field 1 data valid
042 - 047	Field 1 (X axis)
048 - 04D	Field 2 (Y axis)
04E - 053	Field 3 (Z axis)
054 - 059	Field 4 (W axis)

DMC 1350/1380

040	Command	Bit 7 = Binary
041	Format	Bit 7 = 1 for interrogation
042		Bit 0 = S (Coordinated axis - S)
043		Bit 7 = H axis or field 8 data valid Bit 6 = G axis or field 7 data valid Bit 5 = F axis or field 6 data valid Bit 4 = E axis or field 5 data valid Bit 3 = W axis or field 4 data valid Bit 2 = Z axis or field 3 data valid

	Bit 1 = Y axis or field 2 data valid Bit 0 = X axis or field 1 data valid
044 - 049	Field 1 (X axis)
04A - 04F	Field 2 (Y axis)
050 - 055	Field 3 (Z axis)
056 - 05B	Field 4 (W axis)
05C - 061	Field 5 (E axis)
062 - 067	Field 6 (F axis)
068 - 06D	Field 7 (G axis)
06E - 073	Field 8 (H axis)

Below are three examples showing how to send Binary commands to the DMC 1300.

Example: Send the command KP4,,6,8,,20,30 to the DMC 1380 in Binary format.

Address	Value (hex)	Comment
040	B6	Code for KP
041	00	No interrogation
042	00	No coordinated motion
043	D6	X, Z, W, F, G axes active
044 - 049	00 00 00 04 00 00	X data = 4
04A - 04F	00 00 00 00 00 00	Y data = 0
050 - 055	00 00 00 06 00 00	Z data = 6
056 - 05B	00 00 00 08 00 00	W data = 8
05C - 061	00 00 00 00 00 00	E data = 0
062 - 067	00 00 00 14 00 00	F data = 20
068 - 06D	00 00 00 1E 00 00	G data = 30
06F - 073	00 00 00 00 00 00	H data = 0

Example: Send the command BGS to the DMC 1340 in Binary format.

Address	Value (hex)	Comment
040	CE	Code for BG
041	10	Coordinated motion
042 - 059	--	Don't care

Example: Interrogate the DMC 1380 controller with the command ER,?,,?,,?

Address	Value (hex)	Comment
040	BF	Code for ER
041	80	Interrogation
042	00	No coordinated motion
043	2A	Y, W, F axes active
044 - 073	--	Don't care

Response Buffer

DMC 1310/1340 Addresses 070 - 09F

DMC 1350/1380 Addresses 090 - 0C3

The response buffer is used to return the values requested by an interrogation command. The information is always presented as a binary record in similar format to the command record. The data for each axis is 4 bytes integer and 2 bytes fraction.

When the response buffer has valid data, the response buffer semaphore (003) is set. Once the data has been read by the host, the semaphore should be cleared. The response semaphore will always be valid when the command buffer semaphore is cleared to show command done.

The address locations for the responses are as follows.

DMC 1310/1340

070	Command which generated response
071	Format Bit 7 = 1 (interrogation) Bit 6 = Bit 5 = Bit 4 = Bit 3 = W axis or field 4 data valid Bit 2 = Z axis or field 3 data valid Bit 1 = Y axis or field 2 data valid Bit 0 = X axis or field 1 data valid
072 - 077	X axis data
078 - 07D	Y axis data
07E - 083	Z axis data
084 - 089	W axis data

DMC 1350/1380

090	Command which generated response
091	Format Bit 7 = 1 (interrogation)
093	Bit 7 = H axis data valid Bit 6 = G axis data valid Bit 5 = F axis data valid Bit 4 = E axis data valid Bit 3 = W axis data valid Bit 2 = Z axis data valid Bit 1 = Y axis data valid Bit 0 = X axis data valid
094 - 099	X axis data
09A - 09E	Y axis data
0A0 - 0A5	Z axis data
0A6 - 0AB	W axis data
0AC - 0B1	E axis data
0B2 - 0B7	F axis data
0B8 - 0BD	G axis data
0BE - 0C3	H axis data

Example: The command KP? is sent to the command buffer of a DMC 1340. The response buffer would show the following for X, Y, Z and W values of 10,20,30 and 40 respectively.

Address	Value (hex)	Comment
070	B6	Code for KP
071	8F	All axes valid
072 - 077	00 00 00 0A 00 00	X data = 10
078 - 07D	00 00 00 14 00 00	Y data = 20
07E - 083	00 00 00 1E 00 00	Z data = 30
084 - 089	00 00 00 28 00 00	W data = 40

Example: The command ER? is sent to the command buffer of a DMC 1380. The response buffer would show the following for X, Y, Z, W, E, F, G and H values of 100, 200, 300, 400, 500, 600, 700 and 800 respectively.

Address	Value (hex)	Comment
090	BF	Code for ER
091	80	Bit 7 = 1 for interrogation
093	FF	All axes valid
094 - 099	00 00 00 64 00 00	X data = 100
09A - 09F	00 00 00 C8 00 00	Y data = 200
0A0 - 0A5	00 00 01 2C 00 00	Z data = 300
0A6 - 0AB	00 00 01 90 00 00	W data = 400
0AC - 0B1	00 00 01 F4 00 00	E data = 500
0B2 - 0B7	00 00 02 58 00 00	F data = 600
0B8 - 0BD	00 00 02 BC 00 00	G data = 700
0BE - 0C3	00 00 03 20 00 00	H data = 800

Contour Buffer

DMC 1310/1340 Addresses 0A0 - 0BD

DMC 1350/1380 Addresses 0E0 - 101

The contour buffer holds the contour record sent by the host during contour mode. This mode allows for arbitrary profiles by defining a set of positions vs. time. The contour mode is explained in detail in Section 5.

The procedure to send a contour record to the controller is as follows.

1. Enter the contour mode with the CM command.
2. Wait for Bit 7 of the contour semaphore (005) to be clear.
3. Write the contour record to the contour buffer.
4. Set Bit 7 of the contour semaphore.
5. Repeat steps 2 through 5 until the contour record 80 80 is sent ending contour mode.

The general status register's Bit 2 will be set if there is an error either in the timing or in the format of the contour record, and the command buffer error code will help find the cause of the error.

The format of the contour records are as follows.

DMC 1310/1340

0A0	80

0A1	80 to 88 time interval
0A2 - 0A5	Number of counts to move X axis
0A6 - 0A9	Number of counts to move Y axis
0AA - 0AD	Number of counts to move Z axis
0AE - 0B1	Number of counts to move W axis

DMC 1350/1380

0E0	80
0E1	80 to 88 time interval
0E2 - 0E5	Number of counts to move X axis
0E6 - 0E9	Number of counts to move Y axis
0EA - 0ED	Number of counts to move Z axis
0EE - 0F1	Number of counts to move W axis
0F2 - 0F5	Number of counts to move E axis
0F6 - 0F9	Number of counts to move F axis
0FA - 0FD	Number of counts to move G axis
0FE - 101	Number of counts to move H axis

Below is an example of using the contour mode on a DMC 1340 controller.

Address	Value (hex)	Comment
0A0	80	Contour mode
0A1	84	Time between records 16 msec
0A2 - 0A5	00 00 00 10	X move 16 counts
0A6 - 0A9	FF FF FF E6	Y move -26 counts
0AA - 0AD	00 00 02 11	Z move 529 counts
0AE - 0B1	00 00 00 00	W move 0 counts

The contour mode is then terminated using the following command.

Address	Value (hex)	Comment
0A0	80	Contour mode
0A1	80	End contour mode
0A2 - 0A5	XX XX XX XX	Don't care
0A6 - 0A9	XX XX XX XX	Don't care
0AA - 0AD	XX XX XX XX	Don't care
0AE - 0B1	XX XX XX XX	Don't care

Program Buffer

DMC 1310/1340 Addresses 0BE - 0E7

DMC 1350/1380 Addresses 10E - 15F

The program buffer is used for creating and editing application programs, receiving information from application programs, and receiving the program line if an error occurs during program execution. Programs sent to the program buffer must always be in ASCII format. There are two ways to write a program to the buffer.

Writing programs using the Bit 3 system

Programs may be written directly through the COMM 1300 software when using the Bit 3 adapter system. In this instance, the ED command is given to enter the editor mode. Once in the editor mode, commands are written in ASCII, with each line ending in a carriage return. Below are the commands used for working in the COMM 1300 editor mode.

<return>	Save line
<control> I	Insert line
<control> D	Delete line
Up arrow	Previous line (DO NOT USE <ctrl> P)
<control> Q	Quit Editor

When the <control> Q command is issued at the end of editing, the program is automatically downloaded to the controller.

Writing programs to the Program Buffer

Programs may also be written directly to the program buffer. The first two memory locations will contain the program line number whenever the program buffer contains a program line. The program buffer semaphore Bit 7 is set by the DMC 1300 whenever valid data is placed in the program buffer. Bit 6 is set whenever that data is a program line as opposed to an interrogation or output from an MG command.

To create or edit an application program, the ED command is given to put the DMC 1300 in edit mode. This can always be checked by testing Bit 4 of the general status register. The program line number of the program line in the buffer is placed by the DMC 1300 in the first two memory locations, with the program line following. The program buffer semaphore is set to C0 signifying that the buffer contains valid data and that the data is a program line.

At this point the host can alter the contents of the program buffer and invoke any of the editor commands. These commands are as follows.

Command	Command Code	Function
Save Line	9D	Save current line and put the next line in the program buffer.
Previous Line	9B	Put the previous line in the program buffer.
Delete Line	9A	Delete the program line.
Insert Line	99	Insert a new line before the current line.
Quit Edit	9C	Terminate the edit mode.

All program lines must be terminated in a carriage return (0D hex). The editor command is placed in the command buffer 40 by the host and the command semaphore (001 hex) is set (80 hex). When the command semaphore is cleared, another edit command may be executed.

Axis Buffers

DMC 1310/1340 Addresses 100 - 1FF

DMC 1350/1380 Addresses 200 - 43F

The axis buffers contain information on the control of each of the axes. The four buffers are identical in format.

DMC 1310/1340

X	Y	Z	W	
100	140	180	1C0	Status #1 Bit 7 = Axis running (In motion) Bit 6 = 1 - Positional move, 0 - jog Bit 5 = Position absolute move Bit 4 = Find edge Bit 3 = Home Bit 2 = Homing 1st phase complete Bit 1 = Homing 2nd phase complete Bit 0 = Coordinated move
101	141	181	1C1	Status #2 Bit 7 = Minus direction Bit 6 = Contour mode Bit 5 = Profile is in velocity slew Bit 4 = Stopped other than by reaching final destination Bit 3 = Profile is in final deceleration Bit 2 = Latch is armed Bit 1 = Off on error Bit 0 = Motor is off
104	144	184	1C4	Stop Code
105	145	185	1C5	Switches Bit 7 = Latched Bit 6 = State of Latch Bit 3 = State of Forward Limit Switch Bit 2 = State of Reversed Limit Switch Bit 1 = State of Home Bit 0 = SM Jumper installed
106-109	146-149	186-189	1C6-1C9	Motor position
10A-10D	14A-14D	18A-18D	1CA-1CD	Position error
10E-10F	14E-14F	18E-18F	1CE-1CF	Torque
110--113	150-153	190-193	1D0-1D3	Auxiliary encoder
114-117	154-157	194-197	1D4-1D7	Command position
118-11B	158-15B	198-19B	1D8-1DB	Latched position
11C - 11F	15C - 15F	19C - 19F	1DC - 1DF	Velocity (2.2 counts/sample)

DMC 1350/1380

X (E)	Y (F)	Z (G)	W (H)	
200(300)	240(340)	280(380)	2C0(3C0)	Status #1 Bit 7 = Axis running (In motion) Bit 6 = 1 - Positional move, 0 - jog Bit 5 = position absolute move Bit 4 = Find edge Bit 3 = Home Bit 2 = Homing 1st phase complete Bit 1 = Homing 2nd phase complete Bit 0 = Coordinated move

201(301)	241(341)	281(381)	2C1(3C1)	Status #2 Bit 7 = Minus direction Bit 6 = Contour mode Bit 5 = Profile is in velocity slew Bit 4 = Stopped other than by reaching final destination Bit 3 = Profile is in final deceleration Bit 2 = Latch is armed Bit 1 = Off on error Bit 0 = Motor is off
204(304)	244(344)	284(384)	2C4(3C4)	Stop Code
205(305)	245(345)	285(385)	2C5(3C5)	Switches Bit 7 = Latch has occurred Bit 6 = Latch is armed Bit 3 = State of forward limit switch Bit 2 = State of reversed limit switch Bit 1 = State of home Bit 0 = SM Jumper installed
206-209 (306-309)	246-249 (346-349)	286-289 (386-389)	2C6-2C9 (3C6-3C9)	Motor position
20A-20D (30A-30D)	24A-24D (34A-34D)	28A-28D (38A-38D)	2CA-2CD (3CA-3CD)	Position error
20E-20F (30E-30F)	24E-24F (34E-34F)	28A-28D (38A-38D)	2CA-2CD (3CA-3CD)	Torque
210-213 (310-313)	250-253 (350-353)	290-293 (390-393)	2D0-2D3 (3D0-3D3)	Auxiliary encoder
214-217 (314-317)	254-257 (354-357)	294-297 (394-397)	2D4-2D7 (3D4-3D7)	Command position
218-21B (318-31B)	258-25B (358-35B)	298-29B (398-39B)	2D8-2DB (3D8-3DB)	Latched position
21C-21F (31C-31F)	25C-25F (35C-35F)	29C-29F (39C-39F)	2DC-2DF (3DC-3DF)	Velocity (2.2 counts/sample)

The information in the axis buffers is updated by the DMC 1300 automatically at the rate set by the Number of Samples Register (029). The default value is every 2 msec.

In order to insure that these values (and the Sample Count 014-017) remain stable during a read, the following procedure should be followed.

1. Set Bit 7 of the Freeze semaphore (007). This tells the DMC 1300 not to start its update procedure.
2. Wait for Bit 7 of the updating semaphore to be a 0. This is in case the DMC 1300 was already in its update procedure.
3. Perform all the reads needed.

4. Clear the Freeze semaphore.

Coordinate Axis Buffer

DMC 1310/1340 Addresses 200 - 23F

DMC 1350/1380 Addresses 400 - 43F

This buffer gives status information during a coordinated move (VP or CR).

DMC 1310/1340

200	Status #1	Bit 7 = Coordinate move running.
201	Status #2	Bit 7 = Bit 6 = Bit 5 = Profile is in velocity slew. Bit 4 = Stopped other than by reaching final destination. Bit 3 = Profile is in final deceleration. Bit 2 = Bit 1 = Bit 0 =

DMC 1350/1380

400	Status #1	Bit 7 = Coordinate move running.
401	Status #2	Bit 7 = Bit 6 = Bit 5 = Profile is in velocity slew. Bit 4 = Stopped other than by reaching final destination. Bit 3 = Profile is in final deceleration. Bit 2 = Bit 1 = Bit 0 =

Variable Buffer

DMC 1310/1340 Addresses 240 - 3BF

DMC 1350/1380 Addresses 440 - 5BF

An array with 64 variable elements is automatically assigned to the dual-port RAM at locations 240 - 3BF for the DMC 1310/1340 and locations 440 - 5BF for the DMC 1350/1380. These variables have 4 bytes of integer and 2 bytes of fraction. Variables are assigned with the VR[n]= command where n = 0 through 63.

Variable updates are not affected by the freeze update semaphore. Therefore, to ensure that data has not changed during the READ cycle, it is suggested that you read the data twice.

For example, the 22nd variable element on a DMC 1310 would be at address:

22 * 6 + $240 = $2C4

The 22nd variable element on a DMC 1350 would be at address:

22 * 6 + $440 = $4C4

Interrupts

The DMC 1300 board supports the VME Bus vectored interrupts. The interrupt may occur on any one of the seven interrupt levels.

To select the interrupt level, two sets of jumpers must be installed on the board. These are JP13 (IRQ1 - IRQ7) and JP12 (IAD1 - IAD4), and are located on the bottom right side of the board. The two sets work together and must be set correctly for the interrupt procedure to function correctly. The IRQ1 - IRQ7 jumpers set the interrupt priority (IRQ7 is the highest). One jumper should be placed on the level chosen. The IAD1 - IAD4 jumpers are used to put the vector on the bus. They form a 3 bit binary combination, where IAD4 is the most significant bit. The combination must be equal to the IRQ number picked. A jumper causes that bit to be a zero.

For example, to set the interrupt for level 6, a jumper would be placed on IRQ6. The IAD1 - IAD4 jumpers would be as follows.

IAD4	**IAD2**	**IAD1**
1	1	0
R	R	J

where R means jumper removed and J means jumper present.

The interrupt vector is a number between 8 and 255 and must be set by the EI command. Below is an example of setting the interrupt vector to 64 (40 hex).

ASCII EI,,64

Binary 8C 02 00 00 00 00 00 00 00 00 00 00 00 40

There are many events which can generate an interrupt. While more than one event can be enabled, there is only one interrupt on the DMC 1300. Each event can be enabled by the EI instruction or by writing directly to the interrupt mask in the dual-port RAM. The events and their corresponding position in the interrupt mask can be found on page…

The mask can be set in the first field of the EI instruction. The byte that corresponds to 38 is the most significant and 39 the least. For input interrupts, 03A must be set for the corresponding input. If we wished to create an interrupt whenever the Y axis completes its motion, the mask would consist of Bit 2 of the LSB, which is 0002 hex. In decimal, that number is 2, so the commands would be as follows.

ASCII EI2

Binary 8C 01 00 00 00 02 00 00 00 XX XX XX XX

The User Interrupt (UI) instruction is used to generate an interrupt from an application program.

The User Interrupt number will appear in address 033 for the DMC 1310/1340 and 035 for the DMC 1350/1380. The motion complete interrupts will generate an interrupt whenever the controller has finished profiling a motion. The motor motion itself may still not have settled. Bit 1 of 36 will cause an interrupt whenever the position error for any axis exceeds the limits set in the ER instruction.

Bits 5 and 6 of 36 create interrupts which facilitate communication. Command Done interrupts when the command semaphore is cleared at the end of a command. Application Program Stopped generates an interrupt on any termination of an application program (either an EN command, an error or an abort). The Program Buffer Valid interrupt occurs on any write to the program buffer by the DMC 1300. This would be an MG command or an interrogation KP? command in an application program. The Program Pause interrupt is caused by the Program Pause (PP) command.

Chapter 5 Command Basics

Introduction

The DMC 1300 provides over 100 commands for specifying motion and machine parameters. Commands are included to initiate action, interrogate status and configure the digital filter.

Commands can be sent to the DMC 1300 in either ASCII or Binary. In ASCII, the instruction set is BASIC-like and easy to use. Instructions consist of two uppercase letters that correspond phonetically with the appropriate function. For example, the instruction BG begins motion, while ST stops motion. In Binary, commands are fixed format with a command number followed by numeric fields for each axis.

For example, to send a positional move in ASCII format, the following command is sent:

PR 4000,9000 <enter>

where PR is the Position Relative command, 4000 and 9000 are the X and Y positions respectively, and the <enter> terminates the command.

In Binary, the equivalent would be:

C9 07 00 00 0F A0 00 00 00 00 23 28 00 00

where C9 is the Position Relative binary code, 07 specifies parameters for the X and Y fields, and the remaining fields are the X and Y data showing four bytes of integer followed by two bytes of fraction.

Commands can be sent "live" over the bus for immediate execution by the DMC 1300, or an entire group of commands can be downloaded into the DMC 1300 memory for execution at a later time. Combining commands into groups for later execution is referred to as Applications Programming and is discussed in the following chapter.

This section describes the DMC 1300 instruction set and syntax. A summary of commands as well as a complete listing of all DMC 1300 instructions is included in the *Command Reference* chapter.

Command Syntax

ASCII

DMC 1300 instructions are represented by two ASCII upper case characters followed by applicable arguments. A space may be inserted between the instruction and arguments. An <enter> is used to terminate the instruction for processing by the DMC 1300 command interpreter. Note: If you are using a Galil terminal program, commands will not be processed until an <enter> command is given.

IMPORTANT: All DMC 1300 commands are sent in upper case.

Commands may be sent to the controller through the Galil COMM-1300 software if using the Bit 3 system, or written directly to the Command Buffer for a custom VME interface.

For example, the command

PR 4000 <enter> Position relative

PR is the two character instruction for position relative. 4000 is the argument which represents the required position value in counts. The <enter> terminates the instruction. The space between PR and 4000 is optional. This command is sent directly through the command line of the COMM-1300 software with a Bit 3 system. With a custom VME interface, the following hex equivalent is written to the command buffer at address 40.

Address	Command (hex)	Description
40	50	ASCII 'P'
41	52	ASCII 'R'
42	34	ASCII '4'
43	30	ASCII '0'
44	30	ASCII '0'
45	30	ASCII '0'
46	0D	ASCII 'Return'

Bit 7 of the Command Semaphore is then set to 80 hex, which will send this command to the controller.

For specifying data for the X,Y,Z and W axes, commas are used to separate the axes. If no data is specified for an axis, a comma is still needed as shown in the examples below. If no data is specified for an axis, the previous value is maintained. The space between the data and instruction is optional. For controllers with 5 or more axes, the axes are referred to as A,B,C,D,E,F,G,H where X,Y,Z,W and A,B,C,D may be used interchangeably.

Instead of data, some commands request action to occur on an axis or group of axes. For example, ST XY stops motion on both the X and Y axes. Commas are not required in this case since the particular axis is specified by the appropriate letter X Y Z or W. If no parameters follow the instruction, action will take place on all axes. Here are some examples of syntax for requesting action:

BG X	Begin X only
BG Y	Begin Y only
BG XYZW	Begin all axes
BG YW	Begin Y and W only

BG	Begin all axes

1380

For controllers with 5 or more axes, the axes are referred to as A,B,C,D,E,F,G,H. The specifiers X,Y,Z,W and A,B,C,D may be used interchangeably:

BG ABCDEFGH	Begin all axes
BG D	Begin D only

Binary

Commands may also be sent to the DMC 1300 in Binary mode. Most DMC commands will have a corresponding Binary code. Binary commands and any corresponding data are written to the Command Buffer (040). For example, the Binary code for GN is B8 (hex). This code is written to address 040 followed by axis data. The axis data is represented by four bytes of integer and two bytes of data. To send the command GN 5,,7 to the DMC 1300, the following command is sent:

Address	Command (hex)	Description
40	B8	Code for GN
41	05	X and Z axis active
42 - 45	00 00 00 05	X axis integer
46 - 47	00 00	X axis fraction
48 - 4D	-	Y axis data
4E - 51	00 00 00 07	Z axis integer
52 - 53	00 00	Z axis fraction
54 - 59	-	W axis data

This command is sent when Bit 7 of the Command Semaphore is set to 80 (hex). A full listing and explanation of the DMC 1300 address registers can be found in Chapter 4.

Coordinated Motion with more than 1 axis

When requesting action for coordinated motion, the letter S is used to specify the coordinated motion. For example:

BG S	Begin coordinated sequence
BG SW	Begin coordinated sequence and W axis

Program Syntax

Chapter 7 explains the how to write and execute motion control programs.

Controller Response to DATA

When using the Comm1300 software, the DMC 1300 returns a : for valid commands, and a ? for invalid commands.

For example, if the command BG is sent in lower case, the DMC 1300 will return a ?.

:bg <enter>	invalid command, lower case
?	DMC 1300 returns a ?

When the controller receives an invalid command the user can request the error code. The error code will specify the reason for the invalid command response. To request the error code type the command: TC1 For example:

?TC1 <enter>	Tell Code command
1 Unrecognized command	Returned response

Command errors can also be read directly from the address registers. Command errors can be generated either from the Command Buffer or from an application program. When the controller receives an error from the Command Buffer, Bit 0 of the General Status (010) will be set. The reason for the error is read at address 012, with the error codes listed in the TC command. Similarly, when the controller receives an error from an application program, Bit 1 of the General Status (010) will be set. The reason for that error is read at address 013, with the list of error codes listed in the TC command.

There are many reasons for receiving an invalid command response. The most common reasons are: unrecognized command (such as typographical entry or lower case), command given at improper time (such as during motion), or a command out of range (such as exceeding maximum speed). A complete list of all error codes can be found with the description of the TC command in the Command Reference, Chapter 11.

Interrogating the Controller

Interrogation Commands

The DMC 1300 has a set of commands that directly interrogate the controller. When the command is entered through the COMM-1300 software, the requested data is returned in decimal format on the next line followed by a carriage return and line feed. When the command is written to the Command Buffer, the response can be read in the Response Buffer. When there is valid data in the Response Buffer, the Response Buffer Semaphore is set. If the interrogation is sent from an application program, the response is found in the Program Buffer.

Summary of Interrogation Commands

RP	Report Command Position
RL	Report Latch
^R ^V	Firmware Revision Information
SC	Stop Code
TB	Tell Status
TC	Tell Error Code
TD	Tell Dual Encoder

TE	Tell Error
TI	Tell Input
TP	Tell Position
TR	Trace
TS	Tell Switches
TT	Tell Torque
TV	Tell Velocity

For example, the following example illustrates how to display the current position of the X axis:

TP X <enter>	Tell position X
0000000000	Controllers Response
TP XY <enter>	Tell position X and Y
0000000000,0000000000	Controllers Response

Many of these interrogation commands can also be read directly from registers in the DMC 1300. Please refer to Chapter 4 to find the actual address locations of these commands.

Additional Interrogation Methods.

Most commands can be interrogated by using a question mark (?) as the axis specifier. Type the command followed by a ? for each axis requested.

PR ,,?,,?	The controller will return the PR value for the C and E axes
PR ?,?,?,?	The controller will return the PR value for the A,B,C and D axes
PR ,,,,,,,?	The controller will return the PR value for the H axis

The controller can also be interrogated with operands.

Operands

Most DMC 1300 commands have corresponding operands that can be used for interrogation. Operands must be used inside of valid DMC expressions. For example, to display the value of an operand, the user could use the command:

MG ‘operand’

All of the command operands begin with the underscore character (_). For example, the value of the current position on the X axis can be assigned to the variable ‘V’ with the command:

V=_TPX

The Command Reference denotes all commands which have an equivalent operand as "Used as an Operand". Also, see description of operands in Chapter 7.

Command Summary

For a complete command summary, see Chapter 12 *Command Reference*.

Chapter 6 Programming Motion

Overview

The DMC 1300 can be commanded to do the following modes of motion: Absolute and relative independent positioning, jogging, linear interpolation (up to 8 axes), linear and circular interpolation (2 axes with 3rd axis of tangent motion), electronic gearing and contouring. These modes are discussed in the following sections.

The DMC-1310 is a single axis controller and uses X-axis motion only. Likewise, the DMC-1320 uses X and Y, the DMC-1330 uses X,Y and Z, and the DMC-1340 uses X,Y,Z and W. The DMC-1350 uses A,B,C,D, and E. The DMC-1360 uses A,B,C,D,E, and F. The DMC-1370 uses A,B,C,D,E,F and G. The DMC-1380 uses the axes A,B,C,D,E,F,G, and H.

The example applications described below will help guide you to the appropriate mode of motion.

1380 For controllers with 5 or more axes, the specifiers, ABCDEFGH, are used. XYZ and W may be interchanged with ABCD.

Independent Axis Positioning

In this mode, motion between the specified axes is independent, and each axis follows its own profile. The user specifies the desired absolute position (PA) or relative position (PR), slew speed (SP), acceleration ramp (AC), and deceleration ramp (DC), for each axis. On begin (BG), the DMC 1300 profiler generates the corresponding trapezoidal or triangular velocity profile and position trajectory. The controller determines a new command position along the trajectory every sample period until the specified profile is complete. Motion is complete when the last position command is sent by the DMC 1300 profiler. Note: The actual motor motion may not be complete when the profile has been completed, however, the next motion command may be specified.

The Begin (BG) command can be issued for all axes either simultaneously or independently. XYZ or W axis specifiers are required to select the axes for motion. When no axes are specified, this causes motion to begin on all axes.

The speed (SP) and the acceleration (AC) can be changed at any time during motion, however, the deceleration (DC) and position (PR or PA) cannot be changed until motion is complete. Remember, motion is complete when the profiler is finished, not when the actual motor is in position. The Stop

command (ST) can be issued at any time to decelerate the motor to a stop before it reaches its final position.

An incremental position movement (IP) may be specified during motion as long as the additional move is in the same direction. Here, the user specifies the desired position increment, n. The new target is equal to the old target plus the increment, n. Upon receiving the IP command, a revised profile will be generated for motion towards the new end position. The IP command does not require a begin. Note: If the motor is not moving, the IP command is equivalent to the PR and BG command combination.

Command Summary - Independent Axis

COMMAND	DESCRIPTION
PR X,Y,Z,W	Specifies relative distance
PA x,y,z,w	Specifies absolute position
SP x,y,z,w	Specifies slew speed
AC x,y,z,w	Specifies acceleration rate
DC x,y,z,w	Specifies deceleration rate
BG XYZW	Starts motion
ST XYZW	Stops motion before end of move
IP x,y,z,w	Changes position target
IT x,y,z,w	Time constant for independent motion smoothing
AM XYZW	Trippoint for profiler complete
MC XYZW	Trippoint for "in position"

The lower case specifiers (x,y,z,w) represent position values for each axis. For controllers with more than 4 axes, the position values would be represented as a,b,c,d,e,f,g,h.

Operand Summary - Independent Axis

OPERAND	DESCRIPTION
_Acx	Return acceleration rate for the axis specified by ‘x’
_DCx	Return deceleration rate for the axis specified by ‘x’
_SPx	Returns the speed for the axis specified by ‘x’
_PAx	Returns current destination if ‘x’ axis is moving, otherwise returns the current commanded position if in a move.
_PRx	Returns current incremental distance specified for the ‘x’ axis

Example - Absolute Position Movement

PA 10000,20000	Specify absolute X,Y position
AC 1000000,1000000	Acceleration for X,Y
DC 1000000,1000000	Deceleration for X,Y
SP 50000,30000	Speeds for X,Y
BG XY	Begin motion

Example - Multiple Move Sequence

Required Motion Profiles:

X-Axis	500 counts	Position
	10000 count/sec	Speed
	500000 counts/sec^2	Acceleration
Y-Axis	1000 counts	Position
	15000 count/sec	Speed
	500000 counts/sec^2	Acceleration
Z-Axis	100 counts	Position
	5000 counts/sec	Speed
	500000 counts/sec	Acceleration

This example will specify a relative position movement on X, Y and Z axes. The movement on each axis will be separated by 20 msec. Fig. 6.1 shows the velocity profiles for the X,Y and Z axis.

#A	Begin Program
PR 2000,500,100	Specify relative position movement of 1000, 500 and 100 counts for X,Y and Z axes.
SP 15000,10000,5000	Specify speed of 10000, 15000, and 5000 counts / sec
AC 500000,500000,500000	Specify acceleration of 500000 counts / sec^2 for all axes
DC 500000,500000,500000	Specify deceleration of 500000 counts / sec^2 for all axes
BG X	Begin motion on the X axis
WT 20	Wait 20 msec
BG Y	Begin motion on the Y axis
WT 20	Wait 20 msec
BG Z	Begin motion on Z axis
EN	End Program

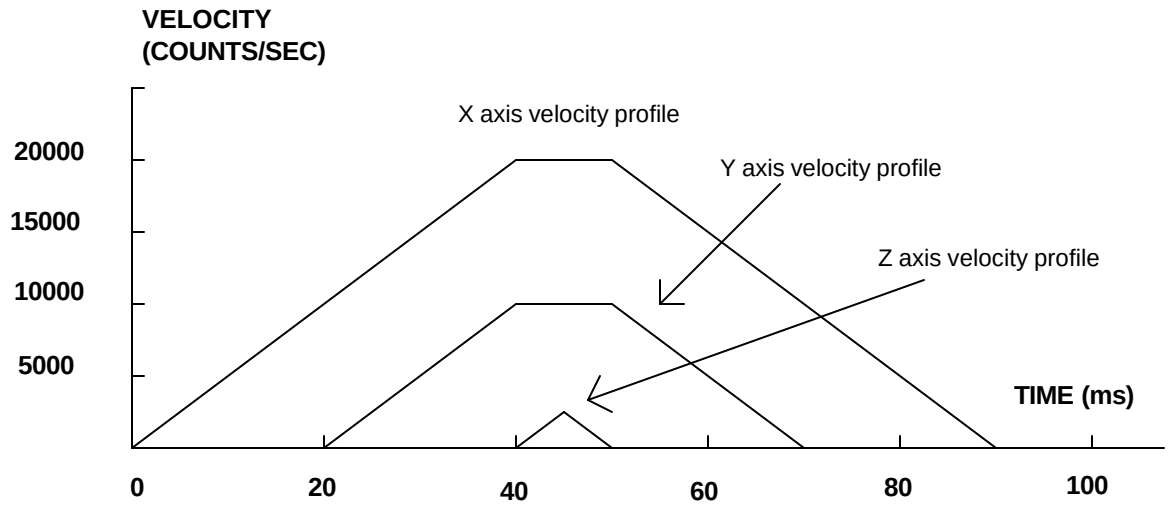


Figure 6.1 - Velocity Profiles of XYZ

Notes on fig 6.1: The X and Y axis have a 'trapezoidal' velocity profile, while the Z axis has a 'triangular' velocity profile. The X and Y axes accelerate to the specified speed, move at this constant speed, and then decelerate such that the final position agrees with the command position, PR. The Z axis accelerates, but before the specified speed is achieved, must begin deceleration such that the axis will stop at the commanded position. All 3 axes have the same acceleration and deceleration rate, hence, the slope of the rising and falling edges of all 3 velocity profiles are the same.

Independent Jogging

The jog mode of motion allows the user to change speed, direction and acceleration during motion. The user specifies the jog speed (JG), acceleration (AC), and the deceleration (DC) rate for each axis. The direction of motion is specified by the sign of the JG parameters. When the begin command is given (BG), the motor accelerates up to speed and continues to jog at that speed until a new speed or stop (ST) command is issued. If the jog speed is changed during motion, the controller will make a accelerated (or decelerated) change to the new speed.

An instant change to the motor position can be made with the use of the IP command. Upon receiving this command, the controller commands the motor to a position which is equal to the specified increment plus the current position. This command is useful when trying to synchronize the position of two motors while they are moving.

Note that the controller operates as a closed-loop position controller while in the jog mode. The DMC 1300 converts the velocity profile into a position trajectory and a new position target is generated every sample period. This method of control results in precise speed regulation with phase lock accuracy.

Command Summary - Jogging

COMMAND	DESCRIPTION
AC x,y,z,w	Specifies acceleration rate
BG X,Y,Z,W	Begins motion
DC x,y,z,w	Specifies deceleration rate
IP x,y,z,w	Increments position instantly

IT x,y,z,w	Time constant for independent motion smoothing
JG +/-x,y,z,w	Specifies jog speed and direction
ST XYZW	Stops motion

Operand Summary - Independent Axis

OPERAND	DESCRIPTION
_ACx	Return acceleration rate for the axis specified by 'x'
_DCx	Return deceleration rate for the axis specified by 'x'
_SPx	Returns the jog speed for the axis specified by 'x'
_TVx	Returns the actual velocity of the axis specified by 'x' (averaged over.25 sec)

Example - Jog in X only

Jog X motor at 50000count/s. After X motor is at its jog speed, begin jogging Z in reverse direction at 25000 count/s.

```
#A
AC 20000,,20000          Specify X,Z acceleration of 20000 cts/sec
DC 20000,,20000          Specify X,Z deceleration of 20000 cts/sec
JG 50000,,-25000         Specify jog speed and direction for X and Z axis
BG XY                    Begin X motion
AS X                     Wait until X is at speed
BG Z                     Begin Z motion
EN
```

Example - Joystick jogging

The jog speed can also be changed using an analog input such as a joystick. Assume that for a 10 Volt input the speed must be 50000 counts/sec.

```
#JOG                     Label
JG0                      Set in Jog Mode
BGX                      Begin motion
#B                       Label for Loop
V1 = @AN[1]              Read analog input
VEL = V1*50000/2047      Compute speed
JG VEL                   Change JG speed
JP #B                    Loop
```

Linear Interpolation Mode

The DMC 1300 provides a linear interpolation mode for 2 or more axes (up to 8 axes for the DMC-1380). In linear interpolation mode, motion between the axes is coordinated to maintain the prescribed vector speed, acceleration, and deceleration along the specified path. The motion path is described in terms of

incremental distances for each axis. An unlimited number of incremental segments may be given in a continuous move sequence, making the linear interpolation mode ideal for following a piece-wise linear path. There is no limit to the total move length.

The LM command selects the Linear Interpolation mode and axes for interpolation. For example, LM YZ selects only the Y and Z axes for linear interpolation.

When using the linear interpolation mode, the LM command only needs to be specified once unless the axes for linear interpolation change.

Specifying Linear Segments

The command LI x,y,z,w or LI a,b,c,d,e,f,g,h specifies the incremental move distance for each axis. This means motion is prescribed with respect to the current axis position. Up to 511 incremental move segments may be given prior to the Begin Sequence (BGS) command. Once motion has begun, additional LI segments may be sent to the controller.

The clear sequence (CS) command can be used to remove LI segments stored in the buffer prior to the start of the motion. To stop the motion, use the instructions STS or AB. The command, ST, causes a decelerated stop. The command, AB, causes an instantaneous stop and aborts the program, and the command AB1 aborts the motion only.

The Linear End (LE) command must be used to specify the end of a linear move sequence. This command tells the controller to decelerate to a stop following the last LI command. If an LE command is not given, an Abort AB1 must be used to abort the motion sequence.

It is the responsibility of the user to keep enough LI segments in the DMC 1300 sequence buffer to ensure continuous motion. If the controller receives no additional LI segments and no LE command, the controller will stop motion instantly at the last vector. There will be no controlled deceleration. LM? or _LM returns the available spaces for LI segments that can be sent to the buffer. 511 returned means the buffer is empty and 511 LI segments can be sent. A zero means the buffer is full and no additional segments can be sent. As long as the buffer is not full, additional LI segments can be sent and loaded through the DMC 1300 Command Buffer.

The instruction _CS returns the segment counter. As the segments are processed, _CS increases, starting at zero. This function allows the host computer to determine which segment is being processed. This information is also available at addresses 018 - 019 of the general registers in the Dual Port RAM.

Specifying Vector Acceleration, Deceleration and Speed:

The commands VS n, VA n, and VD n are used to specify the vector speed, acceleration and deceleration. The DMC 1300 computes the vector speed based on the axes specified in the LM mode. For example, LM XYZ designates linear interpolation for the X,Y and Z axes. The vector speed for this example would be computed using the equation:

$VS^2=XS^2+YS^2+ZS^2$, where XS, YS and ZS are the speed of the X,Y and Z axes.

The controller always uses the axis specifications from LM, not LI, to compute the speed.

In cases where the acceleration causes the system to 'jerk', the DMC 1300 provides a vector motion smoothing function. VT is used to set the S-curve smoothing constant for coordinated moves.

Additional Commands

The DMC 1300 provides commands for additional control of vector motion and program control. Note: Many of the commands used in Linear Interpolation motion also applies Vector motion described in the next section.

Trippoints

The command AV n is the 'After Vector' trippoint, which halts program execution until the vector distance of n has been reached.

In this example, the XY system is required to perform a 90° turn. In order to slow the speed around the corner, we use the AV 4000 trippoint, which slows the speed to 1000 count/s. Once the motors reach the corner, the speed is increased back to 4000 cts / s.

Instruction	Interpretation
#LMOVE	Label
DP ,,0,0	Define position of Z and W axes to be 0
LMXY	Define linear mode between X and Y axes.
LI 5000,0	Specify first linear segment
LI 0,5000	Specify second linear segment
LE	End linear segments
VS 4000	Specify vector speed
BGS	Begin motion sequence
AV 4000	Set trippoint to wait until vector distance of 4000 is reached
VS 1000	Change vector speed
AV 5000	Set trippoint to wait until vector distance of 5000 is reached
VS 4000	Change vector speed
EN	Program end

Specifying Vector Speed for Each Segment

The instruction VS has an immediate effect and, therefore, must be given at the required time. In some applications, such as CNC, it is necessary to attach various speeds to different motion segments. This can be done by the instruction

LI x,y,z,w < n

This instruction attaches the vector speed, n, to the motion segment LI. As a consequence, the program #LMOVE can be written in the alternative form:

Instruction	Interpretation
#ALT	Label for alternative program
DP 0,0	Define Position of X and Y axis to be 0
LMXY	Define linear mode between X and Y axes.
LI 4000,0 <4000	Specify first linear segment with a vector speed of 4000
LI 1000,0 < 1000	Specify second linear segment with a vector speed of 1000

LI 0,5000 < 4000	Specify third linear segment with a vector speed of 4000
LE	End linear segments
BGS	Begin motion sequence
EN	Program end

Command Summary - Linear Interpolation

COMMAND	DESCRIPTION
LM xyzw LM abcdefgh	Specify axes for linear interpolation (same) controllers with 5 or more axes
LM?	Returns number of available spaces for linear segments in DMC 1300 sequence buffer. Zero means buffer full. 512 means buffer empty.
LI x,y,z,w < n LI a,b,c,d,e,f,g,h < n	Specify incremental distances relative to current position, and assign vector speed n.
VS n	Specify vector speed
VA n	Specify vector acceleration
VD n	Specify vector deceleration
BGS	Begin Linear Sequence
CS	Clear sequence
LE	Linear End- Required at end of LI command sequence
LE?	Returns the length of the vector (resets after 2147483647)
AMS	Trippoint for After Sequence complete
AV n	Trippoint for After Relative Vector distance,n
VT	S curve smoothing constant for vector moves

Operand Summary - Linear Interpolation

OPERAND	DESCRIPTION
_AV	Return distance traveled
_CS	Segment counter - returns number of the segment in the sequence, starting at zero.
_LE	Returns length of vector (resets after 2147483647)
_LM	Returns number of available spaces for linear segments in DMC 1300 sequence buffer. Zero means buffer full. 512 means buffer empty.
_VPm	Return the absolute coordinate of the last data point along the trajectory. (m=X,Y,Z or W or A,B,C,D,E,F,G or H)

To illustrate the ability to interrogate the motion status, consider the first motion segment of our example, #LMOVE, where the X axis moves toward the point X=5000. Suppose that when X=3000, the controller is interrogated using the command 'MG _AV'. The returned value will be 3000. The value of _CS, _VPX and _VPY will be zero.

Now suppose that the interrogation is repeated at the second segment when Y=2000. The value of _AV at this point is 7000, _CS equals 1, _VPX=5000 and _VPY=0.

Example - Linear Move

Make a coordinated linear move in the ZW plane. Move to coordinates 40000,30000 counts at a vector speed of 100000 counts/sec and vector acceleration of 1000000 counts/sec^2.

Instruction	Interpretation
#TEST	Label
LM ZW	Specify axes for linear interpolation
LI,,40000,30000	Specify ZW distances
LE	Specify end move
VS 100000	Specify vector speed
VA 1000000	Specify vector acceleration
VD 1000000	Specify vector deceleration
BGS	Begin sequence
AMS	After motion sequence ends
EN	End program

Note that the above program specifies the vector speed, VS, and not the actual axis speeds VZ and VW. The axis speeds are determined by the DMC 1300 from:

$$VS = \sqrt{VZ^2 + VW^2}$$

The resulting profile is shown in Figure 6.2.

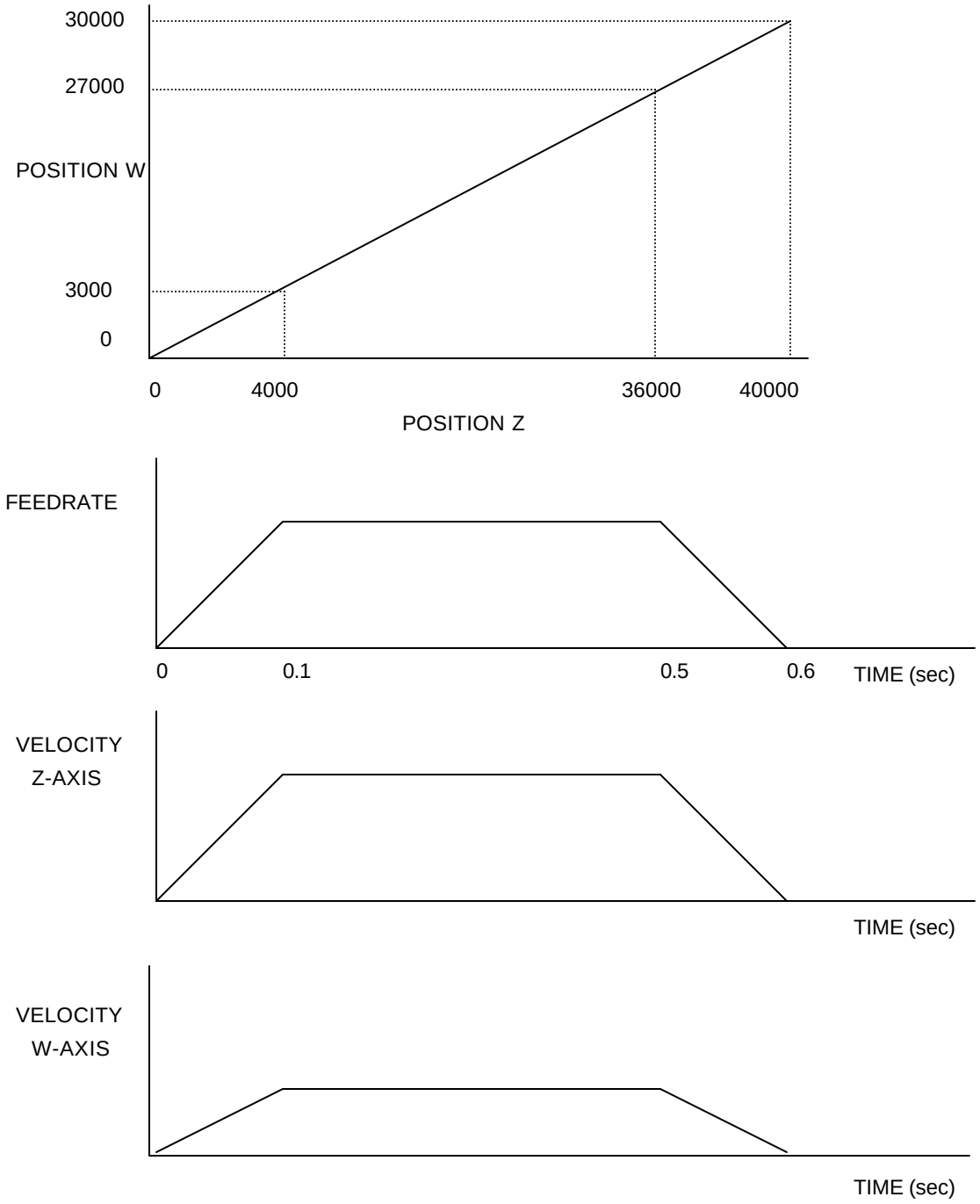


Figure 6.2 - Linear Interpolation

Example - Multiple Moves

This example makes a coordinated linear move in the XY plane. The Arrays VX and VY are used to store 750 incremental distances which are filled by the program #LOAD.

Instruction	Interpretation
#LOAD	Load Program
DM VX [750],VY [750]	Define Array

COUNT=0	Initialize Counter
N=0	Initialize position increment
#LOOP	LOOP
VX [COUNT]=N	Fill Array VX
VY [COUNT]=N	Fill Array VY
N=N+10	Increment position
COUNT=COUNT+1	Increment counter
JP #LOOP,COUNT<750	Loop if array not full
#A	Label
LM XY	Specify linear mode for XY
COUNT=0	Initialize array counter
#LOOP2;JP#LOOP2,_LM=0	If sequence buffer full, wait
JS#C,COUNT=500	Begin motion on 500th segment
LI VX[COUNT],VY[COUNT]	Specify linear segment
COUNT=COUNT+1	Increment array counter
JP #LOOP2,COUNT<750	Repeat until array done
LE	End Linear Move
AMS	After Move sequence done
MG "DONE"	Send Message
EN	End program
#C;BGS;EN	Begin Motion Subroutine

Vector Mode: Linear and Circular Interpolation Motion

The DMC 1300 allows a long 2-D path consisting of linear and arc segments to be prescribed. Motion along the path is continuous at the prescribed vector speed even at transitions between linear and circular segments. The DMC 1300 performs all the complex computations of linear and circular interpolation, freeing the host from this time intensive task.

The coordinated motion mode is similar to the linear interpolation mode. Any pair of two axes may be selected for coordinated motion consisting of linear and circular segments. In addition, a third axis can be controlled such that it remains tangent to the motion of the selected pair of axes. Note that only one pair of axes can be specified for coordinated motion at any given time.

The command VM m,n,p where 'm' and 'n' are the coordinated pair and p is the tangent axis (Note: the commas which separate m,n and p are not necessary). For example, VM XWZ selects the XW axes for coordinated motion and the Z-axis as the tangent.

Specifying Vector Segments

The motion segments are described by two commands; VP for linear segments and CR for circular segments. Once a set of linear segments and/or circular segments have been specified, the sequence is ended with the command VE. This defines a sequence of commands for coordinated motion. Immediately prior to the execution of the first coordinated movement, the controller defines the current

position to be zero for all movements in a sequence. Note: This 'local' definition of zero does not affect the absolute coordinate system or subsequent coordinated motion sequences.

The command, VP xy specifies the coordinates of the end points of the vector movement with respect to the starting point. The command, CR r,q,d define a circular arc with a radius r, starting angle of q, and a traversed angle d. The notation for q is that zero corresponds to the positive horizontal direction, and for both q and d, the counter-clockwise (CCW) rotation is positive.

Up to 511 segments of CR or VP may be specified in a single sequence and must be ended with the command VE. The motion can be initiated with a Begin Sequence (BGS) command. Once motion starts, additional segments may be added.

The Clear Sequence (CS) command can be used to remove previous VP and CR commands which were stored in the buffer prior to the start of the motion. To stop the motion, use the instructions STS or AB1. ST stops motion at the specified deceleration. AB1 aborts the motion instantaneously.

The Vector End (VE) command must be used to specify the end of the coordinated motion. This command requires the controller to decelerate to a stop following the last motion requirement. If a VE command is not given, an Abort (AB1) must be used to abort the coordinated motion sequence.

It is the responsibility of the user to keep enough motion segments in the DMC 1300 sequence buffer to ensure continuous motion. If the controller receives no additional motion segments and no VE command, the controller will stop motion instantly at the last vector. There will be no controlled deceleration. LM? or _LM returns the available spaces for motion segments that can be sent to the buffer. 511 returned means the buffer is empty and 511 segments can be sent. A zero means the buffer is full and no additional segments can be sent. As long as the buffer is not full, additional segments can be sent to the controller through the Command Buffer.

The operand _CS can be used to determine the value of the segment counter. This information is also available at addresses 018 - 019 of the general registers in the Dual Port RAM.

Specifying Vector Acceleration, Deceleration and Speed:

The commands VS n, VA n, and VD n are used to specify the vector speed, acceleration and deceleration. The DMC 1300 computes the vector speed based on the two axes specified in the VM mode. For example, VM YZ designates vector mode for the Y and Z axes. The vector speed for this example would be computed using the equation:

$VS^2=YS^2+ZS^2$, where YS and ZS are the speed of the Y and Z axes.

In cases where the acceleration causes the system to 'jerk', the DMC 1300 provides a vector motion smoothing function. VT is used to set the S-curve smoothing constant for coordinated moves.

