DMC-1000

Manual Rev. 1.5

By Galil Motion Control, Inc.

Todd Shier

Galil Motion Control, Inc.
203 Ravendale Drive
Mountain View, California 94043
Phone: (415) 967-1700
Fax: (415) 967-1751
BBS: (415) 964-8566
Internet Address: galil@netcom.com

Rev 6-95

Mark in tech
support is good.
800 377-6329

Tech Support
Kaushal Shaw
ex 105

This manual provides the information necessary to program the DMC-1000 Series controllers. It is written for the DMC-1040 four-axis controller. Users of the DMC-1030 3-axis controller, DMC-1020 2-axis controller or DMC-1010 1-axis controller should also use this manual, noting that the DMC-1030 uses XYZ axes, the DMC-1020 uses XY axes, and the DMC-1010 uses the X-axis only.

This manual also applies to the DMC-1050 5-axis controller, DMC-1060 6-axis controller, DMC-1070 7-axis controller and DMC-1080 8-axis controller. Here the axes are denoted a,b,c,d,e,f,g,h where X,Y,Z,W may be used interchangeably with a,b,c,d. Instructions pertaining to the DMC-1050 through DMC-1080 are noted in italics throughout this manual. The DMC-1050, DMC-1060, DMC-1070 are referred to collectively as the DMC-1080.

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 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

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

Communication 31

Programming Basics 40

Programming Motion 50

Application Programming 84

Theory of Operation 128

Command Reference 141

Overview

Introduction

The DMC-1000 Series are state-of-the-art motion controllers that plug into the PC Bus. Extended performance capability over the previous generation of controllers includes: 8 MHz encoder input frequency, 14-bit motor command output DAC, +/-2 billion counts total travel per move, faster sample rate, bus interrupts and non-volatile memory for parameter storage. The controllers provide high performance and flexibility while maintaining ease-of-use and low cost.

Designed for maximum system flexibility, the DMC-1000 is available for one, two, three or four axes per card (an add-on card is available for control of five, six, seven or eight axes) and can be interfaced to a variety of motors and drives including step motors, servo motors and hydraulics.

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 that require one encoder on both the motor and the load, auxiliary encoder inputs are included for each axis.

The DMC-1000 is a powerful controller that provides several 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-1000 also provides S-curve profiling to eliminate jerk.

For synchronizing motion with external events, the DMC-1000 includes 8 optoisolated inputs, eight programmable outputs and seven analog inputs. I/O expansion boards provide additional inputs and outputs or interface to OPTO 22 racks. Event triggers can automatically check for elapsed time, distance and motion complete.

Despite its full range of sophisticated features, the DMC-1000 is easy to program. Instructions are represented by two letter commands such as BG for Begin and SP for Speed. Conditional Instructions, Jump Statements, and arithmetic functions are included for writing self-contained applications programs. An internal editor allows programs to be quickly entered and edited, and support software such as the SDK-1000 allows quick system set-up and tuning.

To prevent system damage during machine operation, the DMC-1000 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.

DMC-1000 Functional Elements

The DMC-1000 circuitry can be divided into the following functional groups as shown in Figure 1.1 and discussed below.

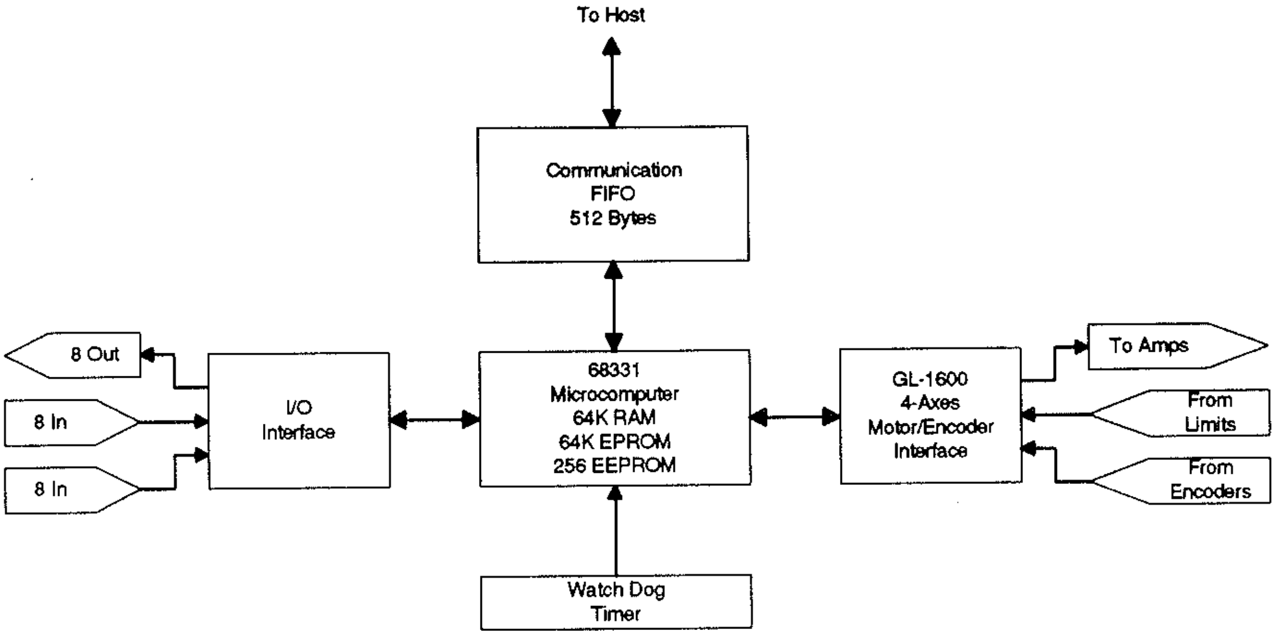


Figure 1.1 - DMC-1000 Functional Elements

Microcomputer Section

The main processing unit of the DMC-1000 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-1000. The EEPROM allows certain parameters to be saved in non-volatile memory upon power down.

Motor Interface

For each axis, a GL-1600 custom, sub-micron gate array performs quadrature decoding of the encoders at up to 8 MHz, generates the +/-10 Volt analog signal (14-Bit D-to-A) for input to a servo amplifier, and generates step and direction signal for step motor drivers. Interface to hardware limits and home inputs is also included.

Communication

The communication interface with the host PC uses a bi-directional FIFO (AM470) and includes PC interrupt handling circuitry.

General I/O

The DMC-1040 provides interface circuitry for eight optoisolated inputs, eight general outputs and seven analog inputs (12-Bit ADC). The DMC-1080 provides 24 inputs and 16 outputs. An auxiliary board, DB-10096, is available for interface to 96 additional I/O. The DB-10072 provides interface to up to three OPTO 22 racks with 24 I/O modules each.

System Elements

As shown in Fig. 1.2, the DMC-1000 is part of a motion control system which includes amplifiers, motors and encoders. These elements are described below.

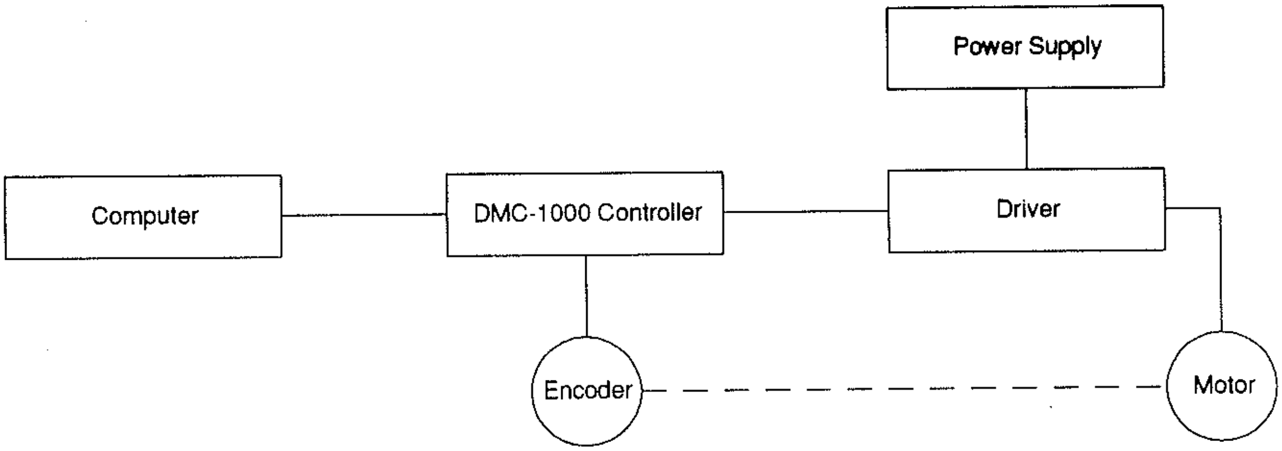


Figure 1.2 - Elements of Servo systems

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 required speed and acceleration. (Galil's "Motion Component Selector" software can help you with motor sizing).

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 be configured to control full-step, half-step, or microstep drives.

Amplifier

For each axis, the power amplifier converts the step/direction or the +/-10 Volt signal from the controller into enough current to drive the motor. As such, 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. Please call Galil if you need help configuring your amplifier.

Encoder

An encoder translates motion into an electrical signal to be fed back into the controller. The DMC-1000 accepts feedback from either a rotary or linear encoder. The preferred encoder is the one with two channels in quadrature, CHA and CHB. This encoder may also have a third channel (or index) for synchronization.When necessary, the DMC-1000 can 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 (8,000,000 quadrature counts/sec). For example, if the encoder line density is 10000 cycles per inch, the maximum speed is 200 inches/second.

The encoder type may be either single-ended (CHA and CHB) or differential (CHA,CHA-,CHB,CHB-). The DMC-1000 decodes either type into quadrature states or four times the number of cycles.

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-1000. Single-ended 12 Volt signals require a bias voltage input to the complementary inputs).

To interface other types of position sensors such as resolvers or absolute encoders, Galil offers the DB-10096 auxiliary card which can be customized for your particular sensor. Contact Galil.

Getting Started

Elements You Need

Before you start, you must get all the necessary system elements. These include:

1. Series Controller and included 60-pin Ribbon cable
2. Servo motors with Optical Encoder (one per axis) or step motors
3. Power Amplifiers
4. Power Supply for Amplifiers
5. PC (Personal Computer - ISA bus)
6. Communication Disk from Galil
7. SDK-1000 Servo Design Software (not necessary, but strongly recommended)
8. Interface Module with screw-type terminals (not necessary, but strongly recommended). An additional ICM-1100 is required for the DMC-1080.

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

For servo motors, the drivers 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 driver should accept step and direction signals. The step width is configured with the CN command and may be selected as 960 nsec, 1920 nsec or 15.36 µsec. For start-up of a step motor system, refer to "Connecting Step Motors" on page 12.

The encoders must have two channels in quadrature, and may have an index channel for synchronization. It is easier when a rotary encoder is mounted directly on the motor shaft, however, other forms of coupling are possible. The limitation on the encoder line density is that if the encoder has N cycles per revolution, the maximum frequency of the encoder must be limited to 2 MHz. For example, if N=5000 pulses/rev, the maximum motor velocity is 400 rev/s or 24,000 rpm.

The SDK-1000 software is highly recommended for first time users of the DMC-1000. It provides step-by-step instructions for system connection, tuning and analysis.

Installing the DMC-1000

The DMC-1000 is installed directly into the ISA expansion bus. The procedure is outlined below.

1. Make sure the PC is in the power-off condition. Unplug power cord from PC.
2. Remove the screws that hold the PC System Unit cover in place. These screws are usually located in the back of the system unit.
3. Remove unit cover.
4. Remove the metal plate covering the expansion bus slot where the is to be inserted. The DMC-1050 through DMC-1080 requires two expansion bus slots.
5. Insert DMC-1000 card in the expansion bus and secure with screw. Make sure the DMC-1000 is configured with appropriate options before inserting in PC Bus. For step motors, SM jumpers must be installed on appropriate axes.
6. Re-secure system unit cover and tighten screws.
7. Insert the 60-pin ribbon cable to J2 connector. (Ends of cable should be terminated appropriately to system components). For step motors, insert the 20-pin J4 connector. Uncommitted I/O and analog inputs are brought in through the 26-pin J5 connector. If you are using a DMC-1080, connect the additional cables.
8. Power-up PC.

Establishing Communication

After you have installed the DMC-1000 controller and powered-up the PC, you should establish communication between the controller and PC. The easiest way to do this is to use the communication disk (COMMDISK) available from Galil.

To use this disk, first boot-up your computer, then insert the disk in drive A. Type, INSTALL, and follow the directions. When you have finished installing the COMMDISK, type, TALK2BUS, to run the DMC-1000 interactive communication program. Once you have established communication, the terminal display should show a colon, :. If you do not receive a colon, the most likely cause is an address conflict in your computer.

We recommend the addresses of 1000 (default) or 816 as likely to be available. Please read the section on communication to determine how to change the address. You need to change the address jumpers and to configure the new address in the communication disk. Note that although the DMC-1080 physically takes up two I/O slots, only addressing the DMC-1040 slot is required.

Consult the factory if you need help.

Encoder Interface (skip if step motors)

If you are using the ICM-1100 Interface module, it connects all the signals to labelled phoenix screw-type connections. Otherwise, you need to interface to the Ribbon signals by other means. (The ICM-1100 makes interconnect very easy, especially for first time users).

Connect the encoder signals according to the following list. If you are using single-ended encoders with no complementary signals, ignore the complementary signal inputs.

Pin #	J2 DMC-1000							
2								5 Volt
1								GND
X	Y	Z	W	E	F	G	H	
33	39	45	51	33	39	45	51	Channel A
34	40	46	52	34	40	46	52	Channel A complementary
35	41	47	53	35	41	47	53	Channel B
36	42	48	54	36	42	48	54	Channel B complementary
37	43	49	55	37	43	49	55	Index
38	44	50	56	38	44	50	56	Index complementary

Note: For step and direction encoders use the Channel A input for the step signal and the Channel B input for the direction signal.

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

If the response of TPX does not vary in spite of the encoder rotation, here are some likely causes and solutions:

Problem - TPX does not vary with encoder rotation

Cause - Wrong connections

Solution - Check the wiring and correct as necessary.

Cause - Encoder failure

Solution - 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 it is established that the encoder failed, replace the encoder.

If you do not have the tools for testing the encoder signals, try a different encoder.

Cause - Hardware failure in the controller.

Solution - To check 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.

Connecting Servo Motors and the 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.

If you are using the integrated amplifier AMP-1140 from Galil, follow the connection diagram of Fig 2.1. The DMC-1080 requires an additional AMP-1140 for the extra axes. With separate amplifiers, use the connection diagram of Fig. 2.2. Connect and turn-on the amplifier power supply.

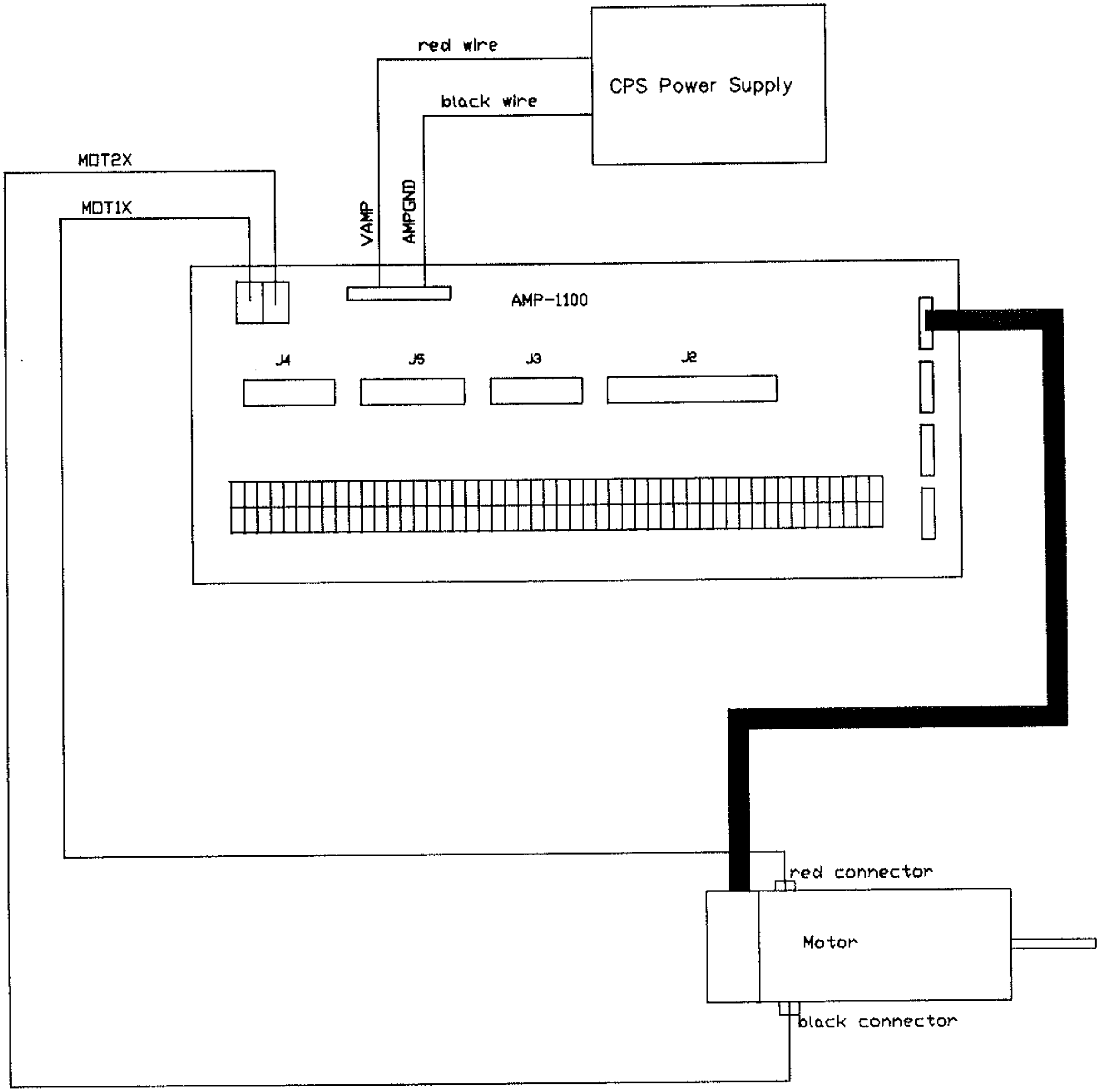
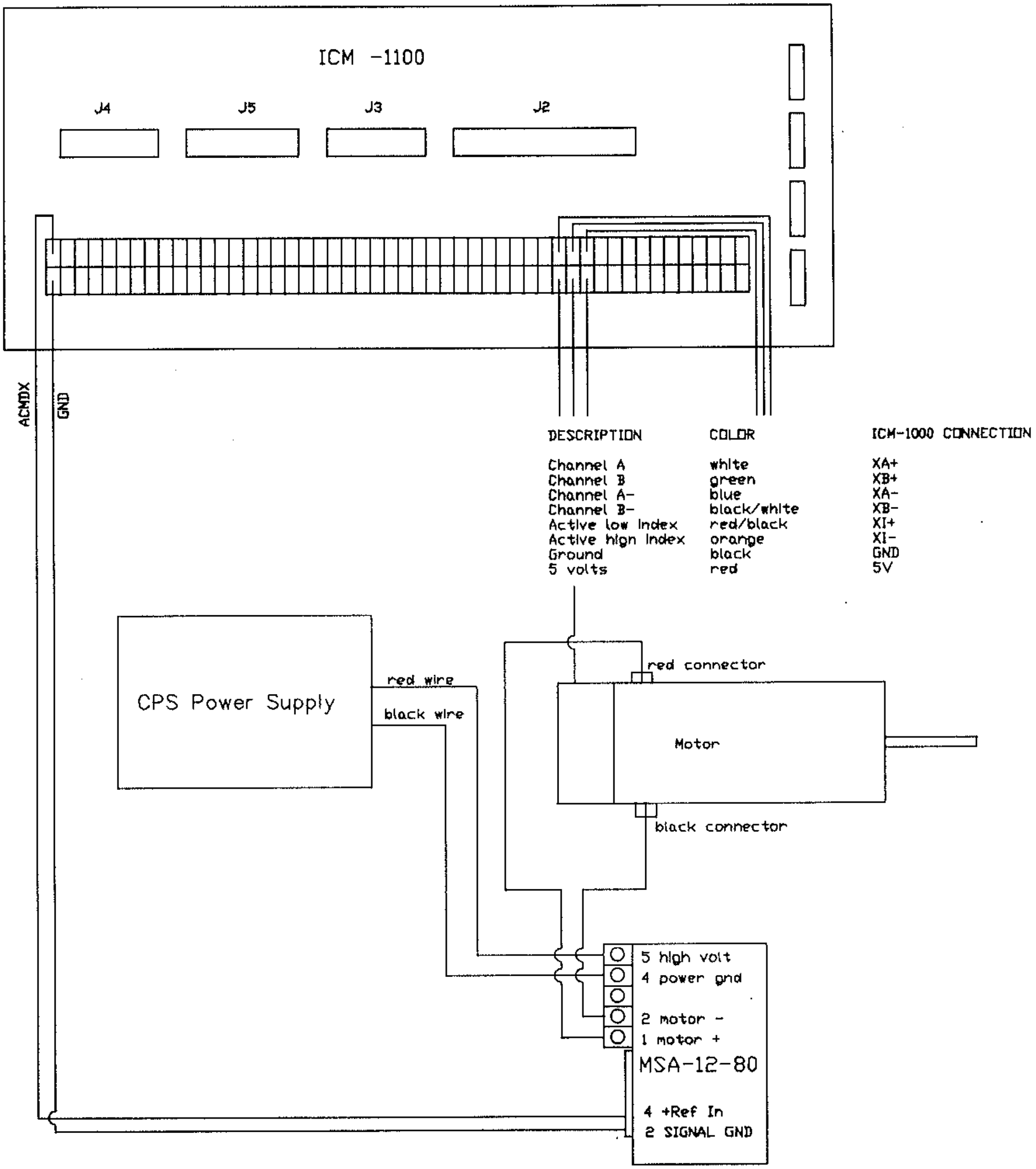


Figure 2-1 - System Connections with the AMP-1100 Amplifier



DESCRIPTION	COLOR	ICM-1000 CONNECTION
Channel A	white	XA+
Channel B	green	XB+
Channel A-	blue	XA-
Channel B-	black/white	XB-
Active low Index	red/black	XI+
Active high Index	orange	XI-
Ground	black	GND
5 volts	red	5V

Figure 2-2 - System Connections with a separate amplifier

The first step is to connect the motor to the amplifier with no connection to the controller. The motor should stand still.

The likely problems at this stage are the following:

Problem	- With no input, motor runs away.
Cause	- Amplifier offset.
Solution	- Adjust the amplifier offset to stop the motion.
Problem	- The motor runs at high speed and does not respond to offset adjustments.
Cause	- Damaged amplifier.
Solution	- Replace or repair amplifier.

Before connecting 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 will result to the computer controller and amplifier.

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

The next step is to close the position loop by connecting the amplifier to the controller. It is assumed that the motor and amplifier are connected together and that the encoder feedback is connected to the controller. All that is needed now is to apply the controller output to the amplifier.

There is an uncertainty about the correct polarity of the feedback. The wrong polarity causes the motor to run away from the starting position. To avoid damage, follow these steps:

Input the commands:

ER 2000 (CR)	Sets error limit
OE 1 (CR)	Sets OFF if excess error

This sets the controller in a shut-off mode if the motor runs away and creates a position error of 2000 counts. For further protection, you may input the command:

TL 1 (CR)	Torque limit of 1

This limits the output signal to 1 volt, resulting in reduced motor torque or speed.

Next, set the gain with the instruction:

GN 1 (CR)	Gain of 1

A low gain level is recommended for starting.

Once the parameters have been set, connect the analog motor command signal (pin J2-25) to the amplifier input.

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

PR 1000 (CR)	Position relative 1000 counts
BG X (CR)	Begin motion on X axis

When the polarity of the feedback is wrong, the motor will attempt to run away, but the controller will shut off when the position error exceeds 2000 counts. In this case the polarity of the loop must be inverted.

The loop polarity may be inverted in several manners. 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-.

Sometimes the feedback polarity is correct but the direction of motion is reversed with respect to the commanded motion. If this is the case, reverse the motor leads AND the encoder signals.

When the feedback polarity is correct, the motor moves in the required direction but stops short of the target due to system friction. The instruction:

TT X (CR)	Tell torque on X

reports the level of the output signal. It will show a non-zero value that is below the friction level.

Once you have established that you have closed the loop with the correct polarity, you can move on to the compensation phase to adjust the PID filter parameters, KP, KD and KI. Once you have completed set-up of the X axis, repeat for the Y,Z and W axes (and E F G H axes if you have a DMC-1080).

Connecting Step Motors

To connect step motors with the DMC-1000 you must follow this procedure:

1. Install an SM jumper on the DMC-1000 for each axis that is stepper.
2. Connect a 20-pin ribbon cable between the J4 connector on the DMC-1000 to the ICM-1100.

   IMPORTANT NOTE -- The pulse and direction signals do not come out the main 60-pin connector. Please request a 20-pin ribbon cable from Galil if you do not have one.
3. Connect step and direction signals from ICM-1100 to respective signals on your step motor driver.
4. Configure DMC-1000 for motor type using MT command. You can configure the DMC-1000 for active high or active low pulses using:

MT 2	active high step motor pulses
MT -2	active low step motor pulses
5. Configure for each axis using X,Y,Z W parameters.
6. Select the pulse width using the CN command where

CN,,,1	pulse width 960 nsec
CN,,,2	pulse width 1920 nsec
CN,,,3	pulse width 15.36 µsec

Step motors operate open-loop and do not require encoder feedback. However, an encoder can be used for position interrogation at any time using the TP command.

The DMC-1000 profiler commands the step motor driver. Here, all DMC-1000 motion commands apply such as PR, PA, VP, CR and JG. The acceleration, eceleration, slew speed and S-curve filtering are also used. However, since the motors run open-loop, the PID filter does not function and the position error is not generated.

The user can interrogate the profiled command position at any time using the Report Position command, RP.

NOTE: The DMC-1000 step mode works best with microstepping. (2000 steps per revolution or greater)

Compensation for 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. More advanced design methods are available with software design tools from Galil, such as the Servo Design Kit (SDK-1000).

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 the process, 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)        Proprotional 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 the 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 that is the case, 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.

Here again, the system may vibrate if the gain is too high. When that is the case, reduce KP. Typically, KP should not be greater than KD/4.

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. When that occurs, simply reduce KI. Now repeat tuning for the Y,Z and W axes (and E,F,G, and H axes if you have a DMC-1080).

Next, you are ready to try a few motion examples

Design Examples

Here are a few examples for tuning and using your controller.

Example 1 - System Set-up

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

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

For the DMC-1080, the X,Y,Z and W axes can also be referred to as the A,B,C,D 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
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.

Instruction	Interpretation
PR 10000	Distance
SP 20000	Speed
DC 100000	Deceleration
AC 100000	Acceleration
BG X	Start Motion

In response, the motor turns and stops.

Example 3 - Multiple Axes

Objective: To 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	Tell position all axes

which returns the four positions of the motors.

Individual axis may be interrogated with the instructions:

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 by the instructions

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 completed, command:

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, 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. The following program illustrates that effect.

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

The X motor will probably not move as the output signal is not 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

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. For example, the instruction

KP ?	Return gain of X axis.

returns the value of the proportional gain of the X axis. Similarly, the instruction

KP ,,?	Return gain of Z axis.

returns the value of the Z axis gain.

KP ?,?,?,?	Return gains of all axes.

returns the gain values for the four axes.

The same procedure applies to other parameters such as KI, KD, FA, etc.

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 memory. They may be executed at a later time.

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.

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.

Now 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
000 #A	Label
001 DP 0	Define current position as zero
002 V1=1000	Set initial value of V1
003 #Loop	Label for loop
004 PA V1	Move X motor V1 counts
005 BG X	Start X motion
006 AM X	After X motion is complete
007 WT 500	Wait 500 ms
008 TP X	Tell position X
009 V1=V1+1000	Increase the value of V1
010 JP #Loop,V1<10001	Repeat if V1<10001
011 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
000 #B	Label
001 DP 0,0	Define initial positions
002 PR 30000,60000	Set targets
003 SP 5000,5000	Set speeds
004 BGX	Start X motion
005 AD 4000	Wait until X moved 4000
006 BGY	Start Y motion
007 AP 6000	Wait until position X=6000
008 SP 2000,50000	Change speeds
009 AP ,50000	Wait until position Y=50000
010 SP ,10000	Change speed of Y
011 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
000 #A;DP0	Label; Define current position as zero
001 PR 4000	Initial position
002 SP 2000	Set speed
003 BGX	Move X
004 AMX	Wait until move is complete
005 WT 500	Wait 500 ms
006 #B	
007 V1 = _TPX	Determine distance to zero
008 PR -V1/2	Command X move 1/2 the distance
009 BGX	Start X motion
010 AMX	After X moved
011 WT 500	Wait 500 ms
012 V1=	Report the value of V1
013 JP #C, V1=0	Exit if position=0
014 JP #B	Repeat otherwise
015 #C;EN	End

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-1000 instruction with an underscore, _.

Example 15 - Control Variables and Offset

Objective: Illustrate the use of variables in iterative loops and use of multiple instructions on one line.

Instruction	Interpretation
000 #A;KI0;DP0;V1=8	Set initial values
001 #B;OF V1;WT 200	Set offset value
002 V2=_TPX;JP #C,@ABS[V2]<2;V2=	Exit if error small, report position
003 V1=V1-1;JP #B	Decrease Offset
004 #C;EN	End

This program starts with a large offset and gradually decreases its value, resulting in decreasing error.

Example 16 - 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 17 - Circular Interpolation

Objective: Move the XY axes in circular mode to form the path shown on Fig. 2.3.

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

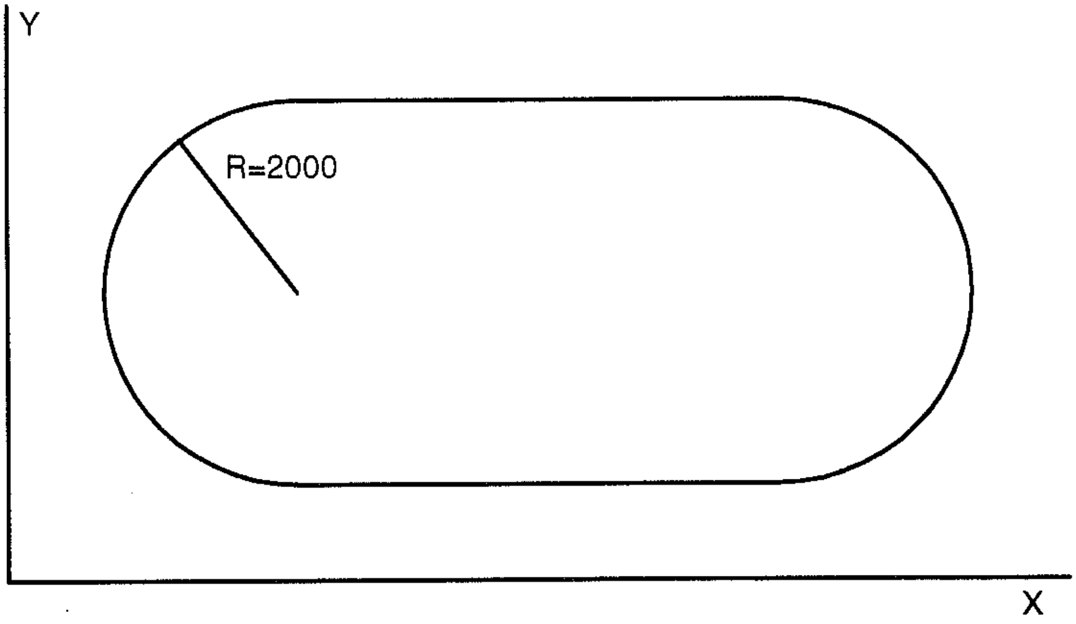


Figure 2-3 - Motion Path for Example 17

Hardware Interface

Encoder Interface

For each axis of motion, the DMC-1000 accepts inputs from incremental encoders with two channels in quadrature, or 90 electrical degrees out of phase. The DMC-1000 performs quadrature decoding of the two signals, resulting in bi-directional position information with a resolution of four times the number of full encoder cycles. For example, a 500 cycle encoder is decoded into 2000 quadrature counts per revolution. An optional third channel or index pulse may be used for homing or synchronization. Several types of incremental encoders may be used: linear or rotary, analog or digital, single-ended or differential. Any line resolution may be used, the only limitation being that the encoder input frequency must not exceed 2,000,000 full cycles/sec (or 8,000,000 quadrature counts/sec). The DMC-1000 also accepts inputs from an additional encoder for each axis. These are called auxiliary encoders and can be used for dual-loop applications.

The encoder inputs are not isolated.

Connections for the various types of encoders are described below.

	Pin #	of J2			Signal
	2				5 Volt
	1				GND
JD2	**E**	**F**	**G**	**H**	
J2	**X or A**	**Y or B**	**Z or C**	**W or D**	
	33	39	45	51	Channel A
	34	40	46	52	Channel A complementary
	35	41	47	53	Channel B
	36	42	48	54	Channel B complementary
	37	43	49	55	Index
	38	44	50	56	Index complementary

Use the above table to connect the signals as needed. For example, when connecting the X encoder with Channels A,B single ended, use pins 33 and 35, and ignore 34 and 36-38.

In a similar manner, the auxiliary encoders may be connected as follows:

	Pin #	of J3			Signal
	19				+5V
	20				GND
JD2	E	F	G	H	
J2	X or A	Y or B	Z or C	W or D	
	18	14	10	6	Channel A
	17	13	9	5	Channel A complementary
	16	12	8	4	Channel B
	15	11	7	3	Channel B complementary

The DMC-1000 can interface to incremental encoders of the pulse and direction type, instead of two channels in quadrature. In that case, replace Channel A by the pulse signal, and Channel B by the direction, and use the CE command to configure the DMC-1000 for pulse and direction encoder format. For pulse and direction format, the DMC-1000 provides a resolution of 1X counts per pulse.

Note that while TTL level signals are common, the DMC-1000 encoder inputs accept signals in the range of +/-12V. If you are using a non-TTL single-ended encoder signal (no complement), to assure proper bias, connect a voltage equal to the average signal to the complementary input. For example, if Channel A varies between 2 and 12V, connect 7 volts to Channel A complement input.

Optoisolated Inputs

The DMC-1000 provides optoisolated digital inputs for limit, home, abort and the uncommitted inputs. The limit and home inputs for each axis have the same common Limit Switch Common (LSCOM). The abort and uncommitted inputs use Input Common (INCOM), which is isolated from the limit and home inputs. The connections are shown in Fig 3.1. For the DMC-1080, a separate LSCOM and INCOM are provided for the A,B,C,D axes and E,F,G,H axes.

If nothing is connected to the inputs, no current flows, resulting in a logic one. A logic zero is generated when at least 1 mA of current flows from the common to the input.

Limit and Home Connections

The Limit and Home inputs use the switch common, LSCOM. LSCOM is the same for the X,Y,Z and W axes. For the DMC-1080, a separate LSCOM is provided for the A,B,C,D, axes and E,F,G,H axes.

When an isolated supply voltage is used, the positive voltage should be connected to LSCOM. The ground of the isolated supply should be connected to the home and limit switches. Current flows (resulting in a logic zero) when the switch is closed.

Figure 3.2 shows the switch and supply connections.

The logical sense of the limit and home inputs may be inverted with the CN command.

An isolated supply in the voltage range between 5 to 28 Volts may be applied directly. For voltages greater than 28 Volts, a resistor, R, is needed in series with the input such that

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

If no isolation is needed, the internal 5 Volt supply may be used to power the switches, as shown in Fig. 3.3. This can be done by connecting a jumper between the pins LSCOM and 5V on either the ICM-1100 or the DMC-1000. LSCOM is the Limit Switch common.

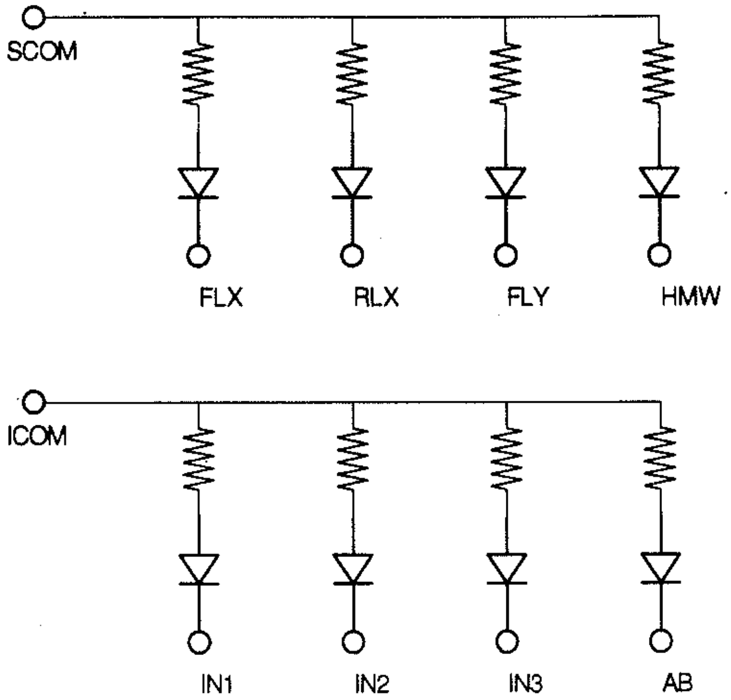


Figure 3-1 - The Optoisolated Inputs

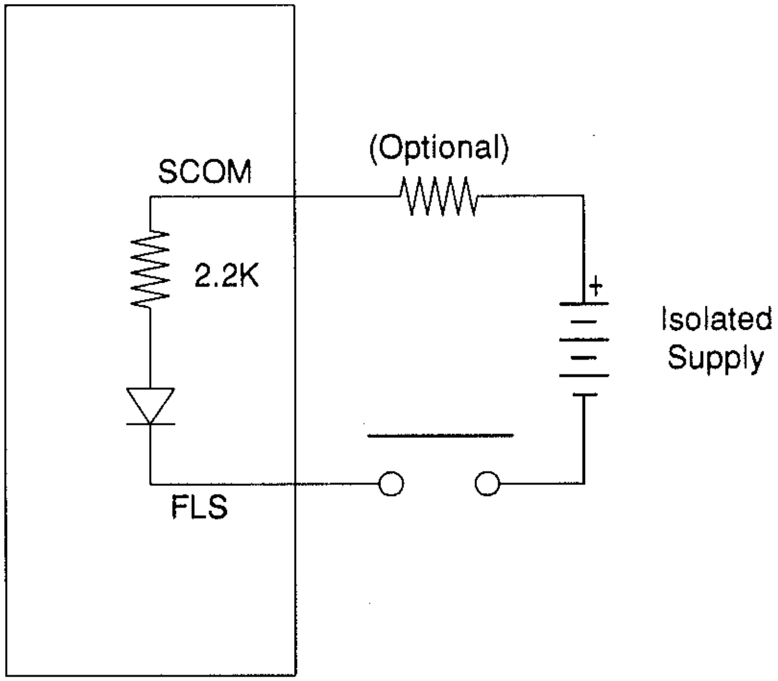


Figure 3-2 - Connecting Limits and Home Switches to an isolated supply

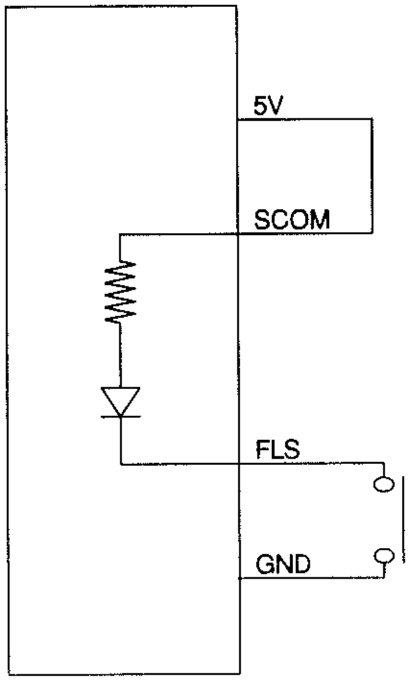


Figure 3-3 - Connecting Limit switches to the internal 5V supply

Abort and Input Connections

The abort and uncommitted inputs use the input common, INCOM. INCOM is isolated from the home and limit common.

An isolated supply voltage should be connected to INCOM. The ground for the isolated supply should be connected to the abort and uncommitted inputs. Current flows (resulting in a Logic Zero) when the switch is closed.

Figure 3.4 shows the connections.

An isolated supply in range from 5 Volts to 28 Volts may be connected directly. For voltages greater than 28 Volts a resistor is needed in series with the input such that:

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

To connect the internal 5 Volt supply, connect a jumper between the pins INCOM and 5V on either the ICM-1100 or the DMC-1000. If you use the internal supply, the inputs will not be isolated.

The DMC-1080 provides 16 isolated inputs and 8 additional TTL inputs. The input commons for the 8 isolated inputs on JD5 and the 8 isolated inputs on J5 are isolated.

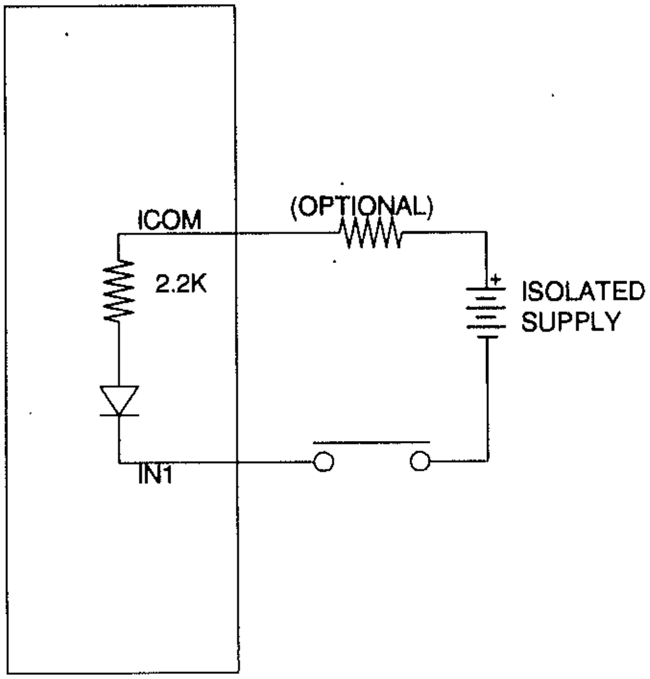


Figure 3-4 - Connecting Abort and uncomitted inputs to an isolated supply

Outputs

The DMC-1000 provides several output signals including eight general outputs, an error signal, ER, and four amplifier enable signals AEN. All the output signals are TTL (0 to 5 volts). The DMC-1080 provides an additional eight outputs (connector JD5, pins 10-17).

