USER MANUAL

DMC-1412/1414

Manual Rev. 2.7

By Galil Motion Control, Inc.

Galil Motion Control, Inc.
270 Technology Way
Rocklin, California 95765
Phone: (916) 626-0101
Fax: (916) 626-0102
Internet Address: support@galilmc.com
URL: www.galilmc.com

Rev Date: 3/08

Using This Manual

Your DMC-1412/1414 motion controller has been designed to work with both servo and stepper type motors. Installation and system setup will vary depending upon whether the controller will be used with stepper motors or servo motors. To make finding the appropriate instructions faster and easier, icons will be next to any information that applies exclusively to one type of system. Otherwise, assume that the instructions apply to all types of systems. The icon legend is shown below.

Attention: Pertains to servo motor use.

Attention: Pertains to stepper motor use.

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 machinery. Galil shall not be liable or responsible for any incidental or consequential damages.

Contents

THIS PAGE LEFT BLANK INTENTIONALLY

Chapter 1 Overview

Introduction

The DMC-1400 series of motion controllers was developed specifically for one axis applications allowing it to be smaller in size (1/2 size card) and lower in cost than multiaxis controllers. This manual covers the two serial based stand-alone controllers in the DMC-1400 Econo series lineup. The DMC-1412 motion controller communicates via the RS-232 serial connection and the DMC-1414 is the equivalent controller integrated with an internal power amplifier. Performance capability of these controllers includes: 8 MHz encoder input frequency, 16-bit motor command output DAC, ±2 billion counts total travel per move, up to 250 μsec sample rate, and non-volatile memory for parameter storage. Designed for maximum system flexibility, the DMC-141X can be interfaced to a variety of motors and drives including step motors, servo motors and hydraulics.

The controller accepts feedback from a quadrature linear or rotary encoder with input frequencies up to 8 million quadrature counts per second. An additional encoder input is available for gearing or cam applications, handwheel inputs, or dual-loop. Modes of motion include jogging, point-to-point positioning, electronic cam, electronic gearing and contouring. Several motion parameters can be specified including acceleration and deceleration rates and slew speed. The DMC-141X also provides motion smoothing to eliminate jerk.

For synchronizing motion with external events, the DMC-141X includes seven digital inputs and three digital outputs. Event triggers can automatically check for elapsed time, distance and motion complete.

The DMC-141X 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 WSDK allows quick system set-up and tuning.

To prevent system damage during machine operation, the DMC-141X provides many error handling features. These include software and hardware limits, automatic shut-off on excessive error, abort input, and user-definable error and limit routines.

The DMC-1412 is designed for stand-alone applications and provides non-volatile storage for programs, variables and array elements. The DMC-1414 provides an internal power amplifier and interconnection terminals for a standard brush DC servo motor.

Overview of Motor Types

The DMC-141X can provide the following types of motor control:

1. Standard servo motors with ±10 volt command signals
2. Brushless servo motors with sinusoidal commutation (DMC-1412 only)
3. Step motors with step and direction signals
4. Other actuators such as hydraulics - For more information, contact Galil.

Standard Servo Motors with ±10 Volt Command Signal

The DMC-141X achieves superior precision through the use of a 16-bit motor command output DAC and a sophisticated PID filter that features velocity and acceleration feedforward, and integration and torque limits.

The controller is configured by the factory for standard servo motor operation. In this configuration, the controller provides an analog signal (±10 volt) to connect to a servo amplifier. This connection is described in Chapter 2. In the case of the DMC-1414, a brush servo amplifier is connected to the analog signal internally.

Brushless Servo Motor with Sinusoidal Commutation

The DMC-1412 can provide sinusoidal commutation for brushless motors (BLM). In this configuration, the controller generates two sinusoidal signals for connections with amplifiers specifically designed for this purpose.

Note: The task of generating sinusoidal commutation may be accomplished in the brushless motor amplifier. If the amplifier generates the sinusoidal commutation signals, only a single command signal is required and the controller should be configured for a standard servo motor (described above).

Sinusoidal commutation in the controller can be used with linear and rotary BLMs. However, the motor velocity should be limited such that a magnetic cycle lasts at least 6 milliseconds*. For faster motors, please contact the factory.

To simplify the wiring, the controller provides a one-time, automatic set-up procedure. The parameters determined by this procedure can then be saved in non-volatile memory to be used whenever the system is powered on.

The DMC-1412 can control BLMs equipped with Hall sensors as well as without Hall sensors. If hall sensors are available, once the controller has been set up, the controller will automatically estimate the commutation phase upon reset. This allows the motor to function immediately upon power up. The hall effect sensors also provides a method for setting the precise commutation phase. Chapter 2 describes the proper connection and procedure for using sinusoidal commutation of brushless motors.

* 6 milliseconds per magnetic cycle assumes a servo update of 1 msec (default rate).

Stepper Motor with Step and Direction Signals

The DMC-141X can control stepper motors. In this mode, the controller provides two signals to connect to the stepper motor: Step and Direction. For stepper motor operation, the controller does not require an encoder and operates the stepper motor in an open loop fashion. Chapter 2 describes the proper connection and procedure for using stepper motors.

DMC-1400 Functional Elements

The DMC-141X circuitry can be divided into the following functional groups as shown in Figure 1.1 and discussed below.

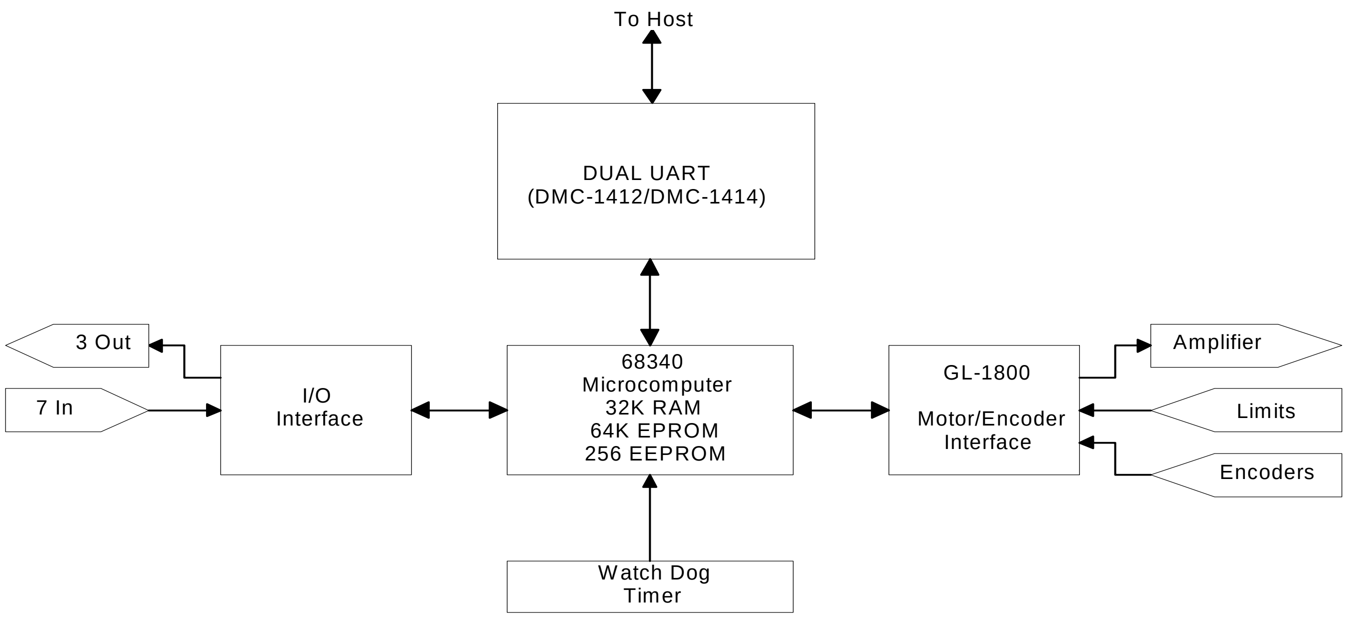


Figure 1.1 - DMC-141X Functional Elements

Microcomputer Section

The main processing unit of the DMC-141X is a specialized 32-bit Motorola 68340 Series Microcomputer with 32K RAM (256K available as an option), 64K EPROM and 128K bytes EEPROM. The RAM provides memory for variables, array elements and application programs. The EPROM stores the firmware of the DMC-141X. The DMC-1412 and DMC-1414 provide 128K EEPROM for storing programs, arrays and variables in addition to parameters upon power down.

Motor Interface

The GL-1800 custom, sub-micron gate array performs quadrature decoding of the encoders at up to 8 MHz, generates a ±10 volt analog signal (16 Bit D-to-A) for input to a servo amplifier, and generates step and direction signal for step motor drivers. For the DMC-1414, this analog command signal feeds directly into the power amplifier, which outputs directly to a brush DC servo motor.

Communication

The DMC-1412 and DMC-1414 provide a main and auxiliary RS232 port for communication. Communication speeds up to 38.4 kbaud are available.

General I/O

The DMC-141X provides interface circuitry for seven TTL inputs and three TTL outputs.

System Elements

As shown in Fig. 1.2, the DMC-141X is part of a motion control system which includes amplifiers, motors and encoders. These elements are described below.

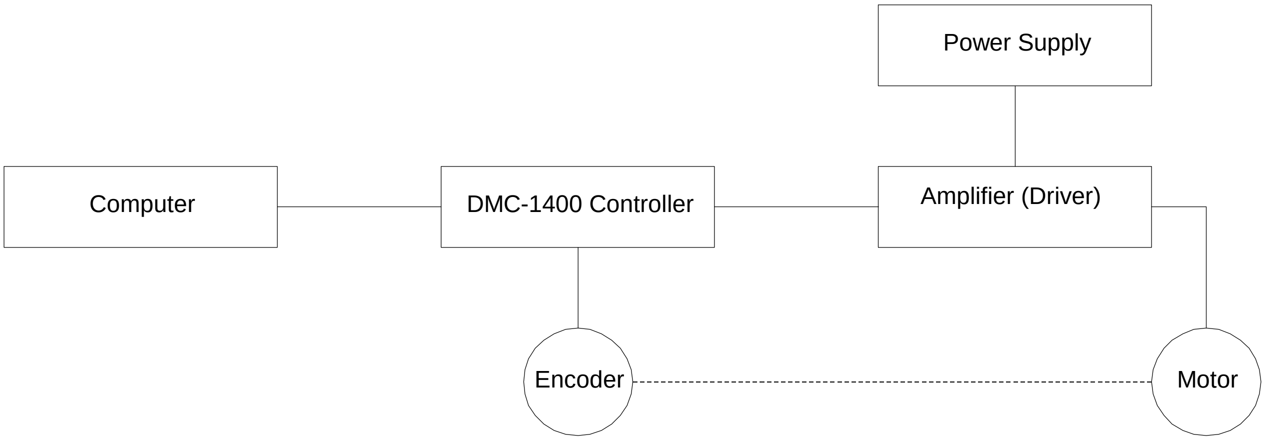


Figure 1.2 - Elements of Servo systems

Motor

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

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

Amplifier (Driver)

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

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

For the DMC-1414, the power amplifier is internal to the unit. This PWM power amplifier requires a single external DC power supply from 20 to 60 volts. The amplifier provides 6 amps continuous and 12 amps peak.

Encoder

An encoder translates motion into electrical pulses which are fed back into the controller. The DMC-141X accepts feedback from either a rotary or linear encoder. Typical encoders provide two channels in quadrature, known as CHA and CHB. This type of encoder is known as a quadrature encoder. Quadrature encoders may be either single-ended (CHA and CHB) or differential (CHA, CHA-, CHB, CHB-). The DMC-141X decodes either type into quadrature states or four times the number of cycles. Encoders may also have a third channel (or index) for synchronization.

The DMC-141X can also interface to encoders with pulse and direction signals.

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

The standard voltage level is TTL (zero to five volts), however, voltage levels up to 12 volts are acceptable. If using differential signals, 12 volts can be input directly to the DMC-141X. Single-ended 12 volt signals require a bias voltage input to the complementary inputs.

Watch Dog Timer

The DMC-141X provides an internal watch dog timer which checks for proper microprocessor operation. The timer toggles the Amplifier Enable Output (AEN) which can be used to switch the amplifiers off in the event of a serious DMC-141X failure. The AEN output is normally high. During power-up and if the microprocessor ceases to function properly, the AEN output will go low. The error light will also turn on at this stage. A reset is required to restore the DMC-141X to normal operation. Consult the factory for a Return Materials Authorization (RMA) Number if your DMC-141X is damaged.

THIS PAGE LEFT BLANK INTENTIONALLY

Chapter 2 Getting Started

The DMC-141X Motion Controller

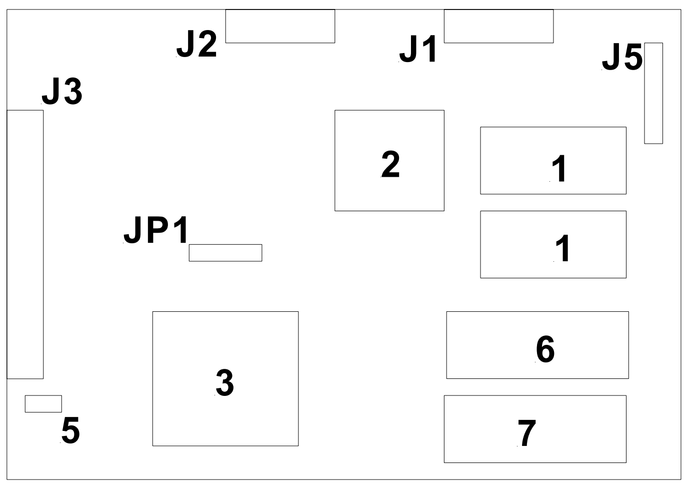


Figure 2.1 - Outline of the DMC-1412

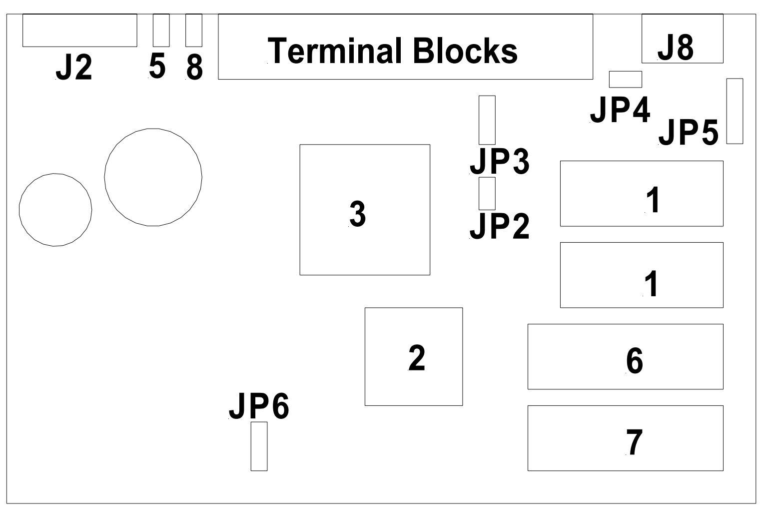


Figure 2.2 - Outline of the DMC-1414

1	**DMC-141X Firmware ROM. Labeled with firmware revision number, i.e. DMC-141X Rev 2.0a**	**J3**	**37 Pin D connection for controller signal break-out (DMC-1412)**
2	**Motorola 68340 microprocessor**	**J5**	**7-Pin power connector for +5, +12 and -12 volt input to board (DMC-1412)**
3	**GL-1800 custom gate array**	**J8**	**Main/Auxiliary 9-Pin serial port (DMC-1414)**
5	**Error LED**	**JP1**	**Master Reset, Stepper Motor and Baud rate selection jumpers (DMC-1412)**
6	**Controller RAM**	**JP2**	**Jumper used for configuring stepper motor operation, labeled as SMX (DMC-1414)**
7	**EEPROM for program/parameter storage**	**JP3**	**Jumpers for selecting Main Serial port as RS232 or RS422 (DMC-1414)**
8	**Controller Reset Switch**	**JP4**	**Jumper for selecting RS485 serial communication (DMC-1414)**
J1	**Main 9-pin Serial Port (DMC-1412)**	**JP5**	**Jumpers for selecting Auxiliary Serial port as RS232 or RS422 (DMC-1414)**
J2	**Auxiliary 9-pin Serial Port (DMC-1412)** **5-Pin power connector for 20-60V DC supply and Motor +/- connections (DMC-1414)**	**JP6**	**Master Reset and Baud rate selection jumpers (DMC-1414)**

Elements You Need

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

1. **DMC-1412 Controller and 37-pin cable (order Cable-37) or DMC-1414.**
2. **Servo motor with Encoder or stepper motor.**
3. **Appropriate motor drive: Servo amp (Power Amplifier or AMP-1460) or stepper drive.**
4. **Power Supply for Amplifier**

   +5 V, ±12 V supply for DMC-1412 card level

   20 V to 60 V DC supply for DMC-1414

7. **Communication CD from Galil**
8. **WSDK Servo Design Software (not necessary, but strongly recommended)**
9. **Interface Module ICM-1460 with screw-type terminals or integrated Interface Module/Amplifier, AMP-1460. (Note: An interconnect module is not necessary, but strongly recommended. The DMC-1414 has a version of the ICM-1460 integrated internally).**

The motors may be servo (brush or brushless type) or steppers. The driver (amplifier) should be suitable for the motor and may be linear or pulse-width-modulated and it 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 +10 V command will generate the maximum required current. For example, if the motor peak current is 10 A, 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.

The DMC-1412 can provide sinusoidal commutation for brushless motors. The driver should accept two sinusoidal signals from the controller and sum them together to output the three phases to the brushless motor.

For step motors, the driver should accept step and direction signals. For start-up of a stepper motor system refer to Step 8c "Connecting Step Motors".

For the DMC-1414, the internal amplifier is a 20 V to 60 V brush PWM amplifier. Therefore, the motor must be a DC brush servo motor.

The WSDK software is highly recommended for first time users of the DMC-141X. It provides step-by-step instructions for system connection, tuning and analysis.

Installing the DMC-1400 Controller

Installation of a complete, operational DMC-141X system consists of 9 steps. These steps will be slightly different depending on the exact model of your controller (DMC-1412 or DMC-1414).

Step 1.	**Determine overall motor configuration.**
Step 2.	**Install jumpers on the DMC-141X.**
Step 3a.	**Connect the AC power and serial cable to the DMC-1412**
OR	
Step 3b.	**Connect the 20 - 60 volt supply and serial cable to the DMC-1414.**
Step 4.	**Install the communications software.**
Step 5.	**Establish communications between the DMC-141X and the host PC.**
Step 6.	**Set-up axis for sinusoidal commutation (DMC-1412 only).**
Step 7.	**Make connections to amplifier and encoder.**
Step 8a.	**Connect standard brush or brushless servo motor.**
OR	
Step 8b.	**Connect brushless motor for sinusoidal commutation. (DMC-1412 only)**
OR	
Step 8c.	**Connect stepper motor**
OR.	
Step 8d.	**Connect brush motor to DMC-1414.**
Step 9.	**Tune servo system**.

Step 1. Determine Overall Motor Configuration

Before setting up the motion control system, the user must determine the desired motor configuration. The DMC-141X can control standard servo motors, sinusoidally commutated brushless motors or stepper motors. For control of other types of actuators, such as hydraulics, please contact Galil.

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

Standard Servo Motor Operation:

The DMC-141X has been set up by the factory for standard servo motor operation providing an analog command signal of ±10 volts. No hardware or software configuration is required for standard servo motor operation.

Sinusoidal Commutation:

Sinusoidal commutation is configured through a single software command, BA. This setting causes the controller to reconfigure the control axis to output two commutated phases. Only the DMC-1412 allows for sinusoidal commutation through the controller.

The single axis of commutation requires two DACs. In standard servo operation, the DMC-1412 has one DAC for the single axis. Issuing the BA command will enable the second DAC for commutation. Further instruction for sinusoidal commutation connections are discussed in Step 6.