Additional Commands

The DMC 1300 provides commands for additional control of vector motion and program control. Note: Many of the commands used in Vector Mode motion also applies Linear Interpolation motion described in the previous section.

Trippoints

The command AV n is the 'After Vector' trippoint, which halts program execution until the vector distance of n has been reached.

Specifying Vector Speed for Each Segment

The vector speed may be specified by the immediate command VS. It can also be attached to a motion segment with the instructions

VP x,y, < n

CR r,θ,δ < n

Both cases assign a vector speed of n count/s to the corresponding motion segment.

Compensating for Differences in Encoder Resolution:

By default, the DMC 1300 uses a scale factor of 1:1 for the encoder resolution when used in vector mode. If this is not the case, the command, ES can be used to scale the encoder counts. The ES command accepts two arguments which represent the number of counts for the two encoders used for vector motion. The smaller ratio of the two numbers will be multiplied by the higher resolution encoder. For more information, see ES command in Chapter 11, Command Summary.

Tangent Motion:

Several applications, such as cutting, require a third axis (i.e. a knife blade), to remain tangent to the coordinated motion path. To handle these applications, the DMC 1300 allows one axis to be specified as the tangent axis. The VM command provides parameter specifications for describing the coordinated axes and the tangent axis.

VM m,n,p	m,n specifies coordinated axes p specifies tangent axis such as X,Y,Z,W or A,B,C,D,E,F,G,H p=N turns off tangent axis

Before the tangent mode can operate, it is necessary to assign an axis via the VM command and define its offset and scale factor via the TN m,n command. m defines the scale factor in counts/degree and n defines the tangent position that equals zero degrees in the coordinated motion plane. The _TN can be used to return the initial position of the tangent axis.

Example - XY Table Control

Assume an XY table with the Z-axis controlling a knife. The Z-axis has a 2000 quad counts/rev encoder and has been initialized after power-up to point the knife in the +Y direction. A 180° circular cut is desired, with a radius of 3000, center at the origin and a starting point at (3000,0). The motion is CCW, ending at (-3000,0). Note that the 0° position in the XY plane is in the +X direction. This corresponds to the position -500 in the Z-axis, and defines the offset. The motion has two parts. First, X,Y and Z are driven to the starting point, and later, the cut is performed. Assume that the knife is engaged with output bit 0.

Instruction	**Interpretation**
#EXAMPLE	Example program
VM XYZ	XY coordinate with Z as tangent
TN 2000/360,-500	2000/360 counts/degree, position -500 is 0 degrees in XY plane
CR 3000,0,180	3000 count radius, start at 0 and go to 180 CCW
VE	End vector
CB0	Disengage knife
PA 3000,0,_TN	Move X and Y to starting position, move Z to initial tangent position
BG XYZ	Start the move to get into position
AM XYZ	When the move is complete
SB0	Engage knife

WT50	Wait 50 msec for the knife to engage
BGS	Do the circular cut
AMS	After the coordinated move is complete
CB0	Disengage knife
MG "ALL DONE"	
EN	End program

Command Summary - Vector Mode Motion

COMMAND	DESCRIPTION
VM m,n	Specifies the axes for the planar motion where m and n represent the planar axes and p is the tangent axis.
VP m,n	Return coordinate of last point, where m=X,Y,Z or W.
CR r,Θ, ±ΔΘ	Specifies arc segment where r is the radius, Θ is the starting angle and ΔΘ is the travel angle. Positive direction is CCW.
VS n	Specify vector speed or feedrate of sequence.
VA n	Specify vector acceleration along the sequence.
VD n	Specify vector deceleration along the sequence.
BGS	Begin motion sequence.
CS	Clear sequence.
AV n	Trippoint for After Relative Vector distance, n.
AMS	Holds execution of next command until Motion Sequence is complete.
TN m,n	Tangent scale and offset.
ES m,n	Ellipse scale factor.
VT	S curve smoothing constant for coordinated moves
LM?	Return number of available spaces for linear and circular segments in DMC 1300 sequence buffer. Zero means buffer is full. 512 means buffer is empty.

Operand Summary - Vector Mode Motion

OPERAND	DESCRIPTION
_VPM	The absolute coordinate of the axes at the last intersection along the sequence.
_AV	Distance traveled.
_LM	Number of available spaces for linear and circular segments in DMC 1300 sequence buffer. Zero means buffer is full. 512 means buffer is empty.
_CS	Segment counter - Number of the segment in the sequence, starting at zero.

When AV is used as an operand, _AV returns the distance traveled along the sequence.

The operands _VPX and _VPY can be used to return the coordinates of the last point specified along the path.

Example:

Traverse the path shown in Fig. 6.3. Feedrate is 20000 counts/sec. Plane of motion is XY

Instruction	Interpretation
VM XY	Specify motion plane
VS 20000	Specify vector speed
VA 1000000	Specify vector acceleration
VD 1000000	Specify vector deceleration
VP -4000,0	Segment AB
CR 1500,270,-180	Segment BC
VP 0,3000	Segment CD
CR 1500,90,-180	Segment DA
VE	End of sequence
BGS	Begin Sequence

The resulting motion starts at the point A and moves toward points B, C, D, A. Suppose that we interrogate the controller when the motion is halfway between the points A and B.

The value of _AV is 2000

The value of _CS is 0

_VPX and _VPY contain the absolute coordinate of the point A

Suppose that the interrogation is repeated at a point, halfway between the points C and D.

The value of _AV is $4000+1500\pi+2000=10,712$

The value of _CS is 2

_VPX,_VPY contain the coordinates of the point C

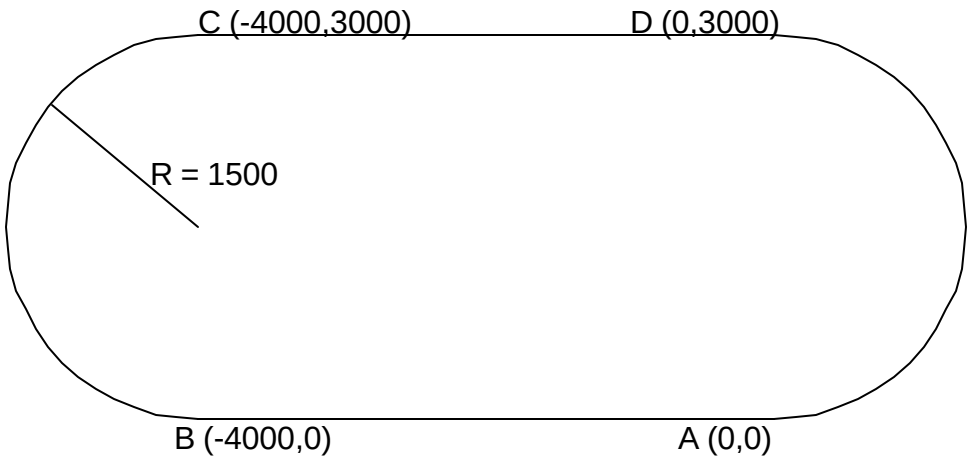


Figure 6.3 - The Required Path

Electronic Gearing

This mode allows up to 8 axes to be electronically geared to one master axis. The master may rotate in both directions and the geared axes will follow at the specified gear ratio. The gear ratio may be different for each axis and changed during motion.

The command GAX or GAY or GAZ or GAW (or GAA or GAB or GAC or GAD or GAE or GAF or GAG or GAH for DMC-1380) specifies the master axis. There may only be one master. GR x,y,z,w specifies the gear ratios for the slaves where the ratio may be a number between +/-127.9999 with a fractional resolution of .0001. GR 0,0,0,0 turns off electronic gearing for any set of axes. A limit switch will also disable electronic gearing for that axis. GR causes the specified axes to be geared to the actual position of the master. The master axis is commanded with motion commands such as PR, PA or JG.

When the master axis is driven by the controller in the jog mode or an independent motion mode, it is possible to define the master as the command position of that axis, rather than the actual position. The designation of the commanded position master is by the letter, C. For example, GACX indicates that the gearing is the commanded position of X.

An alternative gearing method is to synchronize the slave motor to the commanded vector motion of several axes performed by GAS. For example, if the X and Y motor form a circular motion, the Z axis may move in proportion to the vector move. Similarly, if X,Y and Z perform a linear interpolation move, W can be geared to the vector move.

Electronic gearing allows the geared motor to perform a second independent or coordinated move in addition to the gearing. For example, when a geared motor follows a master at a ratio of 1:1, it may be advanced an additional distance with PR, or JG, commands, or VP, or LI.

Command Summary - Electronic Gearing

COMMAND	DESCRIPTION
GA n	Specifies master axis for gearing where: n = X,Y,Z or W or A,B,C,D,E,F,G,H for main encoder as master
	n = CX,CY,CZ or CW or CA, CB, CC, CD, CE, CF,CG,CH for commanded position.
	n = DX,DY,DZ or DW or DA, DB, DC, DD, DE, DF,DG,DH for auxiliary encoders
	n = S vector move as master
GR x,y,z,w	Sets gear ratio for slave axes. 0 disables electronic gearing for specified axis.
GR a,b,c,d,e,f,g,h	Sets gear ratio for slave axes. 0 disables electronic gearing for specified axis.
MR x,y,z,w	Trippoint for reverse motion past specified value. Only one field may be used.
MF x,y,z,w	Trippoint for forward motion past specified value. Only one field may be used.

Operand Summary - Electronic Gearing

COMMAND	DESCRIPTION
GA n	Specifies master axis for gearing where: n = X,Y,Z or W or A,B,C,D,E,F,G,H for main encoder as master

	n = CX,CY,CZ or CW or CA, CB, CC, CD, CE, CF,CG,CH for commanded position.
	n = DX,DY,DZ or DW or DA, DB, DC, DD, DE, DF,DG,DH for auxiliary encoders
	n = S vector move as master
GR x,y,z,w	Sets gear ratio for slave axes. 0 disables electronic gearing for specified axis.
GR a,b,c,d,e,f,g,h	Sets gear ratio for slave axes. 0 disables electronic gearing for specified axis.
MR x,y,z,w	Trippoint for reverse motion past specified value. Only one field may be used.
MF x,y,z,w	Trippoint for forward motion past specified value. Only one field may be used.

Example - Simple Master Slave

Master axis moves 10000 counts at slew speed of 100000 counts/sec. Y is defined as the master. X,Z,W are geared to master at ratios of 5,-.5 and 10 respectively.

GAY	Specify master axes as Y
GR 5,,-.5,10	Set gear ratios
PR ,10000	Specify Y position
SP ,100000	Specify Y speed
BGY	Begin motion

Example - Electronic Gearing

Objective: Run two geared motors at speeds of 1.132 and -0.045 times the speed of an external master. The master is driven at speeds between 0 and 1800 RPM (2000 counts/rev encoder).

Solution: Use a DMC-1330 controller, where the Z-axis is the master and X and Y are the geared axes.

MO Z	Turn Z off, for external master
GA Z	Specify master axis
GR 1.132,-.045	Specify gear ratios

Now suppose the gear ratio of the X-axis is to change on-the-fly to 2. This can be achieved by commanding:

GR 2	Specify gear ratio for X axis to be 2

In applications where both the master and the follower are controlled by the DMC 1300 controller, it may be desired to synchronize the follower with the commanded position of the master, rather than the actual position. This eliminates the coupling between the axes which may lead to oscillations.

For example, assume that a gantry is driven by two axes, X,Y, on both sides. The X-axis is the master and the Y-axis is the follower. To synchronize Y with the commanded position of X, use the instructions:

GA CX	Specify master as commanded position of X
GR,1	Set gear ratio for Y as 1:1
PR 3000	Command X motion
BG X	Start motion on X axis

You may also perform profiled position corrections in the electronic gearing mode. Suppose, for example, that you need to advance the slave 10 counts. Simply command

IP ,10	Specify an incremental position movement of 10 on Y axis.

Under these conditions, this IP command is equivalent to:

PR,10	Specify position relative movement of 10 on Y axis
BGY	Begin motion on Y axis

Often the correction is quite large. Such requirements are common when synchronizing cutting knives or conveyor belts.

Example - Synchronize two conveyor belts with trapezoidal velocity correction.

GAX	Define master axis as X
GR,2	Set gear ratio 2:1 for Y
PR,300	Specify correction distance
SP,5000	Specify correction speed
AC,100000	Specify correction acceleration
DC,100000	Specify correction deceleration
BGY	Start correction

Contour Mode

The DMC 1300 also provides a contouring mode. This mode allows any arbitrary position curve to be prescribed for 1 to 8 axes. This is ideal for following computer generated paths such as parabolic, spherical or user-defined profiles. The path is not limited to straight line and arc segments and the path length may be infinite.

Specifying Contour Segments

The Contour Mode is specified with the command, CM. For example, CMXZ specifies contouring on the X and Z axes. Any axes that are not being used in the contouring mode may be operated in other modes.

A contour is described by position increments which are described with the command, CD x,y,z,w over a time interval, DT n. The parameter, n, specifies the time interval. The time interval is defined as 2^n ms, where n is a number between 1 and 8. The controller performs linear interpolation between the specified increments, where one point is generated for each millisecond.

The contour mode may also be accessed through the Contour Buffer of the Dual Port RAM. Contour data may be sent to this buffer to generate an arbitrary motion profile. The Contour Buffer holds the contour record sent by the host during the contour mode, and is set and cleared by the Contour Semaphore. An error in the contour mode can be checked at Bit 2 of the General Status (010), with the corresponding error code found at 012. A list of the Contour Buffer addresses can be found in Chapter 4.

Consider the trajectory shown in Fig. 6.4. The position X may be described by the points:

Point 1	X=0 at T=0ms
Point 2	X=48 at T=4ms

Point 3	X=288 at T=12ms
Point 4	X=336 at T=28ms

The same trajectory may be represented by the increments

Increment 1	DX=48	Time=4	DT=2
Increment 2	DX=240	Time=8	DT=3
Increment 3	DX=48	Time=16	DT=4

When the controller receives the command to generate a trajectory along these points, it interpolates linearly between the points. The resulting interpolated points include the position 12 at 1 msec, position 24 at 2 msec, etc.

The programmed commands to specify the above example are:

#A	
CMX	Specifies X axis for contour mode
DT 2	Specifies first time interval, 2^2 ms
CD 48;WC	Specifies first position increment
DT 3	Specifies second time interval, 2^3 ms
CD 240;WC	Specifies second position increment
DT 4	Specifies the third time interval, 2^4 ms
CD 48;WC	Specifies the third position increment
DT0;CD0	Exits contour mode
EN	

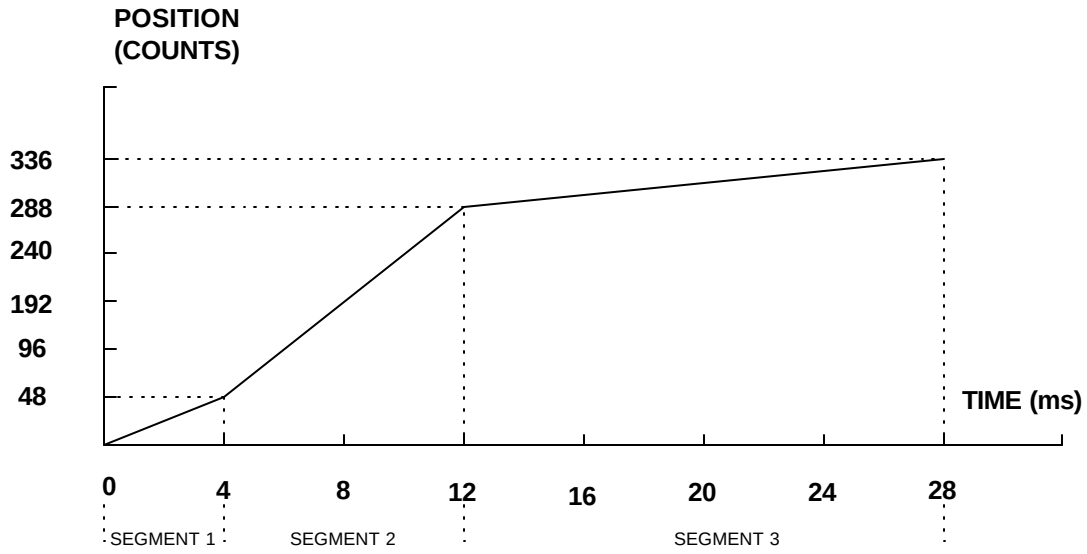


Figure 6.4 - The Required Trajectory

Additional Commands

The command, WC, is used as a trippoint "When Complete". This allows the DMC 1300 to use the next increment only when it is finished with the previous one. Zero parameters for DT followed by zero parameters for CD exit the contour mode.

If no new data record is found and the controller is still in the contour mode, the controller waits for new data. No new motion commands are generated while waiting. If bad data is received, the controller responds with a ?.

Command Summary - Contour Mode

COMMAND	DESCRIPTION
CM XYZW	Specifies which axes for contouring mode. Any non-contouring axes may be operated in other modes.
CM ABCDEFGH	Contour axes for DMC-1380
CD x,y,z,w	Specifies position increment over time interval. Range is +/-32,000. Zero ends contour mode.
CD a,b,c,d,e,f,g,h	Position increment data for DMC-1380
DT n	Specifies time interval 2^n msec for position increment, where n is an integer between 1 and 8. Zero ends contour mode. If n does not change, it does not need to be specified with each CD.
WC	Waits for previous time interval to be complete before next data record is processed.

Operand Summary - Contour Mode

OPERAND	DESCRIPTION
_CS	Return segment number

General Velocity Profiles

The Contour Mode is ideal for generating any arbitrary velocity profiles. The velocity profile can be specified as a mathematical function or as a collection of points.

The design includes two parts: Generating an array with data points and running the program.

Generating an Array - An Example

Consider the velocity and position profiles shown in Fig. 6.5. The objective is to rotate a motor a distance of 6000 counts in 120 ms. The velocity profile is sinusoidal to reduce the jerk and the system vibration. If we describe the position displacement in terms of A counts in B milliseconds, we can describe the motion in the following manner:

$$w = \frac{A}{B}\left(1 - \cos(2p/B)\right)$$

$$X = \frac{AT}{B} - \frac{A}{2p}\sin(2p/B)$$

Note: ω is the angular velocity; X is the position; and T is the variable, time, in milliseconds.

In the given example, A=6000 and B=120, the position and velocity profiles are:

$$X = 50T - (6000/2\pi) \sin (2\pi\ T/120)$$

Note that the velocity, ω, in count/ms, is

$$\omega = 50 [1 - \cos 2\pi\ T/120]$$

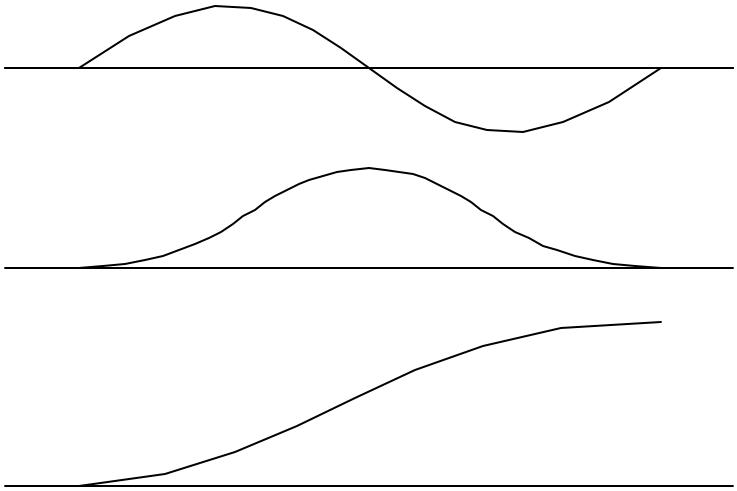


Figure 6.5 - Velocity Profile with Sinusoidal Acceleration

The DMC 1300 can compute trigonometric functions. However, the argument must be expressed in degrees. Using our example, the equation for X is written as:

$$X = 50T - 955 \sin 3T$$

A complete program to generate the contour movement in this example is given below. To generate an array, we compute the position value at intervals of 8 ms. This is stored at the array POS. Then, the difference between the positions is computed and is stored in the array DIF. Finally the motors are run in the contour mode.

Contour Mode Example:

Instruction	Interpretation
#POINTS	Program defines X points
DM POS[16]	Allocate memory
DM DIF[15]	
C=0	Set initial conditions, C is index
T=0	T is time in ms
#A	
V1=50*T	

```
V2=3*T                  Argument in degrees
V3=-955*@SIN[V2]+V1     Compute position
V4=@INT[V3]             Integer value of V3
POS[C]=V4               Store in array POS
T=T+8
C=C+1
JP #A,C<16
#B                      Program to find position differences
C=0
#C
D=C+1
DIF[C]=POS[D]-POS[C]    Compute the difference and store
C=C+1
JP #C,C<15
EN                      End first program
#RUN                    Program to run motor
CMX                     Contour Mode
DT3                     4 millisecond intervals
C=0
#E
CD DIF[C]               Contour Distance is in DIF
WC                      Wait for completion
C=C+1
JP #E,C<15
DT0
CD0                     Stop Contour
EN                      End the program
```

Teach (Record and Play-Back)

Several applications require teaching the machine a motion trajectory. Teaching can be accomplished using the DMC 1300 automatic array capture feature to capture position data. The captured data may then be played back in the contour mode. The following array commands are used:

DM C[n]	Dimension array
RA C[]	Specify array for automatic record (up to 4 for DMC-1340; 8 for DMC-1380)
RD _TPX	Specify data for capturing (such as _TPX or _TPZ)
RC n,m	Specify capture time interval where n is 2n msec, m is number of records to be captured
RC? or _RC	Returns a 1 if recording

Record and Playback Example:

Instruction	Interpretation
#RECORD	Begin Program
DP0	Define position for X axis to be 0
DA*[ ]	De-allocate all arrays
DM XPOS [501]	Dimension 501 element array called XPOS
RA XPOS [ ]	Record Elements into XPOS array
RD_TPX	Element to be recorded is encoder position of X axis
MOX	Motor off for X axis
RC2	Begin Recording with a sample rate of 2 msec
#LOOP1;JP#LOOP1,_RC=1	Loop until all elements have been recorded
#COMPUTE	Routine to determine the difference between consecutive points
DM DX [500]	Dimension a 500 element array to hold contour points
I = 0	Set loop counter
#LOOP2	Loop to calculate the difference
DX[I]=XPOS[I+1]-XPOS[I]	Calculate difference
I=I+1	Update loop counter
JP#LOOP2,I<500	Continue looping until DX is full
#PLAYBK	Routine to play back motion that was recorded
SHX	Servo Here
WT1000	Wait 1 sec (1000 msec)
CMX	Specify contour mode on X axis
DT2	Set contour data rate to be 2 msec
I=0	Set array index to 0
#LOOP3	Subroutine to execute contour points
CD DX[I];WC	Contour data command; Wait for next contour point
I=I+1	Update index
JP#LOOP3,I<500	Continue until all array elements have been executed
DT0	Set contour update rate to 0
CD0	Disable the contour mode (combination of DT0 and CD0)
EN	End program

For additional information about automatic array capture, see Chapter 7, Arrays.

Stepper Motor Operation

When configured for stepper motor operation, several commands are interpreted differently than from servo mode. The following describes operation with stepper motors.

Specifying Stepper Motor Operation

In order to command stepper motor operation, the appropriate stepper mode jumpers must be installed. See chapter 2 for this installation.

Stepper motor operation is specified by the command MT. The argument for MT is as follows:

2 specifies a stepper motor with active low step output pulses

-2 specifies a stepper motor with active high step output pulses

Stepper Motor Smoothing

The command, KS, provides stepper motor smoothing. The effect of the smoothing can be thought of as a simple Resistor-Capacitor (single pole) filter. The filter occurs after the motion profiler and has the effect of smoothing out the spacing of pulses for a more smooth operation of the stepper motor. Use of KS is most applicable when operating in full step or half step operation. KS will cause the step pulses to be delayed in accordance with the time constant specified.

When operating with stepper motors, you will always have some amount of stepper motor smoothing, KS. Since this filtering effect occurs after the profiler, the profiler may be ready for additional moves before all of the step pulses have gone through the filter. It is important to consider this effect since steps may be lost if the controller is commanded to generate an additional move before the previous move has been completed. See the discussion below, *Monitoring Generated Pulses vs Commanded Pulses*.

The general motion smoothing command, IT, can also be used. The purpose of the command, IT, is to smooth out the motion profile and decrease 'jerk' due to acceleration.

Monitoring Generated Pulses vs Commanded Pulses

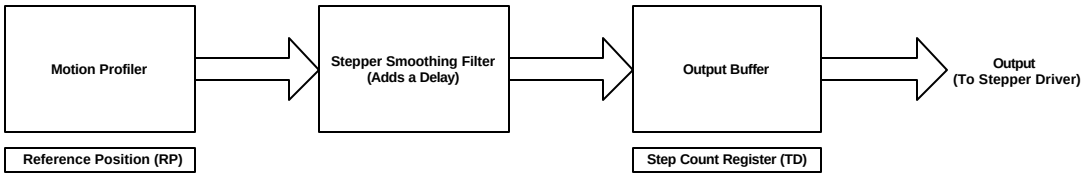
For proper controller operation, it is necessary to make sure that the controller has completed generating all step pulses before making additional moves. This is most particularly important if you are moving back and forth. For example, when operating with servo motors, the trippoint AM (After Motion) is used to determine when the motion profiler is complete and is prepared to execute a new motion command. However when operating in stepper mode, the controller may still be generating step pulses when the motion profiler is complete. This is caused by the stepper motor smoothing filter, KS. To understand this, consider the steps the controller executes to generate step pulses:

First, the controller generates a motion profile in accordance with the motion commands.

Second, the profiler generates pulses as prescribed by the motion profile. The pulses that are generated by the motion profiler can be monitored by the command, RP (Reference Position). RP gives the absolute value of the position as determined by the motion profiler. The command, DP, can be used to set the value of the reference position. For example, DP 0, defines the reference position of the X axis to be zero.

Third, the output of the motion profiler is filtered by the stepper smoothing filter. This filter adds a delay in the output of the stepper motor pulses. The amount of delay depends on the parameter which is specified by the command, KS. As mentioned earlier, there will always be some amount of stepper motor smoothing. The default value for KS is 2 which corresponds to a time constant of 6 sample periods.

Fourth, the output of the stepper smoothing filter is buffered and is available for input to the stepper motor driver. The pulses which are generated by the smoothing filter can be monitored by the command, TD (Tell Dual). TD gives the absolute value of the position as determined by actual output of the buffer. The command, DP sets the value of the step count register as well as the value of the reference position. For example, DP 0, defines the reference position of the X axis to be zero.



Motion Complete Trippoint

When used in stepper mode, the MC command will hold up execution of the proceeding commands until the controller has generated the same number of steps out of the step count register as specified in the commanded position. The MC trippoint (Motion Complete) is generally more useful than AM trippoint (After Motion) since the step pulses can be delayed from the commanded position due to stepper motor smoothing.

Using an Encoder with Stepper Motors

An encoder may be used on a stepper motor to check the actual motor position with the commanded position. If an encoder is used, it must be connected to the main encoder input. Note: The auxiliary encoder is not available while operating with stepper motors. The position of the encoder can be interrogated by using the command, TP. The position value can be defined by using the command, DE.

Note: Closed loop operation with a stepper motor is not possible.

Command Summary - Stepper Motor Operation

COMMAND	DESCRIPTION
DE	Define Encoder Position (When using an encoder)
DP	Define Reference Position and Step Count Register
IT	Motion Profile Smoothing - Independent Time Constant
KS	Stepper Motor Smoothing
MT	Motor Type (2,-2,2.5 or -2.5 for stepper motors)
RP	Report Commanded Position
TD	Report number of step pulses generated by controller
TP	Tell Position of Encoder

Operand Summary - Stepper Motor Operation

OPERAND	DESCRIPTION
_DEx	Contains the value of the step count register
_DPx	Contains the value of the main encoder
_ITx	Contains the value of the Independent Time constant for the 'x' axis
_KS	Contains the value of the Stepper Motor Smoothing Constant for the 'x' axis
_MT	Contains the motor type value for the 'x' axis
_RP	Contains the commanded position generated by the profiler

_TD	Contains the value of the step count register
_TP	Contains the value of the main encoder

Dual Loop (Auxiliary Encoder)

The DMC 1300 provides an interface for a second encoder for each axis except for axes configured for stepper motor operation. When used, the second encoder is typically mounted on the motor or the load, but may be mounted in any position. The most common use for the second encoder is backlash compensation, described below.

The auxiliary encoder may also be used for gearing. In this case, the auxiliary encoder input is used to monitor an encoder which is not under control of the DMC 1300. To use the auxiliary encoder for gearing, the master axis is specified as the auxiliary encoder and GR is used to specify the gear ratios. For more information, see previous section Electronic Gearingon page 76.

The second encoder may be a standard quadrature type, or it may provide pulse and direction. The controller also offers the provision for inverting the direction of the encoder rotation. The main and the auxiliary encoders are configured with the CE command. The command form is CE x,y,z,w (or a,b,c,d,e,f,g,h for controllers with more than 4 axes) where the parameters x,y,z,w each equal the sum of two integers m and n. m configures the main encoder and n configures the auxiliary encoder.

Using the CE Command

m=	Main Encoder	n=	Second Encoder
0	Normal quadrature	0	Normal quadrature
1	Pulse & direction	4	Pulse & direction
2	Reverse quadrature	8	Reversed quadrature
3	Reverse pulse & direction	12	Reversed pulse & direction

For example, to configure the main encoder for reversed quadrature, m=2, and a second encoder of pulse and direction, n=4, the total is 6, and the command for the X axis is

```
CE 6
```

Additional Commands for the Auxiliary Encoder

The command, DE x,y,z,w, can be used to define the position of the auxiliary encoders. For exa mple,

```
DE 0,500,-30,300
```

sets their initial values.

The positions of the auxiliary encoders may be interrogated with the command, DE?. For example

```
DE ?,,?
```

returns the value of the X and Z auxiliary encoders.

The auxiliary encoder position may be assigned to variables with the instructions

```
V1= _DEX
```

The command, TD XYZW, returns the current position of the auxiliary encoder.

The command, DV XYZW, configures the auxiliary encoder to be used for backlash compensation.

Backlash Compensation

There are two methods for backlash compensation using the auxiliary encoders:

Continuous dual loop

Sampled dual loop

To illustrate the problem, consider a situation in which the coupling between the motor and the load has a backlash. To compensate for the backlash, position encoders are mounted on both the motor and the load.

The continuous dual loop combines the two feedback signals to achieve stability. This method requires careful system tuning, and depends on the magnitude of the backlash. However, once successful, this method compensates for the backlash continuously.

The second method, the sampled dual loop, reads the load encoder only at the end point and performs a correction. This method is independent of the size of the backlash. However, it is effective only in point-to-point motion systems which require position accuracy only at the endpoint.

Example - Continuous Dual Loop

Note: In order to have a stable continuous dual loop system, the encoder on the motor must be of equal or higher resolution than the encoder on the load.

Connect the load encoder to the main encoder port and connect the motor encoder to the dual encoder port. The dual loop method splits the filter function between the two encoders. It applies the KP (proportional) and KI (integral) terms to the position error, based on the load encoder, and applies the KD (derivative) term to the motor encoder. This method results in a stable system.

The dual loop method is activated with the instruction DV (Dual Velocity), where

```
DV   1,1,1,1
```

activates the dual loop for the four axes and

```
DV   0,0,0,0
```

disables the dual loop.

Note that the dual loop compensation depends on the backlash magnitude, and in extreme cases will not stabilize the loop. The proposed compensation procedure is to start with KP=0, KI=0 and to maximize the value of KD under the condition DV1. Once KD is found, increase KP gradually to a maximum value, and finally, increase KI, if necessary.

Example - Sampled Dual Loop

In this example, we consider a linear slide which is run by a rotary motor via a lead screw. Since the lead screw has a backlash, it is necessary to use a linear encoder to monitor the position of the slide. For stability reasons, it is best to use a rotary encoder on the motor.

Connect the rotary encoder to the X-axis and connect the linear encoder to the auxiliary encoder of X. Assume that the required motion distance is one inch, and that this corresponds to 40,000 counts of the rotary encoder and 10,000 counts of the linear encoder.

The design approach is to drive the motor a distance, which corresponds to 40,000 rotary counts. Once the motion is complete, the controller monitors the position of the linear encoder and performs position corrections.

This is done by the following program.

Instruction	Interpretation
#DUALOOP	Label
CE 0	Configure encoder
DE0	Set initial value
PR 40000	Main move
BGX	Start motion
#Correct	Correction loop
AMX	Wait for motion completion
V1=10000-_DEX	Find linear encoder error
V2=-_TEX/4+V1	Compensate for motor error
JP#END,@ABS[V2]<2	Exit if error is small
PR V2*4	Correction move
BGX	Start correction
JP#CORRECT	Repeat
#END	
EN	

Command Summary - Using the Auxiliary Encoder

COMMAND	DESCRIPTION
CE	Configure Encoder Type
DE	Define dual (auxiliary) encoder position
DV	Set continous dual loop mode - see description below
GA	Set master axis for gearing - the auxiliary encoder input can be used for gearing
GR	Set gear ratio for gearing - the auxiliary encoder input can be used for gearing
TD	Tell dual (auxiliary) encoder input position.

Operand Summary - Using the Auxiliary Encoder

OPERAND	DESCRIPTION
_CEx	Contains the encoder configuration for the specified axis
_DEx	Contains the current position of the specified auxiliary encoder
_DVx	Contains a '1' or '0' if the specified axis is in continuous dual loop operation.
_GRx	Contains the value of the gear ratio for the specified axis
_TDx	Contains the position of the specified auxiliary encoder.

Motion Smoothing

The DMC 1300 controller allows the smoothing of the velocity profile to reduce the mechanical vibration of the system.

Trapezoidal velocity profiles have acceleration rates which change abruptly from zero to maximum value. The discontinuous acceleration results in jerk which causes vibration. The smoothing of the acceleration profile leads to a continuous acceleration profile and reduces the mechanical shock and vibration.

Using the IT and VT Commands (S curve profiling):

When operating with servo motors, motion smoothing can be accomplished with the IT and VT command. These commands filter the acceleration and deceleration functions to produce a smooth velocity profile. The resulting velocity profile, known as S curve, has continuous acceleration and results in reduced mechanical vibrations.

The smoothing function is specified by the following commands:

IT x,y,z,w	Independent time constant
VT n	Vector time constant

The command, IT, is used for smoothing independent moves of the type JG, PR, PA and the command, VT, is used to smooth vector moves of the type VM and LM.

The smoothing parameters, x,y,z,w and n are numbers between 0 and 1 and determine the degree of filtering. The maximum value of 1 implies no filtering, resulting in trapezoidal velocity profiles. Smaller values of the smoothing parameters imply heavier filtering and smoother moves.

The following example illustrates the effect of smoothing. Fig. 6.6 shows the trapezoidal velocity profile and the modified acceleration and velocity.

Note that the smoothing process results in longer motion time.

Example - Smoothing

PR 20000	Position
AC 100000	Acceleration
DC 100000	Deceleration
SP 5000	Speed
IT .5	Filter for S-curve
BG X	Begin

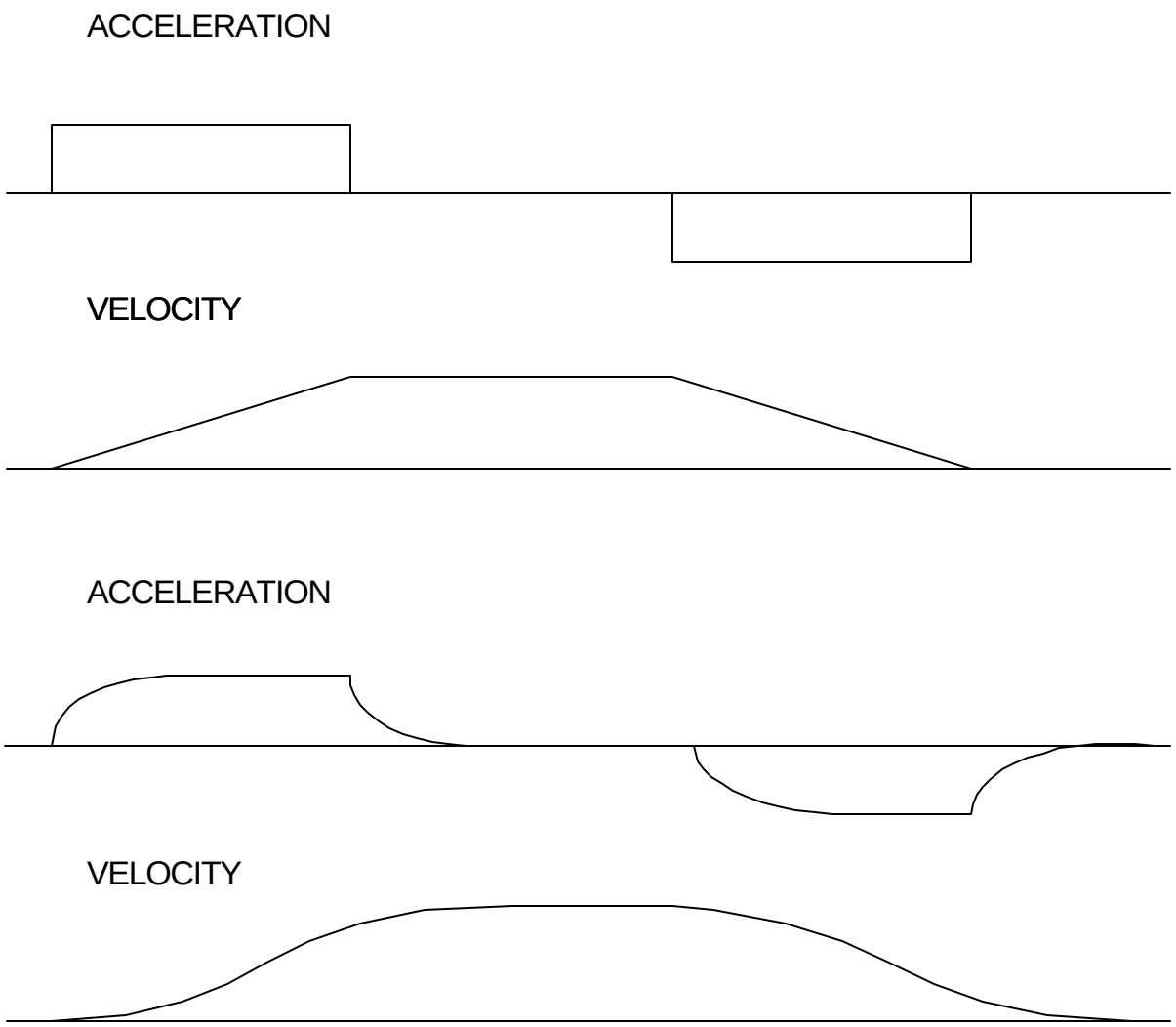


Figure 6.6 - Trapezoidal velocity and smooth velocity profiles

Using the KS Command (Step Motor Smoothing):

When operating with step motors, motion smoothing can be accomplished with the command, KS. The KS command smoothes the frequency of step motor pulses. Similar to the commands, IT and VT, this produces a smooth velocity profile.

The step motor smoothing is specified by the following command:

KS x,y,z,w	where x,y,z,w is an integer from 1 to 16 and represents the amount of smoothing

The command, IT, is used for smoothing independent moves of the type JG, PR, PA and the command, VT, is used to smooth vector moves of the type VM and LM.

The smoothing parameters, x,y,z,w and n are numbers between 0 and 16 and determine the degree of filtering. The minimum value of 1 implies no filtering, resulting in trapezoidal velocity profiles. Larger values of the smoothing parameters imply heavier filtering and smoother moves.

Note that KS is valid only for step motors.

Homing

The Find Edge (FE) and Home (HM) instructions may be used to home the motor to a mechanical reference. This reference is connected to the Home input line. The HM command initializes the motor to the encoder index pulse in addition to the Home input. The configure command (CN) is used to define the polarity of the home input.

The Find Edge (FE) and Home (HM) command status can be read from the Dual Port RAM in the Axis Buffers. These buffers include information on the state of the switch, as well as what phase of homing an axis is currently performing.

The Find Edge (FE) instruction is useful for initializing the motor to a home switch. The home switch is connected to the Homing Input. When the Find Edge command and Begin is used, the motor will accelerate up to the slew speed and slew until a transition is detected on the Homing line. The motor will then decelerate to a stop. A high deceleration value must be input before the find edge command is issued for the motor to decelerate rapidly after sensing the home switch. The velocity profile generated is shown in Fig. 6.7.

The Home (HM) command can be used to position the motor on the index pulse after the home switch is detected. This allows for finer positioning on initialization. The command sequence HM and BG causes the following sequence of events to occur.

1. Upon begin, motor accelerates to the slew speed. The direction of its motion is determined by the state of the homing input. A zero (GND) will cause the motor to start in the forward direction; +5V will cause it to start in the reverse direction. The CN command is used to define the polarity of the home input.
2. Upon detecting the home switch changing state, the motor begins decelerating to a stop.
3. The motor then traverses very slowly back until the home switch toggles again.
4. The motor then traverses forward until the encoder index pulse is detected.
5. The DMC 1300 defines the home position (0) as the position at which the index was detected.

Example:

Instruction	Interpretation
#HOME	Label
AC 1000000	Acceleration Rate
DC 1000000	Deceleration Rate
SP 5000	Speed for Home Search
HM X	Home X
BG X	Begin Motion
AM X	After Complete
MG "AT HOME"	Send Message

EN	End
#EDGE	Label
AC 2000000	Acceleration rate
DC 2000000	Deceleration rate
SP 8000	Speed
FE Y	Find edge command
BG Y	Begin motion
AM Y	After complete
MG "FOUND HOME"	Print message
DP,0	Define position as 0
EN	End

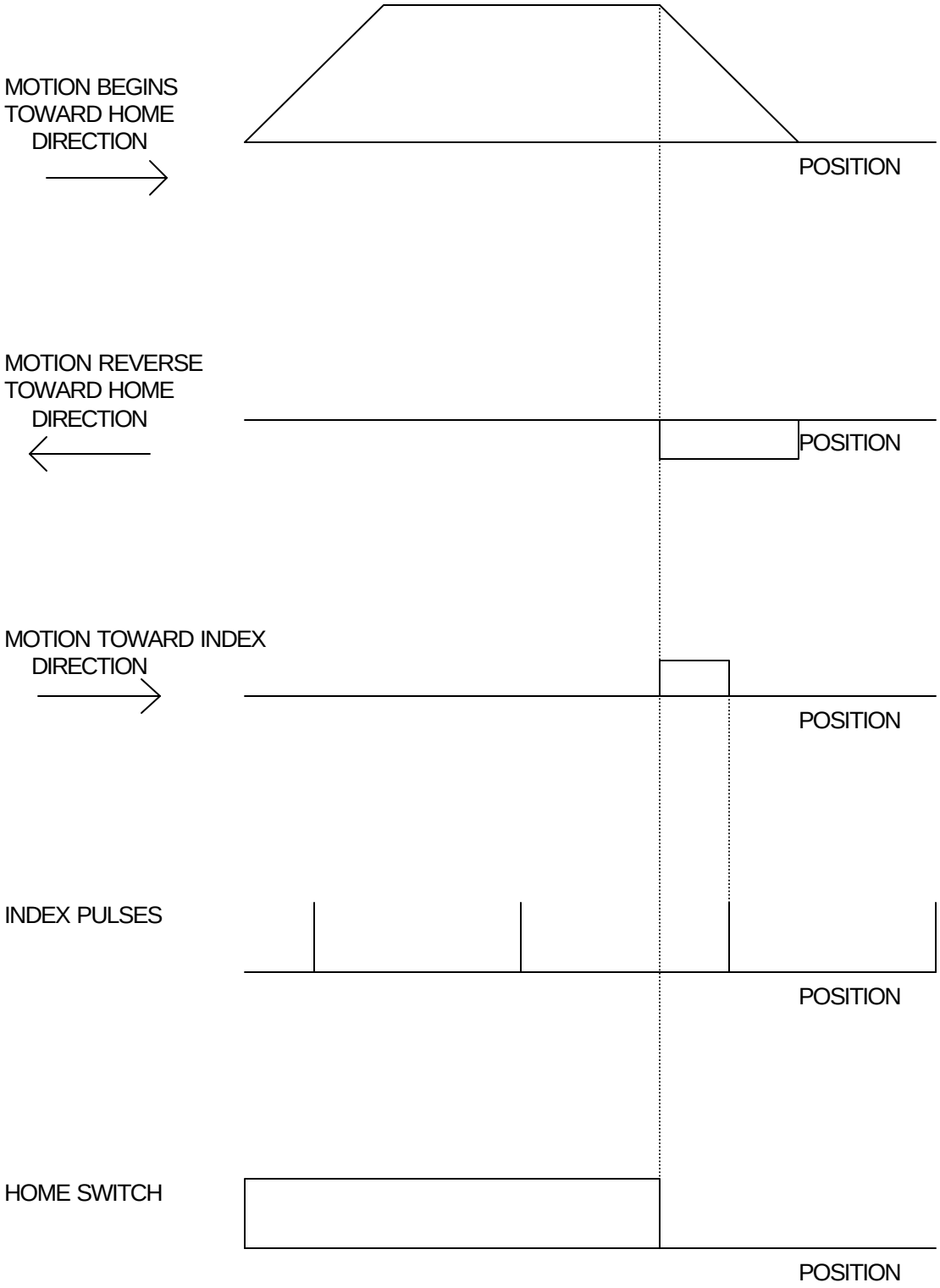


Figure 6.7 - Motion intervals in the Home sequence

High Speed Position Capture (Latch)

Often it is desirable to capture the position precisely for registration applications. The DMC 1300 provides a position latch feature. This feature allows the position of X,Y,Z or W to be captured within 25 microseconds of an external low input signal. The general inputs 1 through 4, and 9 through 12 correspond to each axis.

IN1 X-axis latch	IN 9 E-axis latch
IN2 Y-axis latch	IN10 F-axis latch
IN3 Z-axis latch	IN11 G-axis latch
IN4 W-axis latch	IN12 H-axis latch

Note: To insure a position capture within 25 microseconds, the input signal must be a transition from high to low.

Latch information can be read directly from the Dual Port RAM. Bit 2 of the Status #2 address in the axis buffer will indicate when the latch is armed. The latched position is also available in the corresponding axis buffer.

The DMC 1300 software commands, AL and RL, are used to arm the latch and report the latched position. The steps to use the latch are as follows:

1. Give the AL XYZW command, or ABCDEFGH for DMC-1380, to arm the latch for the specified axis or axes.
2. Test to see if the latch has occurred (Input goes low) by using the _AL X or Y or Z or W command. Example, V1=_ALX returns the state of the X latch into V1. V1 is 1 if the latch has not occurred.
3. After the latch has occurred, read the captured position with the RL XYZW command or _RL XYZW.

Note: The latch must be re-armed after each latching event.

Position Latch Example:

Instruction	Interpretation
#Latch	Latch program
JG,5000	Jog Y
BG Y	Begin motion on Y axis
AL Y	Arm Latch for Y axis
#Wait	#Wait label for loop
JP #Wait,_ALY=1	Jump to #Wait label if latch has not occurred
Result=_RLY	Set value of variable 'Result' equal to the report position of y axis
Result=	Print result
EN	End

THIS PAGE LEFT BLANK INTENTIONALLY

Chapter 7 Application Programming

Overview

The DMC 1300 provides a powerful programming language that allows users to customize the controller for their particular application. Programs can be downloaded into the DMC 1300 memory freeing the host for other tasks. However, the VME host can send commands to the controller at any time, even while a program is being executed.

In addition to standard motion commands, the DMC 1300 provides commands that allow the DMC 1300 to make its own decisions. These commands include conditional jumps, event triggers and subroutines. For example, the command JP#LOOP, n<10 causes a jump to the label #LOOP if the variable n is less than 10.

For greater programming flexibility, the DMC 1300 provides user-defined variables, arrays and arithmetic functions. For example, with a cut-to-length operation, the length can be specified as a variable in a program which the operator can change as necessary.

The following sections in this chapter discuss all aspects of creating applications programs.

Using the DMC 1300 Editor to Enter Programs

Application programs for the DMC 1300 may be created and edited either using the COMM 1300 editor or by writing directly to the Program Buffer.

In the COMM 1300 software, the DMC 1300 provides a line Editor for entering and modifying programs. The Edit mode is entered with the ED instruction. The ED command can only be given when the controller is not running a program.

In the Edit Mode, each program line is automatically numbered sequentially starting with 000. If no parameter follows the ED command, the editor prompter will default to the last line of the program in memory. If desired, the user can edit a specific line number or label by specifying a line number or label following ED.

Instruction	Interpretation
ED	Puts Editor at end of last program
ED 5	Puts Editor at line 5
ED #BEGIN	Puts Editor at label #BEGIN

PROGRAM MEMORY SPACE FOR THE DMC 1300:

DMC-1040	500 lines x 40 characters per line
DMC-1080	1000 lines x 80 characters per line
DMC-1040-MX	2000 lines x 40 characters per line

Line numbers appear as 000,001,002 and so on. Program commands are entered following the line numbers. Multiple commands may be given on a single line as long as the total number of characters doesn't exceed the limits given above.

While in the Edit Mode, the programmer has access to special instructions for saving, inserting and deleting program lines. These special instructions are listed below:

Edit Mode Commands

<RETURN>

Typing the return or enter key causes the current line of entered instructions to be saved. The editor will automatically advance to the next line. Thus, hitting a series of <RETURN> will cause the editor to advance a series of lines. Note, changes on a program line will not be saved unless a <return> is given.

<cntrl>P

The <cntrl>P command moves the editor to the previous line.

<cntrl>I

The <cntrl>I command inserts a line above the current line. For example, if the editor is at line number 2 and <cntrl>I is applied, a new line will be inserted between lines 1 and 2. This new line will be labeled line 2. The old line number 2 is renumbered as line 3.

<cntrl>D

The <cntrl>D command deletes the line currently being edited. For example, if the editor is at line number 2 and <cntrl>D is applied, line 2 will be deleted. The previous line number 3 is now renumbered as line number 2.

<cntrl>Q

The <cntrl>Q quits the editor mode. In response, the DMC 1300 will return a colon.

Programs may also be created by writing the ED command directly to the Program Buffer. This places the controller in the edit mode. The following commands are used to edit or create application programs in the Program Buffer.

(9A hex)	Deletes a line
(99 hex)	Inserts a line before the current one
(9B hex)	Displays the previous line
(9C hex)	Exits the Edit subsystem
(9D hex)	Saves a line

When creating a program, the first program line is loaded into the Program Buffer. 9D is then written to the Command Buffer and set by the Command Semaphore. This stores the first line in the application program. The second line is then written in the same manner. When editing a program, the current line is automatically displayed in the Program Buffer upon sending the ED command. This line is then edited using the same commands.