Analog Inputs

The DMC-1000 has seven analog inputs configured for the range between -10V and 10V. The inputs are decoded by a 12-bit A/D decoder. The impedence of these inputs is 10 KΩ.

Amplifier Interface

The DMC-1000 generates +/-10 Volt range analog signal, ACMD, and ground for each axis. This signal is input to power amplifiers which have been 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-1000 also provides an AEN, amplifier enable signal, to control the status of the amplifier. This signal toggles when the watchdog timer activates, when a motor-off command is given, or when OE1 (Off-on-error is enabled) command is given and the position error exceeds the error limit. As shown in Figure 3.5, AEN can be used to disable the amplifier for these conditions.

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 change the polarity from active low (zero volts = disable) to active high replace the 7407 IC with a 7406.

To change the voltage level, 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 to external supplies with voltages up to 24V.

For each axis, the DMC-1000 provides offset correction potientiometers to compensate for any offset in the analog output.

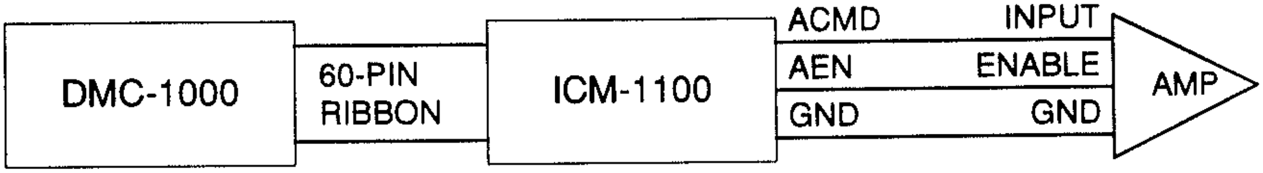


Figure 3-5 - Connecting AEN to an amplifier

Other Inputs

A reset input is TTL level, non-isolated signal. The reset is used to locally reset the DMC-1000 without resetting the PC.

Offset Adjustment

The DMC-1000 has been adjusted at the factory to produce 0 Volts output for a zero digital motor command. If you need to adjust the offset, you can do so by sending the motor off command, MO, to the DMC-1000. This causes a zero digital motor command. Connect a scope or voltmeter to the motor command pin. You should measure zero volts. If not, adjust the appropriate offset potentiometer on the DMC-1000 (for X,Y,Z and W) until a zero voltage is observed.

Multiple Card Synchronization

The sample clocks of multiple DMC-1000's can be synchronized together as follows:

1. Select one card as the master. For the other cards, specify a -1 for the sample time (TM -1). Burn this into the non-volatile memory with the BN command. (Do not specify TM -1 for the master).
2. Connect J3 pin 1 of each of the DMC-1000's.
3. To start motion at the same time on each card, send all commands in the motion profile except for the final ; following the BG commands. After the profiles are sent to each card, send the final semicolons in succession. Motion will start within 1 sample period on each card and the skew will not accumulate.

Communication

Introduction

The DMC-1000 receives commands from a PC/XT/AT or compatible computer. The controller is configured as a standard AT style card that is mapped into the I/O space. Communication between the DMC-1000 and the computer is in the form of ASCII characters where data is sent and received via READ and WRITE registers on the DMC-1000. A handshake is required for sending and receiving data.

The DMC-1000 contains a 512 character write FIFO buffer, which permits sending commands at high speeds ahead of their actual processing by the DMC-1000. It also contains a 512 character read buffer.

This chapter on communication discusses Address Selection, Communication Register Description, A Simplified Method of Communication, Advanced Communication Techniques, and Bus Interrupts.

Address Selection

The DMC-1000 address, N, is selectable by setting the Address Dip Switches A2, A3, A4, A5, A6, A7 and A8, where A2 represents 2^2, A3 represents 2^3 bit and so on. A switch in the ON setting represents that bit is zero 0 and OFF represents that bit is 1. (Note that the address, N, represents the I/O address of the controller. To write to the desired axis X,Y,Z or W, you specify that axis in the instruction command, such as BG Y for begin Y).

The default address of the DMC-1000 is 1000 (A4 and A2 switches ON).

The DMC-1000 can be configured for any 4th address between 512 and 1024. It is the responsibility of the user to assure there are no address conflicts between the DMC-1000 and the computer. The DMC-1000 must not conflict with an address used by the PC or another I/O card.

WARNING: The DMC-1000 address setting must not conflict with an address used by the PC or another I/O card. An address conflict will prevent communication or cause data conflicts resulting in lost characters.

To select an address (N), first make sure it is a number between 512 and 1024 that is divisible by four. Then subtract 512 from N and use the switches A2 through A8 to represent the binary result. A switch in the ON position represents a binary 0 and the OFF position represents binary 1.

Example - Address Selection

1. Select address, N, as 996.
2. Check to see if N is divisible by 4.
3. Subtract 512 from N.

996-512=484

4. Convert result from above into binary.

```
484=1  1  1  1  0  0 1  0  0
    2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
```

5. Let switches A2 through A8 represent bits 2^2 through 2^8 of above,

Where ON= 0, OFF=1

Switch	Position
A2	OFF
A3	ON
A4	ON
A5	OFF
A6	OFF
A7	OFF
A8	OFF

Note: To help calculate the pin settings use the program "Addrcalc.exe" located on the COMMDISK provided by Galil.

Communication Registers

Register	Description	Address	Read/Write
READ	for receiving data	N	Read only
WRITE	for transmitting data	N	Write only
CONTROL	for status control	N+1	Read and Write

The DMC-1000 provides three registers used for communication. The READ register and WRITE register occupy address N and the CONTROL register occupies address N+1 in the I/O space. The READ register is used for receiving data from the DMC-1000. The WRITE register is used to send data to the DMC-1000. The CONTROL register may be read or written to and is used for controlling communication, flags and interrupts.

Simplified Communication Procedure

The simplest approach for talking to the DMC-1000 is to check bits 4 and 5 of the CONTROL register at address N+1. Bit 4 is for WRITE STATUS and bit 5 is for READ STATUS.

Status Bit	Name	Logic State	Meaning
5	READ	0	Data to be read
5	READ	1	No data to be read
4	WRITE	0	Buffer not full, OK to write up to 16 characters
4	WRITE	1	Buffer almost full. Do not send data

READ PROCEDURE

To receive data from the DMC-1000, read the control register at address N+1 and check bit 5. If bit 5 is zero, the DMC-1000 has data to be read in the READ register at address N. Bit 5 must be checked for every character read and should be read until it signifies empty. Reading data from the READ register when the register is empty will result in reading an FF hex.

WRITE PROCEDURE

To send data to the DMC-1000, read the control register at address N+1 and check bit 4. If bit 4 is zero, the DMC-1000 FIFO buffer is not almost full and up to 16 characters may be written to the WRITE register at address N. If bit 4 is one, the buffer is almost full and no additional data should be sent. The size of the buffer may be changed (see "Changing Almost Full Flags" on page 33).

Any high-level computer language such as C, Basic, Pascal or Assembly may be used to communicate with the DMC-1000 as long as the READ/WRITE procedure is followed as described above. Example software drivers are contained on the COM-DISK from Galil.

Advanced Communication Techniques

Changing Almost Full Flags

The Almost Full flag (Bit 4 of the control register) can be configured to change states at a different level from the default level of 16 characters.

The level, m, can be changed from 16 up to 256 in multiples of 16 as follows:

1. Write a 5 to the control register at address N+1.
2. Write the number m-16 to the control register where m is the desired Almost Full level between 16 and 256.

For example, to extend the Almost Full level to 256 bytes, write a 5 to address N+1. Then write a 240 to address N+1.

Clearing FIFO Buffer

The FIFO buffer may be cleared by writing the following sequence:

Read N+1 address

Send 01H to N+1 address

Send 80H to N+1 address

Send 01H to N+1 address

Send 80H to N+1 address

Read N+1 address (Bit 7 will be 1)

It is a good idea to clear any control data before attempting this procedure. Send a no-op instruction, by reading N+1 address, before you start. All data, including data from the DMC-1000, will then be cleared.

Clearing the FIFO is useful for emergency resets or Abort. For example, to Reset the controller, clear the FIFO, then send the RS command.

Interrupts

The DMC-1000 provides a hardware interrupt line that will, when enabled, interrupt the PC. Interrupts free the host from having to poll for the occurrence of certain events such as motion complete or excess position error.

The DMC-1000 uses only one of the PC's interrupts, however, it is possible to interrupt on multiple conditions. The controller provides a register that contains a byte designating each condition.

The user can select the interrupt request level in addition to the interrupt conditions. The user can also send an interrupt with the UI command.

Configuring Interrupts

To use the DMC-1000 interrupt, you must complete the following four steps:

1. The DMC-1000 board must contain one jumper to designate the
2. Interrupt line for the PC bus. The available lines are IRQ2, IRQ3, IRQ4, IRQ5, IRQ7, IRQ9, IRQ10, IRQ11, IRQ12, IRQ14, IRQ15. Place a jumper on the desired line. Only one line may be jumpered. Note that the jumper for IRQ2 and IRQ9 is at the same location. IRQ9 is used for computers wired for the AT standard and IRQ2 is used for computers wired for the XT standard. If you aren't sure, select another interrupt line instead. Note that for the PC bus, only one I/O card can be attached to each interrupt request line.
3. Your host software code must contain an interrupt service routine and must initialize the interrupt vector table in the PC. The interrupt vector table and an example interrupt service routine, INIT_1000.C (included in Galil's COMMDISK) is shown in Appendix 12.8. Failure to have proper interrupt servicing in your host program could cause disastrous results including resetting or "hanging" your computer.
4. The DMC-1000 interrupt hardware must be initialized following each reset. This is done by writing the data 2 followed by 4 to the control register at address N+1.
5. The Interrupt conditions must be enabled with the EI instruction. (The UI instruction does not require EI). The EI instruction has the following format:
6. EI M,N where

The * conditions must be re-enabled after each occurrence.

Bit Number (m)	Condition
0	X motion complete
1	Y motion complete
2	Z motion complete
3	W motion complete
4	E motion complete
5	F motion complete
6	G motion complete
7	H motion complete
8	All axes motion complete
9	Excess position error*
10	Limit switch*
11	Watchdog timer
12	Reserved
13	Application program stopped
14	Command done
15	Inputs* (uses n for mask)

Bit number (n)	Input
0	Input 1
1	Input 2
2	Input 3
3	Input 4
4	Input 5
5	Input 6
6	Input 7
7	Input 8

and $M = \Sigma\, 2^m$

$N = \Sigma\, 2^n$

For example, to select an interrupt for the conditions X motion complete, Z motion complete and excess position error, you would enable bits 0, 2 and 9.

$M = 2^9 + 2^2 + 2^0 = 512 + 4 + 1 = 517$

EI 517

If you want an interrupt for Input 2 only, you would enable bit 15 for the m parameter and bit 1 for the n parameter.

$M = 2^{15} = 32{,}768$

$N = 2^1 = 2$

EI 32768,2

The DMC-1000 also provides 16 User Interrupts which can be sent by sending the command UI n to the DMC-1000, where n is an integer between 0 and 15. The UI command does not require the EI command.

Servicing Interrupts

Once an interrupt occurs, the host computer can read information about the interrupt by first writing the data 6 to the control register at address N + 1. Then the host reads the control register data. The returned data has the following meaning:

Hex Data	Condition
00	No interrupt
D9	Watchdog timer activated
DA	Command done
DB	Application program done
F0 thru FF	User interrupt
E1 thru E8	Input interrupt
C0	Limit switch occurred
C8	Excess position error
D8	All axis motion complete
D7	E axis motion complete
D6	F axis motion complete
D5	G axis motion complete
D4	H axis motion complete
D3	W axis motion complete
D2	Z axis motion complete
D1	Y axis motion complete
D0	X axis motion complete

Example - Interrupts

1) Interrupt on Y motion complete on IRQ5.

Jumper IRQ5 on DMC-1000

Install interrupt service routine in host program

Write data 2, then 4 to address N + 1

Enable bit 1 on EI command, m = 2^1 = 2;   EI 2

PR,5000

BGY

Now, when the motion is complete, IRQ5 will go high, triggering the interrupt service routine. Write a 6 to address N + 1. Then read N + 1 to receive the data D1 hex.

2) Send User Interrupt when at speed

```
#I              Label
PR 1000         Position
SP 5000         Speed
BGX             Begin
ASX             At speed
UI1             Send interrupt
EN              End
```

This program sends an interrupt when the X axis is at its slew speed. After a 6 is written to address N + 1, the data EI will be read at address N + 1.

EI corresponds to UI1.

Controller Response to DATA

Most DMC-1000 instructions are represented by two characters followed by the appropriate parameters. Each instruction must be terminated by a carriage return or semicolon.

Instructions are sent in ASCII, and the DMC-1000 decodes each ASCII character (one byte) one at a time. It takes approximately .5 msec for the controller to decode each command. However, the PC can send data to the controller at a much faster rate because of the FIFO buffer.

After the instruction is decoded, the DMC-1000 returns a colon (:) if the instruction was valid or a question mark (?) if the instruction was not valid or was not recognized.

For instructions requiring data, such as Tell Position (TP), the DMC-1000 will return the data followed by a carriage return, line feed and : .

It is good practice to check for : after each command is sent to prevent errors. An echo function is provided to enable associating the DMC-1000 response with the data sent. The echo is enabled by sending the command EO 1 to the controller.

Programming Basics

Introduction

The DMC-1000 provides over 100 commands for specifying motion and machine parameters. Commands are included to initiate action, interrogate status and configure the digital filter.

The DMC-1000 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, and ST stops the motion.

Commands can be sent "live" over the bus for immediate execution by the DMC-1000, or an entire group of commands can be downloaded into the DMC-1000 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-1000 instruction set and syntax. A complete listing of all DMC-1000 instructions is included in the command reference section.

Command Syntax

DMC-1000 instructions are represented by two ASCII upper case characters followed by applicable arguments. A space may be inserted between the instruction and arguments. A semicolon or <enter> is used to terminate the instruction for processing by the DMC-1000 command interpreter.

IMPORTANT: All DMC-1000 commands are sent in upper case.

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.

For specifying data for the X,Y,Z and W axes, commas are used to separate the axes and preserve axis order as X,Y,Z and W. 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 the DMC-1080, the eight axes are referred to A,B,C,D,E,F,G,H where X,Y,Z,W and A,B,C,D may be used interchangeably.

To view the current values for each command, specify the command followed by a ? for each axis requested. The DMC-1000 provides an alternative method for specifying data. Here data is specified individually using a single axis specifier such as X,Y,Z or W (or A,B,C,D,E,F,G or H for the DMC-1080). An equals sign is used to assign data to that axis. For example:

Example XYZW Syntax for Specifying Data

PR 1000	Specify X only as 1000
PR ,2000	Specify Y only as 2000
PR ,,3000	Specify Z only as 3000
PR ,,,4000	Specify W only as 4000
PR 2000,4000,6000,8000	Specify X Y Z and W
PR ,8000,,9000	Specify Y and W only
PR ?,?,?,?	Request X,Y,Z,W values
PR ,?	Request Y value only

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. The letter S is used to specify a coordinated motion sequence. For the DMC-1080, the eight axes are commanded with ABCDEFGH or XYZWEFGH where XYZW is used interchangeably with ABCD.

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

For the DMC-1080 only:

BG ABCDEFGH	Begin all axes
BG D	Begin D only

Controller Response to Commands

For each valid command entered, the DMC-1000 returns a colon :. If the DMC-1000 decodes a command as invalid, it returns a question mark ?.

NOTE:

The DMC-1000 returns a : for valid commands.

The DMC-1000 returns a ? for invalid commands.

For example, if the command BG is sent in lower case, the DMC-1000 will return a ?.

:bg <enter>	invalid command, lower case
?	DMC-1000 returns a ?

The command Tell Code, TC1, will return the reason for the ? received for the last invalid command.

```
:TC1 <enter>              Tell Code command
1 Unrecognized command    Returned response
```

There are several coded reasons for receiving a ?. Example codes include unrecognized command (such as typographical entry or lower case), a command given at improper time, or a command out of range, such as exceeding maximum speed. A complete listing of all codes is listed in the TC command in the Command Reference section.

For interrogation instructions such as Tell Position (TP) or Tell Status (TS), the DMC-1000 returns the requested data on the next line followed by a carriage return and line feed. The data returned is in decimal format.

```
:TP X <enter>                  Tell Position X
0000000000                     data returned
:TP XY <enter>                 Tell Position X and Y
0000000000,0000000000          data returned
:
```

The format of the returned data can be set using the Position Format (PF) and Variable Format (VF) command.

```
:PF 4 <enter>                  Position Format is 4 integers
:TP X <enter>                  Tell Position
0000                           returned data
```

Command Summary

Each DMC-1000 command is described fully in the command reference section at the end of this manual. A summary of the commands follows.

The commands are grouped in this summary by the following functional categories:

Motion, Program Flow, General Configuration, Control Settings, Status and Error/Limits.

Motion commands are those to specify modes of motion such as Jog Mode or Linear Interpolation, and to specify motion parameters such as speed, acceleration and deceleration, and distance.

Program flow commands are used in Application Programming to control the program sequencer. They include the jump on condition command and event triggers such as after position and after elapsed time.

General configuration commands are used to set controller configurations such as setting and clearing outputs, formatting variables, and motor/encoder type.

The control setting commands include filter settings such as KP, KD and KI and sample time.

Error/Limit commands are used to configure software limits and position error limits.

MOTION

AB	Abort Motion
AC	Acceleration
BG	Begin Motion
CD	Contour Data
CM	Contour Mode
CR	Circle
CS	Clear Motion Sequence
DC	Deceleration
DT	Contour Time Interval
ES	Ellipse Scaling
FE	Find Edge
FI	Find Index
GA	Master Axis for Gearing
GR	Gear Ratio
HM	Home
IP	Increment Position
JG	Jog Mode
LE	Linear Interpolation End
LI	Linear Interpolation Distance
LM	Linear Interpolation mode
PA	Position Absolute
PR	Position Relative
SP	Speed
ST	Stop
TN	Tangent
VA	Vector acceleration
VD	Vector Deceleration
VE	Vector Sequence End
VM	Coordinated Motion Mode
VP	Vector Position
VR	Vector speed ratio
VS	Vector Speed

PROGRAM FLOW

AD	After Distance
AI	After Input
AM	After Motion Complete
AP	After Absolute Position
AR	After Relative Distance
AS	At Speed
AT	After Time
AV	After Vector Distance
EA	Choose ECAM master
EB	Enable CAM
EG	Engage ECAM
EM	CAM cycle command
EN	End Program
EP	CAM interval and starting point
EQ	Disengage ECAM
ET	ECAM table entry
HX	Halt Task
IN	Input Variable
II	Input Interrupt
JP	Jump To Program Location
JS	Jump To Subroutine
MC	After motor is in position
MF	After motion -- forward direction
MG	Message
MR	After motion -- reverse direction
NO	No operation
RE	Return from Error Subroutine
RI	Return from Interrupt
TW	Timeout for in position
WC	Wait for Contour Data
WT	Wait
XQ	Execute Program
ZS	Zero Subroutine Stack

GENERAL CONFIGURATION

AL	Arm Latch
BN	Burn
CB	Clear Bit
CE	Configure Encoder Type
CN	Configure Switches and Stepper
CO	Configure I/O points (DB-10072 option only)
DA	Deallocate Arrays
DE	Define Dual Encoder Position
DL	Download
DM	Dimension Arrays
DP	Define Position
EA	Choose ECAM master
EB	Enable ECAM
ED	Edit Mode
EI	Enable Interrupts
EG	Engage ECAM
EM	Cam cycle command
EO	Echo Off
EP	Cam table interval and starting point
EQ	Disengage ECAM
ET	ECAM table entery
LS	List
MO	Motor Off
MT	Motor Type
OB	Define Output Bit
OP	Output Port
PF	Position Format
QU	Upload array
QD	Download array
RA	Record Array
RC	Record
RD	Record Data
RS	Reset
SB	Set Bit
UI	User Interrupt
UL	Upload
VF	Variable Format

CONTROL FILTER SETTINGS

DV	Damping for dual loop
FA	Acceleration Feedforward
FV	Velocity Feedforward
GN	Gain
IL	Integrator Limit
IT	Smoothing Time Constant - Independent
KD	Derivative Constant
KI	Integrator Constant
KP	Proportional Constant
OF	Offset
SH	Servo Here
TL	Torque Limit
TM	Sample Time
VT	Smoothing Time Constant - Vector
ZR	Zero

STATUS

RP	Report Command Position
RL	Report Latch
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

ERROR AND LIMITS

BL	Reverse Software Limit
ER	Error Limit
FL	Forward Software Limit
OE	Off on Error

EDITOR

ED	Edit mode
<return>	Save line
<cntrl> P	Previous line
<cntrl> I	Insert line
<cntrl> D	Delete line
<cntrl> Q	Quit Editor

ARITHMETIC FUNCTIONS

@SIN	Sine
@COS	Cosine
@ABS	Absolute value
@FRAC	Fraction portion
@INT	Integer portion
@RND	Round
@SQR	Square root
@IN	Return digital input
@AN	Return analog input
+	Add
-	Subtract
*	Multiply
/	Divide
&	And
|	Or
( )	Parentheses

Instruction Set Examples

Below are some examples of simple instructions. It is assumed your system is hooked-up and the motors are under stable servo control. Note, the colon (:) is returned by the controller and appears on the screen. You do not need to type the :.

:DP 0,0,0,0 <enter>	Define all axis positions as 0
:PF 6,6,6,6 <enter>	Define position format as 6 digits
:PR 100,200,300,400 <enter>	Specify X,Y,Z,W position command
:BG <enter>	Begin Motion
:TP <enter>	Tell Position
00100,00200,00300,00400	Returned Position data
:PR?,?,?,? <enter>	Request Position Command
00100,00200,00300,00400	Returned data
:BGX <enter>	Begin X axis only
:TPX <enter>	Tell X position only
00200	Returned position data
:tpx <enter>	Enter invalid command
?	Controller response
:TC1 <enter>	Request error code
1 Unrecognized command	Controller response
:VM XY <enter>	Specify Vector Mode for XY
:VS 10000 <enter>	Specify Vector Speed
:VP 2000,3000 <enter>	Specify Vector Segment
:VP 4000,5000 <enter>	Specify Vector Segment
:LE <enter>	End Vector
:BGS <enter>	Begin Coordinated Sequence
:TPXY <enter>	Tell X and Y position
004200,005200	Returned data

Programming Motion

Overview

The DMC-1000 provides several modes of motion, including independent positioning and jogging of any axis, coordinated motion, electronic cam motion, and electronic gearing. Each one of these modes is discussed in the following sections. Please note the DMC-1010 uses X-axis motion only, the DMC-1020 uses X and Y, the DMC-1030 uses X,Y and Z, and the DMC-1040 uses X,Y,Z and W.

The DMC-1050 uses A,B,C,D, and E. The DMC-1060 uses A,B,C,D,E, and F. The DMC-1070 uses A,B,C,D,E,F and G. The DMC-1080 uses the axes A,B,C,D,E,F,G, and H. For the DMC-1050 through DMC-1080, the axes A,B,C,D can be referred to interchangeably as X,Y,Z,W.

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

Example Application	Mode of Motion	Commands
Absolute or relative positioning where each axis is independent and follows prescribed velocity profile.	Independent Axis Positioning	PA,PR SP,AC,DC
Velocity control where no final endpoint is prescribed. Motion stops on Stop command.	Independent Jogging	JG AC,DC ST
Motion Path described as incremental position points versus time.	Contour Mode	CM CD DT WC
2,3 or 4 axis coordinated motion where path is described by linear segments.	Linear Interpolation	LM LI,LE VS VA,VD
2-D motion path consisting of arc segments and linear segments, such as engraving or quilting.	Coordinated Motion	VM VP CR VS VA,VD VE

Third axis must remain tangent to 2-D motion path, such as knife cutting.	Coordinated motion with tangent axis specified	VM VP CR VS,VA,VD TN VE
Electronic gearing where slave axes are scaled to master axis which can move in both directions.	Electronic Gearing	GA GR
Master/slave where slave axes must follow a master such as conveyer speed.	Electronic Gearing	GA GR
Moving along arbitrary profiles or mathematically prescribed profiles such as sine or cosine trajectories.	Contour Mode	CM CD DT WC
Teaching or Record and Play Back	Contour Mode with Automatic Array Capture	CM CD DT WC RA RD RC
Backlash Correction	Dual Loop	DE
Following a trajectory based on a master encoder position	Electronic Cam	EA EM EP ET EB EG EQ
Motion Smoothing	Applies to all of the above motion. Smooths motion to eliminate vibrations due to jerk (discontinuities in acceleration)	IT, VT

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 (PA) or relative position (PR), slew speed (SP), acceleration ramp (AC), and deceleration ramp (DC), for each axis. On begin (BG), the DMC-1000 profiler generates the corresponding trapezoidal or triangular velocity profile and position trajectory. A new command position along the trajectory is generated every sample period. Motion is complete when the last position command or target position is generated by the DMC-1000 profiler. The actual motor motion may not be complete at this point, 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. No axes specifier implies motion on all the axes. For the DMC-1080, ABCDEFGH axes specifiers are used where XYZ and W may be interchanged with ABCD.

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.

A new position target (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.

Independant Axis Command Summary

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
AM XYZW	Trippoint for profiler complete
MC XYZW	Trippoint for "in position"

For the DMC-1080:

Use a,b,c,d,e,f,g,h to specify axis data above.

The DMC-1000 also allows use of single axis specifiers such as PRY=2000 or SPH=10000.

Example - Absolute Position

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^2	Acceleration

Start X and Y motion at the same time. After 20 msec, start Z motion. If input 1 is high, stop Y motion.

```
#A                          Begin Program
PR 500,1000,100             Specify position
SP 10000,15000,5000         Specify speed
AC 500000,500000,500000     Specify acceleration
DC 500000,500000,500000     Specify deceleration
BG XY                       Begin X and Y
WT 20;BG Z                  Wait 20 msec and begin
JP #B,@IN[1]=0              Jump if input 1 is low
STY                         Stop Y
#B;EN                       End Program
```

Fig. 6.1 shows the velocity profiles for the X,Y and Z axis.

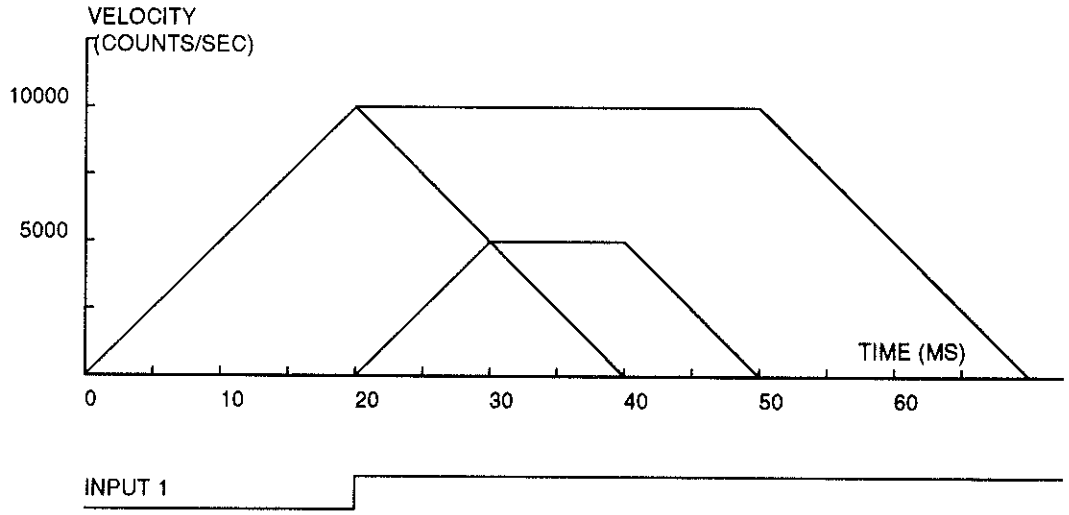


Figure 6.1 - Velocity Profiles of XYZ

Independent Jogging

In this mode, the user specifies the jog speed (JG), acceleration (AC), and the deceleration (DC) rate for each axis. On begin (BG), the motor accelerates up to speed and continues to jog at that speed until a new speed or stop (ST) command is issued. The direction of motion is specified by the sign of the JG parameters.

The jog mode of motion is very flexible because the speed, direction and acceleration can be changed during motion. The IP command can also be used to instantly change the motor position. Upon receiving this command, the motor will instantly try to servo 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.

It should be noted that the controller operates as a closed-loop position controller even while in the jog mode. The DMC-1000 converts the velocity profile into a position trajectory where a new position target is generated every sample period. This method of control results in precise speed regulation with phase lock accuracy.

Jogging Command Summary

JG +/-x,y,z,w	Specifies jog speed and direction
AC x,y,z,w	Specifies acceleration rate
DC x,y,z,w	Specifies deceleration rate
BG XYZW	Begins motion
ST XYZW	Stops motion
IP x,y,z,w	Increments position instantly

For the DMC-1080:

Use a,b,c,d,e,f,g,h to specify axis data above.

The DMC-1000 also allows use of individual axis specifiers such as JGY=2000 or ACH=400000.

Example - Jog in X only

Jog X motor at 50000 count/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
DC 20000,20000      Specify X,Z deceleration
JG 50000,,-25000    Specify X,Z speed and direction
BG X                Begin X motion
AS X                After X 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. Therefore, the calibration factor is 50000/2047 since the DMC-1000 uses a 12-bit ADC resulting in 2047 counts for 10 Volts.

```
#JOY                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-1000 provides a linear interpolation mode for 2,3 or 4 axes (up to 8 axes for the DMC-1080). Here, 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. Several 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 XYZW command selects the Linear Interpolation mode and axes for interpolation. For example, LM YZ selects only the Y and Z axes for linear interpolation. For the DMC-1080, use ABCDEFGH axis specifiers where XYZW may be used interchangeably with ABCD.

The LM command only needs to be specified once unless the axes for linear interpolation change, or another mode such as VM is given.

The LI x,y,z,w or LI a,b,c,d,e,f,g,h command 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 specified.

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 ST command causes a decelerated stop, while the AB command gives an instantaneous stop and aborts the program, while 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-1000 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, LI segments can be sent at PC bus speeds.

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.

Additional commands for linear interpolation are VS n, VA n, and VD n for specifying the vector speed, acceleration and deceleration. VT is used to set the S-curve smoothing constant for coordinated moves. The AV n command is the After Vector trippoint, which waits for the vector distance of n to occur.

For example, note the following program:

```
#Example
DP 0,0
LMXY
LI 5000,0
LI 0,5000
LE
VS 4000
BGS
AV 4000
VS 1000
AV 5000
VS 4000
EN
```

Here the XY system is required to perform 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, we can increase the speed, back to 4000 ct/s, with the tripopint AV 5000.

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 #Example can be written in the alternative form:

```
#ALT
DP 0,0
LMXY
LI 4000,0 <4000
LI 1000;0 < 1000
LI 0,5000 < 4000
LE
BGS
EN
```

The command VR n allows the feedrate, VS, to be scaled between 0 and 10 with a resolution of .0001. This command takes effect immediately and causes VS or < specifications to be scaled. This is a useful feature for feedrate override. VR does not ratio the accelerations. For example, VR .5 results in the specification VS 2000 to be divided in half.

The instruction AV can be used as an operand. _AV returns the distance along the motion sequence.

The instruction _VP returns the absolute coordinate of the last data point along the trajectory. This enables the host to command motion backward in case of tool break.

For example, note the program #Example shown above. Consider the first motion segment, where the X axis moves toward the point X=5000. Now suppose that when X=3000, the controller is interrogated.

The response to _AV will be 3000. The response to _CS is 0 and the responses to _VPX and _VPY are zeros for both.

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

It should be noted that the DMC-1000 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.

Command Summary - Linear Interpolation

LM XYZW	Specify axes for linear interpolation
LM ABCDEFGH	Specify axes for linear interpolation (DMC-1080)
LI x,y,z,w < n	Specify incremental distances relative to current position, and assign vector speed n.
LI a,b,c,d,e,f,g,h < n	
LM or LM?	Returns number of available spaces for linear segments in DMC-1000 sequence buffer. Zero means buffer full. 512 means buffer empty.
VS n	Specify vector speed
VA n	Specify vector acceleration
VD n	Specify vector deceleration
BGS	Begin Linear Sequence
CS	Clear sequence
_CS	Segment counter
_VPm	Return coordinate of last point, where m=X,Y,Z or W or A,B,C,D,E,F,G or H
LE	Linear End- Required at end of LI command sequence
_LE or LE?	Returns length of vector (resets after 2147483647)
AMS	Trippoint for After Sequence complete
AV n	Trippoint for After Relative Vector distance,n
_AV	Return distance travelled
VT	S curve smoothing constant for vector moves

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.

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

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-1000 from:

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

The resulting profile is shown in Figure 6.2.

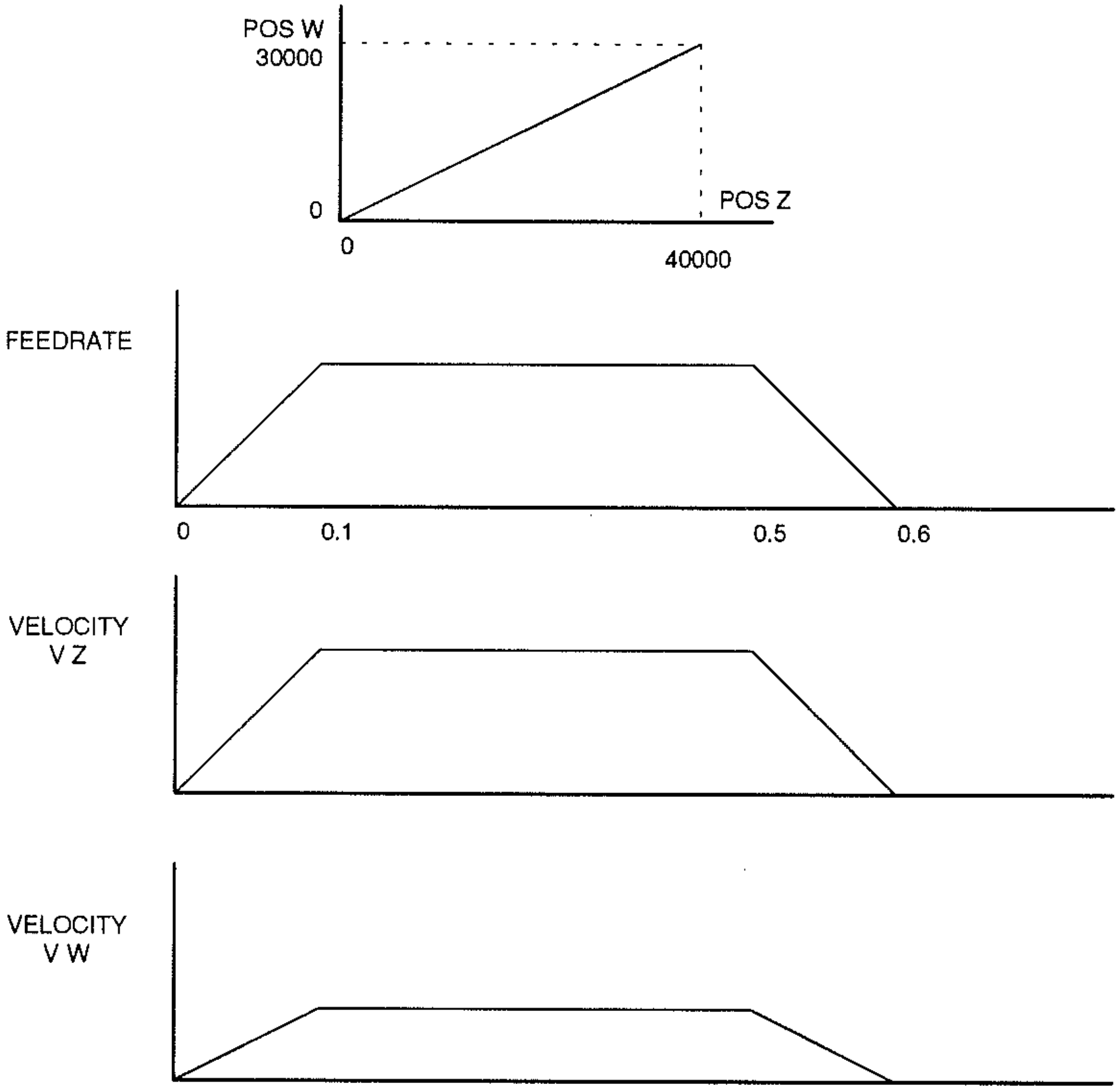


Figure 6.2 - Linear Interpolation

Example - Multiple Moves

Make a coordinated linear move in the XY plane. The Arrays VX and VY are used to store 750 incremental distances that have been filled by the program #LOAD.

```
#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
```

Coordinated Motion Sequences

The DMC-1000 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-1000 performs all the complex computations of linear and circular interpolation, freeing the host PC from this time intensive task.

The coordinated motion mode is similar to the linear interpolation mode. Here, 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.

The VM m,n,p command specifies the axes. m,n are the coordinated pair and p is the tangent. For example, VM X,W,Z selects the XW axes for coordinated motion and the Z-axis as the tangent. Commas are not required.

The motion segments are described by two commands, VP for linear and CR for circular segments. The VP x,y command specifies the end point coordinate of the linear segment, in reference to the starting point. CR r,θ,δ define a circular arc with a radius r, starting angle of θ, and a traversed angle δ. The notation for θ is that zero corresponds to the positive horizontal direction, and for both θ and δ, the counter-clockwise (CCW) rotation is positive.

Up to 511 segments of CR or VP may be given prior to the Begin Sequence (BGS) command. Once motion starts, additional segments may be added.

The Clear Sequence (CS) command can be used to remove VP and CR 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-1000 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 at PC bus speeds.

The instruction _CS returns the segment counter. It allows the host to determine the motion segment being executed.

Additional commands for coordinated motion are VS n, VA n and VD n for specifying the vector speed, acceleration, and deceleration. The AV n command is the After Vector trippoint, which waits for the vector relative distance of n to occur. VT is the s curve smoothing constant used with coordinated motion.

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.

The command VR r causes the vector speed to be scaled by the specified ratio r where r is between 0 and 10. VR does not ratio the accelerations.

The AV trippoint is useful in changing the parameters, such as the vector speed along the sequence.

When AV is used as an operand, _AV returns the distrance travelled along the sequence.

The instruction _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.

```
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.

_AV returns 2000

_CS returns 0

_VPX and _VPY return the absolute coordinate of the point A

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

_AV returns $4000+1500\pi+2000=10{,}712$

_CS returns 2

_VPX,_VPY return the coordinates of the point C

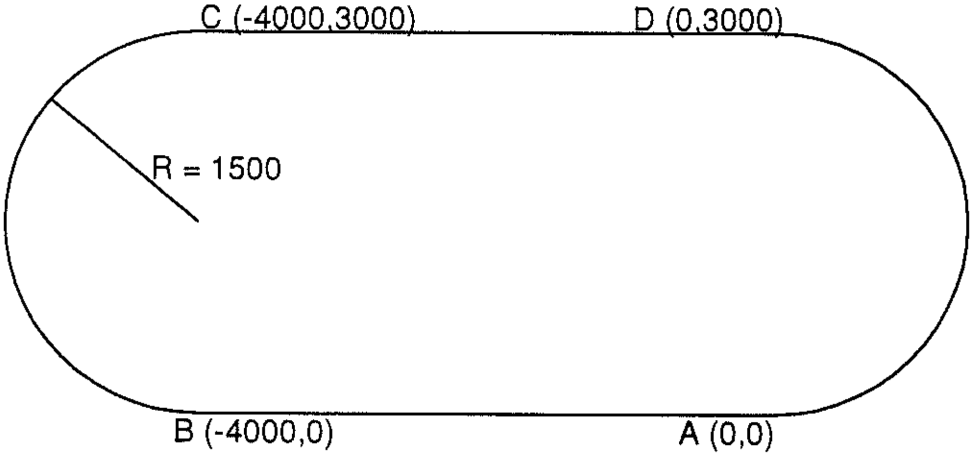


Figure 6.3 - The Required Path

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-1000 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:

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.

```
#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
```

Coordinated Motion Sequence Instructions - Summary

VM m,n,p	Specifies plane for the motion sequence such as X,Y or Z,W. p specifies tangent axis.
VP m,n	Return coordinate of last point, where m=X,Y,Z or W.
_VPm	Specifies the end point of a segment in reference to the starting point of the sequence.
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	Specifies vector speed or feedrate of sequence.
VA n	Specifies vector acceleration along the sequence.
VD n	Specifies vector deceleration along the sequence.
BGS	Begin motion sequence.
AV n	Trippoint for After Relative Vector distance, n.
_AV	Return distance travelled.
AMS	Holds execution of the next command until the Motion Sequence is completed.
_LM or LM?	Return number of available spaces for linear and circular segments in DMC-1000 sequence buffer. Zero means buffer is full. 512 means buffer is empty.
TN m,n	Tangent scale and offset.
ES m,n	Ellipse scale factor.
CS	Clear sequence.
_CS	Segment counter.
VT	s curve smoothing constant for coordinated moves

Electronic Gearing

This mode allows 1,2 or 3 axes (or 4,5,6,7 axes for the DMC-1080) 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-1080) 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.

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

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 = XC,YC,ZC or WC or AC, BC, CC, DC, EC, FC,GC,HC for commanded position as master n=S vector move for master
GR x,y,z,w	Sets gearing mode and gear ratio for slave axes. 0 disables electronic gearing for specified axis.
GR a,b,c,d,e,f,g,h	Sets gearing mode and gear ratio for slave axes. 0 disables electronic gearing for specified axis.
MR x,y,z,w	Trippoint for motion past assigned point in reverse direction. Only one field may be used.
MF x,y,z,w	Trippoint for motion past assigned point in forward direction. 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

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

Solution: Use a DMC-1030 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
```

In several applications where both the master and the follower are controlled by the DMC-1000 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 XC	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

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
```

which is equivalent to PR,10; BGY.