Stepper Motor Operation:

To configure the DMC-141X for stepper motor operation, the controller requires that the command MT be given and a jumper placed. The installation of the stepper motor jumper is discussed in the following section entitled "Configuring Jumpers on the DMC-141X". Further instructions for stepper motor connections are discussed in Step 8c.

Step 2. Configuring Jumpers on the DMC-141X

Master Reset Jumpers

The jumper labeled MRST is the Master Reset jumper. This is located at JP1 for the DMC-1412 and at JP6 for the DMC-1414. When the MRST jumper is connected, the controller will perform a master reset upon power up. Whenever the controller has a master reset, all motion control parameters stored in EEPROM will be ERASED.

Stepper Motor Jumpers

If the DMC-141X will be driving a stepper motor, the stepper mode (SMX) jumper must be connected. This jumper is labeled JP1 for the DMC-1412 and JP2 for the DMC-1414. The jumper location marked OPT is for use by Galil technicians only.

Setting the Baud Rate on the DMC-1412 and DMC-1414

The jumper locations JP1 on the DMC-1412 and JP6 on the DMC-1414 allow the user to select the serial communication baud rate. The baud rate can be set using the following table:

9600 Jumper	38.4K Jumper	Baud Rate
No Jumper	No Jumper	19.2K
Jumper	No Jumper	9600
No Jumper	Jumper	38.4K
Jumper	Jumper	1200

The default baud rate for the controller is 19.2K.

Step 3a. Connecting AC or DC power and the Serial Cable to the DMC-1412

1. Insert 37-pin I/O cable to J3.
2. Use the 9-pin RS232 ribbon cable to connect the MAIN SERIAL port of the DMC-1412 to your computer or terminal communications port. The DMC-1412 main serial port is configured as DATASET. Your computer or terminal must be configured as a DATATERM for full duplex, no parity, 8 bits data, one start bit and one stop bit.

   Your computer needs to be configured as a "dumb" terminal which sends ASCII characters as they are typed to the DMC-1412.
3. Connect the AC cord for the box level controller. AC power requirement is single phase, 50 or 60 Hz at 90 to 260 volts.
4. If you are using the card level DMC-1412, apply ±12 V and +5 V power to the J5 connector.
5. Applying power will turn on the green LED power indicator.

Step 3b. Connecting DC power and the Serial Cable to the DMC-1414

1. Use the 9-pin RS232 ribbon cable to connect the MAIN SERIAL port of the DMC-1414 to your computer or terminal communications port. The DMC-1414 main serial port is configured as DATASET. Your computer or terminal must be configured as a DATATERM for full duplex, no parity, 8 bits data, one start bit and one stop bit.

   Your computer needs to be configured as a "dumb" terminal which sends ASCII characters as they are typed to the DMC-1414.
2. Connect a single, external DC supply from 20 to 60 volts to the 5-pin box connector, labeled AMP V+ and GND. This supply provides power for both the motion controller and internal PWM brush amplifier.

   Warning: Damage to the DMC-1414 will occur if a supply larger than 60 V is connected to the controller.
3. Applying power will turn on the green LED power indicator.

Step 4. Installing the Communications Software

After applying power to the computer, you should install the Galil software that enables communication between the controller and PC. **The CD-ROM used for the following installations is Version 11/01.**

Using DOS:

Using the Galil Software CD-ROM, go to the directory, **July2000 CD/DMCDOS/DISK1**. Type **INSTALL** at the DOS prompt and follow the directions.

Using Windows 98 Second Edition (SE), NT 4, ME, 2000 or XP:

The Galil Software CD-ROM will open an HTML page automatically as soon a Instead, **Explore** the CD and go to the **July2000 CD** folder. To install the basic communications software click on **DMCTERM** and then run the application, **DMCTERM**. The other basic terminal software is called **DMCWIN32** and is located under **July2000 CD/DMCWIN**. The Windows Servo Design Kit (**WSDK32**), which is useful for tuning servos and viewing useful controller information, can be downloaded off the CD as well. However, **WSDK32** is a purchase only software package and is password protected on the CD. Contact Galil for purchase information.

Step 5. Establishing Communication between the DMC-141X and the host PC

Using Galil Software for DOS

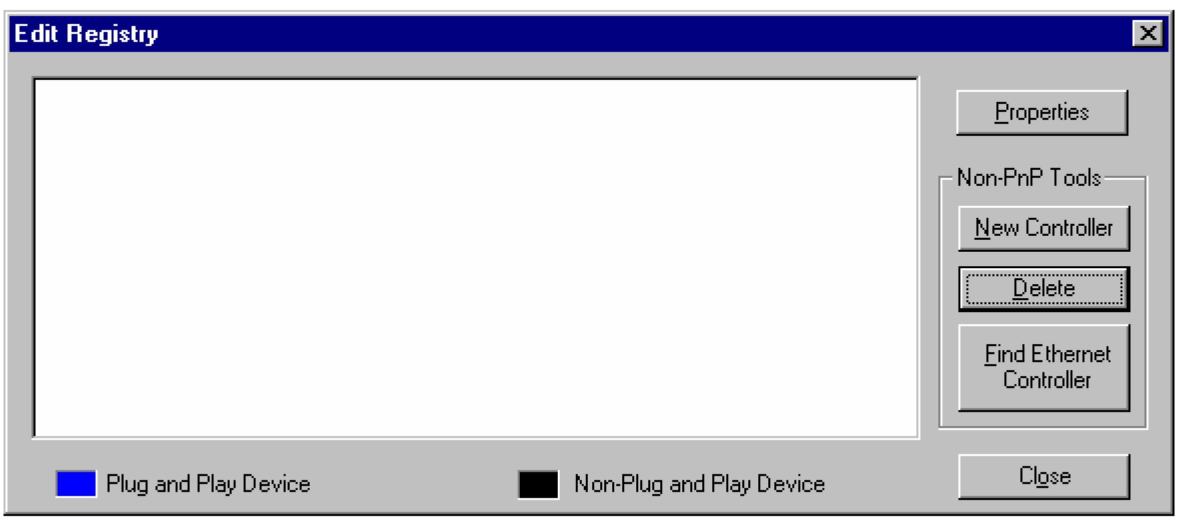
To communicate with the DMC-141X, type DMCTERM at the prompt. You will need to provide information about your controller such as controller type (DMC-1412 or DMC-1414), port number and baud rate. Once you have established communication, the terminal display should show a colon, :. If you do not receive a colon, press the carriage return.

If a colon prompt is not returned, and you are using the DMC-1412 or DMC-1414, there is most likely an incorrect setting of the serial communications port. The user must ensure that the correct communication port and baud rate are specified when attempting to communicate with the controller. Please note that the serial port on the controller must be set for handshake mode for proper communication with Galil software. The user must also insure that the proper serial cable is being used. See appendix for pin-out of serial cable. (NOTE: A "Null Modem" cable will NOT work with the main serial port.)

Using Galil Software for Windows 98 SE, NT 4, ME, 2000 and XP

The registration process for the DMC-1412/1414 controllers in these operating systems is very similar to the Windows 3.x/95/98 procedure.

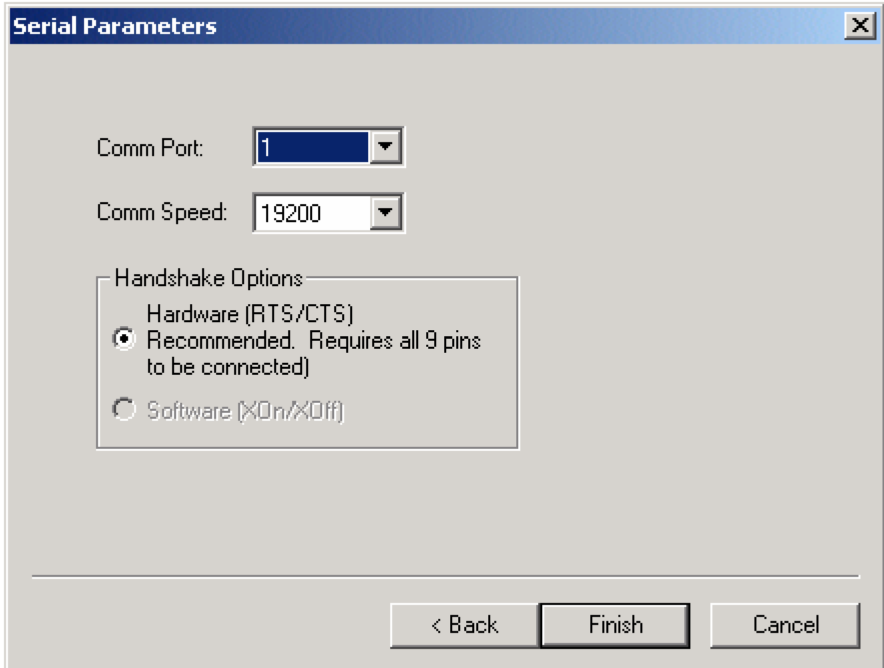
In DMC Terminal or WSDK, the Galil registry is accessed in the File menu by selecting "Register Controller". In DMCWIN, just click on the Registry menu button. The Galil Registry Dialog is shown below.



Select the button that says "New Controller" under the "Non-PnP Tools" and then select DMC-1412 from the pull down menu. Click 'Next'.

Note: The DMC-1414 must be registered as a DMC-1412.

The next step is to select the Comm Port being used on the PC and the Comm Speed for data transfer. Hardware handshaking will be selected by default. Select 'Next', and the controller will be entered into the registry. Connect to the controller by selecting the Terminal utility and choosing the controller from the registry list.



Note: Be sure to configure the Comm Speed jumpers for the same Comm Speed in the Galil Registry. No jumpers on the DMC-1412 and DMC-1414 indicates a Comm Speed of 19200 bits per second.

Sending Test Commands to the Terminal:

After you connect your terminal, press <carriage return> or the <enter> key on your keyboard. In response to carriage return <CR>, the controller responds with a colon, :

Now type

TPX <CR>

This command directs the controller to return the current position of the X axis. The controller should respond with a number such as

0000000

Step 6. Set-up axis for sinusoidal commutation (DMC-1412 only)

* This step is only required when the controller will be used to control a brushless motor with sinusoidal commutation.

The command BA is used to specify sinusoidal commutation mode for the DMC-1412. In this mode the controller will output two sinusoidal phases for the DACs. Once specified, follow the procedure outlined in Step 8b.

Step 7. Make connections to amplifier and encoder

Once you have established communications between the software and the DMC-141X, you are ready to connect the rest of the motion control system. The motion control system generally consists of an ICM-1460 Interface Module, a servo amplifier, and a motor to transform the current from the servo amplifier into torque for motion. Galil also offers the AMP-1460 Interface Module which is an ICM-

1460 equipped with a servo amplifier for a DC motor. The DMC-1414 contains an interconnect module, brush servo amplifier and power supply internally.

A signal breakout board of some type is strongly recommended. If you are using a breakout board from a third party, consult the documentation for that board to insure proper system connection.

If you are using the ICM-1460 or AMP-1460 with the DMC-1412, connect the 37-pin cable between the controller and interconnect module.

System connection procedures will depend on which components are included in your system.

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

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

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

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

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

The amplifier enable signal is used by the controller to disable the motor. This signal is labeled AMPEN on the ICM-1460 and should be connected to the enable signal on the amplifier. Note that many amplifiers designate this signal as the INHIBIT signal. Use the command, MO, to disable the motor amplifiers - check to insure that the motor amplifiers have been disabled (often this is indicated by an LED on the amplifier).

This signal changes under the following conditions: the watchdog timer activates, the motor-off command, MO, is given, or the OE1 command (Enable Off-On-Error) is given and the position error exceeds the error limit. As shown in Figure 3-1, AEN can be used to disable the amplifier for these conditions.

The standard configuration of the AEN signal is TTL active high. In other words, the AEN signal will be high when the controller expects the amplifier to be enabled. The polarity and the amplitude can be changed if you are using the ICM-1460 interface board. To change the polarity from active high (5 volts = enable, zero volts = disable) to active low (zero volts = enable, 5 volts = disable), replace the 7407 IC with a 7406. To change the voltage level of the AEN signal, note the state of the resistor pack on the ICM-1460. When Pin 1 is on the 5 V mark, the output voltage is 0-5 V. To change to 12 volts, pull the resistor pack and rotate it so that Pin 1 is on the 12 volt side. If you remove the resistor pack, the output signal is an open collector, allowing the user to connect an external supply with voltages up to 24 V.

Step C. Connect the encoders

For stepper motor operation, an encoder is optional.

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

The DMC-141X accepts single-ended or differential encoder feedback with or without an index pulse. If you are not using the AMP-1460 or the ICM-1460, you will need to consult the appendix for the encoder pinouts for connection to the motion controller. The AMP-1460 and the ICM-1460 can accept encoder feedback from a 10-pin ribbon cable or individual signal leads. For a 10-pin ribbon cable encoder, connect the cable to the protected header connector labeled JP2. For individual wires, simply match the leads from the encoder you are using to the encoder feedback inputs on the interconnect board. The signal leads are labeled MA+, MB+, and IDX+. These labels represent channel A, channel B, and the INDEX pulse, respectively. For differential encoders, the complement signals are labeled MA-, MB-, and IDX-.

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

Step D. Verify proper encoder operation.

Once the encoder is connected as described above, turn the motor shaft and interrogate the position with the instruction TP <return>. The controller response will vary as the motor is turned.

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

1. The encoder connections are incorrect - check the wiring as necessary.

2. The encoder has failed - using an oscilloscope, observe the encoder signals. Verify that both channels A and B have a peak magnitude between 5 and 12 volts. Note that if only one encoder channel fails, the position reporting varies by one count only. If the encoder failed, replace the encoder. If you cannot observe the encoder signals, try a different encoder.

3. There is a hardware failure in the controller - connect the same encoder to a different axis. If the problem disappears, you probably have a hardware failure. Consult the factory for help.

Step E. Connect Hall Sensors if available.

Hall sensors are only used with sinusoidal commutation on the DMC-1412 and are not necessary for proper operation. The use of hall sensors allows the controller to automatically estimate the commutation phase upon reset and also provides the controller the ability to set a more precise commutation phase. Without hall sensors, the commutation phase must be determined manually.

The hall effect sensors are connected to the digital inputs of the controller. These inputs can be used with the general purpose inputs (bits 1 - 7).

Each set of inputs must use inputs that are in consecutive order. The input lines are specified with the command, BI. For example, if the Hall sensors are connected to inputs 5, 6 and 7, use the instruction:

BI5 <CR>

Step 8a. Connect Standard Brush or Brushless Servo Motor

The following discussion applies to connecting the DMC-141X controller to standard servo motor amplifiers:

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

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

Check the Polarity of the Feedback Loop

It is assumed that the motor and amplifier are connected together and that the encoder is operating correctly (Step D). Before connecting the motor amplifiers to the controller, read the following discussion on setting Error Limits and Torque Limits.

Step A. Set the Error Limit as a Safety Precaution

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

Input the commands:

ER 2000 <CR>	Sets error limit to be 2000 counts
OE 1 <CR>	Disables amplifier when excess error exists

If the motor runs away and creates a position error of 2000 counts, the motor amplifier will be disabled.

Note: This function requires the AEN signal to be connected from the controller to the amplifier.

Step B. Setting Torque Limit as a Safety Precaution

To limit the maximum voltage signal to your amplifier, the DMC-141X controller has a torque limit command, TL. This command sets the maximum voltage output of the controller and can be used to avoid excessive torque or speed when initially setting up a servo system.

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

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

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

TL 1 <CR>	Sets torque limit to 1 volt

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

Step C. Disable motor

Issue the motor off command to disable the motor.

MO <CR>	Turns motor off

Step D. Connecting the Motor

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

Issue the servo here command to turn the motors on. To test the polarity of the feedback, command a move with the instruction:

SH <CR>	Servo Here to turn motors on

PR 1000 <CR>	Position relative 1000 counts
BG <CR>	Begin motion

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

Inverting the Loop Polarity

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

NOTE: To avoid a runaway condition after a Master Reset, it is recommended that the motor wires be physically inverted rather than using the software commands.

Sometimes the feedback polarity is correct (the motor does not attempt to run away) 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.

If the motor moves in the required direction but stops short of the target, it is most likely due to insufficient torque output from the motor command signal ACMD. This can be alleviated by reducing system friction on the motors. The instruction:

TT <CR>	Tell torque

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 (servo system tuning) to adjust the PID filter parameters, KP, KD and KI. It is necessary to accurately tune your servo system to ensure fidelity of position and minimize motion oscillation as described in the next section.

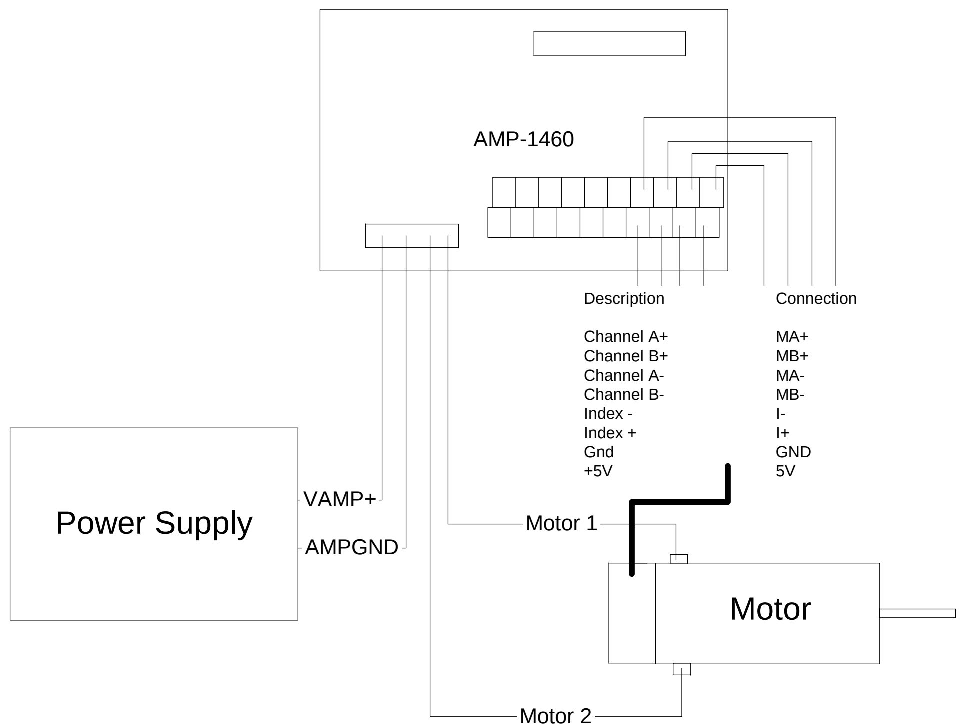


Figure 2.3 - System Connections with the AMP-1460 Amplifier

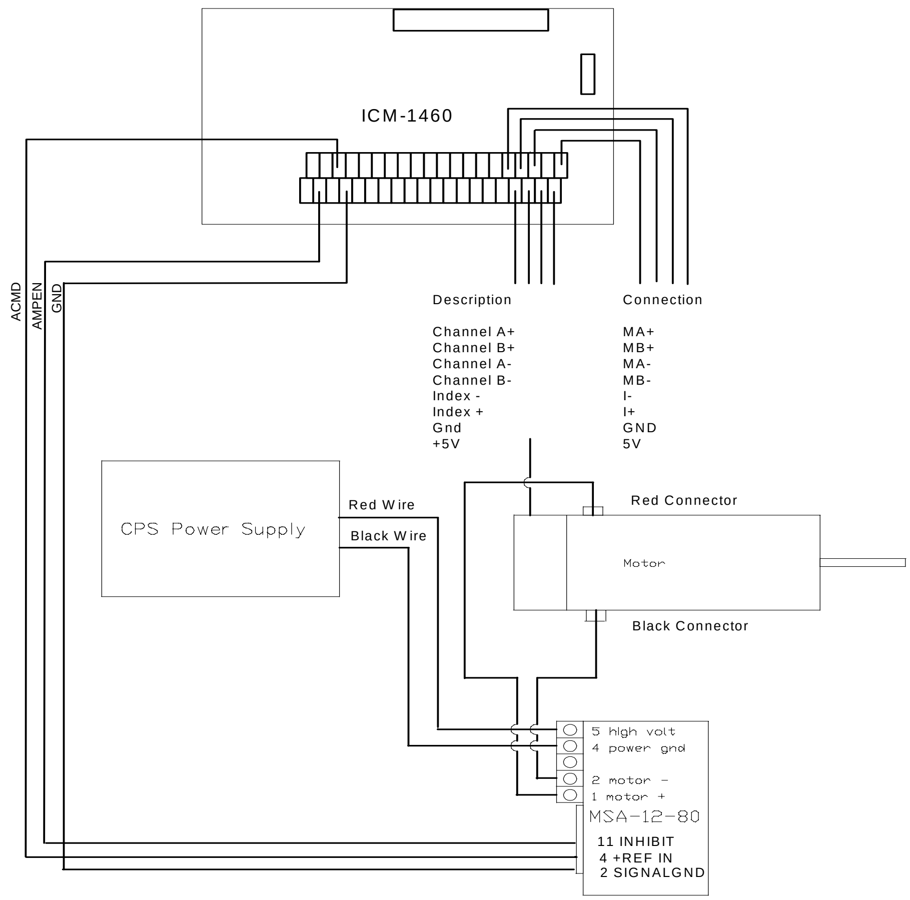


Figure 2.4 System Connections with a separate amplifier (MSA 12-80). This diagram shows the connections for a standard DC Servo Motor and encoder.