The following example shows how to load this simple program into the Program Buffer of a DMC 1340 in ASCII.

```
#TEST
MG"TEST 1"
IP1000
EN
```

1. Write the ED command to the controller Command Buffer to enter the editor mode.

Address	Value (hex)	Characters
40	45	E
41	44	D
42	0D	Return

2. Set the Command Semaphore (001) to load the command.

Address	Value (hex)	Characters
01	80	MSB set high

3. When the Command Semaphore is cleared, write MG"TEST 1" to the Program Buffer.

Address	Value (hex)	Characters
C0	4D	M
C1	47	G
C3	22	"
C4	54	T
C5	45	E
C6	53	S
C7	54	T
C8	20	Space
C9	31	1
CA	22	"
CB	0D	Return

4. Write 9D to the Command Buffer to save the line and advance to the next program line.

Address	Value (hex)	Characters
40	9D	Save current line
41	0D	Return

5. Set the Command Semaphore (001) to load the command.

Address	Value (hex)	Characters
01	80	MSB set high

6. Wait for the Command Semaphore to clear. Load the command IP1000 into the Program Buffer.

Address	Value (hex)	Characters
C0	49	I
C1	50	P
C2	31	1
C3	30	0
C4	30	0
C5	30	0
C6	0D	Return

7. Write 9D to the Command Buffer to save the line and advance to the next program line.

8. Set the Command Semaphore (001) to load the command.

9. Wait for the Command Semaphore to clear. Load the command EN into the Program Buffer.

Address	Value (hex)	Characters
C0	45	E
C1	4E	N
C2	0D	Return

10. Write 9D to the Command Buffer to save the line and advance to the next program line.
11. Set the Command Semaphore (001) to load the command.
12. Wait for the Command Semaphore to clear. Write 9C to the Command Buffer to quit the editor mode. Set the Command Semaphore (001) to the load the command.
13. Write XQ to the Command Buffer to execute the application program.

Address	Value (hex)	Characters
40	58	X
41	51	Q
42	0D	Return

14. Set the Command Semaphore (001) to load the command and begin execution of the program.

Program Format

A DMC 1300 program consists of DMC 1300 instructions combined to solve a machine control application. Action instructions, such as starting and stopping motion, are combined with Program Flow instructions to form the complete program. Program Flow instructions evaluate real-time conditions, such as elapsed time or motion complete, and alter program flow accordingly.

Each DMC 1300 instruction in a program must be separated by a delimiter. Valid delimiters are the semicolon (;) or carriage return. The semicolon is used to separate multiple instructions on a single program line where the maximum number of instructions on a line is limited by 38 characters. A carriage return enters the final command on a program line.

Using Labels in Programs

All DMC 1300 programs must begin with a label and end with an End (EN) statement. Labels start with the pound (#) sign followed by a maximum of seven characters. The first character must be a letter; after that, numbers are permitted. Spaces are not permitted.

The maximum number of labels depends on the controller: 126 for 1-4 axes, 510 for 1-4 axes with the -MX option, and 254 for controllers with 5 or more axes.

Valid labels

Label
#BEGIN
#SQUARE
#X1
#BEGIN1

Invalid labels

Label	Problem

#1Square	Can not use number to begin a label
#SQUAREPEG	Can not use more than 7 characters in a label

Program Example

Instruction	Interpretation
#START	Beginning of the Program
PR 10000,20000	Specify relative distances on X and Y axes
BG XY	Begin Motion
AM	Wait for motion complete
WT 2000	Wait 2 sec
JP #START	Jump to label START
EN	End of Program

The above program moves X and Y 10000 and 20000 units. After the motion is complete, the motors rest for 2 seconds. The cycle repeats indefinitely until the stop command is issued.

Special Labels

The DMC 1300 has some special labels, which are used to define input interrupt subroutines, limit switch subroutines, error handling subroutines, and command error subroutines. See section on "Automatic Subroutines for Monitoring Conditions" on page 114.

#ININT	Label for Input Interrupt subroutine
#LIMSWI	Label for Limit Switch subroutine
#POSERR	Label for excess Position Error subroutine
#MCTIME	Label for timeout on Motion Complete trip point
#CMDERR	Label for incorrect command subroutine

Commenting Programs

Using the command, NO

The DMC 1300 provides a command, NO, for commenting programs. This command allows the user to include up to 37 characters on a single line after the NO command and can be used to include comments from the programmer as in the following example:

```
#PATH
NO 2-D CIRCULAR PATH
VMXY
NO VECTOR MOTION ON X AND Y
VS 10000
NO VECTOR SPEED IS 10000
VP -4000,0
NO BOTTOM LINE
CR 1500,270,-180
```

```
NO HALF CIRCLE MOTION
VP 0,3000
NO TOP LINE
CR 1500,90,-180
NO HALF CIRCLE MOTION
VE
NO END VECTOR SEQUENCE
BGS
NO BEGIN SEQUENCE MOTION
EN
NO END OF PROGRAM
```

Note: The NO command is an actual controller command. Therefore, inclusion of the NO commands will require process time by the controller.

Using REM Statements with the Galil Terminal Software.

If you are using Galil software to communicate with the DMC 1300 controller, you may also include REM statements. 'REM' statements begin with the word 'REM' and may be followed by any comments which are on the same line. The Galil terminal software will remove these statements when the program is downloaded to the controller. For example:

```
#PATH
REM 2-D CIRCULAR PATH
VMXY
REM VECTOR MOTION ON X AND Y
VS 10000
REM VECTOR SPEED IS 10000
VP -4000,0
REM BOTTOM LINE
CR 1500,270,-180
REM HALF CIRCLE MOTION
VP 0,3000
REM TOP LINE
CR 1500,90,-180
REM HALF CIRCLE MOTION
VE
REM END VECTOR SEQUENCE
BGS
REM BEGIN SEQUENCE MOTION
EN
REM END OF PROGRAM
```

These REM statements will be removed when this program is downloaded to the controller.

Executing Programs - Multitasking

The DMC 1300 can run up to four independent programs simultaneously. These programs are called threads and are numbered 0 through 3, where 0 is the main one. Multitasking is useful for executing independent operations such as PLC functions that occur independently of motion.

The main thread differs from the others in the following ways:

1. Only the main thread may use the input command, IN.

2. When input interrupts are implemented for limit switches, position errors or command errors, the subroutines are executed in thread 0.

To begin execution of the various programs, use the following instruction:

XQ #A, n

Where n indicates the thread number. To halt the execution of any thread, use the instruction

HX n

where n is the thread number.

Note that both the XQ and HX commands can be performed by an executing program.

Multitasking Example: Producing Waveform on Output 1 Independent of a Move.

Instruction	Interpretation
#TASK1	Task1 label
AT0	Initialize reference time
CB1	Clear Output 1
#LOOP1	Loop1 label
AT 10	Wait 10 msec from reference time
SB1	Set Output 1
AT -40	Wait 40 msec from reference time, then initialize reference
CB1	Clear Output 1
JP #LOOP1	Repeat Loop1
#TASK2	Task2 label
XQ #TASK1,1	Execute Task1
#LOOP2	Loop2 label
PR 1000	Define relative distance
BGX	Begin motion
AMX	After motion done
WT 10	Wait 10 msec
JP #LOOP2,@IN[2]=1	Repeat motion unless Input 2 is low
HX	Halt all tasks

The program above is executed with the instruction XQ #TASK2,0 which designates TASK2 as the main thread (ie. Thread 0). #TASK1 is executed within TASK2.

Debugging Programs

The DMC 1300 provides commands and operands which are useful in debugging application programs. These commands include interrogation commands to monitor program execution, determine the state of the controller and the contents of the controllers program, array, and variable space. Operands also contain important status information which can help to debug a program.

Trace Commands

The trace command causes the controller to send each line in a program to the host computer immediately prior to execution. Tracing is enabled with the command, TR1. TR0 turns the trace function off. Note: When the trace function is enabled, the line numbers as well as the command line will be displayed as each command line is executed. The status of the trace command can be read at Bit 6 of the General Status register (010).

Single Stepping

The trace command can be used in conjunction with the Program Buffer Control (028) to single step through a program. By setting both the trace and the Program Buffer Control to 1, each line will be displayed as executed, and program flow will not proceed until the Program Buffer has been cleared by the host. This allows for diagnostics of an application program.

Error Code Command

When there is a program error, the DMC 1300 halts the program execution at the point where the error occurs. To display the last line number of program execution, issue the command, MG _ED.

The user can obtain information about the type of error condition that occurred by using the command, TC1. This command reports back a number and a text message which describes the error condition. The command, TC0 or TC, will return the error code without the text message. For more information about the command, TC, see the Command Reference.

Error codes are also read through the Dual Port RAM. Bits 1 and 0 of the General Status register (010) will indicate an error in either an application program or a command respectively. The corresponding error is found at 012 of the General Registers for a Command Buffer error or 013 of the General Registers for an Application Program error. A list of all the error codes is found under the TC command.

Stop Code Command

The status of motion for each axis can be determined by using the stop code command, SC. This can be useful when motion on an axis has stopped unexpectedly. The command SC will return a number representing the motion status. See the command reference for further information. The command SC1 will return the number and the textual explanation of the motion status. The stop code is also available in Axis Buffers of the Dual Port RAM.

RAM Memory Interrogation Commands

For debugging the status of the program memory, array memory, or variable memory, the DMC 1300 has several useful commands. The command, DM ?, will return the number of array elements currently available. The command, DA ?, will return the number of arrays which can be currently defined. For example, a standard DMC 1310 will have a maximum of 1600 array elements in up to 14 arrays. If an array of 100 elements is defined, the command DM ? will return the value 1500 and the command DA ? will return 13.

Operands

In general, all operands provide information which may be useful in debugging an application program. Below is a list of operands which are particularly valuable for program debugging. To display the value of an operand, the message command may be used. For example, since the operand, _ED contains the last line of program execution, the command MG _ED will display this line number.

_ED contains the last line of program execution. Useful to determine where program stopped.

_DL contains the number of available labels.

_UL contains the number of available variables.

_DA contains the number of available arrays.

_DM contains the number of available array elements.

_AB contains the state of the Abort Input

_FLx contains the state of the forward limit switch for the 'x' axis

_RLx contains the state of the reverse limit switch for the 'x' axis

Debugging Example:

The following program has an error. It attempts to specify a relative movement while the X-axis is already in motion. When the program is executed, the controller stops at line 003. The user can then query the controller using the command, TC1. The controller responds with the corresponding explanation:

:ED	Edit Mode
000 #A	Program Label
001 PR1000	Position Relative 1000
002 BGX	Begin
003 PR5000	Position Relative 5000
004 EN	End
<cntrl> Q	Quit Edit Mode
:XQ #A	Execute #A
?003 PR5000	Error on Line 3
:TC1	Tell Error Code
?7 Command not valid while running.	Command not valid while running
:ED 3	Edit Line 3
003 AMX;PR5000;BGX	Add After Motion Done
<cntrl> Q	Quit Edit Mode
:XQ #A	Execute #A

In the Dual Port RAM, Bit 1of the General Status (010) will be set when the program executes line 3. Upon being set, the Application Error Code register (013) will read 07, corresponding to the ‘Command not valid while running’ error. This error will remain valid until cleared by the host or another error occurs.

Program Flow Commands

The DMC 1300 provides instructions to control program flow. The DMC 1300 program sequencer normally executes program instructions sequentially. The program flow can be altered with the use of event triggers, trippoints, and conditional jump statements.

Event Triggers & Trippoints

To function independently from the host computer, the DMC 1300 can be programmed to make decisions based on the occurrence of an event. Such events include waiting for motion to be comp lete, waiting for a specified amount of time to elapse, or waiting for an input to change logic levels.

The DMC 1300 provides several event triggers that cause the program sequencer to halt until the specified event occurs. Normally, a program is automatically executed sequentially one line at a time. When an event trigger instruction is decoded, however, the actual program sequence is halted. The program sequence does not continue until the event trigger is "tripped". For example, the motion complete trigger can be used to separate two move sequences in a program. The commands for the second move sequence will not be executed until the motion is complete on the first motion sequence. In this way, the DMC 1300 can make decisions based on its own status or external events without intervention from a host computer.

DMC 1300 Event Triggers

Command	Function
AM X Y Z W or S (A B C D E F G H)	Halts program execution until motion is complete on the specified axes or motion sequence(s). AM with no parameter tests for motion complete on all axes. This command is useful for separating motion sequences in a program.
AD X or Y or Z or W (A or B or C or D or E or F or G or H)	Halts program execution until position command has reached the specified relative distance from the start of the move. Only one axis may be specified at a time.
AR X or Y or Z or W (A or B or C or D or E or F or G or H)	Halts program execution until after specified distance from the last AR or AD command has elapsed. Only one axis may be specified at a time.
AP X or Y or Z or W (A or B or C or D or E or F or G or H)	Halts program execution until after absolute position occurs. Only one axis may be specified at a time.
MF X or Y or Z or W (A or B or C or D or E or F or G or H)	Halt program execution until after forward motion reached absolute position. Only one axis may be specified. If position is already past the point, then MF will trip immediately. Will function on geared axis.
MR X or Y or Z or W (A or B or C or D or E or F or G or H)	Halt program execution until after reverse motion reached absolute position. Only one axis may be specified. If position is already past the point, then MR will trip immediately. Will function on geared axis.
MC X or Y or Z or W (A or B or C or D or E or F or G or H)	Halt program execution until after the motion profile has been completed and the encoder has entered or passed the specified position. TW x,y,z,w sets timeout to declare an error if not in position. If timeout occurs, then the trippoint will clear and the stopcode will be set to 99. An application program will jump to label #MCTIME.
AI +/- n	Halts program execution until after specified input is at specified logic level. n specifies input line. Positive is high logic level, negative is low level. n=1 through 8 for DMC-1010 to 1040. n=1 through 24 for DMC-1050 to 1080.
AS X Y Z W S (A B C D E F G H)	Halts program execution until specified axis has reached its slew speed.
AT +/-n	Halts program execution until n msec from reference time. AT 0 sets reference. AT n waits n msec from reference. AT -n waits n msec from reference and sets new reference after elapsed time.
AV n	Halts program execution until specified distance along a coordinated path has occurred.
WT n	Halts program execution until specified time in msec has elapsed.

Event Trigger Examples:

Event Trigger - Multiple Move Sequence

The AM trippoint is used to separate the two PR moves. If AM is not used, the controller returns a ? for the second PR command because a new PR cannot be given until motion is complete.

Instruction	Interpretation
#TWOMOVE	Label
PR 2000	Position Command
BGX	Begin Motion
AMX	Wait for Motion Complete
PR 4000	Next Position Move
BGX	Begin 2nd move
EN	End program

Event Trigger - Set Output after Distance

Set output bit 1 after a distance of 1000 counts from the start of the move. The accuracy of the trippoint is the speed multiplied by the sample period.

Instruction	Interpretation
#SETBIT	Label
SP 10000	Speed is 10000
PA 20000	Specify Absolute position
BGX	Begin motion
AD 1000	Wait until 1000 counts
SB1	Set output bit 1
EN	End program

Event Trigger - Repetitive Position Trigger

To set the output bit every 10000 counts during a move, the AR trippoint is used as shown in the next example.

Instruction	Interpretation
#TRIP	Label
JG 50000	Specify Jog Speed
BGX;n=0	Begin Motion
#REPEAT	# Repeat Loop
AR 10000	Wait 10000 counts
TPX	Tell Position
SB1	Set output 1
WT50	Wait 50 msec
CB1	Clear output 1
n=n+1	Increment counter

```
JP #REPEAT,n<5          Repeat 5 times
STX                     Stop
EN                      End
```

Event Trigger - Start Motion on Input

This example waits for input 1 to go low and then starts motion. Note: The AI command actually halts execution of the program until the input occurs. If you do not want to halt the program sequences, you can use the Input Interrupt function (II) or use a conditional jump on an input, such as JP #GO,@IN[1] = -1.

Instruction	Interpretation
#INPUT	Program Label
AI-1	Wait for input 1 low
PR 10000	Position command
BGX	Begin motion
EN	End program

Event Trigger - Set output when At speed

Instruction	Interpretation
#ATSPEED	Program Label
JG 50000	Specify jog speed
AC 10000	Acceleration rate
BGX	Begin motion
ASX	Wait for at slew speed 50000
SB1	Set output 1
EN	End program

Event Trigger - Change Speed along Vector Path

The following program changes the feedrate or vector speed at the specified distance along the vector. The vector distance is measured from the start of the move or from the last AV command.

Instruction	Interpretation
#VECTOR	Label
VMXY;VS 5000	Coordinated path
VP 10000,20000	Vector position
VP 20000,30000	Vector position
VE	End vector
BGS	Begin sequence
AV 5000	After vector distance
VS 1000	Reduce speed
EN	End

Event Trigger - Multiple Move with Wait

This example makes multiple relative distance moves by waiting for each to be complete before executing new moves.

Instruction	Interpretation
#MOVES	Label
PR 12000	Distance
SP 20000	Speed
AC 100000	Acceleration
BGX	Start Motion
AD 10000	Wait a distance of 10,000 counts
SP 5000	New Speed
AMX	Wait until motion is completed
WT 200	Wait 200 ms
PR -10000	New Position
SP 30000	New Speed
AC 150000	New Acceleration
BGX	Start Motion
EN	End

Example - creating an output Waveform Using AT

The following program causes Output 1 to be high for 10 msec and low for 40 msec. The cycle repeats every 50 msec.

Instruction	Interpretation
#OUTPUT	Program label
AT0	Initialize time reference
SB1	Set Output 1
#LOOP	Loop
AT 10	After 10 msec from reference,
CB1	Clear Output 1
AT -40	Wait 40 msec from reference and reset reference
SB1	Set Output 1
JP #LOOP	Jump to location #LOOP and continue executing commands
EN	End of program

Conditional Jumps

The DMC 1300 provides Conditional Jump (JP) and Conditional Jump to Subroutine (JS) instructions for branching to a new program location. Program execution will continue at the location specified by the JP and JS command if the jump condition is satisfied. Conditional jumps are useful for testing events in real-time since they allow the DMC 1300 to make decisions without a host computer. For example, the DMC 1300 can begin execution at a specified label or line number based on the state of an input line.

Using the JP Command:

The JP command will cause the controller to execute commands at the location specified by the label or line number if the condition of the jump statement is satisfied. If no condition is specified, program execution will automatically jump to the specified line. If the condition is not satisfied, the controller continues to execute the next commands in program sequence.

Using the JS Command:

The JS command is significantly different from the JP command. When the condition specified by the JS command is satisfied, the controller will begin execution at the program location specified by the line or label number. However, when the controller reaches an end statement, EN, the controller will jump back to the location of the JS command and resume executing the next commands. This is known as jumping to a subroutine. For more information, see section

Conditional Statements

The conditional statement is satisfied if it evaluates to any value other than zero. The conditional statement can be any valid DMC 1300 numeric operand, including variables, array elements, numeric values, functions, keywords, and arithmetic expressions. If no conditional statement is given, the jump will always occur.

Examples:

Number	V1=6
Numeric Expression	V1=V7*6
	@ABS[V1]>10
Array Element	V1<Count[2]
Variable	V1<V2
Internal Variable	_TPX=0
	_TVX>500
I/O	V1>@AN[2]
	@IN[1]=0

Examples Using JP and JS

Instruction	Interpretation
JP #Loop, COUNT<10	Jump to #Loop if the variable, COUNT, is less than 10
JS #MOVE2,@IN[1]=1	Jump to subroutine #MOVE2 if input 1 is logic level high. After the subroutine MOVE2 is executed, the program sequencer returns to the main program location where the subroutine was called.
JP #BLUE,@ABS[V2]>2	Jump to #BLUE if the absolute value of variable, V2, is greater than 2
JP #C,V1*V7<=V8*V2	Jump to #C if the value of V1 times V7 is less than or equal to the value of V8*V2
JP#A	Jump to #A

Example Using JP command:

Move the X motor to absolute position 1000 counts and back to zero ten times. Wait 100 msec between moves.

Instruction	Interpretation
#BEGIN	Begin Program
COUNT=10	Initialize loop counter
#LOOP	Begin loop
PA 1000	Position absolute 1000
BGX	Begin move
AMX	Wait for motion complete
WT 100	Wait 100 msec
PA 0	Position absolute 0
BGX	Begin move
AMX	Wait for motion complete
WT 100	Wait 100 msec
COUNT=COUNT-1	Decrement loop counter
JP #LOOP,COUNT>0	Test for 10 times through loop
EN	End Program

Command Format - JP and JS

FORMAT:	DESCRIPTION
JS destination, logical condition	Jump to subroutine if logical condition is satisfied
JP destination, logical condition	Jump to location if logical condition is satisfied

The destination is a program line number or label where the program sequencer will jump if the specified condition is satisfied. Note that the line number of the first line of program memory is 0. The comma designates "IF". The logical condition tests two operands with logical operators.

Logical Operators:

OPERATOR	DESCRIPTION
<	less than
>	greater than
=	equal to
<=	less than or equal to
>=	greater than or equal to
<>	not equal

Subroutines

A subroutine is a group of instructions beginning with a label and ending with an end command (EN). Subroutines are called from the main program with the jump subroutine instruction JS, followed by a label or line number, and conditional statement. Up to 8 subroutines can be nested. After the subroutine is executed, the program sequencer returns to the program location where the subroutine was called unless the subroutine stack is manipulated as described in the following section.

Example - Using a Subroutine

Subroutine to draw a square 500 counts on each side. The square starts at vector position 1000,1000.

Instruction	Interpretation
#M	Begin main program
CB1	Clear Output Bit 1 (pick up pen)
VMXY	Specify vector motion between X and Y axes
VP 1000,1000;VE;BGS	Define vector position; move pen
AMS	Wait for after motion trippoint
SB1	Set Output Bit 1 (put down pen)
JS #Square;CB1	Jump to square subroutine
EN	End main program
#Square	Square subroutine
V1=500;JS #L	Define length of side, Jump to subroutine #L
V1=-V1;JS #L	Switch direction, Jump to subroutine #L
EN	End subroutine #Square
#L;PR V1,V1;BGX	Subroutine #L, Define relative position movement on X and Y; Begin motion
AMX;BGY;AMY	After motion on X, Begin Y, Wait for motion on Y to complete
EN	End subroutine #L

Stack Manipulation

It is possible to manipulate the subroutine stack by using the ZS command. Every time a JS instruction, interrupt or automatic routine (such as #POSERR or #LIMSWI) is executed, the subroutine stack is incremented by 1. Normally the stack is restored with an EN instruction. Occasionally it is desirable not to return back to the program line where the subroutine or interrupt was called. The ZS1 command clears 1 level of the stack. This allows the program sequencer to continue to the next line. The ZS0 command resets the stack to its initial value. For example, if a limit occurs and the #LIMSWI routine is executed, it is often desirable to restart the program sequence instead of returning to the location where the limit occurred. To do this, give a ZS command at the end of the #LIMSWI routine.

Automatic Subroutines for Monitoring Conditions

Often it is desirable to monitor certain conditions continuously without tying up the host or DMC 1300 program sequences. The DMC 1300 can monitor several important conditions in the background. These conditions include checking for the occurrence of a limit switch, a defined input, position error, or a command error. Automatic monitoring is enabled by inserting a special, predefined label in the applications program. The pre-defined labels are:

SUBROUTINE	DESCRIPTION
#LIMSWI	Limit switch on any axis goes low
#ININT	Input specified by II goes low
#POSERR	Position error exceeds limit specified by ER
#MCTIME	Motion Complete timeout occurred. Timeout period set by TW command
#CMDERR	Bad command given

For example, the #POSERR subroutine will automatically be executed when any axis exceeds its position error limit. The commands in the #POSERR subroutine could decode which axis is in error and take the appropriate action. In another example, the #ININT label could be used to designate an input interrupt subroutine. When the specified input occurs, the program will be executed automatically.

NOTE: An application program must be running for automatic monitoring to function.

Example - Limit Switch

This program prints a message upon the occurrence of a limit switch. Note, for the #LIMSWI routine to function, the DMC 1300 must be executing an applications program from memory. This can be a very simple program that does nothing but loop on a statement, such as #LOOP;JP #LOOP;EN. Motion commands, such as JG 5000 can still be sent from the PC even while the "dummy" applications program is being executed.

Instruction	**Interpretation**
#LOOP	Dummy Program
JP #LOOP;EN	Jump to Loop
#LIMSWI	Limit Switch Label
MG "LIMIT OCCURRED"	Print Message
RE	Return to main program
XQ #LOOP	Execute Dummy Program
JG 5000	Jog X axis at rate of 5000 counts / sec
BGX	Begin motion on X axis

NOTE: Regarding the #LIMSWI Routine.

Now, when a forward limit switch occurs on the X axis, the #LIMSWI subroutine will be executed.

1) The RE command is used to return from the #LIMSWI subroutine.

2) The #LIMSWI will continue to be executed until the limit switch is cleared (goes high).

3) The #LIMSWI routine will only be executed when the motor is being commanded to move.

Example - Position Error

Instruction	Interpretation
#LOOP	Dummy Program
JP #LOOP;EN	Loop
#POSERR	Position Error Routine
V1=_TEX	Read Position Error
MG "EXCESS POSITION ERROR"	Print Message
MG "ERROR=",V1=	Print Error
RE	Return from Error

While running the 'dummy' program, if the position error on the X axis exceeds that value specified by the ER command, the #POSERR routine will execute.

NOTE: The RE command is used to return from the #POSERR subroutine

NOTE: The #POSERR routine will continue to be executed until the position error is cleared (is less than the ER limit).

Example - Input Interrupt

Instruction	Interpretation
#A	Label
II1	Input Interrupt on 1
JG 30000,,,60000	Jog
BGXW	Begin Motion
#LOOP;JP#LOOP;EN	Loop
#ININT	Input Interrupt
STXW;AM	Stop Motion
#TEST;JP #TEST, @IN[1]=0	Test for Input 1 still low
JG 30000,,,6000	Restore Velocities
BGXW;RI	Begin motion and Return to Main Program
EN	

NOTE: Use the RI command to return from #ININT subroutine.

Example - Motion Complete Timeout

Instruction	Interpretation
#BEGIN	Begin main program
TW 1000	Set the time out to 1000 ms
PA 10000	Position Absolute command
BGX	Begin motion
MCX	Motion Complete trip point
EN	End main program
#MCTIME	Motion Complete Subroutine

MG “X fell short”	Send out a message
EN	End subroutine

This simple program will issue the message “X fell short” if the X axis does not reach the commanded position within 1 second of the end of the profiled mo ve.

Example - Bad Command

Instruction	Interpretation
#BEGIN	Begin main program
IN "ENTER SPEED", SPEED	Prompt for speed
JG SPEED;BGX;	Begin motion
JP #BEGIN	Repeat
EN	End main program
#CMDERR	Command error utility
JP#DONE,_ED<>2	Check if error on line 2
JP#DONE,_TC<>6	Check if out of range
MG "SPEED TOO HIGH"	Send message
MG "TRY AGAIN"	Send message
ZS1	Adjust stack
JP #BEGIN	Return to main program
#DONE	End program if other error
ZS0	Zero stack
EN	End program

The above program prompts the operator to enter a jog speed. If the operator enters a number out of range (greater than 8 million), the #CMDERR routine will be executed prompting the operator to enter a new number.

Mathematical and Functional Expressions

Mathematical Expressions

For manipulation of data, the DMC 1300 provides the use of the following mathematical operators:

OPERATOR	FUNCTION
+	Addition
-	Subtraction
*	Multiplication
/	Division
&	Logical And (Bit-wise)
|	Logical Or (On some computers, a solid vertical line appears as a broken line)
()	Parenthesis

The numeric range for addition, subtraction and multiplication operations is +/-2,147,483,647.9999. The precision for division is 1/65,000.

Mathematical operations are executed from left to right. Calculations within a parentheses have precedence.

Examples of MATHEMATICAL EXPRESSION

SPEED=7.5*V1/2	The variable, SPEED, is equal to 7.5 multiplied by V1 and divided by 2
COUNT=COUNT+2	The variable, COUNT, is equal to the current value plus 2.
RESULT=_TPX-(@COS[45]*40)	Puts the position of X - 28.28 in RESULT. 40 * cosine of 45° is 28.28
TEMP=@IN[1]&@IN[2]	TEMP is equal to 1 only if Input 1 and Input 2 are high

Bit-Wise Operators

The mathematical operators & and | are bit-wise operators. The operator, &, is a Logical And. The operator, |, is a Logical Or. These operators allow for bit-wise operations on any valid DMC 1300 numeric operand, including variables, array elements, numeric values, functions, keywords, and arithmetic expressions. The bit-wise operators may also be used with strings.

Bit-wise operators are useful for separating characters from an input string. When using the input command for string input, the input variable holds 6 bytes of data. Each byte is eight bits, so a number represented as 32 bits of integer and 16 bits of fraction. Each ASCII character is represented as one byte (8 bits), therefore the input variable can hold a six character string. The first character of the string will be placed in the top byte of the variable and the last character will be placed in the lowest significant byte of the fraction. The characters can be individually separated by using bit-wise operations as illustrated in the following example:

Instruction	**Interpretation**
#TEST	Begin main program
IN "ENTER",LEN{S6}	Input character string up to 6 characters into variable ‘LEN’
FLEN=@FRAC[LEN]	Define variable ‘FLEN’ as fractional part of variable ‘LEN’
FLEN=$10000*FLEN	Shift FLEN by 32 bits (Convert fraction, FLEN, to integer)

LEN1=(FLEN&$00FF)*$1000000	Set 4th byte of FLEN = 1st byte of variable LEN1
LEN2=(FLEN&$FF00)*$10000	Set 3rd byte of FLEN = 1st byte of variable of LEN2
LEN3=(LEN&$000000FF)*$1000000	Set 1st byte of variable LEN3 = 4th byte of LEN
LEN4=(LEN&$0000FF00)*$10000	Set 1st byte of variable LEN4 = 3rd byte of LEN
LEN5=(LEN&$00FF0000)*$100	Set 1st byte of variable LEN5 = 2nd byte of LEN
LEN6=(LEN&$FF000000)	Set 1st byte of variable LEN6 = 1st byte of LEN
MG LEN6 {S1}	Display 'LEN6' as string message of 1 char
MG LEN5 {S1}	Display 'LEN5' as string message of 1 char
MG LEN4 {S1}	Display 'LEN4' as string message of 1 char
MG LEN3 {S1}	Display 'LEN3' as string message of 1 char
MG LEN2 {S1}	Display 'LEN2' as string message of 1 char
MG LEN1 {S1}	Display 'LEN1' as string message of 1 char
EN	

This program will accept a string input of up to 6 characters, parse each character, and then display each character. Notice also that the values used for masking are represented in hexadecimal (as denoted by the preceding '$'). For more information, see section *Sending Messages.*

To illustrate further, if the user types in the string "TESTME" at the input prompt, the controller will respond with the following:

T	Response from command MG LEN6 {S1}
E	Response from command MG LEN5 {S1}
S	Response from command MG LEN4 {S1}
T	Response from command MG LEN3 {S1}
M	Response from command MG LEN2 {S1}
E	Response from command MG LEN1 {S1}

Functions

FUNCTION	DESCRIPTION
@SIN[n]	Sine of n (n in degrees, resolution of 1/64,000 degrees, max +/- 4 billion)
@COS[n]	Cosine of n (n in degrees, resolution of 1/64,000 degrees, max +/- 4 billion)
@COM[n]	1's Compliment of n
@ABS[n]	Absolute value of n
@FRAC[n]	Fraction portion of n
@INT[n]	Integer portion of n
@RND[n]	Round of n (Rounds up if the fractional part of n is .5 or greater)
@SQR[n]	Square root of n (Accuracy is +/-.004)
@IN[n]	Return status of digital input n
@OUT[n]	Return status of digital output n

@AN[n]	Return voltage measured at analog input n

Functions may be combined with mathematical expressions. The order of execution of mathematical expressions is from left to right and can be over-ridden by using parentheses.

Examples - Using Functions

V1=@ABS[V7]	The variable, V1, is equal to the absolute value of variable V7.
V2=5*@SIN[POS]	The variable, V2, is equal to five times the sine of the variable, POS.
V3=@IN[1]	The variable, V3, is equal to the digital value of input 1.
V4=2*(5+@AN[5])	The variable, V4, is equal to the value of analog input 5 plus 5, then multiplied by 2.

Variables

The maximum number of variables available with a DMC 1300 controller depends on the controller configuration: 126 variables are available for 1-4 axes controllers, 510 variables with 1-4 axes and the -MX option, and 254 variables with controllers of 5 or mor axes. These variables can be numbers or strings. Variables are useful in applications where specific parameters, such as position or speed, must be able to change. Variables can be assigned by an operator or determined by program calculations. For example, a cut-to-length application may require that a cut length be variable.

Each variable is defined by a name which can be up to eight characters. The name must start with an alphabetic character, however, numbers are permitted in the rest of the name. Spaces are not permitted. Variable names should not be the same as DMC 1300 instructions. For example, PR is not a good choice for a variable name.

In addition to the local variables, the DMC 1300 has 64 variables that are stored as arrays and 'shared' with the Dual Port RAM. These variables can be addressed directly by the VME host. The variables are stored in the Variable Buffer at 240 - 3BF for the DMC 1310/1340 and at 440 - 5BF. Variables are assigned by VR[n] = value where n is a number in the range 0 to 63 and the value is 4 bytes of integer followed by two bytes of fraction.

Examples - Valid Variable Names

POSX

POS1

SPEEDZ

Examples - Invalid Variable Names

Variable	Problem
REALLONGNAME	Cannot have more than 8 characters
124	Cannot begin variable name with a number
SPEED Z	Cannot have spaces in the name

Assigning Values to Variables:

Assigned values can be numbers, internal variables and keywords, functions, controller parameters and strings;

Variables hold 6 bytes of data, 4 bytes of integer (2^{31})followed by two bytes of fraction providing a range of values of +/-2,147,483,647.9999.

Numeric values can be assigned to programmable variables using the equal sign.

Any valid DMC 1300 function can be used to assign a value to a variable. For example, V1=@ABS[V2] or V2=@IN[1]. Arithmetic operations are also permitted.

To assign a string value, the string must be in quotations. String variables can contain up to six characters which must be in quotations.

Variable values may be assigned to controller parameters such as PR or SP.

When using the shared Dual Port RAM variables, values are assigned using the VR[n]= value command.

Examples - Assigning values to variables

Instruction	Interpretation
POSX=_TPX	Assigns returned value from TPX command to variable POSX.
SPEED=5.75	Assigns value 5.75 to variable SPEED
INPUT=@IN[2]	Assigns logical value of input 2 to variable INPUT
V2=V1+V3*V4	Assigns the value of V1 plus V3 times V4 to the variable V2.
VAR="CAT"	Assign the string, CAT, to VAR
PR V1	Assign value of variable V1 to PR command for X axis
SP VS*2000	Assign VS*2000 to SP command

Examples - Dual Port RAM assigned variables

Instruction	Interpretation
VR[22]=200	Assigns the decimal value 200 to variable element number 22. On a DMC 1340, this element is found at address $2CF, with the data 00 00 00 C8 00 00.

Displaying the value of variables at the terminal

Variables may be sent to the screen using the format, variable=. For example, V1= , returns the value of the variable V1.

Example - Using Variables for Joystick

The example below reads the voltage of an X-Y joystick and assigns it to variables VX and VY to drive the motors at proportional velocities, where

10 Volts = 3000 rpm = 200000 c/sec

Speed/Analog input = 200000/10 = 20000

Instruction	Interpretation

#JOYSTIK	Label
JG 0,0	Set in Jog mode
BGXY	Begin Motion
#LOOP	Loop
VX=@AN[1]*20000	Read joystick X
VY=@AN[2]*20000	Read joystick Y
JG VX,VY	Jog at variable VX,VY
JP#LOOP	Repeat
EN	End

Operands

Operands allow motion or status parameters of the DMC 1300 to be incorporated into programmable variables and expressions. An operand contains data and must be used in a valid expression or function. Most DMC 1300 commands have an equivalent operand - which are designated by adding an underscore (_) prior to the DMC 1300 command. Commands which have an associated operand are listed in the Command Reference as "Used as an Operand" .. Yes.

Status commands such as Tell Position return actual values, whereas action commands such as GN or SP return the values in the DMC 1300 registers. The axis designation is required following the command.

Examples of operand usage

POSX=_TPX	Assigns value from Tell Position X to the variable POSX.
GAIN=_GNZ*2	Assigns value from GNZ multiplied by two to variable, GAIN.
JP #LOOP,_TEX>5	Jump to #LOOP if the position error of X is greater than 5
JP #ERROR,_TC=1	Jump to #ERROR if the error code equals 1.

Operands can be used in an expression and assigned to a programmable variable, but they cannot be assigned a value. For example: _GNX=2 is invalid.

The value of an operand can be output to the computer with the message command, MG. IE. MG _TEX sends the current position error value on axis X to the computer.

Special Operands (Keywords)

The DMC 1300 provides a few operands which give access to internal variables that are not accessible by standard DMC 1300 commands.

KEYWORD	FUNCTION
_BGn	*Is equal to a 1 if motion on axis 'n' is complete, otherwise equal to 0.
_DA	*Is equal to the number of arrays available
_DL	*Is equal to the number of available labels for programming
_DM	*Is equal to the available array memory
_HMn	*Is equal to status of Home Switch (equals 0 or 1)
_LFn	Is equal to status of Forward Limit switch input of axis 'n' (equals 0 or 1)
_LRX	Is equal to status of Reverse Limit switch input of axis 'n' (equals 0 or 1)
_UL	*Is equal to the number of available variables
TIME	Free-Running Real Time Clock (off by 2.4% - Resets with power-on). Note: TIME does not use an underscore character (_) as other keywords.

* - These keywords have corresponding commands while the keywords _LF, _LR, and TIME do not have any associated commands. All keywords are listed in the Command Summary, Chapter 11.

Examples of Keywords

Instruction	Interpretation
V1=_LFX	Assign V1 the logical state of the Forward Limit Switch on the X-axis
V3=TIME	Assign V3 the current value of the time clock
V4=_HMW	Assign V4 the logical state of the Home input on the W-axis

Arrays

For storing and collecting numerical data, the DMC 1300 provides array space for 1600 elements or 8000 elements for controllers with 5 or more axes, or with controller with the -MX option. The arrays are one dimensional and up to 14 different arrays may be defined (30 for controllers with 5 or more axes, or the -MX option). Each array element has a numeric range of 4 bytes of integer (2^{31})followed by two bytes of fraction (+/-2,147,483,647.9999).

Arrays can be used to capture real-time data, such as position, torque and analog input values. In the contouring mode, arrays are convenient for holding the points of a position trajectory in a record and playback application.

Defining Arrays

An array is defined with the command DM. The user must specify a name and the number of entries to be held in the array. An array name can contain up to eight characters, starting with an uppercase alphabetic character. The number of entries in the defined array is enclosed in [ ].

Example - USING THE COMMAND, DM

Instruction	Interpretation
DM POSX[7]	Defines an array names POSX with seven entries
DM SPEED[100]	Defines an array named speed with 100 entries
DM POSX[0]	Frees array space

Assignment of Array Entries

Like variables, each array element can be assigned a value. Assigned values can be numbers or returned values from instructions, functions and keywords.

Array elements are addressed starting at count 0. For example the first element in the POSX array (defined with the DM command, DM POSX[7]) would be specified as POSX[0].

Values are assigned to array entries using the equal sign. Assignments are made one element at a time by specifying the element number with the associated array name.

NOTE: Arrays must be defined using the command, DM, before assigning entry values.

Examples - assigning values to array entries

Instruction	Interpretation
DM SPEED[10]	Dimension Speed Array
SPEED[1]=7650.2	Assigns the first element of the array, SPEED the value 7650.2
SPEED[1]=	Returns array element value
POSX[10]=_TPX	Assigns the 10th element of the array POSX the returned value from the tell position command.
CON[2]=@COS[POS]*2	Assigns the second element of the array CON the cosine of the variable POS multiplied by 2.
TIMER[1]=TIME	Assigns the first element of the array timer the returned value of the TIME keyword.

Using a Variable to Address Array Elements

An array element number can also be a variable. This allows array entries to be assigned sequentially using a counter. For example;

Instruction	Interpretation
#A	Begin Program
COUNT=0;DM POS[10]	Initialize counter and define array
#LOOP	Begin loop
WT 10	Wait 10 msec
POS[COUNT]=_TPX	Record position into array element
POS[COUNT]=	Report position
COUNT=COUNT+1	Increment counter
JP #LOOP,COUNT<10	Loop until 10 elements have been stored
EN	End Program

The above example records 10 position values at a rate of one value per 10 msec. The values are stored in an array named POS. The variable, COUNT, is used to increment the array element counter. The above example can also be executed with the automatic data capture feature described below.

Automatic Data Capture into Arrays

The DMC 1300 provides a special feature for automatic capture of data such as position, position error, inputs or torque. This is useful for teaching motion trajectories or observing system performance. Up to four types of data can be captured and stored in four arrays. For controllers with 5 or more axes, up to eight types of data can be captured and stored in eight arrays. The capture rate or time interval may be specified. Recording can done as a one time event or as a circular continuous recording.

Command Summary - Automatic Data Capture

COMMAND	DESCRIPTION
RA n[],m[],o[],p[]	Selects up to four arrays (eight arrays for DMC-1080) for data capture. The arrays must be defined with the DM command.
RD type1,type2,type3,type4	Selects the type of data to be recorded, where type1, type2, type3, and type 4 represent the various types of data (see table below). The order of data type is important and corresponds with the order of n,m,o,p arrays in the RA command.
RC n,m	The RC command begins data collection. Sets data capture time interval where n is an integer between 1 and 8 and designates 2^n msec between data. m is optional and specifies the number of elements to be captured. If m is not defined, the number of elements defaults to the smallest array defined by DM. When m is a negative number, the recording is done continuously in a circular manner. _RD is the recording pointer and indicates the address of the next array element. n=0 stops recording.
RC?	Returns a 0 or 1 where, 0 denotes not recording, 1 specifies recording in progress

Data Types for Recording:

DATA TYPE	DESCRIPTION
_DEX	2nd encoder position (dual encoder)
_TPX	Encoder position
_TEX	Position error
_SHX	Commanded position
_RLX	Latched position
_TI	Inputs
_OP	Output
_TSX	Switches (only bit 0-4 valid)
_SCX	Stop code
_NOX	Status bits
_TTX	Torque (reports digital value +/-8097)

Note: X may be replaced by Y,Z or W for capturing data on other axes, or A,B,C,D,E,F,G,H for DMC 1380.

Operand Summary - Automatic Data Capture

_RC	Returns a 0 or 1 where, 0 denotes not recording, 1 specifies recording in progress
_RD	Returns address of next array element.

Example - Recording into An Array

During a position move, store the X and Y positions and position error every 2 msec.

Instruction **Interpretation**

#RECORD	Begin program
DM XPOS[300],YPOS[300]	Define X,Y position arrays
DM XERR[300],YERR[300]	Define X,Y error arrays
RA XPOS[],XERR[],YPOS[],YERR[]	Select arrays for capture
RD _TPX,_TEX,_TPY,_TEY	Select data types
PR 10000,20000	Specify move distance
RC1	Start recording now, at rate of 2 msec
BG XY	Begin motion
#A;JP #A,RC=1	Loop until done
MG "DONE"	Print message
EN	End program
#PLAY	Play back
N=0	Initial Counter
JP# DONE,N>300	Exit if done
N=	Print Counter
X POS[N]=	Print X position
Y POS[N]=	Print Y position
XERR[N]=	Print X error
YERR[N]=	Print Y error
N=N+1	Increment Counter
#DONE	Done

Deallocating Array Space

Array space may be deallocated using the DA command followed by the array name. DA*[0] deallocates all the arrays.

Output of Data (Numeric and String)

Numerical and string data can be output from the controller using several methods. The message command, MG, can output string and numerical data. Also, the controller can be commanded to return the values of variables and arrays, as well as other information using the interrogation commands (the interrogation commands are described in chapter 5).

Sending Messages

Messages may be sent to the bus using the message command, MG. This command sends specified text and numerical or string data from variables or arrays to the screen.

Text strings are specified in quotes and variable or array data is designated by the name of the variable or array. For example:

```
MG "The Final Value is", RESULT
```

In addition to variables, functions and commands, responses can be used in the message command. For example:

MG "Analog input is", @AN[1]

MG "The Gain of X is", _GNX

The response from the message command when sent through the Command Buffer is found in the Response Buffer. The response from the message command when sent through an application program is found in the Program Buffer. See the MG command in Chapter 12 for more details.

Programmable Hardware I/O

Digital Outputs

The DMC 1300 has an 8-bit uncommitted output port for controlling external events. The DMC-1080 has an additional eight output bits available at JD5 pins 10-17. Each bit on the output port may be set and cleared with the software instructions SB (Set Bit) and CB(Clear Bit), or OB (define output bit).

The outputs may also be set and read through the Dual Port RAM.

Example - Using Set Bit and Clear Bit Commands (SB, CB)

Instruction	Interpretation
SB6	Sets bit 6 of output port
CB4	Clears bit 4 of output port
CB9	Clear bit 9 of output port on DMC-1380

The Output Bit (OB) instruction is useful for setting or clearing outputs depending on the value of a variable, array, input or expression. Any non-zero value results in a set bit.

Example - Using the output bit Command (OB)

Instruction	Interpretation
OB1, POS	Set Output 1 if the variable POS is non-zero. Clear Output 1 if POS equals 0.
OB 2, @IN [1]	Set Output 2 if Input 1 is high. If Input 1 is low, clear Output 2.
OB 3, @IN [1]&@IN [2]	Set Output 3 only if Input 1 and Input 2 are high.
OB 4, COUNT [1]	Set Output 4 if element 1 in the array COUNT is non-zero.

The output port can be set by specifying an 8-bit word using the instruction OP (Output Port). This instruction allows a single command to define the state of the entire 8-bit output port, where 2^0 is output 1, 2^1 is output 2 and so on. A 1 designates that the output is on.

Example - Using the output PORT Command (op)

Instruction	Interpretation
OP6	Sets outputs 2 and 3 of output port to high. All other bits are 0. ($2^1 + 2^2 = 6$)
OP0	Clears all bits of output port to zero

OP 255	Sets all bits of output port to one. ($2^2 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7$)

Example - Using OP to turn on output after move

Instruction	Interpretation
#OUTPUT	Label
PR 2000	Position Command
BG	Begin
AM	After move
SB1	Set Output 1
WT 1000	Wait 1000 msec
CB1	Clear Output 1
EN	End

Digital Inputs

The DMC 1300 has eight digital inputs for controlling motion by local switches. The @IN[n] function returns the logic level of the specified input 1 through 8. For example, a Jump on Condition instruction can be used to execute a sequence if a high condition is noted on an input 3. To halt program execution, the After Input (AI) instruction waits until the specified input has occurred.

Digital inputs on the DMC 1300 may also be read through the Dual Port RAM.

Example - Using the AI command:

Instruction	Interpretation
JP #A,@IN[1]=0	Jump to A if input 1 is low
JP #B,@IN[2]=1	Jump to B if input 2 is high
AI 7	Wait until input 7 is high
AI -6	Wait until input 6 is low

Example - Start Motion on Switch

Motor X must turn at 4000 counts/sec when the user flips a panel switch to on. When panel switch is turned to off position, motor X must stop turning.

Solution: Connect panel switch to input 1 of DMC 1300. High on input 1 means switch is in on position.

Instruction	Interpretation
#S;JG 4000	Set speed
AI 1;BGX	Begin after input 1 goes high
AI -1;STX	Stop after input 1 goes low
AMX;JP #S	After motion, repeat
EN;	

Input Interrupt Function

The DMC 1300 provides an input interrupt function which causes the program to automatically execute the instructions following the #ININT label. This function is enabled using the II m,n,o command. The m specifies the beginning input and n specifies the final input in the range. The parameter o is an interrupt mask. If m and n are unused, o contains a number with the mask. A 1 designates that input to be enabled for an interrupt, where 2^0 is bit 1, 2^1 is bit 2 and so on. For example, II,,5 enables inputs 1 and 3 ($2^0 + 2^2 = 5$).

A low input on any of the specified inputs will cause automatic execution of the #ININT subroutine. The Return from Interrupt (RI) command is used to return from this subroutine to the place in the program where the interrupt had occurred. If it is desired to return to somewhere else in the program after the execution of the #ININT subroutine, the Zero Stack (ZS) command is used followed by unconditional jump statements.

IMPORTANT: Use the RI instruction (not EN) to return from the #ININT subroutine.

Examples - Input Interrupt

Instruction	Interpretation
#A	Label #A
II 1	Enable input 1 for interrupt function
JG 30000,-20000	Set speeds on X and Y axes
BG XY	Begin motion on X and Y axes
#B	Label #B
TP XY	Report X and Y axes positions
WT 1000	Wait 1000 milliseconds
JP #B	Jump to #B
EN	End of program
#ININT	Interrupt subroutine
MG "Interrupt occurred"	Display message
ST XY	Stops motion on X and Y axes
#LOOP;JP #LOOP,@IN[1]=0	Loop until Interrupt cleared
JG 15000,10000	Specify new speeds
WT 300	Wait 300 milliseconds
BG XY	Begin motion on X and Y axes
RI	Return from Interrupt subroutine

Analog Inputs

The DMC 1300 provides seven analog inputs. The value of these inputs in volts may be read using the @AN[n] function where n is the analog input 1 through 7. The resolution of the Analog-to-Digital conversion is 12 bits. Analog inputs are useful for reading special sensors such as temperature, tension or pressure.

The following examples show programs which cause the motor to follow an analog signal. The first example is a point-to-point move. The second example shows a continuous move.

Example - Position Follower (Point-to-Point)

Objective - The motor must follow an analog signal. When the analog signal varies by 10V, motor must move 10000 counts.

Method: Read the analog input and command X to move to that point.

Instruction	Interpretation
#Points	Label
SP 7000	Speed
AC 80000;DC 80000	Acceleration
#Loop	
VP=@AN[1]*1000	Read and analog input, compute position
PA VP	Command position
BGX	Start motion
AMX	After completion
JP #Loop	Repeat
EN	End

Example - Position Follower (Continuous Move)

Method: Read the analog input, compute the commanded position and the position error. Command the motor to run at a speed in proportions to the position error.

Instruction	Interpretation
#Cont	Label
AC 80000;DC 80000	Acceleration rate
JG 0	Start job mode
BGX	Start motion
#Loop	
VP=@AN[1]*1000	Compute desired position
VE=VP-_TPX	Find position error
VEL=VE*20	Compute velocity
JG VEL	Change velocity
JP #Loop	Change velocity
EN	End

Example Applications

Wire Cutter

An operator activates a start switch. This causes a motor to advance the wire a distance of 10". When the motion stops, the controller generates an output signal which activates the cutter. Allowing 100 ms for the cutting completes the cycle.

Suppose that the motor drives the wire by a roller with a 2" diameter. Also assume that the encoder resolution is 1000 lines per revolution. Since the circumference of the roller equals 2π inches, and it corresponds to 4000 quadrature, one inch of travel equals:

$4000/2\pi = 637$ count/inch

This implies that a distance of 10 inches equals 6370 counts, and a slew speed of 5 inches per second, for example, equals 3185 count/sec.

The input signal may be applied to I1, for example, and the output signal is chosen as output 1. The motor velocity profile and the related input and output signals are shown in Fig. 7.1.

The program starts at a state that we define as #A. Here the controller waits for the input pulse on I1. As soon as the pulse is given, the controller starts the forward motion.

Upon completion of the forward move, the controller outputs a pulse for 20 ms and then waits an additional 80 ms before returning to #A for a new cycle.