Often the correction is quite large. Such requirements are common on 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-1000 also provides a contouring mode. This mode allows any arbitrary position curve for 1,2,3 or 4 (5,6,7 or 8 axes for DMC-1080) axes to be prescribed which is ideal for following computer generated paths such as parabolic, spherical or user-defined profiles. Here, the path is not limited to straight line and arc segments. Also, the path length may be infinite.

The Contour Mode (CM) command specifies which axes are to be contoured. Any combination of 1,2,3 or 4 axes (5,6,7 or 8 axes for DMC-1080) may be used. For example, CMXZ specifies contouring on the X and Z axes. Axes non-contouring may be operated in other modes.

The contour is described by position increments, CD x,y,z,w over a time interval, DT n. For the DMC-1080, the contour is described by CD a,b,c,d,e,f,g,h.

The time interval must be 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.

Consider, for example, 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=138 at T=12ms
Point 4	X=302 at T=28ms

The same trajectory may be represented by the increments

Increment 1	DX=48	Time=4	DT=4
Increment 2	DX=90	Time=8	DT=8
Increment 3	DX=164	Time=16	DT=16

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, 22
CD 48;WC      Specifies first position increment
DT 3          Specifies second time interval, 23
CD 90;WC      Specifies second position increment
DT 4          Specifies the third time interval, 24
CD 164;WC     Specifies the third position increment
DT0;CD0       Exits contour mode
EN
```

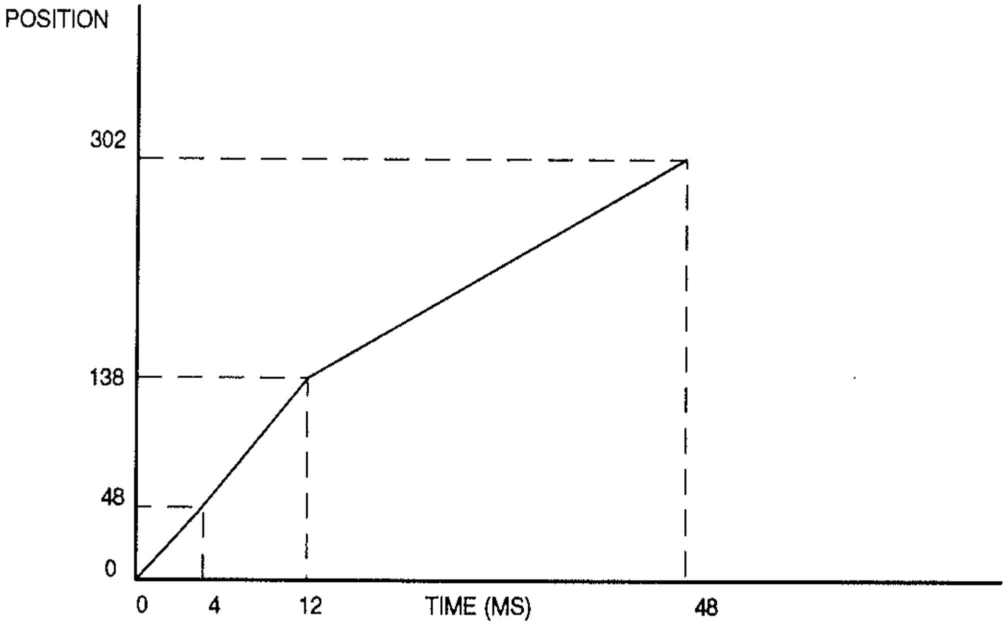


Figure 6.4 - The Required Trajectory

The command, WC, is used as a trippoint "When Complete". This allows the DMC-1000 to use the next increment only when it is finished with the previous one. Zero parameters for DT or 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 ?.

The command _CS, the segment counter, returns the number of the segment being processed. This information allows the host computer to determine when to send additional data.

Summary of Commands for Contour Mode:

CM XYZW	Specifies which axes for contouring mode. Any non-contouring axes may be operated in other modes.
CM ABCDEFGH	Contour axes for DMC-1080
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-1080 .
DT n	Specifies time interval 2n 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.
_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

Consider for example 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. When the position displacement is A counts in B milliseconds, the general expression for the velocity and position profile, where T is the time in milliseconds, is:

$$\omega = \tfrac{A}{B}\left(1 - \cos(2\pi/B)\right)$$

$$X = \tfrac{AT}{B} - \tfrac{A}{2\pi}\sin(2\pi/B)$$

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

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

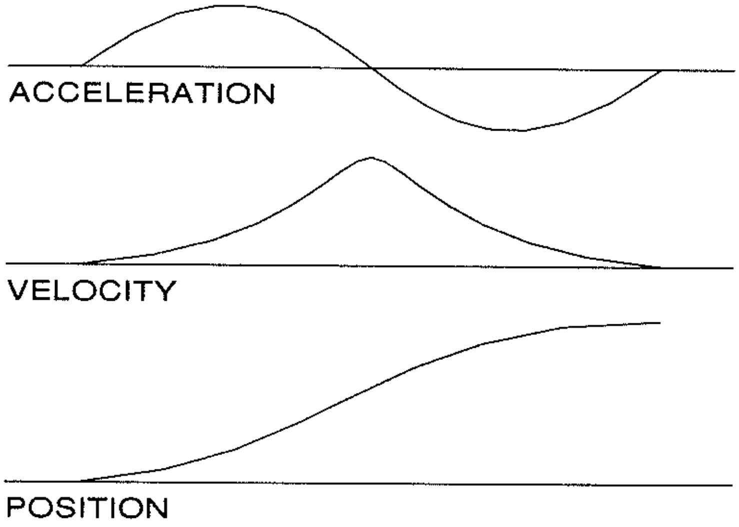


Figure 6.5 - Velocity Profile with Sinusoidal Acceleration

The DMC-1000 can compute trigonometric functions. However, the argument must be expressed in degrees. Accordingly, the equation of X is written as:

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

To generate an array, we compute the position value at intervals of 8 ms. This is stored at the array POS. Later, the difference between the positions is computed and is stored in the array DIF.

The program for storing the values is given below.

```
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 by using the DMC-1000 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-1040; 8 for DMC-1080)
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

Example:

```
#RECORD                  Begin Program
DM XPOS[501]             Dimension array with 501 elements
RA XPOS[]                Specify automatic record
RD _TPX                  Specify X position to be captured
MOX                      Turn X motor off
RC2                      Begin recording; 4 msec interval
#A;JP#A,_RC=1            Continue until done recording
#COMPUTE                 Compute DX
DM DX[500]               Dimension Array for DX
C=0                      Initialize counter
#L                       Label
D=C+1
DELTA=XPOS[D]-XPOS[C]    Compute the difference
DX[C]=DELTA              Store difference in array
C=C+1                    Increment index
JP #L,C<500              Repeat until done
#PLAYBCK                 Begin Playback
CMX                      Specify contour mode
DT2                      Specify time increment
I=0                      Initialize array counter
#B                       Loop counter
CD XPOS[I];WC            Specify contour data I=I+1 Increment array counter JP #B,I<500 Loop until done
DT 0;CD0                 End contour mode
EN                       End program
```

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

Dual Loop (Auxiliary Encoder)

The DMC-1000 provides an interface for a second encoder per axis. The second encoder may be mounted on the motor, the load or in any position.

The second encoder may be of the standard quadrature type, or it may be of the pulse and direction type. The controller also offers the provision for inverting the direction of the encoder rotation.

Auxiliary encoders can not be used for any axis configured as a stepper motor axis.

The configuration of the auxiliary encoder is done by the CE (Configure Encoder) command. This command configures both the main and the second encoder.

The command form is CE x,y,z,w or a,b,c,d,e,f,g,h for DMC-1080 where the parameters x,y,z,w each equals the sum of two integers m and n. m configures the main encoder and n configures the second encoder.

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
```

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

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

sets their initial values.

The positions of the auxiliary encoders may be interrogated with 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
```

Backlash Compensation

The dual loop methods can be used for backlash compensation. This can be done by two approaches:

Continous dual loop

Sampled dual loop

To illustrate the problem, consider that the coupling between the motor and the load has a backlash. The approach is to mount position encoders on both the motor and the load.

The continous 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 continously.

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.

Continous Dual Loop - Example

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.

Sampled Dual Loop - Example

Run a linear slide by a rotary motor via a lead screw. As the lead screw has a backlash, it is necessary to use a linear encoder to monitor the position of the slide. In addition, 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. Let the required motion distance be one inch, and assume 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	

Motion Smoothing (S curve profiling)

The DMC-1000 controller allows the smoothing of the velocity profile to reduce the mechanical vibration of the system. The resulting velocity profile is known as S curve.

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

The smoothing is accomplished by filtering the acceleration profile. The degree of the smoothing is specified by the commands:

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

IT is used for smoothing independent moves of the type JG, PR, PA, whereas 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, where 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 diagrams illustrate the effect of the smoothing. Fig. 6.6 shows the trapezoidal velocity profile and the modified acceleration and velocity.

Also 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

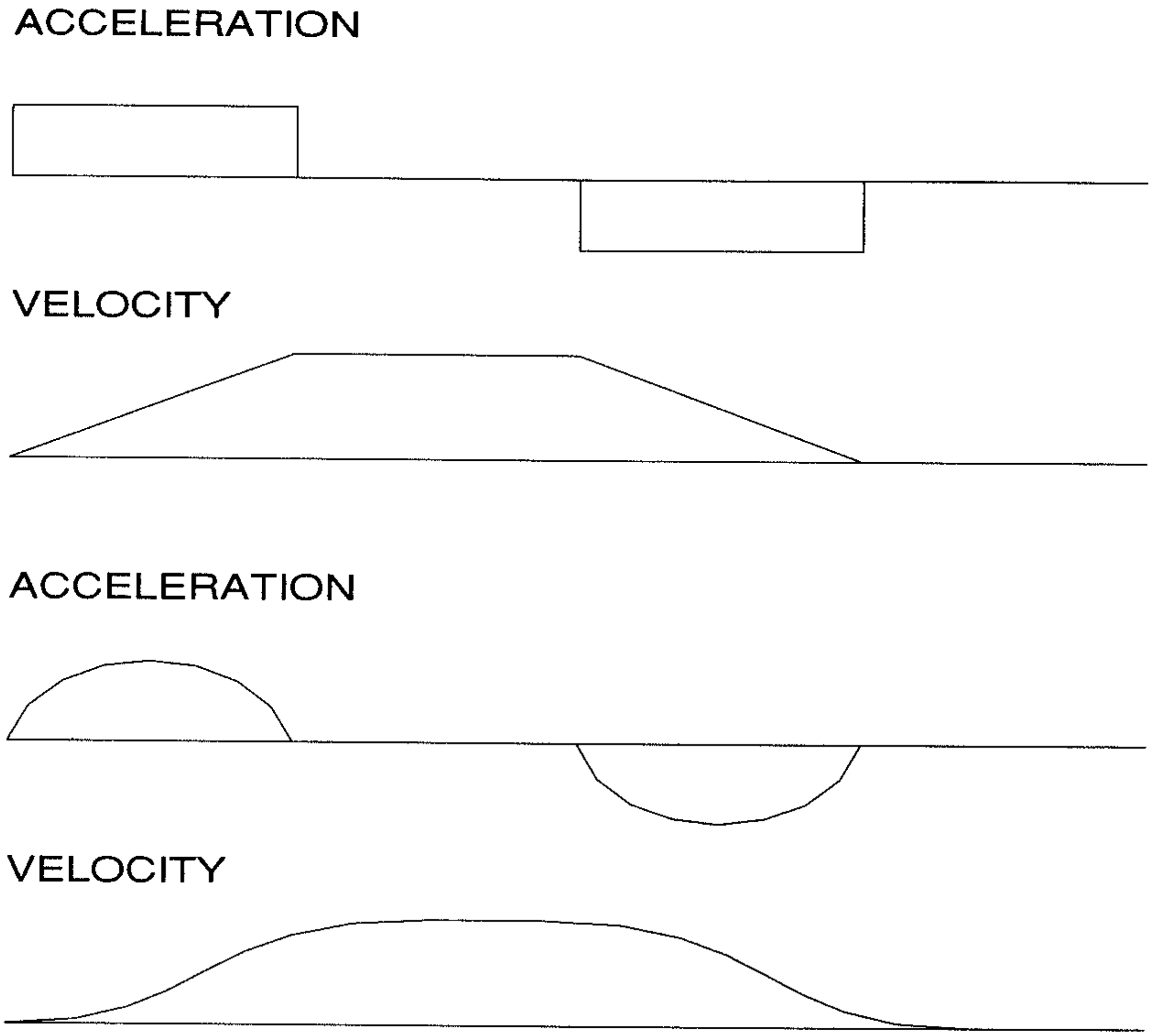


Figure 6.6 - Trapezoidal velocity and smooth velocity profiles

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) 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 HM command and BG command 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-1000 defines the home position (0) as the position at which the index was detected.

Example:

```
#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
```

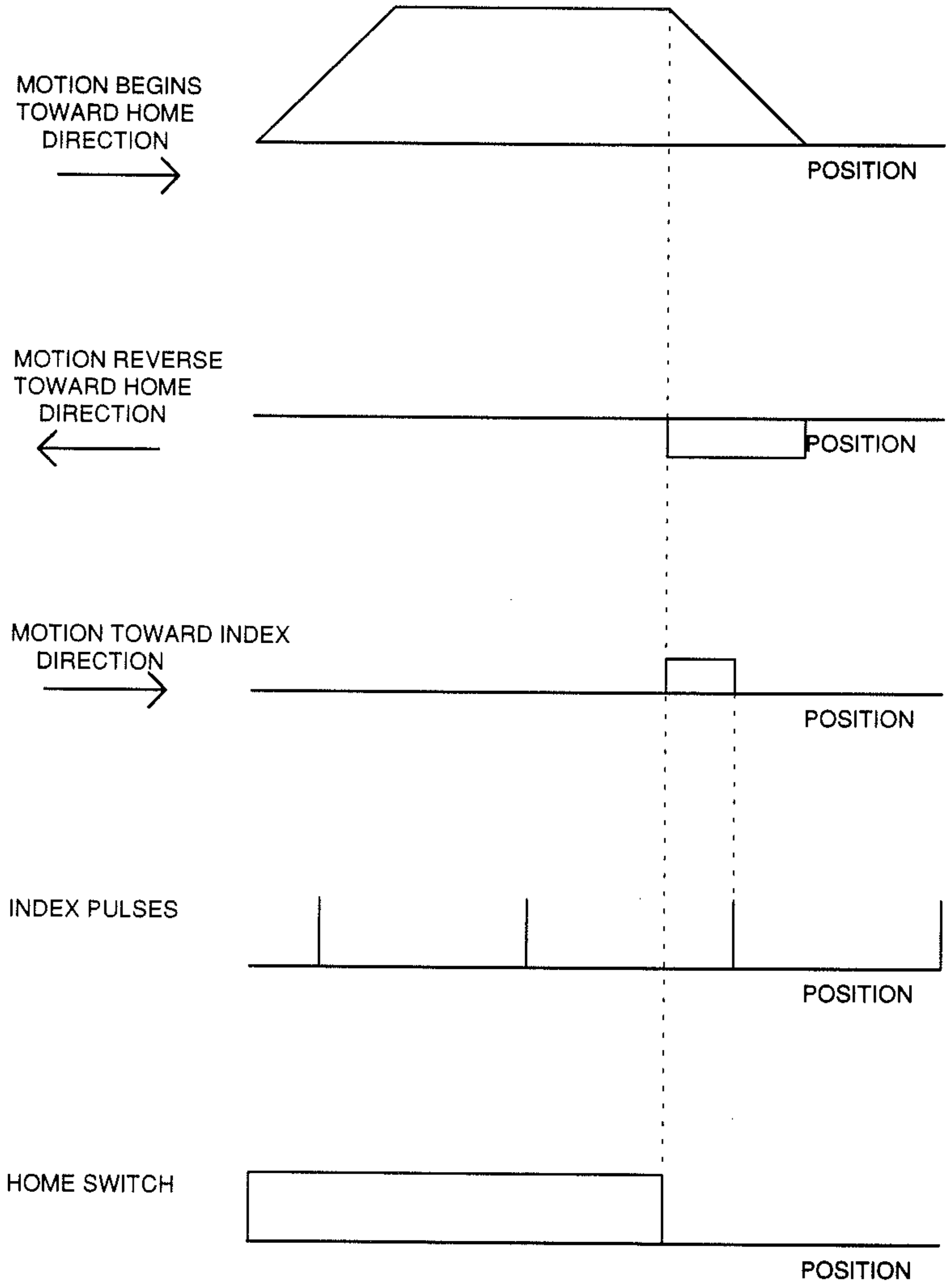


Figure 6.7 - Motion intervals in the Home sequence

High Speed Position Capture

Often it is desirable to capture the position precisely for registration applications. The DMC-1000 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 signal. The external signal is input at inputs 1 through 4, and 9 through 12 on the DMC-1080.

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

The DMC-1000 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-1080, 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.

Example:

```
#Latch              Latch program
JG,5000             Jog Y
BG Y                Begin Y
AL Y                Arm Latch
#Wait               Loop for Latch=1
JP #Wait,_ALY=1     Wait for latch
Result=_RLY         Report position
Result=             Print result
EN                  End
```

Electronic Cam

The electronic cam is a motion control mode which enables the periodic synchronization of several axes of motion when one of the axes is independent and is not necessarily driven by the controller.

The electronic cam is a more general type of electronic gearing which allows a table-based relationship between the axes. It allows synchronizing all the controller axes. Therefore, the DMC-1080 controller may have one master and up to seven slaves. To simplify the presentation, however, we will limit the description to a 4-axis controller.

The first step in the electronic cam mode is selecting the master axis. This is done with the instruction

EAp where p = X,Y,Z,W
p is the selected master axis

To illustrate the procedure of setting the cam mode, consider the cam relationship for the slave axis Y, when the master is X. Such a graphic relationship is shown in Figure 6.8.

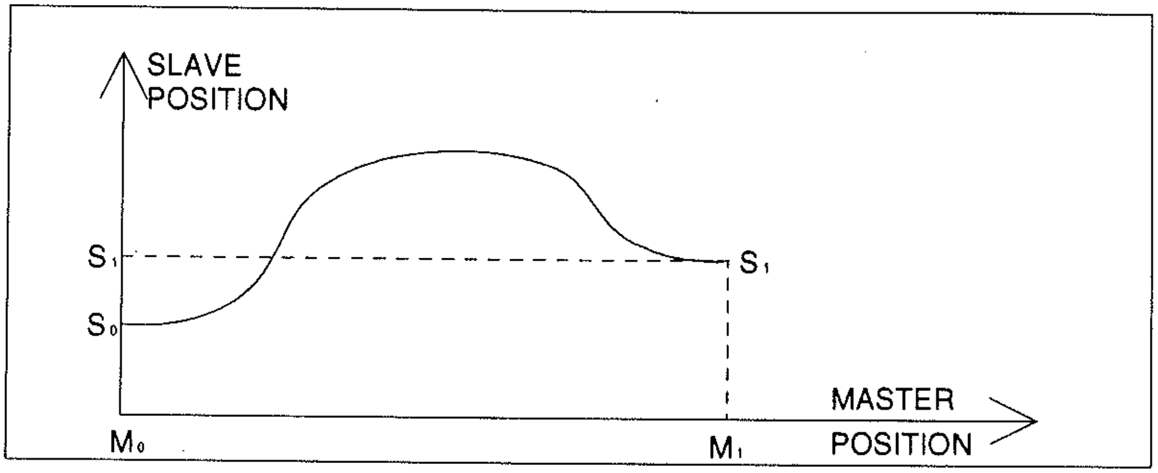


Figure 6.8: Electronic Cam Cycle

The cam cycle starts at the master position M_0 and ends at the master position M_1. This implies that the cycle for the master, CM, is

$$CM = M_1 - M_0$$

The slave axis must equal S_0. when the master position is $M_{0,}$ and S_1 when the master position equals M_1. Over one cycle, the change in slave position, CS, equals

$$CS = S_1 - S_0$$

In the cam mode, the positions of the master and the slave are redefined to fit the values shown in Figure 1. This implies that if the master position increases beyond M_1, the master position is decreased by CM and the slave position is decreased by CS. On the other hand, if the master moves in the negative direction through the point M_0, the master position is increased by CM and the slave position is increased by CS. To specify the values of CM and CS, we use the instruction

EM x,y,z,w

where the values x,y,z,w are the CM or CS values for the corresponding axes.

The range of CM is an integer between 1 and 8,388,607 and the range of CS is an integer between 0 and 2,147,483,647. If CS is negative, its absolute value is specified.

For example, suppose that the cam relationship is as expressed in Fig 6.9.

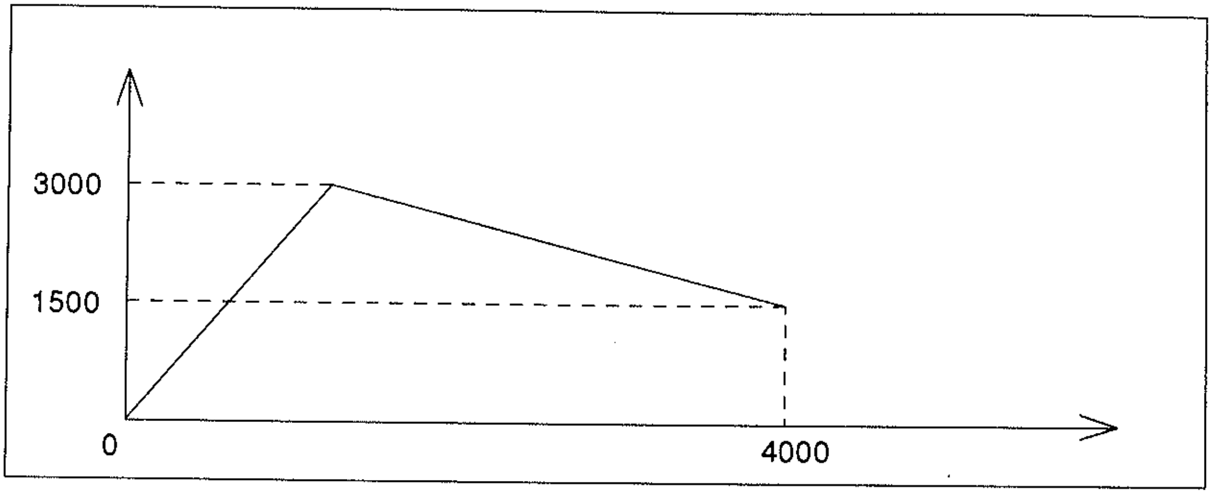


Figure 6.9: Electronic Cam Example

Since the cycles are 4000 and 1500 for X and Y respectively, the command is

EM 4000,1500

The points of the master axis are defined by a table at constant intervals of m counts, with a possible position offset n. These parameters are defined with the instruction

EP m,n

For example, EP 100,500 indicates that the table starts at the master position of 500 and that the following master points are 600, 700, etc.

Finally, the table parameters are defined with the instruction

ET [n] = x,y,z   where n starts at 0 and may go on up to 256

For example,

ET [7] = 100, 300, -200

defines a row in the table. It indicates that the positions X = 100, Y = 300 and
Z = -200 must be synchronized.

Note that only the slave points must be given. The master points are defined by the EP instruction.

To illustrate the process, consider the simple example where X, the master, has a cycle of 4000 counts and Y, the slave, must be synchronized with a gear ratio of 1 during the first half of the cycle and a zero ratio during the second half.

To construct the table we start by selecting the X axis as the master

EAX

Define the cycles as

EM 4000, 2000

Since the table is quite simple, it can be defined by a few points with an interval of 1000.

EP 1000, 0

The first point, when X = 0 is

ET [0] = , 0

It is followed by:

ET[1] = ,1000
ET[2] = , 2000
ET[3] = , 2000
ET[4] = , 2000

Once the cam mode is defined, it can be enabled or disabled with the instruction

EB n where n= 1 enables the cam mode and n = 0 disables it

When the cam mode is enabled, the position of the master is monitored and is redefined as a value within the cycle.

To engage the slave axes at a programmed point, we use the command

EG x,y,z,w where x,y,z,w are the master positions at which the corresponding slave axes must be engaged

If the value of any parameter is outside the range specified by the master cycle, the cam engages that axis immediately. When a slave motor is engaged, its position is redefined to fit within the cycle.

To stop a slave axis, use the instruction

EQ x,y,z,w where x,y,z,w are the master positions at which the corresponding slave axes must be disengaged.

This disengages the slave axis at a specified master position. If the parameter is outside the master cycle, the stopping is instantaneous.

Programmed start and stop can be used only when the master moves forward.

To illustrate the complete process, consider the cam relationship described by the equation:

$$Y = 0.5 * X + 100 \sin (0.18*X)$$

where X is the master, with a cycle of 2000 counts.

The cam table can be constructed manually, point by point, or automatically by a program. The following program includes the set-up.

The instruction EAX defines X as the master axis. The cycle of the master is
CM = 2000. Over that cycle, X varies by CS = 1000. This leads to the instruction EM 2000,1000.

Suppose we want to define a table with 100 segments. This implies increments of 20 counts each. If the master points are to start at zero, the required instruction is EP 20,0.

The following routine computes the table points. As the phase equals 0.18X and X varies in increments of 20, the phase varies by increments of 3.6°. The program then computes the values of Y according to the equation and assigns the values to the table with the instruction ET[N] = ,Y.

Instruction	Interpretation
#SETUP	Label
EAX	Select X as master
EM 2000,1000	Cam cycles
EP 20,0	Master position increments
N = 0	Index
#LOOP	Loop to construct table from equation
*P = N*3.6*	Note 3.6 = 0.18*20
*S = @SIN [P] *100*	Define sine position
*Y = N *10+S*	Define slave position
ET [N] =, Y	Define table
N = N+1	
JP #LOOP, N<=100	Repeat the process
EN	

Now suppose that the slave axis is engaged with a start signal, input 1, but that both the engagement and disengagement points must be done at the center of the cycle: X = 1000 and Y = 500. This implies that Y must be driven to that point to avoid a jump.

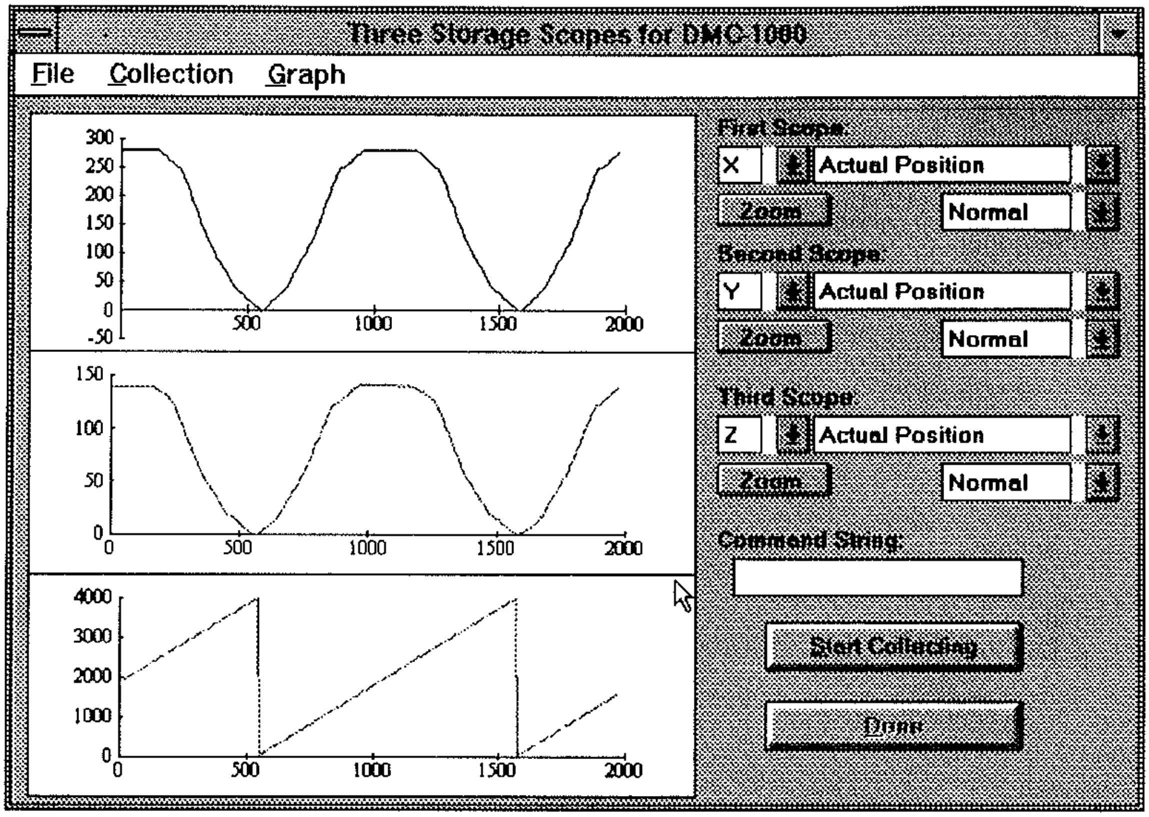
This is done with the program:

Instruction	Interpretation
#RUN	Label
EB1	Enable cam
PA,500	starting position
SP,5000	Y speed
BGY	Move Y motor
AM	After Y moved
AI1	Wait for start signal
EG,1000	Engage slave
AI - 1	Wait for stop signal
EQ,1000	Disengage slave
EN	End

The following example illustrates a cam program with a master axis, Z, and two slaves, X and Y.

Instruction	Interpretation
#A;V1=0	Lable; Initialize variable
PA 0,0;BGXY;AMZXY	Go to position 0,0 on X and Y axes
EA Z	Z axis as the Master for ECAM
EM 0,0,4000	Change for Z is 4000, zero for X, Y
EP400,0	ECAM interval is 400 counts with zero start
ET[0]=0,0	When master is at 0 position; 1st point.
ET[1]=40,20	2nd point in the ECAM table
ET[2]=120,60	3rd point in the ECAM table
ET[3]=240,120	4th point in the ECAM table
ET[4]=280,140	5th point in the ECAM table
ET[5]=280,140	6th point in the ECAM table
ET[6]=280,140	7th point in the ECAM table
ET[7]=240,120	8th point in the ECAM table
ET[8]=120,60	9th point in the ECAM table
ET[9]=40,20	10th point in the ECAM table
ET[10]=0,0	Starting point for next cycle
EB 1	Enable ECAM mode
JGZ=4000	Set Z to jog at 4000
EG 0,0	Engage both X and Y when Master = 0
BGZ	Begin jog on Z axis
#LOOP;JP#LOOP,V1=0	Loop until the variable is set
EQ2000,2000	Disengage X and Y when Master = 2000
MF 2000	Wait until the Master goes to 2000
ST Z	Stop the Z axis motion
EB 0	Exit the ECAM mode
EN	End of the program

The above example shows how the ECAM program is structured and how the commands can be given to the controller. The next page provides the results captured by the WSDK-1000 program showing how the motion will be seen during the ECAM cycles. The first graph is for the X axis, the second graph shows the cycle on the Y axis and the third graph shows the cycle of the Z axis.



First Scope: X Actual Position Normal

Second Scope: Y Actual Position Normal

Third Scope: Z Actual Position Normal

Command String:

Application Programming

Introduction

The DMC-1000 programming language is a powerful language that allows users to customize a program to handle their particular application. Complex programs can be downloaded into the DMC-1000 memory for later execution. Utilizing the DMC-1000 to execute sophisticated programs frees the host computer for other tasks. However, the host computer can still send commands to the controller at any time, even while a program is being executed.

In addition to standard motion commands, the DMC-1000 provides several commands that allow the DMC-1000 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-1000 provides 126 (510 with -MX option) user-defined variables, arrays and arithmetic functions. For example, the length in a cut-to-length operation can be specified as a variable in a program and then be assigned by an operator.

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

Using the DMC-1000 Editor to Enter Programs

Application programs for the DMC-1000 may be created and edited either locally using the DMC-1000 editor or remotely using another editor and then downloading the program into the controller. (Galil's COM DISK and SDK-1000 software provides an UPLOAD and DOWNLOAD utility as part of the Terminal Emulator).

The DMC-1000 provides a line Editor for entering and modifying programs. The Edit mode is entered with the ED instruction. (Note: The ED command can only be given when the controller is in the non-edit mode, which is signified by a colon prompt).

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 last program in memory. If desired, the user can edit a specific line number or label by specifying a line number or label following ED.

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

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 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 labelled 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-1000 will return a colon.

After the Edit session is over, the user may list the entered program using the LS command. If no operand follows the LS command, the entire program will be listed. The user can start listing at a specific line or label using the operand n. A command and new line number or label following the start listing operand specifies the location at which listing is to stop.

Example:

Instruction	Interpretation
:LS	List entire program
:LS 5	Begin listing at line 5
:LS 5,9	List lines 5 thru 9
:LS #A,9	List line label #A thru line 9

Program Format

A DMC-1000 program consists of several DMC-1000 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-1000 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.

All DMC-1000 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 which may be defined is 126 (254 labels are allowed with the DMC-1080 and 510 with the _MX memory expansion option).

Valid labels

#BEGIN

#SQUARE

#X1

#BEGIN1

Invalid labels

#1Square

#123

There are also some special labels, which are used to define input interrupt subroutines, limit switch subroutines, error handling subroutines, and command error subroutines.

Special Labels

#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

Example Program:

#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.

Executing Programs - Multitasking

Up to four programs can run independently. The programs, called threads, are numbered 0 through 3, where 0 is the main one.

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

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

2. In a case of interrupts, due to inputs, limit switches, position errors or command errors, it is the program in thread 0 which jumps to those subroutines.

The execution of the various programs is done with the 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 functions can be performed by an executing program.

Multitasking is useful for executing independent operations such as PLC functions that occur independently of motion. The example below produces a waveform on Output 1 independent of a move.

```
#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. #TASK1 is executed within TASK2.

Debugging Programs

The DMC-1000 provides trace and error code commands which are used in debugging programs. The trace command may be activated using the command, TR1. This command causes each line in a program to be sent out to the communications port immediately prior to execution. The TR1 command is useful for debugging programs. TR0 disables the trace function. The TR command may also be included as part of a program.

If there is a program error, the DMC-1000 will halt program execution at the line number at which an error occurs and display the line. 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 error condition as follows:

Error Code	Error String
1	Unrecognized command
2	Command only valid from program
3	Command not valid in program
4	Operand error
5	Input buffer full
6	Number out of range
7	Command not valid while running
8	Command not valid while not running
9	Variable error
10	Empty program line or undefined label
11	Invalid label or line number
12	Subroutine more than 8 deep
13	JG only valid when running in jog mode
14	EEPROM check sum error
15	EEPROM checkwrite error
16	IP incorrect sign during position move or IP given during forced deceleration
17	D, BN and DL not valid while program running
18	ommand not valid when contouring
20	Begin not valid with motor off
21	Begin not valid while running
22	Begin cannot be executed because of Limit Switch
24	Begin not valid because no sequence defined
25	Variable not given in IN command
29	Not valid during coordinated move
30	Sequence segment too short
31	Total move distance in a sequence > 2 billion
32	More than 511 segments in a sequence
41	Contouring record range error
42	Contour data being sent too slowly
46	Gear axis both master and follower
47	Gearing and coordinated moves cannot run simultaneously
50	Not enough fields
51	Question mark not valid
52	Missing " or string too long

53	Error in {}
54	Question mark part of string
55	Missing [or []
56	Array index invalid or out of range
57	Bad function or array
58	Unrecognized command in a command response (i.e._GNX)
59	Mismatched parentheses
60	Download error - line too long or too many lines
61	Duplicate or bad label
65	IN command must have a comma
66	Array space full
67	Too many arrays or variables
71	IN only valid in task #0
80	Record mode already running
81	No array or source specified
82	Undefined array
90	Only X Y Z W valid operand
95	TM too large for stepper pulse
96	SM jumper needs to be installed for stepper motor operation

TC0 or TC will return the error code only without the text message.

Example:

```
: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
:ED 3                        Edit Line 3
003 AMX;PR5000;BGX           Add After Motion Done
<cntrl> Q                    Quit Edit Mode
:XQ #A                       Execute #A
```

Program Flow Commands

The DMC-1000 provides several instructions that control program flow. Normally, the DMC-1000 program sequencer executes instructions in a program sequentially. Program Flow commands, however, may be used to redirect program flow. A summary of these commands is given below and they are detailed in the following sections.

Program Flow Command Summary

JP	Conditional Jump
JS	Conditional Jump to Subroutine
AD	After Distance Trigger
AI	After Input Trigger
AM	After Motion Complete Trigger
AP	After Absolute Position Trigger
AR	Relative Distance Trigger
AS	After Speed Trigger
MF	Trigger Forward motion
MR	Trigger Reverse motion
MC	Trigger "In position" trigger (TW x,y,z,w sets timeout for in-position)
AV	After Vector Distance Trigger
WC	Wait for Contour Data
WT	Wait for time to elapse

Event Triggers & Trippoints

To function independently from the host computer, the DMC-1000 can be programmed to make decisions based on the occurrence of an event. Such events include waiting for motion to be complete, waiting for a specified amount of time to elapse, or waiting for an input to change logic levels.

The DMC-1000 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-1000 can make decisions based on its own status or external events without intervention from a host computer.

DMC-1000 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 or aux. inputs.
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 or aux. inputs.
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-1040. n=1 through 24 for DMC-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.

```
#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.

```
#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.

```
#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.

```
#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

```
#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.

```
#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

```
#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
```

Define 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.

```
#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     Loop
EN
```

Conditional Jumps

The DMC-1000 provides Conditional Jump (JP) and Conditional Jump to Subroutine (JS) instructions for branching to a new program location based on a specified condition. Unlike event triggers, the conditional jump instruction does not halt the program sequence. Instead, it tests to see if a condition is satisfied and then branches to a new location or subroutine. (A subroutine is a group of commands defined by a label and EN command. After all the commands in the subroutine are executed, a return is made to the main program). If the condition is not satisfied, the program sequence continues to the next program line.

The JP and JS instructions have the following format:

Format:	Meaning
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. The destination is where the program sequencer jumps to if the specified condition is satisfied. The comma designates "IF". The logical condition tests two operands with logical operators. The operands can be any valid DMC-1000 numeric operand, including variables, array elements, numeric values, functions, keywords, and arithmetic expressions.

Logical operators:

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

Operands:

Type	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

The jump statement may also be used without a condition.

Example conditional jump statements are given below:

Conditional	Meaning
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

Conditional jumps are useful for testing events in real-time. They allow the DMC-1000 to make decisions without a host computer. For example, the DMC-1000 can decide between two motion profiles based on the state of an input line. Or, the DMC-1000 can keep track of how many times a motion profile is executed.

Example:

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

```
#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 thru loop
EN                  End Program
```

Subroutines

A subroutine is a group of instructions beginning with a label and ending with an END (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:

An example of a subroutine to draw a square 500 counts per side is given below. The square is drawn at vector position 1000,1000.

```
#M                       Begin Main Program
CB1                      Clear Output Bit 1 (pick up pen)
VP 1000,1000;LE;BGS;AMS  Define vector position; move pen
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
V1=-V1;JS #L             Switch direction
EN                       End subroutine
#L;PR V1,V1;BGX          Define X,Y; Begin X
AMX;BGY;AMY              After motion on X, Begin Y
EN                       End subroutine
```

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-1000 program sequences. The DMC-1000 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:

#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
#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.

Examples:

Example - Limit Switch:

This program prints a message upon the occurrence of a limit switch. Note, for the #LIMSWI routine to function, the DMC-1000 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.

```
:ED                       Edit Mode
000 #LOOP                 Dummy Program
001 JP #LOOP;EN           Jump to Loop
002 #LIMSWI               Limit Switch Label
003 MG "LIMIT OCCURRED"   Print Message
004 RE                    Return to main program
<control> Q               Quit Edit Mode
:XQ #LOOP                 Execute Dummy Program
:JG 5000                  Jog
:BGX                      Begin Motion
```

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

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

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

Example - Position Error

```
:ED                               Edit Mode
000 #LOOP                         Dummy Program
001 JP #LOOP;EN                   Loop
002 #POSERR                       Position Error Routine
003 V1=_TEX                       Read Position Error
004 MG "EXCESS POSITION           Print Message
ERROR"
005 MG "ERROR=",V1=               Print Error
006 RE                            Return from Error
<control> Q                       Quit Edit Mode
:XQ #LOOP                         Execute Dummy Program
:JG 100000                        Jog at High Speed
:BGX                              Begin Motion
```

Now, if the position error on the X axis exceeds that 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).

Input Interrupt Example:

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

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

Motion Complete Timeout Example

```
#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 if the axis does not make it to the position within 1 second of the end of the profiled move.