Step 8b. Connect Brushless Motor for Sinusoidal Commutation (DMC-1412 Hardware Rev D and newer)

The sinusoidal commutation option is available only on the DMC-1412. When using sinusoidal commutation, the parameters for the commutation must be determined and saved in the controllers non-volatile memory. The servo can then be tuned as described in Step 9.

Step A. Disable the motor amplifier

Use the command, MO, to disable the motor amplifiers.

Step B. Connect the motor amplifier to the controller.

The sinusoidal commutation amplifier requires 2 signals, usually denoted as Phase A & Phase B. These inputs should be connected to the two sinusoidal signals generated by the controller. The first signal is the main controller motor output, ACMD. The second signal utilizes the second DAC on the controller and is brought out on the ICM-1460 at pin 38 (ACMD2).

It is not necessary to be concerned with cross-wiring the 1^{st} and 2^{nd} signals. If this wiring is incorrect, the setup procedure will alert the user (Step D).

Step C. Specify the Size of the Magnetic Cycle.

Use the command, BM, to specify the size of the brushless motors magnetic cycle in encoder counts. For example, if you are using a linear motor where the magnetic cycle length is 62 mm, and the encoder resolution is 1 micron, the cycle equals 62,000 counts. This can be commanded with the command:

BM 62000 <CR>

On the other hand, if you are using a rotary motor with 4000 counts per revolution and 3 magnetic cycles per revolution (three pole pairs) the command is

BM 1333.333 <CR>

Step D. Test the Polarity of the DACs and Hall Sensor Configuration.

Use the brushless motor setup command, BS, to test the polarity of the output DACs. This command applies a certain voltage, V, to each phase for some time T, and checks to see if the motion is in the correct direction.

The user must specify the value for V and T. For example, the command

BS 2,700 <CR>

will test the brushless axis with a voltage of 2 volts, applying it for 700 millisecond for each phase. In response, this test indicates whether the DAC wiring is correct and will indicate an approximate value of BM. If the wiring is correct, the approximate value for BM will agree with the value used in the previous step.

Note: In order to properly conduct the brushless setup, the motor must be allowed to move a minimum of one magnetic cycle in both directions.

Note: When using Galil Windows software, the timeout must be set to a minimum of 10 seconds (time-out = 10000) when executing the BS command. This allows the software to retrieve all messages returned from the controller.

If Hall Sensors are Available:

Since the Hall sensors are connected randomly, it is very likely that they are wired in the incorrect order. The brushless setup command indicates the correct wiring of the Hall sensors. The hall sensor wires should be re-configured to reflect the results of this test.

The setup command also reports the position offset of the hall transition point and the zero phase of the motor commutation. The zero transition of the Hall sensors typically occur at 0°, 30° or 90° of the phase commutation. It is necessary to inform the controller about the offset of the Hall sensor and this is done with the instruction, BB.

Step E. Save Brushless Motor Configuration

It is very important to save the brushless motor configuration in non-volatile memory. After the motor wiring and setup parameters have been properly configured, the burn command, BN, should be given.

If Hall Sensors are Not Available:

Without hall sensors, the controller will not be able to estimate the commutation phase of the brushless motor. In this case, the controller could become unstable until the commutation phase has been set using the BZ command (see next step). It is highly recommended that the motor off command be given before executing the BN command. In this case, the motor will be disabled upon power up or reset and the commutation phase can be set before enabling the motor.

Step F. Set Zero Commutation Phase

When an axis has been defined as sinusoidally commutated, the controller must have an estimate for commutation phase. When hall sensors are used, the controller automatically estimates this value upon reset of the controller. If no hall sensors are used, the controller will not be able to make this estimate and the commutation phase must be set before enabling the motor.

If Hall Sensors are Not Available:

To initialize the commutation without Hall effect sensor use the command, BZ. This function drives the motor to a position where the commutation phase is zero, and sets the phase to zero.

The BZ command argument is a real number which represents the voltage to be applied to the amplifier during the initialization. When the voltage is specified by a positive number, the initialization process ends up in the motor off (MO) state. A negative number causes the process to end in the Servo Here (SH) state.

Warning: This command must move the motor to find the zero commutation phase. This movement is instantaneous and will cause the system to jerk. Larger applied voltages will cause more severe motor jerk. The applied voltage will typically be sufficient for proper operation of the BZ command. For systems with significant friction, this voltage may need to be increased and for systems with very small motors, this value should be decreased.

For example,

BZ -2 <CR>

will drive the axis to zero, using a 2 V signal. The controller will then leave the motor enabled. For systems that have external forces working against the motor, such as gravity, the BZ argument must provide a torque 10x the external force. If the torque is not sufficient, the commutation zero may not be accurate.

If Hall Sensors are Available:

The estimated value of the commutation phase is good to within 30°. This estimate can be used to drive the motor but a more accurate estimate is needed for efficient motor operation. There are 3 possible methods for commutation phase initialization:

Method 1. Use the BZ command as described above.

Method 2. Drive the motor close to commutation phase of zero and then use BZ command. This method decreases the amount of system jerk by moving the motor close to zero commutation phase before executing the BZ command. The controller makes an estimate for the number of encoder counts between the current position and the position of zero commutation phase. This value is stored in the operand _BZx. Using this operand the controller can be commanded to move the motor. The BZ command is then issued as described above. For example, to initialize the X axis motor upon power or reset, the following commands may be given:

SH <CR>	Enable X axis motor
PRX=-1*(_BZX) <CR>	Move X motor close to zero commutation phase
BG <CR>	Begin motion on X axis
AM <CR>	Wait for motion to complete on X axis
BZX=-1 <CR>	Drive motor to commutation phase zero and leave motor on

Method 3. Use the command, BC. This command uses the hall transitions to determine the commutation phase. Ideally, the hall sensor transitions will be separated by exactly 60° and any deviation from 60° will affect the accuracy of this method. If the hall sensors are accurate, this method is recommended. The BC command monitors the hall sensors during a move and monitors the Hall sensors for a transition point. When

that occurs, the controller computes the commutation phase and sets it. For example, to initialize the motor upon power or reset, the following commands may be given:

SH <CR>	Enable motor
BC <CR>	Enable the brushless calibration command
PR 50000 <CR>	Command a relative position movement
BG <CR>	Begin motion. When the hall sensors detect a phase transition, the commutation phase is re-set.

Step 8c. Connect Step Motors

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

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

The DMC-141X profiler commands the step motor amplifier. All DMC-141X motion commands apply such as PR, PA, VP, CR and JG. The acceleration, deceleration, slew speed and smoothing are also used. Since step motors run open-loop, the PID filter does not function and the position error is not generated.

To connect step motors with the DMC-141X you must follow this procedure:

Step A. Install SM jumpers

In order for the DMC-141X to operate in stepper mode, the corresponding stepper motor jumper installed. For a discussion of SM jumpers, see section *Step 2. Install jumpers on the DMC-141X.*

Step B. Connect step and direction signals from controller to motor amplifier

Connect the step and direction signals from the controller to respective signals on your step motor amplifier. (These signals are labeled PWM and SIGN on the ICM-1460). Consult the documentation for your step motor amplifier.

Step C. Configure DMC-141X for motor type using MT command.

You can configure the DMC-141X for active high or active low pulses. Use the command MT 2 for active high step motor pulses and MT -2 for active low step motor pulses. *See description of the MT command in the Command Reference.*

Step 8d. Connect Brush Motor to the DMC-1414

The DMC-1414 provides an integrated brush-type amplifier, interconnect module and power supply to be used with DC brush motors.

Warning: The DMC-1414 is powered up in the motor on (SH) condition. It is recommended that the MO command is given before connecting the motor in order to prevent a runaway due to reversed polarity. This command then needs to be burned into the EEPROM with the BN command.

To connect the DC brush motor to the DMC-1414, follow this procedure:

Step A. Disconnect controller power.

Unplug the 5-pin power connector from the front of the unit. This will power down the controller so that the motor may be connected.

Step B. Connect DC brush motor.

Connect the motor leads to the screw terminals corresponding to MOTOR1 and MOTOR2. It is assumed that the encoder is already connected and verified operational.

Step C. Reconnect power to controller.

Reconnect the 5-pin power connector to the DMC-1414. This will power the motor and allow communication with the controller. Test communication by sending the TP command and receiving a valid response.

Step D. Test polarity of feedback loop.

With the hardware connections complete, the next step is to test the polarity of the feedback loop to limit a runaway situation. For this procedure, please refer to *Step 8a. Connect Standard Servo Motor* for the section *Check the Polarity of the Feedback Loop.* Note: Before the PR moves are issued in the tests, but after the error limits have been set, the SH command needs to be sent to turn on the servo motor.

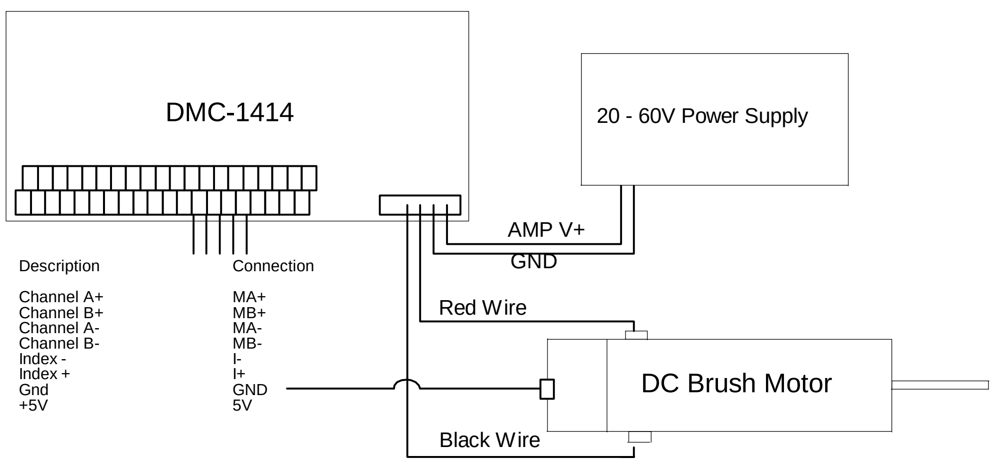


Figure 2.5 System connections for the DMC-1414 with integrated amplifier, interconnect module and amplifier.

Step 9. Tune the Servo System

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 Windows Servo Design Kit (WSDK software).

If the torque limit was set as a safety precaution in the previous step, you may want to increase this value. See Step B of the above section “Setting Torque Limit as a Safety Precaution”.

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

To start, set the integrator to zero with the instruction

KI 0 <CR>   Integrator gain and set the proportional gain to a low value, such as

KP 1 <CR>	Proportional gain
KD 100 <CR>	Derivative gain

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

TE <CR>	Tell error

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

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

KP 10 <CR>	Proportion gain
TE <CR>	Tell error

As the proportional gain is increased, the error decreases.

Again, the system may vibrate if the gain is too high. In this case, reduce KP. Typically, KP should not be greater than KD/4.

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

<CR>

becomes zero. As KI is increased, its effect is amplified and it may lead to vibrations. If this occurs, simply reduce KI.

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

Design Examples

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

Example 1 - System Set-up

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

Instruction	Interpretation
KP 10	Set proportional gain
KD 100	Set damping
KI 1	Set integral
OE 1	Set error off
ER 1000	Set error limit

Example 2 - Profiled Move

Objective: Rotate 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/s^2.

Instruction	Interpretation
PR 10000	Distance
SP 20000	Speed

DC 100000	Deceleration
AC 100000	Acceleration
BG	Start Motion

In response, the motor turns and stops.

Example 3 - Position Interrogation

The position of the axis may be interrogated with the instruction

TP	Tell position

which returns the position of the main encoder.

The position error, which is the difference between the commanded position and the actual position can be interrogated by the instructions

TE	Tell error

Example 4 - Absolute Position

Objective: Command motion by specifying the absolute position.

Instruction	Interpretation
DP 0	Define the current position as 0
PA 7000	Sets the desired absolute position
BG	Start motion

Example 5 - Velocity Control (Jogging)

Objective: Drive the motor at specified speeds.

Instruction	Interpretation
JG 10000	Set Jog Speed
AC 100000	Set acceleration
DC 50000	Set deceleration
BG	Start motion

after a few seconds, command:

JG -40000	New speed and Direction
TV	Returns speed

This causes velocity changes including direction reversal. The motion can be stopped with the instruction

ST	Stop

Example 6 - 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 to 0.2 volts
JG 10000	Set speed
BG	Start motion

The 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 7 - Interrogation

The values of the parameters may be interrogated using a ?. For example, the instruction

KP ?	Return gain

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

Example 8 - 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	Start the motion

Example 9 - 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. For example, in response to the first ED command, the Galil Windows software will open a simple editor window. From this window, the user can type in the following program:

#A	Define label
PR 700	Distance
SP 2000	Speed
BG	Start motion
EN	End program

This program can be downloaded to the controller by selecting the File menu option download. Once this is done, close the editor. Now the program may be executed with the command

XQ #A	Start the program running

Example 10 - Motion Programs with Loops

Motion programs may include conditional jumps as shown below.

Instruction	Interpretation
#A	Label
DP 0	Define current position as zero
V1=1000	Set initial value of V1

#Loop	Label for loop
PA V1	Move motor V1 counts
BG	Start motion
AM	After motion is complete
WT 500	Wait 500 ms
TP	Tell position
V1=V1+1000	Increase the value of V1
JP #Loop,V1<10001	Repeat if V1<10001
EN	End

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

XQ #A	Execute Program #A

Example 11- Motion Programs with Trippoints

The motion programs may include trippoints as shown below.

Instruction	Interpretation
#B	Label
DP	Define initial position
PR 30000	Set target
SP 5000	Set speed
BG	Start motion
AD 4000	Wait until X moved 4000
TP	Tell position
EN	End program

To start the program, command:

XQ #B	Execute Program #B

Example 12 - Control Variables

Objective: To show how control variables may be utilized.

Instruction	Interpretation
#A;DP0	Label; Define current position as zero
PR 4000	Initial position
SP 2000	Set speed
BG	Move
AM	Wait until move is complete
WT 500	Wait 500 ms
#B	
V1 = _TP	Determine distance to zero
PR -V1/2	Command move 1/2 the distance
BG	Start motion
AM	After motion
WT 500	Wait 500 ms
V1=	Report the value of V1
JP #C, V1=0	Exit if position=0

JP #B	Repeat otherwise
#C;EN	End

To start the program, command

XQ #A	Execute Program #A

This program moves the motor to an initial position of 1000 and returns it to zero on increments of half the distance. Note, _TP is an internal variable which returns the value of the position. Internal variables may be created by preceding a DMC-141X instruction with an underscore, _.

Example 13 - Control Variables and Offset

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

Instruction	Interpretation
#A	Set initial values
KI0	
DP0	
V1=8; V2=0	Initializing variables to be used by program
#B	Program label #B
OF V1	Set offset value
WT 200	Wait 200 msec
V2=_TP	Set variable V2 to the current position
JP#C,@ABS[V2]<2	Exit if error small
MG V2	Report value of V2
V1=V1-1	Decrease Offset
JP #B	Return to top of program
#C;EN	End

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

Chapter 3 Hardware Interface

Overview

The DMC-141X provides TTL digital inputs for forward limit, reverse limit, home, and abort signals. The controller also has 7 uncommitted inputs (for general use) as well as 3 TTL outputs. This chapter describes the inputs and outputs and their proper connection.

All of the controller signal lines are accessible through the main 37-pin connector J3 for the DMC-1412. The ICM-1460 provides easy access to these signals through screw terminals. The DMC-1414 provides access to all signals through the integrated screw terminals.

Encoder Interface

The DMC-141X accepts inputs from incremental encoders with two channels in quadrature, or 90 electrical degrees out of phase. The DMC-141X 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-141X also accepts inputs from an additional encoder. This is called the auxiliary encoder and can be used for dual-loop applications.

The encoder inputs are not isolated.

All of the encoder signals for the DMC-1412 are accessible through the ICM-1460 or directly from the interface connector on the controller. The encoder signals for the DMC-1414 are accessible through the integrated screw terminals. The pin-outs of the ICM-1460, the connectors and the DMC-1414 are explained in the appendix.

The DMC-141X 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-141X for pulse and direction encoder format. For pulse and direction format, the DMC-141X provides a resolution of 1X counts per pulse.

Note that while TTL level signals are common, the DMC-141X encoder inputs accept signals in the range of ±12 V. 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 12 V, connect 7 volts to Channel A complement input.

Inputs

The DMC-141X provides buffered digital inputs for limit switches, homing, abort as well as 7 uncommitted inputs. The Limit switches, Home switch, Abort switch and general purpose inputs are all TTL and accessible through the ICM-1460 or DMC-1414 screw terminals. A description of their usage is found below.

Limit Switch Input

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

When a forward or reverse limit switch is activated, the current application program that is running will be interrupted and the controller will automatically jump to the #LIMSWI subroutine if one exists. This is a subroutine which the user can include in any motion control program and is useful for executing specific instructions upon activation of a limit switch. Automatic Subroutines are discussed in Chapter 6.

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

The operands, _LF and _LR, contain the state of the forward and reverse limit switches, respectively. The value of the operand is either a ‘0’ or ‘1’ corresponding to the logic state of the limit switch. Using a terminal program, the state of a limit switch can be printed to the screen with the command, MG _LF or MG _LR. This prints the value of the limit switch operands for the axis. The logic state of the limit switches can also be interrogated with the TS command. For more details on TS see the Command Reference.

Home Switch Input

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

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

There are three homing routines supported by the DMC-141X: Find Edge (FE), Find Index (FI), and Standard Home (HM).

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

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

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

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

Abort Input

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

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

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

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

Uncommitted Digital Inputs

The general use inputs are TTL and are accessible through the ICM-1460 or DMC-1414 as IN1-IN7. These inputs can be interrogated with the use of the command TI (Tell Inputs), the operand _TI and the function @IN[]. (see Chapter 7, Mathematical Functions and Expressions).

NOTE: For systems using the ICM-1460 interconnect module, there is an option to provide opto-isolation on the inputs. In this case, the user provides an isolated power supply (+5 V to +24 V and ground). For more information, consult Galil.

The inputs can be accessed directly from the 37 or 40 pin connector on the controller, also. For a description of the pinouts, consult the appendix.

Outputs

The DMC-141X provides three general use outputs and an error signal output.