Instruction	**Function**
#A	Label
AI1	Wait for input 1
PR 6370	Distance
SP 3185	Speed
BGX	Start Motion
AMX	After motion is complete
SB1	Set output bit 1
WT 20	Wait 20 ms
CB1	Clear output bit 1
WT 80	Wait 80 ms
JP #A	Repeat the process

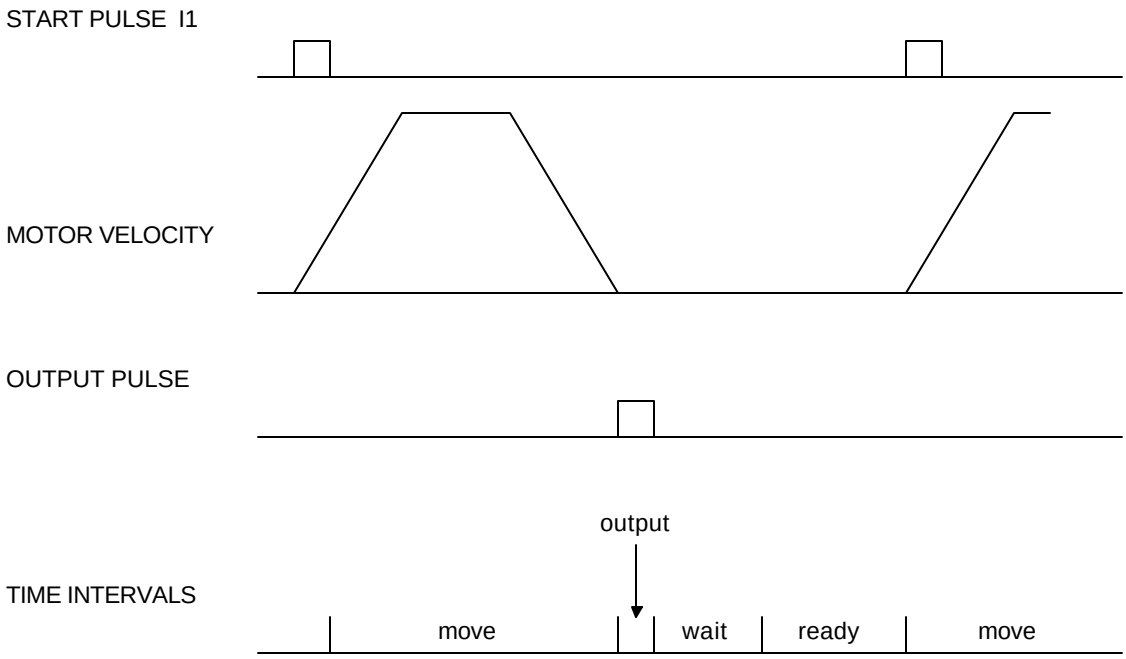


Figure 7.1 - Motor Velocity and the Associated input/output signals

X-Y Table Controller

An X-Y-Z system must cut the pattern shown in Fig. 7.2. The X-Y table moves the plate while the Z-axis raises and lowers the cutting tool.

The solid curves in Fig. 7.2 indicate sections where cutting takes place. Those must be performed at a feedrate of 1 inch per second. The dashed line corresponds to non-cutting moves and should be performed at 5 inch per second. The acceleration rate is 0.1 g.

The motion starts at point A, with the Z-axis raised. An X-Y motion to point B is followed by lowering the Z-axis and performing a cut along the circle. Once the circular motion is completed, the Z-axis is raised and the motion continues to point C, etc.

Assume that all of the 3 axes are driven by lead screws with 10 turns-per-inch pitch. Also assume encoder resolution of 1000 lines per revolution. This results in the relationship:

1 inch = 40,000 counts

and the speeds of

1 in/sec = 40,000 count/sec

5 in/sec = 200,000 count/sec

an acceleration rate of 0.1g equals

$0.1g = 38.6 \text{ in/s2} = 1{,}544{,}000 \text{ count/s}^2$

Note that the circular path has a radius of 2" or 80000 counts, and the motion starts at the angle of 270° and traverses 360° in the CW (negative direction). Such a path is specified with the instruction

CR 80000,270,-360

Further assume that the Z must move 2" at a linear speed of 2" per second. The required motion is performed by the following instructions:

Instruction	Interpretation
#A	Label
VM XY	Circular interpolation for XY
VP 160000,160000	Positions
VE	End Vector Motion
VS 200000	Vector Speed
VA 1544000	Vector Acceleration
BGS	Start Motion
AMS	When motion is complete
PR,,-80000	Move Z down
SP,,80000	Z speed
BGZ	Start Z motion
AMZ	Wait for completion of Z motion
CR 80000,270,-360	Circle
VE	
VS 40000	Feedrate

```
BGS                 Start circular move
AMS                 Wait for completion
PR,,80000           Move Z up
BGZ                 Start Z move
AMZ                 Wait for Z completion
PR -21600           Move X
SP 20000            Speed X
BGX                 Start X
AMX                 Wait for X completion
PR,,-80000          Lower Z
BGZ
AMZ
CR 80000,270,-360   Z second circle move
VE
VS 40000
BGS
AMS
PR,,80000           Raise Z
BGZ
AMZ
VP -37600,-16000    Return XY to start
VE
VS 200000
BGS
AMS
EN
```

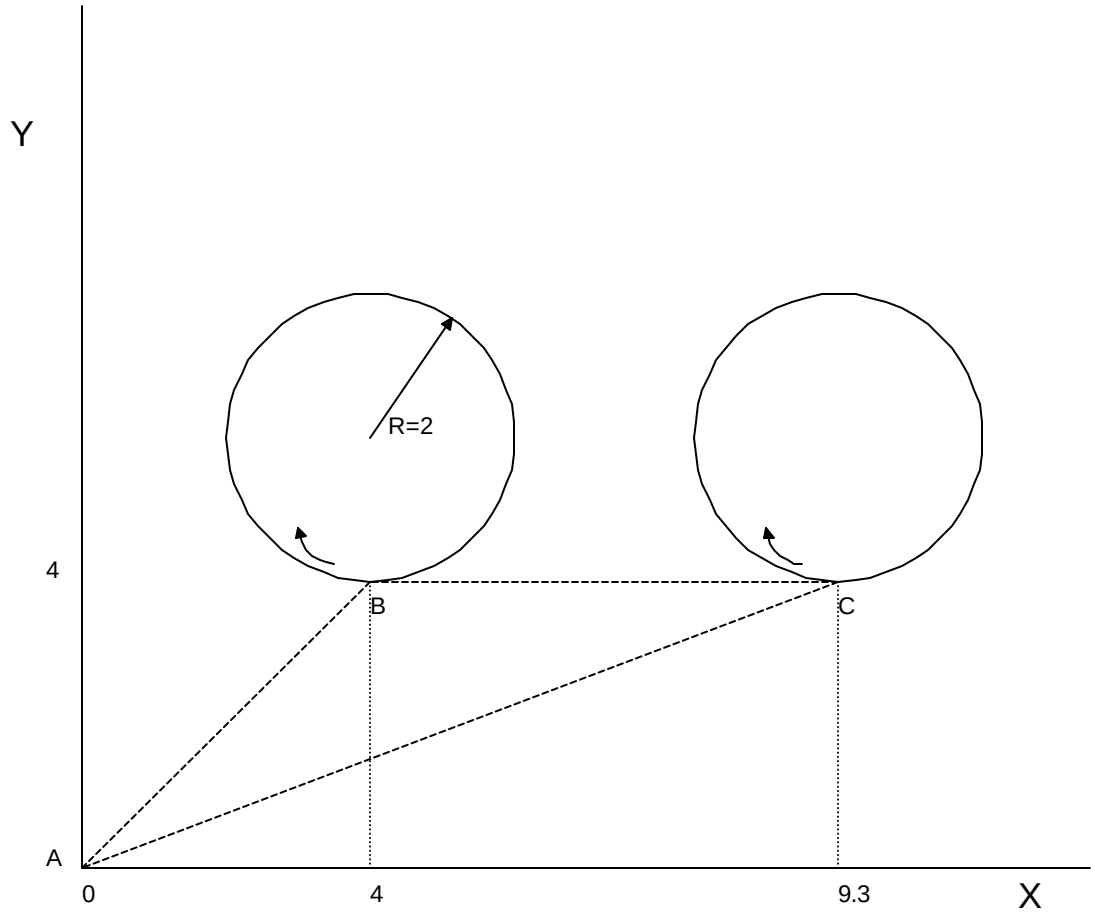


Figure 7.2 - Motor Velocity and the Associated input/output signals

Speed Control by Joystick

The speed of a motor is controlled by a joystick. The joystick produces a signal in the range between -10V and +10V. The objective is to drive the motor at a speed proportional to the input voltage.

Assume that a full voltage of 10 Volts must produce a motor speed of 3000 rpm with an encoder resolution of 1000 lines or 4000 count/rev. This speed equals:

3000 rpm = 50 rev/sec = 200000 count/sec

The program reads the input voltage periodically and assigns its value to the variable VIN. To get a speed of 200,000 ct/sec for 10 volts, we select the speed as

Speed = 20000 x VIN

The corresponding velocity for the motor is assigned to the VEL variable.

Instruction	Interpretation
#A	Label
JG0	Set jog speed of zero
BGX	Begin jogging (at speed zero)
#B	Label
VIN=@AN[1]	Set variable, VIN, to value of analog input 1

VEL=VIN*20000	Set variable, VEL to multiple of variable of VIN
JG VEL	Update jog speed to value of variable VEL
JP #B	Loop back to label, #B
EN	End

Position Control by Joystick

This system requires the position of the motor to be proportional to the joystick angle. Furthermore, the ratio between the two positions must be programmable. For example, if the control ratio is 5:1, it implies that when the joystick voltage is 5 Volts, corresponding to 1028 counts, the required motor position must be 5120 counts. The variable V3 changes the position ratio.

Instruction	Interpretation
#A	Label
V3=5	Initial position ratio
DP0	Define the starting position
JG0	Set motor in jog mode as zero
BGX	Start
#B	
V1=@AN[1]	Read analog input
V2=V1*V3	Compute the desired position
V4=V2-_TPX-_TEX	Find the following error
V5=V4*20	Compute a proportional speed
JG V5	Change the speed
JP #B	Repeat the process
EN	End

Backlash Compensation by Sampled Dual-Loop

The continuous dual loop, enabled by the DV1 function is an effective way to compensate for backlash. In some cases, however, when the backlash magnitude is large, it may be difficult to stabilize the system. In those cases, it may be easier to use the sampled dual loop method described below.

This design example addresses the basic problems of backlash in motion control systems. The objective is to control the position of a linear slide precisely. The slide is to be controlled by a rotary motor, which is coupled to the slide by a leadscrew. Such a leadscrew has a backlash of 4 micron, and the required position accuracy is for 0.5 micron.

The basic dilemma is where to mount the sensor. If you use a rotary sensor, you get a 4 micron backlash error. On the other hand, if you use a linear encoder, the backlash in the feedback loop will cause oscillations due to instability.

An alternative approach is the dual-loop, where we use two sensors, rotary and linear. The rotary sensor assures stability (because the position loop is closed before the backlash) whereas the linear sensor provides accurate load position information. The operation principle is to drive the motor to a given rotary position near the final point. Once there, the load position is read to find the position error and the controller commands the motor to move to a new rotary position which eliminates the position error.

Since the required accuracy is 0.5 micron, the resolution of the linear sensor should preferably be twice finer. A linear sensor with a resolution of 0.25 micron allows a position error of +/-2 counts.

The dual-loop approach requires the resolution of the rotary sensor to be equal or better than that of the linear system. Assuming that the pitch of the lead screw is 2.5mm (approximately 10 turns per inch), a rotary encoder of 2500 lines per turn or 10,000 count per revolution results in a rotary resolution of 0.25 micron. This results in equal resolution on both linear and rotary sensors.

To illustrate the control method, assume that the rotary encoder is used as a feedback for the X-axis, and that the linear sensor is read and stored in the variable LINPOS. Further assume that at the start, both the position of X and the value of LINPOS are equal to zero. Now assume that the objective is to move the linear load to the position of 1000.

The first step is to command the X motor to move to the rotary position of 1000. Once it arrives we check the position of the load. If, for example, the load position is 980 counts, it implies that a correction of 20 counts must be made. However, when the X-axis is commanded to be at the position of 1000, suppose that the actual position is only 995, implying that X has a position error of 5 counts, which will be eliminated once the motor settles. This implies that the correction needs to be only 15 counts, since 5 counts out of the 20 would be corrected by the X-axis. Accordingly, the motion correction should be:

Correction = Load Position Error - Rotary Position Error

The correction can be performed a few times until the error drops below +/-2 counts. Often, this is performed in one correction cycle.

Example - backlash compensation by sampled dual loop

Instruction	Interpretation
#A	Label
DP0	Define starting positions as zero
LINPOS=0	
PR 1000	Required distance
BGX	Start motion
#B	
AMX	Wait for completion
WT 50	Wait 50 msec
LIN POS = _DEX	Read linear position
ER=1000-LINPOS-_TEX	Find the correction
JP #C,@ABS[ER]<2	Exit if error is small
PR ER	Command correction
BGX	Begin motion on X axis
JP #B	Repeat the process
#C	Label
EN	End program

THIS PAGE LEFT BLANK INTENTIONALLY

Chapter 8 Hardware & Software Protection

Introduction

The DMC 1300 provides several hardware and software features to check for error conditions and to inhibit the motor on error. These features help protect the various system components from damage.

WARNING: Machinery in motion can be dangerous! It is the responsibility of the user to design effective error handling and safety protection as part of the machine. Since the DMC 1300 is an integral part of the machine, the engineer should design his overall system with protection against a possible component failure on the DMC 1300. Galil shall not be liable or responsible for any incidental or consequential damages.

Hardware Protection

The DMC 1300 includes hardware input and output protection lines for various error and mechanical limit conditions. These include:

Output Protection Lines

Amp Enable - This signal goes low when the motor off command is given, when the position error exceeds the value specified by the Error Limit (ER) command, or when off-on-error condition is enabled (OE1) and the abort command is given. Each axis amplifier has separate amplifier enable lines. This signal also goes low when the watch-dog timer is activated, or upon reset. *Note: The standard configuration of the AEN signal is TTL active low. Both the polarity and the amplitude can be changed if you are using the ICM-1100 interface board. To make these changes, see section entitled 'Amplifier Interface' pg. 3-25.*

Input Protection Lines

Abort - A low input stops commanded motion instantly without a controlled deceleration. For any axis in which the Off-On-Error function is enabled, the amplifiers will be disabled. This could cause the motor to 'coast' to a stop. If the Off-On-Error function is not enabled, the motor will instantaneously stop and servo at the current position. The Off-On-Error function is further discussed in this chapter.

Forward Limit Switch - Low input inhibits motion in forward direction. If the motor is moving in the forward direction when the limit switch is activated, the motion will decelerate and stop. In addition, if the motor is moving in the forward direction, the controller will automatically jump to the limit switch subroutine, #LIMSWI (if such a routine has been written by the user). The CN command can be used to change the polarity of the limit switches.

Reverse Limit Switch - Low input inhibits motion in reverse direction. If the motor is moving in the reverse direction when the limit switch is activated, the motion will decelerate and stop. In addition, if the motor is moving in the reverse direction, the controller will automatically jump to the limit switch subroutine, #LIMSWI (if such a routine has been written by the user). The CN command can be used to change the polarity of the limit switches.

Software Protection

The DMC 1300 provides a programmable error limit for servo operation. The error limit can be set for any number between 1 and 32767 using the ER n command. The default value for ER is 16384.

Example:

ER 200,300,400,500	Set X-axis error limit for 200, Y-axis error limit to 300, Z-axis error limit to 400 counts, W-axis error limit to 500 counts
ER,1,,10	Set Y-axis error limit to 1 count, set W-axis error limit to 10 counts.

The units of the error limit are quadrature counts. The error is the difference between the command position and actual encoder position. If the absolute value of the error exceeds the value specified by ER, the DMC 1300 will generate several signals to warn the host system of the error condition. These signals include:

Signal or Function	Indication of Error
# POSERR	Jumps to automatic excess position error subroutine
Error Light	Turns on when position error exceeds error limit
OE Function	Shuts motor off by setting AEN output line low if OE1.

The position error of X,Y,Z and W can be monitored during execution using the TE command.

Programmable Position Limits

The DMC 1300 provides programmable forward and reverse position limits. These are set by the BL and FL software commands. Once a position limit is specified, the DMC 1300 will not accept position commands beyond the limit. Motion beyond the limit is also prevented.

Example - Using position limits

Instruction	Interpretation
DP0,0,0	Define Position
BL -2000,-4000,-8000	Set Reverse position limit
FL 2000,4000,8000	Set Forward position limit
JG 2000,2000,2000	Jog
BG XYZ	Begin

(motion stops at forward limits)

Off-On-Error

The DMC 1300 controller has a built in function which can turn off the motors under certain error conditions. This function is know as 'Off-On-Error". To activate the OE function for each axis, specify 1 for X,Y,Z and W axis. To disable this function, specify 0 for the axes. When this function is enabled, the specified motor will be disabled under the following 3 conditions:

1. The position error for the specified axis exceeds the limit set with the command, ER
2. The abort command is given
3. The abort input is activated with a low signal.

The status of the OE command is read through the Dual Port RAM at Bit 1 of Status #2 in the Axis Buffers of the DMC 1300.

Note: If the motors are disabled while they are moving, they may 'coast' to a stop because they are no longer under servo control.

To re-enable the system, use the Reset (RS) or Servo Here (SH) command.

Examples - Using Off-On-Error

OE 1,1,1,1	Enable off-on-error for X,Y,Z and W
OE 0,1,0,1	Enable off-on-error for Y and W axes and disable off-on-error for W and Z axes

Automatic Error Routine

The #POSERR label causes the statements following to be automatically executed if error on any axis exceeds the error limit specified by ER. The error routine must be closed with the RE command. The RE command returns from the error subroutine to the main program.

NOTE: The Error Subroutine will be entered again unless the error condition is gone.

Example - using automatic error subroutine

Instruction	Interpretation
#A;JP #A;EN	"Dummy" program
#POSERR	Start error routine on error
MG "error"	Send message
SB 1	Fire relay
STX	Stop motor
AMX	After motor stops
SHX	Servo motor here to clear error
RE	Return to main program

NOTE: An applications program must be executing for the #POSERR routine to function.

Limit Switch Routine

The DMC 1300 provides forward and reverse limit switches which inhibit motion in the respective direction. There is also a special label for automatic execution of a limit switch subroutine. The

#LIMSWI label specifies the start of the limit switch subroutine. This label causes the statements following to be automatically executed if any limit switch is activated and that axis motor is moving in that direction. The RE command ends the subroutine.

The state of the forward and reverse limit switches may also be tested during the jump -on-condition statement. The _LR condition specifies the reverse limit and _LF specifies the forward limit. X,Y,Z, or W following LR or LF specifies the axis. The CN command can be used to configure the polarity of the limit switches.

Example - using Limit Switch subroutine

Instruction	Interpretation
#A;JP #A;EN	Dummy Program
#LIMSWI	Limit Switch Utility
V1=_LFX	Check if forward limit
V2=_LRX	Check if reverse limit
JP#LF,V1=0	Jump to #LF if forward
JP#LR,V2=0	Jump to #LR if reverse
JP#END	Jump to end
#LF	#LF
MG "FORWARD LIMIT"	Send message
STX;AMX	Stop motion
PR-1000;BGX;AMX	Move in reverse
JP#END	End
#LR	#LR
MG "REVERSE LIMIT"	Send message
STX;AMX	Stop motion
PR1000;BGX;AMX	Move forward
#END	End
RE	Return to main program

NOTE: An applications program must be executing for #LIMSWI to function.

Chapter 9 Troubleshooting

Overview

The following discussion may help you get your system to work.

Potential problems have been divided into groups as follows:

1. Installation
2. Communication
3. Stability and Compensation
4. Operation

The various symptoms along with the cause and the remedy are described in the following tables.

Installation

SYMPTOM	CAUSE	REMEDY
Motor runs away when connected to amplifier with no additional inputs.	Amplifier offset too large.	Adjust amplifier offset
Same as above, but offset adjustment does not stop the motor.	Damaged amplifier.	Replace amplifier.
Same as above, but offset adjustment does not stop the motor.	Damaged amplifier.	Replace amplifier.
Controller does not read changes in encoder position.	Wrong encoder connections.	Check encoder wiring.
Same as above	Bad encoder	Check the encoder signals. Replace encoder if necessary.
Same as above	Bad controller	Connect the encoder to different axis input. If it works, controller failure. Repair or replace.

Communication

SYMPTOM	CAUSE	REMEDY
No communication with host system.	Address selection in communication does not match jumpers.	Check address jumper positions, and change if necessary.

Stability

SYMPTOM	CAUSE	REMEDY
Motor runs away when the loop is closed.	Wrong feedback polarity.	Invert the polarity of the loop by inverting the motor leads (brush type) or the encoder.
Motor oscillates.	Too high gain or too little damping.	Decrease KI and KP. Increase KD.

Operation

SYMPTOM	CAUSE	REMEDY
Controller rejects command. Responded with a ?	Invalid Command	Interrogate the cause with TC or TC1.
Motor does not complete move.	Noise on limit switches stops the motor.	To verify cause, check the stop code (SC). If caused by limit switch noise, reduce noise.
During a periodic operation, motor drifts slowly.	Encoder noise	Interrogate the position periodically. If controller states that the position is the same at different locations it implies encoder noise. Reduce noise. Use differential encoder inputs.
Same as above.	Programming error.	Avoid resetting position error at end of move with SH command.

Chapter 10 Theory of Operation

Overview

The following discussion covers the operation of motion control systems. A typical servo control system consists of the elements shown in Fig 10.1.

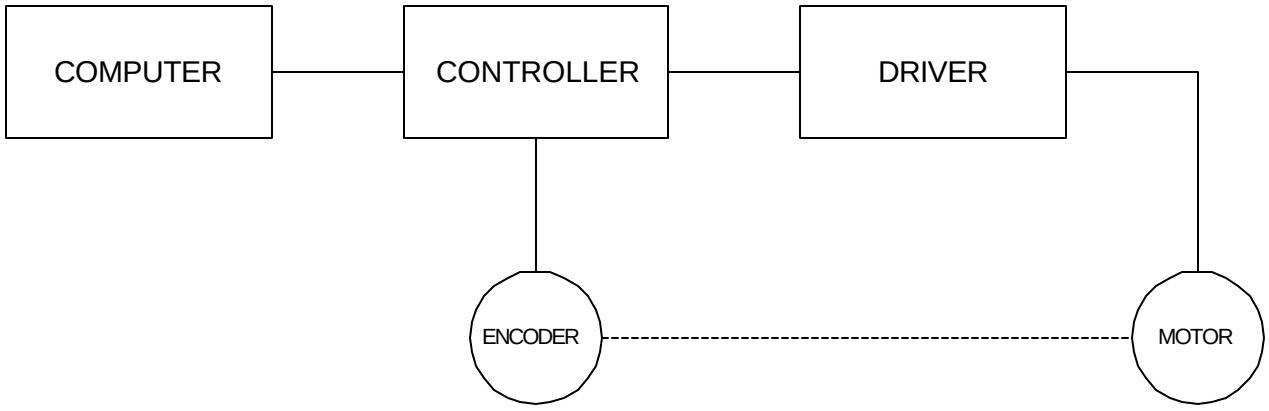


Figure 10.1 - Elements of Servo Systems

The operation of such a system can be divided into three levels, as illustrated in Fig. 10.2. The levels are:

1. Closing the Loop
2. Motion Profiling
3. Motion Programming

The first level, the closing of the loop, assures that the motor follows the commanded position. This is done by closing the position loop using a sensor. The operation at the basic level of closing the loop involves the subjects of modeling, analysis, and design. These subjects will be covered in the following discussions.

The motion profiling is the generation of the desired position function. This function, R(t), describes where the motor should be at every sampling period. Note that the profiling and the closing of the loop are independent functions. The profiling function determines where the motor should be and the closing of the loop forces the motor to follow the commanded position

The highest level of control is the motion program. This can be stored in the host computer or in the controller. This program describes the tasks in terms of the motors that need to be controlled, the distances and the speed.

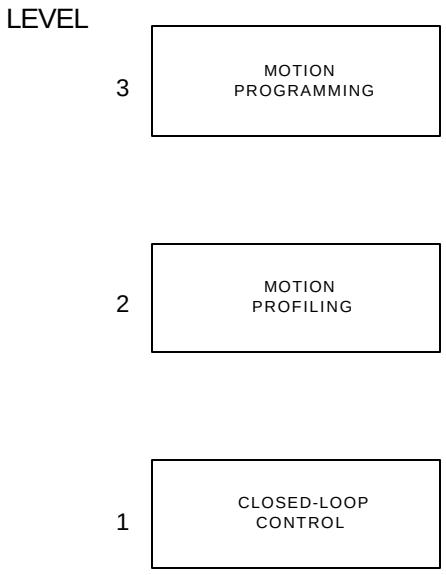


Figure 10.2 - Levels of Control Functions

The three levels of control may be viewed as different levels of management. The top manager, the motion program, may specify the following instruction, for example.

```
PR 6000,4000
SP 20000,20000
AC 200000,00000
BG X
AD 2000
BG Y
EN
```

This program corresponds to the velocity profiles shown in Fig. 10.3. Note that the profiled positions show where the motors must be at any instant of time.

Finally, it remains up to the servo system to verify that the motor follows the profiled position by closing the servo loop.

The following section explains the operation of the servo system. First, it is explained qualitatively, and then the explanation is repeated using analytical tools for those who are more theoretically inclined.

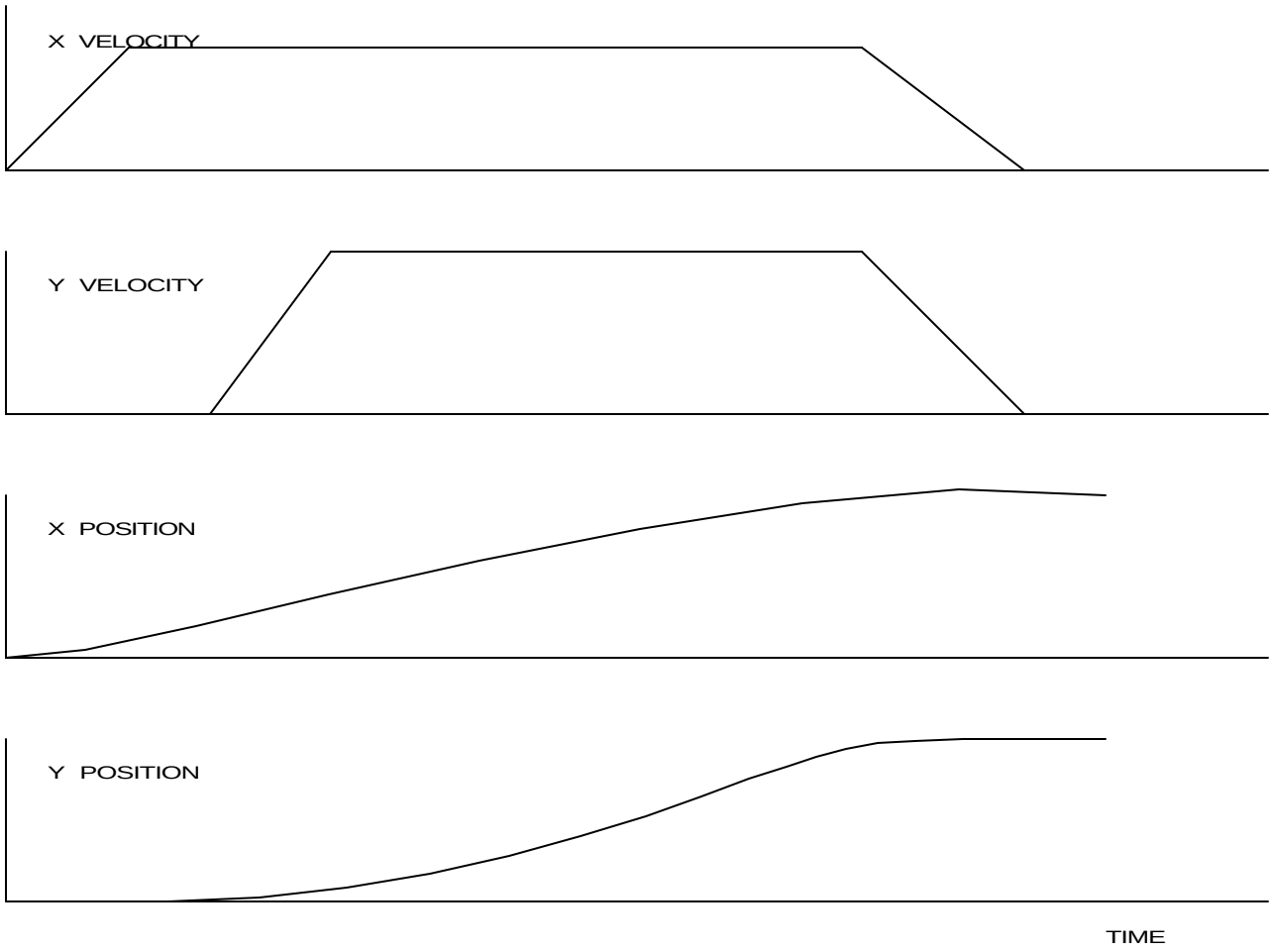


Figure 10.3 - Velocity and Position Profiles

Operation of Closed-Loop Systems

To understand the operation of a servo system, we may compare it to a familiar closed-loop operation, adjusting the water temperature in the shower. One control objective is to keep the temperature at a comfortable level, say 90 degrees F. To achieve that, our skin serves as a temperature sensor and reports to the brain (controller). The brain compares the actual temperature, which is called the feedback signal, with the desired level of 90 degrees F. The difference between the two levels is called the error signal. If the feedback temperature is too low, the error is positive, and it triggers an action which raises the water temperature until the temperature error is reduced sufficiently.

The closing of the servo loop is very similar. Suppose that we want the motor position to be at 90 degrees. The motor position is measured by a position sensor, often an encoder, and the position feedback is sent to the controller. Like the brain, the controller determines the position error, which is the difference between the commanded position of 90 degrees and the position feedback. The controller then outputs a signal that is proportional to the position error. This signal produces a proportional current in the motor, which causes a motion until the error is reduced. Once the error becomes small, the resulting current will be too small to overcome the friction, causing the motor to stop.

The analogy between adjusting the water temperature and closing the position loop carries further. We have all learned the hard way, that the hot water faucet should be turned at the "right" rate. If you turn it too slowly, the temperature response will be slow, causing discomfort. Such a slow reaction is called overdamped response.

The results may be worse if we turn the faucet too fast. The overreaction results in temperature oscillations. When the response of the system oscillates, we say that the system is unstable. Clearly, unstable responses are bad when we want a constant level.

What causes the oscillations? The basic cause for the instability is a combination of delayed reaction and high gain. In the case of the temperature control, the delay is due to the water flowing in the pipes. When the human reaction is too strong, the response becomes unstable.

Servo systems also become unstable if their gain is too high. The delay in servo systems is between the application of the current and its effect on the position. Note that the current must be applied long enough to cause a significant effect on the velocity, and the velocity change must last long enough to cause a position change. This delay, when coupled with high gain, causes instability.

This motion controller includes a special filter which is designed to help the stability and accuracy. Typically, such a filter produces, in addition to the proportional gain, damping and integrator. The combination of the three functions is referred to as a PID filter.

The filter parameters are represented by the three constants KP, KI and KD, which correspond to the proportional, integral and derivative term respectively.

The damping element of the filter acts as a predictor, thereby reducing the delay associated with the motor response.

The integrator function, represented by the parameter KI, improves the system accuracy. With the KI parameter, the motor does not stop until it reaches the desired position exactly, regardless of the level of friction or opposing torque.

The integrator also reduces the system stability. Therefore, it can be used only when the loop is stable and has a high gain.

The output of the filter is applied to a digital-to-analog converter (DAC). The resulting output signal in the range between +10 and -10 Volts is then applied to the amplifier and the motor.

The motor position, whether rotary or linear is measured by a sensor. The resulting signal, called position feedback, is returned to the controller for closing the loop.

The following section describes the operation in a detailed mathematical form, including modeling, analysis and design.

System Modeling

The elements of a servo system include the motor, driver, encoder and the controller. These elements are shown in Fig. 10.4. The mathematical model of the various components is given below.

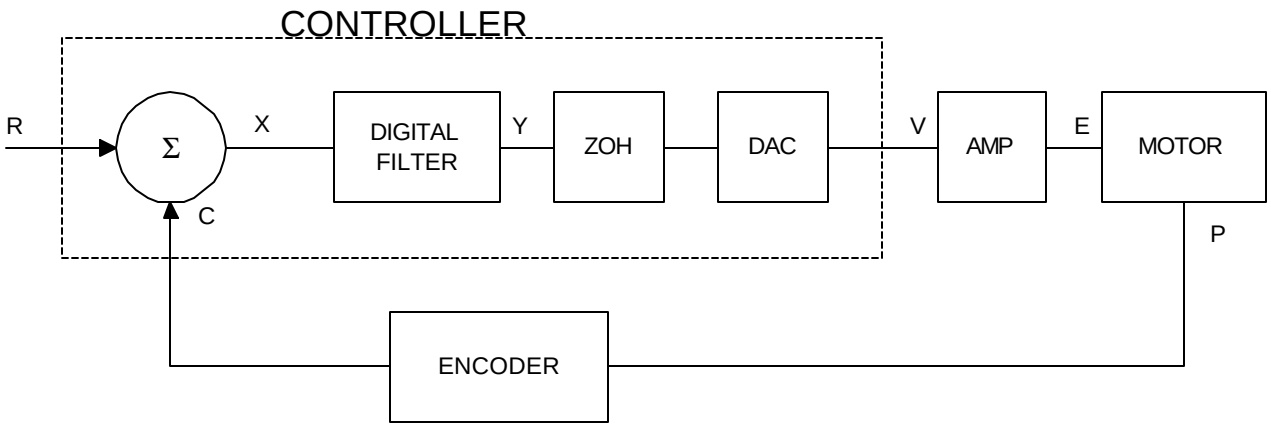


Figure 10.4 - Functional Elements of a Servo Control System

Motor-Amplifier

The motor amplifier may be configured in three modes:

1. Voltage Drive
2. Current Drive
3. Velocity Loop

The operation and modeling in the three modes is as follows:

Voltage Drive

The amplifier is a voltage source with a gain of Kv [V/V]. The transfer function relating the input voltage, V, to the motor position, P, is

$$P/V = K_V / \left[K_t S(ST_m + 1)(ST_e + 1) \right]$$

where

$$T_m = RJ / K_t^2 \quad [s]$$

and

$$T_e = L/R \quad [s]$$

and the motor parameters and units are

K_t	Torque constant [Nm/A]
R	Armature Resistance Ω
J	Combined inertia of motor and load [kg.m^2]
L	Armature Inductance [H]

When the motor parameters are given in English units, it is necessary to convert the quantities to MKS units. For example, consider a motor with the parameters:

K_t = 14.16 oz - in/A = 0.1 Nm/A

R = 2 Ω

J = 0.0283 oz-in-s^2 = 2.10^{-4} kg . m^2

$L = 0.004H$

Then the corresponding time constants are

$T_m = 0.04$ sec

and

$T_e = 0.002$ sec

Assuming that the amplifier gain is Kv = 4, the resulting transfer function is

$P/V = 40/[s(0.04s+1)(0.002s+1)]$

Current Drive

The current drive generates a current I, which is proportional to the input voltage, V, with a gain of Ka. The resulting transfer function in this case is

$P/V = K_a K_t / Js^2$

where Kt and J are as defined previously. For example, a current amplifier with $K_a = 2$ A/V with the motor described by the previous example will have the transfer function:

$P/V = 1000/s^2$ [rad/V]

If the motor is a DC brushless motor, it is driven by an amplifier that performs the commutation. The combined transfer function of motor amplifier combination is the same as that of a similar brush motor, as described by the previous equations.

Velocity Loop

The motor driver system may include a velocity loop where the motor velocity is sensed by a tachometer and is fed back to the amplifier. Such a system is illustrated in Fig. 10.5. Note that the transfer function between the input voltage V and the velocity ω is:

$\omega/V = [K_a K_t/Js]/[1+K_a K_t K_g/Js] = 1/[K_g(sT_1+1)]$

where the velocity time constant, T1, equals

$T1 = J/K_a K_t K_g$

This leads to the transfer function

$P/V = 1/[K_g s(sT1+1)]$

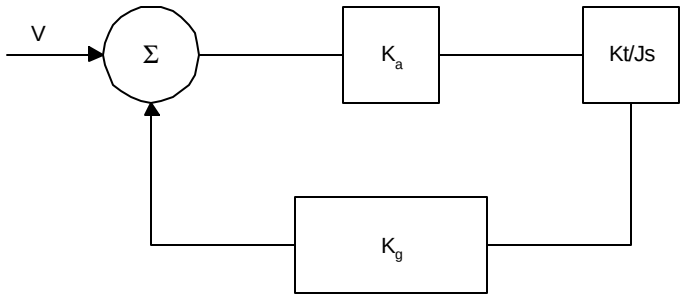
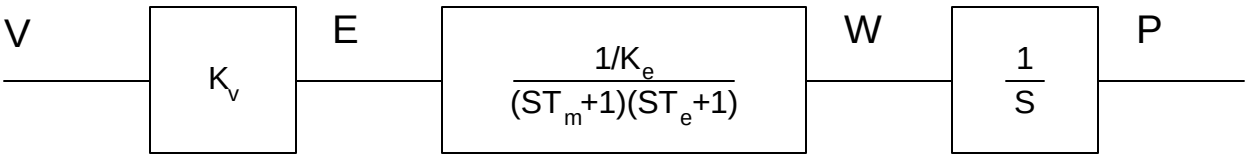


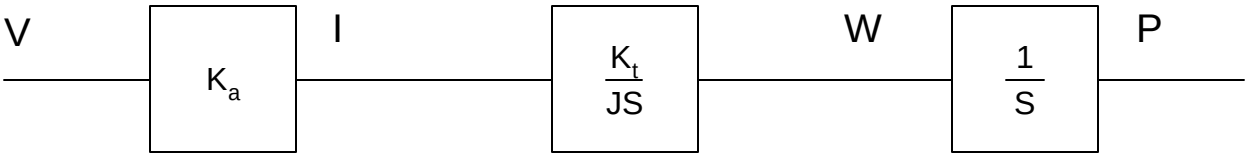
Figure 10.5 - Elements of velocity loops

The resulting functions derived above are illustrated by the block diagram of Fig. 10.6.

VOLTAGE SOURCE



CURRENT SOURCE



VELOCITY LOOP

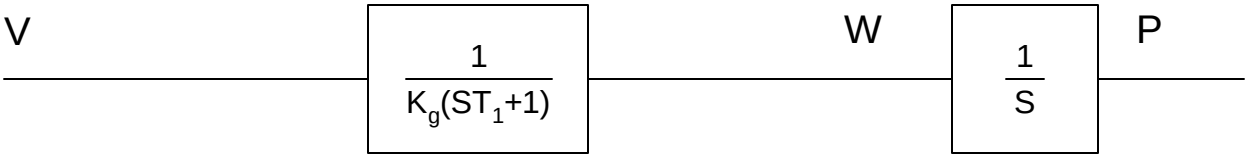


Figure 10.6 - Mathematical model of the motor and amplifier in three operational modes

Encoder

The encoder generates N pulses per revolution. It outputs two signals, Channel A and B, which are in quadrature. Due to the quadrature relationship between the encoder channels, the position resolution is increased to 4N quadrature counts/rev.

The model of the encoder can be represented by a gain of

$K_f = 4N/2\pi$ [count/rad]

For example, a 1000 lines/rev encoder is modeled as

$K_f = 638$

DAC

The DAC or D-to-A converter converts a 16-bit number to an analog voltage. The input range of the numbers is 65536 and the output voltage range is +/-10V or 20V. Therefore, the effective gain of the DAC is

$$K = 20/65536 = 0.0003 \quad [V/count]$$

Digital Filter

The digital filter has a transfer function of $D(z) = K(z-A)/z + Cz/z-1$ and a sampling time of T.

The filter parameters, K, A and C are selected by the instructions KP, KD, KI or by GN, ZR and KI, respectively. The relationship between the filter coefficients and the instructions are:

$K = (KP + KD) \cdot 4$ or $K = GN \cdot 4$

$A = KD/(KP + KD)$ or $A = ZR$

$C = KI/2$

This filter includes a lead compensation and an integrator. It is equivalent to a continuous PID filter with a transfer function G(s).

$$G(s) = P + sD + I/s$$

$$P = K(1-A) = 4 \cdot KP$$

$$D = T \cdot K \cdot A = 4 \cdot T \cdot KD$$

$$I = C/T = KI/2T$$

For example, if the filter parameters of the DMC 1300 are

$$KP = 4$$

$$KD = 36$$

$$KI = 2$$

$$T = 0.001 \text{ s}$$

the digital filter coefficients are

$$K = 160$$

$$A = 0.9$$

$$C = 1$$

and the equivalent continuous filter, G(s), is

$$G(s) = 16 + 0.144s + 1000/s$$

ZOH

The ZOH, or zero-order-hold, represents the effect of the sampling process, where the motor command is updated once per samp ling period. The effect of the ZOH can be modeled by the transfer function

$$H(s) = 1/(1+sT/2)$$

If the sampling period is T = 0.001, for example, H(s) becomes:

$$H(s) = 2000/(s+2000)$$

However, in most applications, H(s) may be approximated as one.

This completes the modeling of the system elements. Next, we discuss the system analysis.

System Analysis

To analyze the system, we start with a block diagram model of the system elements. The analysis procedure is illustrated in terms of the following example.

Consider a position control system with the DMC 1300 controller and the following parameters:

$K_t = 0.1$	Nm/A	Torque constant
$J = 2.10^{-4}$	$kg.m^2$	System moment of inertia
$R = 2$	Ω	Motor resistance
$K_a = 4$	Amp/Volt	Current amplifier gain
KP = 12.5		Digital filter gain
KD = 245		Digital filter zero
KI = 0		No integrator
N = 500	Counts/rev	Encoder line density
T = 1	ms	Sample period

The transfer function of the system elements are:

Motor

$$M(s) = P/I = Kt/Js2 = 500/s^2 \text{ [rad/A]}$$

Amp

$$K_a = 4 \text{ [Amp/V]}$$

DAC

$$K_d = 0.0003 \text{ [V/count]}$$

Encoder

$$K_f = 4N/2\pi = 318 \text{ [count/rad]}$$

ZOH

$$2000/(s+2000)$$

Digital Filter

$$KP = 12.5,\ KD = 245,\ T = 0.001$$

Therefore,

$$D(z) = 50 + 980\ (1-z-1)$$

Accordingly, the coefficients of the continuous filter are:

$$P = 50$$

$$D = 0.98$$

The filter equation may be written in the continuous equivalent form:

$$G(s) = 50 + 0.98s$$

The system elements are shown in Fig. 10.7.

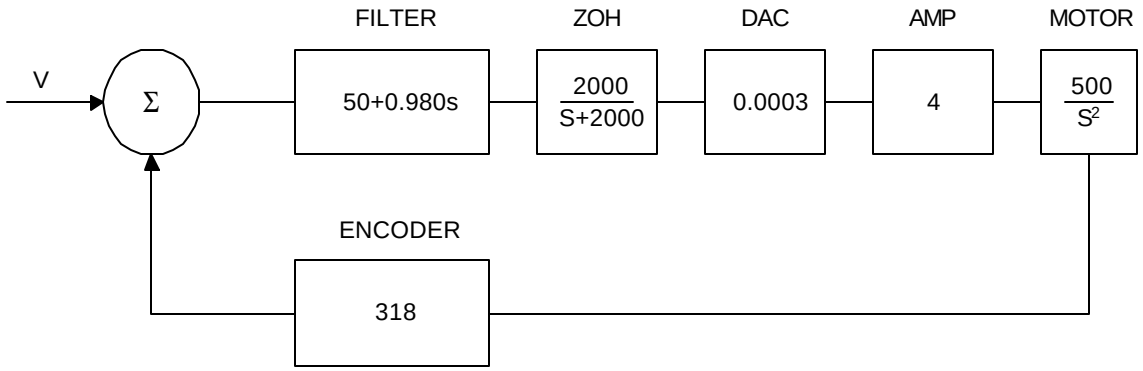


Figure 10.7 - Mathematical model of the control system

The open loop transfer function, A(s), is the product of all the elements in the loop.

$$A = 390{,}000\ (s+51)/[s^2(s+2000)]$$

To analyze the system stability, determine the crossover frequency, ω_C at which A(j ω_C) equals one. This can be done by the Bode plot of A(j ω_C), as shown in Fig. 10.8.

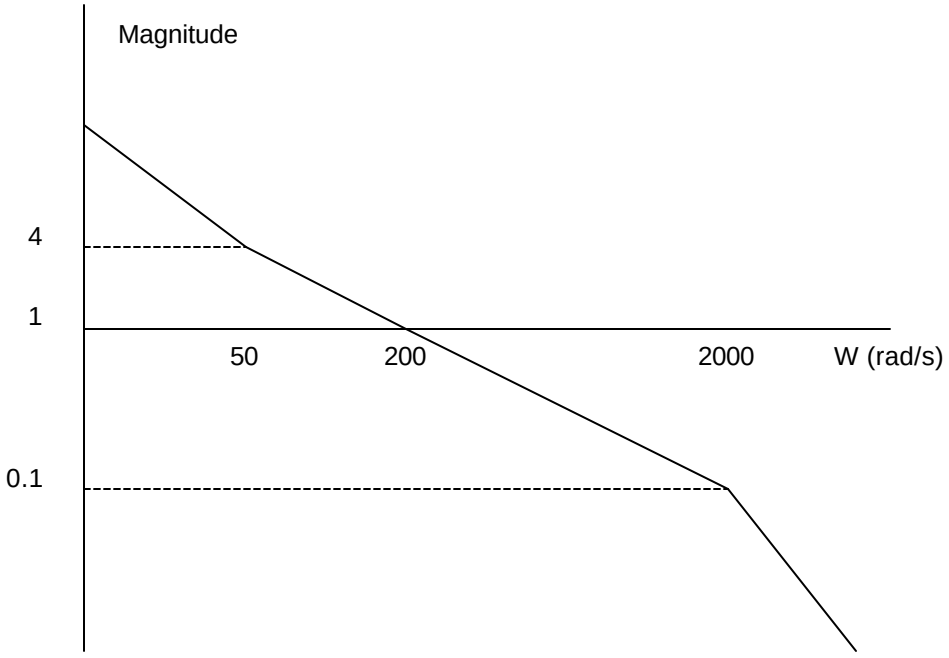


Figure 10.8 - Bode plot of the open loop transfer function

For the given example, the crossover frequency was computed numerically resulting in 200 rad/s.

Next, we determine the phase of A(s) at the crossover frequency.

$$A(j200) = 390{,}000\ (j200+51)/[(j200)^2 \cdot (j200 + 2000)]$$

$$\alpha = \text{Arg}[A(j200)] = \tan^{-1}(200/51) - 180° - \tan^{-1}(200/2000)$$

$$\alpha = 76° - 180° - 6° = -110°$$

Finally, the phase margin, PM, equals

$$PM = 180° + \alpha = 70°$$

As long as PM is positive, the system is stable. However, for a well damped system, PM should be between 30 degrees and 45 degrees. The phase margin of 70 degrees given above indicated overdamped response.

Next, we discuss the design of control systems.

System Design and Compensation

The closed-loop control system can be stabilized by a digital filter, which is preprogrammed in the DMC 1300 controller. The filter parameters can be selected by the user for the best compensation. The following discussion presents an analytical design method.

The Analytical Method

The analytical design method is aimed at closing the loop at a crossover frequency, ω_c, with a phase margin PM. The system parameters are assumed known. The design procedure is best illustrated by a design example.

Consider a system with the following parameters:

K_t	Nm/A	Torque constant
$J = 2.10^{-4}$	$kg.m^2$	System moment of inertia
$R = 2$	Ω	Motor resistance
$K_a = 2$	Amp/Volt	Current amplifier gain
$N = 1000$	Counts/rev	Encoder line density

The DAC of the DMC 1300 outputs +/-10V for a 14-bit command of +/-8192 counts.

The design objective is to select the filter parameters in order to close a position loop with a crossover frequency of $\omega_c = 500$ rad/s and a phase margin of 45 degrees.

The first step is to develop a mathematical model of the system, as discussed in the previous system.

Motor

$$M(s) = P/I = K_t/Js^2 = 1000/s^2$$

Amp

$$K_a = 2 \qquad [Amp/V]$$

DAC

$$K_d = 20/65536 = .0003$$

Encoder

$$K_f = 4N/2\pi = 636$$

ZOH

$$H(s) = 2000/(s+2000)$$

Compensation Filter

$$G(s) = P + sD$$

The next step is to combine all the system elements, with the exception of G(s), into one function, L(s).

$$L(s) = M(s)\, K_a\, K_d\, K_f\, H(s) = 0.3175*10^7/[s^2(s+2000)]$$

Then the open loop transfer function, A(s), is

$$A(s) = L(s)\, G(s)$$

Now, determine the magnitude and phase of L(s) at the frequency $\omega_c = 500$.

$$L(j500) = 0.3175*10^7/[(j500)^2\,(j500+2000)]$$

This function has a magnitude of

$$|L(j500)| = 0.00625$$

and a phase

$$Arg[L(j500)] = -180° - \tan^{-1}(500/2000) = -194°$$

G(s) is selected so that A(s) has a crossover frequency of 500 rad/s and a phase margin of 45 degrees. This requires that

$$|A(j500)| = 1$$

$$Arg\,[A(j500)] = -135°$$

However, since

$$A(s) = L(s)\, G(s)$$

then it follows that G(s) must have magnitude of

$$|G(j500)| = |A(j500)/L(j500)| = 160$$

and a phase

$$arg\,[G(j500)] = arg\,[A(j500)] - arg\,[L(j500)] = -135° + 194° = 59°$$

In other words, we need to select a filter function G(s) of the form

$$G(s) = P + sD$$

so that at the frequency $\omega_c = 500$, the function would have a magnitude of 160 and a phase lead of 59 degrees.

These requirements may be expressed as:

$$|G(j500)| = |P + (j500D)| = 160$$

and

$$arg\,[G(j500)] = \tan^{-1}[500D/P] = 59°$$

The solution of these equations leads to:

$$P = 40\cos 59° = 82.4$$

$$500D = 40\sin 59° = 137.2$$

Therefore,

$$D = 0.2744$$

and

$$G = 82.4 + 0.2744s$$

The function G is equivalent to a digital filter of the form:

$$D(z) = 4 \bullet KP + 4 \bullet KD(1-z^{-1})$$

where

KP = P/4

and

KD = D/ (4 • T)

Assuming a sampling period of T=1ms, the parameters of the digital filter are:

KP = 20.6

KD = 68.6

The DMC 1300 can be programmed with the instruction:

KP 20.6

KD 68.6

In a similar manner, other filters can be programmed. The procedure is simplified by the following table, which summarizes the relationship between the various filters.

Equivalent Filter Form

	DMC 1300
Digital	$D(z) = K(z-A/z) + Cz/(z-1\ \ )$
Digital	$D(z) = 4\ KP + 4\ KD(1-z^{-1}) + KI/2(1-z^{-1})$
KP, KD, KI	$K = (KP + KD) \cdot 4$
	$A = KD/(KP+KD)$
	$C = KI/2$
Digital	$D(z) = 4\ GN(z-ZR)/z + KI\ z/2(z-1)$
GN, ZR, KI	$K = 4\ GN$
	$A = ZR$
	$C = KI/2$
Continuous	$G(s) = P + Ds + I/s$
PID, T	$P = 4\ KP$
	$D = 4\ T*KD$
	$I = KI/2T$

THIS PAGE LEFT BLANK INTENTIONALLY

Chapter 11 Command Reference

Command Descriptions

Each executable instruction is listed in the following section in alphabetical order. Below is a description of the information which is provided for each command.

The two-letter Opcode for each instruction is placed in the upper right corner. Commands that have a binary equivalent list the binary value next to the ASCII command in parenthesis. Below the opcode is a description of the command and required arguments.