Bad Command Example

```
#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

For manipulation of data, the DMC-1000 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. Parentheses can be used and nested four deep. Calculations within a parentheses have precedence.

Examples:

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

The DMC-1000 also provides the following functions:

Function	Command	Meaning
@ABS	Absolute Value*	
@SIN	Sine*	
@COS	Cosine*	
@COM	2's Complement	
@FRAC	Fraction	
@INT	Integer	
@RND	Rounds number .5 and up to next Integer	
@IN[n]	Read digital input n	
@AN[n]	Read analog input n	
@SQR[n]	Square Root Function; Accuacy is +/-.0004	

Functions may be combined with mathematical expressions. The order of execution is from left to right. The units of the SIN and COS functions are in degrees with resolution of 1/128 degrees. The values can be up to +/-4 billion degrees.

Example:

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=@AN[5]	The variable, V4, is equal to the digital value of analog input 5.

Variables

Many motion applications include parameters that are variable. For example, a cut-to-length application often requires that the cut length be variable. The motion process is the same, however, the length is changing.

To accommodate these applications, the DMC-1000 provides for the use of both numeric and string variables. A program can be written in which certain parameters, such as position or speed, are defined as variables. The variables can later be assigned by the operator or determined by the program calculations.

Example:

PR POSX	Assigns variable POSX to PR command
JG RPMY*70	Assigns variable RPMY multiplied by 70 to JG command.

Programmable Variables

The DMC-1000 allows the user to create up to 126 variables. 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. Examples of valid and invalid variable names are:

Valid Variable Names

POSX

POS1

SPEEDZ

Invalid Variable Names

1POS

123

SPEED Z

It is recommended that variable names not be the same as DMC-1000 instructions. For example, PR is not a good choice for a variable name.

The range for numeric variable values is 4 bytes of integer (2^{31})followed by two bytes of fraction (+/- 2,147,483,647.9999).

String variables can contain up to six characters which must be in quotation. Example: VAR="STRING".

Numeric values can be assigned to programmable variables using the equal sign. Assigned values can be numbers, internal variables and keywords, and functions. String values can be assigned to variables using quotations.

Any valid DMC-1000 function can be used to return a value such as V1=@ABS[V2] or V2=@IN[1]. Arithmetic operations are also permitted.

Example:

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

Variable values may be assigned to controller parameters such as GN or PR. Here, an equal is not used. For example:

PR V1	Assign V1 to PR command
SP VS*2000	Assign VS*2000 to SP command

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

```
#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
```

Internal Variables & Keywords

Internal variables allow motion or status parameters from DMC-1000 commands to be incorporated into programmable variables and expressions. Internal variables are designated by adding an underscore (_) prior to the DMC-1000 command. DMC-1000 commands which can be used as internal variables are listed in the Command Reference as "Used as an Operand" .. Yes.

Most DMC-1000 commands can be used as internal variables. Status commands such as Tell Position return actual values, whereas action commands such as GN or SP return the values in the DMC-1000 registers. The X,Y,Z or W or A,B,C,D,E,F,G,H for the DMC-1080, axis designation is required following the command.

Examples:

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.

Internal variables 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 DMC-1000 also provides a few keywords which give access to internal variables that are not accessible by standard DMC-1000 commands.

Keyword	Function
BGX or _BGY or _BGW	Motion Done if 1. Moving if 0.
LFX or _LFY or _LFZ or_LFW	Forward Limit (equals 0 or 1)
LRX or _LRY or _LRZ or LRW	Reverse Limit (equals 0 or 1)
TIME	Free-Running Real Time Clock* (off by 2.4% - Reset on power-on). Note: TIME does not use _.
HMX or _HMY or _HMZ or HMW	Home Switch (equals 0 or 1)

Examples:

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

Example Program:

```
#TIMER                  Timer
INITIME=TIME            Initialize time variable
PR50000;BGX             Begin move
AMX                     After move
ELAPSED=TIME-INTIME     Compute elapsed time
EN                      End program
#LIMSWI                 Limit Switch Routine
JP #FORWARD,_LFX=0      Jump if Forward Limit
AMX                     Wait for Motion Done
PR 1000;BGX;AMX         Move Away from Reverse Limit
JP #END                 Exit
#FORWARD                Forward Label
PR -1000;BGX;AMX        Move Away from Forward Limit
#END                    Exit
RE                      Return to Main Program
```

Arrays

For storing and collecting numerical data, the DMC-1000 provides array space for 1600 elements (8000 elements with DMC-1080 and DMC-1040-MX). 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 learning a position trajectory and later playing it back.

Defining Arrays

An array is defined by a name and number of entries using the DM command. The name can contain up to eight characters, starting with an uppercase alphabetic character.

The number of entries in the defined array is enclosed in [ ].

Up to 14 different arrays (30 with DMC-1080 and DMC-1040-MX) may be defined. The arrays are one dimensional.

Example:

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

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).

Array space may be deallocated using the DA command followed by the array name. DA*[0] deallocates all the arrays.

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.

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: Remember to define arrays using the DM command before assigning entry values.

Examples:

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.

An array element number can also be a variable. This allows array entries to be assigned sequentially using a counter.

For example:

#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.

Arrays may be uploaded and downloaded using the QU and QD commands.

```
QU array[],start,end,comma

QD array[],start,end
```

where array is an array name such as A[].

Start is the first element of array (default=0)

End is the last element of array (default=last element)

Comma -- if comma is a 1, then the array elements are separated by a comma. If not a 1, then the elements are separated by a carriage return.

The file is terminated using <control>Z, <control>Q, <control>D or \

Automatic Data Capture into Arrays

The DMC-1000 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 (eight types for DMC-1080) can be captured and stored in four arrays (eight arrays for DMC-1080). The capture rate or time interval may be specified. Recording can done as a one time event or as a circular continuous recording.

Commands used:

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_TI,TPX,SHZ,_TSY — Selects the type of data to be recorded. See the table below for the various types of data. The order of data type is important and corresponds with the order of n,m,o,p arrays in the RA command. In this example, the _TI input data is stored in the first array selected by 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 2n 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 continuoudly in a circular manner. _RD is the recording pointer and indicates the address of the next array element. n=0 stops recording.

RC? or V=_RC — Returns a 0 or 1 where, 0 denotes not recording, 1 specifies recording in progress

V=_RD — Return address of next array element.

Data Types for Recording

_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-1080.

Example - Recording into An Array

During a position move, store the X and Y positions and position error every 2 msec.

```
#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
```

Input and Output of Data

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 formatting string variables, the {Sn} specifier is required where n is the number of characters, 1 thru 6. Example:

```
MG STR {S3}
```

The above statement returns 3 characters of the string variable named STR.

Numeric data may be formatted using the {Fn.m} expression following the completed MG statement. {$n.m} formats data in HEX instead of decimal. Example:

```
MG "The Final Value is", RESULT {F5.2}
```

The above statement sends the message:

The Final Value is xxxxx.xx

The actual numerical value for the variable, RESULT, is substituted with the format of 5 digits to the left of the decimal and 2 to the right.

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 message command normally sends a carriage return and line feed following the statement. The carriage return and the line feed may be suppressed by sending {N} at the end of the statement. This is useful when a text string needs to surround a numeric value.

Example:

```
#A
JG 50000;BGX;ASX
MG "The Speed is", _TVX {F5.1} {N}
MG "counts/sec"
EN
```

When #A is executed, the above example will appear on the screen as:

```
The speed is 50000 counts/sec
```

The MG command can also be used to configure terminals. Here, any character can be sent by using {^n} where n is any integer between 1 and 255.

Example:

```
MG {^07} {^255}
```

sends the ASCII characters represented by 7 and 255 to the bus.

Summary of Message Functions:

MG	Message command
" "	Surrounds text string
{Fn.m}	Formats numeric values in decimal n digits to the right of the decimal point and m digits to the left
{$n.m}	Formats numeric values in hexadecimal
{^n}	Sends ASCII character specified by integer n
{N}	Suppresses carriage return/line feed
{Sn}	Sends the first n characters of a string variable, where n is 1 thru 6.

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

Example - Printing a Variable

#DISPLAY	Label
PR 1000	Position Command
BGX	Begin
AMX	After Motion
V1=_TPX	Assign Variable V1
V1=	Print V1

Input of Data

The IN command is used to prompt the user to input numeric or string data. The input data is assigned to the specified variable or array element.

A message prompt may be sent to the user by specifying the message characters in quotes.

Example:

```
#A
IN "Enter Length", LENX
EN
```

The above program sends the message:

```
Enter Length
```

to the PC screen and waits for the operator to enter a value. The operator enters the numeric value which is assigned to the variable, LENX. String variables with up to six characters may also be input using the {S} specifier. For example, IN "Enter X,Y or Z", V{S} specifies a string variable to be input.

Cut-to-Length Example

In this example, a length of material is to be advanced a specified distance. When the motion is complete, a cutting head is activated to cut the material. The length is variable, and the operator is prompted to input it in inches. Motion starts with a start button which is connected to input 1.

The load is coupled with a 2 pitch lead screw. A 2000 count/rev encoder is on the motor, resulting in a resolution of 4000 counts/inch. The program below uses the variable LEN, to length. The IN command is used to prompt the operator to enter the length, and the entered value is assigned to the variable LEN.

```
#BEGIN                        LABEL
AC 800000                     Acceleration
DC 800000                     Deceleration
SP 5000                       Speed
LEN=3.4                       Initial length in inches
#CUT                          Cut routine
AI1                           Wait for start signal
IN "enter Length(IN)", LEN    Prompt operator for length in inches
PR LEN *4000                  Specify position in counts
BGX                           Begin motion to move material
AMX                           Wait for motion done
SB1                           Set output to cut
WT100;CB1                     Wait 100 msec, then turn off cutter
JP #CUT                       Repeat process
EN                            End program
```

Formatting Data

Returned numeric values may be formatted in decimal or hexadecimal* with a specified number of digits to the right and left of the decimal point using the PF command.

The Position Format (PF) command formats motion values such as those returned by the Tell Position (TP), Speed? (SP?) and Tell Error (TE) commands.

Position Format is specified by:

```
PF m.n
```

where m is the number of digits to the left of the decimal point (0 thru 10) and n is the number of digits to the right of the decimal point (0 thru 4) A negative sign for m specifies hexadecimal format.

Hex values are returned preceded by a $ and in 2's complement. Hex values should be input as signed 2's complement, where negative numbers have a negative sign. The default format is PF 10.0.

Examples:

:DP21	Define position
:TPX	Tell position
0000000021	Default format
:PF4	Change format to 4 places
:TPX	Tell position
0021	New format
:PF-4	Change to hexadecimal format
:TPX	Tell Position
$0015	Hexadecimal value

The following interrogation commands are affected by the PF commands:

```
DP?
ER?
PA?
PR?
TE
TP
```

If the number of decimal places specified by PF is less than the actual value, a nine appears in all the decimal places.

Example:

:PF2	Format 2 places
:TPX	Tell position
99	Returns 99 if actual position is more than allowed format

The Variable Format (VF) command is used to format variables and array elements. The VF command is specified by:

VF m.n

where m is the number of digits to the left of the decimal point (0 thru 10) and n is the number of digits to the right of the decimal point (0 thru 4).

A negative sign for m specifies hexadecimal format. The default format for VF is VF 10.4

Hex values are returned preceded by a $ and in 2's complement.

```
:V1=10            Assign V1
:V1=              Return V1
0000000010.0000   Default format
:VF2.2            Change format
:V1=              Return V1
10.00             New format
:VF-2.2           Specify hex format
:V1=              Return V1
$0A.00            Hex value
:VF1              Change format
:V1=              Return V1
9                 Overflow
```

The variable format also affects returned values from internal variables such as _GNX.

PF and VF commands are global format commands. Parameters may also be formatted locally by using the {Fn.m} or {$n.m} specification following the variable = . For example:

```
V1={F4.2}                                  Specifies the variable V1 to be returned in a format of 4
                                           digits to left of decimal and 2 to the right.
```

F specifies decimal and $ specifies hexadecimal. n is the number of digits to the left of the decimal, and m is the number of digits to the right of the decimal. The local format is also used with the MG* command.

Examples:

```
:V1=10            Assign V1
:V1=              Return V1
0000000010.0000   Default Format
:V1={F4.2}        Specify local format
0010.00           New format
:V1={$4.2}        Specify hex format
$000A.00          Hex value
:V1="ALPHA"       Assign string "ALPHA" to V1
:V1={S4}          Specify string format first 4 characters
ALPH
```

User Units

Variables and arithmetic operations make it easy to input data in desired user units such as inches or RPM.

For example, an operator can be prompted to input a number in revolutions. The input number is converted into counts by multiplying it by the number of counts/revolution.

The DMC-1000 position parameters such as PR, PA and VP have units of quadrature counts. Speed parameters such as SP, JG and VS have units of counts/sec. Acceleration parameters such as AC, DC, VA and VD have units of counts/sec^2. All input parameters must be converted into these units.

Example:

```
#RUN                                  Label
IN "ENTER # OF REVOLUTIONS",N1        Prompt for revs
PR N1*2000                            Convert to counts
IN "ENTER SPEED IN RPM",S1            Prompt for RPMs
SP S1*2000/60                         Convert to counts/sec
IN "ENTER ACCEL IN RAD/SEC2",A1       Prompt for ACCEL
AC A1*2000/(2*3.14)                   Convert to counts/sec2
BG                                    Begin motion
EN                                    End program
```

Programmable I/O

Digital Outputs

The DMC-1000 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).

For example:

Instruction	Function
SB6	Sets bit 6 of output port
CB4	Clears bit 4 of output port
CB9	Clear bit 9 of output port on DMC-1080

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.

Instruction	Function
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 may also be written to as 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 output is on.

For example:

Instruction	Function
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. $(2^2 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7)$

The output port is useful for firing relays or controlling external switches and events during a motion sequence.

Example - Turn on output after move

#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-1000 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.

Example:

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-1000. High on input 1 means switch is in on position.

Instruction	Function
#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-1000 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

```
#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 has occurred"     Displays the 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-1000 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

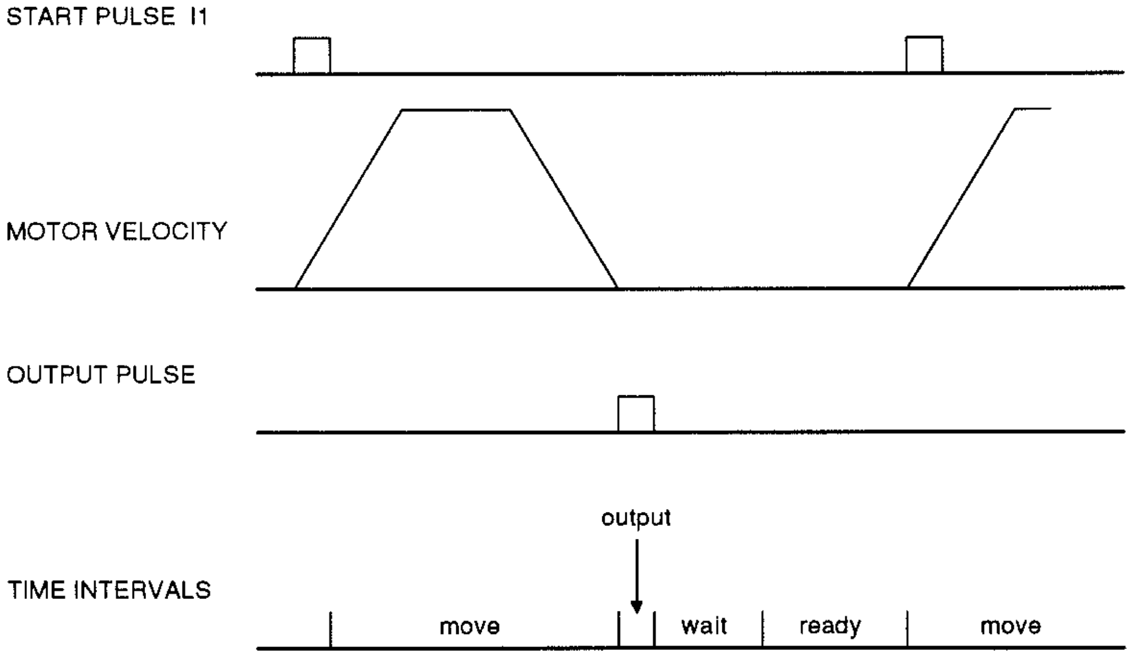


Figure 7.1 - Motor Velocity and the Associated Input/Output signals

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 in/s2 = 1,544,000 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	Function
#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

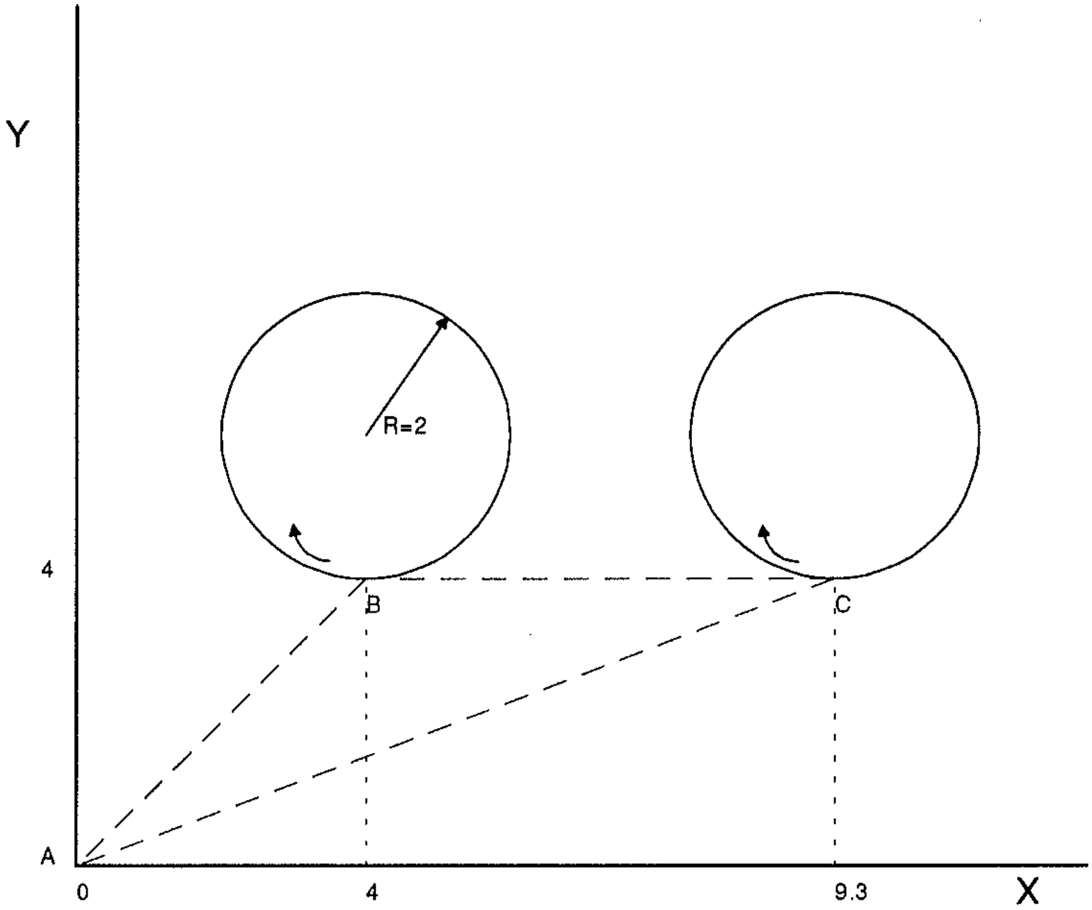


Figure 7.2 - Motor Velocity and the Associated Input/Output signals

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
#A
JG0
BGX
#B
VIN=@AN[1]
VEL=VIN*20000
JG VEL
JP #B
EN

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	Function
#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 Dual-Loop

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 motion program:

Instruction	Function
#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	
JP #B	Repeat the process
#C	
EN	

Error Handling

Introduction

The DMC-1000 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-1000 is an integral part of the machine, the engineer should design his overall system with protection against a possible component failure on the DMC-1000. Galil shall not be liable or responsible for any incidental or consequential damages.

Hardware Protection

The DMC-1000 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.

Input Protection Lines

Abort - A low input stops motion instantly without a controlled deceleration. Also aborts motion program.

Forward Limit Switch - Low input inhibits motion in forward direction. Also causes execution of limit switch subroutine, #LIMSWI, if the motor for that axis is moving in a forward direction. The CN command can be used to change the polarity of the limit switches.

Reverse Limit Switch - Low input inhibits motion in reverse direction. Also causes execution of limit switch subroutine, #LIMSWI, if the motor for that axis is moving in a reverse direction. The CN command can be used to change the polarity of the limit switches.

Software Protection

The DMC-1000 provides a programmable error limit. 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-1000 will generate several signals to warn the host system of the error condition. These signals include:

Signal or Function	State if Error Occurs
# POSERR	Jumps to automatic excess position error subroutine
Error Light	Turns on
OE Function	Shuts motor off if OE1
AEN Output Line	Goes low

The Jump on Condition statement is useful for branching on a given error within a program. The position error of X,Y,Z and W can be monitored during execution using the TE command.

Programmable Position Limits

The DMC-1000 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-1000 will not accept position commands beyond the limit. Motion beyond the limit is also prevented.

Example:

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 software command, Off-on-Error (OE1), turns the motor off when the position error exceeds the limit set by the ER command. The program being executed is also aborted. To re-enable the system, use the Reset (RS) or Servo Here (SH) command. To activate the OE function for each axis, specify 1 for X,Y,Z and W axis. 0 disables off-on-error.

Examples:

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:

```
#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-1000 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.

Limit Switch Example:

```
#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 #RF if reverse
JP#END                Jump to end
#LF                   #LF
MG "FORWARD           Send message
LIMIT"
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.

Troubleshooting

Overview

The following discussion helps with getting the system to work.

For your convenience, the 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

Sympton	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.
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

Sympton	Cause	Remedy
Using COMDISK and TALK2BUS, cannot communicate with controller.	Address selection in communication does not match jumpers.	Check address jumper positions, and change if necessary. The addresses 1000 or 816 are recommended. Note -- for address 1000, A2 and A4 jumpered. For address 816, jumper A7, A6, A3, A2.

STABILITY

Sympton	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

Sympton	Cause	Remedy
Controller rejects command. Responded with a ?	Anything.	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.

Theory of Operation

Overview

The following discussion covers the operation of motion control systems. A typical motion control system consists of the elements shown in Fig 10.1.

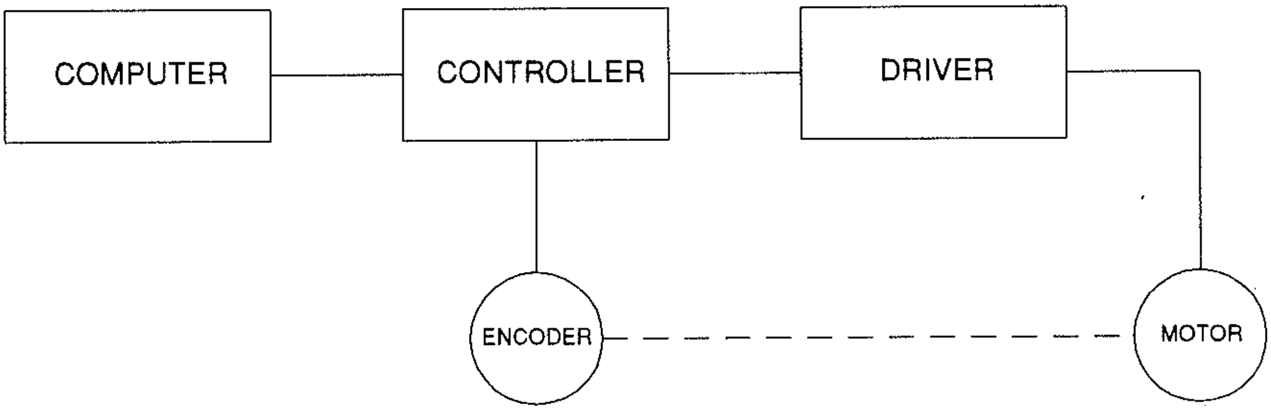


Figure 10.1 - Elements of Servo Systems

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 modelling, 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.

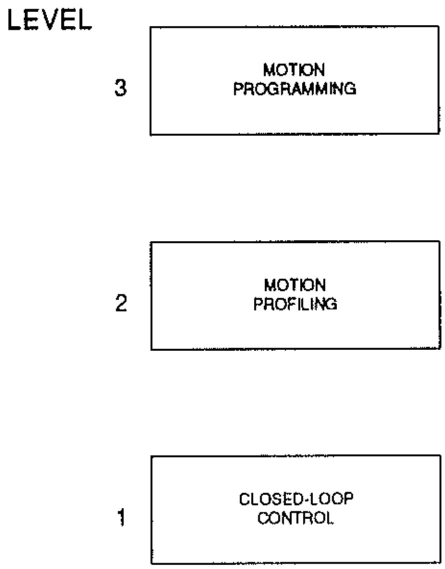


Figure 10.2 - Levels of Control Functions

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 operation of the servo system is done in two manners. First, it is explained qualitatively, in the following section. Later, the explanation is repeated using analytical tools for those who are more theoretically inclined.

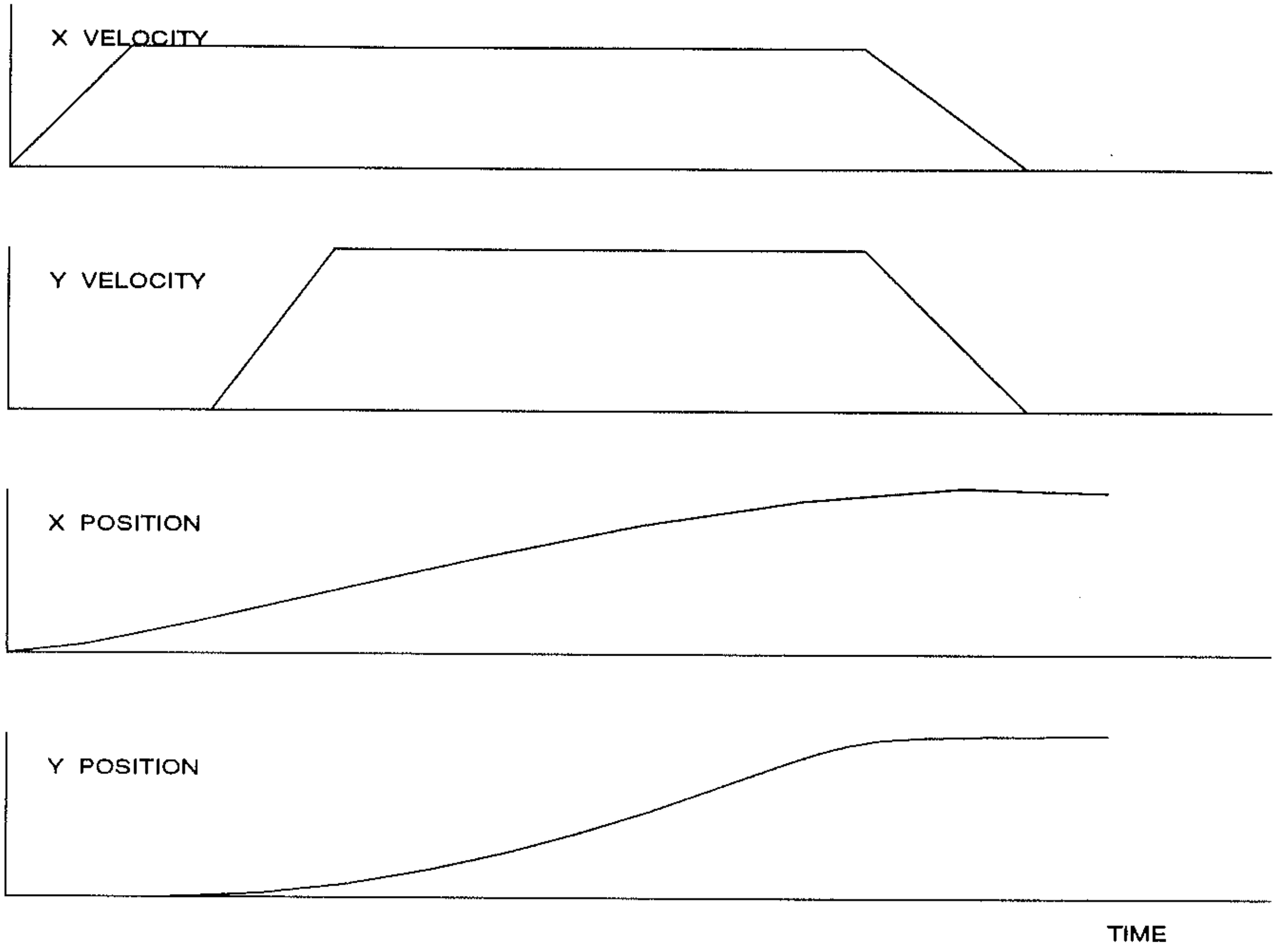


Figure 10.3 - Velocity and Position Profiles

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 modelling, analysis and design.

System Modelling

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.

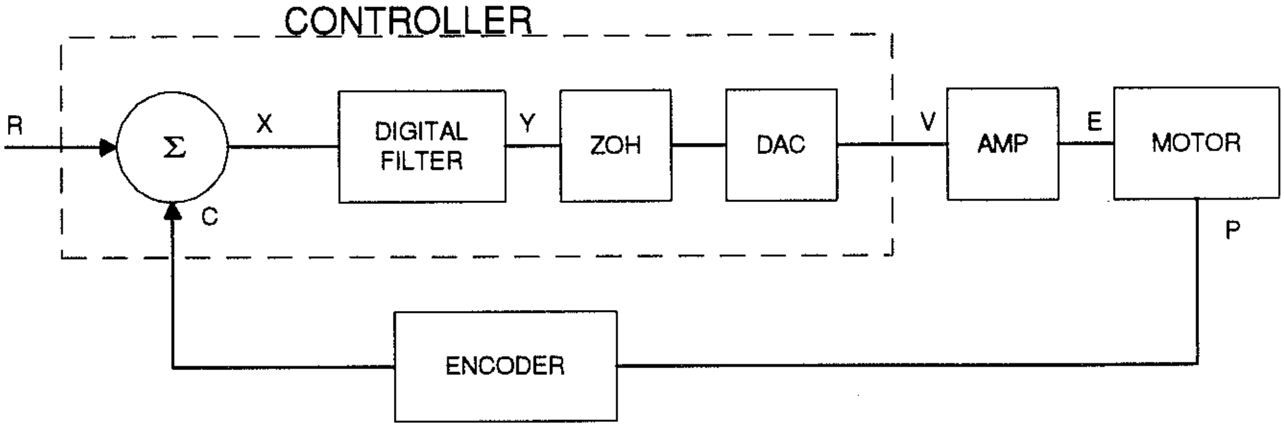


Figure 10.4 - Functional Elements of a Motion Control System

Motor-Amplifier

The motor amplifier may be configured in three modes:

1. Voltage Drive
2. Current Drive
3. Velocity Loop

The operation and modelling in the three modes is as follows:

Voltage Source

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

$$T_e = L/R \qquad [s]$$

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:

$$P/V = 1000/s^2 \qquad [rad/V]$$

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

$$P/V = 1/[K_g s(sT1+1)]$$

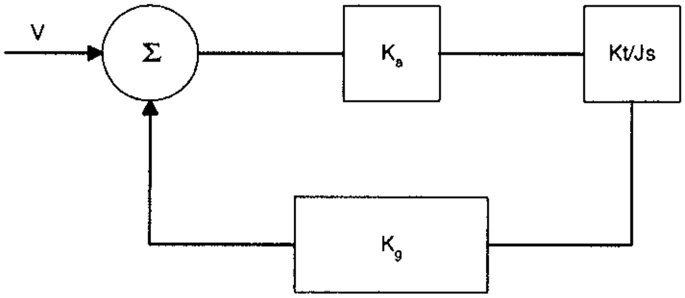


Figure 10.5 - Elements of velocity loops

The resulting functions derived above are illustrated by the block diagram of Fig. 10.6.

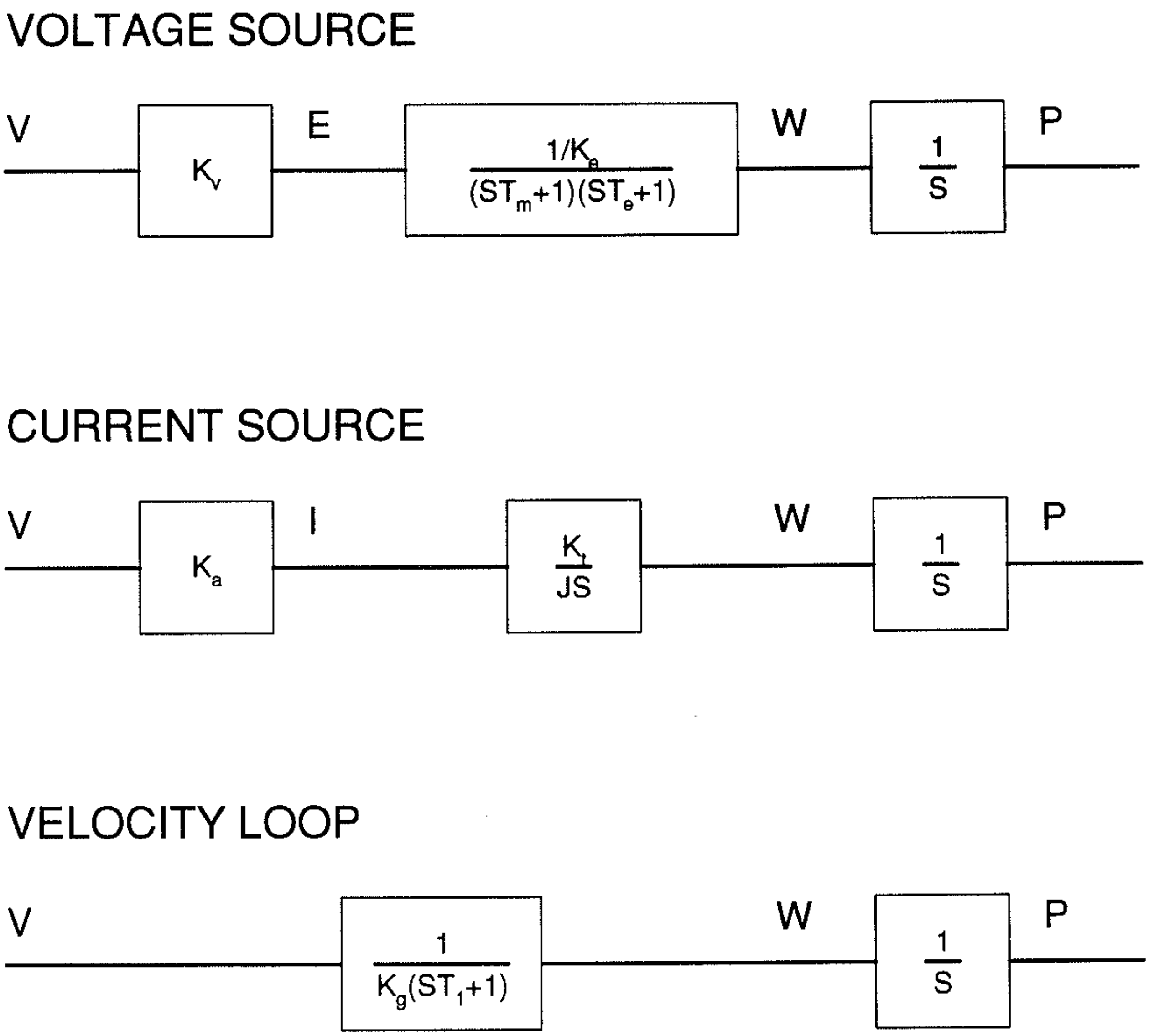


Figure 10.6 - Mathematical model of the motor and amplifier in three operational modes

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 modelled as

$K_f = 638$

DAC

The DAC or D-to-A convertor converts a 14-bit number to an analog voltage. The input range of the numbers is 16384 and the output voltage range is +/-10V or 20V. Therefore, the effective gain of the DAC is

K= 20/16384 = 0.0012 [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	or K = GN
A = KD/(KP + KD)	or A = ZR
C = KI/8	

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) = KP

D = T.K.A = T.KD

I = C/T = KI/8T

For example, if the filter parameters are KP = 4

KD = 36

KI = 2

T = 0.001 s

the digital filter coefficients are

K = 40

A = 0.9

C = 0.25

and the equivalent continuous filter, G(s), is

G(s) = 4 + 0.036s + 250/s

ZOH

The ZOH, or zero-order-hold, represents the effect of the sampling process, where the motor command is updated once per sampling period. The effect of the ZOH can be modelled 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 modelling 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-1000 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

$$M(s) = P/I = Kt/Js2 = 500/s^2 \ [rad/A]$$

Amp

$$K_a = 4 \ [Amp/V]$$

DAC

$$K_d = 0.0012 \ [V/count]$$

Encoder

$$K_f = 4N/2\pi = 318 \ [count/rad]$$

ZOH

$$2000/(s+2000)$$

Digital Filter

$$KP = 12.5, \ KD = 245, \ T = 0.001$$

Therefore,

$$D(z) = 12.5 + 245 \ (1-z-1)$$

Accordingly, the coefficients of the continuous filter are:

$$P = 12.5$$

$$D = 0.245$$

The filter equation may be written in the continuous equivalent form:

$$G(s) = 12.5 + 0.245s = 0.245(s+51)$$

The system elements are shown in Fig. 10.7.

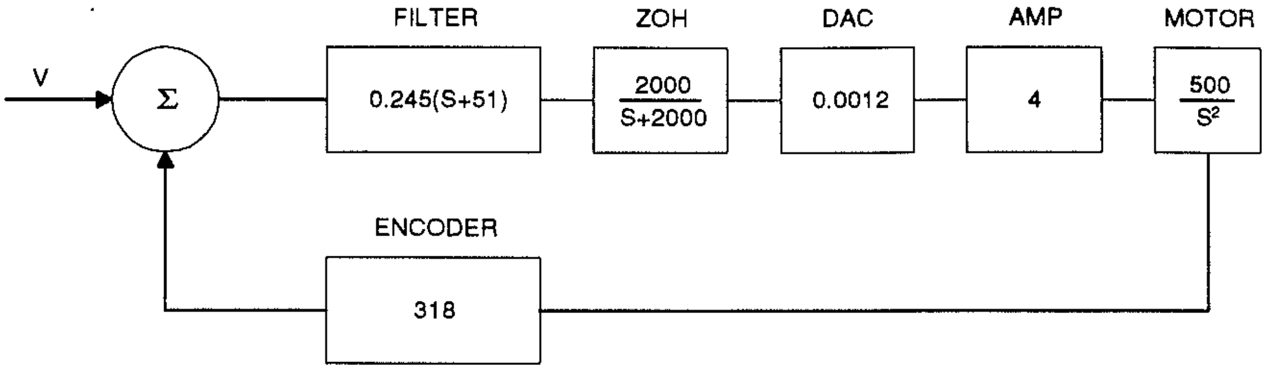


Figure 10.7 - Mathematical model of the control system

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 $A(j\ \omega_c)$ equals one. This can be done by the Bode plot of $A(j\ \omega_c)$, as shown in Fig. 10.8.

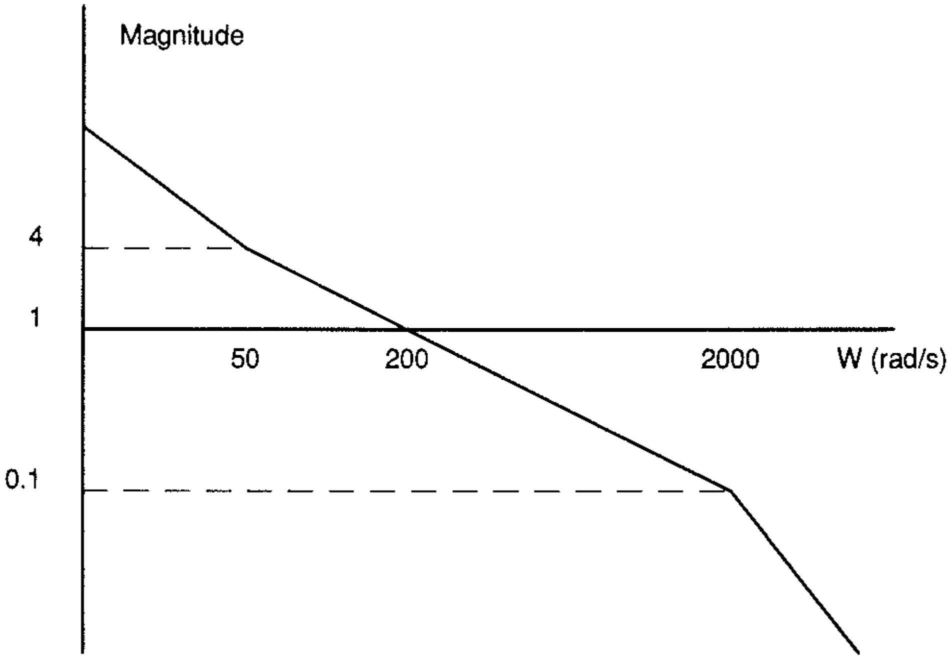


Figure 10.8 - Bode plot of the open loop transfer function

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.

$A(j200) = 390{,}000\ (j200+51)/[(j200)^2\ .\ (j200 + 2000)]$

$\alpha = \text{Arg}[A(j200)] = \tan^{-1}(200/51)-180° -\tan^{-1}(200/2000)$

$\alpha = 76° - 180° - 6° = -110°$

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-1000 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-1000 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

$$K_d = 10/8192$$

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).

$$L(s) = M(s)\, K_a\, K_d\, K_f\, H(s) = 1.27*10^7/[s^2(s+2000)]$$

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 $\omega_c = 500$.

$$L(j500) = 1.27*10^7/[(j500)^2 (j500+2000)]$$

This function has a magnitude of

$$|L(j500)| = 0.025$$

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

$$|G(j500)| = |A(j500)/L(j500)| = 40$$

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 ω_c =500, the function would have a magnitude of 40 and a phase lead of 59 degrees.

These requirements may be expressed as:

$$|G(j500)| = |P + (j500D)| = 40$$

and

$$arg\ [G(j500)] = \tan^{-1}[500D/P] = 59°$$

The solution of these equations leads to:

$$P = 40\cos 59° = 20.6$$

$$500D = 40\sin 59° = 34.3$$

Therefore,

$$D = 0.0686$$

and

$$G = 20.6 + 0.0686s$$

The function G is equivalent to a digital filter of the form:

$$D(z) = KP + KD(1-z^{-1})$$

where

$$KP = P$$

and

$$KD = D/T$$

Assuming a sampling period of T=1ms, the parameters of the digital filter are:

$KP = 20.6$

$KD = 68.6$

The DMC-1000 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

Digital	$D(z) = K(z-A/z) + Cz/z-1$
Digital	$D(z) = KP + KD(1-z^{-1}) + KI/8(1-z^{-1})$
KP, KD, KI	$K = KP + KD$
	$A = KD/(KP+KD)$
	$C = KI/8$
Digital	$D(z) = GN(z-ZR)/z + KI\ z/8(z-1)$
GN, ZR, KI	$K = GN$
	$A = ZR$
	$C = KI/8$
Continuous	$G(s) = P + Ds + I/s$
PID, T	$P = K(1-A) = KP$
	$D = K*A*T = T*KD$
	$I = C/T = KI/8T$

Command Reference

Command Descriptions

Each executable instruction is listed in the following section in alphabetical order.

The two-letter Opcode for each instruction is placed in the upper right corner. Below the opcode is a description of the command and required arguments. As arguments, some commands require actual values to be specified following the instruction. These commands are followed by lower case x,y,z,w. Values may be specified for any axis separately or any combination of axes. Axis values are separated by commas. Examples of valid x,y,z,w syntax are listed below. For the DMC-1080, the axis designaters a,b,c,d,e,f,g,h are used where x,y,z,w can be used interchangeably iwth a,b,c,d.

Valid x,y,z,w syntax	
AC x	Specify x 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 ,y	Specify y only
AC ,y,z	Specify y and z
AC ,,z	Specify z only
AC ,,,w	Specify w only
AC x,,,w	Specify x and w only
AC a,,,d,,f	Specify a,d and f only (DMC-1080)

Where x,y,z and w are replaced by actual values.

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

A ? returns the specified value for that axis. For example, AC ?,?,?,?, returns the acceleration of the X,Y,Z and W axes. The syntax format is the same as x,y,z,w except ? replaces the values.