The general use outputs are TTL and are accessible through the ICM-1460 or DMC-1414 as OUT0, OUT1 and OUT2. These outputs can be turned On and Off with the commands, SB (Set Bit), CB (Clear Bit), OB (Output Bit), and OP (Output Port). For more information about these commands, see the Command Summary. The value of the outputs can be checked with the operand _OP and the function @OUT[n] (see Chapter 7, Mathematical Functions and Expressions).

The error signal output is available on the interconnect module as ERROR. This is a TTL signal which is low when the controller has an error.

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

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

The outputs can be accessed directly from the 37 or 40 pin connector on the controller. For a description of the pinouts, consult the appendix.

Amplifier Interface

The DMC-141X generates a ±10 volt range analog signal, ACMD, and ground (pin 21) for input to power amplifiers which have been sized to drive the motor and load. For best performance, the amplifier 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.

A second DAC output is provided on the DMC-1412 for use as the second phase for sinusoidal commutation.

The ICM-1460 and DMC-1414 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 3.1, AEN can be used to disable the amplifier for these conditions.

The standard configuration of the AEN signal is TTL active high. In other words, the AEN signal will be high when the controller expects the amplifier to be enabled. The polarity and the amplitude can be changed if you are using the ICM-1460 interface board. To change the polarity from active high (5 volts= enable, zero volts = disable) to active low (zero volts = enable, 5 volts= disable), replace the 7407 IC with a 7406. Note that many amplifiers designate the enable input as ‘inhibit’.

To change the voltage level, note the state of the jumper on the ICM-1460. When JP4 has a jumper from “AEN” to “5V” (default setting), the output voltage is 0-5V. To change to 12 volts, pull the jumper and rotate it so that it connects the pins marked “AEN” and “+12V”. If the jumper is removed entirely, the output is an open collector signal, allowing the user to connect to external supplies with voltages up to 24 V.

Note: The DMC-1414 provides an internal DC brush type amplifier. No external amplifier connections are needed.

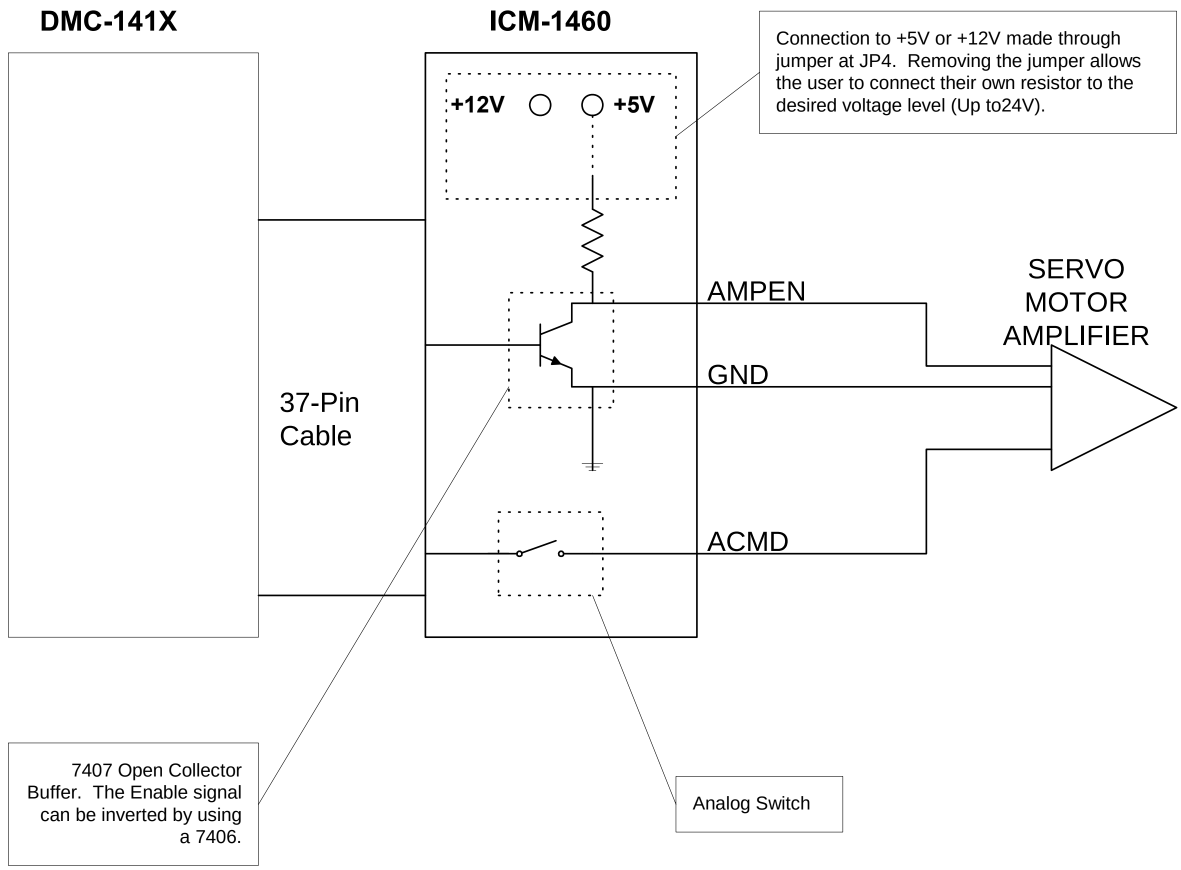


Figure 3.1 - Connecting AEN to an amplifier

Other Inputs

A reset input is TTL level, non-isolated signal. The reset is used to locally reset the DMC-141X without resetting the PC.

THIS PAGE LEFT BLANK INTENTIONALLY

Chapter 4 Communication

Communication - DMC-1412 and DMC-1414

Introduction

The DMC-1412 and DMC-1414 have two RS232 ports. The main port is the data set and the auxiliary port is the data term. The main port baud rate can be configured through the jumper JP1 for the DMC-1412 and through JP6 for the DMC-1414. The auxiliary port for both can be configured with the software command CC. The auxiliary port can either be configured as a general port or for daisy-chain communications. The auxiliary port configuration can be saved using the Burn (BN) instruction.

RS232 Ports

The RS232 pin-out description for the main and auxiliary port is given below. Note, the auxiliary port is essentially the same as the main port except inputs and outputs are reversed. These pin-outs are also listed below.

Note: If you are connecting the RS232 auxiliary port to a terminal or any device which is a DATATERM, it is necessary to use a connector adapter (Null Modem), which changes a dataterm to a dataset.

RS232 - Main Port {P1}

1 CTS – output	6 CTS – output
2 Transmit Data – output	7 RTS – input
3 Receive Data – input	8 CTS – output
4 RTS – input	9 No connect
5 Ground	

RS232 - Auxiliary Port {P2}

1 CTS – input	6 CTS – input
2 Transmit Data – input	7 RTS – output
3 Receive Data – output	8 CTS – input
4 RTS – output	9 5 V
5 Ground	

Note: The DMC-1412 and DMC-1414 can also be configured for RS422. The RS422 conversion should be specified at time of purchase. The pin outs for the RS422 connection are as follows:

RS422 - Main Port {P1}

1 CTS – output	6 CTS + output
2 Transmit Data – output	7 Transmit + output
3 Receive Data – input	8 Receive + input
4 RTS – input	9 RTS + input
5 Ground	

RS422 - Auxiliary Port {P2}

1 CTS – input	6 CTS + input
2 Receive Data – input	7 Receive + input
3 Transmit Data – output	8 Transmit + output
4 RTS – output	9 RTS + output
5 Ground	

RS-485 is also available as a special option on the DMC-1414. Please consult the factory for details.

Configuration

Configure your PC for 8-bit data, one start-bit, one stop-bit, full duplex and no parity. The baud rate for the RS232 communication can be selected by setting the proper switch configuration on JP1 for the DMC-1412 or JP6 for the DMC-1414 according to the table below.

Baud Rate Selection

9600 Label	38.4K Label	Baud Rate
jumper	jumper	**1200**
jumper	no jumper	**9600**
no jumper	no jumper	**19.2K**
no jumper	jumper	**38.4K**

The RS232 main port is configured for hardware handshake where the RTS and CTS lines are used. The CTS line will go high whenever the DMC-141X is not ready to receive additional characters. The RTS line will inhibit the DMC-141X from sending additional characters. Note, the RTS line goes high for inhibit.

The auxiliary port of the DMC-141X can be configured either as a general port or for the daisy-chain. When configured as a general port, the port can be commanded to send ASCII messages to another DMC-141X controller or to a display terminal or panel.

(Configure Communication) at port 2. The command is in the format of:

CC m,n,r,p

where m sets the baud rate, n sets for either handshake or non-handshake mode, r sets for general port or the auxiliary port, and p turns echo on or off.

m - Baud Rate - 300,1200,4800,9600,19200,38400

n - Handshake - 0=No; 1=Yes

r - Mode - 0=General Port; 1=Daisy-chain

p - Echo - 0=Off; 1=On; Valid only if r=0

Note, for the handshake of the auxiliary port, the roles for the RTS and CTS lines are reversed.

Example:

CC 1200,0,0,1	Configure communication at port 2, with 1200 baud, no handshake, general port and echo turned on.

Daisy-Chaining

Up to eight DMC-141X controllers may be connected in a daisy-chain. One DMC-141X is connected to the host terminal via the RS232 at port 1 or the main port. Port 2 or the auxiliary port of that DMC-141X is then brought into port 1 of the next DMC-141X, and so on. The address of each of the DMC-141X is configured by the SAn command where n is a number between 0 and 7.

NOTE: The SA value may be saved by the BN command.

To communicate with any one of the DMC-141X units, give the command "%A", where A is the address of the board. All instructions following this command will be sent only to the board with that address. Only when a new %A command is given will the instruction be sent to another board. The only exception is "!" command. To talk to all the DMC-141X boards in the daisy-chain at one time, insert the character "!" before the software command. All boards receive the command, but only address 0 will echo.

Note: The CC command must be specified to configure the port P2 of each unit. Each controller in the chain must have this CC specified for daisy chain *with the exception of the terminating controller.*

Example: 2-axis motion system. Address 0 is a DMC-1412. Another DMC-1412 is set for Address 1.

Controller	**Required Motion:**
Address 0	X Axis is 500 counts
Address 1	X Axis is 700 counts

Software Command	**Interpretation**
%0	Talk only to controller 0 (First DMC-1412)
PR 500	Specify X distance
%1	Talk only to controller board 1 (Second DMC-1412)
PR 700	Specify X distance
!BG	Begin motion on both controllers

Unsolicited Messages Generated by Controller

When the controller is executing a program, it may generate responses which will be sent via the main RS-232 port. This response could be generated as a result of messages using the MG or IN command OR as a result of a command error. These responses are known as unsolicited messages since they are not generated as the direct response to a command.

Messages can be directed to a specific port using the specific Port arguments - see MG and IN commands described in the Command Reference. If the port is not explicitly given, unsolicited messages will be sent to the default port.

The controller has a special command, CW, which can affect the format of unsolicited messages. This command is used by Galil Software to differentiate response from the command line and unsolicited messages. The command, CW1 causes the controller to set the high bit of ASCII characters to 1 of all unsolicited characters. This may cause characters to appear garbled to some terminals. This function can be disabled by issuing the command, CW2. For more information, see the CW command in the Command Reference.

When hardware handshaking is used characters which are generated by the controller are placed in a FIFO buffer before they are sent out of the controller. The size of the RS-232 buffer is 128 bytes. When this buffer becomes full, the controller must either stop executing commands or ignore additional characters generated for output. The command CW,1 causes the controller to ignore all output from the controller while the FIFO is full. The command, CW ,0 causes the controller to stop executing new commands until more room is made available in the FIFO. This command can be very useful when hardware handshaking is being used and the communication line between controller and terminal will be disconnected. In this case, characters will continue to build up in the controller until the FIFO is full. For more information, see the CW command in the Command Reference.

Controller Response to DATA

Most DMC-141X 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-141X decodes each ASCII character (one byte) one at a time. It takes approximately .5 msec for the controller to decode each command.

After the instruction is decoded, the DMC-141X 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 at Tell Position (TP), the DMC-141X 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-141X response with the data sent. The echo is enabled by sending the command EO 1 to the controller.

Galil Software Tools and Libraries

API (Application Programming Interface) software is available from Galil. The API software is written in C and is included in the Galil Software CD. They can be used for development under DOS and Windows environments (16 and 32 bit Windows). With the API's, the user can incorporate already existing library functions directly into a C program.

Galil has also developed a Visual Basic Toolkit. This provides VBXs, 16-bit OCX's and 32-bit OCXs for handling all of the DMC-141X communications including support of interrupts. These objects install directly into Visual Basic and are part of the run-time environment. For more information, contact Galil.

Chapter 5 Programming Basics

Introduction

The DMC-141X 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-141X 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 serial link for immediate execution by the DMC-141X, or an entire group of commands can be downloaded into the DMC-141X 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-141X instruction set and syntax. A complete listing of all DMC-141X instructions is included in the DMC-1400 Series Command Reference.

Command Syntax

DMC-141X 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-141X command interpreter. Note: If you are using a Galil terminal program, commands will not be processed until an <enter> command is given. This allows the user to separate many commands on a single line and not begin execution until the user gives the <enter> command.

IMPORTANT: All DMC-141X 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.

To view the current values for each command, specify the command followed by a ?

Example Syntax for Specifying Data

PR 1000	Specify as 1000
PR ?	Interrogate value in PR register

Controller Response to Commands

The DMC-141X returns a : for valid commands.

The DMC-141X returns a ? for invalid commands.

For example, if the command BG is sent in lower case, the DMC-141X will return a ?.

:bg <enter>	invalid command, lower case
?	DMC-141X returns a ?

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

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

There are several coded reasons for receiving an invalid command response. The most common reasons are and 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.

Interrogating the Controller

Interrogation Commands

The DMC-141X has a set of commands that directly interrogate the controller. When the command is entered, the requested data is returned in decimal format on the next line followed by a carriage return and line feed. The format of the returned data can be changed using the Position Format (PF), Variable Format (VF) and Leading Zeros (LZ) command. See Chapter 7 and the Command Reference.

Summary of Interrogation Commands

RP	Report Command Position
RL	Report Latch
^R ^V	Firmware Revision Information
SC	Stop Code
TB	Tell Status
TC	Tell Error Code
TD	Tell Dual Encoder
TE	Tell Error
TI	Tell Input
TP	Tell Position
TR	Trace
TS	Tell Switches
TT	Tell Torque
TV	Tell Velocity

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

TP <enter>	Tell position
0000000000	Controllers Response

Interrogating Current Commanded Values.

Most commands can be interrogated by using a question mark (?). Type the command followed by a ?.

PR ?	Request X axis value

The controller can also be interrogated with operands.

Operands

Most DMC-141X commands have corresponding operands that can be used for interrogation. Operands must be used inside of valid DMC expressions. For example, to display the value of an operand, the user could use the command:

MG ‘operand’ where ‘operand’ is a valid DMC operand

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

V=_TP

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

Command Summary

Each DMC-141X command is described fully in the Command Reference 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 Position Relative 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
DC	Deceleration
DT	Contour Time Interval
FE	Find Edge
FI	Find Index
GR	Gear Ratio
HM	Home
IP	Increment Position
JG	Jog Mode
PA	Position Absolute
PR	Position Relative
SP	Speed
ST	Stop

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
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
BA	Brushless Axis (DMC-1412 only)
BB	Brushless Phase Begins (DMC-1412 only)
BC	Brushless Calibration (DMC-1412 only)
BD	Brushless Degrees (DMC-1412 only)
BI	Brushless Inputs (DMC-1412 only)
BM	Brushless Modulo (DMC-1412 only)
BN	Burn
BO	Brushless Offset (DMC-1412 only)
BP	Burn Program (DMC-1412, DMC-1414 only)
BS	Brushless Setup (DMC-1412 only)
BV	Burn Variable (DMC-1412, DMC-1414 only)
BZ	Brushless Zero (DMC-1412 only)
CB	Clear Bit
CC	Configure Communication (DMC-1412, DMC-1414 only)
CE	Configure Encoder Type
CN	Configure Switches and Stepper
DA	Deallocate Arrays
DE	Define Dual Encoder Position
DL	Download
DM	Dimension Arrays
DP	Define Position
EB	Enable ECAM
ED	Edit Mode
EG	Engage ECAM
EM	Cam cycle command
EO	Echo Off
EP	Cam table interval and starting point
EQ	Disengage ECAM
ET	ECAM table entry
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
SA	Set Address for daisy chaining
SB	Set Bit
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
KS	Stepper Smoothing Constant
OF	Offset
SH	Servo Here
TL	Torque Limit
TM	Sample Time
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 <enter>	Define axis position as 0
:PF 6 <enter>	Define position format as 6 digits
:PR 100 <enter>	Specify position command
:BG <enter>	Begin Motion
:TP <enter>	Tell Position
00100	Returned Position data
:PR? <enter>	Request Position Command
00100	Returned data
:tp	Enter invalid command

?	Controller response
:TC1 <enter>	Request error code
1 Unrecognized command	Controller response

Chapter 6 Programming Motion

Overview

The DMC-141X provides several modes of motion, including independent positioning and jogging, electronic cam electronic gearing, and contouring. Each one of these modes is discussed in the following sections.

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 axis follows prescribed velocity profile.	Point-to-Point Positioning	PA,PR SP,AC,DC,IT
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
Electronic gearing where axis is scaled to auxiliary encoder which can move in both directions.	Electronic Gearing	GR
Master/slave where slave axis must follow a master such as conveyer speed.	Electronic Gearing	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	DV

Following a trajectory based on a master encoder position	Electronic Cam	EM EP ET EB EG EQ
Motion Smoothing	Applies to all of the above motion. Smoothes motion to eliminate vibrations due to jerk (discontinuities in acceleration)	IT

Point - to - Point Positioning

In this mode, motion between the specified axes is independent, and each axis follows its own profile. The user specifies the desired absolute position (PA) or relative position (PR), slew speed (SP), acceleration ramp (AC), and deceleration ramp (DC), for each axis. On begin (BG), the DMC-141X profiler generates the corresponding trapezoidal or triangular velocity profile and position trajectory. The controller determines a new command position along the trajectory every sample period until the specified profile is complete. Motion is complete when the last position command is sent by the DMC-141X profiler. Note: The actual motor motion may not be complete when the profile has been completed, however, the next motion command may be specified.

The speed (SP) and the acceleration (AC) can be changed at any time during motion, however, the deceleration (DC) and position (PR or PA) cannot be changed until motion is complete. Remember, motion is complete when the profiler is finished, not when the actual motor is in position. The Stop command (ST) can be issued at any time to decelerate the motor to a stop before it reaches its final position.

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

Command Summary - Point to Point Positioning

PR n	Specifies relative distance (counts)
PA n	Specifies absolute position (counts)
SP n	Specifies slew speed (counts/sec)
AC n	Specifies acceleration rate (counts/sec ^{2})
DC n	Specifies deceleration rate (counts/sec ^{2})
BG	Starts motion
ST	Stops motion before end of move
IT	Time constant for independent motion smoothing
IP n	Changes position target by increment of n
AM	Trippoint for profiler complete
MC	Trippoint for "in position"

Operand Summary - Point to Point Positioning

_AC	Return acceleration rate
_DC	Return deceleration rate
_SP	Return speed
_PA	Returns current destination if axis is moving, otherwise returns current commanded position.
_PR	Returns current incremental distance

Example - Absolute Position

PA 10000	Specify absolute position of 10,000 counts
AC 1000000	Acceleration of 1,000,000 counts/sec ^{2}
DC 1000000	Deceleration of 1,000,000 counts/sec ^{2}
SP 50000	Speeds of 50,000 counts/sec
BG	Begin motion

Independent Jogging

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

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

Note that the controller operates as a closed-loop position controller while in the jog mode. The DMC-141X 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.

Command Summary - Jogging

JG +/- n	Specifies jog speed and direction
AC n	Specifies acceleration rate
DC n	Specifies deceleration rate
BG	Begins motion
IT	Time constant for independent motion smoothing
ST	Stops motion
IP n	Increments position instantly

Operand Summary - Jogging

_AC	Return acceleration rate
_DC	Return deceleration rate
_SP	Return speed
_TV	returns the actual velocity of the axis (averaged over .25 sec)

Example - Jog in X only

Jog motor at 50000 count/s.