Axes Arguments

Some commands require the user to identify the specific axes to be affected. These commands are followed by uppercase X,Y,Z, W or A,B,C,D,E,F,G and H. No commas are needed and the order of axes is not important. Do not insert any spaces prior to any command. For example, STX; AMX is invalid because there is a space after the semicolon. When no argument is given, the command is executed for all axes.

Valid XYZW syntax

SH X	Servo Here, X only
SH XYW	Servo Here, X,Y and W axes
SH XZW	Servo Here, X,Z and W axes
SH XYZW	Servo Here, X,Y,Z and W axes
SH BCAD	Servo Here, A,B,C and D axes (Note: ABCD IS the same as XYZW)
SH ADEG	Servo Here, A,D,E and G axes (Note: AD is the same as XW)
SH H	Servo Here, H axis only
SH	Servo Here, all axes

Parameter Arguments

Some commands require numerical arguments to be specified following the instruction. In the argument description, these commands are followed by lower case x,y,z,w or a,b,c,d,e,f,g,h where the lowercase letter represents the value. Values may be specified for any axis separately or any combination of axes. The argument for each axis is separated by commas. Examples of valid syntax are listed below.

Valid x,y,z,w syntax

AC x	Specify argument for x axis only
AC x,y	Specify x and y only
AC x,,z	Specify x and z only
AC x,y,z,w	Specify x,y,z,w
AC a,b,c,d	Specify arguments for a,b,c,d (Note: a,b,c,d are the same as x,y,z,w)
AC ,b,,,e	Specify b and e axis only (Note: b and y axis are the same)
AC ,,,e,f	Specify e and f (Note: e and z axis are the same)

Where x,y,z,w and a,b,c,d,e,f,g and h are replaced by actual values.

Direct Command Arguments

An alternative method for specifying data is to set data for individual axes using an axis designator followed by an equals sign. The * symbol defines data for all axes to be the same. For example:

PRY=1000	Sets Y axis data at 1000
PR*=1000	Sets all axes to 1000

Interrogation

Most commands accept a question mark (?) as an argument. This argument causes the controller to return parameter information listed in the command description. Type the command followed by a ? for each axis requested. The syntax format is the same as the parameter arguments described above except '?' replaces the values.

PR ?	The controller will return the PR value for the X axis
PR ,,,?	The controller will return the PR value for the W axis
PR ?,?,?,?	The controller will return the PR value for the A,B,C and D axes
PR ,,,,,,,?	The controller will return the PR value for the H axis

Operand Usage

Most commands have a corresponding operand that can be used for interrogation. The Operand Usage description provides proper syntax and the value returned by the operand. Operands must be used inside of valid DMC expressions. For example, to display the value of an operand, the user could use the command:

MG ‘operand’

All of the command operands begin with the underscore character (_). For example, the value of the current position on the X axis can be assigned to the variable ‘V’ with the command:

V=_TPX

Usage Description

The Usage description specifies the restrictions on proper command usage. The following provides an explanation of the command information provided:

"While Moving" states whether or not the command is valid while the controller is performing a previously defined motion.

"In a program" states whether the command may be used as part of a user-defined program.

"Command Line" states whether the command may be used other than in a user-defined program.

"Can be Interrogated" states whether or not the command can be interrogated by using the ? as a command argument.

"Used as an Operand" states whether the command has an associated operand.

Default Description

In the command description, the DEFAULT section provides the default values for controller setup parameters. These parameters can be changed and the new values can be saved in the controller's non-volatile memory by using the command, BN. If the setup parameters are not saved in non-volatile memory, the default values will automatically reset when the system is reset. A reset occurs when the power is turned off and on, when the reset button is pushed, or the command, RS, is given.

When a master reset occurs, the controller will always reset all setup parameters to their default values and the non-volatile memory is cleared to the factory state. A master reset is executed by the command, <ctrl R> <ctrl S> <Return> OR by powering up or resetting the controller with the MRST jumper or dip switch on.

For example, the command KD is used to set the Derivative Constant for each axis. The default value for the derivative constant is 64. If this parameter is not set by using the command, KD, the controller will automatically set this value to 64 for each axis. If the Derivative Constant is changed but not saved in non-volatile memory, the default value of 64 will be used if the controller is reset or upon power up of the controller. If this value is set and saved in non-volatile memory, it will be restored upon reset until a master reset is given to the controller.

The default format describes the format for numerical values which are returned when the command is interrogated. The format value represents the number of digits before and after the decimal point.

Servo and Stepper Motor Notation:

Your motion controller has been designed to work with both servo and stepper type motors. Installation and system setup will vary depending upon whether the controller will be used with stepper motors, or servo motors. To make finding the appropriate instructions faster and easier, icons will be next to any information that applies exclusively to one type of system. Otherwise, assume that the instructions apply to all types of systems. The icon legend is shown below.

Attention: Pertains to servo motor use.

Attention: Pertains to stepper motor use.

AB (Binary D3)

FUNCTION: Abort

DESCRIPTION:

AB (Abort) stops a motion instantly without a controlled deceleration. If there is a program operating, AB also aborts the program unless a 1 argument is specified. The command, AB, will shut off the motors for any axis in which the off-on-error function is enabled (see command "OE" on page 243).

ARGUMENTS: AB n where

n = no argument or 1

1 aborts motion without aborting program, 0 aborts motion and program

AB aborts motion on all axes in motion and cannot stop individual axes.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	---
Default Format	---

RELATED COMMANDS:

"SH" on page 263	Turns servos back on if they were shut-off by Abort and OE1.

EXAMPLES:

AB	Stops motion
OE 1,1,1,1	Enable off-on-error
AB	Shuts off motor command and stops motion
#A	Label - Start of program
JG 20000	Specify jog speed on X-axis
BGX	Begin jog on X-axis
WT 5000	Wait 5000 msec
AB1	Stop motion without aborting program
WT 5000	Wait 5000 milliseconds
SH	Servo Here
JP #A	Jump to Label A
EN	End of the routine

Hint: *Remember to use the parameter 1 following AB if you only want the motion to be aborted. Otherwise, your application program will also be aborted.*

AC (Binary CC)

FUNCTION: Acceleration

DESCRIPTION:

The Acceleration (AC) command sets the linear acceleration rate of the motors for independent moves, such as PR, PA and JG moves. The parameters input will be rounded down to the nearest factor of 1024. The units of the parameters are counts per second squared. The acceleration rate may be changed during motion. The DC command is used to specify the deceleration rate.

ARGUMENTS: AC x,y,z,w ACX=x AC a,b,c,d,e,f,g,h where

x,y,z,w are unsigned numbers in the range in the range 1024 to 67107840

"?" returns the acceleration value for the specified axes.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	25600
In a Program	Yes	Default Format	8.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		

OPERAND USAGE:

_ACx contains the value of acceleration for the specified axis.

RELATED COMMANDS:

"DC" on page 191	Specifies deceleration rate.
"FA" on page 204	Feedforward Acceleration
"IT" on page 219	Smoothing constant - S-curve

EXAMPLES:

AC 150000,200000,300000,400000	Set X-axis acceleration to 150000, Y-axis to 200000 counts/sec^2, the Z-axis to 300000 counts/sec^2, and the W-axis to 400000 count/sec^2.
AC ?,?,?,?	Request the Acceleration
0149504,0199680,0299008,0399360	Return Acceleration
	(resolution, 1024)
V=_ACY	Assigns the Y acceleration to the variable V

Hint: *Specify realistic acceleration rates based on your physical system such as motor torque rating, loads, and amplifier current rating. Specifying an excessive acceleration will cause large following error during acceleration and the motor will not follow the commanded profile. The acceleration feedforward command FA will help minimize the error.*

AD (Binary A2)

FUNCTION: After Distance

DESCRIPTION:

The After Distance (AD) command is a trippoint used to control the timing of events. This command will hold up the execution of the following command until one of the following conditions have been met:

1. The commanded motor position crosses the specified relative distance from the start of the move.
2. The motion profiling on the axis is complete.
3. The commanded motion is in the direction which moves away from the specified position.

The units of the command are quadrature counts. Only one axis may be specified at a time. The motion profiler must be on or the trippoint will automatically be satisfied.

ARGUMENTS: AD x or AD,y or AD,,z or AD,,,w ADX=x AD a,b,c,d,e,f,g,h where

x,y,z,w are unsigned integers in the range 0 to 2147483647 decimal.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		

RELATED COMMANDS:

"AD" on page 164	After distance for repetitive triggering
"AV" on page 173	After distance for vector moves

EXAMPLES:

#A;DP0,0,0,0	Begin Program
PR 10000,20000,30000,40000	Specify positions
BG	Begin motion
AD 5000	After X reaches 5000
MG "Halfway to X";TPX	Send message
AD ,10000	After Y reaches 10000
MG "Halfway to Y";TPY	Send message
AD ,,15000	After Z reaches 15000
MG "Halfway to Z";TPZ	Send message
AD ,,,20000	After W reaches 20000
MG "Halfway to W";TPW	Send message
EN	End Program

Hint: *The AD command is accurate to the number of counts that occur in 2 msec. Multiply your speed by 2 msec to obtain the maximum position error in counts. Remember AD measures incremental*

distance from start of move on one axis.

AI (Binary A1)

FUNCTION: After Input

DESCRIPTION:

The AI command is used in motion programs to wait until after the specified input has occurred. If n is positive, it waits for the input to go high. If n is negative, it waits for n to go low.

ARGUMENTS: AI +/-n where

n is an integer in the range 1 to 8 decimal

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		

RELATED COMMANDS:

@IN[n]	Function to read input 1 through 8
"II" on page 215	Input interrupt
#ININT	Label for input interrupt

EXAMPLES:

#A	Begin Program
AI 8	Wait until input 8 is high
SP 10000	Speed is 10000 counts/sec
AC 20000	Acceleration is 20000 counts/sec2
PR 400	Specify position
BG X	Begin motion
EN	End Program

Hint: *The AI command actually halts execution until specified input is at desired logic level. Use the conditional Jump command (JP) or input interrupt (II) if you do not want the program sequence to halt.*

AL (Binary 90)

FUNCTION: Arm Latch

DESCRIPTION:

The AL command enables the latching function (high speed main or auxiliary position capture) of the controller. When the position latch is armed, the main or auxiliary encoder position will be captured upon a low going signal. Each axis has a position latch and can be activated through the general inputs: Input 1 (X or A axis), Input 2 (Y or B axis), Input 3 (Z or C axis), Input 4 (W or D axis), Input 5 (E axis), Input 6 (F axis), Input 7 (6 axis). The command RL returns the captured position for the specified axes. When interrogated the AL command will return a 1 if the latch for that axis is armed or a zero after the latch has occurred. The CN command will change the polarity of the latch.

ARGUMENTS: AL XYZW where

X,Y,Z,W specifies the X,Y,Z,W axes.

DPRAM:

The latch status can be read at bit 2 of the Status #2 address in the Axis Buffer. Bit 6 of the Switches address in the Axis Buffer will also indicate the status of the latch, while Bit 7 of that address will indicate when the latch has occurred.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

DEFAULTS:

Default Value	0
Default Format	1.0

OPERAND USAGE:

_ALx contains the state of the specified latch. 0 = not armed, 1 = armed.

RELATED COMMANDS:

"RL" on page 254 Report Latch

EXAMPLES:

```
#START          Start program
ALY             Arm Y-axis latch
JG,50000        Set up jog at 50000 counts/sec
BGY             Begin the move
#LOOP           Loop until latch has occurred
JP #LOOP,_ALY=1
RLY             Transmit the latched position
EN              End of program
```

AM (Binary A4)

FUNCTION: After Move

DESCRIPTION:

The AM command is a trippoint used to control the timing of events. This command will hold up execution of the following commands until the current move on the specified axis or axes is completed. Any combination of axes or a motion sequence may be specified with the AM command. For example, AM XY waits for motion on both the X and Y axis to be complete. AM with no parameter specifies that motion on all axes is complete.

ARGUMENTS: AM XYZW or AMS where

X,Y,Z,W specifies X,Y,Z or W axis and S specifies sequence. No argument specifies that motion on all axes is complete.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	0
Default Format	1.0

RELATED COMMANDS:

"BG" on page 174	_BGx contains a 0 if motion complete

EXAMPLES:

#MOVE	Program MOVE
PR 5000,5000,5000,5000	Position relative moves
BG X	Start the X-axis
AM X	After the move is complete on X,
BG Y	Start the Y-axis
AM Y	After the move is complete on Y,
BG Z	Start the Z-axis
AM Z	After the move is complete on Z
BG W	Start the W-axis
AM W	After the move is complete on W
EN	End of Program

Hint: *AM is a very important command for controlling the timing between multiple move sequences. For example, if the X-axis is in the middle of a position relative move (PR) you cannot make a position absolute move (PAX, BGX) until the first move is complete. Use AMX to halt the program sequences until the first motion is complete. AM tests for profile completion. The actual motor may still be moving. Another method for testing motion complete is to check for the internal variable, _BG, being equal to zero.*

AP (Binary A3)

FUNCTION: After Absolute Position

DESCRIPTION:

The After Position (AP) command is a trippoint used to control the timing of events. This command will hold up the execution of the following command until one of the following conditions have been met:

1. The commanded motor position crosses the specified absolute position.
2. The motion profiling on the axis is complete.
3. The commanded motion is in the direction which moves away from the specified position.

The units of the command are quadrature counts. Only one axis may be specified at a time. The motion profiler must be on or the trippoint will automatically be satisfied

ARGUMENTS: APx or AP,y or AP,,z or AP,,,w APX=x AP abcdefgh where

x,y,z,w are signed integers in the range -2147483648 to 2147483647 decimal

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		

RELATED COMMANDS:

"AD" on page 164	Trippoint for relative distances
“MF" on page 236	Trippoint for forward motion

EXAMPLES:

#TEST	Program B
DP0	Define zero
JG 1000	Jog mode (speed of 1000 counts/sec)
BG X	Begin move
AP 2000	After passing the position 2000
V1=_TPX	Assign V1 X position
MG "Position is", V1=	Print Message
ST	Stop
EN	End of Program

Hint: *The accuracy of the AP command is the number of counts that occur in 2 msec. Multiply the speed by 2 msec to obtain the maximum error. AP tests for absolute position. Use the AD command to measure incremental distances.*

AR (Binary CF)

FUNCTION: After Relative Distance

DESCRIPTION:

The After Relative (AR) command is a trippoint used to control the timing of events. This command will hold up the execution of the following command until one of the following conditions have been met:

1. The commanded motor position crosses the specified relative distance from either the start of the move or the last AR or AD command.
2. The motion profiling on the axis is complete.
3. The commanded motion is in the direction which moves away from the specified position.

The units of the command are quadrature counts. Only one axis may be specified at a time. The motion profiler must be on or the trippoint will automatically be satisfied.

ARGUMENTS: ARx or AR,y or AR,,z or AR,,,w ARX=X AR abcdefgh where

x,y,z,w are unsigned integers in the range 0 to 2147483647 decimal.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

"AV" on page 173	Trippoint for after vector position for coordinated moves
"AP (Binary A3)" on page 169	Trippoint for after absolute position

EXAMPLES:

#A;DP 0,0,0,0	Begin Program
JG 50000,,,7000	Specify speeds
BG XW	Begin motion
#B	Label
AR 25000	After passing 25000 counts of relative distance on X-axis
MG "Passed_X";TPX	Send message on X-axis
JP #B	Jump to Label #B
EN	End Program

Hint: *AR is used to specify incremental distance from last AR or AD command. Use AR if multiple position trippoints are needed in a single motion sequence.*

AS (Binary A5)

FUNCTION: At Speed

DESCRIPTION:

The AS command is a trippoint that occurs when the generated motion profile has reached the specified speed. This command will hold up execution of the following command until the speed is reached. The AS command will operate after either accelerating or decelerating. If the speed is not reached, the trippoint will be triggered after the motion is stopped (after deceleration).

ARGUMENTS: AS X or AS Y or AS Z or AS W or AS S AS ABCDEFGH where

XYZWS specifies X,Y,Z,W axis or sequence

DPRAM:

Bit 5 of the Status #2 address in the Axis Buffer will indicate if the controller is at slew speed.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	-
Default Format	-

EXAMPLES:

#SPEED	Program A
PR 100000	Specify position
SP 10000	Specify speed
BG X	Begin X
ASX	After speed is reached
MG "At Speed"	Print Message
EN	End of Program

WARNING:

The AS command applies to a trapezoidal velocity profile only with linear acceleration. AS used with S-curve profiling will be inaccurate.

AT (Binary A7)

FUNCTION: At Time

DESCRIPTION:

The AT command is a trippoint which is used to hold up execution of the next command until after the specified time has elapsed. The time is measured with respect to a defined reference time. AT 0 establishes the initial reference. AT n specifies n msec from the reference. AT -n specifies n msec from the reference and establishes a new reference after the elapsed time period.

ARGUMENTS: AT n where

n is a signed integer in the range 0 to 2 Billion

n = 0 defines a reference time at current time

positive n waits n msec from reference

negative n waits n msec from reference and sets new reference after elapsed time period

(AT -n is equivalent to AT n; AT 0)

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	0
Default Format	-

EXAMPLES:

The following commands are sent sequentially

AT 0	Establishes reference time 0 as current time
AT 50	Waits 50 msec from reference 0
AT 100	Waits 100 msec from reference 0
AT -150	Waits 150 msec from reference 0 and sets new reference at 150
AT 80	Waits 80 msec from new reference (total elapsed time is 230 msec)

AV (Binary AB)

FUNCTION: After Vector Distance

DESCRIPTION:

The AV command is a trippoint which is used to hold up execution of the next command during coordinated moves such as VP,CR or LI. This trippoint occurs when the path distance of a sequence reaches the specified value. The distance is measured from the start of a coordinated move sequence or from the last AV command. The units of the command are quadrature counts.

ARGUMENTS: AV n where

n is an unsigned integer in the range 0 to 2147483647 decimal

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

DEFAULTS:

Default Value	0
Default Format	-

OPERAND USAGE:

_AV contains the vector distance from the start of the sequence. _AV is valid in the linear mode, LM and in the vector mode, VM.

EXAMPLES:

#MOVE;DP 0,0	Label
LMXY	Linear move for X,Y
LI 1000,2000	Specify distance
LI 2000,3000	Specify distance
LE	
BGS	Begin
AV 500	After path distance = 500,

BG (Binary CE)

FUNCTION: Begin

DESCRIPTION:

The BG command starts a motion on the specified axis or sequence.

ARGUMENTS: BG XYZWS BG ABCDEFGH where

XYZW are X,Y,Z,W axes and S is coordinated sequence

DPRAM:

Bit 7 of the Status #1 address in the Axis Buffer will indicate if there is motion on a given axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

DEFAULTS:

Default Value	0
Default Format	-

OPERAND USAGE:

_BG contains a '0' if motion complete on the specified axis, otherwise contains a '1'.

RELATED COMMANDS:

"AM" on page 168	After motion complete
"ST" on page 265	Stop motion

EXAMPLES:

PR 2000,3000,,5000	Set up for a relative move
BG XYW	Start the X,Y and W motors moving
HM	Set up for the homing
BGX	Start only the X-axis moving
JG 1000,4000	Set up for jog
BGY	Start only the Y-axis moving
YSTATE=_BGY	Assign a 1 to YSTATE if the Y-axis is performing a move
VP 1000,2000	Specify vector position
VS 20000	Specify vector velocity
BGS	Begin coordinated sequen0ce
VMXY	Vector Mode
VP 4000,-1000	Specify vector position
VE	Vector End
PR ,,8000,5000	Specify Z and W position
BGSZW	Begin sequence and Z,W motion
MG _BGS	Displays a 1 if coordinated sequence move is running

Hint: *You cannot give another BG command until current BG motion has been completed. Use the AM trippoint to wait for motion complete between moves. Another method for checking motion complete is to test for _BG being equal to 0.*

BL (Binary C7)

FUNCTION: Reverse Software Limit

DESCRIPTION:

The BL command sets the reverse software limit. If this limit is exceeding during motion, motion on that axis will decelerate to a stop. Reverse motion beyond this limit is not permitted. The reverse limit is activated at X-1, Y-1, Z-1, W-1. To disable the reverse limit, set X,Y,Z,W to -2147483648. The units are in quadrature counts.

ARGUMENTS: BL x,y,z,w BLX=x BL a,b,c,d,e,f,g,h where

x,y,z,w are signed integers in the range -2147483648 to 2147483647.

-214783648 turns off the reverse limit.

"?" returns the reverse software limit for the specified axes.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	-214783648
Default Format	Position format

OPERAND USAGE:

_BLx contains the value of the reverse software limit for the specified axis.

RELATED COMMANDS:

"FL" on page 207	Forward Limit

EXAMPLES:

#TEST	Test Program
AC 1000000	Acceleration Rate
DC 1000000	Deceleration Rate
BL -15000	Set Reverse Limit
JG -5000	Jog Reverse
BGX	Begin Motion
AMX	After Motion (limit occurred)
TPX	Tell Position
EN	End Program

Hint: *Galil Controllers also provide hardware limits.*

BN (Binary B0)

FUNCTION: Burn

DESCRIPTION:

The BN command saves controller parameters, variables, arrays and applications programs shown below in Flash EEPROM memory. This command typically takes 1 second to execute and must not be interrupted. The controller returns a : when the Burn is complete.

PARAMETERS SAVED DURING BURN:

AC	ER	OP
BL	FL	PF
CB	GA	SB
CE	GR	SP
CN	IL	TL
CO	KD (ZR converted to KD)	TM
CW	KI	VA
DV	KP (GN converted to KP)	VD
DC	MO (MOTOR OFF or ON)	VF
EO	MT	VS
PL	OE	VT

ARGUMENTS: None

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

_BN contains the serial number of the controller.

EXAMPLES:

KD 100	Set damping term for X axis
KP 10	Set proportional gain term for X axis
KI 1	Set integral gain term for X axis
AC 200000	Set acceleration
DC 150000	Set deceleration rate
SP 10000	Set speed
MT -1	Set motor type for X axis to be type '-1', reversed polarity servo motor
MO	Turn motor off

BP (Binary B2)

FUNCTION: Burn Program

DESCRIPTION::

The BP command saves the application program in non-volatile EEPROM memory. This command typically takes up to 10 seconds to execute and must not be interrupted. The controller returns a : when the Burn is complete.

ARGUMENTS: None

USAGE:		**DEFAULTS:**	
While Moving	No	Default Value	---
In a Program	No		
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"BN" on page 177	Burn Parameters
"BV" on page 179	Burn Variable

Note: This command may cause the Galil software to issue the following warning "A time-out occurred while waiting for a response from the controller". This warning is normal and is designed to warn the user when the controller does not respond to a command within the timeout period. This occurs because this command takes more time than the default timeout of 1 sec. The timeout can be changed in the Galil software but this warning does not affect the operation of the controller or software.

BV (Binary B2)

FUNCTION: Burn Variables

DESCRIPTION::

The BV command saves the controller variables in non-volatile EEPROM memory. This command typically takes up to 2 seconds to execute and must not be interrupted. The controller returns a : when the Burn is complete.

ARGUMENTS: None

USAGE:

While Moving	No
In a Program	Yes
Not in a Program	Yes
Can be Interrogated	No
Used in an Operand	No

DEFAULTS:

Default Value	---

RELATED COMMANDS:

“BN” on page 27	Burn Parameters
“BP” on page 29	Burn Program

Note: This command may cause the Galil software to issue the following warning "A time-out occurred while waiting for a response from the controller". This warning is normal and is designed to warn the user when the controller does not respond to a command within the timeout period. This occurs because this command takes more time than the default timeout of 1 sec. The timeout can be changed in the Galil software but this warning does not affect the operation of the controller or software.

CB (Binary 8E)

FUNCTION: Clear Bit

DESCRIPTION:

The CB command sets the specified output bit low. CB can be used to clear the outputs of ext ended I/O which have been configured as outputs.

ARGUMENTS: CB n, where

n is an integer corresponding to the output bit to be cleared. The first output bit is specified as 1.

DPRAM:

The status of the output ports are located at address 02B on the DMC 1310/1340 or 02E-02F on the DMC 1350/1380. Writing to these addresses will change the state of the output ports.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

"SB" on page 261	Set Bit
"OP" on page 245	Define output port (bytewise).

CD (No Binary)

FUNCTION: Contour Data

DESCRIPTION:

The CD command specifies the incremental position on X,Y,Z and W axes. The units of the command are in quadrature counts. This command is used only in the Contour Mode (CM).

ARGUMENTS: CD x,y,z,w CDX=x CD a,b,c,d,e,f,g,h where

x,y,z,w are integers in the range of +/-32762

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

"CM" on page 183	Contour Mode
"WC" on page 295	Wait for Contour
"DT" on page 195	Time Increment
"CS" on page 188	_CS is the Segment Counter

EXAMPLES:

CM XYZW	Specify Contour Mode
DT 4	Specify time increment for contour
CD 200,350,-150,500	Specify incremental positions on X,Y,Z and W axes X-axis moves 200 counts Y-axis moves 350 counts Z-axis moves -150 counts W-axis moves 500 counts
WC	Wait for complete
CD 100,200,300,400	New position data
WC	Wait for complete
DT0	Stop Contour
CD 0,0,0,0	Exit Mode

CE (Binary F2)

FUNCTION: Configure Encoder

DESCRIPTION:

The CE command configures the encoder to the quadrature type or the pulse and direction type. It also allows inverting the polarity of the encoders. The configuration applies independently to the four main axes encoders and the four auxiliary encoders.

ARGUMENTS: CE x,y,z,w CEX=x CE a,b,c,d,e,f,g,h where

x,y,z,w are integers in the range of 0 to 15. Each integer is the sum of two integers n and m which configure the main and the auxiliary encoders. The values of m and n are

M =	MAIN ENCODER TYPE	N =	AUXILIARY ENCODER TYPE
0	Normal quadrature	0	Normal quadrature
1	Normal pulse and direction	4	Normal pulse and direction
2	Reversed quadrature	8	Reversed quadrature
3	Reversed pulse and direction	12	Reversed pulse and direction

For example: x = 6 implies m = 2 and n = 4, both encoders are reversed quadrature.

"?" returns the value of the encoder configuration for the specified axes.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	O
Default Format	2.0

OPERAND USAGE:

_CEx contains the value of encoder type for the axis specified by ‘x’.

RELATED COMMANDS:

"MT" on page 240	Specify motor type

EXAMPLES:

CE 0, 3, 6, 2	Configure encoders
CE ?,?,?,?	Interrogate configuration
V = _CEX	Assign configuration to a variable

Note: *When using pulse and direction encoders, the pulse signal is connected to CHA and the direction signal is connected to CHB.*

CM (Binary D4)

FUNCTION: Contouring Mode

DESCRIPTION:

The Contour Mode is initiated by the instruction CM. This mode allows the generation of an arbitrary motion trajectory with any of the axes. The CD command specified the position increment, and the DT command specifies the time interval.

The command, CM?, can be used to check the status of the Contour Buffer. A value of 1 returned from the command CM? indicates that the Contour Buffer is full. A value of 0 indicates that the Contour Buffer is empty.

ARGUMENTS: CM XYZW CM ABCDEFGH where

the argument specifies the axes to be affected.

CM? returns a 1 if the contour buffer is full and 0 if the contour buffer is empty.

DPRAM:

The contour mode status can be read at bit 5 of address 010 of the General Status and at bit 6 of the Status #2 address in the Axis Buffer.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	0
Default Format	2.0

OPERAND USAGE:

_CM contains a '0' if the contour buffer is empty, otherwise contains a '1'.

RELATED COMMANDS:

"CD" on page 181	Contour Data
"WC" on page 295	Wait for Contour
"DT" on page 195	Time Increment

EXAMPLES:

V=_CM;V=	Return contour buffer status
CM?	Return contour buffer status
CM XZ	Specify X,Z axes for Contour Mode

CN (Binary F3)

FUNCTION: Configure

DESCRIPTION:

The CN command configures the polarity of the limit switches, the home switch and the latch input.

ARGUMENTS: CN m,n,o where

m,n,o are integers with values 1 or -1.

m =	1	Limit switches active high
	-1	Limit switches active low
n =	1	Home switch configured to drive motor in forward direction when input is high. See HM and FE commands.
	-1	Home switch configured to drive motor in reverse direction when input is high. See HM and FE commands
o =	1 *	Latch input is active high
	-1	Latch input is active low

*Note: The latch function will occur within 25usec only when used in active low mode.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	-1.-1.-1.-1
Default Format	2.0

RELATED COMMANDS:

"AL" on page 167	Arm latch

EXAMPLES:

CN 1,1	Sets limit and home switches to active high
CN,, -1	Sets input latch active low

Hint: *To use step motors, connect the 20-pin connector on the DMC-1000 and install the SM jumpers.*

CP (Binary 9E)

FUNCTION: Clear Program

DESCRIPTION:

The CP command clears an application program from the EEPROM memory. This command can take up to 10 seconds to complete.

ARGUMENTS: None

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	No	Default Format	-
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		

RELATED COMMANDS:

"BP" on page …	Burn Program

CR (Binary E1)

FUNCTION: Circle

DESCRIPTION:

The CR command specifies a 2-dimensional arc segment of radius, r, starting at angle, θ, and traversing over angle Δθ. A positive Δθ denotes counterclockwise traverse, negative Δθ denotes clockwise. The VE command must be used to denote the end of the motion sequence after all CR and VP segments are specified. The BG (Begin Sequence) command is used to start the motion sequence. All parameters, r, θ, Δθ, must be specified. Radius units are in quadrature counts. θ and Δθ have units of degrees. The parameter n is optional and describes the vector speed that is attached to the motion segment.

ARGUMENTS: CR r,θ,Δθ < n where

r is an unsigned real number in the range 10 to 6000000 decimal (radius)

θ a signed number in the range 0 to +/-32000 decimal (starting angle in degrees)

Δθ is a signed real number in the range 0.0001 to +/-32000 decimal (angle in degrees)

n specifies a vector speed to be taken into effect at the execution of the vector segment. n is an unsigned even integer between 0 and 8,000,000 for servo motor operation and between 0 and 2,000,000 for stepper motors.

Note: The product r * Δθ must be limited to $+/-4.5\ 10^8$

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

"VP" on page 291	Vector Position
"VS" on page 293	Vector Speed
"VD" on page 287	Vector Deceleration
"VA" on page 286	Vector Acceleration
"VM" on page 289	Vector Mode
"VE" on page 288	End Vector
"BG" on page 174	BGS - Begin Sequence

EXAMPLES:

VMXY	Specify vector motion in the X and Y plane
VS 10000	Specify vector speed
CR 1000,0,360	Generate circle with radius of 1000 counts, start at 0 degrees and complete one circle in counterclockwise direction.
CR 1000,0,360 < 40000	Generate circle with radius of 1000 counts, start at 0 degrees and complete one circle in counterclockwise direction and use a vector speed of 40000.

VE	End Sequence
BGS	Start motion

CS (Binary E2)

FUNCTION: Clear Sequence

DESCRIPTION:

The CS command will remove VP, CR or LI commands stored in a motion sequence. Note, after a sequence has been run, the CS command is not necessary to put in a new sequence. This command is useful when you have incorrectly specified VP, CR or LI commands.

Note: *This command is not valid for single axis controllers..*

ARGUMENTS: None

DPRAM:

Similar to _CS, address 018 and 019 in the Dual Port RAM show which coordinated move segment is currently being run.

USAGE:		**DEFAULTS:**	
While Moving	No	Default Value	---
In a Program	Yes	Default Format	---
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		

OPERAND USAGE:

When used as an operand, _CS contains the number of the segment in the sequence, starting at zero. The operand _CS is valid in the Linear mode, LM, Vector mode, VM, and contour mode, CM.

RELATED COMMANDS:

"CR" on page 185	Circular Interpolation Segment
"LI" on page 229	Linear Interpolation Segment
"LM" on page 231	Linear Interpolation Mode
"VM" on page 289	Vector Mode
"VP" on page 291	Vector Position

EXAMPLES:

#CLEAR	Label
VP 1000,2000	Vector position
VP 4000,8000	Vector position
CS	Clear vectors
VP 1000,5000	New vector
VP 8000,9000	New vector
VE	End Sequence
BGS	Begin sequence
EN	End of Program

CW (No Binary)

FUNCTION: Copyright information / Data Adjustment bit on/off

DESCRIPTION:

The CW command has a dual usage. The CW command will return the copyright information when the argument, n is 0. Otherwise, the CW command is used as a communications enhancement for use by the Servo Design Kit software. When turned on, the communication enhancement sets the MSB of unsolicited, returned ASCII characters to 1. Unsolicited ASCII characters are those characters which are returned from the controller without being directly queried from the terminal. This is the case when a program has a command that requires the controller to return a value or string.

ARGUMENTS: CW n.m where

n is a number, either 0,1, 2 or ?:

0 causes the controller to return the copyright information

1 causes the controller to set the MSB of unsolicited returned characters to 1

2 causes the controller to not set the MSB of unsolicited characters.

? returns the copyright information for the controller.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	2, 0
Default Format	

OPERAND USAGE:

_CW contains the value of the data adjustment bit. 2 = off, 1 = on

Note: *The CW command can cause garbled characters to be returned by the controller. The default state of the controller is to disable the CW command, however, the Galil Servo Design Kit software and terminal software may sometimes enable the CW command for internal usage. If the controller is reset while the Galil software is running, the CW command could be reset to the default value which would create difficulty for the software. It may be necessary to re-enable the CW command. The CW command status can be stored in EEPROM.*

DA (No Binary)

FUNCTION: Deallocate the Variables & Arrays

DESCRIPTION:

The DA command frees the array and/or variable memory space. In this command, more than one array or variable can be specified for deallocation of memories. Different arrays and variables are separated by comma when specified in one command. The argument * deallocates all the variables, and *[0] deallocates all the arrays.

ARGUMENTS: DA c[0],variable-name where

c[0] = Defined array name

variable-name = Defined variable name

* - Deallocates all the variables

*[0] - Deallocates all the arrays

DA ? returns the number of arrays available on the controller.

USAGE:		**DEFAULTS:**
While Moving	Yes	Default Value
In a Program	Yes	Default Format
Command Line	Yes	
Can be Interrogated	Yes	
Used as an Operand	Yes	

OPERAND USAGE:

_DA contains the total number of arrays available. For example, before any arrays have been defined, the operand _DA on a standard DMC-1310 is 14. If an array is defined, the operand _DA will return 13.

CONTROLLER	**NUMBER OF AVAILABLE ARRAYS**
DMC-1310 thru DMC-1340	14
DMC-1350 thru DMC-1380	30
DMC-1310-MX thru DMC-1340-MX	30

RELATED COMMANDS:

"DM" on page 193	Dimension Array

EXAMPLES: ‘Cars’ and ‘Sales’ are arrays and ‘Total’ is a variable.

DM Cars[400],Sales[50]	Dimension 2 arrays
Total=70	Assign 70 to the variable Total
DA Cars[0],Sales[0],Total	Deallocate the 2 arrays & variables
DA*[0]	Deallocate all arrays
DA *,*[0]	Deallocate all variables and all arrays

Note: *Since this command deallocates the spaces and compacts the array spaces in the memory, it is possible that execution of this command may take longer time than 2 ms.*

DC (Binary CD)

FUNCTION: Deceleration

DESCRIPTION:

The Deceleration command (DC) sets the linear deceleration rate of the motors for independent moves such as PR, PA and JG moves. The parameters will be rounded down to the nearest factor of 1024 and have units of counts per second squared.

ARGUMENTS: DC x,y,z,w DCX=x DC a,b,c,d,e,f,g,h where

x,y,z,w are unsigned numbers in the range 1024 to 67107840

"?" returns the deceleration value for the specified axes.

USAGE:

While Moving	Yes*
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	256000
Default Format	8.0

* When moving, the DC command can only be specified while in the jog mode.

OPERAND USAGE:

_DCx contains the deceleration rate for the specified axis.

RELATED COMMANDS:

"AC" on page 163	Acceleration
"PR" on page 247	Position Relative
"PA" on page 246	Position Absolute
"SP" on page 264	Speed
"JG" on page 220	Jog
"BG" on page 174	Begin
"IT" on page 219	Smoothing

EXAMPLES:

PR 10000	Specify position
AC 2000000	Specify acceleration rate
DC 1000000	Specify deceleration rate
SP 5000	Specify slew speed
BG	Begin motion

Note: *The DC command may be changed during the move in JG move, but not in PR or PA move.*

DE (Binary C4)

FUNCTION: Dual (Auxiliary) Encoder Position

DESCRIPTION:

The DE x,y,z,w command defines the position of the auxiliary encoders. The auxiliary encoders may be used for dual-loop applications.

The DE command defines the current motor position when used with stepper motors. DE ? returns the commanded reference position of the motor. The units are in steps.

Note: The auxiliary encoders are not available for the stepper axis or for the axis where output compare is active.

ARGUMENTS: DE x,y,z,w DEX=x DE a,b,c,d,e,f,g,h where

x,y,z,w are signed integers in the range -2147483647 to 2147483648 decimal

"?" returns the position of the auxiliary encoders for the specified axes.

DPRAM:

DE can be read through the Axis Buffer for the corresponding axis, ie. DMC 1340 X-axis is read at addresses 110 - 113 or DMC 1380 X-axis at addresses 210 - 213.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	0,0,0,0
Default Format	Position Format

OPERAND USAGE:

_DEx contains the current position of the specified auxiliary encoder.

EXAMPLES:

DE 0,100,200,400	Set the current auxiliary encoder position to 0,100,200,400 on X,Y,Z and W axes
DE?,?,?,?	Return auxiliary encoder positions
DUALX=_DEX	Assign auxiliary encoder position of X-axis to the variable DUALX

Hint: *Dual encoders are useful when you need an encoder on the motor and on the load. The encoder on the load is typically the auxiliary encoder and is used to verify the true load position. Any error in load position is used to correct the motor position.*

DM (No Binary)

FUNCTION: Dimension

DESCRIPTION:

The DM command defines a single dimensional array with a name and n total elements. The first element of the defined array starts with element number 0 and the last element is at n-1.

ARGUMENTS: DM c[n] where

c is a name of up to eight characters, starting with an uppercase alphabetic character.
n specifies the size of the array (number of array elements).

DM ? returns the number of array elements available.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	---
Default Format	---

OPERAND USAGE:

_DM contains the available array space. For example, before any arrays have been defined, the operand _DM on a standard DMC-1310 will return 1600. If an array of 100 elements is defined, the operand _DM will return 1500.

CONTROLLER	AMT. OF AVAILABLE ARRAY SPACE
DMC-1310 thru DMC-1340	1600 elements
DMC-1350 thru DMC-1380	8000 elements
DMC-1310-MX thru DMC-1340-MX	8000 elements

RELATED COMMANDS:

"DA" on page 190	Deallocate Array

EXAMPLES:

DM Pets[5],Dogs[2],Cats[3]	Define dimension of arrays, pets with 5 elements; Dogs with 2 elements; Cats with 3 elements
DM Tests[1600]	Define dimension of array Tests with 1600 elements

DP (Binary C3)

FUNCTION: Define Position

DESCRIPTION:

The DP command sets the current motor position and current command positions to a user specified value. The units are in quadrature counts.

The DP command sets the commanded reference position for axes configured as steppers. The units are in steps.

ARGUMENTS: DP x,y,z,w DPX=x DP a,b,c,d,e,f,g,h where

x,y,z,w are signed integers in the range -2147483648 to 2147483647 decimal.

"?" returns the current position of the motor for the specified axes.

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	0,0,0,0
Default Format	Position Format

OPERAND USAGE:

_DPx contains the current position of the specified axis.

EXAMPLES:

DP 0,100,200,400	Sets the current position of the X-axis to 0, the Y-axis to 100, the Z-axis to 200, and the W-axis to 400
DP ,-50000	Sets the current position of Y-axis to -50000. The Y,Z and W axes remain unchanged.
DP ?,?,?,?	Interrogate the position of X,Y,Z and W axis.
0000000,-0050000,0000200,0000400	Returns all the motor positions
DP ?	Interrogate the position of X axis
0000000	Returns the X-axis motor position

Hint: *The DP command is useful to redefine the absolute position. For example, you can manually position the motor by hand using the Motor Off command, MO. Turn the servo motors back on with SH and then use DP0 to redefine the new position as your absolute zero.*

DT (No Binary)

FUNCTION: Delta Time

DESCRIPTION:

The DT command sets the time interval for Contouring Mode. Sending the DT command once will set the time interval for all following contour data until a new DT command is sent. 2^n milliseconds is the time interval. Sending DT0 followed by CD0 command terminates the Contour Mode.

ARGUMENTS: DT n where

n is an integer in the range 0 to 8. 0 terminates the Contour Mode. n=1 thru 8 specifies the time interval of 2^n samples.

The default time interval is n=1 or 2 msec for a sample period of 1 msec.

DT ? returns the value for the time interval for contour mode.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	0
Default Format	1.0

OPERAND USAGE:

_DT contains the value for the time interval for Contour Mode

RELATED COMMANDS:

"CM" on page 183	Contour Mode
"CD" on page 181	Contour Data
"WC" on page 295	Wait for next data

EXAMPLES:

DT 4	Specifies time interval to be 16 msec
DT 7	Specifies time interval to be 128 msec
#CONTOUR	Begin
CMXY	Enter Contour Mode
DT 4	Set time interval
CD 1000,2000	Specify data
WC	Wait for contour
CD 2000,4000	New data
WC	Wait
DT0	Stop contour
CD0	Exit Contour Mode
EN	End

DV (Binary F4)

FUNCTION: Dual Velocity (Dual Loop)

DESCRIPTION:

The DV function changes the operation of the filter. It causes the KD (derivative) term to operate on the dual encoder instead of the main encoder. This results in improved stability in the cases where there is a backlash between the motor and the main encoder, and where the dual encoder is mounted on the motor.

ARGUMENTS: DV x,y,z,w where

x,y,z,w may be 0 or 1. 0 disables the function. 1 enables the dual loop.

"?" returns a 0 if dual velocity mode is disabled and 1 if enabled for the specified axes.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	0
Default Format	1.0

OPERAND USAGE:

_DVx contains the state of dual velocity mode for specified axis. 0 = disabled, 1 = enabled.

RELATED COMMANDS:

"KD" on page 223	Damping constant
"FV" on page 208	Velocity feedforward

EXAMPLES:

DV 1,1,1,1	Enables dual loop on all axes
DV 0	Disables DV on X axis
DV,,11	Enables dual loop on Z axis and WX axis. Other axes remain unchanged.
DV 1,0,1,0	Enables dual loop on X and Z axis. Disables dual loop on Y and W axis.

Hint: *The DV command is useful in backlash and resonance compensation.*

ED (Binary 98)

FUNCTION: Edit

DESCRIPTION:

Using Galil COMM 1300 Terminal Software: The ED command puts the controller into the Edit subsystem. In the Edit subsystem, programs can be created, changed, or destroyed. The commands in the Edit subsystem are:

<cntrl>D	Deletes a line
<cntrl>I	Inserts a line before the current one
<cntrl>P	Displays the previous line
<cntrl>Q	Exits the Edit subsystem
<return>	Saves a line

Using your own VME host system: Programs can be created or edited directly by writing ED (Binary 98) to the command buffer. The current program line in the buffer is displayed and can be modified using the following commands:

(9A hex)	Deletes a line
(99 hex)	Inserts a line before the current one
(9B hex)	Displays the previous line
(9C hex)	Exits the Edit subsystem
(9D hex)	Saves a line

USAGE:

Used as an Operand Yes

OPERAND USAGE:

_ED contains the line number of the last line to have an error.

EXAMPLES:

```
ED
000 #START
001 PR 2000
002 BGX
003 SLKJ                          Bad line
004 EN
005 #CMDERR                       Routine which occurs upon a command error
006 V=_ED
007 MG "An error has occurred" {n}
008 MG "In line", V{F3.0}
009 ST
010 ZS0
011 EN
```

Hint: *Remember to quit the Edit Mode prior to executing a program.*

EI (Binary 8C)

FUNCTION: Enable Interrupts

DESCRIPTION:

The EI command enables interrupt conditions such as motion complete or excess error. The conditions are selected by the parameter m where m is the bit mask for the selected conditions as shown below. Prior to using interrupts, jumpers must be placed on the DMC 1300 to select the interrupt priority (IRQ1 - IRQ7) and vector placement (IAD1 - IAD4). The interrupt vector must also be set using the third field of the EI command. An interrupt service routine must be incorporated in your host program. Refer to section 4.3 for more details on the interrupt settings.

ARGUMENTS: EI m,n,o where

EI 0,0 clears the interrupt mask

m is interrupt condition mask

n is input mask

0 is the vector numbered 8 - 255

DPRAM:

The settings for the EI mask, as well as the status of the interrupts, are available in the Dual Port RAM. The interrupt status can be found at address 30 through 33 for the DMC 1310/1340 and address 30 through 35 for the DMC 1350/1380. Below are the addresses for the EI mask.

DMC 1310/1340

Address/Bit #	Condition	Address/Bit #	Condition
034/Bit 7	Inputs (Use n for mask)	035/Bit 7	Contour interrupt
034/Bit 6	Command done	035/Bit 6	NA
034/Bit 5	Application program paused	035/Bit 5	NA
034/Bit 4	NA	035/Bit 4	NA
034/Bit 3	Watchdog timer	035/Bit 3	W axis motion complete
034/Bit 2	Limit switch occurred	035/Bit 2	Z axis motion complete
034/Bit 1	Excess position error	035/Bit 1	Y axis motion complete
034/Bit 0	All axes motion complete	035/Bit 0	X axis motion complete

DMC 1350/1380

Address/Bit #	Condition	Address/Bit #	Condition
036/Bit 6	Command done	039/Bit 7	H axis motion complete
036/Bit 5	Application program stopped	039/Bit 6	G axis motion complete
036/Bit 4	NA	039/Bit 5	F axis motion complete
036/Bit 3	Watchdog timer	039/Bit 4	E axis motion complete
036/Bit 2	Limit switch occurred	039/Bit 3	W axis motion complete

036/Bit 1	Excess position error	039/Bit 2	Z axis motion complete
036/Bit 0	Inputs	039/Bit 1	Y axis motion complete
037/Bit 6	Contour interrupt	039/Bit 0	X axis motion complete
038/Bit 0	All axes motion complete		

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	0
Default Format	---

RELATED COMMANDS:

"UI" on page 285 User interrupt

EXAMPLES:

1. Specify interrupts for all axes motion complete and limit switch with a vector of 8 on a DMC 1340.

   Enable bits 0 and 2 of address 34.

   EI mask will reflect 2 byte value of 05 00 (hex)

   EI 1280,,8

2. Specify interrupt on Input 3 and contour interrupt with a vector of 10 on a DMC 1380.

   Enable bit 0 of address 36 and bit 6 of address 37 on m and bit 2 on n.

   EI mask will reflect 4 byte value of 01 00 40 00 (hex).

   EI 16793600,4,10

Note: The EI command on the DMC 1310/1340 will pass a 2 byte mask, while the EI command for the DMC 1350/1380 will pass a 4 byte mask. Care should be taken to insure that the correct interrupt is set by reading the corresponding interrupt mask register.

EN (Binary 84)

FUNCTION: End

DESCRIPTION:

The EN command is used to designate the end of a program or subroutine. If a subroutine was called by the JS command, the EN command ends the subroutine and returns program flow to the point just after the JS command.

The EN command is used to end the automatic subroutines #MCTIME, #CMDERR, and #COMINT. When the EN command is used to terminate the #COMINT communications interrupt subroutine, there are two arguments; the first determines whether trippoints will be restored upon completion of the subroutine and the second determines whether the communication interrupt will be re-enabled.

ARGUMENTS: EN m, n where

m=0 Return from #COMINT without restoring trippoint

m=1 Return from subroutine and restore trippoint

n=0 Return from #COMINT without restoring interrupt

n=1 Return from communications interrupt #COMINT and restore interrupt

Note1: The default values for the arguments are 0. For example EN,1 and EN0,1 have the same effect.

Note2: Trippoints cause a program to wait for a particular event. The AM command, for example, waits for motion on all axes to complete. If the #COMINT subroutine is executed due to a communication interrupt while the program is waiting for a trippoint, the #COMINT can end by continuing to wait for the trippoint as if nothing happened, or clear the trippoint and continue executing the program at the command just after the trippoint. The EN arguments will specify how the #COMINT routine handles trippoints.

Note3: Use the RE command to return from the interrupt handling subroutines #LIMSWI and #POSERR. Use the RI command to return from the #ININT subroutine.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	n=0, m=0
Default Format	

RELATED COMMANDS:

“RE” on page 252	Return from error subroutine
“RI” on page 253	Return from interrupt subroutine

EXAMPLES:

#A	Program A
PR 500	Move X axis forward 500 counts
BGX	Pause the program until the X axis completes the motion
AMX	Move X axis forward 1000 counts
PR 1000	Set another Position Relative move
BGX	Begin motion
EN	End of Program

Note: *Instead of EN, use the RE command to end the error subroutine and limit subroutine. Use the RI command to end the input interrupt (ININT) subroutine.*

ER (Binary 88)

FUNCTION: Error Limit

DESCRIPTION:

The ER command sets the magnitude of the X,Y,Z and W-axis position errors that will trigger an error condition. When the limit is exceeded, the Error output will go low (true). If the Off On Error (OE1) command is active, the motors will be disabled. The units of ER are quadrature counts.

ARGUMENTS: ER x,y,z,w ERX=x ER a,b,c,d,e,f,g,h where

x,y,z,w are unsigned numbers in the range 1 to 32767

"?" returns the value of the Error limit for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	16384
Default Format	Position Format

OPERAND USAGE:

_ERx contains the value of the Error limit for the specified axis.

RELATED COMMANDS:

"OE" on page 243	Off-On Error
#POSERR	Automatic Error Subroutine

EXAMPLES:

ER 200,300,400,600	Set the X-axis error limit to 200, the Y-axis error limit to 300, the Z-axis error limit to 400, and the W-axis error limit to 600.
ER ,1000	Sets the Y-axis error limit to 1000, leave the X-axis error limit unchanged.
ER ?,?,?,?	Return X,Y,Z and W values
00200,00100,00400,00600	
ER ?	Return X value
00200	
V1=_ERX	Assigns V1 value of ERX
V1=	Returns V1
00200	

Hint: *The error limit specified by ER should be high enough as not to be reached during normal operation. Examples of exceeding the error limit would be a mechanical jam, or a fault in a system component such as encoder or amplifier.*

ES (Binary EB)

FUNCTION: Ellipse Scale

DESCRIPTION:

The ES command divides the resolution of one of the axes in a vector mode. This allows the generation of an ellipse instead of a circle.

The command has two parameters, m and n, (ES m,n), and it applies to the axes designated by the VM command (VMXY, for example). When m>n, the resolution of the first axis (X in the example), will be divided by the ratio m/n. When m<n, the resolution of the second axis (Y in the example), will be divided by n/m. The resolution change applies for the purpose of generating the VP and CR commands. Note that this command results in one axis moving a distance specified by the CR and VP commands while the other one moves a larger distance.

ARGUMENTS: ES m,n where

m and n are positive integers in the range between 1 and 65,535.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	1,1
Default Format	

RELATED COMMANDS:

"VM" on page 289	Vector Mode
"CR" on page 186	Circle move
"VP" on page 291	Vector position

EXAMPLES:

VMXY;ES3,4	Divide Y resolution by 4/3
VMZX;ES2,3	Divide X resolution by 3/2

FA (Binary C1)

FUNCTION: Acceleration Feedforward

DESCRIPTION:

The FA command sets the acceleration feedforward coefficient, or returns the previously set value. This coefficient, when scaled by the acceleration, adds a torque bias voltage during the acceleration phase and subtracts the bias during the deceleration phase of a motion.