Other commands require action on the X,Y,Z or W axis to be specified. These commands are followed by uppercase X,Y,Z or W. Action for a particular axis or any combination is specified by writing X,Y,Z or W. No commas are needed. Valid XYZW syntax is listed below. The DMC-1080 uses ABCDEFGH axis designators where XYZW can be used interchangeably with ABCD.

Valid XYXW 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 Y	Servo Here, Y only
SH YZW	Servo Here, Y,Z and W axes
SH Z	Servo Here, Z only
SH	Servo Here, X,Y,Z and W axes
SH W	Servo Here, W only
SH ZW	Servo Here, Z and W axes
SH ABFG	Servo Here, A,B,F,G axes

Where X,Y,Z and W specify axis.

The Usage description specifies the restrictions on allowable execution. "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. "Not in a program" 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 with sending ? to return the specified value. "Used in Operand" states whether a command can be used to generate a value for another command or variable (i.e. V=_GNX). "Default Format" defines the format of the value with number of digits before and after the decimal point. Finally, "Default Value" defines the values the instruction's parameters will have after a Master Reset.

When downloading commands to the DMC-1000, do not insert a space prior to any command. For example, STX; AMX is invalid because there is a space after the semicolon.

AB

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. If the command Off-on-Error (OE1) has been activated, AB will shut off the motor commands.

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	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"SH" on page 246	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

HINTS: 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

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 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 value

USAGE:

While Moving	Yes	Default Value	25600
In a Program	Yes	Default Format	8.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"DC" on page 169	Specifies deceleration rate.
"FA" on page 190	Feedforward Acceleration.
"IT" on page 205	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

HINTS: 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

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 the position command has reached the specified relative distance from the start of the move. The units of the command are quadrature counts. Only one axis may be specified at a time.

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 Only one axis may be specified at a time

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"AR" on page 151	After distance for repetitive triggering
"AV" on page 154	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
```

HINTS:

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

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

where n is an integer in the range 1 to 24 decimal for DMC-1080

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

@IN[n]	Function to read input 1 through 8
"II" on page 200	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

FUNCTION: Arm Latch

DESCRIPTION:

The AL command enables the latching function (high speed position capture) of the controller. When the ALXYZW command is used to arm the position latches, the encoder position will be captured upon a low going signal on Input 1 (X axis), Input 2 (Y axis), Input 3 (Z axis), Input 4 (W axis), Input 9 (E axis), Input 10 (F axis), Input 11 (G axis) and Input 12 (H axis). The command RL returns the captured position for the specified axes. When interrogated or used in an operand 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   AL ABCDEFGH

where X,Y,Z,W specifies the X,Y,Z,W axes

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"RL" on page 240	Report Latch

EXAMPLES:

```
000 #START              Start program
001 ALY                 Arm Y-axis latch
002 JG,50000            Set up jog at 50000 counts/sec
003 BGY                 Begin the move
004 #LOOP               Loop until latch has occurred
005 JP #LOOP,_ALY=1
006 RLY                 Transmit the latched position
007 EN                  End of program
```

AM

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 XYZWS

where X,Y,Z,S,W specifies X,Y,Z or W axis or sequence. No argument specifies that motion on all axes is complete.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"BG" on page 155 (_BG returns 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
#F;DP 0,0,0,0               Program F
PR 5000,6000,7000,8000      Position relative moves
BG                          Start X,Y,Z and W axes
AM                          After motion complete on all axes
MG "DONE";TP                Print message
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

FUNCTION: After Absolute Position

DESCRIPTION:

The AP command is a trippoint used to control the timing of events. This command will hold up the execution of the following command until the absolute actual position of the motor crosses the position specified. The units of the command are in quadrature counts. Only one axis may be specified at a time. This trippoint will also be cleared by the completion of the move.

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:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"AD" on page 145	for relative distances

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

FUNCTION: After Relative Distance

DESCRIPTION:

The After Relative Distance command is a trippoint used to control the timing of events. This command will hold up execution of the following command until the specified relative distance has reached from either last AR or AD command, or from the start of the move. Only one axis may be specified at a time. The units of the command are quadrature counts.

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 Only one axis may be specified at a time

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"AV" on page 154	after vector position for coordinated moves
"AP" on page 150	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

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

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

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

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	Default Value	0
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

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

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.

When used as an operand, _AV returns the vector distance from the start of the sequence. _AV is valid in the linear mode, LM, and in the vector mode, VM

ARGUMENTS: AV n

where n is an unsigned integer in the range 0 to 2147483647 decimal

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

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,
MG "Path>500";TPXY    Print Message
EN                    End Program
```

HINT: *Remember AV is the vector distance along the path where* $AV^2=X^2+Y^2+Z^2+W^2$

BG

FUNCTION: Begin

DESCRIPTION:

The BG command starts a motion on the specified axis or sequence. When used in an operand, the BG command will return a 1 if the controller is performing a move of that axis.

ARGUMENTS: BG XYZWS   BG ABCDEFGH

where XYZW are X,Y,Z,W axes and S is coordinated sequence

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"AM" on page 148	After motion complete
"ST" on page 248	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 sequence
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

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.

USAGE:

While Moving	Yes	Default Value	-214783648
In a Program	Yes	Default Format	Position format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"FL" on page 193	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: The DMC-1000 also provides hardware limits.

BN

FUNCTION: Burn

DESCRIPTION:

The BN command saves the certain controller parameters shown below in non-volatile EEPROM memory. Programs are not saved and must be downloaded from the host computer upon power-up. This command typically takes 1 second to execute and must not be interrupted. The controller returns a : when the Burn is complete.

Controller Parameters Saved During Burn:

AC
SP
KP (GN converted to KP)
KI
ER
VA
VS
IT
MT
TM
OP
BL
VF
DV
CN
MOTOR State (IN SERVO or motor off)

DC
GR
KD (ZR converted to KD)
TL
IL
VD
VT
CO
EO
OE
GA
FL
PF
CE

ARGUMENTS: None

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

```
#MAIN          Label
II1            Enable input interrupt on input 1
PR 5000        Move relative position of 5000 counts
SP 10000       Set speed
AC 200000      Set acceleration
BGX            Begin motion X-axis
AMX            After motion complete
WT 1000        Wait 1000 msec
JP #MAIN       Jump to MAIN label
#ININT         Input interrupt routine
AM             Wait for motion complete
WT 1000        Wait 1000 msec
BN             Burn program; may take up to 15 seconds
RI 1           Return to main program and restore trippoint
```

CB

FUNCTION: Clear Bit

DESCRIPTION:

The CB command clears one of eight bits on the output port. The CB and SB (Set Bit) instructions can be used to control the state of output lines.

ARGUMENTS: CB n,

where n is an integer in the range 1 to 8 decimal

where n is an integer in the range 1 to 16 decimal for DMC-1080

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"SB" on page 244	Set Bit
"OP" on page 229	Define all outputs

EXAMPLES:

CB 0	Clear output bit 0
CB 7	Clear output bit 7
CB 16	Clear output bit 16 (DMC-1080 only)

CD

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	Default Value	0
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"CM" on page 162	Contour Mode
"WC" on page 278	Wait for Contour
"DT" on page 174	Time Increment
"CS" on page 167	_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

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.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	2.0
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

CE 0, 3, 6, 2	Configure encoders
CE ?,?,?,?	Interrogate configuration
V = _CEX	Assign configuration to a variable

CM

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 CM? or _CM commands can be used to check the status of the Contour Buffer.

When CM? is sent and the returned value is 1, it indicates the Contour Buffer is full. If the returned value is 0, it indicates the Contour Buffer is empty, and a new CD command issued.

ARGUMENTS: CM XYZW   CM ABCDEFGH

where XYZW specify the X,Y,Z,W axes

USAGE:

While Moving	No	Default Value	---
In a Program	No	Default Format	1.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	No		

RELATED COMMANDS:

"CD" on page 160	Contour Data
"WC" on page 278	Wait for Contour
"DT" on page 174	Time Increment

EXAMPLES:

V=_CM;V=	Return Contour Buffer Status
1	Contour Buffer is full
CM XZ	Specify X,Z axes for Contour Mode

CN

FUNCTION: Configure

DESCRIPTION:

The CN command configures the polarity of the limit switches, the home switch and the latch input. It also sets the width of the pulses in the step motor command.

ARGUMENTS: CN m,n,o,p

where m,n,o are integers with values 1 or -1. P is 1,2 or 3.

m =	1	Limit switches active high
	-1	Limit switches active low
n =	1	Home switch active high
	-1	Home switch active low
o =	1	Latch input active high
	-1	Latch input active low
p =	1	Pulse duration 960 ns
	2	Pulse duration 1920 ns
	3	Pulse duration 15.36 µs

USAGE:

While Moving	Yes	Default Value	-1,-1,-1,-1
In a Program	Yes	Default Format	2.0
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"MT" on page 224	Motor Type

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.*

CO

FUNCTION: Configure Outputs

DESCRIPTION:

The CO configures the first 48 I/O points (9-56) of the DB-10072 expansion board in blocks of 8.

ARGUMENTS: CON

Where N is a six bit number where each bit is assigned to a block of inputs. (blocks 0 thru 7). $N = n_0 + 2*n_1 + 4*n_2 + 8*n_3 +16* n_4 +32* n_5$ where each bit is a one if the block of 8 I/O points is to be set as an output. For example, if blocks 0,1,2,and 4 are to be configured as outputs, CO 11 is issued.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

CO 0	Configure all points as inputs
CO 2	Configures I/O points 17-24 as outputs

Hint: See appedix for more information on the DB-10072 and the CO command.

CR

FUNCTION: Circle

DESCRIPTION:

The CR command specifies a 2-dimensional arc segment of radius, r, starting at angle, θ, and traversing over angle $\Delta\theta$. A positive $\Delta\theta$ denotes counterclockwise traverse, negative $\Delta\theta$ 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, $\Delta\theta$, must be specified. Radius units are in quadrature counts. θ and $\Delta\theta$ 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,$\Delta\theta$ < n

where r is an unsigned real number in the range 10 to 6000000 decimal

θ is an unsigned number in the range 0 to +/-32000 decimal

$\Delta\theta$ is a signed real number in the range 0.0001 to +/-32000 decimal

n is an unsigned even number between 0 and 8,000,000

The product r * $\Delta\theta$ must be limited to +/-4.5 10^8

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"VP" on page 273	Vector Position
"VS" on page 276	Vector Speed
"VD" on page 268	Vector Deceleration
"VA" on page 267	Vector Acceleration
"VM" on page 271	Vector Mode
"VE" on page 269	End Vector
"BG" on page 155	BGS - Begin Sequence

EXAMPLES:

VM XY	Specify motion plane
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

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.

When used as an operand, _CS returns the number of the segment in the sequence,starting at zero. The instruciton _CS is valid in the Linear mode, LM, Vector mode, VM, and contour mode, CM.

NOTE: This command is not valid for the DMC-1010.

ARGUMENTS: None

USAGE:

While Moving	No	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

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

DA

FUNCTION: Deallocate the Variables & Arrays

DESCRIPTION:

The DA command frees the memory of the space occupied by an array or variable. 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 only limit is the 40 character per line. The * argument deallocates all the variables, and *[0] deallocates all the arrays.

ARGUMENTS: DA c[0],d,etc.

where c[0] - Defined array name

d - Defined variable name

* - Deallocates all the variables

*[0] - Deallocates all the arrays

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"DM" on page 172	Dimension Array

EXAMPLES: Cars and Salesmen are arrays and Total is a variable.

DM Cars[400],Salesmen[50]	Dimension 2 arrays
Total=70	Assign 70 to the variable Total
DA Cars[0],Salesmen[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

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 value

USAGE:

While Moving	Yes	Default Value	256000
In a Program	Yes	Default Format	8.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"AC" on page 144	Acceleration
"PR" on page 232	Position Relative
"PA" on page 230	Position Absolute
"SP" on page 247	Speed
"JG" on page 206	Jog
"BG" on page 155	Begin
"IT" on page 205IT	Smoothing constant - S-curve

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

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. DE ? or _DEX returns the position of the auxiliary encoders.

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

USAGE:

While Moving	Yes	Default Value	0,0,0,0
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

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.

DL

FUNCTION: Download

DESCRIPTION:

The DL command transfers a data file from the host computer to the DMC-1000. Instructions in the file will be accepted as a datastream without line numbers. The file is terminated using <control> Z, <control> Q, <control> D, or \

If no parameter is specified, downloading a data file will clear any programs in the DMC-1000 RAM. The data is entered beginning at line 0. If there are too many lines or too many characters per line, the DMC-1000 will return a ?. To begin the download following a label, that label may be specified following DL. Or, the # argument may be used with DL to append a file at the end of the DMC-1000 program in RAM. DO NOT insert any spaces before each command.

When used as an operand, _DL gives the number of available labels. The total number of labels is 126 for the DMC-1000 and 500 for DMC-1000-MX.

ARGUMENTS: DL n

n = no argument	Downloads program beginning at line 0. Erases programs in RAM.
n = #Label valid program label.	Begins download at line following #Label where label may be any
n = #	Begins download at end of program in RAM.

USAGE:

While Moving	Yes	Default Value	---
In a Program	No	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

"UL" on page 266	Upload

EXAMPLES:

DL;	Begin download
#A;PR 4000;BGX	Data
AMX;MG DONE	Data
EN	Data
<control> Z	End download

DM

FUNCTION: Dimension

DESCRIPTION:

The DM command defines a single dimensional array with a name and n total elements. The maximum number of arrays which the user can define is limited to fourteen. With the fourteen arrays, the maximum number of elements is limited to 1600 (8000 for DMC-1080 and DMC-1040-MX). If the user needs only one array, then the limitation to maximum number of elements is still 1600. The first element of the defined array starts with element number 0 and the last element is at n-1.

t 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 is the number of entries from 1 to 1600 for the DMC-1040 and 1 to 8000 for the DMC-1040-MX and DMC-1080.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"DA" on page 168	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

FUNCTION: Define Position

DESCRIPTION:

The DP command sets the current motor position and current command positions to a user specified value. If a ? is used, it returns the current position of the motor. The units are in quadrature counts.

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

USAGE:

While Moving	No	Default Value	0,0,0,0
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

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 ?,?,?,?	
0000000,-0050000,0000200,0000400	Returns all the motor positions
:DP ?	
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

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.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"CM" on page 162	Contour Mode
"CD" on page 160	Contour Data
"WC" on page 278	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

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 funtion. 1 enables the dual loop.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.0
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

"KD" on page 209	Damping constant
"FV" on page 194	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.

EA

FUNCTION: Choose ECAM master

DESCRIPTION:

The EA command selects the master axis for the electronic cam mode. Any axis may be chosen.

ARGUMENTS: EA p

where p is XYZW or EFGH

USAGE:

While moving	Yes
In a program	Yes
Not in a program	Yes
Can be interrogated	No
Used as an operand	No

RELATED COMMANDS:

EM on page 182

EP on page 185

ET on page 189

EB on page 177

EXAMPLES:

EAY	Select Y as a master for ECAM

EB

FUNCTION: Enable ECAM

DESCRIPTION:

The EB function enables or disables the cam mode. In this mode, the starting position of the master axis is specified within the cycle. When the EB command is given, the master axis is modularized.

ARGUMENTS: EB n

where n = 1 starts cam mode and n = 0 stops cam mode.

USAGE:

While moving	Yes
In a program	Yes
Not in a program	Yes
Can be interrogated	No
Used as an operand	Yes

RELATED COMMANDS:

EM
EP

EXAMPLES:

EB1	Starts ECAM mode
EB0	Stops ECAM mode
B = _EB	Return status of cam mode

ED

FUNCTION: Edit

DESCRIPTION:

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

The ED command can be used in an operand, where it will return the line number of the last line to have an error.

EXAMPLES:

```
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 or listing a program.

EG

FUNCTION: ECAM go (engage)

DESCRIPTION:

The EG command engages an ECAM slave axis at a specified position of the master. _EGX returns 1 if the X axis is engaged. If a value is specified outside of the master's range, the slave will engage immediately. Once a slave motor is engaged, its position is redefined to fit within the cycle.

ARGUMENTS: EG x,y,z,w

where x,y,z,w are the master positions at which the X,Y,Z,W axis must be engaged.

USAGE:

While moving	Yes
In a program	Yes
Not in a program	Yes
Can be interrogated	No
Used as an operand	Yes

RELATED COMMANDS:

EB
EQ

EXAMPLES:

EG 700,1300	Engages the X and Y axes at the master position 700 and 1300 respectively.
B = _EPY	Return the status of Y axis, 1 if engaged.

NOTE: This command is not a trippoint. This command will not hold the execution of the program flow. If the execution needs to be held until master position is reached, use MF or MR command.

EI

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 the EI command, one IRQ line must be jumpered on the DMC-1000 and the data 2 and 4 written to the control register at address N + 1. An interrupt service routine must also be incorporated in your host program. Refer to Section 4.6 for details.

ARGUMENTS: EI m,n where m is interrupt condition mask

n is input mask

m = Bit number (2^m)	Condition
0	X motion complete
1	Y motion complete
2	Z motion complete
3	W motion complete
4	E motion complete (1080 only)
5	F motion complete (1080 only)
6	G motion complete (1080 only)
7	H motion complete (1080 only)
8	All axes motion complete
9	Excess position error*
10	Limit switch*
11	Watchdog timer
12	Reserved
13	Application program stopped
14	Command done
15	Inputs* (uses n for mask)

The * conditions must be re-enabled after each occurrence.

n = Bit number (2^n)	
0	Input 1
1	Input 2
2	Input 3
3	Input 4
4	Input 5
5	Input 6
6	Input 7
7	Input 8

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"UI" on page 265	User interrupt

EXAMPLES:

1. Specify interrupts for all axes motion complete and limit switch.

   Enable bits 8 and 10. $m = 2^8 + 2^{10} = 256 + 1024 = 1280$

   EI 1280

2. Specify interrupt on Input 3.

   Enable bit 15 on m and bit 2 on n.

   $m = 2^{15} = 32768$

   $n = 2^2 = 4$

   EI 32768,4

EM

FUNCTION: Cam cycles

DESCRIPTION:
The EM command is part of the ECAM mode. It is used to define the change in position over one complete cycle of the master. The field for the master axis is the cycle of the master position. For the slaves, the field defines the net change in one cycle. If a slave will return to its original position at the end of the cycle, the change is zero. If the change is negative, specify the absolute value.

ARGUMENTS: EM x,y,z,w
where the parameters are positive integers in the range between 1 and 8,388,607 for the master axis and between 1 and 2,147,483,647 for a slave axis.

USAGE:

While moving	Yes
In a program	Yes
Not in a program	Yes
Can be interrogated	No
Used as an operand	Yes

RELATED COMMANDS:

EA

EP

ET

EB

EXAMPLES:

EAZ	Select Z axis as master for ECAM.
EM 0,3000,2000	Define the changes in X and Y to be 0 and 3000 respectively. Define master cycle as 2000.
V = _EMX	Return cycle of X

EN

FUNCTION: End

DESCRIPTION:

The EN command is used to designate the end of a program or subroutine. If a subroutine has been called by the JS instruction, the EN command will return execution to the instruction after the JS command. If there is no subroutine being executed, the program terminates.

ARGUMENTS: EN m

where m=0 does not restore trippoint; m=1 restores trippoint.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	No		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

#A	Program A
PR 500	Relative position move
BGX	Begin the move
AMX	After the move is complete
PR 1000	Set another Position Relative move
BGX	Begin the move
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.

EO

FUNCTION: Echo

DESCRIPTION:

The EO command turns the echo on or off. If the echo is off, characters input over the bus will not be echoed back.

ARGUMENTS: EO n

where n=0 or 1. 0 turns echo off, 1 turns echo on.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.0
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

EO 0	Turns echo off
EO 1	Turns echo on

EP

FUNCTION: Cam table intervals and starting point

DESCRIPTION:

The EP command defines the ECAM table intervals and offset. The offset is the master position of the first ECAM table entry. The interval is the difference of the master position between 2 consequetive table entries. This command effectively defines the size of the ECAM table. The parameter m is the interval and n is the starting point. Up to 257 points may be specified.

ARGUMENTS: EP m,n

where m is a positive integer in the range between 1 and 32, 767 and n is an integer between -2,147,483,648 and 2,147,483,647.

USAGE:

While moving	Yes
In a program	Yes
Not in a program	Yes
Can be interrogated	No
Used as an operand	Yes (m only)

RELATED COMMANDS:

EM
ET

EXAMPLE:

EP 20,100	Sets the cam master points to 100,120,140 . . .
D = _EP	Returns interval (m)

EQ

FUNCTION: ECAM quit (disengage)

DESCRIPTION:
The EQ command disengages an electronic cam slave axis at the specified master position. Separate points can be specified for each axis. If a value is specified outside of the master's range, the slave will disengage immediately. _EQX returns 1 if axis is waiting to start, 2 if axis is waiting to stop, 3 if both waiting to start and stop and 0 if ECAM engaged or already stopped.

ARGUMENTS: EQ x,y,z,w
where x,y,z,w are the master positions at which the XYZW axes are to be disengaged.

USAGE:

While moving	Yes
In a program	Yes
Not in a program	Yes
Can be interrogated	No
Used as an operand	Yes

RELATED COMMANDS:
EG

EXAMPLES:

EQ 300,700	Disengages the X and Y motors at master positions 300 and 700 respectively.

NOTE: This command is not a trippoint. This command will not hold the execution of the program flow. If the execution needs to be held until master position is reached, use MF or MR command.

ER

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: ERX = x ER a,b,c,d,e,f,g,h

x,y,z,w are unsigned numbers in the range 1 to 32767

USAGE:

While Moving	Yes	Default Value	16384
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"OE" on page 227	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

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	Default Value	1,1
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"VM" on page 271	Vector Mode
"CR" on page 165	Circle move
"VP" on page 273	Vector position

EXAMPLES:

VMXY;ES3,4	Divide Y resolution by 4/3
VMZX;ES2,3	Divide X resolution by 3/2

ET

FUNCTION: Electronic cam table

DESCRIPTION:

The ET command sets the ECAM table entries for the slave axes.. The values of the master axes are not required. The slave entry (n) is the position of the slave axes when the master is at the point n * i + o, where i and o are the interval and offset as determined by the EP command.

ARGUMENTS: ET [n] = x,y,z,w

where n is an integer between 0 and 256 and the parameters x,y,z,w are integers in the range between -2,147,438,648, and 2,147,438,647.

USAGE:

While moving	Yes
In a program	Yes
Not in a program	Yes
Can be interrogated	No
Used as an operand	No

RELATED COMMANDS:

EA
EM
EB
EP

EXAMPLES:

ET [7] = 1000,300,500	Specifies the position of the slave axes X, Y and Z that must be synchronized with the eighth increment of the master.

FA

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 · AC · 1.5 · 10^{-7}

Deceleration Feedforward Bias = FA · DC · 1.5 · 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

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	4.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"FV" on page 194	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

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.

USAGE:

While Moving	No	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"FI" on page 192	Find Index
"HM" on page 198	Home
"BG" on page 155	Begin
"AC" on page 144	Acceleration Rate
"DC" on page 169	Deceleration Rate
"SP" on page 247	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

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:

While Moving	No	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"FE" on page 191	Find Edge
"HM" on page 198	Home
"BG" on page 155	Begin
"AC" on page 144	Acceleration Rate
"DC" on page 169	Deceleration Rate
"SP" on page 247	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

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

USAGE:

While Moving	Yes	Default Value	2147483647
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"BL" on page 156	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:** The DMC-1000 also provides hardware limits.*

FV

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 = $1.22 \cdot 10^{-6} \cdot \mathrm{FV} \cdot$ 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

where x,y,z,w are unsigned numbers in the range 0 to 8191 decimal

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	3.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"FA" on page 190	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

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.

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	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"GR" on page 197	Gear Ratio

EXAMPLES:

#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

HINT: Using the command position as the master axis is useful for gantry applications. Using the vector motion as master is useful in generating Helical motion.

GN

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.

USAGE:

While Moving	Yes	Default Value	70
In a Program	Yes	Default Format	4
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"ZR" on page 281	Zero
"KI" on page 210	Integrator
"KP" on page 211	Proportional
"KD" on page 209	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

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

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	3.4
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"GA" on page 195	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

FUNCTION: Home

DESCRIPTION:

The HM command performs a three-stage homing sequence. The first stage consists of the motor moving at the user programmed speed until it sees 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 state consists of the motor changing directions and slowly approaching the transition again. When it detects the transition, it stops.

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.

When the HM command is used in an operand (_HMX), it returns the state of the home switch.

When a step motor is used, the homing function consists of the first two stages.

ARGUMENTS: None

USAGE:

While Moving	No	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

"CN" on page 163	Configure Home
"FI" on page 192	Find Index Only
"FE" on page 191	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

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.

When used as an operand, _HXn returns the running status of thread n with:

0	Thread not running
1	Thread is running
2	Thread has stopped at trippoint

ARGUMENTS: HXn

where n is an integer in the range of 0 to 3 to indicate the strand number.

USAGE:

While Moving	Yes	Default Value	n = 0
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

"XQ" on page 280	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

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. The RI command is used to return from the #ININT routine. The RI command also re-enables input interrupts. To avoid returning to the main program on an interrupt, use the command ZS to zero the subroutine stack.

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	Default Value	0
In a Program	Yes	Default Format	3.0 (mask only)
Not in a Program	No		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"RI" on page 239	Return from Interrupt
#ININT	Interrupt Subroutine
"AI" on page 146	Trippoint for input

EXAMPLES:

```
#A                     Program A
II 1                   Specify interrupt on input 1
JG 5000                Specify jog speed
BGX                    Begin motion
#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
RI                              Return to main program
```

IL

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 0 to 9.9988 Volts.

USAGE:

While Moving	Yes	Default Value	9.9988
In a Program	Yes	Default Format	1.4
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"KI" on page 210	Integrator

EXAMPLES:

KI 2,3,5,8	Integrator constants
IL 3,2,7,2	Integrator limits
IL ?	Returns the X-axis limit
3.0000	

IN

FUNCTION: Input Variable

DESCRIPTION:

The IN command allows a variable to be input from a keyboard. When the IN command is executed in a program, the prompt message is displayed. The operator then enters the variable value followed by a carriage return. The entered value is assigned to the specified variable name.

The IN command holds up execution of following commands in a program until a carriage return or semicolon is detected. If no value is given prior to a semicolon or carriage return, the previous variable value is kept. Input Interrupts, Error Interrupts and Limit Switch Interrupts will still be active.

ARGUMENTS: IN "m",n

where:

m is prompt message

n is the variable name

The limit on the number of characters for n and m are such that the total number of characters per line are 40 characters or less.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	Position Format
Not in a Program	No		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

Operator specifies length of material to be cut in inches and speed in inches/sec (2 pitch lead screw, 2000 counts/rev encoder).

```
#A                              Program A
IN "Enter Speed(in/sec)"V1      Prompt operator for speed
IN "Enter Length(in)",V2        Prompt for length
V3=V1*4000                      Convert units to counts/sec
V4=V2*4000                      Convert units to counts
SP V3                           Speed command
PR V4                           Position command
BGX                             Begin motion
AMX                             Wait for motion complete
MG "MOVE DONE"                  Print Message
EN                              End Program
```

IP

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 indeterminant due to a variable diameter pulley.

ARGUMENTS: IP x,y,z,w IPX = x IP a,b,c,d,e,f,g,h

x,y,z,w are signed numbers in the range -2147483648 to 2147483647 decimal.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	7.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	No		

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

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

USAGE:

While Moving	Yes	Default Value	1.0
In a Program	Yes	Default Format	1.4
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"VT" on page 277	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

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 +/-8,000,000 decimal

USAGE:

While Moving	Yes	Default Value	16385
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	No		

RELATED COMMANDS:

"BG" on page 155	Begin
"ST" on page 248	Stop
"AC" on page 144	Acceleration
"DC" on page 169	Deceleration
"IP" on page 204	Increment Position
"TV" on page 263	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

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:

While Moving	Yes
In a Program	Yes
Not in a Program	No
Can be Interrogated	No
Used in 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

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. These can be nested 8 deep in the standard controller and 16 deep in the DMC-1000-MX.

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

> greater than

= equal to

<= less than or equal to

>= greater than or 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:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	No		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"EN" on page 183	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

FUNCTION: Derivative Constant

DESCRIPTION:

KD designates the derivative constant in the controller filter. The filter transfer function is

D(z) = KP + KD(z-1)/z + KI z/8(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.

USAGE:

While Moving	Yes	Default Value	64
In a Program	Yes	Default Format	4.2
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"KP" on page 211	Proportional Constant
"KI" on page 210	Integral

EXAMPLES:

KD 100,200,300,400.25	Specify KD
KD ?,?,?,?	Return KD
0100.00,0200.00,0300.00,0400.25	

KI

FUNCTION: Integrator

DESCRIPTION:

The KI command sets the integral gain of the control loop. It fits in the control equation as follows:

D(z) = KP + KD(z-1)/z + KI z/8(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

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	4.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"GN" on page 196	Gain
"KP" on page 211	Proportional Gain
"KD" on page 209	Derivative
"ZR" on page 281	Zero
"IL" on page 202	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

FUNCTION: Proportional Constant

DESCRIPTION:

KP designates the proportional constant in the controller filter. The filter transfer function is

D(z) = KP + KD(z-1)/z + KI z/8(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.

USAGE:

While Moving	Yes	Default Value	6
In a Program	Yes	Default Format	4.2
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"KP" on page 211	Proportional Gain
"KI" on page 210	Integral constant

LE

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. LE? or _LE returns the length of the vector in counts. The VE command is interchangeable with the LE command.

ARGUMENTS: None

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"LI" on page 213	Linear Distance
"BG" on page 155	BGS - Begin Sequence
"LM" on page 215	Linear Interpolation Mode
"VS" on page 276	Vector Speed
"VA" on page 267	Vector Acceleration
"VD" on page 268	Vector Deceleration

EXAMPLES:

```
LM ZW                                Specify linear interpolation mode
LI ,,100,200                         Specify linear distance
LE                                   End linear move
BGS                                  Begin motion
```

LI

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 DMC-1000 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 DMC-1000 sequence buffer to ensure continuous motion. 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. It should be noted that the DMC-1000 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 LI a,b,c,d,e,f,g,h

x,y,z,w are signed integers in the range -8,388,607 to 8,388,607 and represent incremental move distance

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"LE" on page 212	Linear end
"BG" on page 155	BGS - Begin sequence
"LM" on page 215	Linear Interpolation Mode
"CS" on page 167	Clear Sequence
"VS" on page 276	Vector Speed
"VA" on page 267	Vector Acceleration
"VD" on page 268	Vector Deceleration

EXAMPLES:

LM XYZ	Specify linear interpolation mode
LI 1000,2000,3000	Specify distance
LE	Last segment
BGS	Begin sequence

LM

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 DMC-1000 sequence buffer has room for additional segments. _LM or LM? may be used to return the number of spaces available in the sequence buffer for additional LI commands. 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 DMC-1000 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

XYZW denote X,Y,Z or W axes

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"LE" on page 212	Linear end
"LI" on page 213	Linear distances
"BG" on page 155	BGS - Begin sequence
"VA" on page 267	Vector acceleration
"VS" on page 276	Vector Speed
"VD" on page 268	Vector deceleration
"AV" on page 154	Vector distance
"CS" on page 167	_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

LS

FUNCTION: List

DESCRIPTION:

The LS command sends a listing of the programs in memory the PC bus. The listing will start with the line pointed to by the first parameter, which can be either a line number or a label. If no parameter is specified, it will start with line 0. The listing will end with the line pointed to by the second parameter--again either a line number or label. If no parameter is specified, the listing will go to the last line of the program.

ARGUMENTS: LS n,m

where n,m are valid numbers from 0 to 499, or labels. n is the first line to be listed, m is the last.

USAGE:

While Moving	Yes	Default Value	0,Last Line
In a Program	No	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

```
:LS #A,6          List program starting at #A through line 6
002 #A
003 PR 500
004 BGX
005 AM
006 WT 200
```

HINT: Remember to quit the Edit Mode <cntrl> Q prior to giving the LS command.

MC

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.

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:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"BG" on page 155	(returns a 0 if motion complete)
"AM" on page 148	
"TW" on page 264	

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

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:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"AD" on page 145	for relative distances
"MR" on page 223	
"AP" on page 150	

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

FUNCTION: Message

DESCRIPTION:

The MG command sends data out the bus. This can be used to alert an operator, send instructions or return a variable value.

ARGUMENTS: MG"m",V{Fm.n} {N}

"m" is a text message including letters, numbers, symbols or <ctrl>G (up to 35 characters).

V is the variable name.

{Fm.n} designates decimal format with m digits to left of decimal, and n to the right. Hex format is specified by {$m.n}.

{N} suppresses carriage return line feed.

{Sn} specifies format for string variable where n is 1 thru 6

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	Variable Format
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

Case 1: Message command displays ASCII strings

MG "Good Morning" — Displays the string

Case 2: Message command displays variables or arrays with specified format

MG "The Answer is", Total {F4.2} — Displays the string with the content of variable TOTAL in local format of 4 digits before and 2 digits after the decimal point.

Case 3: Message command sends any ASCII characters to the port.

MG {^13}, {^30}, {^37}, {N} — Sends carriage return, characters 0 and 7 followed by no carriage return line feed command to the port.

MO

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 X,Y,Z,W are XYZW axes

USAGE:

While Moving	No	Default Value	0
In a Program	Yes	Default Format	1.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"SH" on page 246	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

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	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"AD" on page 145	for relative distances
"MF" on page 220	
"AP" on page 150	

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

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

-2 - Step motor - Active low pulses

USAGE:

While Moving	Yes	Default Value	1,1,1,1
In a Program	Yes	Default Format	1
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"CN" on page 163	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 jumpers and use the J4 20-pin connector on the DMC-1000 to bring out the Pulse and Direction signals. Use the CN command to configure the pulse width.

NO

FUNCTION: No Operation

DESCRIPTION:

The NO command performs no action in a sequence, but can be used as a comment in a program. After the NO, up to 37 characters can be given to form a program comment. This helps to document a program.

ARGUMENTS: NO m,

where m is any group of letter, number, symbol or <cntrl>G

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in 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

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 is 1 through 8 denoting output bit (DMC-1040)

where n is 1 through 16 (DMC-1080)

expression is any valid logical expression, variable or array element.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

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, otherwise set bit 3
OB N, COUNT[1]	If element 1 in the array is zero, clear bit N

OE

FUNCTION: Off on Error

DESCRIPTION:

The OE command sets the DMC-1000 to shut off the motor command if a position error in excess of the limit specified by the ER command occurs. The OE1 command enables this function, while the OE0 command disables it. When the OE command is enabled, an Abort either from the Abort input or the Abort command will shut off the motor.

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

where x,y,z,w may be 0 or 1. 0 disables function. 1 enables off-on-error.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"AB" on page 143	Abort
"ER" on page 187	Error limit
"SH" on page 246	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

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,dc,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 .001.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.4
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

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

FUNCTION: Output Port

DESCRIPTION:

The OP command sends 8 bits of data to the output port of the controller. You can use the output port to control external switches and relays. The n parameter is used to specify the number of bits effected starting with the LSB. The other bits are masked. For example, if n=2, only outputs 1 and 2 will be changed by OP m. If the n parameter is not specified, all bits will be changed.

ARGUMENTS: OP m,n

where m is an integer in the range 0 to 255 decimal, or $0 to FF hexadecimal

n is an integer in the range 1 to 8 decimal

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	3.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"SB" on page 244	Set output bit
"CB" on page 159	Clear output bit

EXAMPLES:

OP 0	Clear Output Port -- all bits
OP $85	Set outputs 1,3,8; clear the others
OP $85,2	Set 1; clears 2. Other unchanged.
MG _OP	Message out the status of the outputs

PA

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

USAGE:

While Moving	No	Default Value	---
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"PR" on page 232	Position relative
"SP" on page 247	Speed
"AC" on page 144	Acceleration
"DC" on page 169	Deceleration
"BG" on page 155	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
0000000,0000000,0000000,0000000	
: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.

PF

FUNCTION: Position Format

DESCRIPTION:

The PF command allows the user to format the position numbers such as those returned by TP. The number of digits of integers and the number of digits of fractions can be selected with this command. An extra digit for sign and a digit for decimal point will be added to the total

number of digits. If PF is minus, the format will be hexadecimal and a dollar sign will precede the characters. Hex numbers are displayed as 2's complement with the first bit used to signify the sign.

If a number exceeds the format, the number will be displayed as the maximum possible positive or negative number (i.e. 999.99, -999, $8000 or $7FF).

ARGUMENTS: PF m.n

where m is an integer between -8 and 10

n is an integer between 0 and 4

The negative sign for m specifies hexadecimal representation.

USAGE:

While Moving	Yes	Default Value	10.0
In a Program	Yes	Default Format	10.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

EXAMPLES:

:TPX	Tell position of X
0000000000	Default format
:PF 5.2	Change format to 5 digits of integers and 2 of fractions
:TPX	Tell Position
00021.00	
PF-5.2	New format Change format to hexadecimal*
:TPX	Tell Position
$00015.00	Report in hex

PR

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

USAGE:

While Moving	No	Default Value	0
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"PA" on page 230	Position Absolute
"BG" on page 155	Begin
"AC" on page 144	Acceleration
"DC" on page 169	Deceleration
"SP" on page 247	Speed
"IP" on page 204	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
0000000100,0000000200,0000000300,0000000400	
:PR 500	
:BG	The X-axis will go 500 counts on the next move while the Y-axis will go its previously set relative distance.

QD

FUNCTION: Download Array

DESCRIPTION:

The QD command transfers array data from the host computer to the DMC-1500. QD array[],start,end requires that the array name be specified along with the first element of the array and last element of the array. The downloaded array is terminated by a <control>Z, <control>Q, <control>D or \.

ARGUMENTS: QD array[],start,end

where array[] is valid array name start is first element of array (default=0) end is last element of array (default=last element)

RELATED COMMANDS:

"QU" on page 234	Upload array

QU

FUNCTION: Upload Array

DESCRIPTION:

The QU command transfers array data from the DMC-1500 to a host computer. QU array[],start,end,comma requires that the array name be specified along with the first element of the array and last element of the array. If comma is 1, then the array elements will be separated by a comma. Otherwise, the elements will be separated by a carriage return. The uploaded array will be followed by a <control>Z as an end of text marker.

ARGUMENTS: QU array[],start,end,comma

where array[] is valid array name start is first element of array (default=0) end is last element of array (default=last element) comma -- if it is a 1, then elements are separated by a comma, else a carriage return

RELATED COMMANDS:

"QD" on page 233	Download array

RA

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	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"DM" on page 172	Dimension Array
"RD" on page 237	Record Data
"RC" on page 236	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

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 2n 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? or V=_RC

returns a 1 if recording

returns a 0 if not recording

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"DM" on page 172	Dimension Array
"RD" on page 237	Record Data
"RA" on page 235	Record Array Mode

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

FUNCTION: Record Data

DESCRIPTION:

The RD command specifies the data type to be captured for the Record Array (RA) mode. _RD returns the address for the next array element for recording. The command type includes:

_DEX	2nd encoder position (dual loop)
_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

where X,Y,Z or W may be specified.

ARGUMENTS: RD _TI,_TPX,_SVZ,_TSY

where _TI specifies the data type to be captured. The order is important. Each of the four data types correspond with the array specified in the RA command.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

"RA" on page 235	Record Array
"RC" on page 236	Record Interval
"DM" on page 172	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

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	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	No		
Can be Interrogated	No		
Used in an Operand	No		

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

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	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

#ININT	Input interrupt subroutine
"II" on page 200	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

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 X,Y,Z,W are X,Y,Z,W axes

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMAND:

"AL" on page 147	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
```

RP

FUNCTION: Reference Position

DESCRIPTION:

This command returns the commanded reference position of the motor(s).

ARGUMENTS: RP XYZW   RP ABCDEFGH

where XYZW are X,Y,Z,W axes

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMAND:

"TP" on page 259

"TE" on page 254

Note: The relationship between RP, TP and TE is that the position error, _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
: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: The RP command is useful when operating step motors.

RS

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	Default Value	---
In a Program	No	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

RS	Reset the controller

<control>R<control>S

FUNCTION: Master Reset

DESCRIPTION:

The Master Reset command resets the DMC-1000 to factory default settings and erases EEPROM. You can also perform a hardware master reset by installing a jumper on the DMC-1000 at the location labeled MRST and performing power-up. Remove the jumper after this procedure.

SB

FUNCTION: Set Bit

DESCRIPTION:

The SB command sets one of eight bits on the output port.