#A	
AC 20000	Specify acceleration as 20000 counts/sec ^{2}
DC 20000	Specify deceleration as 20000 counts/sec ^{2}
JG 50000	Specify speed and direction as 50000 counts/sec
BG	Begin motion
EN	

Electronic Gearing

This mode allows the main encoder axis to be electronically geared to the auxiliary encoder. The master may rotate in both directions and the geared axis will follow at the specified gear ratio. The gear ratio may be changed during motion.

GR specifies the gear ratio for the slave where the ratio may be a number between ±127.9999 with a fractional resolution of 0.0001. GR 0 turns off electronic gearing. A limit switch will also disable electronic gearing.

Electronic gearing allows the geared motor to perform a second independent 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.

Command Summary - Electronic Gearing

GR n	Sets gearing mode and gear ratio. 0 disables electronic gearing.
MR n	Trippoint for motion past assigned point in reverse direction.
MF n	Trippoint for motion past assigned point in forward direction.

Example - Electronic Gearing

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

GR 1.132	Specify gear ratio and enable gear mode

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

GR 2	Specify gear ratio for X axis to be 2

Electronic Cam

The electronic cam is a motion control mode that enables the periodic synchronization of the motor with the auxiliary encoder that is the master.

The electronic cam is a more general type of electronic gearing that allows a table-based relationship between the motor and master.

To illustrate the procedure of setting the cam mode, consider the cam relationship shown in Figure 6.1.

Step 1. Specify the master cycle and the change in the slave axis

In the electronic cam mode, the position of the master is always expressed within one cycle. In this example, the position of the master is always expressed in the range between 0 and 6000. Similarly, the slave position is also redefined such that it starts at zero and ends at 1500. At the end of a cycle when the master is 6000 and the slave is 1500, the positions of both the aux encoder and the x axis are defined to zero. To specify the master cycle and the slave cycle change, we use the instruction EM.

```
EM n,m
```

where n specifies the cycle of the slave axis, and m specifies the cycle of the master aux encoder.

The cycle of the master is limited to 8,388,607 whereas the slave change per cycle is limited to 2,147,483,647. If the change is a negative number, the absolute value is specified. For the given example, the cycle of the master is 6000 counts and the change in the slave is 1500. Therefore, we use the instruction:

```
EM 1500,6000
```

Step 2. Specify the master interval and starting point.

Next we need to construct the ECAM table. The table is specified at uniform intervals of master positions. Up to 256 intervals are allowed. The size of the master interval and the starting point are specified by the instruction:

```
EP m,n
```

where m is the interval width in counts, and n is the starting point.

For the given example, we can specify the table by specifying the position at the master points of 0, 2000, 4000 and 6000. We can specify that by

```
EP 2000,0
```

Step 3. Specify the slave positions.

Next, we specify the slave positions with the instruction

```
ET[n]=x
```

where n indicates the order of the point.

The value, n, starts at zero and may go up to 256. The parameter x indicate the corresponding slave position. For this example, the table may be specified by

```
ET[0]=0
ET[1]=3000
ET[2]=2250
ET[3]=1500
```

This specifies the ECAM table.

Step 4. Enable the ECAM

To enable the ECAM mode, use the command

```
EB n
```

where n=1 enables ECAM mode and n=0 disables ECAM mode.

Step 5. Engage the slave motion

To engage the slave motion, use the instruction

```
EG n
```

where n is the master position at which the slave must be engaged.

If the value of any parameter is outside the range of one cycle, the cam engages immediately. When the cam is engaged, the slave position is redefined, modulo one cycle.

Step 6. Disengage the slave motion

To disengage the cam, use the command

EQ n

where n is the master position at which the slave axis 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 only be used when the master moves forward

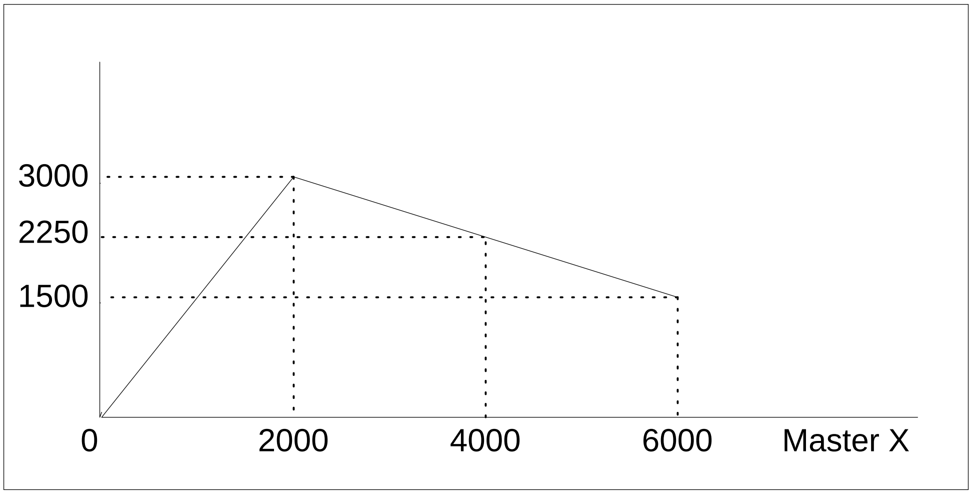


Figure 6.1: Electronic Cam Example

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 cycle of the master is 2000. Over that cycle, X varies by 1000. This leads to the instruction EM 1000,2000.

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 X according to the equation and assigns the values to the table with the instruction ET[N] = X.

Instruction	Interpretation
#SETUP	Label
EM 1000,2000	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
X = N *10+S	Define slave position
ET [N] = X	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: Master Aux. Encoder = 1000 and X = 500. This implies that X must be driven to that point to avoid a jump.

This is done with the program:

Instruction	Interpretation
#RUN	Label
EB1	Enable cam
PA500	starting position
SP5000	speed
BGX	Move motor
AMX	After moved
AI1	Wait for start signal
EG 1000	Engage slave
AI - 1	Wait for stop signal
EQ 1000	Disengage slave
EN	End

Contour Mode

The DMC-141X also provides a contouring mode. This mode allows any arbitrary position curve for the axis to be prescribed which is ideal for following computer generated paths or user-defined profiles.

Specifying Contour Segments

The Contour Mode (CM) command specifies the contour mode. The contour is described by position increments, CD n over a time interval, DT n.

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. 6.3. 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:

Instruction	Interpretation
#A	
CM	Specifies contour mode
DT 2	Specifies first time interval, 2^2
CD 48;WC	Specifies first position increment
DT 3	Specifies second time interval, 2^3
CD 90;WC	Specifies second position increment
DT 4	Specifies the third time interval, 2^4
CD 164;WC	Specifies the third position increment
DT0;CD0	Exits contour mode
EN	

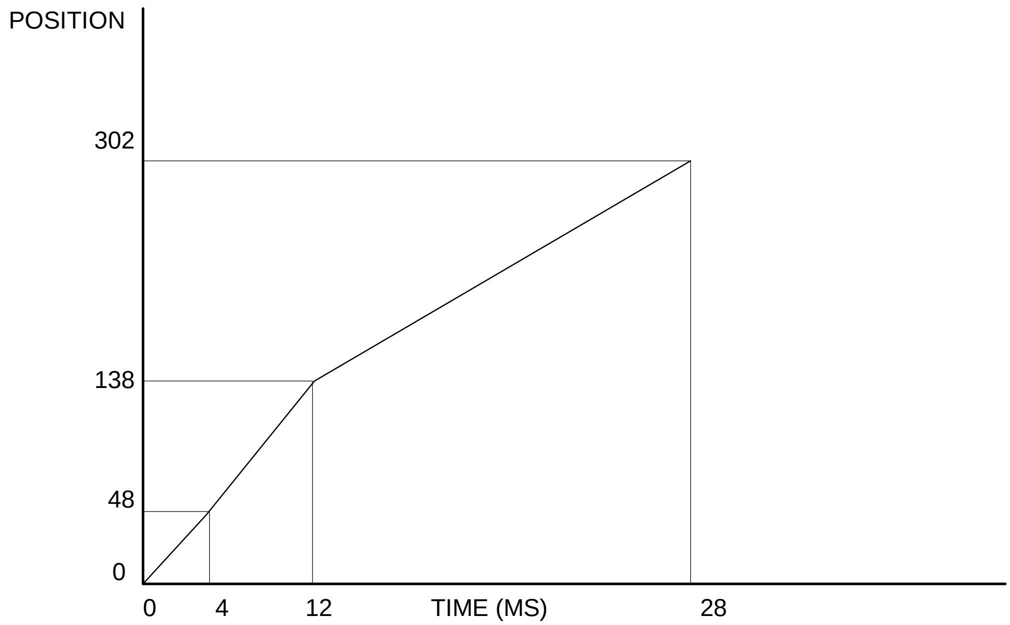


Figure 6.3 - The Required Trajectory

Additional Commands

The command, WC, is used as a trippoint "When Complete". This allows the DMC-141X 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 ?.

Command Summary - Contour Mode

CM	Specifies contouring mode.
CD n	Specifies position increment over time interval. Range is ±32,000. Zero ends contour mode.
DT n	Specifies time interval 2^n msec for position increment, where n is an integer between 1 and 8. Zero ends contour mode. If n does not change, it does not need to be specified with each CD.
WC	Waits for previous time interval to be complete before next data record is processed.

General Velocity Profiles

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

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

Generating an Array - An Example

Consider for example the velocity and position profiles shown in Fig. 6.4. The objective is to rotate a motor a distance of 6000 counts in 120 ms. The velocity profile uses sinusoidal acceleration 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:

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

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

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

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

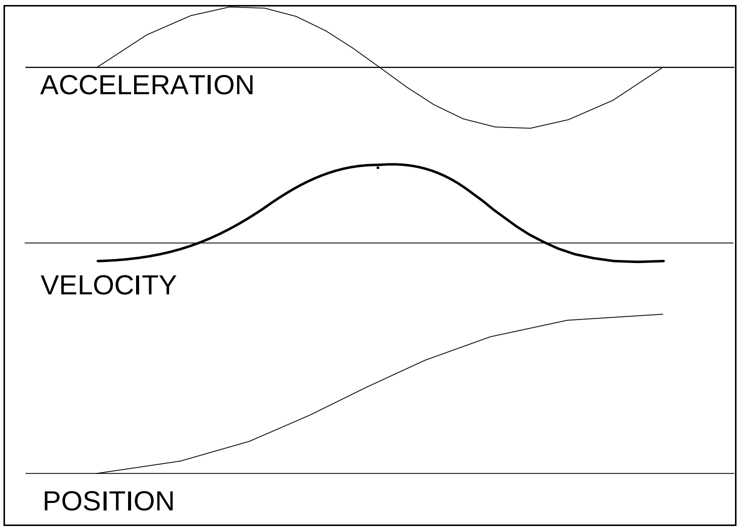


Figure 6.4 - Velocity Profile with Sinusoidal Acceleration

The DMC-141X can compute trigonometric functions. However, the argument must be expressed in degrees. Accordingly, the equation of X is written as:

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

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

Contour Mode Example

Instruction	Interpretation
#POINTS	Program defines X points
DM POS[16]	Allocate memory
DM DIF[15]	
C=0	Set initial conditions, C is index
T=0	T is time in ms
#A	
V1=50*T	
V2=3*T	Argument in degrees
V3=-955*@SIN[V2]+V1	Compute position
V4=@INT[V3]	Integer value of V3

POS[C]=V4	Store in array POS
T=T+8	
C=C+1	
JP #A,C<16	
#B	Program to find position differences
C=0	
#C	
D=C+1	
DIF[C]=POS[D]-POS[C]	Compute the difference and store
C=C+1	
JP #C,C<15	
EN	End first program
#RUN	Program to run motor
CM	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-141X 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:

Command Summary - Teach Mode

DM C[n]	Dimension array
RA C[]	Specify array for automatic record
RD _TP	Specify data for capturing
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:

Instruction	**Interpretation**
#RECORD	Begin Program
DM POS[501]	Dimension array with 501 elements
RA POS[]	Specify automatic record
RD _TP	Specify position to be captured
MO	Turn motor off

```
RC2                     Begin recording; 4 msec interval
#A;JP#A,_RC=1           Continue until done recording
#COMPUTE                Compute D
DM DX[500]              Dimension Array for D
C=0                     Initialize counter
#L                      Label
D=C+1
DELTA=POS[D]-POS[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
SHA                     Hold position
CM                      Specify contour mode
DT2                     Specify time increment
I=0                     Initialize array counter
#B                      Loop counter
CD DX[I];WC             Specify contour data; Wait until contour completes
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.

Stepper Motor Operation

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

Specifying Stepper Motor Operation

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

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

2 specifies a stepper motor with active low step output pulses

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

2.5 specifies a stepper motor with active low step output pulses and reversed direction

-2.5 specifies a stepper motor with active high step output pulse and reversed direction

Stepper Motor Smoothing

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

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

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

Monitoring Generated Pulses vs. Commanded Pulses

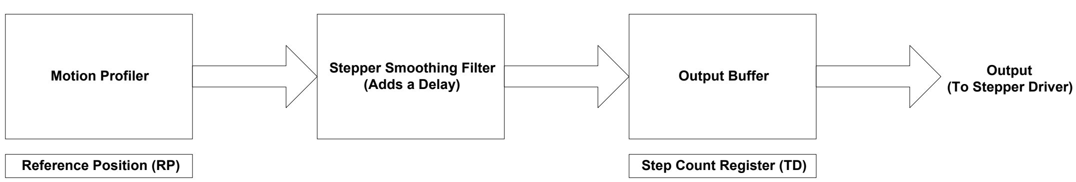
For proper controller operation, it is necessary to make sure that the controller has completed generating all step pulses before making additional moves. This is most particularly important if you are moving back and forth. For example, when operating with servo motors, the trippoint AM (After Motion)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

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

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

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

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



Motion Complete Trippoint

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

Using an Encoder with Stepper Motors

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

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

Command Summary - Stepper Motor Operation

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

Operand Summary - Stepper Motor Operation

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

Dual Loop (Auxiliary Encoder)

The DMC-141X provides an interface for a second encoder except when the controller is configured for stepper motor operation. When used, the second encoder is typically mounted on the motor or the load, but may be mounted in any position. The most common use for the second encoder is backlash compensation, described below.

The 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. The main and auxiliary encoders are configured with the CE command. The command form is CE x where x equals the sum of n and m below.

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 is

CE 6

Additional Commands for the Auxiliary Encoder

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

DEO

sets the initial value.

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

DE ?

returns the value of the auxiliary encoder.

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

V1=_DE

The current position of the auxiliary encoder may also be interrogated with the TD command.

Backlash Compensation

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

1. Continuous dual loop
2. Sampled dual loop

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

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

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

Continuous 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.

Note: It is recommended that the resolution of the rotary encoder be greater than the effective resolution of the load encoder for stability.

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

DV 1

activates the dual loop for the four axes and

DV 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

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

Connect the rotary encoder to the main encoders input and connect the linear encoder to the auxiliary encoder input. 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
BG	Start motion
#Correct	Correction loop
AM	Wait for motion completion
V1=10000-_DE	Find linear encoder error
V2=-_TE/4+V1	Compensate for motor error
JP#END,@ABS[V2]<2	Exit if error is small
PR V2*4	Correction move
BG	Start correction
JP#Correct	Repeat
#END	
EN	

Motion Smoothing

The DMC-141X controller allows the smoothing of the velocity profile to reduce the mechanical vibration of the system.

Trapezoidal velocity profiles have acceleration rates which change abruptly from zero to maximum value. The discontinuous acceleration results in 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.

Using the IT Command:

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

IT n	Independent time constant

It is used for smoothing profiled moves of the type JG, PR, and PA.

The smoothing parameter, n, is a number between 0 and 1 and determines 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 example illustrates the effect of the smoothing. Fig. 6.5 shows the trapezoidal velocity profile and the modified acceleration and velocity.

Note that the smoothing process results in longer motion time.

Example - Smoothing

Instruction	Interpretation
PR 20000	Position
AC 100000	Acceleration
DC 100000	Deceleration
SP 5000	Speed
IT .5	Filter for Smoothing
BG	Begin

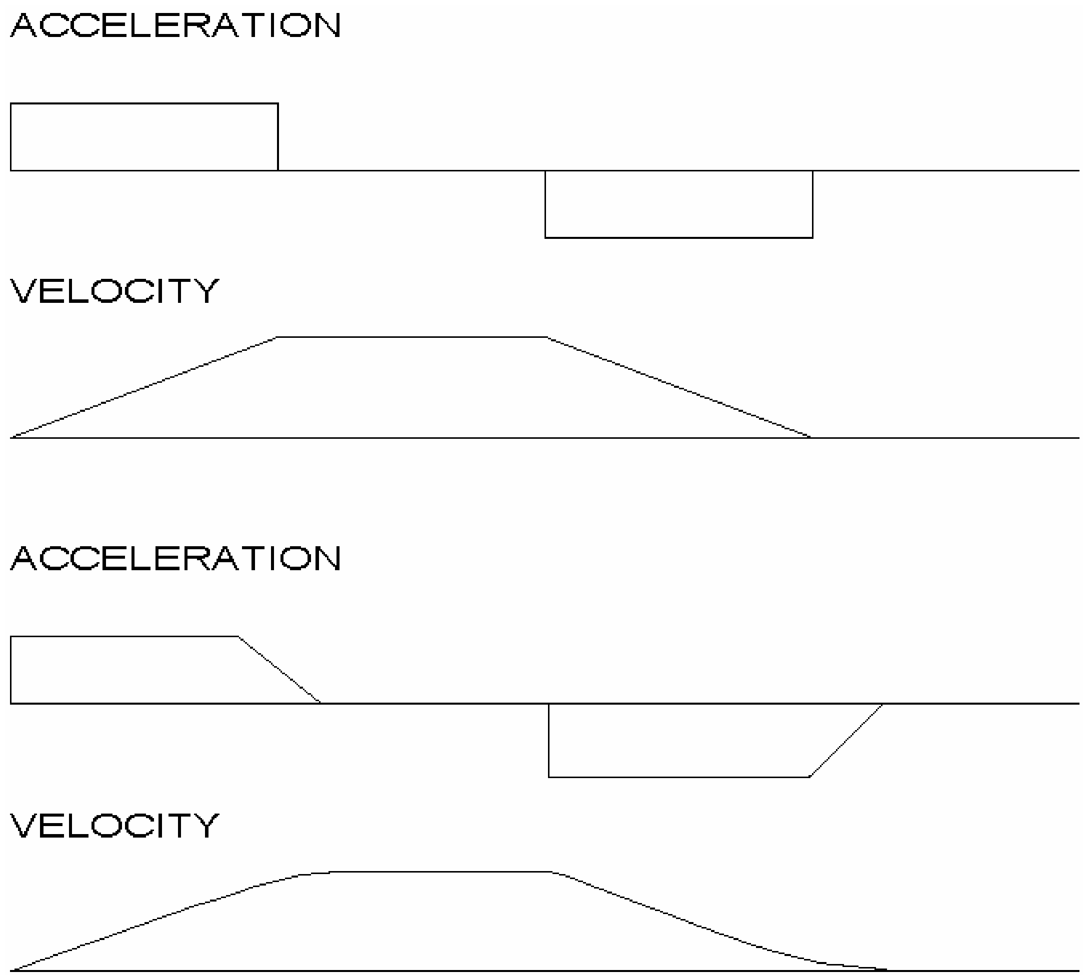


Figure 6.5 - Trapezoidal velocity and smooth velocity profiles

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 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.

Stage 1:

Upon begin, the motor accelerates to the slew speed specified by the JG or SP commands. The direction of its motion is determined by the state of the homing input. If _HMX reads 1 initially, the motor will go in the reverse direction first (direction of decreasing encoder counts). If _HMX reads 0 initially, the motor will go in the forward direction first. CN is the command used to define the polarity of the home input. With CN,-1 (the default value) a normally open switch will make _HMX read 1 initially, and a normally closed switch will make _HMX read zero. Furthermore, with CN,1 a normally open switch will make _HMX read 0 initially, and a normally closed switch will make _HMX read 1. Therefore, the CN command will need to be configured properly to ensure the correct direction of motion in the home sequence.

Upon detecting the home switch changing state, the motor begins decelerating to a stop.

Note: The direction of motion for the FE command also follows these rules for the state of the home input.

Stage 2:

The motor then traverses at 256 counts/sec in the opposite direction of Stage 1 until the home switch toggles again. If Stage 3 is in the opposite direction of Stage 2, the motor will stop immediately at this point and change direction. If Stage 2 is in the same direction as Stage 3, the motor will never stop, but will smoothly continue into Stage 3.

Stage 3:

The motor traverses forward at 256 counts/sec until the encoder index pulse is detected. The motor then stops immediately.

The DMC-141X defines the home position as the position at which the index was detected and sets the encoder reading at this point to zero.

The 4 different motion possibilities for the home sequence are shown in the following table.

			Direction of Motion		
Switch Type	CN Setting	Initial _HMX state	Stage 1	Stage 2	Stage 3
Normally Open	CN,-1	1	Reverse	Forward	Forward
Normally Open	CN,1	0	Forward	Reverse	Forward
Normally Closed	CN,-1	0	Forward	Reverse	Forward
Normally Closed	CN,1	1	Reverse	Forward	Forward

Example: Homing

Instruction	Interpretation
#HOME	Label
CN,-1	Configure the polarity of the home input
AC 1000000	Acceleration Rate
DC 1000000	Deceleration Rate
SP 5000	Speed for Home Search
HM	Home
BG	Begin Motion
AM	After Complete
MG "AT HOME"	Send Message
EN	End

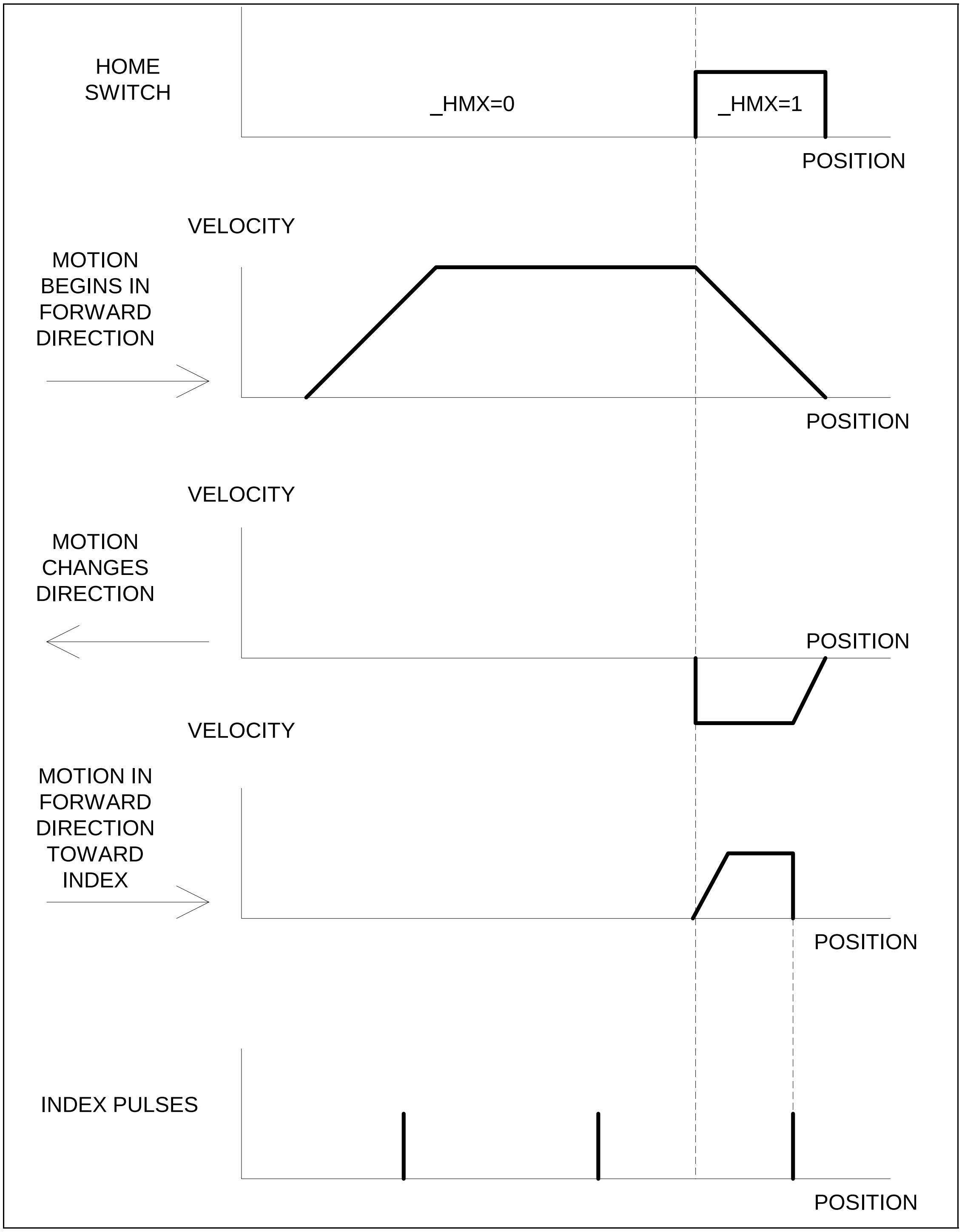


Figure 6.6 – Homing Sequence for Normally Closed Switch and CN,-1

Example: Find Edge

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

High Speed Position Capture

Often it is desirable to capture the position precisely for registration applications. The DMC-141X provides a position latch feature. This feature allows the position to be captured in less than 1 μsec of the external low or high input signal.

The DMC-141X 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 command, to arm the latch.
2. Test to see if the latch has occurred (Input 1 goes low) by using the _AL command. Example, V1=_AL returns the state of the latch into V1. V1 is 1 if the latch has not occurred.
3. After the latch has occurred, read the captured position with the report latch RL command or _RL.

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

Example: High Speed Latch

Instruction	**Interpretation**
#Latch	Latch program
JG 5000	Jog
BG	Begin
AL	Arm Latch
#Wait	Loop for Latch=1
JP #Wait,_AL=1	Wait for latch
Result=_RL	Report position
Result=	Print result
EN	End

THIS PAGE LEFT BLANK INTENTIONALLY

Chapter 7 Application Programming

Introduction

The DMC-141X provides a powerful programming language that allows users to customize the controller for their particular application. Programs can be downloaded into the DMC-141X memory freeing 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-141X provides several commands that allow the DMC-141X 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-141X provides 126 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. The program memory size is 250 lines X 40 characters.

Using the DMC-141X Editor to Enter Programs

Application programs for the DMC-141X may be created and edited either locally using the DMC-141X editor or remotely using another editor and then downloading the program into the controller. (Galil's Terminal and WSDK-software provides an editor UPLOAD and DOWNLOAD utilities).

The DMC-141X 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

NOTE: The ED command only accepts a parameter (e.g., #BEGIN) in a DOS Window. For general purposes, the editing features described in this section are not applicable when not in DOS mode.

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 40 characters per line.

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.

<ctrl>P

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

<ctrl>I

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

<ctrl>D

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

<ctrl>Q

The <ctrl>Q quits the editor mode. In response, the DMC-141X 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 through 9
:LS #A,9	List line label #A through line 9

Program Format

A DMC-141X program consists of several DMC-141X 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-141X 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 40 characters. A carriage return enters the final command on a program line.

Using Labels in Programs

All DMC-141X 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

Valid labels

#BEGIN

#SQUARE

#X1

#BEGIN1

Invalid labels

#1Square

#123

Example Program:

Instruction	Interpretation
#START	Beginning of the Program
PR 10000	Specify relative distances
BG	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 the motor 10,000 counts. After the motion is complete, the motor rests for 2 seconds. The cycle repeats indefinitely until the stop command is issued.

Special Labels

The DMC-141X also has some special labels, which are used to define input interrupt subroutines, limit switch subroutines, error handling subroutines, and command error subroutines. The following table lists the automatic subroutines supported by the controller. Sample programs for these subroutines can be found in the section *Automatic Subroutines for Monitoring Conditions.*

#AUTO	Starts Program on power-up or Reset
#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
#COMINT	Label for communication interrupt

Commenting Programs

Using the Command, NO

The DMC-141X provides a command, NO, for commenting programs. This command allows the user to include up to 38 characters on a single line after the NO command and can be used to include comments from the programmer as in the following example:

```
#MOVE
NO ABSOLUTE POINT TO POINT MOVE
NO SPEED 10000 COUNTS/SECOND
SP 10000
NO ACCELERATION 100000 COUNTS/SEC^2
AC 100000
NO DECELERATION 100000 COUNTS/SEC^2
DC 100000
NO MOVE TO ABSOLUTE POSITION 150000
PA 150000
NO BEGIN MOVE
BG
NO AFTER MOVE COMPLETES
AM
NO MOVE TO ABSOLUTE POSITION 0
PA 0
NO BEGIN MOVE
BG
NO AFTER MOVE
AM
NO END PROGRAM
EN
```

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

Using REM Statements with the Galil Terminal Software.

If you are using Galil software to communicate with the DMC-141x controller, you may also include REM statements. 'REM' statements begin with the word 'REM' and may be followed by any comments which are on the same line. The Galil terminal software will remove these statements when the program is downloaded to the controller. For example:

```
#PATH
PA 10000
REM SIMPLE MOVE
SP 10000
REM SPEED IS 10000
AC 100000
REM ACCELERATION IS 100000
DC 100000
REM DECELERATION IS 100000
BG
```