Acceleration Feedforward Bias = $FA \cdot AC \cdot 1.5 \cdot 10^{-7}$

Deceleration Feedforward Bias = $FA \cdot DC \cdot 1.5 \cdot 10^{-7}$

The Feedforward Bias product is limited to 10 Volts. FA will only be operational during independent moves.

ARGUMENTS: FA x,y,z,w where

x,y,z,w are unsigned numbers in the range 0 to 8191 decimal with a resolution of 0.25.

"?" returns the value of the feedforward acceleration coefficient for the specified axis.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	4.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		

OPERAND USAGE:

_FAx contains the value of the feedforward acceleration coefficient for the specified axis.

RELATED COMMANDS:

"FV" on page 208	Velocity feedforward

EXAMPLES:

AC 500000,1000000	Set feedforward coefficient to 10 for the X-axis
FA 10,15	and 15 for the Y-axis. The effective bias will be 0.75V for X and 2.25V for Y.
FA ?,?	Return X and Y values
010,015	

Note: *If the feedforward coefficient is changed during a move, then the change will not take effect until the next move.*

FE (Binary D1)

FUNCTION: Find Edge

DESCRIPTION:

The FE command moves a motor until a transition is seen on the homing input for that axis. The direction of motion depends on the initial state of the homing input (use the CN command to configure the polarity of the home input). Once the transition is detected, the motor decelerates to a stop.

This command is useful for creating your own homing sequences.

ARGUMENTS: FE XYZW FE ABCDEFGH where

X,Y,Z,W specify XYZ or W axis. No argument specifies all axes.

DPRAM:

Bit 4 of the Status #1 address in the axis buffer gives the status of the FE command.

USAGE:	**DEFAULTS:**	
While Moving	No	Default Value
In a Program	Yes	Default Format
Command Line	Yes	
Can be Interrogated	No	
Used as an Operand	No	

RELATED COMMANDS:

"FI" on page 206	Find Index
"HM" on page 212	Home
"BG" on page 174	Begin
"AC" on page 163	Acceleration Rate
"DC" on page 191	Deceleration Rate
"SP" on page 264	Speed for search

EXAMPLES:

FE	Set find edge mode
BG	Begin all axes
FEX	Only find edge on X
BGX	
FEY	Only find edge on Y
BGY	
FEZW	Find edge on Z and W
BGZW	

Hint: *Find Edge only searches for a change in state on the Home Input. Use FI (Find Index) to search for the encoder index. Use HM (Home) to search for both the Home input and the Index. Remember to specify BG after each of these commands.*

FI (Binary D6)

FUNCTION: Find Index

DESCRIPTION:

The FI and BG commands move the motor until an encoder index pulse is detected. The controller looks for a transition from low to high. When the transition is detected, motion stops and the position is defined as zero. To improve accuracy, the speed during the search should be specified as 500 counts/s or less. The FI command is useful in custom homing sequences. The direction of motion is specified by the sign of the JG command.

ARGUMENTS: FI XYZW Where

X,Y,Z,W specify XYZ or W axis. No argument specifies all axes.

USAGE:	**DEFAULTS:**	
While Moving	No	Default Value
In a Program	Yes	Default Format
Command Line	Yes	
Can be Interrogated	No	
Used as an Operand	No	

RELATED COMMANDS:

"FE" on page 205	Find Edge
"HM" on page 212	Home
"BG" on page 174	Begin
"AC" on page 163	Acceleration Rate
"DC" on page 191	Deceleration Rate
"SP" on page 264	Search Speed

EXAMPLES:

#HOME	Home Routine
JG 500	Set speed and forward direction
FIX	Find index
BGX	Begin motion
AMX	After motion
MG "FOUND INDEX"	

Hint: *Find Index only searches for a change in state on the Index. Use FE to search for the Home. Use HM (Home) to search for both the Home input and the Index. Remember to specify BG after each of these commands.*

FL (Binary C6)

FUNCTION: Forward Software Limit

DESCRIPTION:

The FL command sets the forward software position limit. If this limit is exceeded during motion, motion on that axis will decelerate to a stop. Forward motion beyond this limit is not permitted. The forward limit is activated at X+1, Y+1, Z+1, W+1. The forward limit is disabled at 2147483647. The units are in counts.

ARGUMENTS: FL x,y,z,w FLX=x FL a,b,c,d,e,f,g,h where

x,y,z,w are signed integers in the range -2147483648 to 2147483647

2147483647 turns off the forward limit

"?" returns the value of the forward limit switch for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	2147483647
Default Format	Position Format

OPERAND USAGE:

_FLx contains the value of the forward limit switch for the specified axis.

RELATED COMMANDS:

"BL" on page 176	Reverse Limit

EXAMPLES:

FL 150000	Set forward limit to 150000 counts on the X-axis
#TEST	Test Program
AC 1000000	Acceleration Rate
DC 1000000	Deceleration Rate
FL 15000	Forward Limit
JG 5000	Jog Forward
BGX	Begin
AMX	After Limit
TPX	Tell Position
EN	End

Hint: *Galil controllers also provide hardware limits.*

FV (Binary C5)

FUNCTION: Velocity Feedforward

DESCRIPTION:

The FV command sets the velocity feedforward coefficient, or returns the previously set value. This coefficient, generates an output bias signal in proportions to the commanded velocity.

Velocity feedforward bias = $1.22 \cdot 10^{-6} \cdot FV \cdot$ Velocity [in ct/s].

For example, if FV=10 and the velocity is 200,000 count/s, the velocity feedforward bias equals 2.44 volts.

ARGUMENTS: FV x,y,z,w FVX=x FV a,b,c,d,e,f,g,h where

x,y,z,w are unsigned numbers in the range 0 to 8191 decimal

"?" returns the feedforward velocity for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	0
Default Format	3.0

OPERAND USAGE:

_FVx contains the feedforward velocity for the specified axis.

RELATED COMMANDS:

"FA" on page 204 Acceleration feedforward

EXAMPLES:

FV 10,20	Set feedforward coefficients to 10 and 20 for x
JG 30000,80000	and y respectively. This produces 0.366 volts for x and 1.95 volts for y.
FV ?,?	Return the x and y values.
010,020	

GA (No Binary)

FUNCTION: Master Axis for Gearing

DESCRIPTION:

The GA command specifies the master axis for electronic gearing. Only one master may be specified. The master may be the main encoder input, auxiliary encoder input, or the commanded position of any axis. The master may also be the commanded vector move in a coordinated motion of LM or VM type. When the master is a simple axis, it may move in any direction and the slave follows. When the master is a commanded vector move, the vector move is considered positive and the slave will move forward if the gear ratio is positive, and backward if the gear ratio is negative. The slave axes and ratios are specified with the GR command and gearing is turned off by the command GR0.

ARGUMENTS: GA n where

n = X or Y or Z or W or A,B,C,D,E,F,G,H for main encoder as axis master

n = CX or CY or CZ or CW or CA,CB,CC,CD,CE,CF,CG,CH for command position as master axis

n = S for vector motion as master

n = DX or DY or DZ or DW or DA,DB,DC,DD,DE,DF,DG,DH for auxiliary encoder as master

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	
Default Format	

RELATED COMMANDS:

"GR" on page 210	Gear Ratio

EXAMPLES FOR DMC-1000 AND DMC-1500:

#GEAR	Gear program
GAX	Specify X axis as master
GR ,.5,-2.5	Specify Y and Z ratios
JG 5000	Specify master jog speed
BGX	Begin motion
WT 10000	Wait 10000 msec
STX	Stop

GN (Binary B8)

FUNCTION: Gain

DESCRIPTION:

The GN command sets the gain of the control loop or returns the previously set value. It fits in the z-transform control equation as follows:

D(z) = GN(z-ZR)/z

ARGUMENTS: GN x,y,z,w GNX=x GN a,b,c,d,e,f,g,h where

x,y,z,w are unsigned integers in the range 0 to 2047 decimal.

"?" returns the value of the gain for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	70
Default Format	4

OPERAND USAGE:

_GNx contains the value of the gain for the specified axis, 'x'.

RELATED COMMANDS:

"ZR" on page 298	Zero
"KI" on page 224	Integrator
"KP" on page 225	Proportional
"KD" on page 223	Derivative

EXAMPLES:

GN 12,14,15,20	Set X-axis gain to 12 Set Y-axis gain to 14 Set Z-axis gain to 15 Set W-axis gain to 20
GN 6	Set X-axis gain to 6 Leave other gains unchanged
GN ,8	Set Y-axis gain to 8 Leave other gains unchanged
GN ?,?,?,?	Returns X,Y,Z,W gains
0006,0008,0015,0020	
GN ?	Returns X gain
0006	
GN ,?	Returns Y gain
0008	

GR (Binary D7)

FUNCTION: Gear Ratio

DESCRIPTION:

GR specifies the Gear Ratios for the geared axes in the electronic gearing mode. The master axis is defined by the GAX or GAY or GAZ or GAW command. The gear ratio may be different for each geared axis and range between +/-127.9999. The slave axis will be geared to the actual position of the master. The master can go in both directions. GR 0,0,0,0 disables gearing for each axis. A limit switch also disables the gearing.

ARGUMENTS: GR x,y,z,w GRX=x GR a,b,c,d,e,f,g,h where

x,y,z,w are signed numbers in the range +/-127, with a fractional resolution of .0001.

0 disables gearing

"?" returns the value of the gear ratio for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	0
Default Format	3.4

OPERAND USAGE:

_GRx contains the value of the gear ratio for the specified axis.

RELATED COMMANDS:

"GA" on page Page …	Master Axis

EXAMPLES:

#GEAR	
MOY	Turn off servo to Y motor
GAY	Specify master axis as Y
GR .25,,-5	Specify X and Z gear ratios
EN	End program

Now when the Y motor is rotated by hand, the X will rotate at 1/4th the speed and Z will rotate 5 times the speed in the opposite direction.

HM (Binary D0)

FUNCTION: Home

DESCRIPTION:

The HM command performs a three-stage homing sequence for servo systems and two stage sequence for stepper motor operation.

For servo motor operation:

The first stage consists of the motor moving at the user programmed speed until detecting a transition on the homing input for that axis. The direction for this first stage is determined by the initial state of the Homing Input. Once the homing input changes state, the motor decelerates to a stop. The state of the homing input can be configured using the CN command.

The second stage consists of the motor changing directions and slowly approaching the transition again. When the transition is detected, the motor is stopped instantaneously..

The third stage consists of the motor slowly moving forward until it detects an index pulse from the encoder. It stops at this point and defines it as position 0.

For stepper mode operation, the sequence consists of the first two stages. The frequency of the motion in stage 2 is 256 cts/ sec.

ARGUMENTS: None

DPRAM:

Bits 1, 2 and 3 of the Status #1 address in the Axis Buffer gives the state of the HM command. Bit 1 shows if home has been found, bit 2 shows if the 1st phase of the homing routine has completed, and bit 3 shows if the 2nd phase of the homing routine has completed.

USAGE:		**DEFAULTS:**
While Moving	No	Default Value
In a Program	Yes	Default Format
Command Line	Yes	
Can be Interrogated	No	
Used as an Operand	Yes	

OPERAND USAGE:

_HMx contains the state of the home switch for the specified axis

RELATED COMMANDS:

"CN" on page 184	Configure Home
"FI" on page 206	Find Index Only
"FE" on page 205	Find Home Only

EXAMPLES:

HM	Set Homing Mode for all axes
BG	Home all axes
BGX	Home only the X-axis
BGY	Home only the Y-axis
BGZ	Home only the Z-axis
BGW	Home only the W-axis

Hint: *You can create your own custom homing sequence by using the FE (Find Home Sensor only) and FI (Find Index only) commands.*

HX (Binary 97)

FUNCTION: Halt Execution

DESCRIPTION:

The HX command halts the execution of any of the four programs that may be running independently in multitasking. The parameter n specifies the program to be halted.

ARGUMENTS: HXn where

n is an integer in the range of 0 to 3 which indicates the thread number.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

DEFAULTS:

Default Value	n = 0
Default Format	

OPERAND USAGE:

When used as an operand, _HXn contains the running status of thread n with:

0	Thread not running
1	Thread is running
2	Thread has stopped at trippoint

RELATED COMMANDS:

"XQ" on page 297	Execute program

EXAMPLES:

XQ #A	Execute program #A, thread zero
XQ #B,3	Execute program #B, thread three
HX0	Halt thread zero
HX3	Halt thread three

II (Binary II)

FUNCTION: Input Interrupt

DESCRIPTION:

The II command enables the interrupt function for the specified inputs. m specifies the beginning input and n specifies the final input in the range. For example, II 2,4 specifies interrupts occurring for Input 2, Input 3 and Input 4. m=0 disables the Input Interrupts. If only the m parameter is given, only that input will generate an interrupt.

The parameter o is an interrupt mask for all eight inputs. If m and n are unused, o contains a number with the mask. A 1 designates that input to be enabled for an interrupt.

Example: II,,5 enables inputs 1 and 3

If any of the specified inputs go low during program execution, the program will jump to the subroutine with label #ININT. Any trippoints set by the program will be cleared but can be re-enabled by the proper termination of the interrupt subroutine using RI. The RI command is used to return from the #ININT routine.

ARGUMENTS: II m,n,o where

m is an integer in the range 0 to 8 decimal

n is an integer in the range 1 to 8 decimal

o is an integer in the range 0 to 255 decimal

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	0
Default Format	3.0 (mask only)

RELATED COMMANDS:

"RI" on page 253	Return from Interrupt
#ININT	Interrupt Subroutine
"AI" on page 166	Trippoint for input

EXAMPLES:

#A	Program A
II 1	Specify interrupt on input 1
JG 5000;BGX	Specify jog and begin motion on X axis
#LOOP;JP #LOOP	Loop
EN	End Program
#ININT	Interrupt subroutine
STX;MG "INTERRUPT"	Stop X, print message
AMX	After stopped
#CLEAR;JP#CLEAR,@IN[1]=0	Check for interrupt clear

```
BGX                     Begin motion
RI0                     Return to main program, don't re-enable trippoints
```

IL (Binary B5)

FUNCTION: Integrator Limit

DESCRIPTION:

The IL command limits the effect of the integrator function in the filter to a certain voltage. For example, IL 2 limits the output of the integrator of the X-axis to the +/-2 Volt range.

A negative parameter also freezes the effect of the integrator during the move. For example, IL -3 limits the integrator output to +/-3V. If, at the start of the motion, the integrator output is 1.6 Volts, that level will be maintained through the move. Note, however, that the KD and KP terms remain active in any case.

ARGUMENTS: IL x,y,z,w ILX=x IL a,b,c,d,e,f,g,h where

x,y,z,w are numbers in the range -9.9988 to 9.9988 Volts with a resolution of 0.0003.

"?" returns the value of the integrator limit for the specified axis.

USAGE:

		DEFAULTS:	
While Moving	Yes	Default Value	9.9988
In a Program	Yes	Default Format	1.4
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		

USAGE:

_ILx contains the value of the integrator limit for the specified axis.

RELATED COMMANDS:

"KI" on page 224	Integrator

EXAMPLES:

KI 2,3,5,8	Integrator constants
IL 3,2,7,2	Integrator limits
IL ?	Returns the X-axis limit
3.0000	

IP (Binary CF)

FUNCTION: Increment Position

DESCRIPTION:

The IP command allows for a change in the command position while the motor is moving. This command does not require a BG. The command has three effects depending on the motion being executed. The units of this are quadrature.

Case 1: Motor is standing still

An IP x,y,z,w command is equivalent to a PR x,y,z,w and BG command. The motor will move to the specified position at the requested slew speed and acceleration.

Case 2: Motor is moving towards specified position

An IP x,y,z,w command will cause the motor to move to a new position target, which is the old target plus x,y,z,w. x,y,z,w must be in the same direction as the existing motion.

Case 3: Motor is in the Jog Mode

An IP x,y,z,w command will cause the motor to instantly try to servo to a position x,y,z,w from the present instantaneous position. The SP and AC parameters have no effect. This command is useful when synchronizing 2 axes in which one of the axis' speed is indeterminate due to a variable diameter pulley.

Warning: When the mode is in jog mode, an IP will create an instantaneous position error. In this mode, the IP should only be used to make incremental position movements.

ARGUMENTS: IP x,y,z,w IPX=x IP a,b,c,d,e,f,g,h where

x,y,z,w are signed numbers in the range -2147483648 to 2147483647 decimal.

"?" returns the current position of the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	No

DEFAULTS:

Default Value	
Default Format	7.0

EXAMPLES:

IP 50	50 counts with set acceleration and speed
#CORRECT	Label
AC 100000	Set acceleration
JG 10000;BGX	Jog at 10000 counts/sec rate
WT 1000	Wait 1000 msec
IP 10	Move the motor 10 counts instantaneously
STX	Stop Motion

IT (Binary BC)

FUNCTION: Independent Time Constant - Smoothing Function

DESCRIPTION:

The IT command filters the acceleration and deceleration functions in independent moves of JG, PR, PA type to produce a smooth velocity profile. The resulting profile, known as S-curve, has continuous acceleration and results in reduced mechanical vibrations. IT sets the bandwidth of the filter where 1 means no filtering and 0.004 means maximum filtering. Note that the filtering results in longer motion time.

ARGUMENTS: IT x,y,z,w ITX=x IT a,b,c,d,e,f,g,h where

x,y,z,w are positive numbers in the range between 0.004 and 1.0 with a resolution of 1/256.

"?" returns the value of the independent time constant for the specified axis.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	
In a Program	Yes	Default Format	7.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		

OPERAND USAGE:

_ITx contains the value of the independent time constant for the specified 'x' axis.

RELATED COMMANDS:

"VT" on page 294	Vector Time Constant for smoothing vector moves

EXAMPLES:

IT 0.8, 0.6, 0.9, 0.1	Set independent time constants for x,y,z,w axes
IT ?	Return independent time constant for X-axis
0.8	

JG (Binary CB)

FUNCTION: Jog

DESCRIPTION:

The JG command sets the jog mode. The parameters following the JG set the slew speed of the axes. Use of the question mark returns the previously entered value or default value. The units of this are counts/second.

ARGUMENTS: JG x,y,z,w JGX=x JG a,b,c,d,e,f,g,h where

x,y,z,w are signed numbers in the range 0 to +/-12,000,000 decimal

For stepper motor operation, the maximum value is 2,000,000 steps/ second.

"?" returns the absolute value of the jog speed for the specified axis.

DPRAM:

A 0 at bit 6 of the Status #1 address in the Axis Buffer indicates the axis is in jog mode. Bit 7 of the Status #2 address will indicate the direction of the jog.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	16385
Default Format	Position Format

OPERAND USAGE:

_JGx contains the absolute value of the jog speed for the specified axis.

RELATED COMMANDS:

"BG" on page 174	Begin
"ST" on page 265	Stop
"AC" on page 163	Acceleration
"DC" on page 191	Deceleration
"IP" on page 218	Increment Position
"TV" on page 283	Tell Velocity

EXAMPLES:

JG 100,500,2000,5000	Set for jog mode with a slew speed of 100 counts/sec for the X-axis, 500 counts/sec for the Y-axis, 2000 counts/sec for the Z-axis, and 5000 counts/sec for W-axis.
BG	Begin Motion
JG ,,-2000	Change the Z-axis to slew in the negative direction at -2000 counts/sec.

JP (No Binary)

FUNCTION: Jump to Program Location

DESCRIPTION:

The JP command causes a jump to a program location on a specified condition. The program location may be any program line number or label. The condition is a conditional statement which uses a logical operator such as equal to or less than. A jump is taken if the specified condition is true.

ARGUMENTS: JP location,condition where

location is a program line number or label

condition is a conditional statement using a logical operator

The logical operators are:

< less than

> greater than

= equal to

<= less than or equal to

>= greater than or equal to

<> not equal to

USAGE:		**DEFAULTS:**	
	While Moving	Yes	Default Value
	In a Program	Yes	Default Format
	Command Line	No	
	Can be Interrogated	No	
	Used as an Operand	No	

EXAMPLES:

JP #POS1,V1<5	Jump to label #POS1 if variable V1 is less than 5
JP #A,V7*V8=0	Jump to #A if V7 times V8 equals 0
JP #B	Jump to #B (no condition)

Hint: *JP is similar to an IF, THEN command. Text to the right of the comma is the condition that must be met for a jump to occur. The destination is the specified label before the comma.*

JS (No Binary)

FUNCTION: Jump to Subroutine

DESCRIPTION:

The JS command will change the sequential order of execution of commands in a program. If the jump is taken, program execution will continue at the line specified by the destination parameter, which can be either a line number or label. The line number of the JS command is saved and after the next EN command is encountered (End of subroutine), program execution will continue with the instruction following the JS command. There can be a JS command within a subroutine.

Note: Subroutines may be nested 16 deep in the standard DMC-1300 controller.

A jump is taken if the specified condition is true. Conditions are tested with logical operators. The logical operators are:

< less than or equal to	<= less than or equal to
> greater than	>= greater than or equal to
= equal to	<> not equal

ARGUMENTS: JS destination, condition where

destination is a line number or label

condition is a conditional statement using a logical operator

USAGE:		**DEFAULTS:**
While Moving	Yes	Default Value
In a Program	Yes	Default Format
Command Line	No	
Can be Interrogated	No	
Used as an Operand	No	

RELATED COMMANDS:

"EN" on page 200	End

EXAMPLES:

JS #SQUARE,V1<5	Jump to subroutine #SQUARE if V1 is less than 5
JS #LOOP,V1<>0	Jump to #LOOP if V1 is not equal to 0
JS #A	Jump to subroutine #A (no condition)

KD (Binary B7)

FUNCTION: Derivative Constant

DESCRIPTION:

KD designates the derivative constant in the controller filter. The filter transfer function is

$D(z) = 4 \cdot KP + 4 \cdot KD(z-1)/z + KIz/2\ (z-1)$

For further details on the filter see the section Theory of Operation.

ARGUMENTS: KD x,y,z,w KDX=x KD a,b,c,d,e,f,g,h where

x,y,z,w are unsigned numbers in the range 0 to 4095.875 with a resolution of 1/8.

"?" returns the value of the derivative constant for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	64
Default Format	4.2

OPERAND USAGE:

_KDx contains the value of the derivative constant for the specified axis.

RELATED COMMANDS:

"KI" on page 224	Integrator
"KP" on page 225	Proportional

EXAMPLES:

KD 100,200,300,400.25	Specify KD
KD ?,?,?,?	Return KD
0100.00,0200.00,0300.00,0400.25	

KI (Binary BA)

FUNCTION: Integrator

DESCRIPTION:

The KI command sets the integral gain of the control loop. It fits in the control equation as follows:

$D(z) = 4 \cdot KP + 4 \cdot KD(z-1)/z + KI\ z/2(z-1)$

The integrator term will reduce the position error at rest to zero.

ARGUMENTS: KI x,y,z,w KIX=x KI a,b,c,d,e,f,g,h where

x,y,z,w are unsigned numbers in the range 0 to 2047.875 with a resolution of 1/8.

"?" returns the value of the derivative constant for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	0
Default Format	4.0

OPERAND USAGE:

_KIx contains the value of the derivative constant for the specified axis.

RELATED COMMANDS:

"KP" on page 225	Proportional Constant
"KI" on page 224	Integrator
"IL" on page 217	Integrator Limit

EXAMPLES:

KI 12,14,16,20	Specify x,y,z,w-axis integral
KI 7	Specify x-axis only
KI ,,8	Specify z-axis only
KI ?,?,?,?	Return X,Y,Z,W
0007,0014,0008,0020	KI values

KP (Binary B6)

FUNCTION: Proportional Constant

DESCRIPTION:

KP designates the proportional constant in the controller filter. The filter transfer function is

$D(z) = 4 \cdot KP + 4 \cdot KD(z-1)/z + KI\ z/2(z-1)$

For further details see the section Theory of Operation.

ARGUMENTS: KP x,y,z,w KPX=x KP a,b,c,d,e,f,g,h where

x,y,z,w are unsigned numbers in the range 0 to 1023.875 with a resolution of 1/8.

"?" returns the value of the proportional constant for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	6
Default Format	4.2

OPERAND USAGE:

_KPx contains the value of the proportional constant for the specified axis.

RELATED COMMANDS:

"KP" on page 225	Proportional Constant
"KI" on page 224	Integrator
"IL" on page 217	Integrator Limit

KS (Binary ?)

FUNCTION: Step Motor Smoothing

DESCRIPTION:

The KS parameter smoothes the frequency of the step motor pulses. Larger values of KS provide greater smoothness. This parameter will also increase the motion time by 3KS sampling periods. KS adds a single pole low pass filter onto the output of the motion profiler. This function smoothes out the generation of step pulses and is most useful when operating in full or half step mode.

Note: KS will delay the step output.

ARGUMENTS: KS x,y,z,w KSX=x KS a,b,c,d,e,f,g,h where

x,y,z,w are positive integers in the range between .5 and 8 with a resolution of 1/32.

"?" returns the value of the derivative constant for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	2
Default Format	4.0

OPERAND USAGE:

_KSx contains the value of the derivative constant for the specified axis.

RELATED COMMANDS:

“MT” on page 221	Motor Type

EXAMPLES:

KS 2, 4 , 8	Specify x,y,z axes
KS 5	Specify x-axis only
KS ,,15	Specify z-axis only

Hint: *KS is valid for step motor only.*

LE (Binary E6)

FUNCTION: Linear Interpolation End

DESCRIPTION:

LE signifies the end of a linear interpolation sequence. It follows the last LI specification in a linear sequence. After the LE specification, the controller issues commands to decelerate the motors to a stop. The VE command is interchangeable with the LE command.

ARGUMENTS:

LE? returns the length of the vector in counts.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		

OPERAND USAGE:

_LE contains the length of the vector in counts.

RELATED COMMANDS:

"LI" on page 229	Linear Distance
"BG" on page 174	BGS - Begin Sequence
"LM" on page 231	Linear Interpolation Mode
"VS" on page 293	Vector Speed
"VA" on page 286	Vector Acceleration
"VD" on page 287	Vector Deceleration

EXAMPLES:

LM ZW	Specify linear interpolation mode
LI ,,100,200	Specify linear distance
LE	End linear move
BGS	Begin motion

_LF* (No Binary)

FUNCTION: Forward Limit Switch Operand (Keyword)

DESCRIPTION:

The _LF operand contains the state of the forward limit switch for the specified axis.

_LFx where x is the specified axis.

DPRAM:

Bit 3 of the Switches address in the Axis Buffer will tell the status of the forward limit switch on an axis, ie. bit 3 of address 105 for the DMC 1340 X-axis forward limit switch and bit 3 of address 205 for the DMC 1380 X-axis forward limit switch.

EXAMPLES:

MG _LF X	Display the status of the X axis forward limit switch

* This is an Operand - Not a command.

LI (Binary E9)

FUNCTION: Linear Interpolation Distance

DESCRIPTION:

The LI x,y,z,w command specifies the incremental distance of travel for each axis in the Linear Interpolation (LM) mode. LI parameters are relative distances given with respect to the current axis positions. Up to 511 LI specifications may be given ahead of the Begin Sequence (BGS) command. Additional LI commands may be sent during motion when the controller sequence buffer frees additional spaces for new vector segments. The Linear End (LE) command must be given after the last LI specification in a sequence. This command tells the controller to decelerate to a stop at the last LI command. It is the responsibility of the user to keep enough LI segments in the controller's sequence buffer to ensure continuous motion.

LM ? returns the available spaces for LI segments that can be sent to the buffer. 511 returned means the buffer is empty and 511 LI segments can be sent. A zero means the buffer is full and no additional segments can be sent. It should be noted that the controller computes the vector speed based on the axes specified in the LM mode. For example, LM XYZ designates linear interpolation for the X,Y and Z axes. The speed of these axes will be computed from $VS^2=XS^2+YS^2+ZS^2$ where XS, YS and ZS are the speed of the X,Y and Z axes. If the LI command specifies only X and Y, the speed of Z will still be used in the vector calculations. The controller always uses the axis specifications from LM, not LI, to compute the speed. The parameter n is optional and can be used to define the vector speed that is attached to the motion segment.

ARGUMENTS: LI x,y,z,w < n LI a,b,c,d,e,f,g,h where

x,y,z,w and a,b,c,d,e,f,h are signed integers in the range -8,388,607 to 8,388,607 and represent incremental move distance

n specifies a vector speed to be taken into effect at the execution of the linear segment. n is an unsigned even integer between 0 and 8,000,000 for servo motor operation and between 0 and 2,000,000 for stepper motors.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		

RELATED COMMANDS:

"LE" on page 227	Linear end
"BG" on page 174	BGS - Begin sequence
"LM" on page 231	Linear Interpolation Mode
"CS" on page 188	Clear Sequence
"VS" on page 293	Vector Speed
"VA" on page 286	Vector Acceleration
"VD" on page 287	Vector Deceleration

EXAMPLES:

LM XYZ	Specify linear interpolation mode
LI 1000,2000,3000	Specify distance
LE	Last segment
BGS	Begin sequence

LM (Binary E8)

FUNCTION: Linear Interpolation Mode

DESCRIPTION:

The LM XYZW command specifies the linear interpolation mode where XYZW denote the axes for linear interpolation. Any set of 1,2,3 or 4 axes may be used for linear interpolation. LI x,y,z,w commands are used to specify the travel distances for linear interpolation. The LE command specifies the end of the linear interpolation sequence. Several LI commands may be given as long as the controller sequence buffer has room for additional segments. Once the LM command has been given, it does not need to be given again unless the VM command has been used.

It should be noted that the controller computes the vector speed based on the axes specified in the LM mode. For example, LM XYZ designates linear interpolation for the X,Y and Z axes. The speed of these axes will be computed from $VS^2=XS^2+YS^2+ZS^2$, where XS, YS and ZS are the speed of the X,Y and Z axes. If the LI command specifies only X and Y, the speed of Z will still be used in the vector calculations. The controller always uses the axis specifications from LM, not LI, to compute the speed.

ARGUMENTS: LM XYZW LM ABCDEFGH where

XYZW denote X,Y,Z or W axes

LM? will return the number of spaces available in the sequence buffer for additional LI commands.

DPRAM:

Bit 0 of the Status #1 address in the Axis Buffer indicates if the controller is in the coordinated motion mode.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

_LM contains the number of spaces available in the sequence buffer for additional LI commands

RELATED COMMANDS:

"LE" on page 227	Linear end
"LI" on page 229	Linear Distance
"VA" on page 286	Vector acceleration
"VS" on page 293	Vector Speed
"VD" on page 287	Vector deceleration
"AV" on page 173	Vector distance
"CS" on page 188	_CS - Sequence counter

EXAMPLES:

LM XYZW	Specify linear interpolation mode
VS 10000; VA 100000;VD 1000000	Specify vector speed, acceleration and deceleration
LI 100,200,300,400	Specify linear distance
LI 200,300,400,500	Specify linear distance
LE; BGS	Last vector, then begin motion

_LR* (Binary ?)

FUNCTION: Reverse Limit Switch Operand (Keyword)

DESCRIPTION:

*The _LR operand contains the state of the reverse limit switch for the specified axis.

_LRx where x is the specified axis.

DPRAM:

Bit 2 of the Switches address in the Axis Buffer will tell the status of the reverse limit switch on an axis, ie. bit 2 of address 105 for the DMC 1340 X-axis reverse limit switch and bit 2 of address 205 for the DMC 1380 X-axis reverse limit switch.

EXAMPLES:

MG _LR X	Display the status of the X axis reverse limit switch

*Note: This is an Operand - Not a command

MC (Binary D8)

FUNCTION: Motion Complete - "In Position"

DESCRIPTION:

The MC command is a trippoint used to control the timing of events. This command will hold up execution of the following commands until the current move on the specified axis or axes is completed and the encoder reaches or passes the specified position. Any combination of axes or a motion sequence may be specified with the MC command. For example, MC XY waits for motion on both the X and Y axis to be complete. MC with no parameter specifies that motion on all axes is complete. TW x,y,z,w sets the timeout to declare an error if the encoder is not in position within the specified time. If a timeout occurs, the trippoint will clear and the stopcode will be set to 99. An application program will jump to the special label #MCTIME.

When used in stepper mode, the controller will hold up execution of the proceeding commands until the controller has generated the same number of steps as specified in the commanded position. The actual number of steps that have been generated can be monitored by using the interrogation command TD. Note: The MC command is useful when operating with stepper motors since the step pulses can be delayed from the commanded position due to the stepper motor smoothing function, KS.

ARGUMENTS: MC XYZW MC ABCDEFGH where

X,Y,Z,W specifies X,Y,Z or W axis or sequence. No argument specifies that motion on all axes is complete.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		

RELATED COMMANDS:

"BG" on page 174	Begin
"AM" on page 168	After Move
"TW" on page 284	Timeout

EXAMPLES:

#MOVE	Program MOVE
PR 5000,5000,5000,5000	Position relative moves
BG X	Start the X-axis
MC X	After the move is complete on X,
BG Y	Start the Y-axis
MC Y	After the move is complete on Y,
BG Z	Start the Z-axis
MC Z	After the move is complete on Z

BG W	Start the W -axis
MC W	After the move is complete on W
EN	End of Program
#F;DP 0,0,0,0	Program F Position
PR 5000,6000,7000,8000	relative moves
BG	Start X,Y,Z and W axes
MC	After motion complete on all axes
MG "DONE"; TP	Print message
EN	End of Program

Hint: *MC can be used to verify that the actual motion has been completed.*

MF (Binary D9)

FUNCTION Forward Motion to Position

DESCRIPTION:

The MF command is a trippoint used to control the timing of events. This command will hold up the execution of the following command until the specified motor moves forward and crosses the position specified. The units of the command are in quadrature counts. Only one axis may be specified at a time. The MF command can also be used when the encoder is the master and not under servo control.

ARGUMENTS: MFx or MF,y or MF,,z or MF,,,w MFX=X MF abcdefgh where

x,y,z,w are signed integers in the range -2147483648 to 2147483647 decimal

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		

RELATED COMMANDS:

"AD" on page 164	Trippoint for after Relative Distances
"AP" on page 169	Trippoint for after Absolute Position

EXAMPLES:

#TEST	Program B
DP0	Define zero
JG 1000	Jog mode (speed of 1000 counts/sec)
BG X	Begin move
MF 2000	After passing the position 2000
V1=_TPX	Assign V1 X position
MG "Position is", V1= ST	Print Message Stop
EN	End of Program

Hint: *The accuracy of the MF command is the number of counts that occur in 2 msec. Multiply the speed by 2 msec to obtain the maximum error. MF tests for absolute position. The MF command can also be used when the specified motor is driven independently by an external device.*

MG (Binary 81)

FUNCTION: Message

DESCRIPTION:

The MG command sends data to the host. This can be used to alert an operator, send instructions or return a variable value. The command can send one ASCII string and one binary value. If the command is sent

ARGUMENTS: MG "m", V where

"m" is a text message including letters, numbers, symbols or <ctrl>G (up to 31 characters).

V is a variable name or array element

Note: Multiple text, variables, and ASCII characters may be used, each must be separated by a comma.

Note: The order of arguments is not important.

DPRAM:

The MG command submitted through the command buffer can be read in the response buffer, with the ASCII string being read in the Y-axis data address and any binary data being read in the X-axis data address. When the MG command is submitted through the program buffer, the response can be read in the program buffer, with the ASCII string read at the Y-axis data address and the binary data being read in the X-axis data address. Data is displayed as 4 bytes of integer with 2 bytes of fraction.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	-
Default Format	Variable Format

EXAMPLES:

Case 1: Message command displays ASCII strings

MG "Good Morning" Displays the string Good Morning

Case 2: Message command displays variables or arrays

MG "The Answer is", Total Displays the string with the content of variable TOTAL in local format of 4 bytes before and 2 bytes after the decimal point.

MO (Binary BD)

FUNCTION: Motor Off

DESCRIPTION:

The MO command shuts off the control algorithm. The controller will continue to monitor the motor position. To turn the motor back on use the Servo Here command (SH).

ARGUMENTS: MO XYZW MO ABCDEFGH where

XYZW specify the axes to be turned off.

"?" returns the state of the motor for the specified axis.

DPRAM:

Bit 0 of the Status #2 address of the Axis Buffer will show a 0 if the servo is in the motor off state.

USAGE:		**DEFAULTS:**	
While Moving	No	Default Value	0
In a Program	Yes	Default Format	1.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		

OPERAND USAGE:

_MOx contains the state of the motor for the specified axis.

RELATED COMMANDS:

"SH" on page 263	Servo Here

EXAMPLES:

MO	Turn off all motors
MOX	Turn off the X motor. Leave the other motors unchanged
MOY	Turn off the Y motor. Leave the other motors unchanged
MOZX	Turn off the Z and X motors. Leave the other motors unchanged
SH	Turn all motors on
Bob=_MOX	Sets Bob equal to the X-axis servo status
Bob=	Return value of Bob. If 1, in motor off mode, If 0, in servo mode

Hint: *The MO command is useful for positioning the motors by hand. Turn them back on with the SH command.*

MR (No Binary)

FUNCTION: Reverse Motion to Position

DESCRIPTION:

The MR command is a trippoint used to control the timing of events. This command will hold up the execution of the following command until the specified motor moves backward and crosses the position specified. The units of the command are in quadrature counts. Only one axis may be specified at a time. The MR command can also be used when the encoder is the master and not under servo control.

ARGUMENTS: MRx or MR,y or MR,,z or MR,,,w MRX=X MR abcdefgh where

x,y,z,w are signed integers in the range -2147483648 to 2147483647 decimal

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	
Default Format	

RELATED COMMANDS:

"AD" on page 164	Trippoint for Relative Distances
"AP" on page 169	Trippoint for after Absolute Position

EXAMPLES:

#TEST	Program B
DP0	Define zero
JG 1000	Jog mode (speed of 1000 counts/sec)
BG X	Begin move
MR -3000	After passing the position -3000
V1=_TPX	Assign V1 X position
MG "Position is", V1= ST	Print Message Stop
EN	End of Program

Hint: *The accuracy of the MR command is the number of counts that occur in 2 msec. Multiply the speed by 2 msec to obtain the maximum error. MR tests for absolute position. The MR command can also be used when the specified motor is driven independently by an external device.*

MT (Binary F5)

FUNCTION: Motor Type

DESCRIPTION:

The MT command selects the type of the motor and the polarity of the drive signal. Motor types include standard servo motors which require a voltage in the range of +/- 10 Volts, and step motors which require pulse and direction signals. The polarity reversal inverts the analog signals for servo motors, and inverts logic level of the pulse train, for step motors.

ARGUMENTS: MT x,y,z,w MTX=x MT a,b,c,d,e,f,g,h where

x,y,z,w are integers with

1 - Servo motor

-1 - Servo motor reversed polarity

2 - Step motor with active low step pulses

-2 - Step motor with active high step pulses

"?" returns the value of the motor type for the specified axis.

DPRAM:

Bit 0 of the Switches address in the Axis Buffer will indicate the stepper motor jumpers are installed for the axis. For example, a 1 at bit 0 of address 105 on a DMC 1340 indicates the SM jumper is installed for the X-axis.

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	1,1,1,1
Default Format	1

OPERAND USAGE:

_MTx contains the value of the motor type for the specified axis.

RELATED COMMANDS:

"CN" on page 184	Configure step pulse width

EXAMPLES:

MT 1,-1,2,2	Configure x as servo, y as reverse servo, z and w as steppers
MT ?,?	Interrogate motor type
V=_MTX	Assign motor type to variable

Hint: *When using step motors, you must install the SM jumper for each axis. The step and direction signals are accessed through the J4 20-pin connector on the controller.*

NO (No Binary)

FUNCTION: No Operation

DESCRIPTION:

The NO command performs no action in a sequence, but can be used as a comment in a program. This helps to document a program.

ARGUMENTS: NO m where

m is any group of letter, number, symbol or <cntrl>G

For DMC 1340: up to 37 characters can follow the NO command

For DMC 1380 or DMC 1340-MX: up to 77 characters can follow the NO command

USAGE:	**DEFAULTS:**	
While Moving	Yes	Default Value
In a Program	Yes	Default Format
Command Line	Yes	
Can be Interrogated	No	
Used as an Operand	No	

EXAMPLES:

#A	Program A
NO	No Operation
NO This Program	No Operation
NO Does Absolutely	No Operation
NO Nothing	No Operation
EN	End of Program

OB (Binary 92)

FUNCTION: Output Bit

DESCRIPTION:

The OB n, logical expression command defines output bit n = 1 through 8 as either 0 or 1 depending on the result from the logical expression. Any non-zero value of the expression results in a one on the output.

ARGUMENTS: OB n, expression where

n denotes the output bit 1 though 8 for the DMC 1310/1340 or 1 through 16 for the DMC 1350/1380 and -MX.

expression is any valid logical expression, variable or array element.

DPRAM:

The status of the output ports are located at address 02B on the DMC 1310/1340 or 02E-02F on the DMC 1350/1380. Writing to these addresses will change the state of the output ports.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	
Default Format	

EXAMPLES:

OB 1, POS 1	If POS 1 is non-zero, Bit 1 is high.
	If POS 1 is zero, Bit 1 is low
OB 2, @IN[1]&@IN[2]	If Input 1 and Input 2 are both high, then
	Output 2 is set high
OB 3, COUNT[1]	If the element 1 in the array is zero, clear bit 3
OB N, COUNT[1]	If element 1 in the array is zero, clear bit N

OE (Binary C0)

FUNCTION: Off on Error

DESCRIPTION:

The OE command causes the controller to shut off the motor command if a position error exceeds the limit specified by the ER command occurs or an abort occurs from either the abort input or on AB command.

If a position error is detected on an axis, and the motion was under an independent move, only that axis will be shut off. However, if the motion is a coordinated mode of the types VM, LM or CM, all the participating axes will be stopped.

ARGUMENTS: OE x,y,z,w OEX=x OE a,b,c,d,e,f,g,h where

the argument may be 0 or 1. 0 disables function. 1 enables off-on-error function.

"?" returns the state of the off an error function for the specified axis.

DPRAM:

The status of the Off/On error can be read at bit 1 of the Status #2 address in the Axis Buffer.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		

OPERAND USAGE:

_OEx contains the status of the off-on-error function for the specified axis. 0 = off, 1 = on

RELATED COMMANDS:

"AB" on page 162	Abort
"ER" on page 202	Error limit
"SH" on page 263	Servo Here
#POSERR	Error Subroutine

EXAMPLES:

OE 1,1,1,1	Enable OE on all axes
OE 0	Disable OE on X-axis other axes remain unchanged
OE ,,1,1	Enable OE on Z-axis and W-axis other axes remain unchanged
OE 1,0,1,0	Enable OE on X and Z-axis Disable OE on Y and W axis

Hint: *The OE command is useful for preventing system damage on excessive error.*

OF (Binary C2)

FUNCTION: Offset

DESCRIPTION:

The OF command sets a bias voltage in the motor command output or returns a previously set value. This can be used to counteract gravity or an offset in an amplifier.

ARGUMENTS: OF x,y,z,w OFX=x OF a,b,c,d,e,f,g,h where

x,y,z,w are signed numbers in the range -9.998 to 9.998 volts with resolution of 0.0003.

"?" returns the offset for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	0
Default Format	1.0

OPERAND USAGE:

_OFx contains the offset for the specified axis.

EXAMPLES:

OF 1,-2,3,5	Set X-axis offset to 1, the Y-axis offset to -2, the Z-axis to 3, and the W-axis to 5
OF -3	Set X-axis offset to -3 Leave other axes unchanged
OF ,0	Set Y-axis offset to 0 Leave other axes unchanged
OF ?,?,?,?	Return offsets
-3.0000,0.0000,3.0000,5.0000	
OF ?	Return X offset
-3.0000	
OF ,?	Return Y offset
0.0000	

OP (Binary 8F)

FUNCTION: Output Port

DESCRIPTION:

The OP command sends data to the output ports of the controller. You can use the output port to control external switches and relays.

The first parameter controls the first output port (bits 1-8) and the second output port (bits 9-16) if the controller has 5 or more axes.

ARGUMENTS: OP m,n where

m is an integer in the range 0 to 65535 decimal, or $0 to FF hexadecimal. (0 to 255 for 4 axes or less). n is an integer in the range 0 to 16772215.

OP ? returns the value of the first argument, m

OP ,? returns the value of the second argument, n.

DPRAM:

The status of the output ports are located at address 02B on the DMC 1310/1340 or 02E-02F on the DMC 1350/1380. Writing to these addresses will change the state of the output ports.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	0
Default Format	3.0

OPERAND USAGE:

_OP0 contains the value of the first argument, m

_OP1 contains the value of the second argument, n.

RELATED COMMANDS:

"SB" on page 261	Set output bit
"CB" on page 180	Clear output bit
"OB" on page 242"	Output Byte

EXAMPLES:

OP 0	Clear Output Port -- all bits
OP $85	Set outputs 1,3,8; clear the others
MG-OP0	Returns the first parameter "m"
MG-OP1	Returns the second parameter "n"

PA (Binary C8)

FUNCTION: Position Absolute

DESCRIPTION:

The PA command will set the final destination of the next move. The position is referenced to the absolute zero. If a ? is used, then the current destination (current command position if not moving, destination if in a move) is returned. For each single move, the largest position move possible is +/-2147483647. Units are in quadrature counts.

ARGUMENTS: PA x,y,z,w PAX=x PA a,b,c,d,e,f,g,h where

x,y,z,w are signed integers in the range -2147483647 to 2147483648 decimal

DPRAM:

Bit 5 of the Status #1 address in the Axis Buffer indicates when the controller is performing a position absolute move. Bit 7 of the Status #2 address will show the direction.

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	-
Default Format	Position Format

OPERAND USAGE:

_PAx contains current destination (current command position if not moving, destination if in a move).

RELATED COMMANDS:

"PR" on page 247	Position relative
"SP" on page 264	Speed
"AC" on page 163	Acceleration
"DC" on page 191	Deceleration
"BG" on page 174	Begin

EXAMPLES:

:PA 400,-600,500,200	X-axis will go to 400 counts Y-axis will go to -600 counts Z-axis will go to 500 counts W-axis will go to 200 counts
:PA ?,?,?,?	Returns the current commanded position
400, -600, 500, 200	
:BG	Start the move
:PA 700	X-axis will go to 700 on the next move while the
:BG	Y,Z and W-axis will travel the previously set relative distance if the preceding move was a PR move, or will not move if the preceding move was a PA move.

PP (No Binary)

FUNCTION: Program Pause

DESCRIPTION:

PP suspends the execution of the application program and sets the appropriate semaphore bit. PP is useful when data needs to be input from a host. The program is resumed when the host clears the appropriate semaphore bit depending on which task thread had been paused.

ARGUMENTS: None

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	No		
Can be Interrogated	No		
Used as an Operand	No		

EXAMPLES:

#A	Label
MG “INPUT SPEED”	Send message
PP	Program Pause - host sends SPEED value and clears semaphore bit to resume
JG SPEED	Jog at input speed
BGX	Begin motion
EN	End program

PR (Binary C9)

FUNCTION: Position Relative

DESCRIPTION:

The PR command sets the incremental distance and direction of the next move. The move is referenced with respect to the current position. If a ? is used, then the current incremental distance is returned (even if it was set by a PA command). Units are in quadrature counts.

ARGUMENTS: PR x,y,z,w PRX=x PR a,b,c,d,e,f,g,h where

x,y,z,w are signed integers in the range -2147483648 to 2147483647 decimal.

"?" returns the current incremental distance for the specified axis.

DPRAM:

Bit 6 of the Status #1 address in the Axis Buffer will show a 1 if the controller is performing a positional move. Bit 7 of the Status #2 address in the Axis Buffer will indicate the direction.

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	No

DEFAULTS:

Default Value	0
Default Format	Position Format

OPERAND USAGE:

_PRx contains the current incremental distance for the specified axis.

RELATED COMMANDS:

"PA" on page 246	Position Absolute
"BG" on page 174	Begin
"AC" on page 163	Acceleration
"DC" on page 191	Deceleration
"SP" on page 264	Speed
"IP" on page 218	Increment Position

EXAMPLES:

:PR 100,200,300,400	On the next move the X-axis will go 100 counts,
:BG	the Y-axis will go to 200 counts forward, Z-axis will go 300 counts and the W-axis will go 400 counts.
:PR ?,?,?	Return relative distances
0000000100,0000000200,0000000300	
:PR 500	Set the relative distance for the X axis to 500
:BG	The X-axis will go 500 counts on the next move while the Y-axis will go its previously set relative distance.

Used as an Operand	No

RA (No Binary)

FUNCTION: Record Array

DESCRIPTION:

The RA command selects one through four arrays for automatic data capture. The selected arrays must be dimensioned by the DM command. The data to be captured is specified by the RD command and time interval by the RC command.

ARGUMENTS: RA n [],m [],o [],p [] RA n[],m[],o[],p[],q[],r[],s[],t[] where

n,m,o and p are dimensioned arrays as defined by DM command. The [] contain nothing.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

"DM" on page 193	Dimension Array
"RD" on page 251	Record Data
"RC" on page 250	Record Interval

EXAMPLES:

#Record	Label
DM POS[100]	Define array
RA POS[]	Specify Record Mode
RD _TPX	Specify data type for record
RC 1	Begin recording at 2 msec intervals
PR 1000;BG	Start motion
EN	End

Hint: *The record array mode is useful for recording the real-time motor position during motion. The data is automatically captured in the background and does not interrupt the program sequencer. The record mode can also be used for a teach or learn of a motion path.*

RC (Binary F0)

FUNCTION: Record

DESCRIPTION:

The RC command begins recording for the Automatic Record Array Mode (RA). RC 0 stops recording.

ARGUMENTS: RC n,m where

n is an integer 1 thru 8 and specifies 2^n samples between records. RC 0 stops recording.

m is optional and specifies the number of records to be recorded. If m is not specified, the DM number will be used. A negative number for m causes circular recording over array addresses 0 to m-1. The address for the array element for the next recording can be interrogated with _RD.

RC? returns status of recording. '1' if recording, '0' if not recording.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

_RC contains status of recording. '1' if recording, '0' if not recording.

RELATED COMMANDS:

"DM" on page 193	Dimension Array
"RD" on page 251	Record Data

EXAMPLES:

#RECORD	Record
DM Torque[1000]	Define Array
RA Torque[]	Specify Record Mode
RD _TTX	Specify Data Type
RC 2	Begin recording and set 4 msec between records
JG 1000;BG	Begin motion
#A;JP #A,_RC=1	Loop until done
MG "DONE RECORDING"	Print message
EN	End program

RD (No Binary)

FUNCTION: Record Data

DESCRIPTION:

The RD command specifies the data type to be captured for the Record Array (RA) mode. The command type includes:

_DEx	2nd encoder
_TPx	Position
_TEx	Position error
_SHx	Commanded position
_RLx	Latched position
_TI	Inputs
_OP	Outputs
_TSx	Switches, only 0-4 bits valid
_SCx	Stop code
_TTx	Tell torque (Note: the values recorded for torque are in the range of +/- 32767 where 0 is 0 torque, -32767 is -10 volt command output, and +32767 is +10 volt.

where 'x' is the axis specifier.

ARGUMENTS: RD _TI,_TPX,_SVZ,_TSY where

The order is important. Each of the four data types correspond with the array specified in the RA command.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

_RD contains the address for the next array element for recording.