ARGUMENTS: SB n

where n is an integer in the range 1 to 8 decimal.

where n is an integer in the range 1 to 16 (DMC-1080)

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMAND

"OP" on page 229	Configure output port
"CB" on page 159	Clear Bit

EXAMPLES:

SB 5	Set output line 5
SB 1	Set output line 1

SC

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
0	Motors are running, independent mode
1	Motors stopped at commanded independent position
2	Decelerating or stopped by FWD limit switches
3	Decelerating or stopped by REV limit switches
4	Decelerating or stopped by Stop Command (ST)
6	Stopped by Abort input
7	Stopped by Abort command (AB)
8	Decelerating or stopped by Off-on-Error (OE1)
9	Stopped after Finding Edge (FE)
10	Stopped after Homing (HM)
50	Contour running
51	Contour Stop
99	MC timeout
100	Motors are running, vector sequence
101	Motors stopped at commanded vector

ARGUMENTS: SC XYZW SC ABCDEFGH

where XYZW or ABCDEFGH are the axes

USAGE:

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		

EXAMPLES:

```
Tom=_SCW                                   Assign the Stop Code of W to variable Tom
```

SH

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 XYZW are X,Y,Z,W axes

USAGE:

While Moving	No	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"MO" on page 222	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
```

SP

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. The units of the parameter are in counts per second.

ARGUMENTS: SP x,y,z,w SPX=x SP a,b,c,d,e,f,g,h

where x,y,z, are unsigned numbers in the range 0 to 8,000,000

USAGE:

While Moving	Yes	Default Value	25000
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"AC" on page 144	Acceleration
"DC" on page 169	Deceleration
"PR" on page 232	Position Relation
"PA" on page 230	Position Absolute
"BG" on page 155	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

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 XYZW are X,Y,Z,W axes. No parameters will stop motion on all axes and stop program.

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"BG" on page 155	Begin Motion
"AB" on page 143	Abort Motion
"AM" on page 148	Wait for motion end
"DC" on page 169	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

FUNCTION: Tell Status Byte

DESCRIPTION:

The TB command returns status information from the controller.

	High Denotes
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: None

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	1.0
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

TB	Tell status information from the controller
65	Executing program and echo on
	(26 + 20 = 64 + 1 = 65)

TC

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.

Error Codes:

1	Unrecognized command
2	Command only valid from program
3	Command not valid in program
4	Operand error
5	Input buffer full
6	Number out of range
7	Command not valid while running
8	Command not valid while not running
9	Variable error
10	Empty program line or undefined label
11	Invalid label or line number
12	Subroutine more than 8 deep
13	JG only valid when running in jog mode
14	EEPROM check sum error
15	EEPROM checkwrite error
16	IP incorrect sign during position move or IP given during forced deceleration
17	ED, BN and DL not valid while program running
18	Command not valid when contouring
20	Begin not valid with motor off
21	Begin not valid while running
22	Begin cannot be executed because of Limit Switch
24	Begin not valid because no sequence defined
25	Variable not given in IN command
29	Not valid during coordinated move
30	Sequence segment too short
31	Total move distance in a sequence > 2 billion
32	More than 511 segments in a sequence
41	Contouring record range error

42	Contour data being sent too slowly
46	Gear axis both master and follower
47	Gearing and coordinated moves cannot run simultaneously
50	Not enough fields
51	Question mark not valid
52	Missing " or string too long
53	Error in { }
54	Question mark part of string
55	Missing [or []
56	Array index invalid or out of range
57	Bad function or array
58	Unrecognized command in a command response (i.e._GNX)
59	Mismatched parentheses
60	Download error - line too long or too many lines
61	Duplicate or bad label
65	IN command must have a comma
66	Array space full
67	Too many arrays or variables
71	IN only valid in task #0
80	Record mode already running
81	No array or source specified
82	Undefined array
90	Only X Y Z W valid operand
95	TM too large for stepper pulse
96	SM jumper needs to be installed for stepper motor operation

ARGUMENTS: TC n

n=0 returns code only

n=1 returns code and message

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	3.0
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

EXAMPLES:

:GF32	Bad command
?TC	Tell error code
001	Unrecognized command

TD

FUNCTION: Tell Dual Encoder

DESCRIPTION:

This command returns the current position of the dual (auxiliary) encoder(s).

ARGUMENTS: TD XYZW TD ABCDEFGH

where XYZW are X,Y,Z,W axes

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"DE" on page 170	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

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 XYZW are X,Y,Z,W axes

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

"OE" on page 227	Off On Error
"ER" on page 187	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

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	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

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	3.0
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

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

TL

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 a resolution of .001 volt.

USAGE:

While Moving	Yes	Default Value	9.9988
In a Program	Yes	Default Format	1.4
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

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

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-1010 is 250 µsec; 375 µsec for the DMC-1020; 500 µsec for the DMC-1030 and 500 µsec for the DMC-1040.

USAGE:

While Moving	Yes	Default Value	1000
In a Program	Yes	Default Format	5.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

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

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, at which the resulting angle of the tangent axis equals zero in the coordinated motion plane. The tangent axis is specified with the VM n,m,p command where p is the tangent axis. _TN gives you the position of the first tangent point. Tangent 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. m is a signed number in the range +/-127 with a fractional resolution of 0.0001.

n is the absolute position at which the tangent angle is zero

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"VM" on page 271	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

FUNCTION: Tell Position

DESCRIPTION:

This command returns the current position of the motor(s).

ARGUMENTS: TP XYZW TP ABCDEFGH

where XYZW are X,Y,Z,W axes

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

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

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

USAGE:

While Moving	Yes	Default Value	TR0
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

TS

FUNCTION: Tell Switches

DESCRIPTION:

TS returns the state of the Home, Forward Limit and Reverse Limits for each axis. TS also returns error, motion and motor 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 XYZW designate X,Y,Z,W axes

USAGE:

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		

EXAMPLES:

:V1=_TSY	Assigns value of TSY to the variable V1

TT

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 XYZW specify X,Y,Z,W axes

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	1.4
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"TL" on page 256	Torque Limit

EXAMPLES:

V1=_TTX	Assigns value of TTX to variable, V1
TTX	Report torque on X
-0.2843	Torque is -.2843 volts

TV

FUNCTION: Tell Velocity

DESCRIPTION:

The TV command returns the actual velocity of the axes in units of quadrature count/s.

ARGUMENTS: TV XYZW TV ABCDEFGH

where XYZW specifies X,Y,Z,W axes

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	7.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

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

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 completeion 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 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

USAGE:

While Moving	Yes	Default Value	32766
In a Program	Yes	Default Format	
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"MC" on page 218

UI

FUNCTION: User Interrupt

DESCRIPTION:

The UI command causes an interrupt on the selected IRQ line. There are 16 user interrupts where UI n, n = 0 through 15. Prior to using the UI command, one IRQ line must be jumpered on the DMC-1000 and the data 2 and 4 written to the control register at address N + 1. An interrrupt service routine must also be incorporated in your host program. The interrupt condition can be read by writing a 6 to address N + 1 and then reading address N + 1. Refer to Section 4.6 for details.

ARGUMENTS: UI n where n is an integer between 0 and 15.

EXAMPLES:

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

EXAMPLES:

#I	Label
PR 10000	Position relative
SP 5000	Speed
BGX	Begin motion
AS	Wait for at speed
UI 1	Send interrupt 1
EN	End program

This program sends an interrupt to the selected IRQ line. The host writes a 6 to address N + 1 and then reads address N + 1 to receive data E1 which corresponds to UI1.

UL

FUNCTION: Upload

DESCRIPTION:

The UL command transfers data from the DMC-1000 to a host computer through port 1. Programs are sent without line numbers. The Uploaded program will be followed by a <control>Z or a \ as an end of text marker.

When used as an operand, _UL gives the number of available variables. The total number of variables is 126 for DMC-1000 and 500 for DMC-1000-MX.

ARGUMENTS: None

USAGE:

While Moving	Yes	Default Value	---
In a Program	No	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMAND:

"DL" on page 171	Download

EXAMPLES:

UL;	Begin upload
#A	Line 0
NO This is an Example	Line 1
NO Program	Line 2
EN	Line 3
<cntrl>Z	Terminator

VA

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.

USAGE:

While Moving	Yes	Default Value	262144
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"VS" on page 276	Vector Speed
"VP" on page 273	Vector Position
"VE" on page 269	End Vector
"CR" on page 165	Circle
"VM" on page 271	Vector Mode
"BG" on page 155	Begin Sequence
"VD" on page 268	Vector Deceleration
"VT" on page 277	Vector smoothing constant - S-curve

EXAMPLES:

```
VA 1024          Set vector acceleration to 1024 counts/sec2
VA ?             Return vector acceleration
00001024
VA 20000         Set vector acceleration
VA ?
0019456          Return vector acceleration
ACCEL=_VA        Assign variable, ACCEL, the value of VA
```

NOTE: This command is not valid in the DMC-1010.

VD

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.

USAGE:

While Moving	No	Default Value	262144
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"VA" on page 267	Vector Acceleration
"VS" on page 276	Vector Speed
"VP" on page 273	Vector Position
"CR" on page 165	Circle
"VE" on page 269	Vector End
"VM" on page 271	Vector Mode
"BG" on page 155	Begin Sequence
"VT" on page 277	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
```

NOTE: This command is not valid in the DMC-1010.

VE

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 ? or _VE returns the length of the vector in counts. VE is equivalent to the LE command.

ARGUMENTS: None

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"VM" on page 271	Vector Mode
"VS" on page 276	Vector Speed
"VA" on page 267	Vector Acceleration
"VD" on page 268	Vector Deceleration
"CR" on page 165	Circle
"VP" on page 273	Vector Position
"BG" on page 155	Begin Sequence
"CS" on page 167	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
```

VF

FUNCTION: Variable Format

DESCRIPTION:

The VF command allows the variables and arrays to be formatted for number of digits before and after the decimal point. When displayed, the value m respresents the number of digits before the decimal point, and the value n represents the number of digits after the decimal point. When in hexadecimal, the string will be preceded by a $. Hex numbers are displayed as 2's complement with the first bit used to signify the sign.

If a number exceeds the format, the number will be displayed as the maximum possible positive or negative number (i.e. 999.99, -999, $8000 or $7FF).

ARGUMENTS: VF m.n

where m and n are unsigned numbers in the range 0<m<10 and 0<n<4. A negative m specifies hexadecimal format

USAGE:

While Moving	Yes	Default Value	10.4
In a Program	Yes	Default Format	2.1
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

EXAMPLES:

VF 5.3	Sets 5 digits of integers and 3 digits after the decimal point
VF 8.0	Sets 8 digits of integers and no fractions
VF -4.0	Specify hexadecimal format with 4 bytes to the left of the decimal

VM

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-1000 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 ? or _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 nm is the plane of motion of any two axes X,Y,Z,W or

A,B,C,D,E,F,G,H

p is the tangent axis X,Y,Z,W or A,B,C,D,E,F,G,H. N turns off tangent.

USAGE:

While Moving	No	Default Value	X,Y
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"VP" on page 273	Vector Position
"VS" on page 276	Vector Speed
"VA" on page 267	Vector Acceleration
"VD" on page 268	Vector Deceleration
"CR" on page 165	Circle
"VE" on page 269	End Vector Sequence
"BG" on page 155	Begin Sequence
"CS" on page 167	Clear Sequence
"CS" on page 167	_CS - Segment counter
"VT" on page 277	Vector smoothing constant -- S-curve
"AV" on page 154	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

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.

When used as an operand, _VPX, _VPY, _VPZ, _VPW return 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.

ARGUMENTS: VP n,m < N

where n,m are signed integers in the range -2147483648 to 2147483647 The length of each segment must be limited to $8 \cdot 10^6$. N is an unsigned even integer between 0 and 8,000,000

USAGE:

While Moving	No	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

"CR" on page 165	Circle
"VM" on page 271	Vector Mode
"VA" on page 267	Vector Acceleration
"VD" on page 268	Vector Deceleration
"VE" on page 269	Vector End
"VS" on page 276	Vector Speed
"BG" on page 155	Begin Sequence
"VT" on page 277	Vector smoothing constant - S-curve

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
```

NOTE: This command is not valid in the DMC-1010.

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.*

VR

FUNCTION: Vector Speed Ratio

DESCRIPTION:

The VR r command multiplies the vector speed specifications given by VS or < by the value specified by r. r is between 0 and 10 with a resolution of .0001. VR takes effect immediately and will ratio all the following VS commands and any <n specifications used on VP, CR or LI segments. VR doesn't ratio the accelerations. VR is useful for feedrate override.

ARGUMENTS: VR r

where r is between 0 and 10 with a resolution of .0001

USAGE:

While Moving	Yes	Default Value	1
In a Program	Yes	Default Format	
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"VS" on page 276

EXAMPLES:

```
#A                      Vector Program
VMXY                    Vector Mode
VP 1000,2000            Vector Position
CR 1000,0,360           Specify Arc
VE                      End Sequence
VS 2000                 Vector Speed
BGS                     Begin Sequence
AMS                     After Motion
JP#A                    Repeat Move
#SPEED                  Speed Override
VR@AN[1]*.1             Read analog input, compute ratio
JP#SPEED                Loop
XQ#A,0; XQ#SPEED,1      Execute task 0 and 1 simultaneously
```

VS

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.

ARGUMENTS: VS n

where n is an unsigned number in the range 2 to 8000000 decimal

USAGE:

While Moving	Yes	Default Value	8192
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"VA" on page 267	Vector Acceleration
"VP" on page 273	Vector Position
"CR" on page 165	Circle
"LM" on page 215	Linear Interpolation
"VM" on page 271	Vector Mode
"BG" on page 155	Begin Sequence
"VE" on page 269	Vector End

EXAMPLES:

VS 2000	Define vector speed as 2000 counts/sec
VS ?	Return vector speed
002000	

VT

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

USAGE:

While Moving	Yes	Default Value	1.0
In a Program	Yes	Default Format	1.4
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"IT" on page 205	Independent Time Constant for smoothing independent moves

EXAMPLES:

VT 0.8	Set vector time constant
VT ?	Return vector time constant
0.8	

WC

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	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	No		

RELATED COMMANDS:

"CM" on page 162	Contour Mode
"CD" on page 160	Contour Data
"DT" on page 174	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

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:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in 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

FUNCTION: Execute Program

DESCRIPTION:

The XQ command starts a previously entered program. Execution will start at the label or line number specified. Up to four programs may be executed simultaneously to perform multitasking.

The function can be used as an operand where _XQn returns the line number for thread n, and -1 if thread n is not running.

ARGUMENTS: XQ #A,n XQm,n

where A is a program name of up to seven characters

where m is a line number

where n is the thread number (0,1,2 or 3) for multitasking

USAGE:

While Moving	Yes	Default Value	n = 0
In a Program	Yes	Default Format	---
Not in a Program	Yes		
Can be Interrogated	No		
Used in an Operand	Yes		

RELATED COMMANDS:

"HX" on page 199	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

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

USAGE:

While Moving	Yes	Default Value	.9143
In a Program	Yes	Default Format	3.0
Not in a Program	Yes		
Can be Interrogated	Yes		
Used in an Operand	Yes		

RELATED COMMANDS:

"GN" on page 196	Gain
"KD" on page 209	Derivative
"KP" on page 211	Proportional
"KI" on page 210	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

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 instruction _ZSM where M=X,Y,Z,W to iterrogate the status of the stacks for threads 0,1,2 or 3 respectively. The response, an integer between zero and seven, indicates zero for beginning condition and seven for the deepest value.

ARGUMENTS: ZS n

where 0 returns stack to original condition

1 eliminates one return on stack

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Not in a Program	No		
Can be Interrogated	Yes		
Used in an Operand	Yes		

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 14-bit DAC or .0012 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 selectable at 15.36 µsec, 1920 nsec or 960 nsec pulse width. 500,000 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 (available on DMC-1080 only)
IN[17] through IN[24] Inputs	TTL (available on DMC-1080 only)

Power

+5V	750 mA
+12V	40 mA
-12V	40mA

Performance Specifications

Minimum Servo Loop Update Time:	DMC-1010 -- 250 µsec
	DMC-1020 -- 375 µsec
	DMC-1030 -- 500 µsec
	DMC-1040 -- 500 µsec
Position Accuracy:	+/-1 quadrature count
Velocity Accuracy:	
Long Term	Phase-locked, better than .005%
Short Term	System dependent
Position Range:	+/-2,147,483,647 counts per move
Velocity Range:	Up to 8,000,000 counts/sec
Velocity Resolution:	2 counts/sec
Motor Command Resolution:	14 Bits or .0012V
Variable Range:	+/-2 billion
Variable Resolution:	$1 \cdot 10^{-4}$
Array Size:	1600 elements
	8000 elements - DMC-1040-MX and DMC-1080
Program Size:	500 lines x 40 characters
	1000 lines x 80 characters: DMC-1080
	2000 lines x 40 characters: DMC-1040-MX

Connectors

J2 - 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
21	Latch Y	22	Latch Z
23	Latch W	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 - 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 - 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 - 20 pin IDC

Stepper DIR/STEP
can sink 12ma

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 - 60 pin

For test only.

J7 - 10 pin

For test only.

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

CONNECTORS FOR DMC-1080 AUXILIARY BOARD FOR E,F,G,H AXES

JD2 - 60 pin IDC - Main

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-1040 card

JD5 - 26 pin IDC - I/O

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 - 60 pin - Connects to DMC-1040 J6

Pin-Out Description

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 32.6 KHz. 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 selected by the CN command and may be either .96 µsec, 1.92 µ sec or 15.33 µsec. 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 motors.

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-1080 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.*

Jumpers

Label Function (If jumpered)

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.
A2-A8	Seven Dip Switches for Address Selection. (Please follow silkscreen; not switch labels)
MRST	Master Reset enable. Returns controller to factory default settings and erases EEPROM. Requires power-on or RESET to be activated.
OPT	Reserved
IRQ 2/9	Interrupt Request line (Jumper one only)
IRQ 3	
IRQ 4	
IRQ 5	
IRQ 7	
IRQ 10	
IRQ 11	
IRQ 12	
IRQ 15	
IRQ 14	

Adjustment Pots

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

Accessories and Options

DMC-1010	Single Axis Controller
DMC-1020	Two-Axis Controller
DMC-1030	Three-Axis Controller
DMC-1040	Four-Axis Controller
DMC-1050	Five-Axis Controller
DMC-1060	Six-Axis Controller
DMC-1070	Seven-Axis Controller
DMC-1080	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.
DB-10096	Auxiliary board for additional 64 inputs, 32 output I/O. Can be configured for other sensors.
N23-54-1000	Servo motor; NEMA 23; 54 oz-in continuous
N34-150-1000	Servo motor; NEMA 34; 150 oz-in, continous
COM-Disk	MS-DOS Terminal Emulator and Software Sources
SDK-1000	Servo Design Software
OPINT	Operator Interface Software for PC
CAD-to-DMC	Autocad to DMC Translator
VBX Toolkit	Visual Basic VBX Extensions

WSDK-1000

WSDK (Windows based Servo Design Kit) has a number of functions that simplify the task of setting up a complete servo system. Each step of the servo system assembly process is addressed:

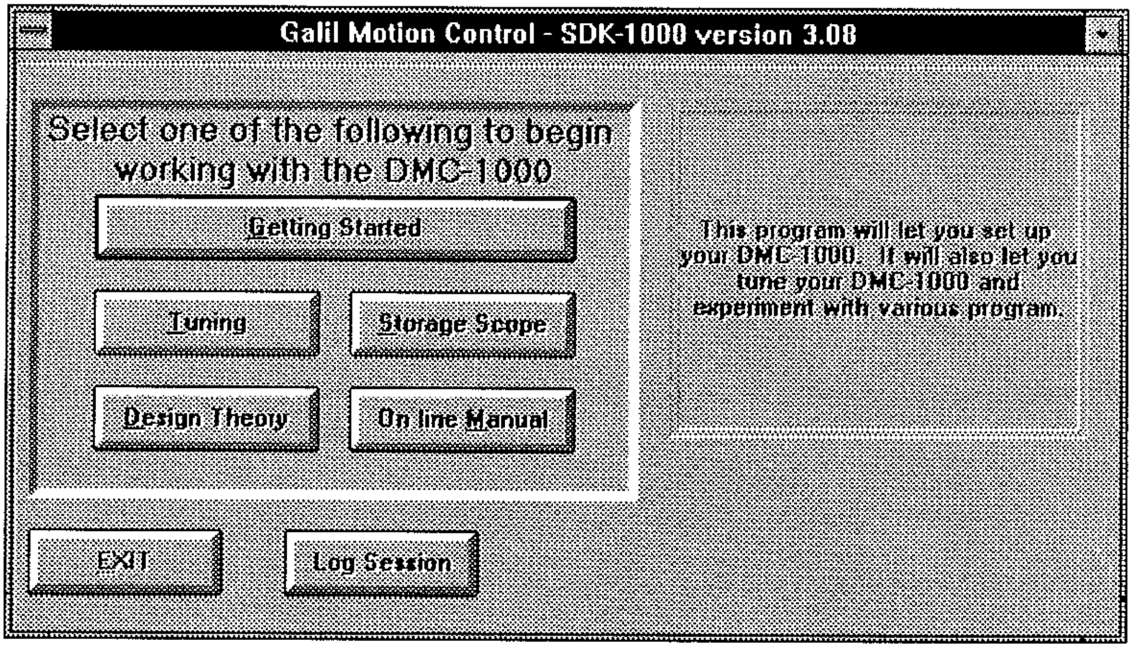
- Setting up communications
- Connecting the system components
- Automatic tuning of the servo motors
- Evaluation of the systems performance

Software Installation

This software must be installed under Windows. From the program manager select the 'File' menu then the 'Run' option. Enter "b:\setup" to install the software from the b: floppy drive. WSDK will be installed and a program group created with the WSDK-1000 icon.

Running SDK-1000

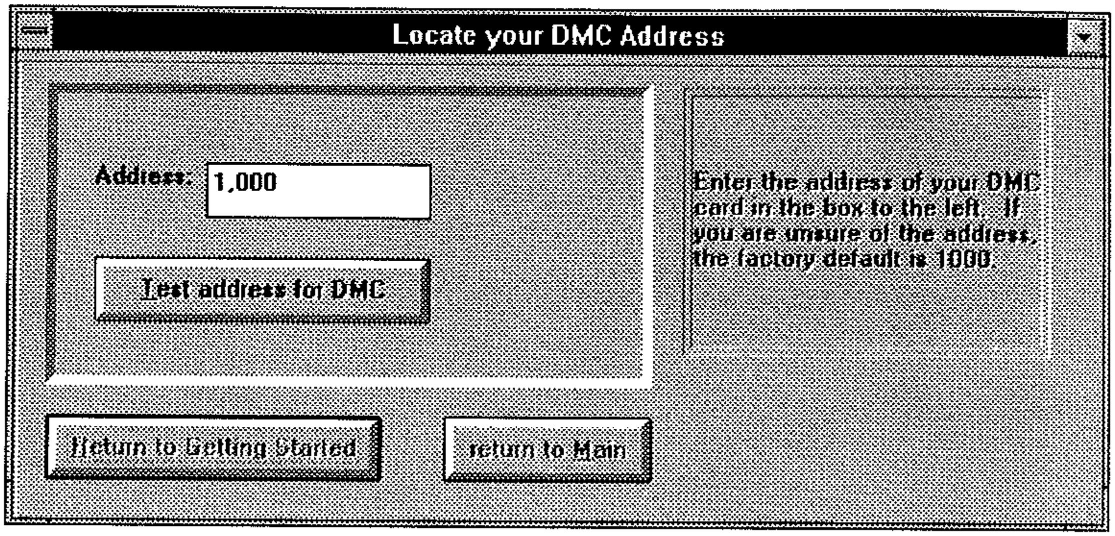
Double click on the SDK-1000 icon created during the installation process. The main window will appear. Use this window as a starting point to all the functions within SDK-1000. Refer to this picture in the following descriptions:


Galil Motion Control - SDK-1000 version 3.08
Select one of the following to begin working with the DMC-1000
Getting Started
Tuning
Storage Scope
Design Theory
On line Manual
EXIT
Log Session

This program will let you set up your DMC-1000. It will also let you tune your DMC-1000 and experiment with various program.

Setting up Communications

Prior to running WSDK for the first time make sure the serial cable is connected and the handshake dipswitch on the DMC-1000 is set to the ON position. To display the communications setup screen select the 'Getting Started' button on the main screen, then the 'Test address for DMC' button:



Locate your DMC Address

Address: 1,000

Enter the address of your DMC card in the box to the left. If you are unsure of the address, the factory default is 1000.

After setting the parameters test them with the "Test address for DMC" button. If a controller is not found check that the dip switch settings on the controller match the conditions selected on the screen.
If a controller is still not found quit the program, exit and restart Windows, and run SDK-1000 again. If the controller is still not found try changing the address of the card to another address, like 816.
(Remember to set the software to that value as well.)

Once the controller is found hit the "Return to Getting Started" button.

Connecting Components

Select the "Define Elements" button to select the type of motors, amplifiers and load the servo system contains. Values are for display purposes only and are not used in the automatic tuning portions of the software. Theoretical tuning parameters will be compared to actual values only in the theoretical evaluation sections of SDK-1000. This section can be skipped completely if this comparison is not required.

Select the "Connect Elements" button to begin putting the system together. Under this menu are a number of screens that will help explain the connections between the controller and ICM-1100 as well as the connection of encoders, amplifiers, and motors to the system.

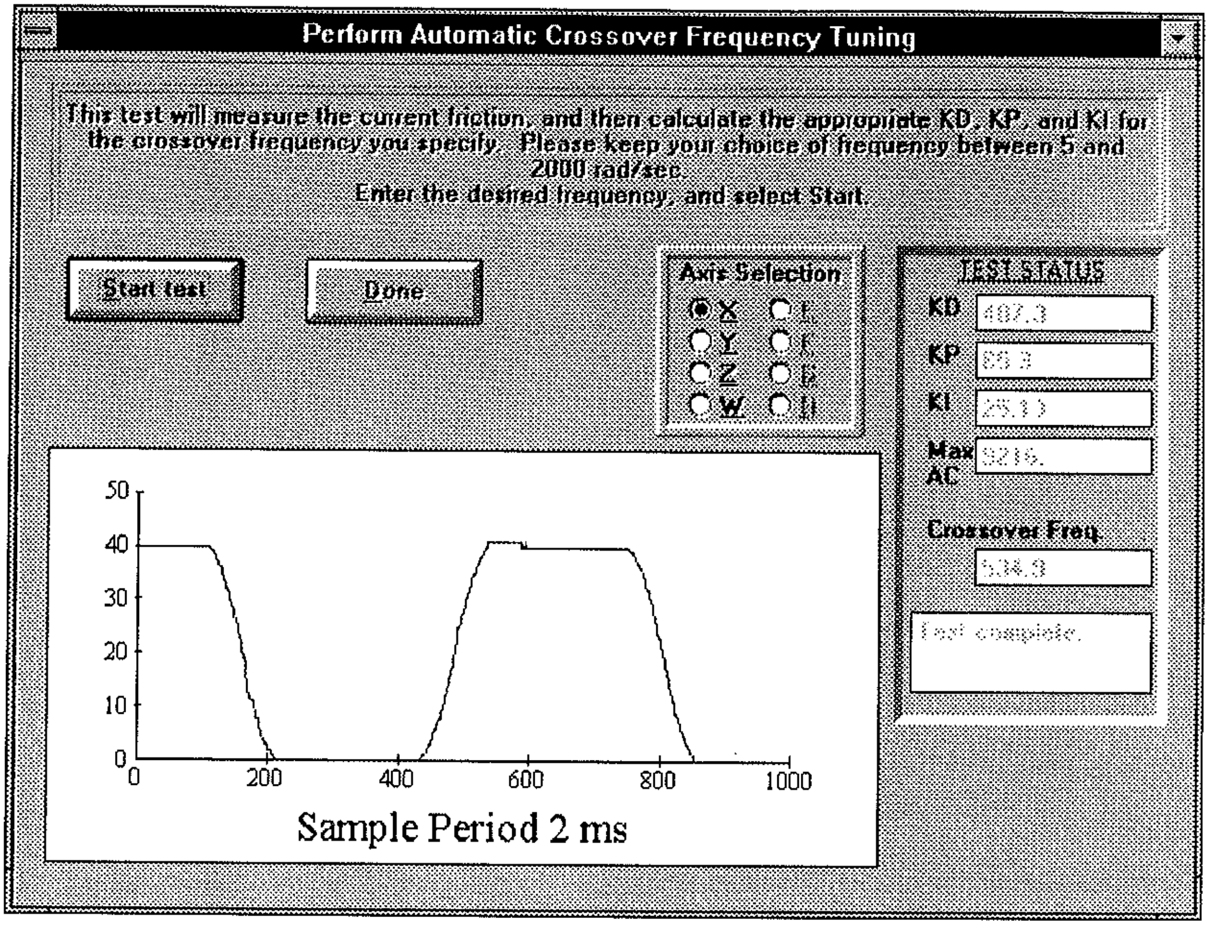
Automatic Tuning of the System

From the main screen select the "Tuning" button. There are a number of tuning routines available; at least one will provide adequate tuning parameters. If one type of tuning does not provide acceptable performance try another one. If any routine provides suitable values there is no need to continue with further autotuning routines.

> *Note: Most routines will produce tuning parameters that are not optimal. Some manual adjustments may be required to create desired results.*

Listed below are the routines currently offered in WSDK-1000:

- **Crossover Frequency:** This routine produces PID parameter values that will provide the best system response at the frequency (in rads/s) specified. The crossover frequency value should be set to the bandwidth of the system for best results.
- **Automatic Crossover Frequency:** Uses the same basic tuning routines as the standard crossover but will try a number of frequencies and select the one that gives the best system response. Use this routine if the bandwidth of the system is unknown.



Perform Automatic Crossover Frequency Tuning

This test will measure the current friction, and then calculate the appropriate KD, KP, and KI for the crossover frequency you specify. Please keep your choice of frequency between 5 and 2000 rad/sec.
Enter the desired frequency, and select Start.

Axis Selection

● X ○ E
○ Y ○ F
○ Z ○ G
○ W ○ H

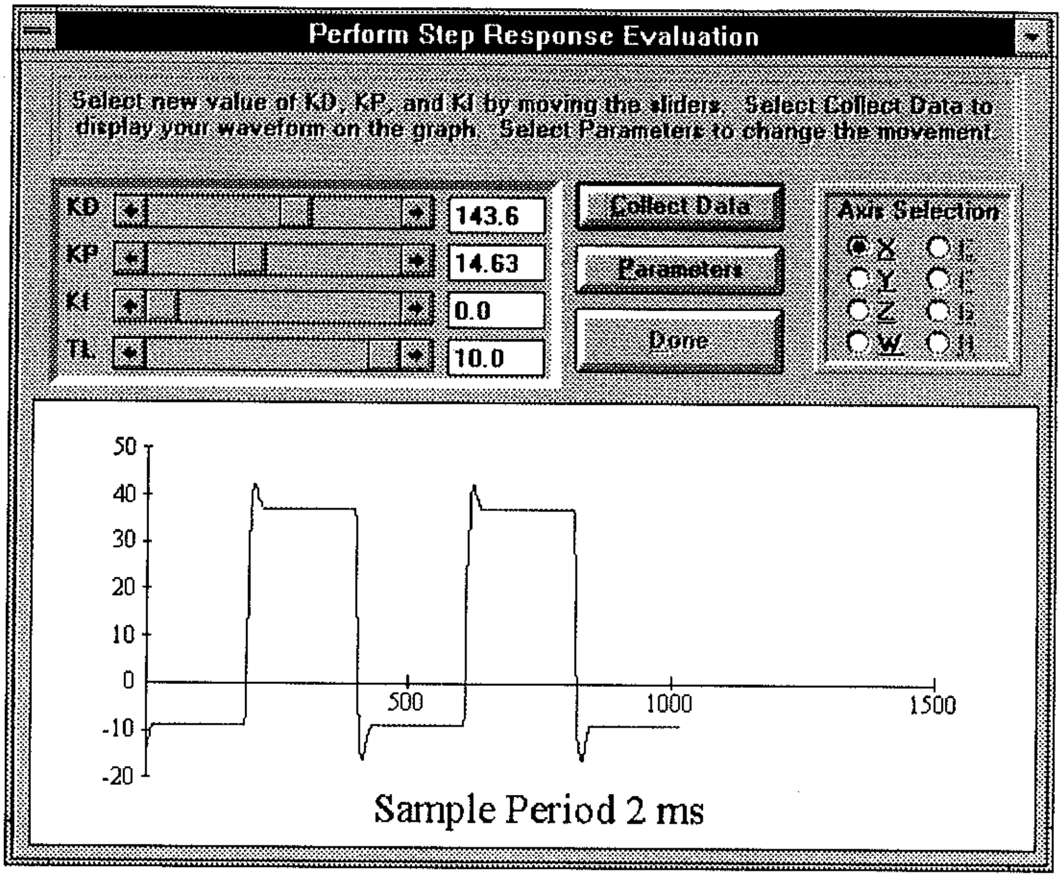
TEST STATUS

KD 407.3
KP 83.3
KI 25.1)
Max AC 9216.

Crossover Freq
534.9

Test complete.

- **Conservative Tuning:** This routine tunes the system by increasing the gain until instability occurs , then backs the gain down. As the name suggest, the resulting parameters are very conservative but will provide an operational system.
- **Manual Tuning:** Unlike the other tuning methods, this is not an autotuning routine. All tuning parameters can be adjusted with the sliders. Whenever the values are changed they are sent to the controller giving instant feedback as to the effect those changes made to the stability of the system.



Select new value of KD, KP, and KI by moving the sliders. Select Collect Data to display your waveform on the graph. Select Parameters to change the movement.

KD 143.6
KP 14.63
KI 0.0
TL 10.0

● X
○ Y
○ Z
○ W

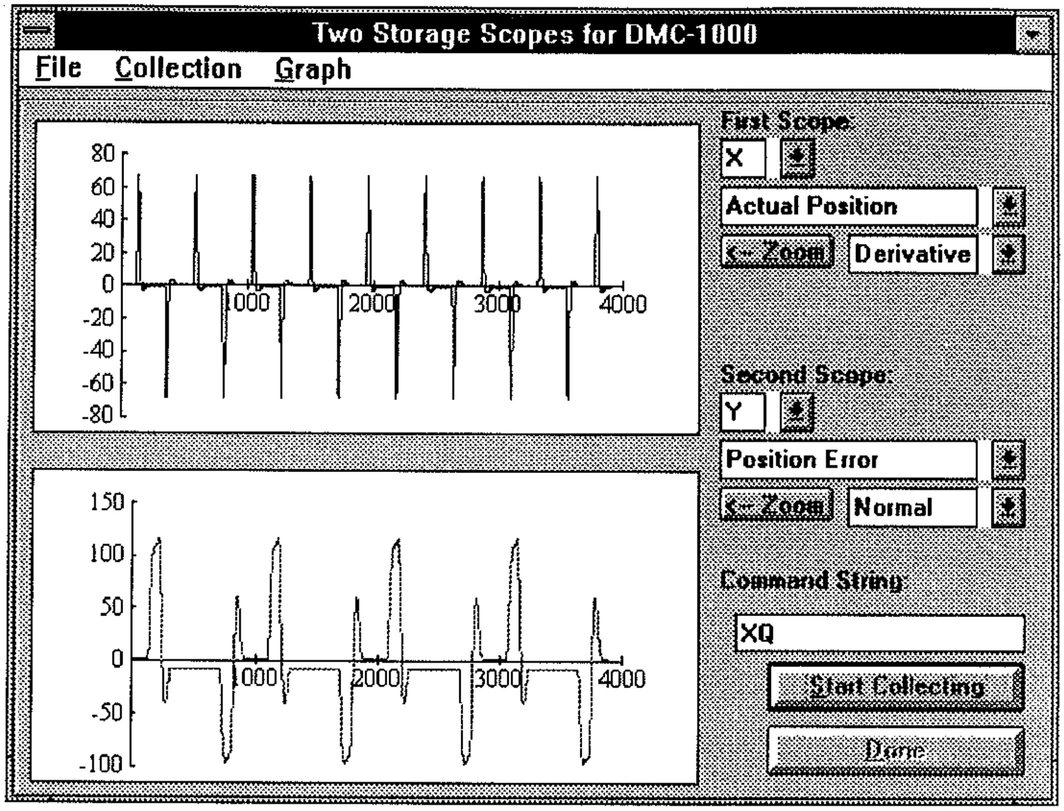
Sample Period 2 ms

Evaluation of System Performance

Once the tuning is complete the system should be ready for operation. Checking the response characteristics of the system can be done in two ways; use the autographing feature contained in each tuning procedure, or use the storage scope functions to design a custom test.

Storage scopes can be used to graph data from the card to the PC screen. Data can be presented versus time or some other axis. To use the scopes follow these simple steps:

- Select "Storage Scope" from the main screen
- Select the number of scopes to display
- From the lists provided, set the data to be displayed



First Scope: X

Actual Position

Derivative

Second Scope: Y

Position Error

Normal

Command String: XQ

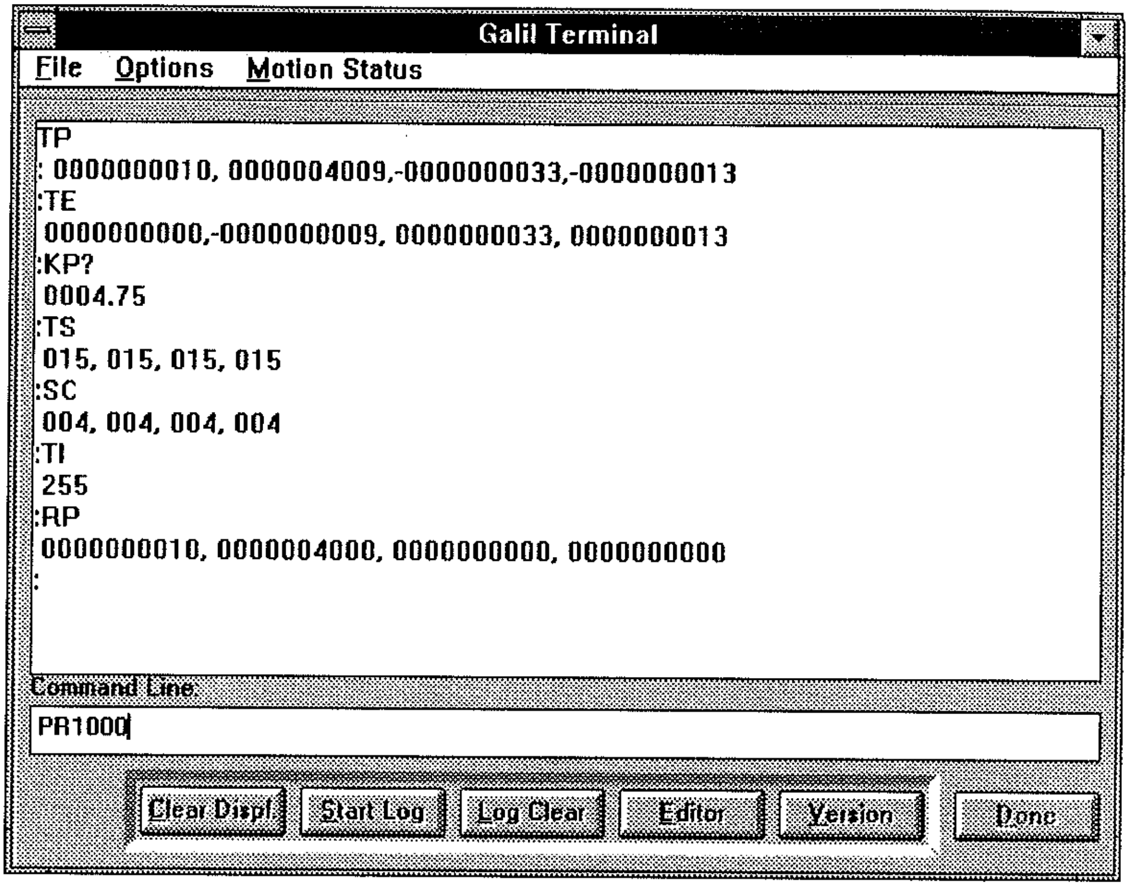
- Bring up the terminal (under the File menu)
- From the terminal type in a single move or entire program (see using the Terminal, below)
- Return to the storage scope and type in the command string. This string is sent at the same time data collection begins, making sure data collection is synchronized to the start of the motion.
- Select the "Start Collecting" button the gather and display the information

Other features of the scope under the 'Collection' menu include:

- Number of Sample Points: Increase this value to see more data on the graph
- Sample Period: Sets the period between consecutive data points

Terminal and Program Editor

Included in the SDK-1000 program is a terminal screen used to send commands to the controller and display any response. This tool provides a direct link to the system- allowing simple programming and status interrogation. To open the terminal select the "Storage Scope" button from the main menu then the "Terminal" button from the storage scope screen.



```
TP
: 0000000010, 0000004009,-0000000033,-0000000013
:TE
 0000000000,-0000000009, 0000000033, 0000000013
:KP?
 0004.75
:TS
 015, 015, 015, 015
:SC
 004, 004, 004, 004
:TI
 255
:RP
 0000000010, 0000004000, 0000000000, 0000000000
:
```

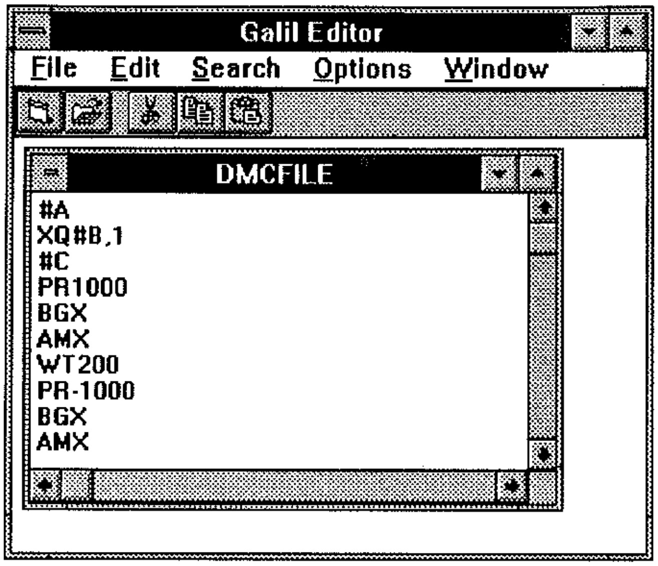
Command Line:

```
PR1000
```

Functions available in the terminal include a status polling windows located under the "Motion Status" menu, sending a master reset under the "Options" menu and viewing the version of the controllers firmware by selecting the "Version" button.

Program Editor

To bring up the editor select the "Editor" button or type "ED" in the command line section of the terminal. This brings up a simple text editor best suited for smaller programs (under 500 lines).



```
#A
XQ#B,1
#C
PR1000
BGX
AMX
WT200
PR-1000
BGX
AMX
```

Once a program is created it can be downloaded to the controller using the "Download" selection under the "File" menu. Likewise a file existing in the controller can be displayed by selecting "Upload File" from the same menu. Hitting the escape button on the keyboard will remove the editor from view, but the text information is not lost until SDK-1000 is shut down.

Design Theory

Included with SDK-1000 is a great deal of information pertaining to the design theory of motion control systems. This section, under the "Design Theory" button, is broken into two sections: system evaluation tools, and theoretical tutorial. The system evaluation tools include a frequency response test and a step response test. These can be used to help an experienced controls engineer evaluate the servo system. The design theory section contains information on the transfer functions of each system component and an overall transfer function of the servo loop.

On-line Help

On-line help is available for the SDK-1000 program. Please use this on-line feature for a more detailed description of the functions available in SDK-1000. The DMC-1000 manual is also available as on-line help. This resource is most useful as a lookup tool: using the index functions to look up a particular subject (i.e. "encoders").

Dip Switch Address Settings

Use this table to find the dip switch settings for any of the available addresses of the DMC-1000.

Address	Dip A8	Dip A7	Dip A6	Dip A5	Dip A4	Dip A3	Dip A2
512	x	x	x	x	x	x	x
516	x	x	x	x	x	x	
520	x	x	x	x	x		x
524	x	x	x	x	x		
528	x	x	x	x		x	x
532	x	x	x	x		x	
536	x	x	x	x			x
540	x	x	x	x			
544	x	x	x		x	x	x
548	x	x	x		x	x	
552	x	x	x		x		x
556	x	x	x		x		
560	x	x	x			x	x
564	x	x	x			x	
568	x	x	x				x
572	x	x	x				
576	x	x		x	x	x	x
580	x	x		x	x	x	
584	x	x		x	x		x
588	x	x		x	x		
592	x	x		x		x	x
596	x	x		x		x	
600	x	x		x			x
604	x	x		x			
608	x	x			x	x	x
612	x	x			x	x	
616	x	x			x		x
620	x	x			x		
624	x	x				x	x
628	x	x				x	
632	x	x					x
636	x	x					
640	x		x	x	x	x	x
644	x		x	x	x	x	
648	x		x	x	x		x
652	x		x	x	x		
656	x		x	x		x	x
660	x		x	x		x	
664	x		x	x			x
668	x		x	x			
672	x		x		x	x	x
676	x		x		x	x	

Address	Dip A8	Dip A7	Dip A6	Dip A5	Dip A4	Dip A3	Dip A2
680	x		x		x		x
684	x		x		x		
688	x		x			x	x
692	x		x			x	
696	x		x				x
700	x		x				
704	x			x	x	x	x
708	x			x	x	x	
712	x			x	x		x
716	x			x	x		
720	x			x		x	x
724	x			x		x	
728	x			x			x
732	x			x			
736	x				x	x	x
740	x				x	x	
744	x				x		x
748	x				x		
752	x					x	x
756	x					x	
760	x						x
764	x						
768		x	x	x	x	x	x
772		x	x	x	x	x	
776		x	x	x	x		x
780		x	x	x	x		
784		x	x	x		x	x
788		x	x	x		x	
792		x	x	x			x
796		x	x	x			
800		x	x		x	x	x
804		x	x		x	x	
808		x	x		x		x
812		x	x		x		
816		x	x			x	x
820		x	x			x	
824		x	x				x
828		x	x				
832		x		x	x	x	x
836		x		x	x	x	
840		x		x	x		x
844		x		x	x		
848		x		x		x	x
852		x		x		x	
856		x		x			x
860		x		x			
864		x			x	x	x
868		x			x	x	

Address	Dip A8	Dip A7	Dip A6	Dip A5	Dip A4	Dip A3	Dip A2
872		x			x		x
876		x			x		
880		x				x	x
884		x				x	
888		x					x
892		x					
896			x	x	x	x	x
900			x	x	x	x	
904			x	x	x		x
908			x	x	x		
912			x	x		x	x
916			x	x		x	
920			x	x			x
924			x	x			
928			x		x	x	x
932			x		x	x	
936			x		x		x
940			x		x		
944			x			x	x
948			x			x	
952			x				x
956			x				
960				x	x	x	x
964				x	x	x	
968				x	x		x
972				x	x		
976				x		x	x
980				x		x	
984				x			x
988				x			
992					x	x	x
996					x	x	
1000					x		x
1004					x		
1008						x	x
1012						x	
1016							x
1020							

PC/AT Interrupts and Their Vectors

(These occur on the first 8259)

IRQ	VECTOR	USAGE
0	8 or 08h	Timer chip (DON'T USE THIS!)
1	9 or 09h	Keyboard (DON'T USE THIS!)
2	10 or 0ah	Cascade from second 8259 (DON'T USE THIS!)
3	11 or 0bh	COM2:
4	12 or 0ch	COM1:
5	13 or 0dh	LPT2:
6	14 or 0eh	Floppy (DON'T USE THIS!)
7	15 or 0fh	LPT1:

(These occur on the second 8259)

IRQ	VECTOR	USAGE
8	104 or 70h	Real-time clock (DON'T USE THIS!)
9	105 or 71h	Redirect-cascade (DON'T USE THIS!)
10	106 or 72h	
11	107 or 73h	
12	108 or 74h	Mouse DSR
13	109 or 75h	Math Co-processor exception
14	110 or 76h	Fixed Disk (DON'T USE THIS!)
15	111 or 77h	

ICM-1100 Interconnect Module

The ICM-1100 Interconnect Module provides easy connections between the DMC-1000 series controllers and other system elements, such as amplifiers, encoders, and external switches. The ICM-1100 accepts each DMC-1000 ribbon cable (for J2, J3, J4 and J5) and breaks them into screw-type terminals. Each screw terminal is labelled 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-1000 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

ICM-1100 CONNECTIONS

Screw Terminals Internal DMC-1000 Connection

Screw Terminal No.	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	
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

Screw Terminal No.	Label	I/O	J2	J3	J4	J5	Description
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			21	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
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-

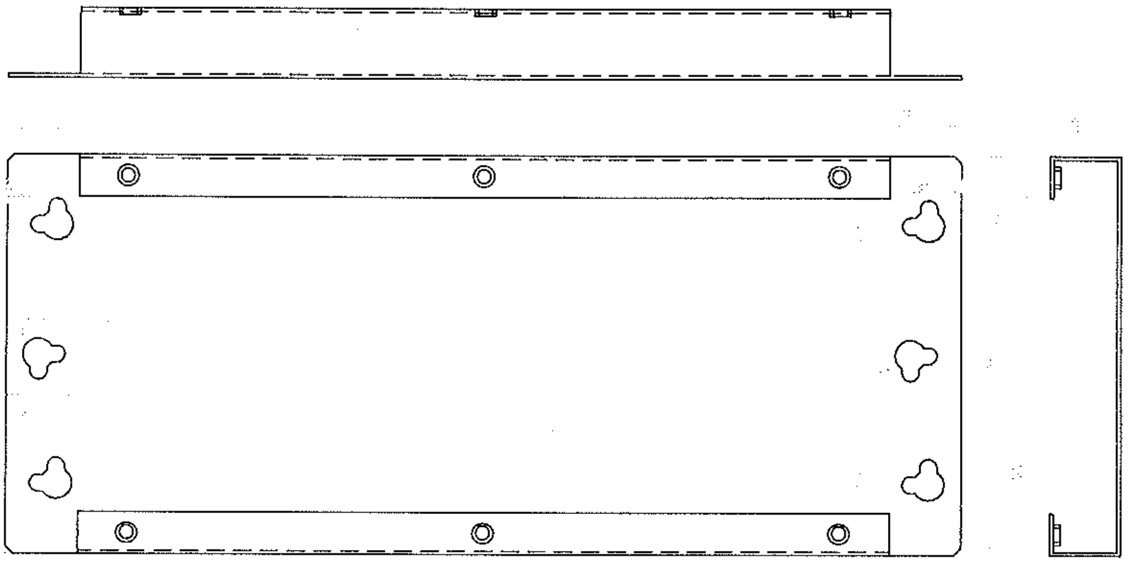
Screw Terminal No.	Label	I/O	J2	J3	J4	J5	Description
81	XI+	I	37				X Main encoder I+
82	XI-	I	38				X Main encoder I-
83	YA+	I	39				Y Main encoder A+
84	YA-	I	40				Y Main encoder A-
85	YB+	I	41				Y Main encoder B+
86	YB-	I	42				Y Main encoder B-
87	YI+	I	43				Y Main encoder I+
88	YI-	I	44				Y 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-
101	+12V		57				
102	-12V		58				
103	5V		2,59	19	15	9	
104	GND		1,60	20	20	8	

JX6, JY6, JZ6, JW6* (10-pin IDC)

1	A	2	5V
3	GND	4	
5	A-	6	A
7	B-	8	B
9	I-	10	I

*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-1000. 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-1000 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-1000 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

DB-10072 OPTO-22 Expansion Option

The DB-10072 is a separate full-length PC card designed to work with OPTO-22 I/O isolation products that feature the 50 pin IDC connector (i.e.: OPTO-22 model number G4PB24). It connects to the DMC-1000 and provides 72 I/O points. Three 50-pin cables may be connected to the card, each handling up to 24 I/O points. The first 48 I/O points can be configured through software (I/O conf iguration options shown below). I/O points 9 through 56 can be configured as inputs or outputs (in groups of 8). I/O points 57 through 80 are always inputs.

A new command, CO, is used to configure blocks of 8 bits as inputs or outputs. The command has one field:

CO n

n is a 6-bit number defined below.

Each 8-bit block is represented by one bit. That bit equals one if the block is to be used as outputs:

Outputs	Block	Bit
9-16	1	2^5
17-24	2	2^4
25-32	3	2^3
33-40	4	2^2
41-48	5	2^1
49-56	6	2^0

For example, if blocks 3, 5 and 6 are inputs, then CO 11 should be issued. (Note: n base2 is 001011which equals 11 in base 10). This parameter, and the state of the outputs, can be stored in the EEPROM with the BN command. If no value has been set, the default of CO 0 is used (all blocks are inputs).

When configured as an output, each I/O point may be defined with the SBn and CBn commands (where n=9 through 56). OBn can also be used with n=9 through 56.

The OQ command is implemented in a similar fashion as with the DB-10096. OQ m,n,o (where m, n and o range from 0 to 65535) may be used to set the state of 16 bits at one time. The data fields define the outputs as follows:

Field	Most significant to least significant byte
m	block 2 to 1
n	block 4 to 3
o	block 6 to 5

When OQ is used as an operand, a 0 will return the current state of blocks 2 to 1, a 1 returns 4 to 3, and a 2 returns 6 to 5. Example: MG_OQ2 returns the state of the bits in blocks 6 and 5.

When accessing I/O blocks configured as inputs, use the TIn command. The operant 'n' refers to the block to be read (n=1 to 9). Individual bits can be queried using the @IN[n] command (where n=9 to 80).

Examples:

If the commands below are issued

```
OQ,259
MG OQ1
```

the response is "259".

If the commands below are issued

```
OQ0
SB9
OB 10,1
MG OQ0
```

the response is "3".

If the command below is issued

```
MG @IN[17]
```

the response is the least significant bit of block 2 (assuming block 2 is configured as input).

Three cables connect the DB-10072 to OPTO-22 products. One cable is located at the back of the card and may be connected from outside the PC (J1). The other two are attached from within the PC (J2 and J3). Pinouts are described below:

J1 Pinout

Pin	Block	Bit No.	SBn.@IN[n]	Pin	
1	3	7	32	2	Ground
3	3	6	31	4	Ground
5	3	5	30	6	Ground
7	3	4	29	8	Ground
9	3	3	28	10	Ground
11	3	2	27	12	Ground
13	3	1	26	14	Ground
15	3	0	25	16	Ground
17	2	7	24	18	Ground
19	2	6	23	20	Ground
21	2	5	22	22	Ground
23	2	4	21	24	Ground
25	2	3	20	26	Ground
27	2	2	19	28	Ground
29	2	1	18	30	Ground
31	2	0	17	32	Ground
33	1	7	16	34	Ground
35	1	6	15	36	Ground
37	1	5	14	38	Ground
39	1	4	13	40	Ground
41	1	3	12	42	Ground
43	1	2	11	44	Ground
45	1	1	10	46	Ground
47	1	0	9	48	Ground
49	volts			50	Ground

J2 Pinout

Pin	Block	Bit No.	SBn,@IN[n]	Pin	
1	6	7	56	2	Ground
3	6	6	55	4	Ground
5	6	5	54	6	Ground
7	6	4	53	8	Ground
9	6	3	52	10	Ground
11	6	2	51	12	Ground
13	6	1	50	14	Ground
15	6	0	49	16	Ground
17	5	7	48	18	Ground
19	5	6	47	20	Ground
21	5	5	46	22	Ground
23	5	4	45	24	Ground
25	5	3	44	26	Ground
27	5	2	43	28	Ground
29	5	1	42	30	Ground
31	5	0	41	32	Ground
33	4	7	40	34	Ground
35	4	6	39	36	Ground
37	4	5	38	38	Ground
39	4	4	37	40	Ground
41	4	3	36	42	Ground
43	4	2	35	44	Ground
45	4	1	34	46	Ground
47	4	0	33	48	Ground
49	volts			50	Ground

J3 Pinout (Note: All points are inputs on this cable):

Pin	Block	Bit No.	@IN[n]
1	9	7	80
3	9	6	79
5	9	5	78
7	9	4	77
9	9	3	76
11	9	2	75
13	9	1	74
15	9	0	73
17	8	7	72
19	8	6	71
21	8	5	70
23	8	4	69
25	8	3	68
27	8	2	67
29	8	1	66
31	8	0	65
33	7	7	64
35	7	6	63
37	7	5	62
39	7	4	61
41	7	3	60
43	7	2	59
45	7	1	58
47	7	0	57
49	volts		

Pin	
2	Ground
4	Ground
6	Ground
8	Ground
10	Ground
12	Ground
14	Ground
16	Ground
18	Ground
20	Ground
22	Ground
24	Ground
26	Ground
28	Ground
30	Ground
32	Ground
34	Ground
36	Ground
38	Ground
40	Ground
42	Ground
44	Ground
46	Ground
48	Ground
50	Ground

DB-10096 I/O Expansion

The DB-10096 is an attachment board that provides an additional 64 inputs and 32 TTL-level outputs. (Other I/O configurations are available). The inputs are pulled up to 5 Volts with 4.7K resistors.

The 64 inputs may be read individually using the @IN[n] function where n=1 through 8 represent the standard 8 inputs on the DMC-1000 and n=9 through 72 represent the 64 inputs on the DB-10096. For example, V1=@IN[9] reads input 9 on the DB-10096 and assigns the value to variable V1.

Inputs may also be read in groups of 8 using the TIn command where n=0 through 8. n=0 reads inputs 1 through 8 on the DMC-1000, n=1 reads inputs 9 through 16 on the DB-10096, n=2 reads inputs 17 through 24 and so on as shown in the table below. For example, if inputs 17 through 24 are high, V1=_TI2 assigns the value 255 to variable V1.

TIn	Inputs
0	1-8
1	9-16
2	17-24
3	25-32
4	33-40
5	41-48
6	49-56
7	57-64
8	65-72

The AI command is only available for inputs 1 through 8 on the DMC-1000.

The 32 outputs are controlled using the SBn, CBn and OBn instructions where n=1 through 8 represent the 8 outputs on the DMC-1000 and n=9 through 40 represent the 32 outputs available on the DB-10096.

A new command, OQ, is available with the DB-10096. This command has two fields addressing 16 outputs each.

OQ m,n

The first field m controls outputs 9 to 24. The second field n controls 25 to 40. When OQ is used in an operand, a 0 will return inputs 9-24 and a 1 will return 25-40. For example, if outputs 9 and 10 are high and all others are low, then MG_OQ0 returns a 3.

The DB-10096 piggy-backs to the DMC-1000 with a ribbon cable. The inputs and outputs are available via two 60-pin ribbon cables, J1 (at side of DB-10096) and J2 (at the top). Pinouts are described below:

J1 Pinout

1 out 9	2 out 10
3 out 11	4 out 12
5 out 13	6 out 14
7 out 15	8 out 16
9 out 17	10 out 18
11 out 19	12 out 20
13 out 21	14 out 22
15 out 23	16 out 24
17 NC	18 GND
19 GND	20 out 25
21 out 26	22 out 27
23 out 28	24 out 29
25 out 30	26 out 31
27 out 32	28 out 33
29 out 34	30 out 35
31 out 36	32 out 37
33 out 38	34 out 39
35 out 40	36 NC
37 NC	38 GND
39 GND	40 NC
41 in 9	42 in 10
43 in 11	44 in 12
45 in 13	46 in 14
47 in 15	48 in 16
49 in 17	50 in 18
51 in 19	52 in 20
53 in 21	54 in 22
55 in 23	56 in 24
57 NC	58 GND
59 GND	60 5 Volts

J2 Pinout

1 in 25	2 in 26
3 in 27	4 in 28
5 in 29	5 in 30
7 in 31	8 in 32
9 in 33	10 in 34
11 in 35	12 in 36
13 in 37	14 in 38
15 in 39	16 in 40
17 NC	18 GND
19 GND	20 in 41
21 in 42	22 in 43
23 in 44	24 in 45
25 in 46	26 in 47
27 in 48	28 in 49
29 in 50	30 in 51
31 in 52	32 in 53
33 in 54	34 in 55
35 in 56	36 NC
37 NC	38 GND
39 GND	40 NC
41 in 57	42 in 58
43 in 59	44 in 60
45 in 61	46 in 62
47 in 63	48 in 64
49 in 65	50 in 66
51 in 67	52 in 68
53 in 69	54 in 70
55 in 71	56 in 72
57 NC	58 GND
59 GND	60 5 Volts