RELATED COMMANDS:

"RC" on page 250	Record Interval
"DM" on page 193	Dimension Array

EXAMPLES:

DM ERRORX[50],ERRORY[50]	Define array
RA ERRORX[],ERRORY[ ]	Specify record mode
RD _TEX,_TEYS	Specify data type
RC1	Begin record
JG 1000;BG	Begin motion

RE (No Binary)

FUNCTION: Return from Error Routine

DESCRIPTION:

The RE command is used to end a position error handling subroutine or limit switch handling subroutine. The error handling subroutine begins with the #POSERR label. The limit switch handling subroutine begins with the #LIMSWI. An RE at the end of these routines causes a return to the main program. Care should be taken to be sure the error or limit switch conditions no longer occur to avoid re-entering the subroutines. If the program sequencer was waiting for a trippoint to occur, prior to the error interrupt, the trippoint condition is preserved on the return to the program if RE1 is used. RE0 clears the trippoint. To avoid returning to the main program on an interrupt, use the ZS command to zero the subroutine stack.

ARGUMENTS: RE n where

n = 0 or 1

0 clears the interrupted trippoint

1 restores state of trippoint

USAGE:

While Moving	No
In a Program	Yes
Command Line	No
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

#POSERR	Error Subroutine
#LIMSWI	Limit Subroutine

EXAMPLES:

#A;JP #A;EN	Label for main program
#POSERR	Begin Error Handling Subroutine
MG "ERROR"	Print message
SB1	Set output bit 1
RE	Return to main program and clear trippoint

Hint: *An applications program must be executing for the #LIMSWI and #POSERR subroutines to function.*

RI (No Binary)

FUNCTION: Return from Interrupt Routine

DESCRIPTION:

The RI command is used to end the interrupt subroutine beginning with the label #ININT. An RI at the end of this routine causes a return to the main program. The RI command also re-enables input interrupts. If the program sequencer was interrupted while waiting for a trippoint, such as WT, RI1 restores the trippoint on the return to the program. RI0 clears the trippoint. To avoid returning to the main program on an interrupt, use the command ZS to zero the subroutine stack. This turns the jump subroutine into a jump only.

ARGUMENTS: RI n where

n = 0 or 1

0 clears interrupt trippoint

1 restores trippoint

USAGE:

While Moving	No
In a Program	Yes
Command Line	No
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

#ININT	Input interrupt subroutine
"II" on page 215	Enable input interrupts

EXAMPLES:

#A;II1;JP #A;EN	Program label
#ININT	Begin interrupt subroutine
MG "INPUT INTERRUPT"	Print Message
SB 1	Set output line 1
RI 1	Return to the main program and restore trippoint

Hint: *An applications program must be executing for the #ININT subroutine to function.*

RL (Binary F1)

FUNCTION: Report Latched Position

DESCRIPTION:

The RL command will return the last position captured by the latch. The latch must first be armed by the AL command and then a 0 must occur on the appropriate input. (Input 1,2,3 and 4 for X,Y,Z and W, respectively). The armed state of the latch can be configured using the CN command.

ARGUMENTS: RL XYZW RL ABCDEFGH where

the argument specifies the axes to be affected

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	Position Format
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		

OPERAND USAGE:

_RLx contains the latched position of the specified axis.

RELATED COMMAND:

"AL" on page 167 Arm Latch

EXAMPLES:

JG ,5000	Set up to jog the Y-axis
BGY	Begin jog
ALY	Arm the Y latch; assume that after about 2 seconds, input goes low
RLY	Report the latch
10000	

RM (Binary B1)

FUNCTION: Response Mode

DESCRIPTION:

The RM command sets the communication mode from the program buffer. This command determines what happens if there is an outgoing message in the buffer and another message needs to be sent. Either the new data is lost, the old data is lost or the program execution is suspended until the buffer is read. This command has the same response as writing to 028 hex in the Dual Port RAM.

ARGUMENTS: RM n

n = 0 New data is lost

n = 1 Program execution suspended until buffer is read

n = 2 Old data is lost

DPRAM:

Status of the RM command can be read at address 028. Writing to this address will change the state of the RM command.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	-
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		

RP (No Binary)

FUNCTION: Reference Position

DESCRIPTION:

This command returns the commanded reference position of the motor(s).

ARGUMENTS: RP XYZW RP ABCDEFGH where

the argument specifies the axes to be affected

DPRAM:

The commanded position of an axis can be read in the corresponding Axis Buffer, ie. read addresses 114 - 117 for the DMC 1340 X-axis, addresses 214 - 217 for the DMC 1380 X-axis.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	Position Format
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		

OPERAND USAGE:

_RPx contains the commanded reference position for the specified axis.

RELATED COMMAND:

"TP" on page 278 Tell Position

Note: *The relationship between RP, TP and TE: TEX equals the difference between the reference position, RPX, and the actual position, _TPX.*

EXAMPLES: Assume that XYZ and W axes are commanded to be at the positions 200, -10, 0, -110 respectively. The returned units are in quadrature counts.

:PF 7	Position format of 7
0:RP	
0000200,-0000010,0000000,-0000110	Return X,Y,Z,W reference positions
RPX	
0000200	Return the X motor reference position
RPY	
-0000010	Return the Y motor reference position
PF-6.0	Change to hex format
RP	
$0000C8,$FFFFF6,$000000,$FFFF93	Return X,Y,Z,W in hex
Position=_RPX	Assign the variable, Position, the value of RPX

Hint: *RP command is useful when operating step motors since it provides the commanded position in steps when operating in stepper mode.*

RS (Binary AC)

FUNCTION: Reset

DESCRIPTION:

The RS command resets the state of the processor to its power-on condition. The previously saved state of the controller, along with parameter values, and saved sequences are restored.

USAGE:

While Moving	Yes
In a Program	No
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	0
Default Format	-

<control>R<control>S

FUNCTION: Master Reset

DESCRIPTION:

The Master Reset command resets the controller to factory default settings and erases EEPROM.

A master reset can also be performed by installing a jumper on the controller at the location labeled MRST and resetting the controller (power cycle or pressing the reset button). Remove the jumper after this procedure.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	No	Default Format	-
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		

<control>R<control>V

FUNCTION: Revision Information

DESCRIPTION:

The Revision Information command causes the controller to return firmware revision information.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	No	Default Format	-
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		

SB (Binary 8D)

FUNCTION: Set Bit

DESCRIPTION:

The SB command sets one of eight bits on the output port or one of 16 bits if the controller has 5 or more axes.

ARGUMENTS: SB n where

n is an integer in the range 1 to 8 decimal for 1 - 4 axes.

n is an integer in the range 1 to 16 for 5 or more axes.

DPRAM:

The status of the output ports are located at address 02B on the DMC 1310/1340 or 02E-02F on the DMC 1350/1380. Writing to these addresses will change the state of the output ports.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		

RELATED COMMAND

"CB" on page 180	Clear Bit

EXAMPLES:

SB 5	Set output bit 5
SB 10	Set output bit 10

SC (No Binary)

FUNCTION: Stop Code

DESCRIPTION:

The SC command allows the user to determine why a motor stops. The controller responds with the stop code as follows:

CODE	MEANING	CODE	MEANING
0	Motors are running, independent mode	9	Stopped after Finding Edge (FE)
1	Motors stopped at commanded independent position	10	Stopped after Homing (HM)
2	Decelerating or stopped by FWD limit switches	50	Contour running
3	Decelerating or stopped by REV limit switches	51	Contour Stop
4	Decelerating or stopped by Stop Command (ST)	99	MC timeout
6	Stopped by Abort input	100	Motors are running, vector sequence
7	Stopped by Abort command (AB)	101	Motors stopped at commanded vector
8	Decelerating or stopped by Off-on-Error (OE1)		

ARGUMENTS: SC XYZW SC ABCDEFGH where

the argument specifies the axes to be affected

DPRAM:

The stop code for any given axis can be read in the corresponding Axis Buffer. For example, the SC for a DMC 1340 is read at 104, while the SC for a DMC 1380 is read at 204.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

DEFAULTS:

Default Value	-
Default Format	3.0

OPERAND USAGE:

_SCx contains the value of the stop code for the specified axis.

EXAMPLES:

Tom=_SCW	Assign the Stop Code of W to variable Tom

SH (Binary BB)

FUNCTION: Servo Here

DESCRIPTION:

The SH commands tells the controller to use the current motor position as the command position and to enable servo control here.

This command can be useful when the position of a motor has been manually adjusted following a motor off (MO) command.

ARGUMENTS: SH XYZW SH ABCDEFGH where

the argument specifies the axes to be affected

DPRAM:

Bit 0 of the Status #2 address of the Axis Buffer will show a 1 if the servo is in the servo here state.

USAGE:		**DEFAULTS:**	
While Moving	No	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		

RELATED COMMANDS:

"MO" on page 238	Motor-off

EXAMPLES:

SH	Servo X,Y,Z,W motors
SHX	Only servo the X motor, the Y,Z and W motors remain in its previous state.
SHY	Servo the Y motor; leave the X,Z and W motors unchanged
SHZ	Servo the Z motor; leave the X,Y and W motors unchanged
SHW	Servo the W motor; leave the X,Y and Z motors unchanged

Note: *The SH command changes the coordinate system. Therefore, all position commands given prior to SH, must be repeated. Otherwise, the controller produces incorrect motion.*

SP (Binary CA)

FUNCTION: Speed

DESCRIPTION:

This command sets the slew speed of any or all axes for independent moves, or it will return the previously set value. The parameters input will be rounded down to the nearest factor of 2 and the units of the parameter are in counts per second. Note: Negative values will be interpreted as the absolute value.

ARGUMENTS: SP x,y,z,w SPX=x SP a,b,c,d,e,f,g,h where

x,y,z,w or a,b,c,d,e,f,g,h are unsigned numbers in the range 0 to 8,000,000 for servo motors OR

x,y,z,w or a,b,c,d,e,f,g,h are unsigned numbers in the range 0 to 2,000,000 for stepper motors

"?" returns the speed for the specified axis.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	25000
In a Program	Yes	Default Format	Position Format
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		

OPERAND USAGE:

_SPx contains the speed for the specified axis.

RELATED COMMANDS:

"AC" on page 163	Acceleration
"DC" on page 191	Deceleration
“PA” on page 246	Position Absolute
"PR" on page 247	Position Relation
"BG" on page 174	Begin

EXAMPLES:

PR 2000,3000,4000,5000	Specify x,y,z,w parameter
SP 5000,6000,7000,8000	Specify x,y,z,w speeds
BG	Begin motion of all axes
AM Z	After Z motion is complete

Note: *For vector moves, use the vector speed command (VS) to change the speed. SP is not a "mode" of motion like JOG (JG).*

ST (Binary D2)

FUNCTION: Stop

DESCRIPTION:

The ST command stops motion on the specified axis. Motors will come to a decelerated stop. If ST is given without an axis specification, program execution will stop in addition to XYZW. XYZW specification will not halt program execution.

ARGUMENTS: ST XYZW ST ABCDEFGH where

the argument specifies the axes to be affected

No parameters will stop motion on all axes and stop program.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		

RELATED COMMANDS:

"BG" on page 174	Begin Motion
"AB" on page 162	Abort Motion
"AM" on page 168	Wait for motion end
"DC" on page 191	Deceleration rate

EXAMPLES:

ST X	Stop X-axis motion
ST S	Stop coordinated sequence
ST XYZW	Stop X,Y,Z,W motion
ST	Stop program and XYZW motion
ST SZW	Stop coordinated XY sequence, and Z and W motion

Hint: *Use the after motion complete command, AM, to wait for motion to be stopped.*

TB (No Binary)

FUNCTION: Tell Status Byte

DESCRIPTION:

The TB command returns status information from the controller as a decimal number. Each bit of the status byte denotes the following condition when the bit is set (high):

BIT	STATUS
Bit 7	Controller addressed
Bit 6	Executing program
Bit 5	Contouring
Bit 4	Executing error or limit switch routine
Bit 3	Input interrupt enabled
Bit 2	Executing input interrupt routine
Bit 1	0 (Reserved)
Bit 0	Echo on

ARGUMENTS:

TB ? returns the status byte

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	-
Default Format	1.0

OPERAND USAGE:

_TB Contains the status byte

EXAMPLES:

"TB" on page 266	Tell status information from the controller
65	Executing program and Echo is on ($2^6 + 2^0 = 64 + 1 = 65$)

TC (No Binary)

FUNCTION: Tell Error Code

DESCRIPTION:

The TC command returns a number between 1 and 255. This number is a code that reflects why a command was not accepted by the controller. This command is useful when the controller halts execution of a program at a command or when the response to a command is a question mark. Entering the TC command will provide the user with a code as to the reason. After TC has been read, it is set to zero. TC 1 returns the text message as well as the numeric code.

ARGUMENTS: TC n where

n=0 returns code only

n=1 returns code and message

TC ? returns the error code

CODE	EXPLANATION	CODE	EXPLANATION
1	Unrecognized command	50	Not enough fields
2	Command only valid from program	51	Question mark not valid
3	Command not valid in program	52	Missing " or string too long
4	Operand error	53	Error in {}
5	Input buffer full	54	Question mark part of string
6	Number out of range	55	Missing [or []
7	Command not valid while running	56	Array index invalid or out of range
8	Command not valid when not running	57	Bad function or array
9	Variable error	58	Unrecognized command in a command response (i.e._GNX)
10	Empty program line or undefined label	59	Mismatched parentheses
11	Invalid label or line number	60	Download error - line too long or too many lines
12	Subroutine more than 16 deep	61	Duplicate or bad label
13	JG only valid when running in jog mode	62	Too many labels
14	EEPROM check sum error	65	IN command must have a comma
15	EEPROM write error	66	Array space full
16	IP incorrect sign during position move or IP given during forced deceleration	67	Too many arrays or variables
17	ED, BN and DL not valid while program running	71	IN only valid in task #0
18	Command not valid when	80	Record mode already running

	contouring		
19	Application strand already executing	81	No array or source specified
20	Begin not valid with motor off	82	Undefined Array
21	Begin not valid while running	83	Not a valid number
22	Begin not possible due to Limit Switch	84	Too many elements
24	Begin not valid because no sequence defined	90	Only X Y Z W valid operand
25	Variable not given in IN command	96	SM jumper needs to be installed for stepper motor operation
28	S operand not valid	100	Not valid when running ECAM
29	Not valid during coordinated move	101	Improper index into ET (must be 0-256)
30	Sequence segment too short	102	No master axis defined for ECAM
31	Total move distance in a sequence > 2 billion	103	Master axis modulus greater than 256*EP value
32	More than 511 segments in a sequence	104	Not valid when axis performing ECAM
41	Contouring record range error	105	EB1 command must be given first
42	Contour data being sent too slowly	118	Controller has GL1600 not GL1800
46	Gear axis both master and follower		

DPRAM:

Bit 0 and bit 1 of address 010 in the General Registers indicates if there is an error in either an application program or command from the command buffer. Address 012 of the General Registers will specify which error was generated from the command buffer, while address 013 will specify which error was generated from the application program.

USAGE: **DEFAULTS:**

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	3.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

USAGE:

_TC contains the error code

EXAMPLES:

:GF32	Bad command
?TC	Tell error code

001	Unrecognized command

TD (No Binary)

FUNCTION: Tell Dual Encoder

DESCRIPTION::

This command returns the current position of the dual (auxiliary) encoder(s). Auxiliary encoders are not available for stepper axes or for the axis where output compare is used.

When operating with stepper motors, the TD command returns the number of counts that have been output by the controller.

ARGUMENTS: TD XYZW TD ABCDEFGH where

the argument specifies the axes to be affected

DPRAM:

The auxiliary encoder position for an axis can be read in the corresponding Axis Buffer, ie. addresses 110 through 113 for the DMC 1340 X-axis, or addresses 210 through 213 for the DMC 1380 X-axis.

USAGE:

While Moving	Yes
In a Program	Yes
Not in a Program	Yes
Can be Interrogated	No
Used in an Operand	Yes

DEFAULTS:

Default Value	0
Default Format	Position Format

RELATED COMMANDS:

"DE" on page 192 Dual Encoder

EXAMPLES:

:PF 7	Position format of 7
:TD	Return X,Y,Z,W Dual encoders
0000200,-0000010,0000000,-0000110	
TDX	Return the X motor Dual encoder
0000200	
DUAL=_TDX	Assign the variable, DUAL, the value of TDX

TE (No Binary)

FUNCTION: Tell Error

DESCRIPTION::

This command returns the current position error of the motor(s). The range of possible error is 2147483647. The Tell Error command is not valid for step motors since they operate open-loop.

ARGUMENTS: TE XYZW TE ABCDEFGH where

the argument specifies the axes to be affected

DPRAM:

The position error for an axis can be read in the corresponding Axis Buffer, ie. addresses 10A through 10D for the DMC 1340 X-axis, or addresses 20A through 20D for the DMC 1380 X-axis.

USAGE:

While Moving	Yes
In a Program	Yes
Not in a Program	Yes
Can be Interrogated	No
Used in an Operand	Yes

DEFAULTS:

Default Value	0
Default Format	Position Format

RELATED COMMANDS:

"OE" on page 243	Off On Error
"ER" on page 202	Error Limit
#POSERR	Error Subroutine

EXAMPLES:

TE	Return all position errors
00005,-00002,00000,00006	
TEX	Return the X motor position error
00005	
TEY	Return the Y motor position error
-00002	
Error =_TEX	Sets the variable, Error, with the X-axis position error

Hint: *Under normal operating conditions with servo control, the position error should be small. The position error is typically largest during acceleration.*

TI (Binary E0)

FUNCTION: Tell Inputs

DESCRIPTION:

This command returns the state of the general inputs. TI or TI0 return inputs I1 through I8, TI1 returns I9 through I16 and TI2 returns I17 through I24.

	TI or TI0	TI1	TI2
MSB Bit 7	Input 8	Input 16	Input 24
LSB Bit 6	Input 7	Input 15	Input 23
LSB Bit 5	Input 6	Input 14	Input 22
LSB Bit 4	Input 5	Input 13	Input 21
LSB Bit 3	Input 4	Input 12	Input 20
LSB Bit 2	Input 3	Input 11	Input 19
LSB Bit 1	Input 2	Input 10	Input 18
LSB Bit 0	Input 1	Input 9	Input 17

ARGUMENTS: TIn where

n equals 0, 1 or 2

TI ? returns the status byte of input block 0

DPRAM:

Input status can be read from the Dual Port RAM at address 02A for the DMC 1310/1340 or addresses 02A through 02C for the DMC 1350/1380.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	-
Default Format	1.0

OPERAND USAGE:

_TIn contains the status byte of the input block specified by 'n'. Note that the operand can be masked to return only specified bit information - see section on Bitwise operations.

EXAMPLES:

TI	
08	Input 4 is high, others low
TI	
00	All inputs low
Input=_TI	Sets the variable, Input, with the TI value
TI	
255	All inputs high

TIME*

FUNCTION: Time Operand (Keyword)

DESCRIPTION:

*The TIME operand returns the value of the intenal free running, real time clock. The returned value represents the number of servo loop updates and is based on the TM command. The default value for the TM command is 1000. With this update rate, the operand TIME will increase by 1 count every update of approximately 1000usec. Note that a value of 1000 for the update rate (TM command) will actually set an update rate of 1/1024 seconds. Thus the value returned by the TIME operand will be off by 2.4% of the actual time.

The clock is reset to 0 with a standard reset or a master reset.

The keyword, TIME, does not require an underscore "_" as does the other operands.

USAGE:

Used as an Operand	Yes	Format	TIME

EXAMPLES:

MG TIME	Display the value of the internal clock

TL (Binary BE)

FUNCTION: Torque Limit

DESCRIPTION:

The TL command sets the limit on the motor command output. For example, TL of 5 limits the motor command output to 5 volts. Maximum output of the motor command is 9.998 volts.

ARGUMENTS: TL x,y,z,w TLX=x TL a,b,c,d,e,f,g,h where

x,y,z,w are unsigned numbers in the range 0 to 9.998 volts with resolution of 0.003 volts

"?" returns the value of the torque limit for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	-
Default Format	1.0

OPERAND USAGE:

_TLx contains the value of the torque limit for the specified axis.

EXAMPLES:

TL 1,5,9,7.5	Limit X-axis to 1volt Limit Y-axis to 5 volts Limit Z-axis to 9 volts Limit W-axis to 7.5 volts
TL ?,?,?,?	Return limits
1.0000,5.0000,9.0000,	
7.5000	
TL ?	Return X-axis limit
1.0000	

TM (Binary AE)

FUNCTION: Time

DESCRIPTION:

The TM command sets the sampling period of the control loop. Changing the sampling period will uncalibrate the speed and acceleration parameters. A negative number turns off the internal clock allowing for an external source to be used as the time base. The units of this command are μsec.

ARGUMENTS: TM n where

n is an integer in the range 250 to 20000 decimal with resolution of 125 microseconds. The minimum sample time for the DMC-1310 is 250 μsec; 375 μsec for the DMC-1320; 500 μsec for the DMC-1330; 500 μsec for the DMC-1340; 625 μsec for the DMC-1350; 750 μsec for the DMC-1360; 875 μsec for the DMC-1370; 1000 μsec for the DMC-1380.

"?" returns the value of the sample time.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	-
Default Format	1.0

OPERAND USAGE:

_TM contains the value of the sample time.

EXAMPLES:

TM -1000	Turn off internal clock
TM 2000	Set sample rate to 2000 [EQN "[mu]"]sec (This will cut all speeds in half and all acceleration in fourths)
TM 1000	Return to default sample rate

TN (Binary EC)

FUNCTION: Tangent

DESCRIPTION:

The TN m,n command describes the tangent axis to the coordinated motion path. m is the scale factor in counts/degree of the tangent axis. n is the absolute position of the tangent axis where the tangent axis is aligned with zero degrees in the coordinated motion plane. The tangent axis is specified with the VM n,m,p command where p is the tangent axis. The tangent function is useful for cutting applications where a cutting tool must remain tangent to the part.

ARGUMENTS: TN m,n where

m is the scale factor in counts/degree, in the range between -127 and 127 with a fractional resolution of 0.004

When operating with stepper motors, m is the scale factor in steps / degree

n is the absolute position at which the tangent angle is zero, in the range between +/- $2 \cdot 10^9$

TN ? returns the first position value for the tangent axis.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	--
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		

OPERAND USAGE:

_TN contains the first position value for the tangent axis. This allows the user to correctly position the tangent axis before the motion begins.

RELATED COMMANDS:

"VM" on page 289	Vector mode

EXAMPLES:

VM X,Y,Z	Specify coordinated mode for X and Y-axis; Z-axis is tangent to the motion path
TN 100,50	Specify scale factor as 100 counts/degree and 50 counts at which tangent angle is zero
VP 1000,2000	Specify vector position X,Y
VE	End Vector
BGS	Begin coordinated motion with tangent axis

TP (No Binary)

FUNCTION: Tell Position

DESCRIPTION:

This command returns the current position of the motor(s).

ARGUMENTS: TP XYZW TP ABCDEFGH where

the argument specifies the axes to be affected

DPRAM:

The actual position for an axis can be read in the corresponding Axis Buffer, ie. addresses 106 through 109 for the DMC 1340 X-axis, or addresses 206 through 209 for the DMC 1380 X-axis.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	--
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		

OPERAND USAGE:

_TPx contains the current position value for the specified axis.

EXAMPLES:

Assume the X-axis is at the position 200 (decimal), the Y-axis is at the position -10 (decimal), the Z-axis is at position 0, and the W-axis is at -110 (decimal). The returned parameter units are in quadrature counts.

:PF 7	Position format of 7
:TP	Return X,Y,Z,W positions
0000200,-0000010,0000000,-0000110	
TPX	Return the X motor position
0000200	
TPY	Return the Y motor position
-0000010	
PF-6.0	Change to hex format
TP	Return X,Y,Z,W in hex
$0000C8,$FFFFF6,$000000,$FFFF93	
Position=_TPX	Assign the variable, Position, the value of TPX

TR (Binary AF)

FUNCTION: Trace

DESCRIPTION:

The TR command causes each instruction in a program to be sent out the communications port prior to execution. TR1 enables this function and TR0 disables it. The trace command is useful in debugging programs.

ARGUMENTS: TR n where

n=0 or 1

0 disables function

1 enables function

DPRAM:

Bit 6 of address 010 in the General Registers tells the status of the trace command.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	TR0
Default Format	--

TS (Binary DF)

FUNCTION: Tell Switches

DESCRIPTION:

TS returns status information of the Home switch, Forward Limit switch and Reverse Limit switch, error conditions, motion condition and motor state. The value returned by this command is decimal and represents an 8 bit value (decimal value ranges from 0 to 255). Each bit represents the following status information:

BIT	STATUS
Bit 7	Axis in motion if high
Bit 6	Axis error exceeds error limit if high
Bit 5	X motor off if high
Bit 4	Undefined
Bit 3	Forward Limit X inactive
Bit 2	Reverse Limit X inactive
Bit 1	Home X
Bit 0	Latched

ARGUMENTS: TS XYZW TS ABCDEFGH where

the argument specifies the axes to be affected

DPRAM:

The status bits of this command differ from the switches byte in the Axis Buffers. Refer to the address location and description in Chapter 4.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	3.0
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		

OPERAND USAGE:

_TS contains the current status of the switches.

EXAMPLES:

V1=_TSY	Assigns value of TSY to the variable V1
V1=	Interrogate value of variable V1
015 (returned value)	Decimal value corresponding to bit pattern 00001111
	Y axis not in motion (bit 7 - value of 0)
	Y axis error limit not exceeded (bit 6 value of 0)
	Y axis motor is on (bit 5 value of 0)
	Y axis forward limit is inactive (bit 3 value of 1)
	Y axis reverse limit is inactive (bit 2 value of 1)
	Y axis home switch is high (bit 1 value of 1)
	Y axis latch is not armed (bit 0 value of 1)

TT (No Binary)

FUNCTION: Tell Torque

DESCRIPTION:

The TT command reports the value of the analog output signal, which is a number between -9.998 and 9.998 volts.

ARGUMENTS: TT XYZW TT ABCDEFGH where

the argument specifies the axes to be affected

DPRAM:

The torque output of the controller can be read in the corresponding Axis Buffer. For example, X- axis torque for the DMC 1340 is read at addresses 10E through 10F, while X-axis torque for the DMC 1380 is read at addresses 20E through 20F.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

DEFAULTS:

Default Value	-
Default Format	1.4

OPERAND USAGE:

_TTx contains the value of the torque for the specified axis.

RELATED COMMANDS:

"TL" on page 275	Torque Limit

EXAMPLES:

V1=_TTX	Assigns value of TTX to variable, V1
TTX	Report torque on X
-0.2843	Torque is -.2843 volts

TV (No Binary)

FUNCTION: Tell Velocity

DESCRIPTION:

The TV command returns the actual velocity of the axes in units of quadrature count/s. The value returned includes the sign.

ARGUMENTS: TV XYZW TV ABCDEFGH where

the argument specifies the axes to be affected

DPRAM:

The actual velocity of an axis can be read in the corresponding Axis Buffer, ie. 11C through 11F for the X-axis velocity of the DMC 1340 and 21C through 21F for the X-axis velocity of the DMC 1380.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	7.0
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		

OPERAND USAGE:

_TVx contains the value of the velocity for the specified axis.

EXAMPLES:

VELX=_TVX	Assigns value of X-axis velocity to the variable VELX
TVX	Returns the Y-axis velocity
0003420	

Note: *The TV command is computed using a special averaging filter (over approximately .25 sec). Therefore, TV will return average velocity, not instaneous velocity.*

TW (No Binary)

FUNCTION: Timeout for IN-Position (MC)

DESCRIPTION:

The TW x,y,z,w command sets the timeout in msec to declare an error if the MC command is active and the motor is not at or beyond the actual position within n msec after the completion of the motion profile. If a timeout occurs, then the MC trippoint will clear and the stopcode will be set to 99. An application program will jump to the special label #MCTIME. The RE command should be used to return from the #MCTIME subroutine.

ARGUMENTS: TW x,y,z,w TWX=X TW a,b,c,d,e,f,g,h where

x,y,z,w specifies timeout in msec range 0 to 32767 msec -1 disables the timeout.

"?" returns the timeout in msec for the MC command for the specified axis.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	32766
In a Program	Yes	Default Format	
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		

OPERAND USAGE:

_TWx contains the timeout in msec for the MC command for the specified axis.

RELATED COMMANDS:

"MC" on page 234	Motion Complete trippoint

UI (Binary 8B)

FUNCTION: User Interrupt

DESCRIPTION:

The UI command causes an interrupt on the selected IRQ line. Prior to using interrupts, jumpers must be placed on the controller to select the interrupt priority (IRQ1 - IRQ7) and vector placement (IAD1 - IAD4). An interrupt service routine must be incorporated into the VME host program.

ARGUMENTS: UI n where

n is an integer between 0 and 15.

DPRAM:

The user interrupt status may be read at address 030, bit 4 of the General Registers, while address 033 will show the user interrupt number.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	0
Default Format	-

EXAMPLES:

#I	Label
EI,,8	Enable interrupt vector 8
PR 10000	Position relative
SP 5000	Speed
BGX	Begin motion
AS	Wait for at speed
UI 08	Send interrupt 1
EN	End program

This program sends an interrupt to the selected IRQ line using vector 8. A read at address 030 will show a 01, while a read at address 033 will show a 08.

VA (Binary E3)

FUNCTION: Vector Acceleration

DESCRIPTION:

This command sets the acceleration rate of the vector in a coordinated motion sequence. The parameter input will be rounded down to the nearest factor of 1024. The units of the parameter is counts per second squared.

ARGUMENTS: VA n where

n is an unsigned number in the range 1024 to 68,431,360 decimal.

"?" returns the value of the vector acceleration for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	262144
Default Format	Position Format

OPERAND USAGE:

_VAx contains the value of the vector acceleration for the specified axis.

RELATED COMMANDS:

"VS" on page 293	Vector Speed
"VP" on page 291	Vector Position
"VE" on page 288	End Vector
"CR" on page 186	Circle
"VM" on page 289	Vector Mode
"BG" on page 174	Begin Sequence
"VD" on page 287	Vector Deceleration
"VT" on page 294	Vector smoothing constant - S-curve

EXAMPLES:

VA 1024	Set vector acceleration to 1024 counts/sec^2
VA ?	Return vector acceleration
00001024	
VA 20000	Set vector acceleration
VA ?	
0019456	Return vector acceleration
ACCEL=_VA	Assign variable, ACCEL, the value of VA

VD (Binary E5)

FUNCTION: Vector Deceleration

DESCRIPTION:

This command sets the deceleration rate of the vector in a coordinated motion sequence. The parameter input will be rounded down to the nearest factor of 1024. The units of the parameter is counts per second squared.

ARGUMENTS: VD n where

n is an unsigned number in the range 1024 to 68,431,360 decimal.

"?" returns the value of the vector deceleration for the specified axis.

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	262144
Default Format	Position Format

OPERAND USAGE:

_VDx contains the value of the vector deceleration for the specified axis.

RELATED COMMANDS:

"VA" on page 286	Vector Acceleration
"VS" on page 293	Vector Speed
"VP" on page 291	Vector Position
"CR" on page 186	Circle
"VE" on page 288	Vector End
"VM" on page 289	Vector Mode
"BG" on page 174	Begin Sequence
"VT" on page 294	Smoothing constant - S-curve

EXAMPLES:

#VECTOR	Vector Program Label
VMXY	Specify plane of motion
VA1000000	Vector Acceleration
VD 5000000	Vector Deceleration
VS 2000	Vector Speed
VP 10000, 20000	Vector Position
VE	End Vector
BGS	Begin Sequence

VE (Binary E6)

FUNCTION: Vector Sequence End

DESCRIPTION:

VE is required to specify the end segment of a coordinated move sequence. VE would follow the final VP or CR command in a sequence. VE is equivalent to the LE command.

ARGUMENTS:

VE ? returns the length of the vector in counts.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

_VE contains the length of the vector in counts.

RELATED COMMANDS:

"VM" on page 289	Vector Mode
"VS" on page 293	Vector Speed
"VA" on page 286	Vector Acceleration
"VD" on page 287	Vector Deceleration
"CR" on page 186	Circle
"VP" on page 291	Vector Position
"BG" on page 174	Begin Sequence
"CS" on page 188	Clear Sequence

EXAMPLES:

VM XY	Vector move in XY
VP 1000,2000	Linear segment
CR 0,90,180	Arc segment
VP 0,0	Linear segment
VE	End sequence
BGS	Begin motion

VM (Binary E7)

FUNCTION: Coordinated Motion Mode

DESCRIPTION:

The VM command specifies the coordinated motion mode and the plane of motion. This mode may be specified for motion on any set of two axes.

The motion is specified by the instructions VP and CR, which specify linear and circular segments. Up to 511 segments may be given before the Begin Sequence (BGS) command. Additional segments may be given during the motion when the DMC 1300 buffer frees additional spaces for new segments.

The Vector End (VE) command must be given after the last segment. This tells the controller to decelerate to a stop during the last segment.

It is the responsibility of the user to keep enough motion segments in the buffer to ensure continuous motion. VM ? returns the available spaces for motion segments that can be sent to the buffer.

511 returns means that the buffer is empty and 511 segments may be sent. A zero means that the buffer is full and no additional segments may be sent.

ARGUMENTS: VM nmp where

n and m specifies the plane of vector motion. The parameters can be any two axes of X,Y,Z,W or A,B,C,D,E,F,G,H. The parameter, p, is the tangent axis X,Y,Z,W or A,B,C,D,E,F,G,H. A value of N for the parameter, p, turns off tangent.

Vector Motion can be specified for one axis by specifying the parameter, m, as N. This allows for sinusoidal motion on 1 axis..

DPRAM:

Bit 0 of the Status #1 address in the Axis Buffer indicates if the controller is in the coordinated motion mode.

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	X,Y
Default Format	-

OPERAND USAGE:

_VM contains instantaneous commanded vector velocity.

RELATED COMMANDS:

"VP" on page 291	Vector Position
"VS" on page 293	Vector Speed
"VA" on page 286	Vector Acceleration
"VD" on page 287	Vector Deceleration
"CR" on page 186	Circle
"VE" on page 288	End Vector Sequence
"BG" on page 174	Begin Sequence

"CS" on page 188	Clear Sequence
"CS" on page 188	_CS - Segment counter
"VT" on page 294	Vector smoothing constant -- S-curve
"AV" on page 173	Vector distance

EXAMPLES:

VM X,Y	Specify coordinated mode for X,Y
CR 500,0,180	Specify arc segment
VP 100,200	Specify linear segment
VE	End vector
BGS	Begin sequence

VP (Binary B2)

FUNCTION Vector Position

DESCRIPTION:

The VP command defines the target coordinates of a straight line segment in a 2 axis motion sequence. The axes are chosen by the VM command. The motion starts with the Begin sequence command. The units are in quadrature counts, and are a function of the vector scale factor. For three or four axis linear interpolation, use the LI command.

ARGUMENTS: VP n,m < n where

n and m are signed integers in the range -2147483648 to 2147483647. The length of each segment must be limited to $8 \cdot 10^6$.

n specifies a vector speed to be taken into effect at the execution of the vector segment. n is an unsigned even integer between 0 and 8,000,000 for servo motor operation and between 0 and 2,000,000 for stepper motors.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		

OPERAND USAGE:

_VPx contains the absolute coordinate of the axes at the last intersection along the sequence. For example, during the first motion segment, this instruction returns the coordinate at the start of the sequence. The use as an operand is valid in the linear mode, LM, and in the Vector mode, VM.

RELATED COMMANDS:

"CR" on page 186	Circle
"VM" on page 289	Vector Mode
"VA" on page 286	Vector Acceleration
"VD" on page 287	Vector Deceleration
"VE" on page 288	Vector End
"VS" on page 293	Vector Speed
"BG" on page 174	Begin Sequence
"VT" on page 294	Vector smoothing

EXAMPLES:

#A	Program A
VM X,Y	Specify motion plane
VP 1000,2000	Specify vector position X,Y
CR 1000,0,360	Specify arc
VE	Vector end
VS 2000	Specify vector speed
VA 400000	Specify vector acceleration
BGS	Begin motion sequence
EN	End Program

Hint: *The first vector in a coordinated motion sequence defines the origin for that sequence. All other vectors in the sequence are defined by their endpoints with respect to the start of the move sequence.*

VS (Binary E4)

FUNCTION: Vector Speed

DESCRIPTION:

The VS command specifies the speed of the vector in a coordinated motion sequence in either the LM or VM modes. The parameter input is rounded down to the nearest factor of 2. The units are counts per second. VS may be changed during motion.

Vector Speed can be calculated by taking the square root of the sum of the squared values of speed for each axis specified for vector or linear interpolated motion.

ARGUMENTS: VS n where

n specifies the rate

n is an unsigned number in the range 2 to 8,000,000 decimal for servo motors and 2 to 8,000,000 decimal for stepper motors

VS ? returns the vector speed.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

DEFAULTS:

Default Value	8192
Default Format	-

OPERAND USAGE:

_VS contains the vector speed.

RELATED COMMANDS:

"VA" on page 286	Vector Acceleration
"VP" on page 291	Vector Position
"CR" on page 186	Circle
"LM" on page 231	Linear Interpolation
"VM" on page 289	Vector Mode
"BG" on page 174	Begin Sequence
"VE" on page 288	Vector End

EXAMPLES:

VS 2000	Define vector speed as 2000 counts/sec
VS ?	Return vector speed
002000	

Hint: *Vector speed can be attached to individual vector segments. For more information, see description of VP, CR, and LI commands.*

VT (Binary EA)

FUNCTION: Vector Time Constant - S curve

DESCRIPTION:

The VT command filters the acceleration and deceleration functions in vector moves of VM, LM type to produce a smooth velocity profile. The resulting profile, known as S-curve, has continuous acceleration and results in reduced mechanical vibrations. VT sets the bandwidth of the filter, where 1 means no filtering and 0.004 means maximum filtering. Note that the filtering results in longer motion time.

ARGUMENTS: VT n where

n is a positive number in the range between 0.004 and 1.0, with a resolution of 1/256.

VT ? returns the vector time constant.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	1.0
In a Program	Yes	Default Format	1.4
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		

OPERAND USAGE:

_VT contains the vector time constant.

RELATED COMMANDS:

"IT" on page 219	Independent Time Constant for smoothing independent moves

EXAMPLES:

VT 0.8	Set vector time constant
VT ?	Return vector time constant
0.8	

WC (No Binary)

FUNCTION: Wait for Contour Data

DESCRIPTION:

The WC command acts as a flag in the Contour Mode. After this command is executed, the controller does not receive any new data until the internal contour data buffer is ready to accept new commands. This command prevents the contour data from overwriting on itself in the contour data buffer.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

DEFAULTS:

Default Value	1.0
Default Format	1.4

RELATED COMMANDS:

"CM" on page 183	Contour Mode
"CD" on page 181	Contour Data
"DT" on page 195	Contour Time

EXAMPLES:

CM XYZW	Specify contour mode
DT 4	Specify time increment for contour
CD 200,350,-150,500	Specify incremental position on X,Y,Z and W X-axis moves 200 counts Y-axis moves 300 counts Z-axis moves -150 counts W-axis moves 500 counts
WC	Wait for contour data to complete
CD 100,200,300,400	
WC	Wait for contour data to complete
DT 0	Stop contour
CD 0,0,0,0	Exit mode

WT (Binary A6)

FUNCTION: Wait

DESCRIPTION:

The WT command is a trippoint used to time events. After this command is executed, the controller will wait for the number of samples specified before executing the next command. If the TM command has not been used to change the sample rate from 1 msec, then the units of the Wait command are milliseconds.

ARGUMENTS: WT n where

n is an integer in the range 0 to 2 Billion decimal

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		

EXAMPLES: Assume that 10 seconds after a move is over a relay must be closed.

#A	Program A
PR 50000	Position relative move
BGX	Begin the move
AMX	After the move is over
WT 10000	Wait 10 seconds
SB 0	Turn on relay
EN	End Program

Hint: *To achieve longer wait intervals, just stack multiple WT commands.*

XQ (Binary 82)

FUNCTION: Execute Program

DESCRIPTION:

The XQ command begins execution of a program residing in the program memory of the programs may be executed simultaneously with the DMC-1300.

ARGUMENTS: XQ #A,n XQm,n where

A is a program name of up to seven characters.

m is a line number

n is an integer representing the thread number for multitasking in the range of 0 to 3.

NOTE: The arguments for the command, XQ, are optional. If no arguments are given, the first program in memory will be executed as thread 0.

DPRAM:

Bit 7 of address 010 in the General Registers indicates if an application strand is executing.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

DEFAULTS:

Default Value of n:	0
Default Format	-

OPERAND USAGE:

_XQn contains the current line number of execution for thread n, and -1 if thread n is not running.

RELATED COMMANDS:

"HX" on page 214	Halt execution

EXAMPLES:

XQ #Apple,0	Start execution at label Apple, thread zero
XQ #data,2	Start execution at label data, thread two
XQ 0	Start execution at line 0

Hint: *Don't forget to quit the edit mode first before executing a program!*

ZR (Binary B9)

FUNCTION: Zero

DESCRIPTION:

The ZR command sets the compensating zero in the control loop or returns the previously set value. It fits in the control equation as follows:

$D(z) = GN(z-ZR/z)$

ARGUMENTS: ZR x,y,z,w ZRX=x ZR a,b,c,d,e,f,g,h where

x,y,z,w are unsigned numbers in the range 0 to 1 decimal with a resolution of 1/256.

"?" returns the value of the compensating zero for the specified axis.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	.9143
In a Program	Yes	Default Format	3.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		

OPERAND USAGE:

_ZRx contains the value of the compensating zero for the specified axis.

RELATED COMMANDS:

"GN" on page 210	Gain
"KD" on page …	Derivative
"KP" on page 225	Proportional
"KI" on page 224	Integral Gain

EXAMPLES:

ZR .95,.9,.8,.822	Set X-axis zero to 0.95, Y-axis to 0.9, Z-axis to 0.8, W-axis zero to 0.822
ZR ?,?,?,?	Return all zeroes
0.9527,0.8997,0.7994,0.8244	
ZR ?	Return X zero only
0.9527	
ZR ,?	Return Y zero only
0.8997	

ZS (Binary 83)

FUNCTION: Zero Subroutine Stack

DESCRIPTION:

The ZS command is only valid in an application program and is used to avoid returning from an interrupt (either input or error). ZS alone returns the stack to its original condition. ZS1 adjusts the stack to eliminate one return. This turns the jump to subroutine into a jump. Do not use RI (Return from Interrupt) when using ZS. To re-enable interrupts, you must use II command again.

The status of the stack can be interrogated with the operand _ZSx - see operand usage below.

ARGUMENTS: ZS n where

0 returns stack to original condition

1 eliminates one return on stack

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Can be Interrogated	No
Used as an Operand	Yes

DEFAULTS:

Default Value	0
Default Format	3.0

OPERAND USAGE:

_ZSn contains the stack level for the specified thread where n = 0,1,2 or 3. Note: n can also be specified using X (thread 0), Y(thread 1), Z(thread 2) or W(thread 3) .

EXAMPLES:

II1	Input Interrupt on 1
#A;JP #A;EN	Main program
#ININT	Input Interrupt
MG "INTERRUPT"	Print message
S=_ZS	Interrogate stack
S=	Print stack
ZS	Zero stack
S=_ZS	Interrogate stack
S=	Print stack
EN	End

Appendices

Electrical Specifications

Servo Control

ACMD Amplifier Command:	+/-10 Volts analog signal. Resolution 16-bit DAC or .0003 Volts. 3 mA maximum
A+,A-,B+,B-,IDX+,IDX- Encoder and Auxiliary	TTL compatible, but can accept up to +/-12 Volts. Quadrature phase on CHA,CHB. Can accept single-ended (A+,B+ only) or differential (A+,A-,B+,B-). Maximum A,B edge rate: 8 MHz. Minimum IDX pulse width: 120 nsec.

Stepper Control

Pulse	TTL (0-5 Volts) level at 50% duty cycle. 2,000,000 pulses/sec maximum frequency
Direction	TTL (0-5 Volts)

Input/Output

Uncommitted Inputs, Limits, Home Abort Inputs:	2.2K ohm in series with optoisolator. Requires at least 1 mA for on. Can accept up to 28 Volts without additional series resistor. Above 28 Volts requires additional resistor.
AN[1] thru AN[7] Analog Inputs:	Standard configuration is +/-10 Volt. 12-Bit Analog-to-Digital convertor.
OUT[1] thru OUT[8] Outputs:	TTL.
OUT[9] through OUT [16] Outputs	TTL (only available on controllers with 4 or more axes)
IN[17] through IN[24] Inputs	TTL (only available on controllers with 4 or more axes)

Power

+5V	750 mA
+12V	40 mA
-12V	40mA

Performance Specifications

Minimum Servo Loop Update Time:	DMC-1310 -- 250 µsec
	DMC-1320 -- 375 µsec
	DMC-1330 -- 500 µsec
	DMC-1340 -- 500 µsec
Position Accuracy:	+/-1 quadrature count
Velocity Accuracy:	
Long Term	Phase-locked, better than .005%
Short Term	System dependent
Position Range:	+/-2147483647 counts per move
Velocity Range:	Up to 8,000,000 counts/sec
Velocity Resolution:	2 counts/sec
Motor Command Resolution:	14 Bits or .0012V for DMC 1300, 16 bit or 0.0003 for
	DMC 1300-18
Variable Range:	+/-2 billion
Variable Resolution:	$1 \cdot 10^{-4}$
Array Size:	1600 elements
	8000 elements - DMC-1340-MX and DMC-1380
Program Size:	500 lines x 40 characters
	1000 lines x 80 characters: DMC-1380
	2000 lines x 40 characters: DMC-1340-MX

Connectors for DMC 1300 Main Board

J2 - Main (60 pin IDC)

1 Ground	2 5 Volts
3 Error	4 Reset
5 Limit Common	6 Forward Limit - X
7 Reverse Limit - X	8 Home - X
9 Forward Limit - Y	10 Reverse Limit - Y
11 Home - Y	12 Forward Limit - Z
13 Reverse Limit - Z	14 Home - Z
15 Forward Limit - W	16 Reverse Limit - W
17 Home - W	18 Output 1
19 Input Common	20 Latch X Input 1
21 Latch Y Input 2	22 Latch Z
23 Latch W Input 4	24 Abort input
25 Motor Command X	26 Amp enable X
27 Motor Command Y	28 Amp enable Y
29 Motor Command Z	30 Amp enable Z
31 Motor Command W	32 Amp enable W
33 A+X	34 A-X
35 B+X	36 B-X
37 I+X	38 I-X
39 A+Y	40 A-Y
41 B+Y	42 B-Y
43 I+Y	44 I-Y
45 A+Z	46 A-Z
47 B+Z	48 B-Z
49 I+Z	50 I-Z
51 A+W	52 A-W
53 B+W	54 B-W
55 I+W	56 I-W
57 +12V	58 -12V
59 5V	60 Ground

J5 - General I/O (26 pin IDC)

1 Analog 1	2 Analog 2
3 Analog 3	4 Analog 4
5 Analog 5	6 Analog 6
7 Analog 7	8 Ground
9 5 Volts	10 Output 1
11 Output 2	12 Output 3
13 Output 4	14 Output 5
15 Output 6	16 Output 7
17 Output 8	18 Input 8
19 Input 7	20 Input 6
21 Input 5	22 Input 4 (Latch W)
23 Input 3 (latch Z)	24 Input 2 (Latch Y)
25 Input 1 (latch X)	26 Input Common (Isolated 5 Volts)

J3 - Aux Encoder (20 pin IDC)

1 Sample clock	2 Synch
3 B-Aux W	4 B+Aux W
5 A-Aux W	6 A+Aux W
7 B-Aux Z	8 B+Aux Z
9 A-Aux Z	10 A+Aux Z
11 B-Aux Y	12 B+Aux Y
13 A-Aux Y	14 A+Aux Y
15 B-Aux X	16 B+Aux X
17 A-Aux X	18 A+Aux X
19 5 Volt	20 Ground

J4 - Driver (20 pin IDC)

1 Motor Command X	2 Amp enable X
3 PWM X/STEP X	4 Sign X/DIR X
5 NC	6 Motor Command Y
7 Amp enable Y	8 PWM Y/STEP Y
9 Sign Y/DIR Y	10 NC
11 Motor command Z	12 Amp enable Z
13 PWM Z/STEP Z	14 Sign Z/DIR Z
15 5 Volt	16 Motor command W
17 Amp enable W	18 PWM W/STEP W
19 Sign W/DIR W	20 Ground

J6 - Daughter Board Connector (60 pin)

For use only with a Galil daughter board.

J7 - 10 pin

For test only.

Connectors for Auxiliary Board (Axes E,F,G,H)

JD2 - Main (60 pin IDC)

1 Ground	2 5 Volts
3 N.C.	4 N.C.
5 Limit Common	6 Forward Limit - E
7 Reverse Limit - E	8 Home - E
9 Forward Limit - F	10 Reverse Limit - F
11 Home F	12 Forward Limit - G
13 Reverse Limit - G	14 Home - G
15 Forward Limit - H	16 Reverse Limit - H
17 Home H	18 Output 9
19 Input Common	20 Latch E
21 Latch F	22 Latch G
23 Latch H	24 Input 24
25 Motor Command E	26 Amp enable E
27 Motor Command F	28 Amp enable F
29 Motor Command G	30 Amp enable G
31 Motor Command H	32 Amp enable H
33 Channel A+ E	34 Channel A - E
35 Channel B+ E	36 Channel B- E
37 Channel I+ E	38 Channel I- E
39 Channel A+ F	40 Channel A - F
41 Channel B+ F	42 Channel B- F
43 Channel I+ F	44 Channel I- F
45 Channel A+ G	46 Channel A - G
47 Channel B+ G	48 Channel B- G
49 Channel I+ G	50 Channel I- G
51 Channel A+ H	52 Channel A - H
53 Channel B+ H	54 Channel B- H
55 Channel I+ H	56 Channel I- H
57 +12V	58 -12V
59 5V	60 Ground

NOTE: The ABCD axes and other I/O are located on the main DMC 1300 card

JD5 - I/O (26 pin IDC)

1 Input 17 (TTL)	2 Input 18 (TTL)
3 Input 19 (TTL)	4 Input 20 (TTL)
5 Input 21 (TTL)	6 Input 22 (TTL)
7 Input 23 (TTL)	8 Ground
9 5 Volts	10 Output 9
11 Output 10	12 Output 11
13 Output 12	14 Output 13
15 Output 14	16 Output 15
17 Output 16	18 Input 16
19 Input 15	20 Input 14
21 Input 13	22 Input 12 (Latch H)
23 Input 11 (Latch G)	24 Input 10 (Latch F)
25 Input 9 (Latch E)	26 Input Common (Isolated 5 Volts)

JD3 - 20 pin IDC - Auxiliary Encoders

1 N.C.	2 N.C.
3 Aux. B- H	4 Aux. B+ H
5 Aux. A - H	6 Aux. A+ H
7 Aux. B- G	8 Aux. B+ G
9 Aux. A - G	10 Aux. A+ G
11 Aux. B- F	12 Aux. B+ F
13 Aux. A - F	14 Aux. A+ F
15 Aux. B- E	16 Aux. B+ E
17 Aux. A - E	18 Aux. A+ E
19 5 Volt	20 Ground

JD4 - 20 pin IDC - Amplifiers

1 Motor Command E	2 Amp enable E
3 PWM E/Step E	4 Sign E/Dir E
5 NC	6 Motor Command F
7 Amp enable F	8 PWM F/Step F
9 Sign F/Dir F	10 NC
11 Motor Command G	12 Amp enable G
13 PWM G/Step G	14 Sign G/Dir G
15 5 Volt	16 Motor Command H
17 Amp enable H	18 PWM H/Step H
19 Sign H/Dir H	20 Ground H

JD6 - Daughterboard Connector (60 pin)

Connects to DMC 1300 Main Board, connector J6

Pin-Out Description for DMC 1300

Outputs	
Analog Motor Command	+/- 10 Volt range signal for driving amplifier. In servo mode, motor command output is updated at the controller sample rate. In the motor off mode, this output is held at the OF command level.
Amp Enable	Signal to disable and enable an amplifier. Amp Enable goes low on Abort and OE1.
PWM/STEP OUT	PWM/STEP OUT is used for directly driving power bridges for DC servo motors or for driving step motor amplifiers. For servo motors: If you are using a conventional amplifier that accepts a +/- 10 Volt analog signal, this pin is not used and should be left open. The switching frequency is 33.4 Khz for DMC 1300 and 16.7 Khz for DMC 1300-18 . The PWM output is available in two formats: Inverter and Sign Magnitude. In the Inverter mode, the PWM signal is .2% duty cycle for full negative voltage, 50% for 0 Voltage and 99.8% for full positive voltage. In the Sign Magnitude Mode (Jumper SM), the PWM signal is 0% for 0 Voltage, 99.6% for full voltage and the sign of the Motor Command is available at the sign output.
PWM/STEP OUT	For stepmotors: The STEP OUT pin produces a series of pulses for input to a step motor driver. The pulses may either be low or high. The pulse width is 50%. Upon Reset, the output will be low if the SM jumper is on. If the SM jumper is not on, the output will be Tristate.
Sign/Direction	Used with PWM signal to give the sign of the motor command for servo amplifiers or direction for step mo tors.
Error	The signal goes low when the position error on any axis exceeds the value specified by the error limit command, ER.
Output 1-Output 8 Output 9-Output 16 (DMC-1380 only)	These 8 TTL outputs are uncommitted and may be designated by the user to toggle relays and trigger external events. The output lines are toggled by Set Bit, SB, and Clear Bit, CB, instructions. The OP instruction is used to define the state of all the bits of the Output port.

Inputs

Encoder, A+, B+	Position feedback from incremental encoder with two channels in quadrature, CHA and CHB. The encoder may be analog or TTL. Any resolution encoder may be used as long as the maximum frequency does not exceed 8,000,000 quadrature states/sec. The controller performs quadrature decoding of the encoder signals resulting in a resolution of quadrature counts (4 x encoder cycles). Note: Encoders that produce outputs in the format of pulses and direction may also be used by inputting the pulses into CHA and direction into Channel B and using the CE command to configure this mode.
Encoder Index, I+	Once-Per-Revolution encoder pulse. Used in Homing sequence or Find Index command to define home on an encoder index.
Encoder, A-, B-, I-	Differential inputs from encoder. May be input along with CHA, CHB for noise immunity of encoder signals. The CHA- and CHB- - inputs are optional.
Auxiliary Encoder, Aux A+, Aux B+, Aux I+, Aux A-, Aux B-, Aux I-	Inputs for additional encoder. Used when an encoder on both the motor and the load is required.
Abort	A low input stops commanded motion instantly without a controlled deceleration. Also aborts motion program.
Reset	A low input resets the state of the processor to its power-on condition. The previously saved state of the controller, along with parameter values, and saved sequences are restored.
Forward Limit Switch	When active, inhibits motion in forward direction. Also causes execution of limit switch subroutine, #LIMSWI. The polarity of the limit switch may be set with the CN command.
Reverse Limit Switch	When active, inhibits motion in reverse direction. Also causes execution of limit switch subroutine, #LIMSWI. The polarity of the limit switch may be set with the CN command.
Home Switch	Input for Homing (HM) and Find Edge (FE) instructions. Upon BG following HM or FE, the motor accelerates to slew speed. A transition on this input will cause the motor to decelerate to a stop. The polarity of the Home Switch may be set with the CN command.
Input 1 - Input 8 *Input 9 - Input 16 isolated* *Input 17 - Input 23 - TTL*	Uncommitted inputs. May be defined by the user to trigger events. Inputs are checked with the Conditional Jump instruction and After Input instruction or Input Interrupt. Input 1 is latch X, Input 2 is latch Y, Input 3 is latch Z and Input 4 is latch W if the high speed position latch function is enabled.
Latch	High speed position latch to capture axis position within 20 nano seconds on occurrence of latch signal. AL command arms latch. Input 1 is latch X, Input 2 is latch Y, Input 3 is latch Z and Input 4 is latch W. *Input 9 is latch E, Input 10 is latch F, Input 11 is latch G, Input 12 is latch H.*

Jumper Description for DMC 1300

JUMPER	LABEL	FUNCTION (IF JUMPERED)
JP9	LSCOM	Connect LSCOM to 5V
	INCOM	Connect INCOM to 5V
JP11	A12	Address selection jumpers. Default is no jumpers for base
	A13	address of F0 00
	A14	
	A15	
JP12	IAD4	Interrupt address jumpers This three bit number must equal
	IAD2	the IRQ number selected, ie. IAD2 and IAD4 jumpered for
	IAD1	IRQ6.
JP13	IRQ7	Interrupt request jumpers. One of these must be
	IRQ6	jumpered to enable an interrupt line, and a service routine
	IRQ5	written to the host. In addition, the interrupt address
	IRQ4	jumpers (IAD) must be set and the EI command sent with a
	IRQ3	corresponding vector.
	IRQ2	
	IRQ1	
JP20	SMX	For each axis, the SM jumper selects the SM
	SMY	magnitude mode for servo motors or selects
	SMZ	stepper motors. If you are using stepper
	SMW	motors, SM must always be jumpered. The Analog command is not valid with SM jumpered.
	OPT	Reserved
JP21	MRST	Master Reset enable. Returns controller to factory default settings and erases EEPROM. Requires power-on or RESET to be activated.

Offset Adjustments for DMC 1300

X offset	Used to null ACMD offset for X axis
Y offset	Used to null ACMD offset for Y axis
Z offset	Used to null ACMD offset for Z axis
W offset	Used to null ACMD offset for W axis

Note: These adjustments are made at the Galil factory and should not need adjustment under most applications.

Accessories and Options

DMC-1310	Single Axis Controller
DMC-1320	Two-Axis Controller
DMC-1330	Three-Axis Controller
DMC-1340	Four-Axis Controller
DMC-1350	Five-Axis Controller
DMC-1360	Six-Axis Controller
DMC-1370	Seven-Axis Controller
DMC-1380	Eight-Axis Controller
ICM-1100*	Interface board
AMP-1110	Single axis amplifier
AMP-1120	Two-axis amplifier
AMP-1130	Three-axis amplifier
AMP-1140	Four-axis amplifier
-MX option	Memory expansion option to 2000 lines, 8000 array elements, 254 labels and 254 variables
-AF option	Analog feedback option. Uses analog feedback for servo loop.
N23-54-1000	Servo motor; NEMA 23; 54 oz-in continuous
N34-150-1000	Servo motor; NEMA 34; 150 oz-in, continous
COMM 1300	Terminal emulator for use with DMC 1300 and Bit 3 VME system

ICM-1100 Interconnect Module

The ICM-1100 Interconnect Module provides easy connections between the DMC 1300 series controllers and other system elements, such as amplifiers, encoders, and external switches. The ICM-1100 accepts each DMC 1300 ribbon cable (for J2, J3, J4 and J5) and breaks them into screw-type terminals. Each screw terminal is labeled for quick connection of system elements.

The ICM-1100 is packaged as a circuit board mounted to a metal enclosure. A version of the ICM-1100 is also available with servo amplifiers (see AMP-11X0).

Features

- Breaks out all DMC 1300 ribbon cables into individual screw-type terminals.
- Clearly identifies all terminals
- Provides jumper for connecting limit and input supplies to 5 volt supply from PC.
- Available with on-board servo drives (see AMP-1100).
- 10-pin IDC connectors for encoders.

Specifications

Dimensions	5.7" x 13.4" x 2.4"
Weight	2.2 pounds

AMP/ICM-1100 CONNECTIONS

Screw Terminals			Internal DMC 1300 Connection				
Terminal #	Label	I/O	J2	J3	J4	J5	Description
1	GND		1				Ground
2	ACMDX	O	25		1		X input to servo amp
3	AENX	O	26		2		X amp enable
4	PULSX	O			3		X pulse input for stepper
5	DIRX	O			4		X direction input for stepper
6	ACMDY	O	27		6		Y amp input
7	AENY	O	28		7		Y amp enable
8	PULSY	O			8		Y pulse for stepper
9	DIRY	O			9		Y direction for stepper
10	ACMDZ	O	29		11		Z amp input
11	AENZ	O	30		12		Z amp enable
12	PULSZ	O			13		Z pulse for stepper
13	DIRZ	O			14		Z direction for stepper
14	ACMDW	O	31		16		W amp input
15	AENW	O	32		17		W amp enable
16	PULSW	O			18		W pulse for stepper

17	DIRW	O			19		W direction for stepper
18	AN1	I				1	Analog Input 1
19	AN2	I				2	Analog Input 2
20	AN3	I				3	Analog Input 3
21	AN4	I				4	Analog Input 4
22	AN5	I				5	Analog Input 5
23	AN6	I				6	Analog Input 6
24	AN7	I				7	Analog Input 7
25	GND		1,60	20	20	8	

Terminal #	Label	I/O	J2	J3	J4	J5	Description
26	OUT1	O	18			10	Digital Output 1
27	OUT2	O				11	Digital Output 2
28	OUT3	O				12	Digital Output 3
29	OUT4	O				13	Digital Output 4
30	OUT5	O				14	Digital Output 5
31	OUT6	O				15	Digital Output 6
32	OUT7	O				16	Digital Output 7
33	OUT8	O				17	Digital Output 8
34	INP8	I				18	Uncommitted Input 8
35	INP7	I				19	Uncommitted Input 7
36	INP6	I				20	Uncommitted Input 6
37	INP5	I				21	Uncommitted Input 5
38	INP4/LW	I	23			22	Uncommitted Input 4
39	INP3/LZ	I	22			23	Uncommitted Input 3
40	INP2/LY	I	21			24	Uncommitted Input 2
41	INP1/LX	I	20			25	Uncommitted Input 1
42	INCOM		19			26	Input common
43	GND		1,60	20	20	8	Ground
44	WAB-	I		3			W Auxiliary encoder B-
45	WAB+	I		4			W Auxiliary encoder B+
46	WAA-	I		5			W Auxiliary encoder A-
47	WAA+	I		6			W Auxiliary encoder A+
48	ZAB-	I		7			Z Auxiliary encoder B-
49	ZAB+	I		8			Z Auxiliary encoder B+
50	ZAA-	I		9			Z Auxiliary encoder A-
51	ZAA+	I		10			Z Auxiliary encoder A+
52	YAB-	I		11			Y Auxiliary encoder B-
53	YAB+	I		12			Y Auxiliary encoder B+

54	YAA-	I		13			Y Auxiliary encoder A-
55	YAA+	I		14			Y Auxiliary encoder A+
56	XAB-	I		15			X Auxiliary encoder B-
57	XAB+	I		16			X Auxiliary encoder B+
58	XAA-	I		17			X Auxiliary encoder A-
59	XAA+	I		18			X Auxiliary encoder A+
60	GND		1,60	20	20	8	Ground
61	5V		2,59	19	15	9	5 Volts
62	LSCOM		5				X Limit common
63	FLSX	I	6				X Forward limit
64	RLSX	I	7				X Reverse limit

Terminal #	Label	I/O	J2	J3	J4	J5	Description
65	HOMEX	I	8				X Home Input
66	FLSY	I	9				Y Forward limit
67	RLSY	I	10				Y Reverse limit
68	HOMEY	I	11				Y Home
69	FLSZ	I	12				Z Forward limit
70	RLSZ	I	13				Z Reverse limit
71	HOMEZ	I	14				Z Home
72	FLSW	I	15				W Forward limit
73	RLSW	I	16				W Reverse limit
74	HOMEW	I	17				W Home
75	GND		1,60	20	20	8	Ground
76	ABORT	I	24				Abort input
77	XA+	I	33				X Main encoder A+
78	XA-	I	34				X Main encoder A-
79	XB+	I	35				X Main encoder B+
80	XB-	I	36				X Main encoder B-
81	XI+	I	37				X Main encoder I+
82	XI-	I	38				X Main encoder I-
83	YA+	I	39				Y Main encoder A+
84	YA-	I	40				Y Main encoder A-
85	YB+	I	41				Y Main encoder B+
86	YB-	I	42				Y Main encoder B-
87	YI+	I	43				Y Main encoder I+
88	YI-	I	44				Y Main encoder I-
89	ZA+	I	45				Z Main encoder A+
90	ZA-	I	46				Z Main encoder A-
91	ZB+	I	47				Z Main encoder B+
92	ZB-	I	48				Z Main encoder B-
93	ZI+	I	49				Z Main encoder I+
94	ZI-	I	50				Z Main encoder I-
95	WA+	I	51				W Main encoder A+
96	WA-	I	52				W Main encoder A-
97	WB+	I	53				W Main encoder B+
98	WB-	I	54				W Main encoder B-
99	WI+	I	55				W Main encoder I+
100	WI-	I	56				W Main encoder I-

Terminal #	Label	I/O	J2	J3	J4	J5	Description
101	+12V		57				
102	-12V		58				
103	5V		2,59	19	15	9	
104	GND		1,60	20	20	8	

J2 - Main (60 pin IDC)

J3 - Aux Encoder (20 pin IDC)

J4 - Driver (20 pin IDC)

J5 - General I/O (26 pin IDC)

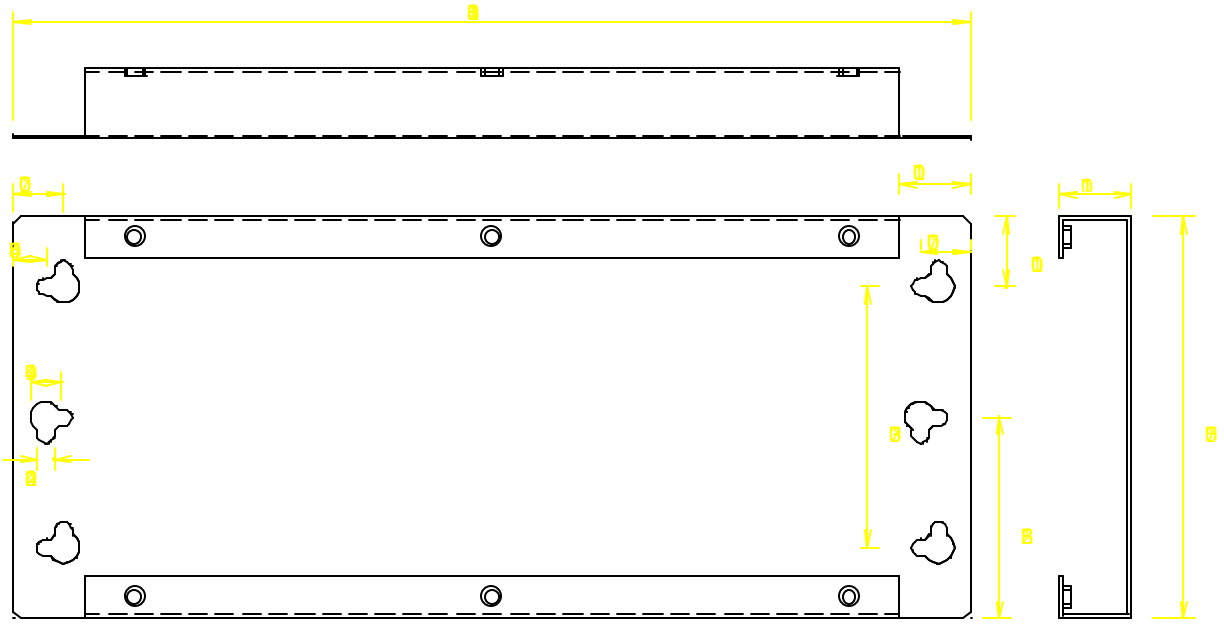
Connectors are the same as described in section entitled "Connectors for DMC 1300 Main Board". see pg. 303

JX6, JY6, JZ6, JW6 - Encoder Input (10 pin IDC)

1 CHA	2 +VCC
3 GND	4 No Connection
5 CHA -	6 CHA
7 CHB -	8 CHB
9 INDEX -	10 INDEX

CAUTION: The ICM-1100 10-pin connectors are designed for the N23 and N34 encoders from Galil. If you are using Galil's Motor-5-500, Motor-50-1000 or Motor-500-1000, you must cut encoder wires 5, 6, 7 and 9.

ICM-1100 Drawing



AMP-11x0 Mating Power Amplifiers

The AMP-11X0 series are mating, brush-type servo amplifiers for the DMC 1300. The AMP-1110 contains one amplifier; the AMP-1120, two amplifiers; the AMP-1130, three; and the AMP-1140, four. Each amplifier is rated for 7 amps continuous, 10 amps peak at up to 80 volts. The gain of the AMP-11X0 is 1 amp/volt. The AMP-11X0 requires an external DC supply. The AMP-11X0 connects directly to the DMC 1300 ribbon connectors, and screw-type terminals are provided for connection to motors, encoders and external switches.

Features

- 6 amps continuous, 10 amps peak; 20 to 80 volts.
- Available with 1, 2, 3, or 4 amplifiers.
- Connects directly to DMC 1300 series controllers via ribbon cables.
- Screw-type terminals for easy connection to motors, encoders and switches.
- Steel mounting plate with 1/4" keyholes.

Specifications

Minimum motor inductance:	1 mH
PWM frequency	30 KHz
Ambient operating temperature	0-70° C
Dimensions	5.7" x 13.4" x 2.5"
Weight	4 pounds
Mounting	Keyholes - 1/4Φ
Gain	1 amp/volt

Coordinated Motion - Mathematical Analysis

The terms of coordinated motion are best explained in terms of the vector motion. The vector velocity, Vs, which is also known as the feed rate, is the vector sum of the velocities along the X and Y axes, Vx and Vy.

$$Vs = \sqrt{Vx^2 + Vy^2}$$

The vector distance is the integral of Vs, or the total distance traveled along the path. To illustrate this further, suppose that a string was placed along the path in the X-Y plane. The length of that string represents the distance traveled by the vector motion.

The vector velocity is specified independently of the path to allow continuous motion. The path is specified as a collection of segments. For the purpose of specifying the path, define a special X-Y coordinate system whose origin is the starting point of the sequence. Each linear segment is specified by the X-Y coordinate of the final point expressed in units of resolution, and each circular arc is defined by the arc radius, the starting angle, and the angular width of the arc. The zero angle corresponds to the positive direction of the X-axis and the CCW direction of rotation is positive. Angles are expressed in degrees, and the resolution is 1/256th of a degree. For example, the path shown in Fig. 12.2 is specified by the instructions:

VP	0,10000
CR	10000, 180, -90
VP	20000, 20000

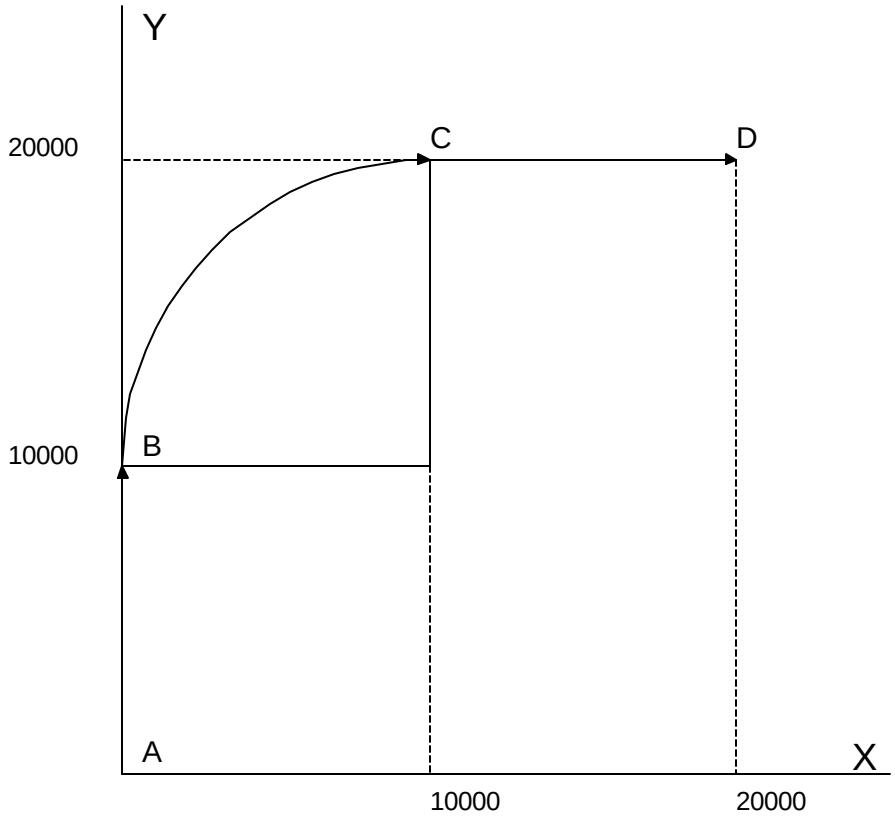


Figure 12.2 - X-Y Motion Path

The first line describes the straight line vector segment between points A and B. The next segment is a circular arc, which starts at an angle of 180° and traverses -90°. Finally, the third line describes the linear segment between points C and D. Note that the total length of the motion consists of the segments:

A-B	Linear	10000 units
B-C	Circular	$\frac{R\lvert\Delta q\rvert 2p}{360} = 15708$
C-D	Linear	1000
	Total	35708 counts

In general, the length of each linear segment is

$$L_k = \sqrt{Xk^2 + Yk^2}$$

Where Xk and Yk are the changes in X and Y positions along the linear segment. The length of the circular arc is

$$L_k = R_k\lvert\Delta\Theta_k\rvert 2p/360$$

The total travel distance is given by

$$D = \sum_{k=1}^{n} L_k$$

The velocity profile may be specified independently in terms of the vector velocity and acceleration.

For example, the velocity profile corresponding to the path of Fig. 12.2 may be specified in terms of the vector speed and acceleration.

VS	100000
VA	2000000

The resulting vector velocity is shown in Fig. 12.3.

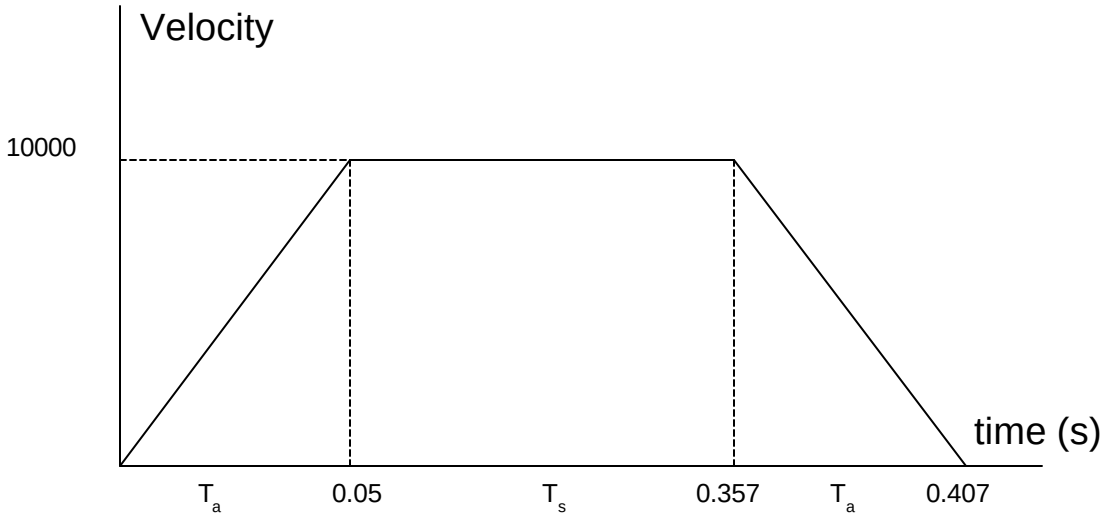


Figure 12.3 - Vector Velocity Profile

The acceleration time, T_a, is given by

$$T_a = \frac{VS}{VA} = \frac{100000}{2000000} = 0.05s$$

The slew time, Ts, is given by

$$T_s = \frac{D}{VS} - T_a = \frac{35708}{100000} - 0.05 = 0.307s$$

The total motion time, Tt, is given by

$$T_t = \frac{D}{VS} + T_a = 0.407s$$

The velocities along the X and Y axes are such that the direction of motion follows the specified path, yet the vector velocity fits the vector speed and acceleration requirements.

For example, the velocities along the X and Y axes for the path shown in Fig. 12.2 are given in Fig. 12.4.

Fig. 12.4a shows the vector velocity. It also indicates the position point along the path starting at A and ending at D. Between the points A and B, the motion is along the Y axis. Therefore,

Vy = Vs

and

Vx = 0

Between the points B and C, the velocities vary gradually and finally, between the points C and D, the motion is in the X direction.

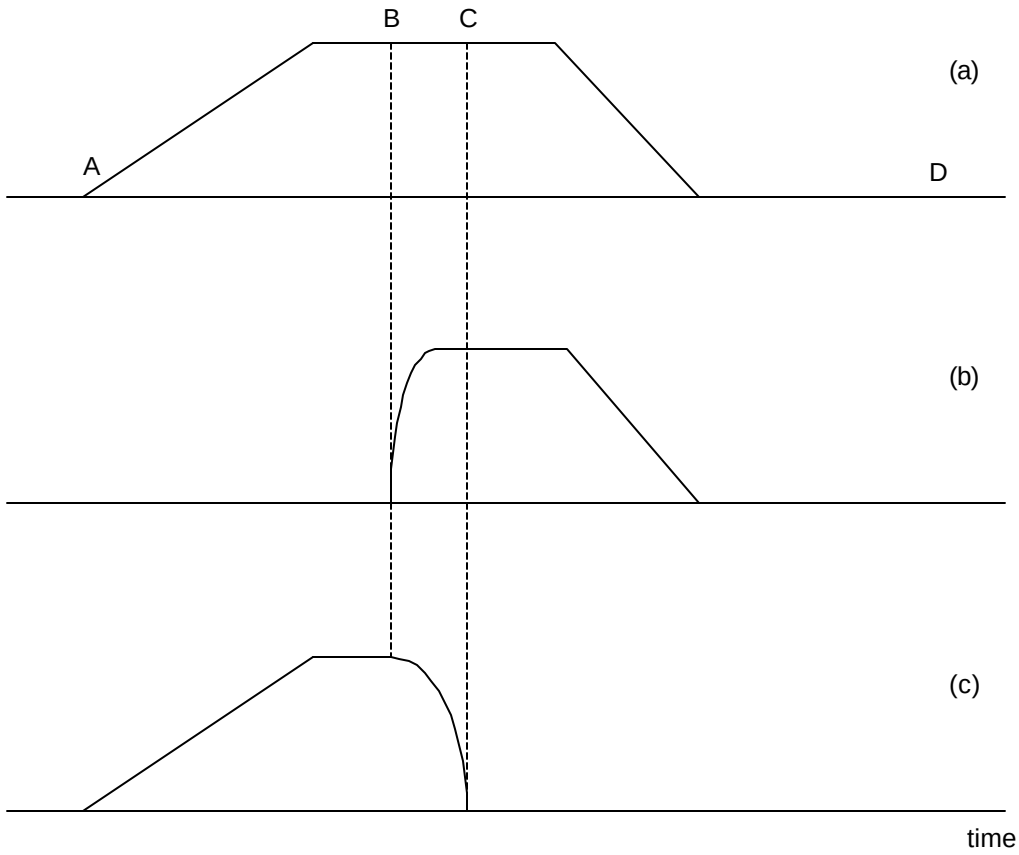


Figure 12.4 - Vector and Axes Velocities

DMC 500/DMC 1300 Comparison

Modes of Motion	DMC 500	DMC 1300
Relative positioning	Yes	Yes
Absolute positioning	Yes	Yes
Velocity control	Yes	Yes
Linear interpolation	XY only	Up to 4 axes
Circular interpolation	XY only	Any 2 axes plus 3rd tangent
Maximum number of segments in motion path	255	Infinite, continuous vector feed
Contouring	Yes	Yes
Electronic gearing	No	Yes
S-curve profiling	No	Yes
Programmable acceleration rate	Yes	Yes
Programmable deceleration rate	Yes	No

Specifications	DMC 500	DMC 1300
Maximum encoder frequency	2×10^6 counts/s	8×10^6 counts/s
DAC resolution	10-bits	14-bits or 16-bits
Maximum move length	8×10^6	2×10^9
Sample time	1 msec	0.5 msec (4 axes)
Program memory	500 lines, 32 chr	500 lines, 40 chr
EEPROM memory for parameter storage	None	Yes
Number of variables	64 (V0-V63)	126; symbolic up to 8 chrs, in addition to 64 (V0-V63).
Number of array elements	None	1600 (up to 14 arrays); 8000 (30 arrays) for DMC 1380 or DMC 1340-MX
Digital filter type	GN,ZR,KI	KP,KI,KD with velocity and acceleration feedforward and integrator limit

Hardware	DMC 500	DMC 1300
Maximum # of axes/card	3	4 (8 for DMC-1380)
Analog inputs	8 with DMC-63010	7 standard

Digital inputs	8 TTL	8 optoisolated (24 for DMC-1380)
Digital outputs	8 TTL	8 TTL (16 for DMC-1380)
High speed position latch	None	Yes
Dual encoder inputs	None	Yes
Motor command output	+/- 10V	+/- 10V and step/direction

DMC 500/DMC 1300 Command Comparison

Unchanged Commands

AB	Abort motion
AC	Acceleration rate
AD	After distance trippoint
AI	After input trippoint
AM	After motion trippoint
AP	After absolute position trippoint
AS	After at speed trippoint
BG	Begin motion
CB	Clear output bit
CM	Contour mode
CP	Clear program
CR	Circular segment
CS	Clear motion sequence
DP	Define position
ED	Edit mode
EN	End program
EO	Echo ON/OFF
ER	Define error limit
FA	Acceleration feedforward
FE	Find edge
GN	Gain
HM	Home
II	Interrupt for input
IP	Increment position
JG	Jog mode
JP	Conditional jump
JS	Conditional jump subroutine
KI	Integrator gain
MG	Message

MO	Motor off
NO	No-op
OE	Automatic error shut-off
OF	Offset
OP	Write output port
PA	Position absolute
PP	Program pause
PR	Position relative
RE	Return from error subroutine
RI	Return from interrupt subroutine
RM	Response mode
RS	Reset controller
SB	Set output bit
SC	Stop code/status
SH	Servo here
SP	Slew speed
ST	Stop motion/program
TB	Tell status byte
TC	Tell error code
TE	Tell error
TI	Tell inputs
TL	Torque limit
TM	Sample time
TP	Tell position
TR	Trace
TS	Tell switches
TT	Tell torque
VA	Vector acceleration
V[n]=	Variable definition
VP	Vector position
VS	Vector speed
WT	Programmable timer
XQ	Execute program
ZR	Filter zero
ZS	Zero subroutine stack

New Commands

AL	Arm latch

AR	After relative distance trippoint
AT	After time
AV	After vector distance trippoint
A[i]=n	Define array element
BL	Set reverse software limit
BN	Burn EEPROM
CD	Contour data
CE	Configure encoder
CN	Configure inputs and step motor
DA	Deallocate variables and arrays
DC	Deceleration
DE	Dual encoder position
DM	Dimension array
DT	Delta time for contouring
DV	Dual Velocity
EI	Enable interrupts
ES	Ellipse scale
FI	Search for encoder index
FL	Set forward software limit
FV	Velocity feedforward
GA	Specify master axis for gearing
GR	Specify gear ratio
HX	Halt task
IL	Integrator limit
IT	Independent time constant for smoothing
KD	Derivative constant
KP	Proportional constant
KS	Stepper Smoothing Constant
LE	Linear interpolation end
LI	Linear interpolation distance
LM	Linear interpolation mode
MT	Motor type
OB	Output Bit
RA	Record array
RC	Record
RD	Record data
RP	Report command position
TN	Tangent
TV	Tell velocity

VD	Vector deceleration
VE	Vector sequence end
VM	Coordinated motion mode
VT	Vector time constant - S-curve
WC	Wait for contour data

Deleted Commands

Deleted	Commands	Comments
DB	Deadband	Not necessary
DC	Decimal mode	Use local format; PF,VF
DD	Define dual encoder position	DE
DR	Set DAC resolution	14-bits only
HX	Hex mode	Use local format; PF,VF
LA	Arm latch	Replaced by AL command
LN	Learn mode	Use Record mode; RA and RD
MF	Master frequency	Use Electronic Gearing: GA & GR
MP	Master position	Use Electronic Gearing; GA & GR
MS	Master/slave mode	Use Electronic Gearing; GA & GR
P	Axis position (equate)	Use _TP
PC	Latch position	Use _RP
PD	Dual encoder position	Use _DE
PE	Position error (equate)	Use _TE
PL	Pole	Not required with KP, KD, KI
RC	Report when complete	Use AM or _BG
RM	Acceleration ramp	Use IT
SE	Specify encoder type	Use CE
SV	Servo	Use SH
TA	Enable S-curve	Use IT
TD	Tell dual encoder	Use MG _DE
TF	Tell master frequency Use Electronic Gearing; GA & GR	
TV	Enable S-curve	Use VT
VR	Specify S-curve	Use VT
ZM	Zero master Use Electronic Gearing; GA & GR	

List of Other Publications

"Step by Step Design of Motion Control Systems"

by Dr. Jacob Tal

"Motion Control Applications"

by Dr. Jacob Tal

"Motion Control by Microprocessors"

by Dr. Jacob Tal

Contacting Us

Galil Motion Control

203 Ravendale Drive

Mountain View, CA 94043

Phone: 650-967-1700

Fax: 650-967-1751

BBS: 650-964-8566 (8-N-1) up to 14,400 baud.

Internet address: support@galilmc.com

URL: www.galilmc.com

FTP: galilmc.com

WARRANTY

All products manufactured by Galil Motion Control are warranted against defects in materials and workmanship. The warranty period for controller boards is 1 year. The warranty period for all other products is 180 days.

In the event of any defects in materials or workmanship, Galil Motion Control will, at its sole option, repair or replace the defective product covered by this warranty without charge. To obtain warranty service, the defective product must be returned within 30 days of the expiration of the applicable warranty period to Galil Motion Control, properly packaged and with transportation and insurance prepaid. We will reship at our expense only to destinations in the United States.

Any defect in materials or workmanship determined by Galil Motion Control to be attributable to customer alteration, modification, negligence or misuse is not covered by this warranty.

EXCEPT AS SET FORTH ABOVE, GALIL MOTION CONTROL WILL MAKE NO WARRANTIES EITHER EXPRESSED OR IMPLIED, WITH RESPECT TO SUCH PRODUCTS, AND SHALL NOT BE LIABLE OR RESPONSIBLE FOR ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES.

Index

A

Abort 1, 25–27, 31, 66, 72, 139, 141, 162, 301, 303, 310–11, 317, 325
  Off-On-Error 11, 27, 31, 139, 141, 162, 243
  Stop Motion 66, 72, 116, 142, 265
Absolute Position 19, 61–62, 107–8, 113, 169–70, 194, 236, 239, 277, 325
Absolute Value 112, 119–20, 140
Acceleration 163, 171, 204–8, 218–20, 276, 286–88, 322–23, 324–26, 328
Accessories 313
Address 125–26, 144, 250–51, 329
AMP-1100 14, 314
Amplifier Enable 32–33, 139
Amplifier Gain 4, 150, 153, 155
Analog Input 1, 3, 25, 32, 120–21, 123, 128, 130–31, 136, 301, 315, 324
Arithmetic Functions 1, 97, 112, 118, 121
Arm Latch 94, 167, 326–28
Array 3, 70, 80–83, 97, 104, 112, 118, 123–28, 190, 193, 242, 302, 313, 324, 327
Automatic Subroutine 101, 114
  CMDERR 101, 115, 117
  LIMSWI 25, 101, 114–15, 140–42, 252
  MCTIME 101, 107, 115, 116, 234, 284
  POSERR 101, 114–15, 140–41, 202, 243, 252
Auxiliary Board 306
Auxiliary Encoder 1, 25, 76, 83–87, 83–87, 182, 183, 192, 209, 308, 311, 315
  Dual Encoder 87, 126, 192, 196

B

Backlash 86–87, 136, 196
Backlash Compensation
  Dual Loop 83–87, 83–87, 136, 196
Begin Motion 325
Binary 159
Bit-Wise 118
Burn 177
  EEPROM 3, 177, 189, 259
  Non-volatile memory 1–3
  Variables 179
Bypassing Optoisolation 30

C

Capture Data
  Record 80, 82, 123, 127, 249–51
Circle 133, 185, 186, 203
Circular Interpolation 1, 23–24, 71–72, 76, 125, 133, 250, 289
Clear Bit 128, 180
Clear Sequence 66, 68, 72, 74, 188
Clock 123, 274, 276
  Sample Time 276
  Update Rate 274
CMDERR 101, 115, 117
Code 159, 167, 182, 192, 196, 202, 205–6, 209, 212, 234, 236, 239, 262, 284
Command
  Syntax 55–56, 159–60
Command Summary 60, 123, 126
Commanded Position 62–64, 76–77, 117, 126, 131, 145–47, 209
Communication 3, 189
Compare Function 192, 270
Compensation
  Backlash 86–87, 136, 196
Conditional jump 1, 21, 27, 97, 110–12, 130, 166, 221–22
Configuration
  Jumper 30, 144, 184, 240, 259
Connector 25, 28, 33, 184, 240
Contour Mode 78–83, 181, 183, 188, 195, 295
Control Filter
  Damping 144, 148
  Gain 210, 224–25
  Integrator 148, 152–53, 217
  Proportional Gain 148
Coordinated Motion 57, 70–72, 209, 277, 286–87, 289, 292, 293
  Circular 1, 23–24, 71–72, 76, 125, 133, 250, 289
  Contour Mode 78–83, 181, 183, 188, 195, 295
  Electronic Gearing 1, 72–78, 209, 211
  Gearing 1, 72–78, 209, 211
  Linear Interpolation 23, 64–68, 64–68, 70, 76, 78, 227, 229–31, 291
  Vector Mode 173–74, 203, 291
Cosine 118–19, 124
Cycle Time
  Clock 123, 274, 276

D

DAC 1, 148, 152–53, 155
Damping 144, 148
Data Capture 125–26, 249
Data Output
  Set Bit 128, 180, 261
Debugging 104, 279
Deceleration 1, 162–63, 171, 191, 204–7, 219–20, 294
Default Setting 259
  Master Reset 161, 259, 260, 274
Differential Encoder 12, 14, 144
Digital Filter 152–53, 155–57
Digital Input 25, 27, 119, 129
Digital Output 119, 128
  Clear Bit 128, 180
Dip Switch
  Address 250–51, 329
Download 97
Dual Encoder 87, 126, 192, 196
  Backlash 86–87, 136, 196
  Dual Loop 83–87, 83–87, 136, 196
Dual Loop 83–87, 83–87, 136, 196
  Backlash 86–87, 136, 196

E

Echo 266
Edit Mode 21, 97–98, 105, 197, 297
Editor 21, 97–98
EEPROM 3, 177, 189, 259
  Non-Volatile Memory 1–3
Electronic Gearing 1, 72–78, 209, 211
Ellipse Scale 74, 203
Enable
  Amplifer Enable 32–33, 139
Encoder 58
  Auxiliary Encoder 1, 25, 76, 83–87, 83–87, 182, 183, 192, 209, 308, 311, 315
  Differential 12, 14, 144
  Dual Encoder 87, 126, 192, 196
  Index Pulse 12, 26, 91, 206, 212
  Quadrature 1–3, 4, 132, 140, 151, 164, 169, 170, 173, 176, 181–82, 186, 194, 202, 218, 236, 239, 246, 248, 256, 278, 283, 291
Error
  Codes 267, 268
Error Code 159, 167, 182, 192, 196, 202, 205–6, 209, 212, 234, 236, 239, 262, 284
Error Handling 25, 101, 114–15, 140–42, 252
Error Limit 11, 13, 17, 31, 115, 139–41, 202, 280
  Off-On-Error 11, 27, 31, 139, 141, 162, 243
Example
  Wire Cutter 131
Execute Program 21–22, 297

F

Feedforward Acceleration 204
Feedrate 74, 110, 133
Filter Parameter
  Damping 144, 148
  Gain 210, 224–25
  Integrator 148, 152–53, 217
  PID 14, 148, 152, 157
  Proportional Gain 148
  Stability 87, 136, 143–44, 196
Find Edge 26, 91, 205–6
Frequency 1, 5, 154–56, 226
  Sample Time 276
Function 27, 117–24, 133, 135, 136
Functions
  Arithmetic 97, 112, 118, 121

G

Gain 210, 224–25
  Proportional 148
Gear Ratio 76–77, 209, 211
Gearing 1, 72–78, 209, 211

H

Halt 67, 72, 103–7, 110–11, 129, 166, 168, 214, 265
  Abort 1, 25–27, 31, 66, 72, 139, 141, 162, 301, 303, 310–11, 317, 325
  Off-On-Error 11, 27, 31, 139, 141, 162, 243
  Stop Motion 66, 72, 116, 142, 265
Hardware 1, 25, 128, 139, 176, 207, 259
  Address 125–26, 144, 250–51, 329
  Amplifier Enable 32–33, 139
  Clear Bit 128, 180
  Jumper 30, 144, 184, 240, 259
  Offset Adjustment 33, 143
  Output of Data 127
  Set Bit 128, 180, 261
  Torque Limit 275
  TTL 5, 25, 27, 32–33, 139
Home Input 26, 91, 123, 205–6
Homing 26, 91, 205–6, 212
  Find Edge 26, 91, 205–6

I

I/O
  Amplifier Enable 32–33, 139
  Analog Input 120–21, 123, 128, 130–31, 136
  Clear Bit 128, 180

Digital Input 25, 27, 119, 129
Digital Output 119, 128
Home Input 26, 91, 123, 205–6
Output of Data 127
Set Bit 128, 180, 261
TTL 5, 25, 27, 32–33, 139
ICM-1100 7–9, 11, 25, 30, 31, 139
Independent Motion
Jog 19, 108–10, 115–17, 122, 136, 140, 191, 218, 220, 264
Index Pulse 12, 26, 91, 206, 212
ININT 101, 115, 130, 166, 201, 215, 253
Input
Analog 120–21, 123, 128, 130–31, 136
Digital 119, 129
Input Interrupt 101, 110, 115, 130, 201, 215, 266
ININT 101, 115, 130, 166, 201, 215, 253
Inputs
Analog 1, 3, 25, 32, 301, 315, 324
Installation 8–9, 143
Integrator 148, 152–53, 217
Interconnect Module
AMP-1100 314
ICM-1100 11, 25, 30, 31, 139
Interface
Terminal 121
Internal Variable 23, 112, 120, 122, 168
Interrogation 18, 20, 58–59, 127, 218
Interrupt 1–3, 101–3, 110, 114–15, 130, 177, 198–99, 201, 215, 249, 252–53, 266, 285, 299
Invert 182, 240
Invert Loop Polarity 144

J

Jog 19, 108–10, 115–17, 122, 136, 140, 191, 218, 220, 264
Joystick 121, 135–36
Jumper 30, 144, 184, 240, 259

K

Keyword 112, 118, 120, 123–24, 228, 233, 274
TIME 123–24, 274
KS 226

L

Label 30, 97–103, 107–15, 122–23, 133, 136–37, 141, 215, 221–22, 234, 252–53, 259, 284–85
LIMSWI 140–42
POSERR 140–41
Special Label 101, 141, 234, 284
Latch 31, 94, 167, 184, 254
Arm Latch 94, 167, 326–28
Data Capture 125–26, 249
Position Capture 94, 167
Record 80, 82, 123, 127, 249–51
Teach 82, 249
Limit
Torque Limit 13, 20
Limit Switch 25–27, 31, 101–3, 114–15, 123, 140–42, 144, 184, 207, 211, 228, 233, 252, 262, 266
LIMSWI 25, 101, 114–15, 140–42, 252
Linear Interpolation 23, 64–68, 64–68, 70, 76, 78, 227, 229–31, 291
Clear Sequence 66, 68, 72, 74, 188
Logical Operator 113, 221–22

M

Masking
Bit-Wise 118
Master Reset 161, 259, 260, 274
Math Function
Absolute Value 112, 119–20, 140
Bit-Wise 118
Cosine 118–19, 124
Logical Operator 113, 221–22
Sine 119–20
Mathematical Expression 117, 120
MCTIME 101, 107, 115, 116, 234, 284
Memory 1–3, 21, 97, 104, 113, 115, 123, 177, 190
Array 3, 70, 80–83, 97, 104, 112, 118, 123–28, 190, 193, 242, 302, 313, 324, 327
Download 97
Message 104, 115–17, 119, 127–28, 130, 141–42
Modelling 145, 148–49, 153
Motion Complete
MCTIME 101, 107, 115, 116, 234, 284
Motion Smoothing 1, 89, 90
S-Curve 66, 72, 171, 219, 294
Motor Command 1, 13, 20, 33, 152, 243–44, 275
Moving
Acceleration 163, 171, 204–8, 218–20, 276, 286–88, 322–23, 324–26, 328
Begin Motion 325
Circular 1, 23–24, 71–72, 76, 125, 133, 250, 289
Multitasking 103, 214
Execute Program 21–22, 297
Halt 67, 72, 103–7, 110–11, 129, 166, 168, 214, 265

N

Non-volatile memory
Burn 177
Non-Volatile Memory 1–3

O

OE
  Off-On-Error 139, 141, 162, 243
Off-On-Error 11, 27, 31, 139, 141, 162, 243
Offset Adjustment 33, 143
Operand
  Internal Variable 23, 112, 120, 122, 168
Operators
  Bit-Wise 118
Optoisolation 25, 27–29, 31
  Home Input 26, 91, 123, 205–6
Output
  Amplifier Enable 32–33, 139
  ICM-1100 11, 25, 30, 31
  Interconnect Module 7–9
  Motor Command 1, 13, 20, 33, 152, 243–44, 275
Output of Data 127
  Clear Bit 128, 180
  Set Bit 128, 180, 261

P

PID 14, 148, 152, 157
Play Back 127
Plug and Play 198, 285
POSERR 101, 114–15, 140–41, 202, 243, 252
  Position Error 13, 19, 164, 224, 243, 256
Position Capture 94, 167
  Latch 31, 94, 167, 184, 254
  Teach 82, 249
Position Error 11, 13, 19, 31, 101, 114–15, 122, 125–26, 131, 136, 139–41, 144, 147, 164, 224, 243, 256
  POSERR 202, 243, 252
Position Follow 131
Position Limit 140, 207
Program Flow 100, 106
  Interrupt 1–3, 101–3, 110, 114–15, 130, 177, 198–99, 201, 215, 249, 252–53, 266, 285, 299
  Stack 114, 117, 130, 215, 252–53, 296, 299
Programmable 1, 128, 136, 140
  EEPROM 3, 177, 189, 259
Programming
  Halt 103–7, 110–11, 129, 166, 168, 214, 265
Proportional Gain 148
Protection
  Error Limit 11, 13, 17, 31, 115, 139–41, 202, 280
  Torque Limit 13, 20, 275
PWM 4

Q

Quadrature 1–3, 4, 132, 140, 151, 164, 169, 170, 173, 176, 181–82, 186, 194, 202, 218, 236, 239, 246, 248, 256, 278, 283, 291
Quit
  Abort 1, 25–27, 31, 66, 72, 139, 141, 162, 301, 303, 310–11, 317, 325
  Stop Motion 66, 72, 116, 142, 265

R

Record 80, 82, 123, 127, 249–51
  Latch 31, 94, 167, 184, 254
  Position Capture 94, 167
  Teach 82, 249
Register 122
Reset 25, 32, 111, 139, 141, 161, 189, 258–59, 258–59, 274
  Master Reset 161, 259, 260, 274
  Standard 274

S

Sample Time 276
  Update Rate 274
Save
  Burn 177
  Non-Volatile Memory 1–3
SB
  Set Bit 128, 180, 261
Scaling
  Ellipse Scale 74, 203
S-Curve 171, 219, 294
  Motion Smoothing 1, 90
Selecting Address 125–26, 144, 250–51, 329
Set Bit 128, 180, 261
Sine 119–20
Single-Ended 4, 12, 14
Slew 1, 107, 110, 132, 218, 220, 264
Smoothing 1, 66, 68, 72, 74, 83–90, 219, 226
  KS 226
Software
  Terminal 121
Special Label 101, 141, 234, 284
Specification 227, 229–31, 265
Stability 87, 136, 143–44, 148, 154, 196
Stack 114, 117, 130, 215, 252–53, 296, 299
  Zero Stack 117, 130
Standard Reset 274
Status 68, 104–6, 122, 126, 183, 189, 190, 214, 243, 245, 250, 266–72, 280, 299
  Interrogation 18, 20, 58–59, 75, 127, 218
  Stop Code 126, 144, 262
Step Motor 1–4, 7, 90–91, 184, 226, 240

KS, Smoothing 1, 66, 68, 72, 74, 83–90, 219, 226
Smoothing 226
Stop
Abort 1, 25–27, 31, 66, 72, 139, 141, 162, 301, 303, 310–11, 317, 325
Stop Code 126, 144, 159, 167, 182, 192, 196, 202, 205–6, 209, 212, 234, 236, 239, 262, 284
Stop Motion 66, 72, 116, 142, 265
Subroutine 25, 101, 111–15, 130, 140–41, 215, 222, 252–53, 284, 299
Automatic Subroutine 101, 114
Synchronization 4
Syntax 55–56, 159–60

T

Tangent 71, 73–74, 277, 289
Teach 82, 249
Data Capture 125–26, 249
Latch 31, 94, 167, 184, 254
Play-Back 127
Position Capture 94, 167
Record 80, 82, 123, 127, 249–51
Tell Error
Position Error 13, 19, 164, 224, 243, 256
Tell Position 108, 122–24
Terminal 26, 30, 121, 189
Theory 145, 223, 225
Damping 144, 148
Digital Filter 152–53, 155–57
Modelling 145, 148–49, 153
PID 14, 148, 152, 157
Stability 87, 136, 143–44, 148, 154, 196
Time
Clock 123, 274, 276
Sample Time 276
Update Rate 274
TIME 123–24, 274
Time Interval 78–80, 82, 125, 183, 195, 249
Timeout 101, 107, 115, 116, 234, 284
MCTIME 101, 107, 115, 116, 234, 284
Torque Limit 13, 20, 275
Trigger 1, 97, 106, 108–11, 171, 202
Trippoint 107–8, 114, 164, 168–75, 169, 170, 214–15, 234, 236, 239, 252–53, 284, 296
Troubleshooting 143
TTL 5, 25, 27, 32–33, 139
Tuning
Stability 87, 136, 143–44, 148, 154, 196

U

Update Rate 274
Sample Time 276

V

Variable
Internal 23, 112, 120, 122, 168
Vector Acceleration 23–24, 68–69, 74, 133, 286–88
Vector Deceleration 23–24, 68–69, 74
Vector Mode 173–74, 203, 291
Circle 133, 185, 186, 203
Circular Interpolation 1, 23–24, 71–72, 76, 125, 133, 250, 289
Clear Sequence 66, 68, 72, 74, 188
Ellipse Scale 74, 203
Feedrate 74, 110, 133
Tangent 71, 73–74, 277, 289
Vector Speed 23–24, 65–72, 74, 110, 133, 186, 229–31, 264

W

Wire Cutter 131

X

XQ
Execute Program 21–22, 297

Z

Zero Stack 117, 130