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
```

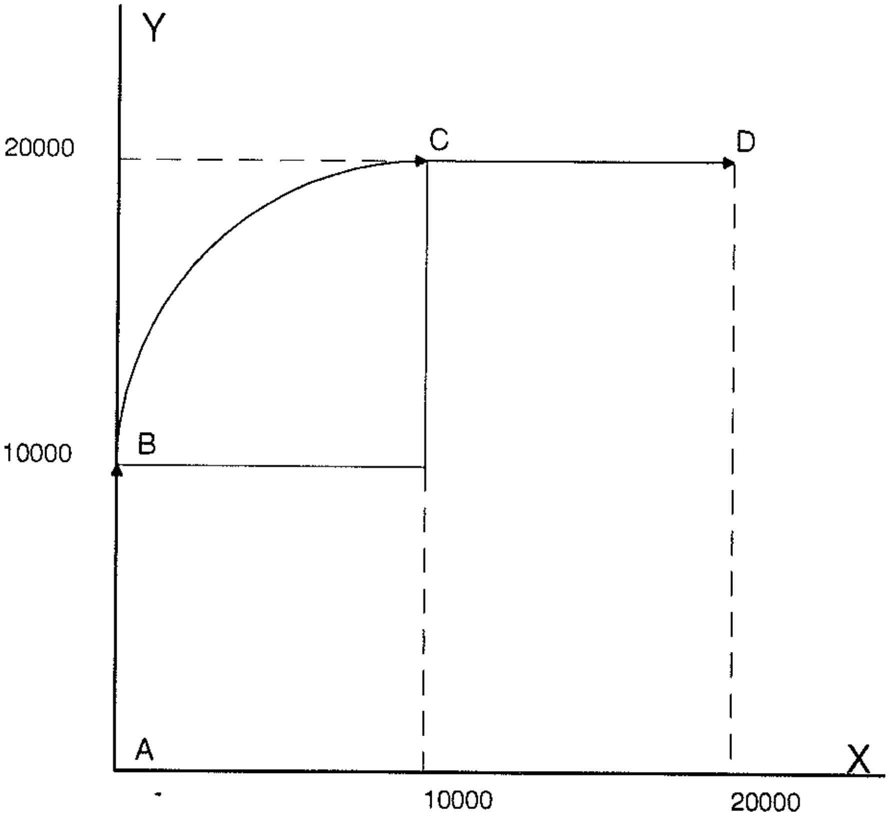


Figure 12.2 - X-Y Motion Path

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	$\frac{R\|\Delta\theta\|2\pi}{360} = 15708$
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

$$L_k = R_k|\Delta\Theta_k|2\pi/360$$

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.

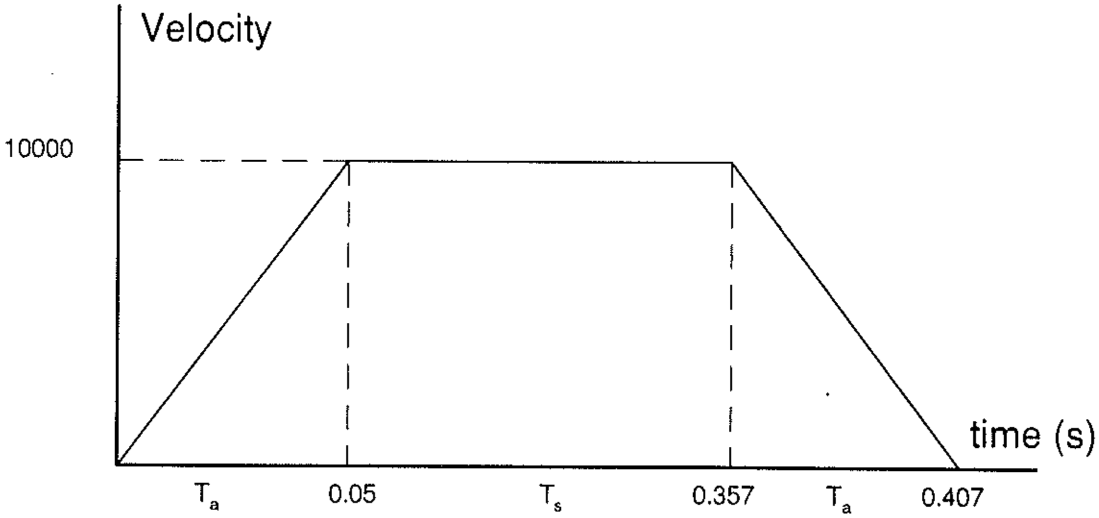


Figure 12.3 - Vector Velocity Profile

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.

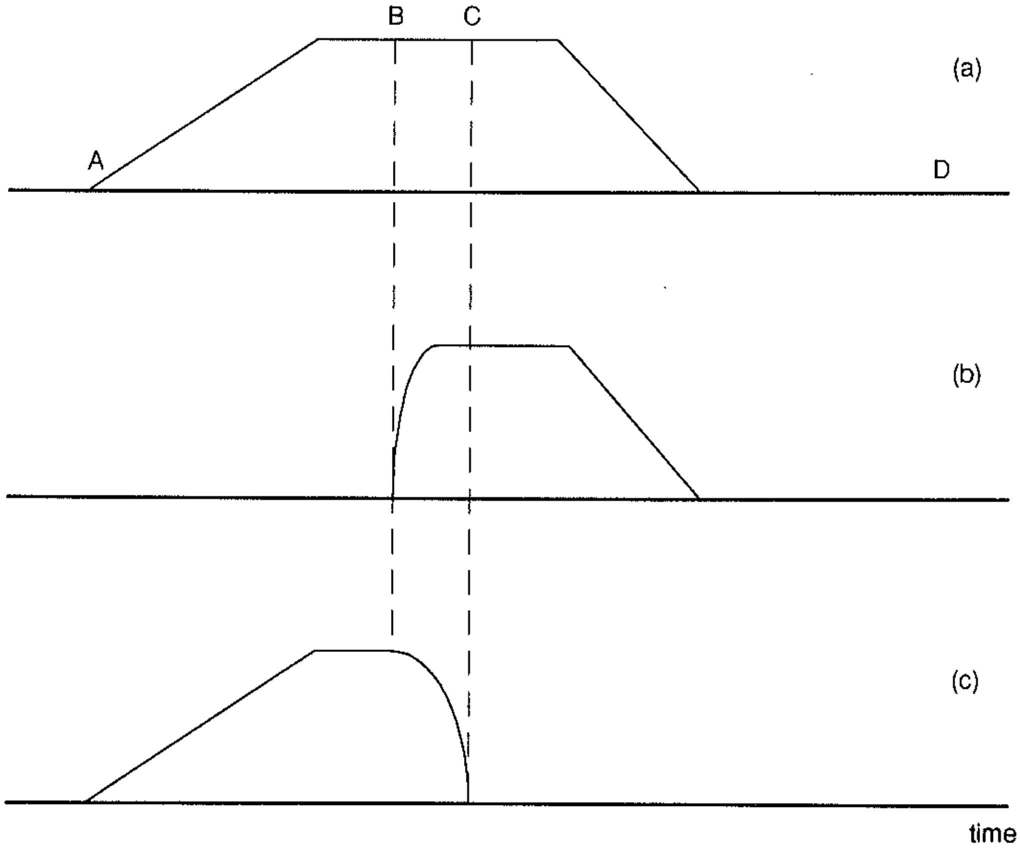


Figure 12.4 - Vector and Axes Velocities

DMC-600/DMC-1000 Comparison

Modes of Motion	DMC-600	DMC-6X1	DMC-1000
Relative positioning	Yes	Yes	Yes
Absolute positioning	Yes	Yes	Yes
Velocity control	Yes	Yes	Yes
Linear interpolation	XY only	XY only	Up to 4 axes
Circular interpolation	XY only	XY only	Any 2 axes plus 3rd tangent
Maximum number of segments in motion path	255	255	Infinite, continuous vector feed
Contouring	Yes	Yes	Yes
Electronic gearing	No	Yes	Yes
S-curve profiling	No	Yes	Yes
Programmable acceleration rate	Yes	Yes	Yes
Programmable deceleration rate	Yes	No	No

Specifications	DMC-600	DMC-6X1	DMC-1000
Maximum encoder frequency	.5 x 10^6 counts/s	2 x 10^6 counts/s	8 x 10^6 counts/s
DAC resolution	8-bits	12-bits	14-bits
Maximum move length	8 x 10^6	8 x 10^6	2 x 10^9
Sample time	1 msec	1 msec	0.5 msec (4 axes)
Program memory	500 lines, 32 chr	500 lines, 32 chr	500 lines, 40 chr
EEPROM memory for parameter storage	None	None	Yes
Number of variables	64 (V0-V63)	64 (V0-V63)	126; symbolic up to 8 chrs.
Number of array elements	None	1000 (1 array)	1600 (up to 14 arrays)
Digital filter type	GN,ZR,KI	GN,ZR,KI	KP,KI,KD with velocity and acceleration feedforward and integrator limit

Hardware	DMC-600	DMC-6X1	DMC-1000
Maximum # of axes/card	3	3	4 (8 for DMC-1080)
Analog inputs	8 with DMC-63010	8 with DMC-63010	7 standard
Digital inputs	8 TTL	8 TTL	8 optoisolated (24 for DMC-1080)
Digital outputs	8 TTL	8 TTL	8 TTL (16 for DMC-1080)
High speed position latch	None	Yes	Yes
Dual encoder inputs	none	Yes	Yes
Motor command output	+/- 10V	+/- 10V	+/- 10V and step/direction

DMC-600/DMC-1000 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
CR	Circular segment
CS	Clear motion sequence
DL	Download program
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
IN	Input prompt
IP	Increment position
JG	Jog mode
JP	Conditional jump
JS	Conditional jump subroutine
KI	Integrator gain
LS	List program
MG	Message
MO	Motor off
NO	No-op
OE	Automatic error shut-off
OF	Offset
OP	Write output port
PA	Position absolute
PR	Position relative
RE	Return from error subroutine
RI	Return from interrupt subroutine
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
UL	Upload program
VA	Vector acceleration
Vn=	Variable definition
VP	Vector position
VS	Vector speed
WT	Programmable timer
XG	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
CO	Configure I/O points (DB-10072 only)
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
LE	Linear interpolation end
LI	Linear interpolation distance
LM	Linear interpolation mode
MT	Motor type
OB	Output Bit
PF	Position format
RA	Record array
RC	Record
RD	Record data
RP	Report command position
TN	Tangent
TV	Tell velocity
VD	Vector deceleration
VE	Vector sequence end
VF	Variable format
VM	Coordinated motion mode
VT	Vector time constant - S-curve
WC	Wait for contour data

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	

DMC-600/DMC-1000 Pin-out Conversion Table

DMC-600	DMC-1000	Function
Pin #	Pin #	
1	1	Ground
2	2	+5V
3	3	Error
4	4	Reset
5	29	Motor command Z
6	27	Motor command Y
7	25	Motor command X
8	19 (INCOM)	Ground
9	49	Index Z
10	50	Index - Z
11	47	CH B Z
12	48	CH B- Z
13	45	CH A Z
14	46	CH A- Z
15	43	Index Y
16	44	Index - Y
17	41	CH B Y
18	42	CH B- Y
19	39	CH A Y
20	40	CH A- Y
21	12	Forward limit Z

22	13	Reverse limit Z
23	14	Home Z
24	9	Forward limit Y
25	10	Reverse limit Y
26	11	Home Y
27	24	Abort
28	9 (J5) isolated 5V	+5V
29	37	Index X
30	38	Index - X
31	35	CH B X
32	36	CH B- X
33	33	CH A X
34	34	CH A- X
35	6	Forward limit X
36	7	Reverse limit X
37	8	Home X
38	19 (input common)	Ground
39		N.C.
40	18 (J5)	Input 8
41	19 (J5)	Input 7
42	20 (J5)	Input 6
43	21 (J5)	Input 5
44	22 (J5)	Input 4
45	23 (J5)	Input 3
46	24 (J5)	Input 2
47	25 (J5)	Input 1
48	8 (J5)	Ground
49	17 (J5)	Output 7
50	16 (J5)	Output 6
51	15 (J5)	Output 5
52	14 (J5)	Output 4
53	13 (J5)	Output 3
54	12 (J5)	Output 2
55	11 (J5)	Output 1
56	18 or 10 (J5)	Output 0
57	57	+12 V
58	58	- 12 V
59	59	+ 5 V
60	60	Ground

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

Galil Motion Control
575 Maude Court
Sunnyvale, CA 94086
Phone: 408-746-2300
Fax: 408-746-2315
BBS: 408-746-3860 (8-N-1) up to 14,400 baud.
Internet address: galil@netcom.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.

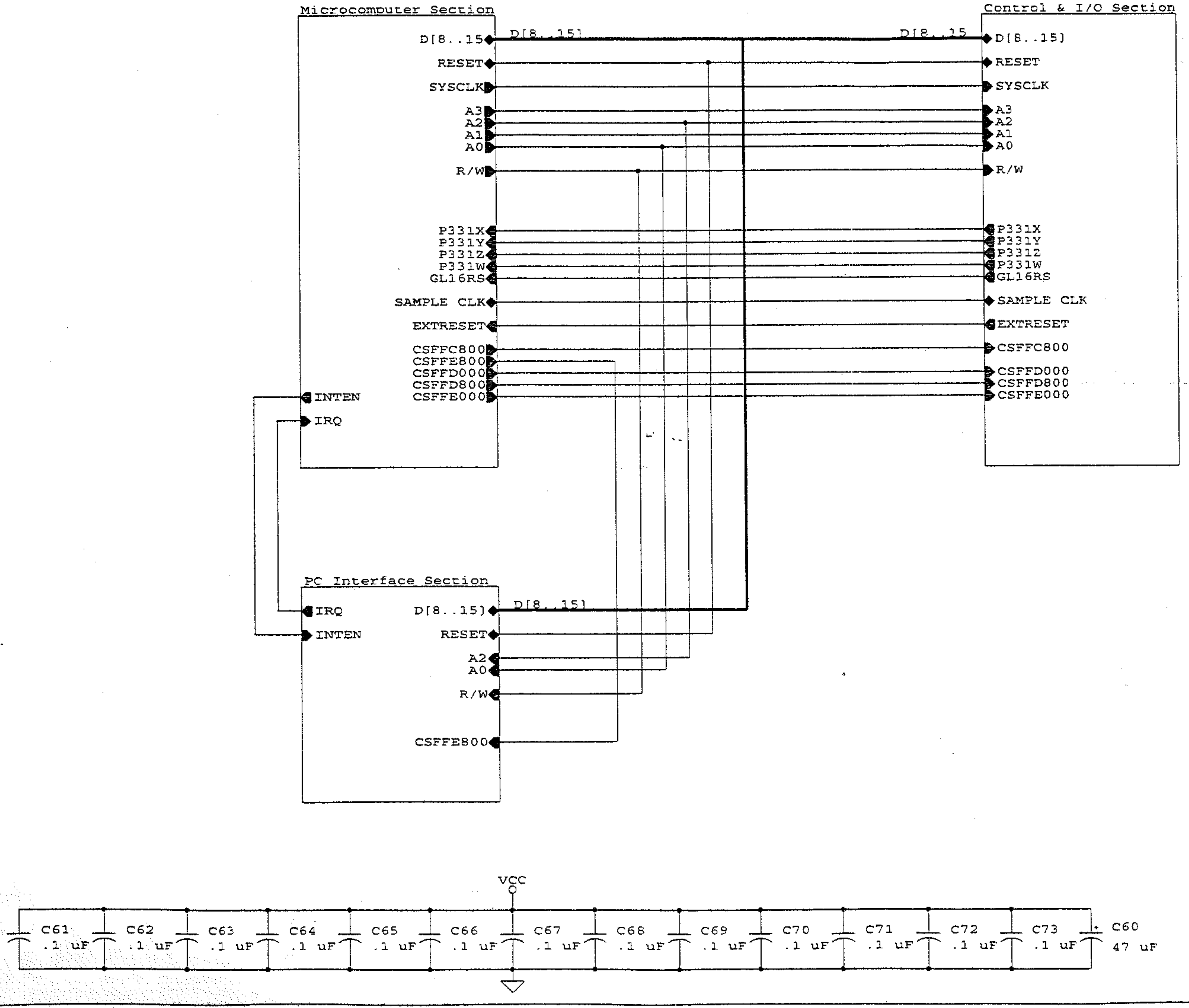
Any defect in materials or workmanship determined by Galil Motion Control to be attributable to customer alteration, modification, negligence or misuse is not covered by this warranty.

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.

COPYRIGHT (10-94)

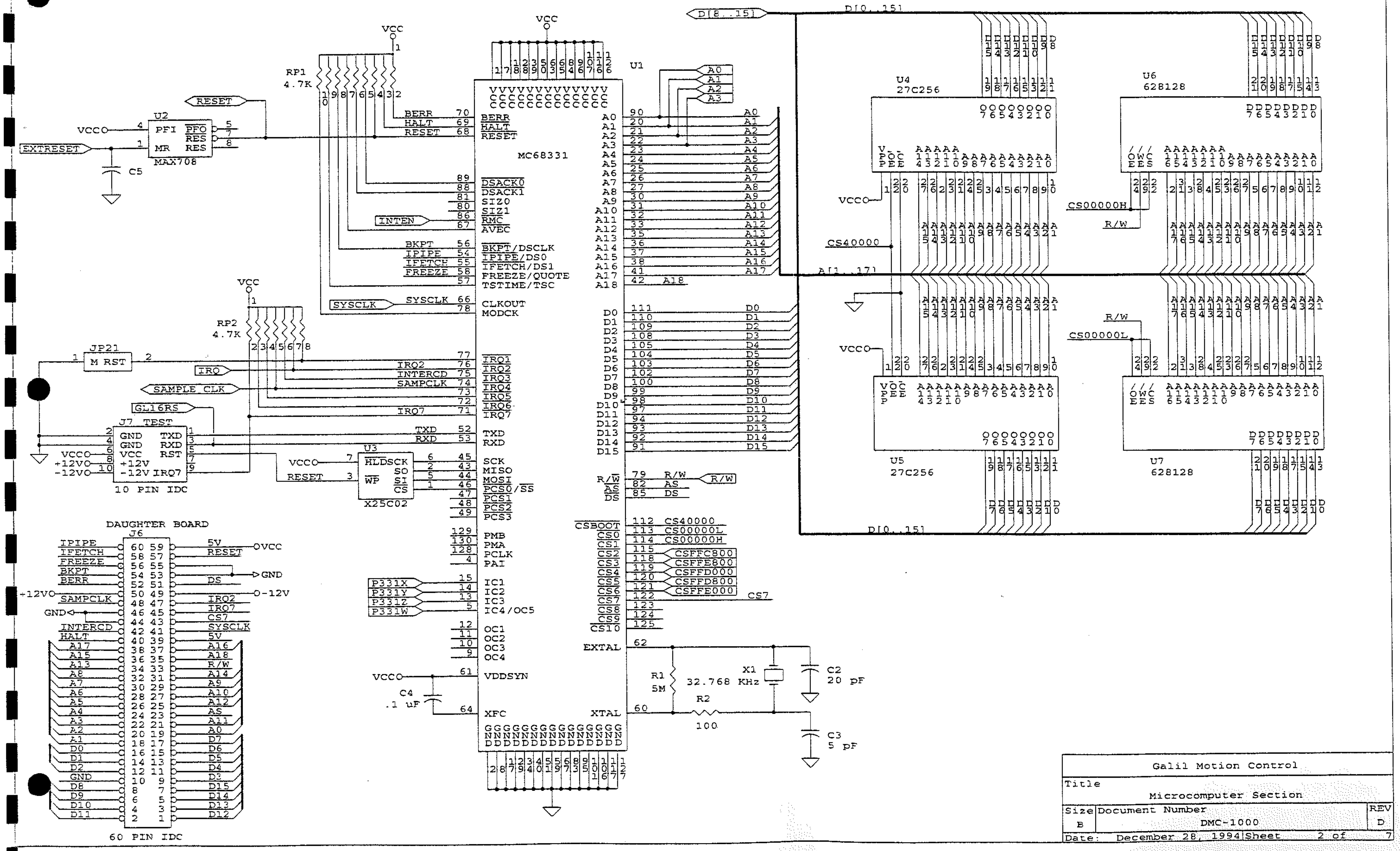
The software code contained in this Galil product is protected by copyright and must not be reproduced or disassembled in any form without prior written consent of Galil Motion Control, Inc.

Index



GALIL MOTION CONTROL

Title	DMC-1000	
Size B	Document Number DMC-1000	REV D
Date:	May 1, 1995	Sheet 1 of 7

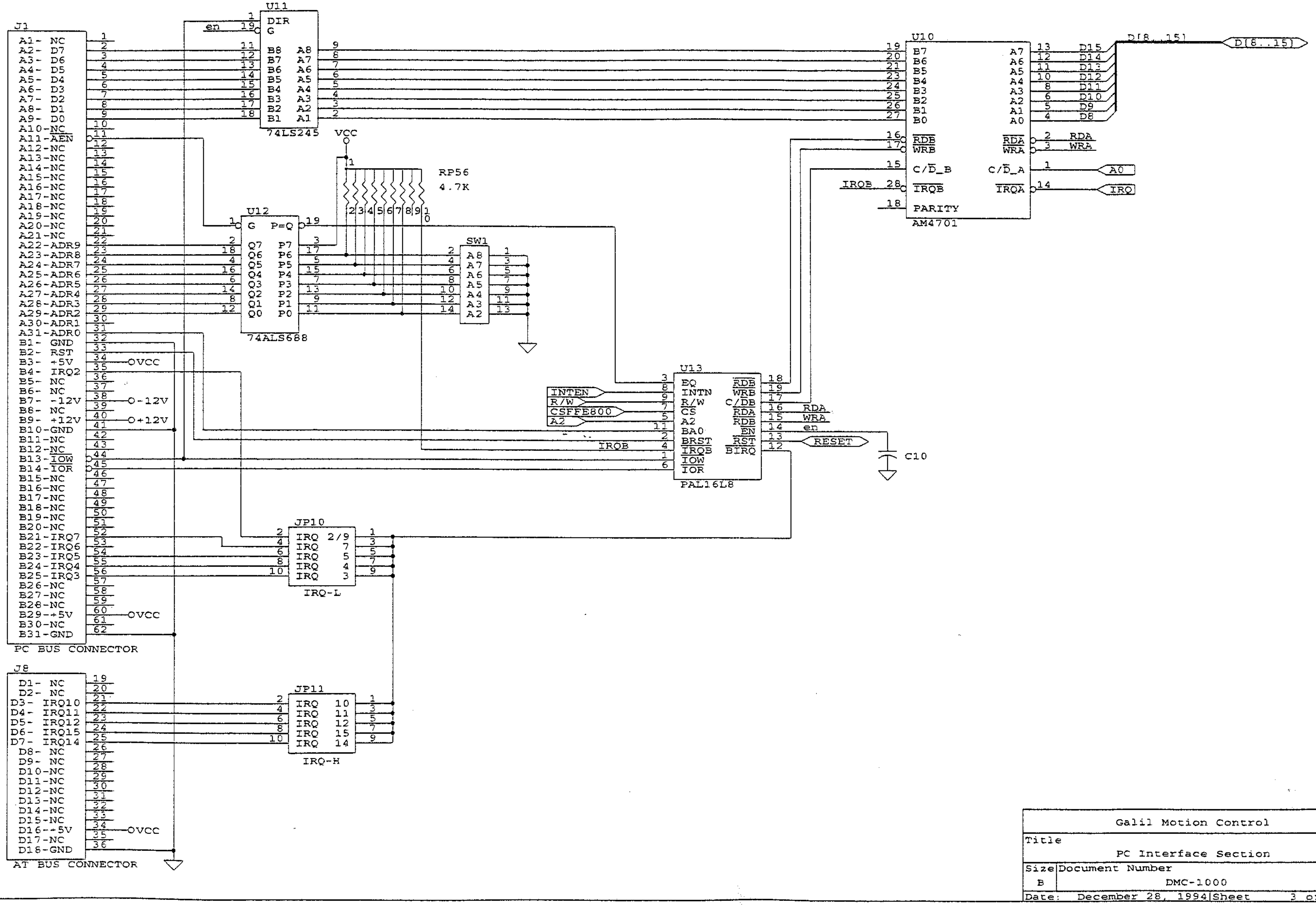


Galil Motion Control

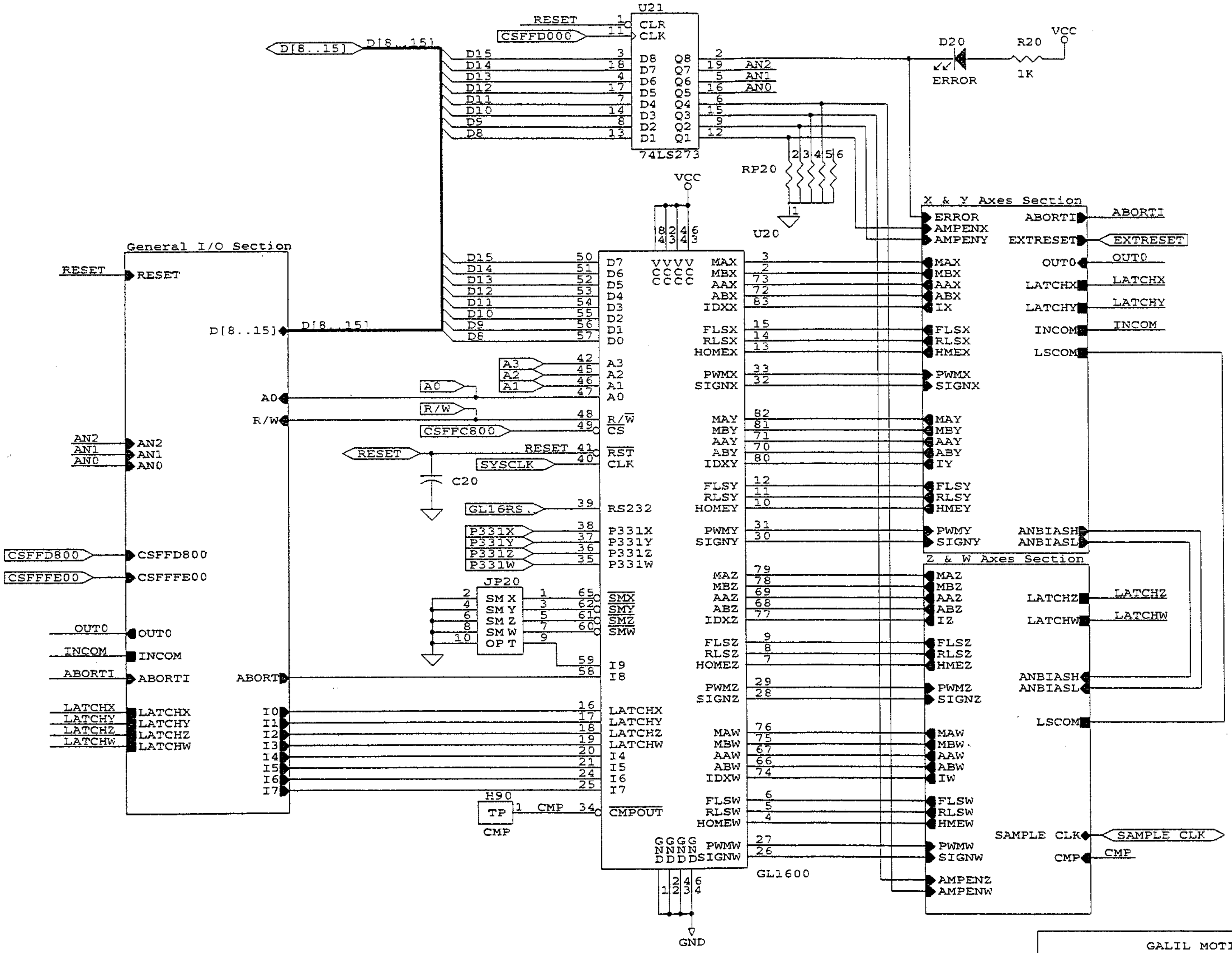
Title Microcomputer Section

Size B Document Number DMC-1000 REV D

Date: December 28, 1994 Sheet 2 of 7



Galil Motion Control			
Title	PC Interface Section		
Size	Document Number		REV
B	DMC-1000		D
Date: December 28, 1994	Sheet	3 of	7

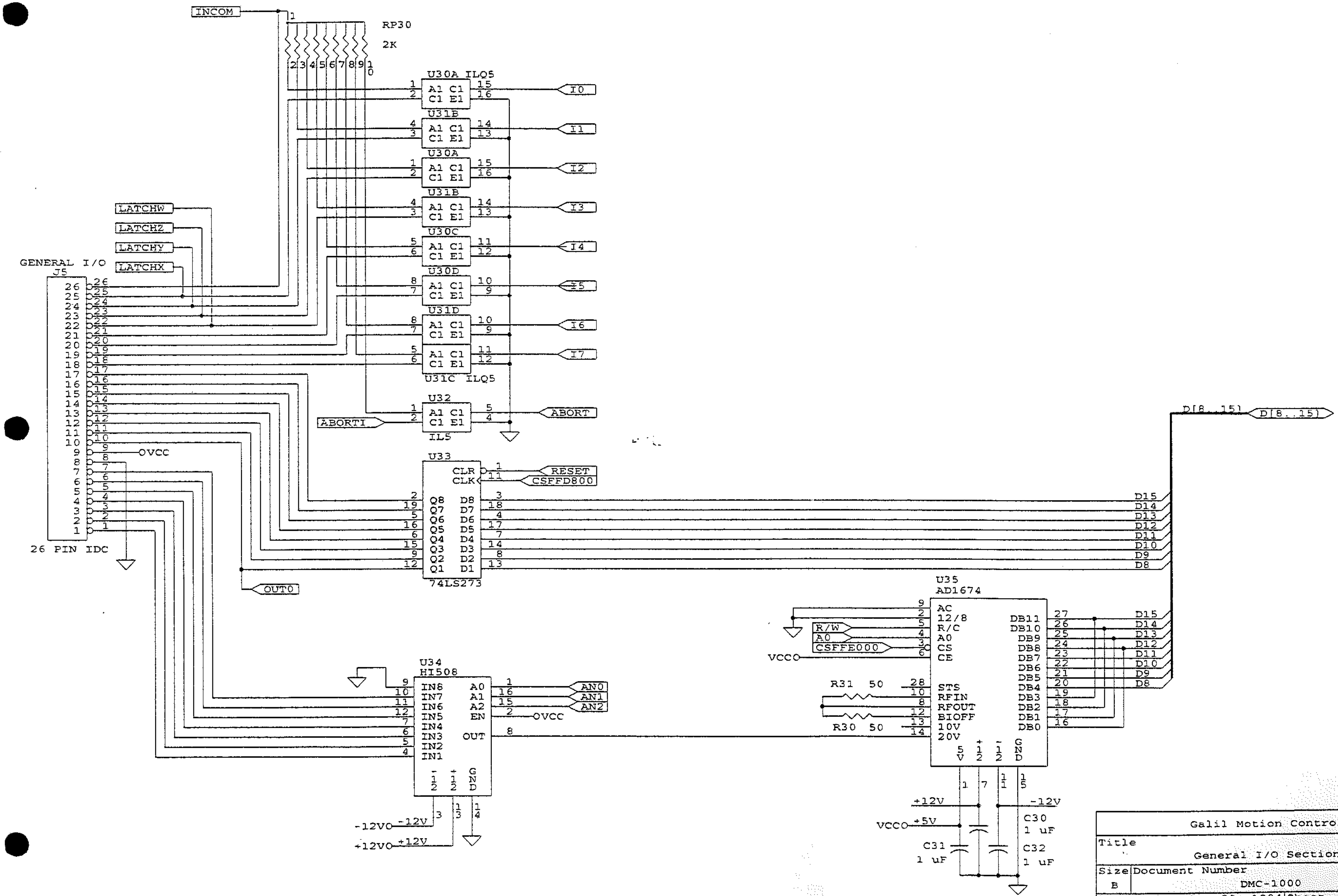


GALIL MOTION CONTROL

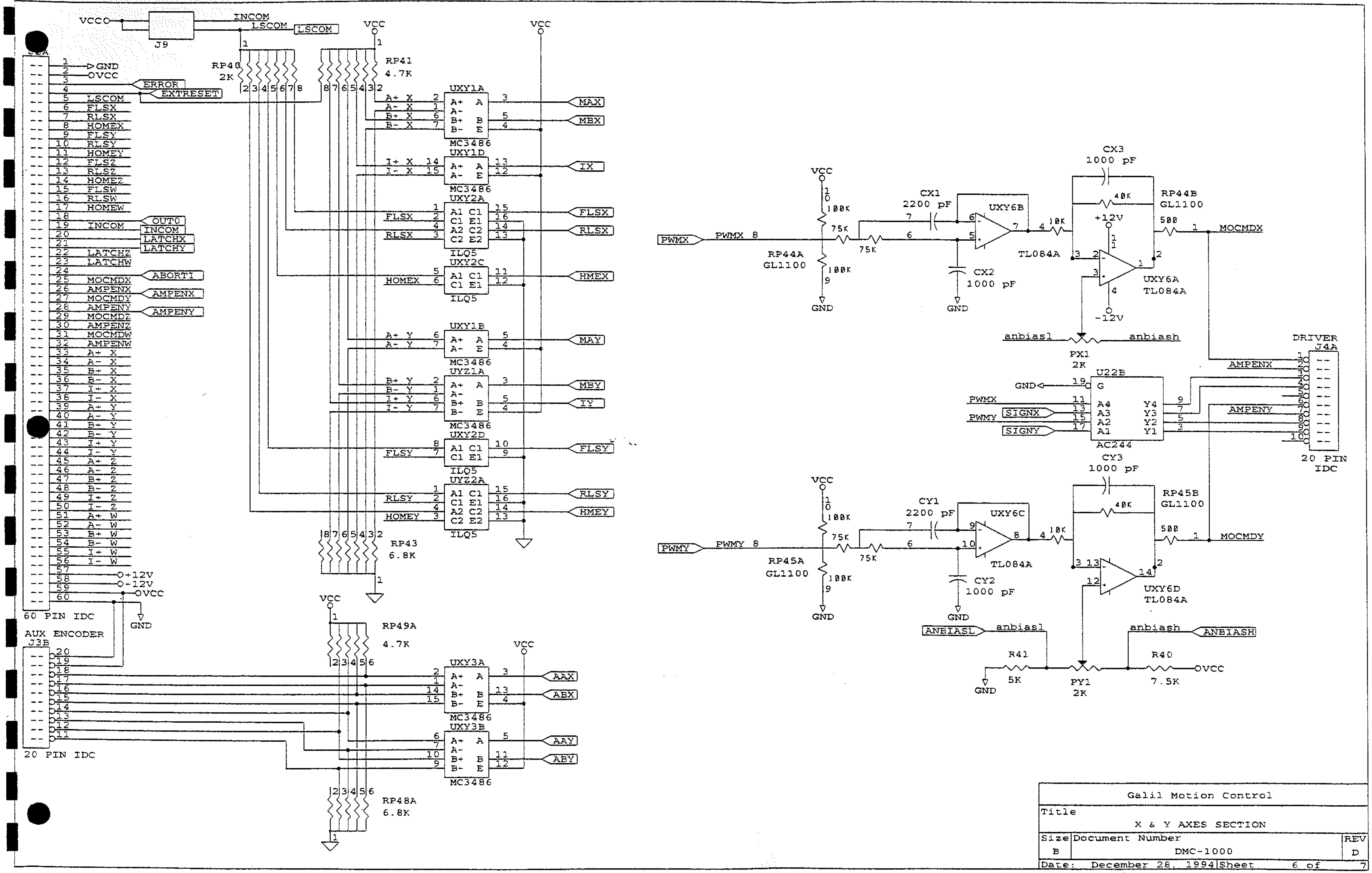
Title
I/O

Size	Document Number	REV
B	DMC-1000	D

Date: May 1, 1995 Sheet 4 of 7



Galil Motion Control		
Title	General I/O Section	
Size B	Document Number DMC-1000	REV D
Date: December 28, 1994	Sheet 5 of 7	

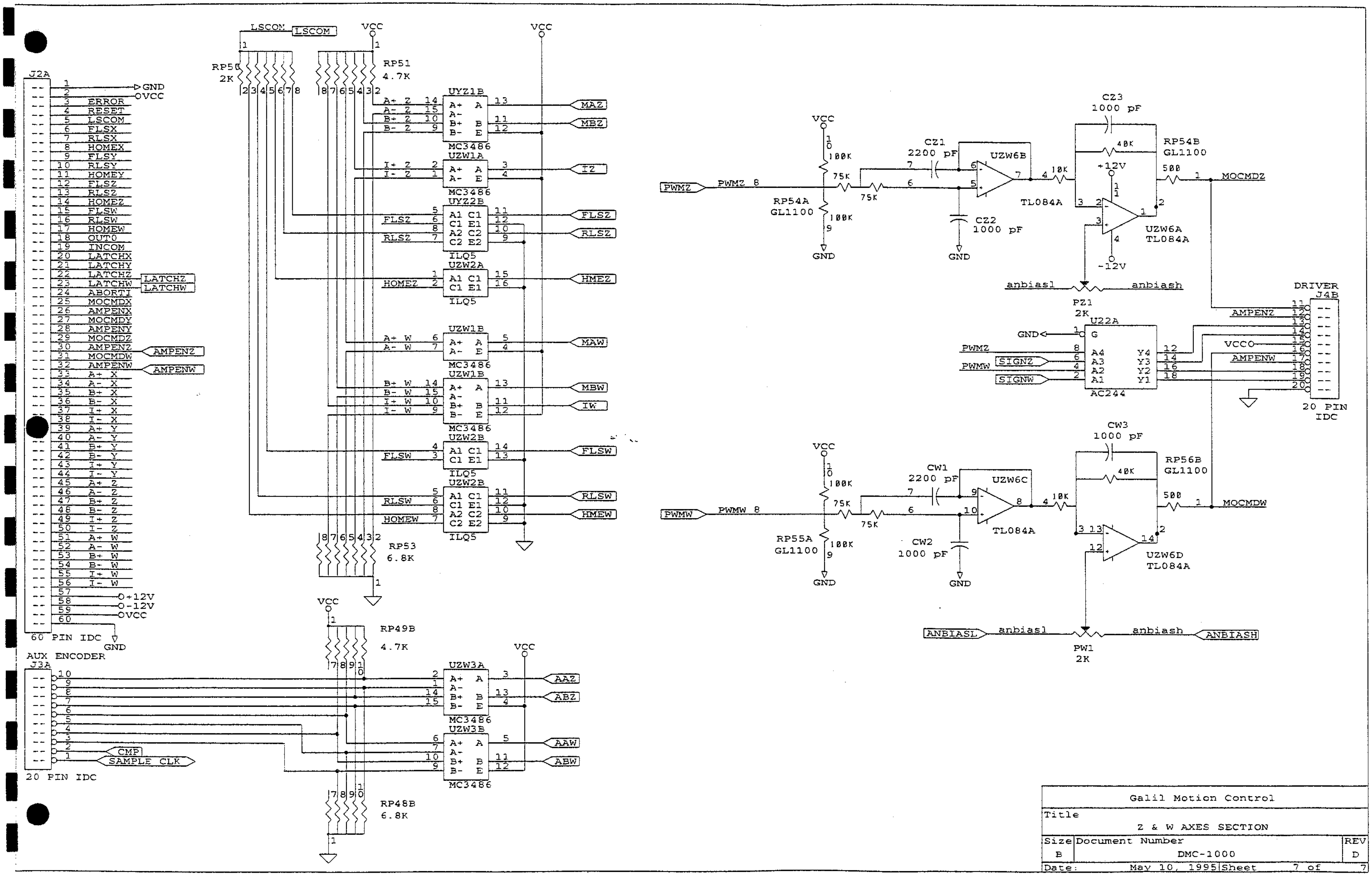


Galil Motion Control

Title
X & Y AXES SECTION

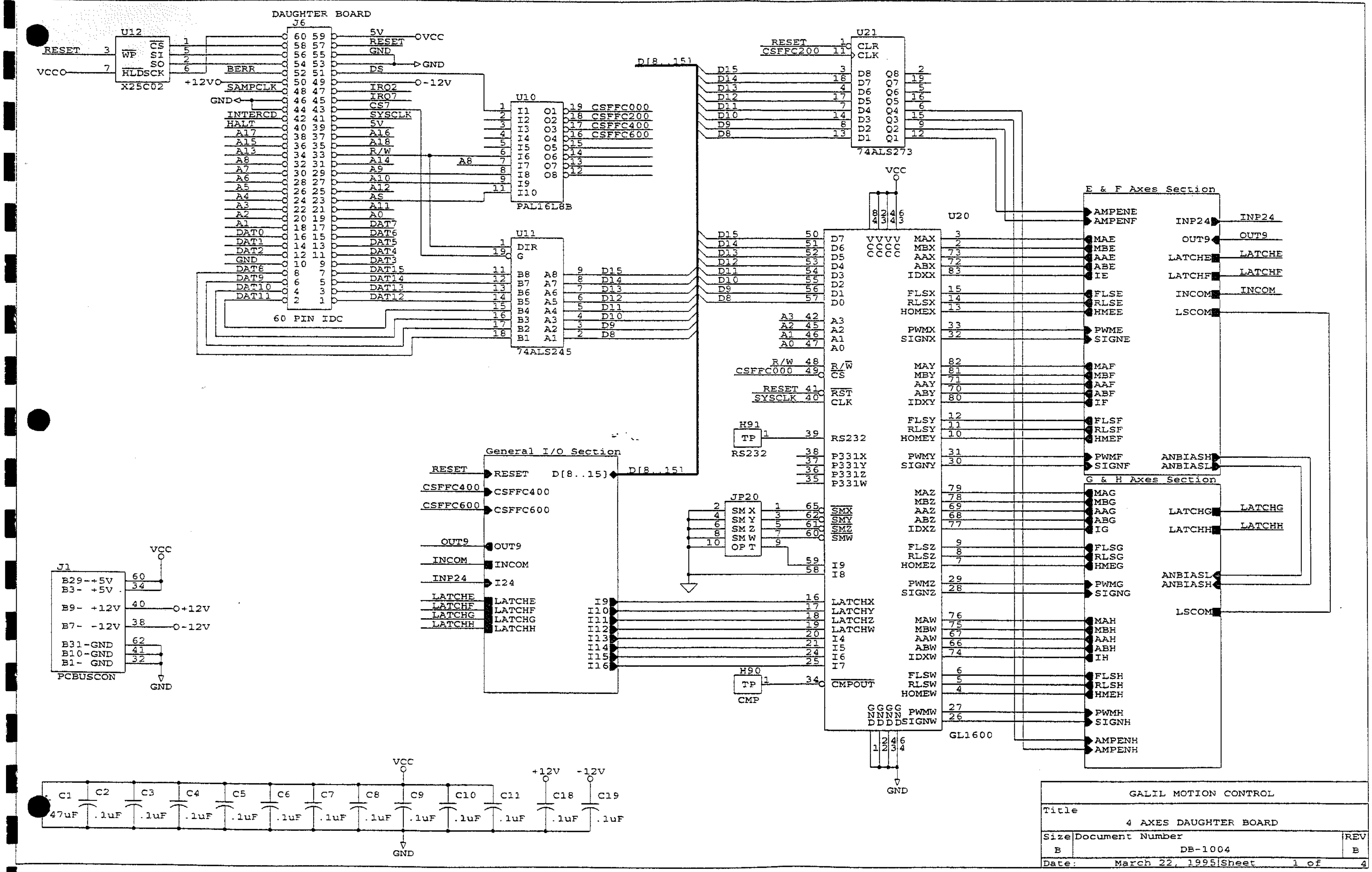
Size	Document Number	REV
B	DMC-1000	D

Date: December 28, 1994 Sheet 6 of 7



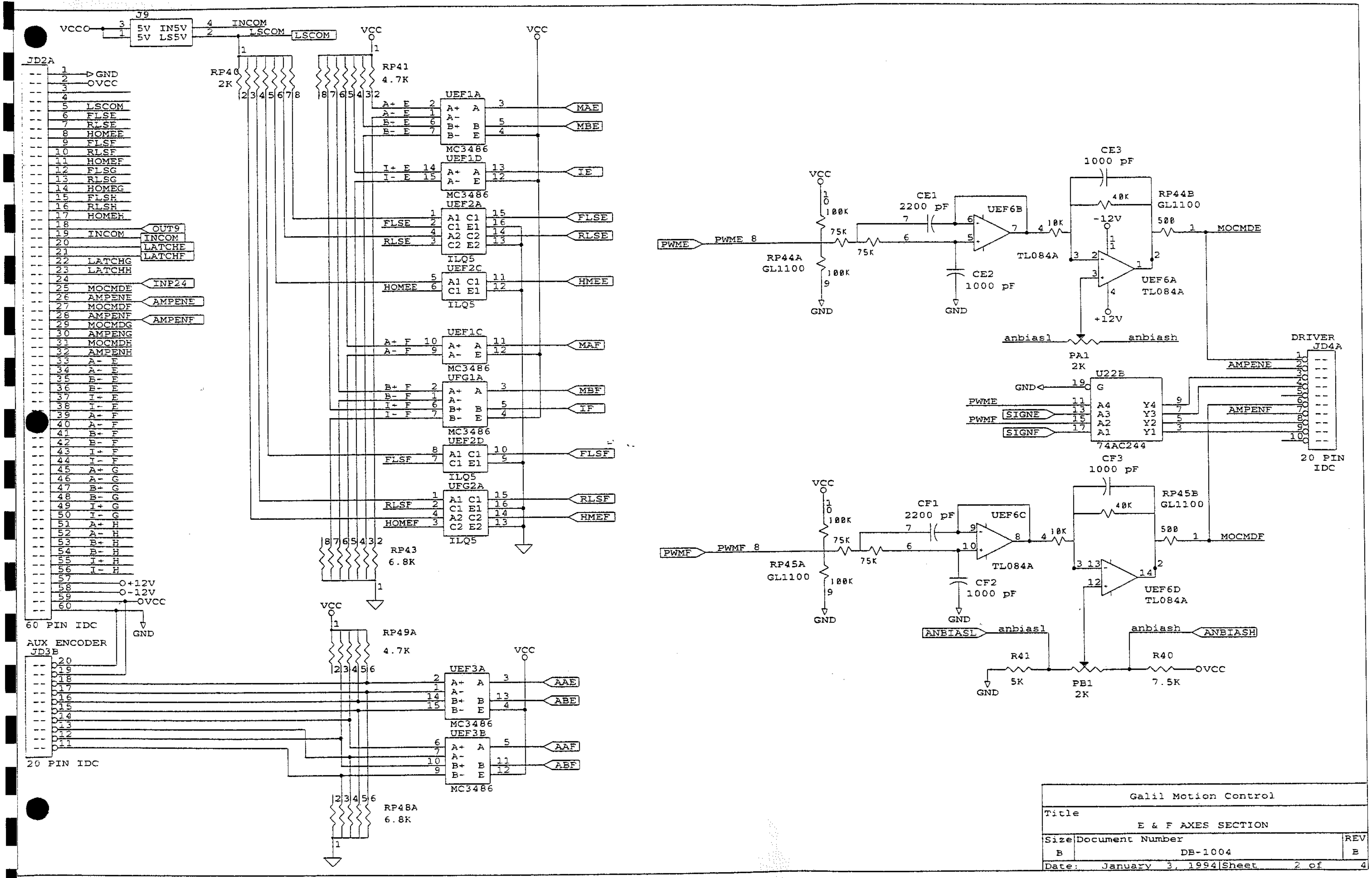
Galil Motion Control

Title	Z & W AXES SECTION	
Size B	Document Number DMC-1000	REV D
Date: May 10, 1995	Sheet 7 of 7	

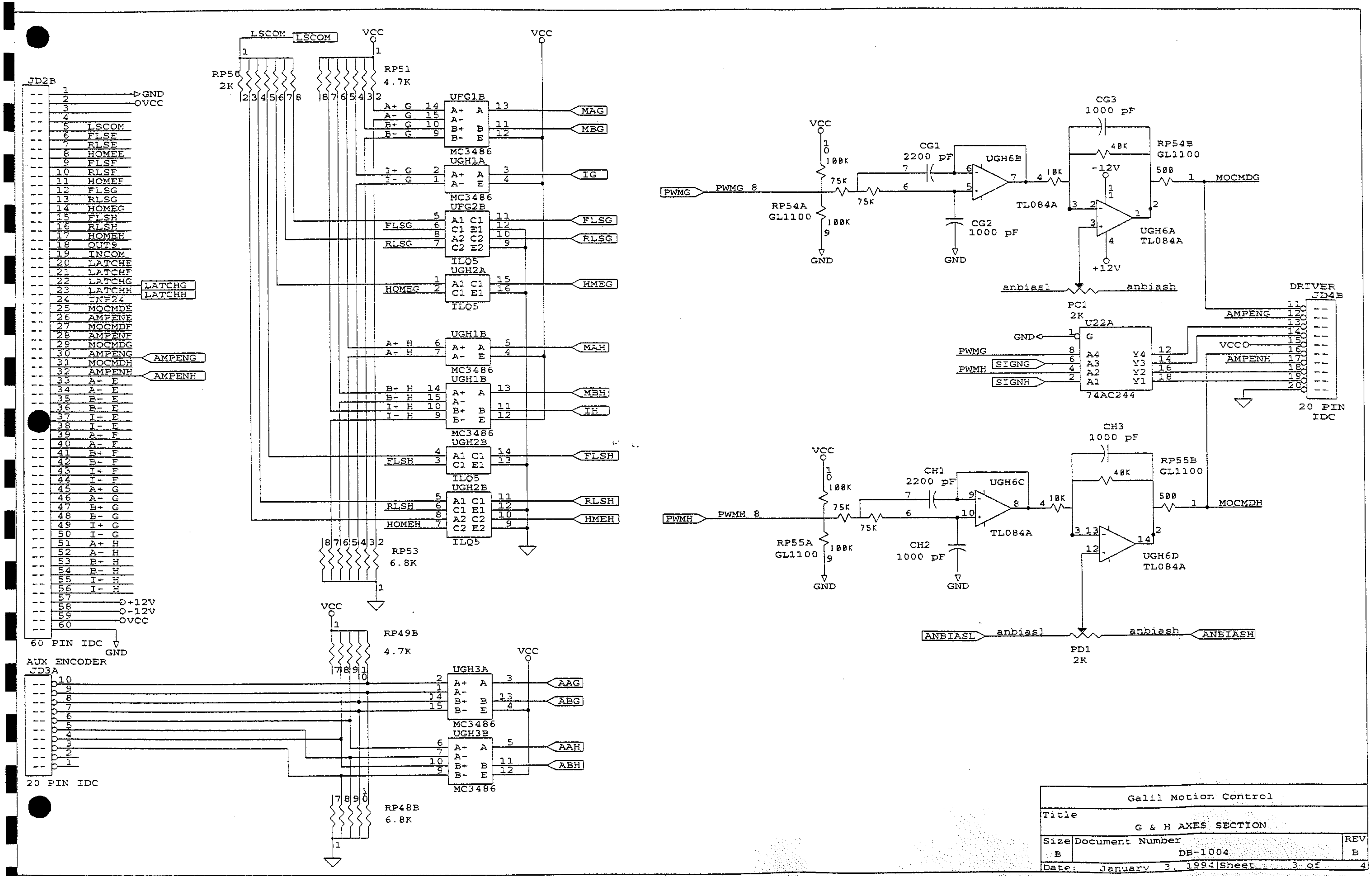


GALIL MOTION CONTROL

Title		
	4 AXES DAUGHTER BOARD	
Size	Document Number	REV
B	DB-1004	B
Date:	March 22, 1995	Sheet 1 of 4



Galil Motion Control		
Title E & F AXES SECTION		
Size B	Document Number DB-1004	REV B
Date: January 3, 1994	Sheet 2 of 4	

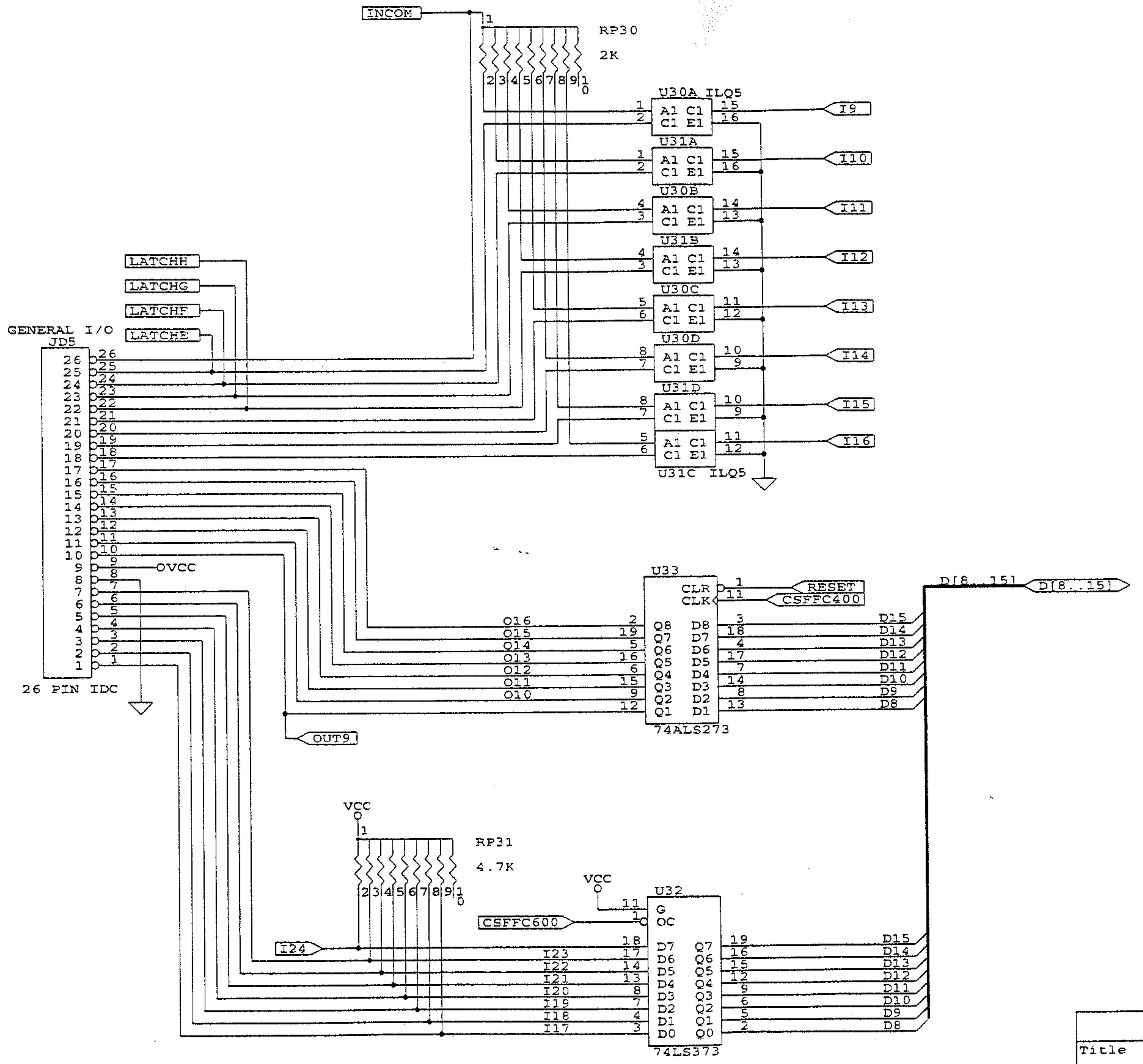


Galil Motion Control

Title G & H AXES SECTION

Size	Document Number	REV
B	DB-1004	B

Date: January 3, 1994 Sheet 3 of 4

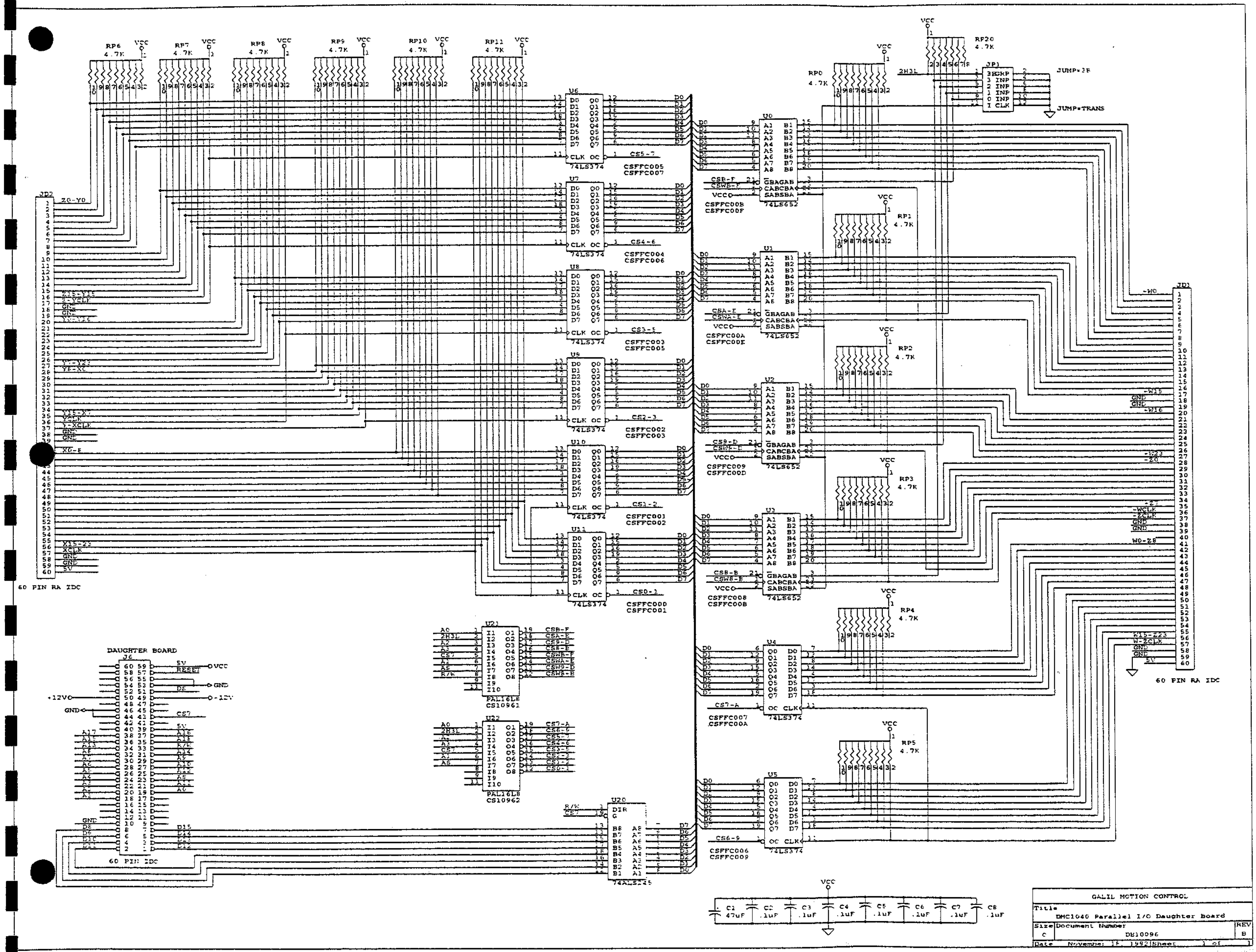


Galil Motion Control

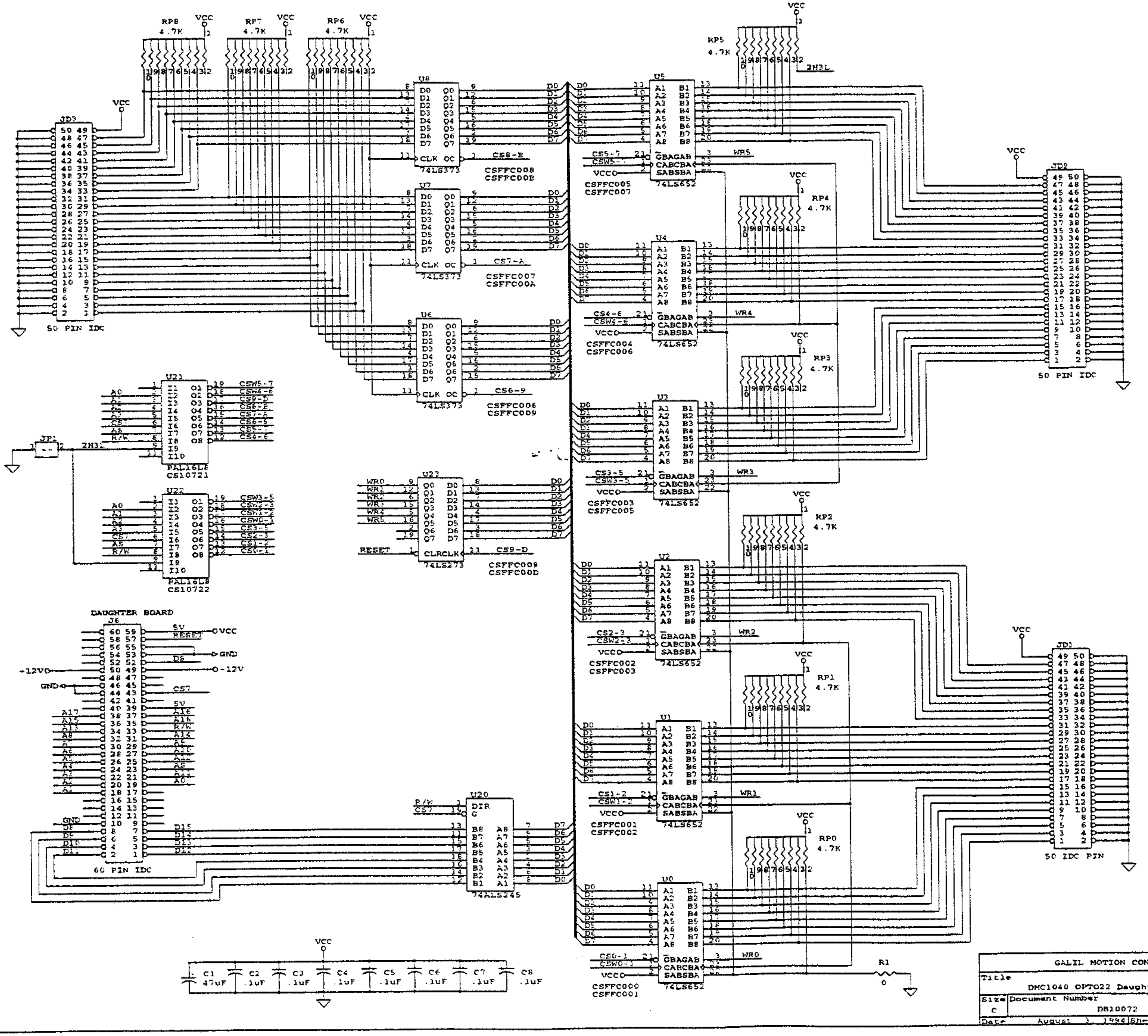
Title
GENERAL I/O SECTION

Size	Document Number	REV
B	DB-1004	B

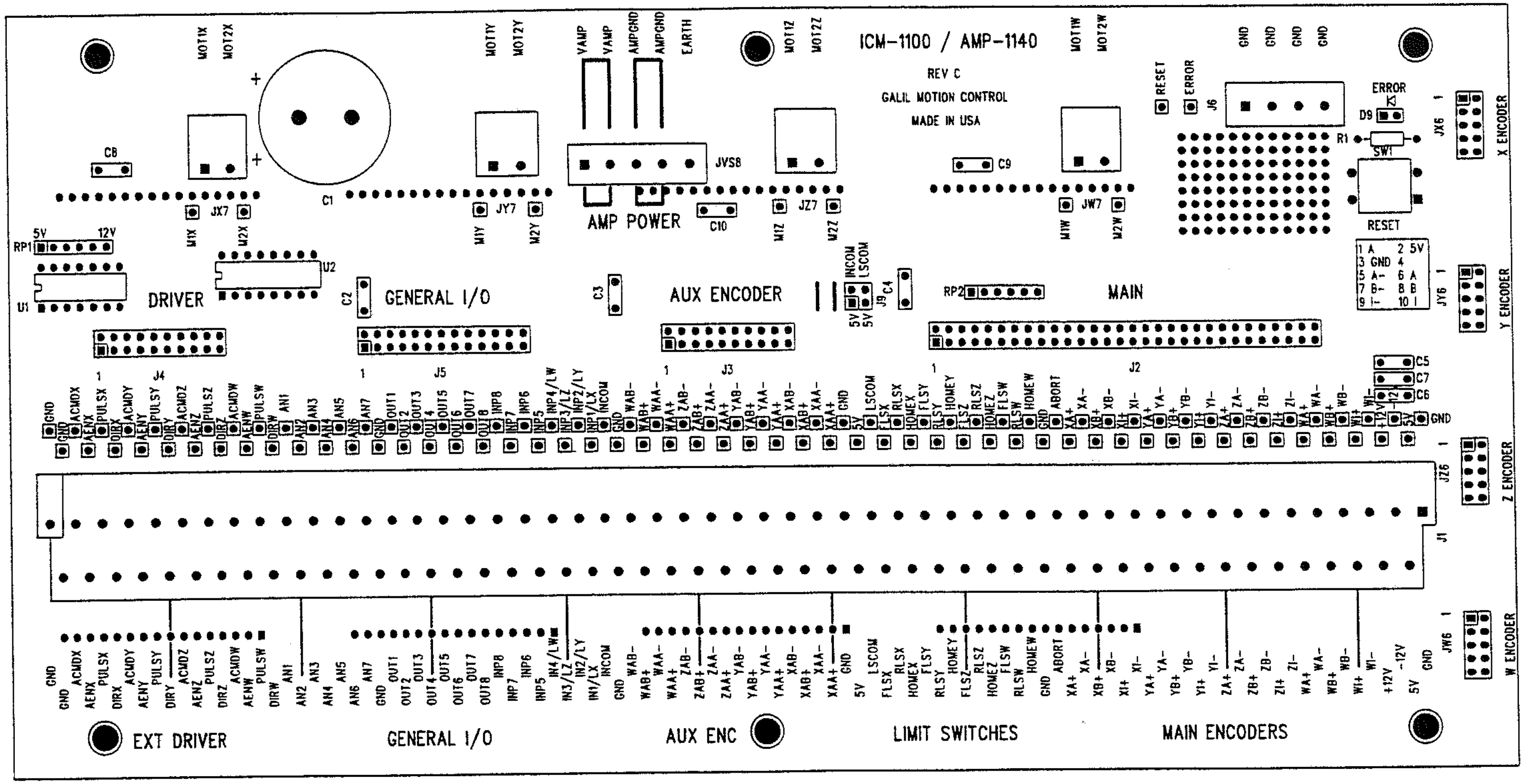
Date: January 3, 1994 Sheet 4 of 4


RP6 4.7K
RP7 4.7K
RP8 4.7K
RP9 4.7K
RP10 4.7K
RP11 4.7K
RP0 4.7K
RP1 4.7K
RP2 4.7K
RP3 4.7K
RP4 4.7K
RP5 4.7K
RP20 4.7K
VCC
U6 74LS374 CS5-7 CSFFC005 CSFFC007
U7 74LS374 CS4-6 CSFFC004 CSFFC006
U8 74LS374 CS3-5 CSFFC003 CSFFC005
U9 74LS374 CS2-3 CSFFC002 CSFFC003
U10 74LS374 CS1-2 CSFFC001 CSFFC002
U11 74LS374 CS0-1 CSFFC000 CSFFC001
U0 74LS652 CSB-F CSWB-F CSFFC00B CSFFC00F
U1 74LS652 CSA-E CSWA-E CSFFC00A CSFFC00E
U2 74LS652 CS9-D CSW9-D CSFFC009 CSFFC00D
U3 74LS652 CS8-B CSW8-B CSFFC008 CSFFC00B
U4 74LS374 CS7-A CSFFC007 CSFFC00A
U5 74LS374 CS6-9 CSFFC006 CSFFC009
JP1 JUMP=3F JUMP=TRANS
JD2 60 PIN RA IDC
JD1 60 PIN RA IDC
DAUGHTER BOARD J6 60 PIN IDC
+12V GND -12V RESET
U21 PAL16L8 CS10961
U22 PAL16L8 CS10962
U20 74ALS245 DIR G
C1 47uF C2 .1uF C3 .1uF C4 .1uF C5 .1uF C6 .1uF C7 .1uF C8 .1uF

GALIL MOTION CONTROL		
Title DMC1040 Parallel I/O Daughter Board		
Size C	Document Number DB10096	REV B
Date November 15, 1992	Sheet 1 of 1	



GALIL MOTION CONTROL		
Title DMC1040 OPTO22 Daughter board		
Size C	Document Number DB10072	REV A
Date August 3, 1994	Sheet 1 of 1	



SILK SCREEN

ICM-1100 REV C