```
REM BEGIN MOTION
AM
REM WAIT FOR AFTER MOTION
EN
REM END OF PROGRAM
```

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

Executing Programs - Multitasking

The DMC-141X can run up to two programs simultaneously. The programs, called threads, are numbered 0 and 1, where 0 is the main thread.

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

1. Only the main thread may use the input command, IN. Note: This is NOT the @IN used to check general input status.
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.

Instruction	Interpretation
#TASK1	Task1 label
AT0	Initialize reference time
CB1	Clear Output 1
#LOOP1	Loop1 label
AT 10	Wait 10 msec from reference time
SB1	Set Output 1
AT –40	Wait 40 msec from reference time, then initialize reference
CB1	Clear Output 1
JP #LOOP1	Repeat Loop1
#TASK2	Task2 label
XQ #TASK1,1	Execute Task1
#LOOP2	Loop2 label
PR 1000	Define relative distance
BGX	Begin motion
AMX	After motion done
WT 10	Wait 10 msec

JP #LOOP2,@IN[2]=1	Repeat motion unless Input 2 is low
HX	Halt all tasks

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

Debugging Programs

The DMC-141X provides commands and operands which are useful in debugging application programs. These commands include interrogation commands to monitor program execution, determine the state of the controller and the contents of the controllers program, array, and variable space. Operands also contain important status information which can help to debug a program.

Trace Commands

The trace command causes the controller to send each line in a program to the host computer immediately prior to execution. Tracing is enabled with the command, TR1. TR0 turns the trace function off. Note: When the trace function is enabled, the line numbers as well as the command line will be displayed as each command line is executed.

Data which is output from the controller is stored in an output FIFO buffer. The output FIFO buffer can store up to 512 characters of information. In normal operation, the controller places output into the FIFO buffer. The software on the host computer monitors this buffer and reads information as needed. When the trace mode is enabled, the controller will send information to the FIFO buffer at a very high rate. In general, the FIFO will become full since the software is unable to read the information fast enough. When the FIFO becomes full, program execution will be delayed until it is cleared. If the user wants to avoid this delay, the command CW,1 can be given. This command causes the controller to throw away the data which can not be placed into the FIFO. In this case, the controller does not delay program execution.

Error Code Command

When there is a program error, the DMC-141X halts the program execution at the point where the error occurs. To display the last line number of program execution, issue the command, MG _ED.

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

Stop Code Command

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

RAM Memory Interrogation Commands

For debugging the status of the program memory, array memory, or variable memory, the DMC-141X has several useful commands. The command, DM ?, will return the number of array elements currently available. The command, DA ?, will return the number of arrays which can be currently defined. For example, a standard DMC-141X controller will have a maximum of 1000 array elements in up to 6 arrays. If an array of 100 elements is defined, the command DM ? will return the value 900 and the command DA ? will return 5.

To list the contents of the variable space, use the interrogation command LV (List Variables). To list the contents of array space, use the interrogation command, LA (List Arrays). To list the contents of the Program space, use the interrogation command, LS (List). To list the application program labels only, use the interrogation command, LL (List Labels).

Operands

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

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

_DL contains the number of available labels (126 max.)

_UL contains the number of available variables (126 max.)

_DA contains the number of available arrays (6 max.)

_DM contains the number of available array elements (1000 max.)

_AB contains the state of the Abort Input

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

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

Debugging Example:

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

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

Program Flow Commands

The DMC-141X provides several instructions that control program flow. The DMC-141X sequencer normally executes program instructions sequentially. The program flow can be altered with the use of event triggers, trippoints and conditional jump statements.

Command Summary - Program Flow

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 sets timeout for in-position)
WC	Wait for Contour Data
WT	Wait for time to elapse

Event Triggers & Trippoints

To function independently from the host computer, the DMC-141X 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-141X 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-141X can make decisions based on its own status or external events without intervention from a host computer.

DMC-141X Event Triggers

Command	Function
AM	Halts program execution until the profiled motion is complete.
AD n	Halts program execution until position command has reached the specified relative distance from the start of the move.
AR n	Halts program execution until after specified distance from the last AR or AD command has elapsed.
AP n	Halts program execution until after absolute position occurs.

MF n	Halt program execution until after forward motion reached absolute position. If position is already past the point, then MF will trip immediately. Will function on geared axis or aux. inputs.
MR n	Halt program execution until after reverse motion reached absolute position. If position is already past the point, then MR will trip immediately. Will function on geared axis or aux. inputs.
MC n	Halt program execution until after the motion profile has been completed and the encoder has entered or passed the specified position. TW sets timeout to declare an error if not in position. If timeout occurs, then the trippoint will clear and the stop code 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 7.
AS n	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.
WT n	Halts program execution until specified time in msec has elapsed.

Event Trigger Examples:

Event Trigger - Multiple Move Sequence

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

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

Event Trigger - Set Output after Distance

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

Instruction	**Interpretation**
#SETBIT	Label
SP 10000	Speed is 10000
PA 20000	Specify Absolute position
BG	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 10,000 counts during a move, the AR trippoint is used as shown in the next example.

Instruction	Interpretation
#TRIP	Label
JG 50000	Specify Jog Speed
BG;n=0	Begin Motion
#REPEAT	# Repeat Loop
AR 10000	Wait 10000 counts
TP	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
ST	Stop
EN	End

Event Trigger - Start Motion on Input

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

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

Event Trigger - Set output when At speed

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

Event Trigger - Multiple move with wait

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

Instruction	Interpretation
#OUTPUT	Program label
AT0	Initialize time reference
SB1	Set Output 1
#LOOP	Loop
AT 10	After 10 msec from reference,
CB1	Clear Output 1
AT -40	Wait 40 msec from reference and reset reference
SB1	Set Output 1
JP #LOOP	Loop
EN	

Conditional Jumps

The DMC-141X provides Conditional Jump (JP) and Conditional Jump to Subroutine (JS) instructions for branching to a new program location based on a specified condition. The conditional jump determines if a condition is satisfied and then branches to a new location or subroutine. Unlike event triggers, the conditional jump instruction does not halt the program sequence. Conditional jumps are useful for testing events in real-time. They allow the DMC-141X to make decisions without a host computer. For example, the DMC-141X can decide between two motion profiles based on the sate of an input line.

Command Format - JP and JS

Format:	Description
JS destination, logical condition	Jump to subroutine if logical condition is satisfied
JP destination, logical condition	Jump to location if logical condition is satisfied

The destination is a program line number or label where the program sequencer will jump if the specified condition is satisfied. Not that the line number of the first line of program memory is 0. The comma designates "IF". The logical condition tests two operands with logical operators.

Logical operators:

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

Conditional Statements

The conditional statement is satisfied if it evaluates to any value other than zero. The conditional statement can be any valid DMC-141X numeric operand, including variables, array elements, numeric values, functions, keywords, and arithmetic expressions. If no conditional statement is given, the jump will always occur.

Examples:

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

Multiple Conditional Statements

The DMC-141X will accept multiple conditions in a single jump statement. The conditional statements are combined in pairs using the operands “&” and “|”, representing the logical AND and logical OR respectively. The “&” operand between any two conditions, requires that both statements must be true for the combined statement to be true. The “|” operand between any two conditions, requires that only one statement be true for the combined statement to be true. *Note: Each condition must be placed in parentheses for proper evaluation by the controller. In addition, the DMC-141X executes operations from left to right. For further information on Mathematical Expressions and the bit-wise operators ‘&’ and ‘|’, see pg 7- 86.*

For example, using variables named V1, V2, V3 and V4:

JP #TEST, (V1<V2) & (V3<V4)

In this example, this statement will cause the program to jump to the label #TEST if V1 is less than V2 and V3 is less than V4. To illustrate this further, consider this same example with an additional condition:

JP #TEST, ((V1<V2) & (V3<V4)) | (V5<V6)

This statement will cause the program to jump to the label #TEST under two conditions; 1. If V1 is less than V2 and V3 is less than V4. <u>OR</u> 2. If V5 is less than V6.

Using the JP Command:

If the condition for the JP command is satisfied, the controller branches to the specified label or line number and continues executing commands from this point. If the condition is not satisfied, the controller continues to execute the next commands in sequence.

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

Example Using JP command:

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

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

Subroutines

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

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.

Auto-Start Routine

The DMC-1412 and DMC-1414 have a special label for automatic program execution. A program which has been saved into the controllers non-volatile memory can be automatically executed upon power up or reset by beginning the program with the label #AUTO. The program must be saved into non-volatile memory using the command, BP.

Automatic Subroutines for Monitoring Conditions

Often it is desirable to monitor certain conditions continuously without tying up the host or DMC-141X program sequences. The DMC-141X 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
#COMINT	Communication Interrupt (DMC-1412 and DMC-1414 only)

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

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

Example - Limit Switch

This program prints a message upon the occurrence of a limit switch. Note, for the #LIMSWI routine to function, the DMC-141X 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 JG5000 can still be sent from the PC even while the “dummy” applications program is being executed.

Instruction	Interpretation
#TEST	Test program
JG1000	Set jog speed on X axis
BG	Begin motion on the X axis
#LOOP	Dummy Program for endless loop
JP #LOOP;EN	Jump to #LOOP label
#LIMSWI	Limit Switch Label
MG "LIMIT OCCURRED"	Print Message
RE	Return to main program

Now, when a forward limit switch occurs, 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.

NOTE: The #LIMSWI routine is only executed when the motor is being commanded to move.

Example - Position Error

Instruction	Interpretation
#MAIN	Main program
JG10000	Set jog speed
BG	Begin jog
#LOOP	Dummy Program
JP #LOOP;EN	Loop
#POSERR	Position Error Routine
V1=_TE	Read Position Error
MG "EXCESS POSITION ERROR"	Print Message
MG "ERROR=",V1=	Print Error
RE	Return from Error

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).

Example - Input Interrupt

Instruction	Interpretation
#A	Label
II1	Input Interrupt on 1
JG 30000	Jog
BG	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
JG 30000	Restore Velocities
BG;RI	Begin motion and Return to Main Program
EN	

When Input 1 changes in state from high to low, the #ININT subroutine will be executed.

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

Example - Motion Complete Timeout

Instruction	Interpretation
#BEGIN	Begin main program
TW 1000	Set the time out to 1000 ms
PA 10000	Position Absolute command
BG	Begin motion

MC	Motion Complete trip point
EN	End main program
#MCTIME	Motion Complete Subroutine
MG “X Fell Short”	Send out a message
EN	End subroutine

This simple program will issue the message “X Fell Short” if the axis does not reach the commanded position within 1 second of the end of the profiled move.

Example - Command Error

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

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

Mathematical and Functional Expressions

Mathematical Operators

For manipulation of data, the DMC-141X provides the use of the following mathematical operators:

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

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

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

Examples:

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=_TP-(@COS[45]*40)	Puts the position - 28.28 in RESULT. 40 * cosine of 45° is 28.28
TEMP=@IN[1]&@IN[2]	TEMP is equal to 1 only if Input 1 and Input 2 are high

Bit-Wise Operators

The mathematical operators & and | are bit-wise operators. The operator, &, is a Logical And. The operator, |, is a Logical Or. These operators allow for bit-wise operations on any valid DMC-141X numeric operand, including variables, array elements, numeric values, functions, keywords, and arithmetic expressions. The bit-wise operators may also be used with strings. This is useful for separating characters from an input string. When using the input command for string input, the input variable will hold up to 6 characters. These characters are combined into a single value which is represented as 32 bits of integer and 16 bits of fraction. Each ASCII character is represented as one byte (8 bits), therefore the input variable can hold up to six characters. The first character of the string will be placed in the top byte of the variable and the last character will be placed in the lowest significant byte of the fraction. The characters can be individually separated by using bit-wise operations as illustrated in the following example:

#TEST	Begin main program
IN "ENTER",LEN{S6}	Input character string of up to 6 characters into variable ‘LEN’
FLEN=@FRAC[LEN]	Define variable ‘FLEN’ as fractional part of variable ‘LEN’
FLEN=$10000*FLEN	Shift FLEN by 32 bits (IE - convert fraction, FLEN, to integer)
LEN1=(FLEN&$00FF)	Mask top byte of FLEN and set this value to variable ‘LEN1’
LEN2=(FLEN&$FF00)/$100	Let variable, ‘LEN2’ = top byte of FLEN
LEN3=LEN&$000000FF	Let variable, ‘LEN3’ = bottom byte of LEN
LEN4=(LEN&$0000FF00)/$100	Let variable, ‘LEN4’ = second byte of LEN
LEN5=(LEN&$00FF0000)/$10000	Let variable, ‘LEN5’ = third byte of LEN
LEN6=(LEN&$FF000000)/$1000000	Let variable, ‘LEN6’ = fourth byte of LEN
MG LEN6 {S4}	Display ‘LEN6’ as string message of up to 4 chars
MG LEN5 {S4}	Display ‘LEN5’ as string message of up to 4 chars
MG LEN4 {S4}	Display ‘LEN4’ as string message of up to 4 chars
MG LEN3 {S4}	Display ‘LEN3’ as string message of up to 4 chars
MG LEN2 {S4}	Display ‘LEN2’ as string message of up to 4 chars
MG LEN1 {S4}	Display ‘LEN1’ as string message of up to 4 chars
EN	

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

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

T	Response from command MG LEN6 {S4}
E	Response from command MG LEN5 {S4}
S	Response from command MG LEN4 {S4}
T	Response from command MG LEN3 {S4}
M	Response from command MG LEN2 {S4}
E	Response from command MG LEN1 {S4}

Functions

Function	Description
@ABS[n]	Absolute Value of n
@SIN[n]	Sine of n (n in degrees, with range of -32768 to 32767 and 16-bit fractional resolution)
@COS[n]	Cosine of n (n in degrees, with range of -32768 to 32767 and 16-bit fractional resolution)
@COM[n]	1's Complement of n
@FRAC[n]	Fraction portion of n
@INT[n]	Integer portion of n
@RND[n]	Round of n (Rounds up if the fractional part of n is .5 or greater)
@IN[n]	Return digital input at general input n (where n starts at 1)
@AN[n]	Return analog input at general analog in n (where n starts at 1)
@SQR[n]	Square root of n (Accuracy is ±0.004)

Functions may be combined with mathematical expressions. The order of execution is from left to right.

Examples:

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

For applications that require a parameter that is a variable, the DMC-141X provides 126 variables. These variables can be numbers or strings. 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. For example, a cut-to-length application may require that a cut length be variable.

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-141X 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. Variable names should not be the same as DMC-141X instructions. For example, PR is not a good choice for a variable name.

Examples of valid and invalid variable names are:

Valid Variable Names

POSX

POS1

SPEEDZ

Invalid Variable Names

REALLONGNAME	; Cannot have more than 8 characters
123	; Cannot begin variable name with number
SPEED Z	; Cannot have spaces in the name

For the DMC-1412 and DMC-1414, the BV command will save array and variable-values upon power down.

Assigning Values to Variables:

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

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

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

Any valid DMC-141X function can be used to assign a value to a variable. For example, V1=@ABS[V2] or V2=@IN[1]. Arithmetic operations are also permitted.

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

Example:

POSX=_TP	Assigns returned value from TP 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

Assigning Variable Values to Controller Parameters

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

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

Displaying the Value of Variables at the Terminal

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

Operands

Operands allow motion or status parameters of the DMC-141X to be incorporated into programmable variables and expressions. Most DMC-141X commands have an equivalent operand - which are designated by adding an underscore (_) prior to the DMC-141X command. The command reference indicates which commands have an associated operand.

Status commands such as Tell Position return actual values, whereas action commands such as KP or SP return the values in the DMC-141X registers.

Examples of Operands

POSX=_TP	Assigns value from Tell Position to the variable POSX.
GAIN=_KP*2	Assigns value from KP multiplied by two to variable, GAIN.
JP #LOOP,_TE>5	Jump to #LOOP if the position error is greater than 5
JP #ERROR,_TC=1	Jump to #ERROR if the error code equals 1.

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

Special Operands (Keywords)

The DMC-141X also provides a few additional operands which give access to internal variables that are not accessible by standard DMC-141X commands.

KEYWORD	**FUNCTION**
_BG	*Returns a 1 if motion on axis is complete, otherwise returns 0.
_BN	*Returns serial # of the board.
_DA	*Returns the number of arrays available
_DL	*Returns the number of available labels for programming
_DM	*Returns the available array memory
_HM	*Returns status of Home Switch (equals 0 or 1)
_LF	Returns status of Forward Limit switch input of axis (equals 0 or 1)
_LRX	Returns status of Reverse Limit switch input of axis (equals 0 or 1)
_UL	*Returns the number of available variables
TIME	Free-Running Real Time Clock (off by 2.4% - Resets with power-on). Note: TIME does not use an underscore character (_) as other keywords.

* - These keywords have corresponding commands while the keywords _LF, _LR and TIME do not have any associated commands. All keywords are listed in the Command Summary, Chapter 11.

Examples of Keywords

V1=_LF	Assign V1 the logical state of the Forward Limit Switch
V3=TIME	Assign V3 the current value of the time clock
V4=_HM	Assign V4 the logical state of the Home input

Example Program:

Instruction	Interpretation
#TIMER	Timer
INITIME=TIME	Initialize time variable
PR50000;BG	Begin move
AM	After move
ELAPSED=TIME-INTIME	Compute elapsed time
EN	End program
#LIMSWI	Limit Switch Routine

JP #FORWARD,_LF=0	Jump if Forward Limit
AM	Wait for Motion Done
PR 1000;BG;AM	Move Away from Reverse Limit
JP #END	Exit
#FORWARD	Forward Label
PR -1000;BG;AM	Move Away from Forward Limit
#END	Exit
RE	Return to Main Program

Arrays

For storing and collecting numerical data, the DMC-141X provides array space for 1000 elements. The arrays are one dimensional and up to 6 different arrays may be defined. Each array element has a numeric range of 4 bytes of integer (2^{31}) followed by two bytes of fraction (±2,147,483,647.9999).

Arrays can be used to capture real-time data, such as position, torque and error values. In the contouring mode, arrays are convenient for holding the points of a position trajectory in a record and playback application.

Defining Arrays

An array is defined with the command DM. The user must specify a name and the number of entries to be held in the array. An array name can contain up to eight characters, starting with an uppercase alphabetic character. The number of entries in the defined array is enclosed in [].

Example:

DM POSX[7]	Defines an array names POSX with seven entries
DM SPEED[100]	Defines an array named speed with 100 entries
DM POSX[0]	Frees array space

Assignment of Array Entries

Like variables, each array element can be assigned a value. Assigned values can be numbers or returned values from instructions, functions and keywords.

Array elements are addressed starting at count 0. For example the first element in the POSX array (defined with the DM command, DM POSX[7]) would be specified as POSX[0].

Values are assigned to array entries using the equal sign. Assignments are made one element at a time by specifying the element number with the associated array name.

NOTE: Arrays must be defined using the command, DM, before assigning entry values.

Examples:

DM SPEED[10]	Dimension Speed Array
SPEED[1]=7650.2	Assigns the first element of the array, SPEED the value 7650.2
SPEED[1]=	Report array element value
POSX[10]=_TP	Assigns the 10th element of the array POS the returned value from the tell position command.
CON[2]=@COS[POS]*2	Assigns the second element of the array CON the cosine of the variable POS multiplied by 2.
TIMER[1]=TIME	Assigns the first element of the array timer the returned value of the TIME keyword.

Using a Variable to Address Array Elements

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

For example:

Instruction	**Interpretation**
#A	Begin Program
COUNT=0;DM POS[10]	Initialize counter and define array
#LOOP	Begin loop
WT 10	Wait 10 msec
POS[COUNT]=_TP	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.

Uploading and Downloading Arrays to On Board Memory

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

QU array[],start,end,delim

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)

Delim specifies whether the array data is separated by a comma (delim=1) or a carriage return (delim=0).

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

Automatic Data Capture into Arrays

The DMC-141X 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. Two types of data can be captured and stored in two arrays. The capture rate or time interval may be specified. Recording can be done as a one time event or as a circular continuous recording.

Commands Summary - Automatic Data Capture

Command	**Description**
RA n[],m[]	Selects up to two arrays for data capture. The arrays must have been defined with the DM command.
RD type1, type2	Selects the type of data to be recorded, where type1 and type2 represent the various types of data (see table below). The order of data type is important and corresponds with the order of n,m arrays in the RA command.

RC n,m	The RC command begins data collection. Sets data capture time interval where n is an integer between 1 and 8 and designates 2^n msec between data. m is optional and specifies the number of elements to be captured. If m is not defined, the number of elements defaults to the smallest array defined by DM. When m is a negative number, the recording is done continuously in a circular manner. _RD is the recording pointer and indicates the address of the next array element. n=0 stops recording.
RC?	Returns a 0 or 1 where, 0 denotes not recording, 1 specifies recording in progress.

Data Types for Recording

Data Type	Description
_DE	2nd encoder position (dual encoder)
_TP	Encoder position
_TE	Position error
_SH	Commanded position
_RL	Latched position
_TI	Inputs
_OP	Output
_TS	Switches (only bit 0-4 valid)
_SC	Stop code
_NO	Status bits
_TT	Torque

Operand Summary - Automatic Data Capture

_RC	Returns a 0 or 1 where 0 denotes not recording, 1 denotes recording in progress.
_RD	Returns address of next array element.

Example - Recording into An Array

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

Instruction	Interpretation
#RECORD	Begin program
DM XPOS[300]	Define position array
DM XERR[300]	Define error array
RA XPOS[],XERR[]	Select arrays for capture
RD _TP, _TE	Select data types
PR 10000	Specify move distance
RC1	Start recording now, at rate of 2 msec
BG	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
XPOS[N]=	Print position

XERR[N]=	Print error
N=N+1	Increment Counter
#DONE	Done
EN	End Program

Deallocating Array Space

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

Input of Data (Numeric and String)

Input of Data

The command, IN, is used to prompt the user to input numeric or string data. Using the IN command, the user may specify a message prompt by placing a message in quotations. When the controller executes an IN command, the controller will wait for the input of data. The input data is assigned to the specified variable or array element.

An Example for Inputting Numeric Data

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

In this example, the message "Enter Length" is displayed on the computer screen. The controller waits for the operator to enter a value. The operator enters the numeric value which is assigned to the variable, LENX.

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.

Instruction	Interpretation
#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
BG	Begin motion to move material
AM	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

Operator Data Entry Mode

The Operator Data Entry Mode provides for unbuffered data entry through the main RS-232 port. In this mode, the input will not be interpreted as DMC commands. For example, input such as ST or JG will not be recognized as commands. In this mode, the DMC-141X provides a buffer for receiving characters. This mode may only be used when executing an applications program.

The Operator Data Entry Mode may be specified for either Port 1 or Port 2 or both. The mode may be exited with the \ or <escape> key.

NOTE: Operator Data Entry Mode cannot be used for high rate data transfer.

For Port 1:	Use the third field of the CI command to set the Data Mode. A 1 specifies Operator Data Mode, a 0 disables the Data Mode.
For Port 2:	Use the third field of the CC command to set the Data Mode. A 0 configures P2 as a general port for the Operator Data Mode.

To capture and decode characters in the Operator Data Mode, the DMC-141X provides the following special keywords:

Port 1 (Main) Keyword	Port 2 (Aux.) Keyword	Function
P1CH	P2CH	Contains the last character received
P1ST	P2ST	Contains the received string
P1NM	P2NM	Contains the received number
P1CD	P2CD	Contains the status code: -1 Mode Disabled 0 Nothing received 1 Received character, but not <enter> 2 Received string, not a number 3 Received number

Note: The value of P1CD and P2CD returns to zero after the corresponding string or number is read.

These keywords may be used in an applications program to decode data. They may be used in conditional statements with logical operators.

Examples:

JP #LOOP,P2CD< >3	Checks to see if status code is 3 (number received)
JP #P,P1CH="V"	Checks if last character received was a V
PR P2NM	Assigns received number to position
JS #XAXIS,P1ST="X"	Checks to see if received string is X

Using Communications Interrupt

The DMC-141X provides a special interrupt for communication allowing the application program to be interrupted by input from the user. The interrupt is enabled using the CI command. The syntax for the command is CI m,n,o:

m=0	Don't interrupt Port 1
1	Interrupt on <enter> Port 1

2	Interrupt on any character Port 1
-1	Clear any characters in buffer
n=0	Don't interrupt Port 2
1	Interrupt on <enter> Port 2
2	Interrupt on any character Port 2
-1	Clear any characters in buffer
o=0	Disable operator data mode for P1
1	Enable operator data mode for P1

The #COMINT label is used for the communication interrupt. For example, the DMC-141X can be configured to interrupt on any character received on Port 2. The #COMINT subroutine is entered when a character is received and the subroutine can decode the characters. At the end of the routine the EN command is used. EN,1 will re-enable the interrupt and return to the line of the program where the interrupt was called, EN will just return to the line of the program where it was called without re-enabling the interrupt. As with any automatic subroutine, a program must be running in thread 0 at all times for it to be enabled.

Example - Using the #COMINT Routine

A DMC-1412 is used to jog the axis. The speed of the axis may be changed during motion by specifying the new speed value. An S stops motion.

Command	**Interpretation**
#AUTO	Label for Auto Execute
SPEEDX=10000	Initial X speed
CI ,2	Set Port 2 for Character Interrupt
JG SPEEDX	Specify jog mode speed for X axis
BGX	Begin motion
#PRINT	Routine to print message to terminal
MG{P2}"TO CHANGE SPEEDS"	Print message
MG{P2}"TYPE X"	
MG{P2}"TYPE S TO STOP"	
#JOGLOOP	Loop to change Jog speeds
JG SPEEDX	Set new jog speed
JP #JOGLOOP	
EN	End of main program
#COMINT	Interrupt routine
CI,0	Clear interrupt
JP #A,P2CH="X"	Check for X
JP #B,P2CH="S"	Check for S
ZS1;CI,2;JP#JOGLOOP	Jump if not X,Y,S
#A;JS#NUM	
SPEEDX=VAL	New X speed
ZS1;CI,2;JP#PRINT	Jump to Print
#B;ST;AMX;CI,-1	Stop motion on S
MG{P2}"THE END"	

ZS;EN,1	End-Re-enable interrupt
#NUM	Routine for entering new jog speed
MG "ENTER",P2CH{S},"AXIS SPEED" {N}	Prompt for value
#NUMLOOP; CI,-1	Check for enter
#NMLP	Routine to check input from terminal
JP #NMLP,P2CD<2	Jump to error if string
JP #ERROR,P2CD=2	Read value
VAL=P2NM	
EN	End subroutine
#ERROR;CI,-1	Error Routine
MG "INVALID-TRY AGAIN"	Error message
JP #NMLP	
EN	End

Inputting String Variables

String variables with up to six characters may be input using the specifier, {Sn} where n represents the number of string characters to be input. If n is not specified, six characters will be accepted. For example, IN “Enter X, Y or Z”,V{S} specifies a string variable to be input.

Output of Data (Numeric and String)

Numerical and string data can be output from the controller using several methods. The message command, MG, can output string and numerical data. Also, the controller can be commanded to return the values of variables and arrays, as well as other information using the interrogation commands (the interrogation commands are described in Chapter 5).

Sending Messages

Messages may be sent to the bus using the message command, MG. This command sends specified text and numerical or string data from variables or arrays to the screen.

Text strings are specified in quotes and variable or array data is designated by the name of the variable or array. For example:

```
MG “The Final Value is”,RESULT
```

In addition to variables, functions and commands, responses can be used in the message command. For example:

```
MG " Input 1 is", @IN[1]
MG "The Proportional Gain of X is", _KP
```

Formatting Messages

String variables can be formatted using the specifier, {Sn} where n is the number of characters, 1 through 6. For example:

```
MG STR {S3}
```

This 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. The actual numerical value will be formatted with n characters to the left of the decimal and m characters to the right of the decimal. Leading zeros will be used to display specified format. For example:

MG “The Final Value is”, RESULT{F5.2}

If the value of the variable RESULT is equal to 4.1, this statement returns the following:

The Final Value is 00004.10.

If the value of the variable RESULT is equal to 999999.999, the above message statement returns the following:

The Final Value is 99999.99

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;BG;AS
MG "The Speed is", _TV {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

Using the MG Command to Configure Terminals

The MG command can also be used to configure a terminal. Any ASCII character can be sent by using the format {^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:

Function	Description
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 through 6.
{P2}	Sends the message to auxiliary Serial Port (DMC-141X only)

Displaying Variables and Arrays

Variables may also be sent to the screen using the format, variable= **or** array[x]=. For example, V1= , returns the value of the variable V1.

Example - Printing a Variable and an array element

#DISPLAY	Label
PR 1000	Position Command
BG	Begin
AM	After Motion
V1=_TP	Assign Variable V1
V1=	Print V1
EN	End

Interrogation Commands

The DMC-141x has a set of commands that directly interrogate the controller. When these commands are entered, the requested data is returned in decimal format on the next line followed by a carriage return and line feed. The format of the returned data can be changed using the Position Format (PF), and Leading Zeros (LZ) command. For a complete description of interrogation commands, see chapter 5.

Using the PF Command to Format Response from Interrogation Commands

The command, PF, can change format of the values returned by theses interrogation commands:

BL ?	LE ?
DE ?	PA ?
DP ?	PR ?
EM ?	TN ?
FL ?	VE ?
IP ?	TE
TP	

The 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.

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.

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

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
:PF2	Format 2 places
:TPX	Tell Position
99	Returns 99 if position greater than 99

Removing Leading Zeros from Response to Interrogation Response

The leading zeros on data returned as a response to interrogation commands can be removed by the use of the command, LZ.

Example - Using the LZ command

LZ0	Disables the LZ function
TP	Tell Position Interrogation Command
-0000000009, 0000000005, 0000000000, 0000000007	Response from Interrogation Command (With Leading Zeros)
LZ1	Enables the LZ function
TP	Tell Position Interrogation Command
-9, 5, 0, 7	Response from Interrogation Command (Without Leading Zeros)

Local Formatting of Response of Interrogation Commands

The response of interrogation commands may be formatted locally. To format locally, use the command, {Fn.m} or {$n.m} on the same line as the interrogation command. The symbol F specifies that the response should be returned in decimal format 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. For example:

Examples:

TP {F2.2}	Tell Position in decimal format 2.2
-05.00, 05.00, 00.00, 07.00	Response from Interrogation Command
TP {$4.2}	Tell Position in hexadecimal format 4.2
FFFB.00,$0005.00,$0000.00,$0007.00	Response from Interrogation Command

Formatting Variables and Array Elements

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

Local Formatting of Variables

PF and VF commands are global format commands that affect the format of all relevant returned values and variables. Variables may also be formatted locally. To format locally, use the command, {Fn.m} or {$n.m} following the variable name and the '=' symbol. 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. For example:

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	

The local format is also used with the MG* command.

Converting to User Units

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

The DMC-141X position parameters such as PR and PA have units of quadrature counts. Speed parameters such as SP and JG have units of counts/sec. Acceleration parameters such as AC and DC have units of counts/sec^2. The controller interprets time in milliseconds.

All input parameters must be converted into these units. For example, an operator can be prompted to input a number in revolutions. A program could be used such that the input number is converted into counts by multiplying it by the number of counts/revolution.

Example:

Instruction	Interpretation
#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 Hardware I/O

Digital Outputs

The DMC-141X has a 3-bit uncommitted output port for controlling external events. Each bit on the output port may be set and cleared with the software instructions SB (Set Bit) and CB(Clear Bit), OB (define output bit) and OP (Output port).

For example:

Instruction	**Function**
SB2	Set bit 2 of output port
CB1	Clears bit 1 of output port
CB3	Clear bit 3 of output port

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.

The output port may also be written to as a 3-bit word using the instruction

OP (Output Port). This instruction allows a single command to define the state of the entire 3-bit output port, where 2^0 is output 1, 2^1 is output 2 and 2^2 is output 3. A 1 designates that bit is on. The value in the output port is the sum of bits 0, 1, and 2.

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

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

Example - Turn on Output After Move

Instruction	**Interpretation**
#OUTPUT	Label
PR 2000	Position Command
BG	Begin
AM	After move
SB1	Set Output 1
WT 1000	Wait 1000 msec
CB1	Clear Output 1
EN	End

Digital Inputs

The DMC-141X has seven digital inputs for controlling motion by local switches. The @IN[n] function returns the logic level of the specified input 1 through 7. 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-141X. High on input 1 means switch is in on position.

Instruction	Function
#S;JG 4000	Set speed
AI 1;BG	Begin after input 1 goes high
AI -1;ST	Stop after input 1 goes low
AM;JP #S	After motion, repeat
EN	

Input Interrupt Function

The DMC-141X provides an input interrupt function which causes the program to automatically execute the instructions following the #ININT label. This function is enabled using the II m,n,o command. The m specifies the beginning input and n specifies the final input in the range. The parameter o is an interrupt mask. If m and n are unused, o contains a number with the mask. A 1 designates that input to be enabled for an interrupt, where 2^0 is bit 1, 2^1 is bit 2 and so on. For example, II,,5 enables inputs 1 and 3 ($2^0 + 2^2 = 5$).

A low input on any of the specified inputs will cause automatic execution of the #ININT subroutine. The Return from Interrupt (RI) command is used to return from this subroutine to the place in the program where the interrupt had occurred. If it is desired to return to somewhere else in the program after the execution of the #ININT subroutine, the Zero Stack (ZS) command is used followed by unconditional jump statements.

IMPORTANT: Use the RI instruction (not EN) to return from the #ININT subroutine.

Examples - Input Interrupt

Instruction	Interpretation
#A	Label #A
II 1	Enable input 1 for interrupt function
JG 30000	Set speed
BG	Begin motion
#B	Label #B
TP	Report position

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	Stops motion
#LOOP;JP #LOOP,@IN[1]=0	Loop until Interrupt cleared
JG 15000	Specify new speeds
WT 300	Wait 300 milliseconds
BG	Begin motion
RI	Return from Interrupt subroutine

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:

$$4000/2\pi = 637 \text{ count/inch}$$

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
BG	Start Motion
AM	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

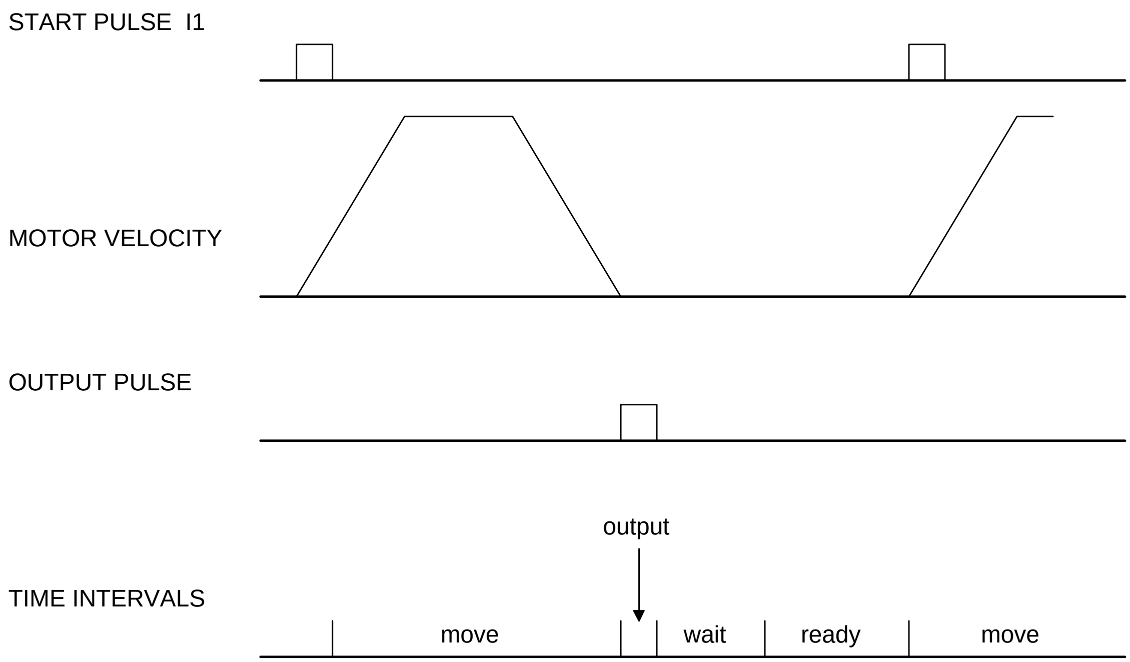


Figure 7.1 - Motor Velocity and the Associated Input/Output signals

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 lead screw. Such a lead screw 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	Interpretation
#A	Label
DP0	Define starting positions as zero
LINPOS=0	
PR 1000	Required distance
BG	Start motion
#B	
AM	Wait for completion
WT 50	Wait 50 msec
LIN POS = _DE	Read linear position
ER=1000-LINPOS-_TE	Find the correction
JP #C,@ABS[ER]<2	Exit if error is small
PR ER	Command correction
BG	
JP #B	Repeat the process
#C	
EN	

Chapter 8 Error Handling

Introduction

The DMC-141X 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-141X is an integral part of the machine, the engineer should design his overall system with protection against a possible component failure on the DMC-141X. Galil shall not be liable or responsible for any incidental or consequential damages.

Hardware Protection

The DMC-141X 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. This signal also goes low when the watch-dog timer is activated, or upon reset. *Note: The standard configuration of the AEN signal is TTL active low. Both the polarity and the amplitude can be changed if you are using the ICM-1460 interface board. To make these changes, see section entitled 'Amplifier Interface' pg. 3-32.*

Error Output - The error output is a TTL signal which indicates an error condition in the controller. This signal is available on the interconnect module as ERROR. When the error signal is low, this indicates one of the following error conditions:

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

Input Protection Lines

Abort - A low input stops commanded motion instantly without a controller deceleration. Any motion program currently running will also be stopped. When the Off-On-Error function is enabled, the amplifiers will be disabled. This could cause the motor to 'coast' to a stop. If the Off-On-Error function is not enabled, the motor will instantaneously stop and servo at the current position. The Off-On-Error function is further discussed in this chapter.

Forward Limit Switch - Low input inhibits motion in forward direction. If the motor is moving in the forward direction when the limit switch is activated, the motion will decelerate and stop. In addition, if the motor is moving in the forward direction, the controller will automatically jump to the limit switch subroutine, #LIMSWI (if such a routine has been written by the user). The CN command can be used to change the polarity of the limit switches. To query the state of a forward limit switch, type MG_LFx where x is the specified axis.

Reverse Limit Switch - Low input inhibits motion in reverse direction. If the motor is moving in the reverse direction when the limit switch is activated, the motion will decelerate and stop. In addition, if the motor is moving in the reverse direction, the controller will automatically jump to the limit switch subroutine, #LIMSWI (if such a routine has been written by the user). The CN command can be used to change the polarity of the limit switches. To query the state of a reverse limit switch, type MG_LRx where x is the specified axis.

Software Protection

The DMC-141X 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	Set error limit for 200

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-141X 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 can be monitored during execution using the TE command.

Programmable Position Limits

The DMC-141X 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-141X will not accept position commands beyond the limit. Motion beyond the limit is also prevented.

Example:

Instruction	Interpretation
DP0	Define Position
BL -2000	Set Reverse position limit
FL 2000	Set Forward position limit
JG 2000	Jog
BG	Begin

In this example, the motor will jog forward at a speed of 2000 cts/sec., until it is stopped by the forward software limit at position 2000.

Off-On-Error

The DMC-141X controller has a built in function which can turn off the motors under certain error conditions. This function is known as 'Off-On-Error'. To activate the OE function, specify a 1. To disable this function, specify a 0. When this function is enabled, the motor will be disabled under the following 3 conditions:

1. The position error for the specified axis exceeds the limit set with the command, ER.
2. The abort command is given.
3. The abort input is activated with a low signal.

Note: If the motors are disabled while they are moving, they may 'coast' to a stop because they are no longer under servo control.

To re-enable the system, use the Reset (RS) or Servo Here (SH) command.

Examples:

OE 1	Enable off-on-error
OE 0	Disable off-on-error

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:

Instruction	Interpretation
#A;JP #A;EN	"Dummy" program
#POSERR	Start error routine on error
MG "error"	Send message
SB 1	Fire relay
ST	Stop motor
AM	After motor stops
SH	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-141X 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. The CN command can be used to configure the polarity of the limit switches.

Limit Switch Example:

Instruction	Interpretation
#A;JP #A;EN	Dummy Program
#LIMSWI	Limit Switch Utility
V1=_LF	Check if forward limit
V2=_LR	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 LIMIT"	Send message
ST;AM	Stop motion
PR-1000;BG;AM	Move in reverse
JP#END	End
#LR	#LR
MG "REVERSE LIMIT"	Send message
ST;AM	Stop motion
PR1000;BG;AM	Move forward
#END	End
RE	Return to main program

NOTE: An applications program must be executing for #LIMSWI to function.

Chapter 9 Troubleshooting

Overview

The following discussion 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

SYMPTOM	DIAGNOSIS	CAUSE	REMEDY
Motor runs away with no connections from controller to amplifier input.	Adjusting offset causes the motor to change speed.	1. Amplifier has an internal offset. 2. Damaged amplifier.	Adjust amplifier offset. Amplifier offset may also be compensated by use of the offset configuration on the controller (see the OF command). Replace amplifier.
Motor is enabled even when MO command is given	The SH command disables the motor	1. The amplifier requires the -LAEN option on the Interconnect Module	Contact Galil
Unable to read the auxiliary encoders.	No auxiliary encoder inputs are working	1. Auxiliary Encoder Cable is not connected	Connect Auxiliary Encoder cable
Unable to read main or auxiliary encoder input.	The encoder does not work when swapped with another encoder input.	1. Wrong encoder connections. 2. Encoder is damaged 3. Encoder configuration incorrect.	Check encoder wiring. For single ended encoders (CHA and CHB only) do not make any connections to the CHA- and CHB- inputs. Replace encoder Check CE command

Unable to read main or auxiliary encoder input.	The encoder works correctly when swapped with another encoder input.	1. Wrong encoder connections. 2. Encoder configuration incorrect. 3. Encoder input or controller is damaged	Check encoder wiring. For single ended encoders (CHA and CHB only) do not make any connections to the CHA- and CHB- inputs. Check CE command Contact Galil
Encoder Position Drifts	Swapping cables fixes the problem	1. Poor Connections / intermittent cable	Review all terminal connections and connector contacts.
Encoder Position Drifts	Significant noise can be seen on CHA and / or CHB encoder signals	1. Noise	Shield encoder cables Avoid placing power cables near encoder cables Avoid Ground Loops Use differential encoders Use ±12V encoders

Communication

SYMPTOM	DIAGNOSIS	CAUSE	REMEDY
Cannot communicate with controller.	Galil software returns error message when communication is attempted.	1. Bad or wrong comm. Port 2. Wrong baud rate selected 3. Bad comm. cable	Try different comm. Port or computer Make sure baud rate selected on controller agrees with software Make sure cable is straight through connection. Swap cable if necessary.

Stability

SYMPTOM	DIAGNOSIS	CAUSE	REMEDY
Servo motor runs away when the loop is closed.	Reversed Motor Type corrects situation (MT -1)	1. Wrong feedback polarity.	Reverse Motor or Encoder Wiring (remember to set Motor Type back to default value: MT 1)
Motor oscillates.		2. Too high gain or too little damping.	Decrease KI and KP. Increase KD.

Operation

SYMPTOM	DIAGNOSIS	CAUSE	REMEDY
Controller rejects commands.	Response of controller from TC1 diagnoses error.	1. Anything	Correct problem reported by TC1
Motor Doesn’t Move	Response of controller from TC1 diagnoses error.	2. Anything	Correct problem reported by SC

THIS PAGE LEFT BLANK INTENTIONALLY

Chapter 10 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.

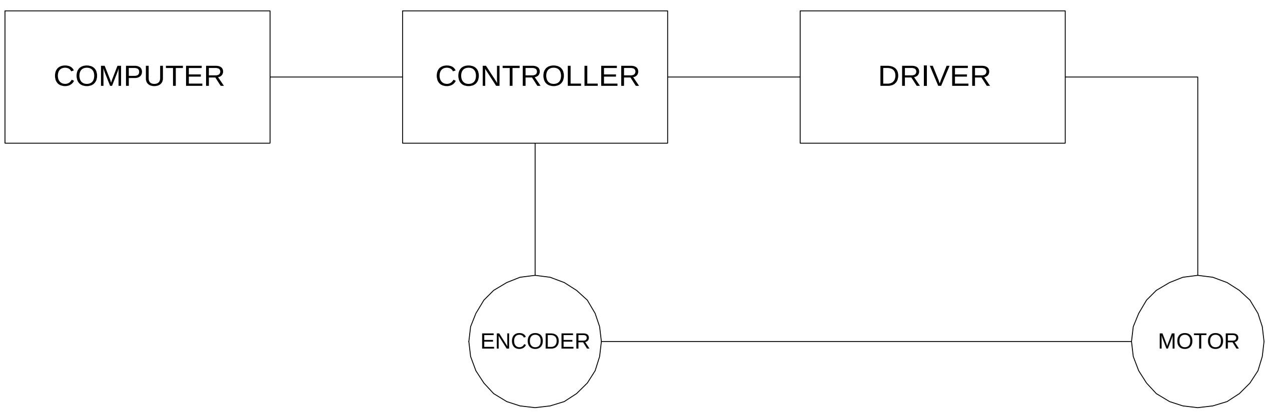


Figure 10.1 - Elements of Servo Systems

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 modeling, analysis, and design. These subjects will be covered in the following discussions.

The motion profiling is the generation of the desired position function. this function, R(t), describes where the motor should be at every sampling period. Note that the profiling and the closing of the loop are independent functions. The profiling function determines where the motor should be and the closing of the loop forces the motor to follow the commanded position

The highest level of control is the motion program. This can be stored in the host computer or in the controller. This program describes the tasks in terms of the motors that need to be controlled, the distances and the speed.

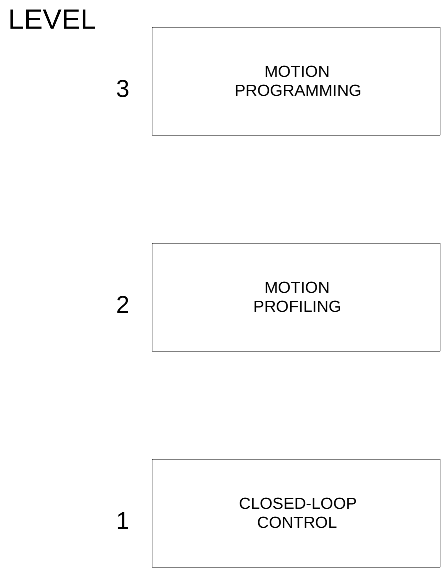


Figure 10.2 - Levels of Control Functions

The three levels of control may be viewed as different levels of management. The top manager, the motion program, may specify the following instruction, for example.

PR 6000

SP 20000

AC 200000

BG

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 modeling, analysis and design.

System Modeling

The elements of a servo system include the motor, driver, encoder and the controller. These elements are shown in Fig. 10.4. The mathematical model of the various components is given below.

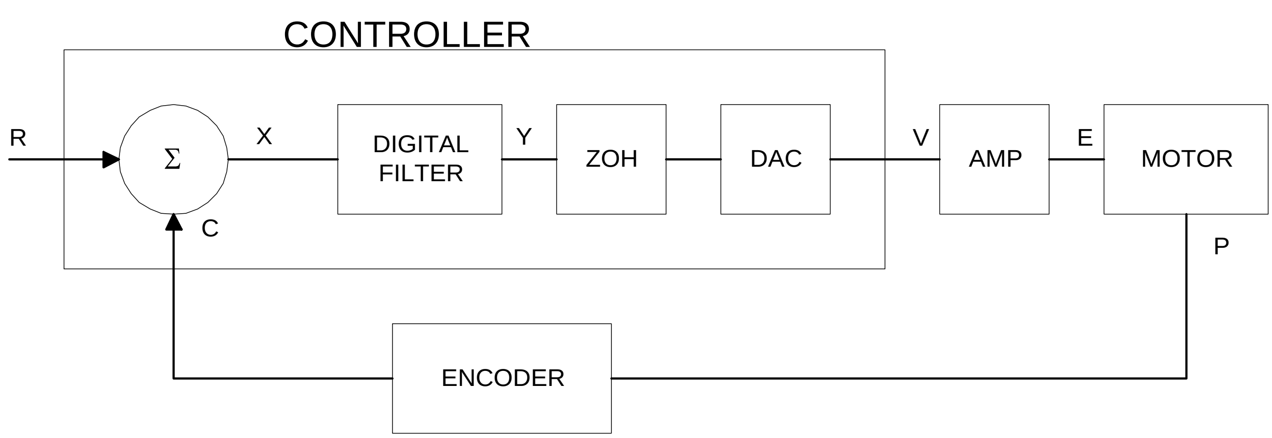


Figure 10.4 - Functional Elements of a Motion Control System

Motor-Amplifier

The motor amplifier may be configured in three modes:

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

The operation and modeling 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

$$P/V = K_V / \left[K_t S\left(ST_m + 1\right)\left(ST_e + 1\right)\right]$$

where

$$T_m = RJ/K_t^2 \quad \text{[s]}$$

and

$$T_e = L/R \quad \text{[s]}$$

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

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

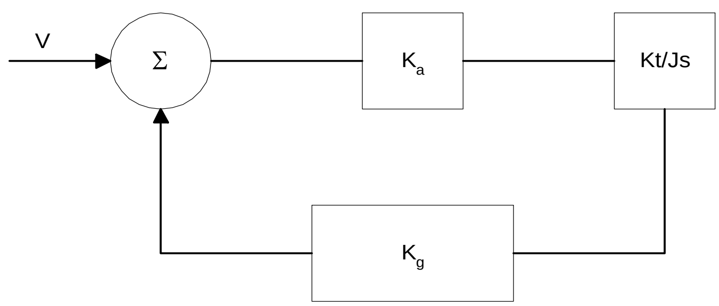


Figure 10.5 - Elements of velocity loops

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

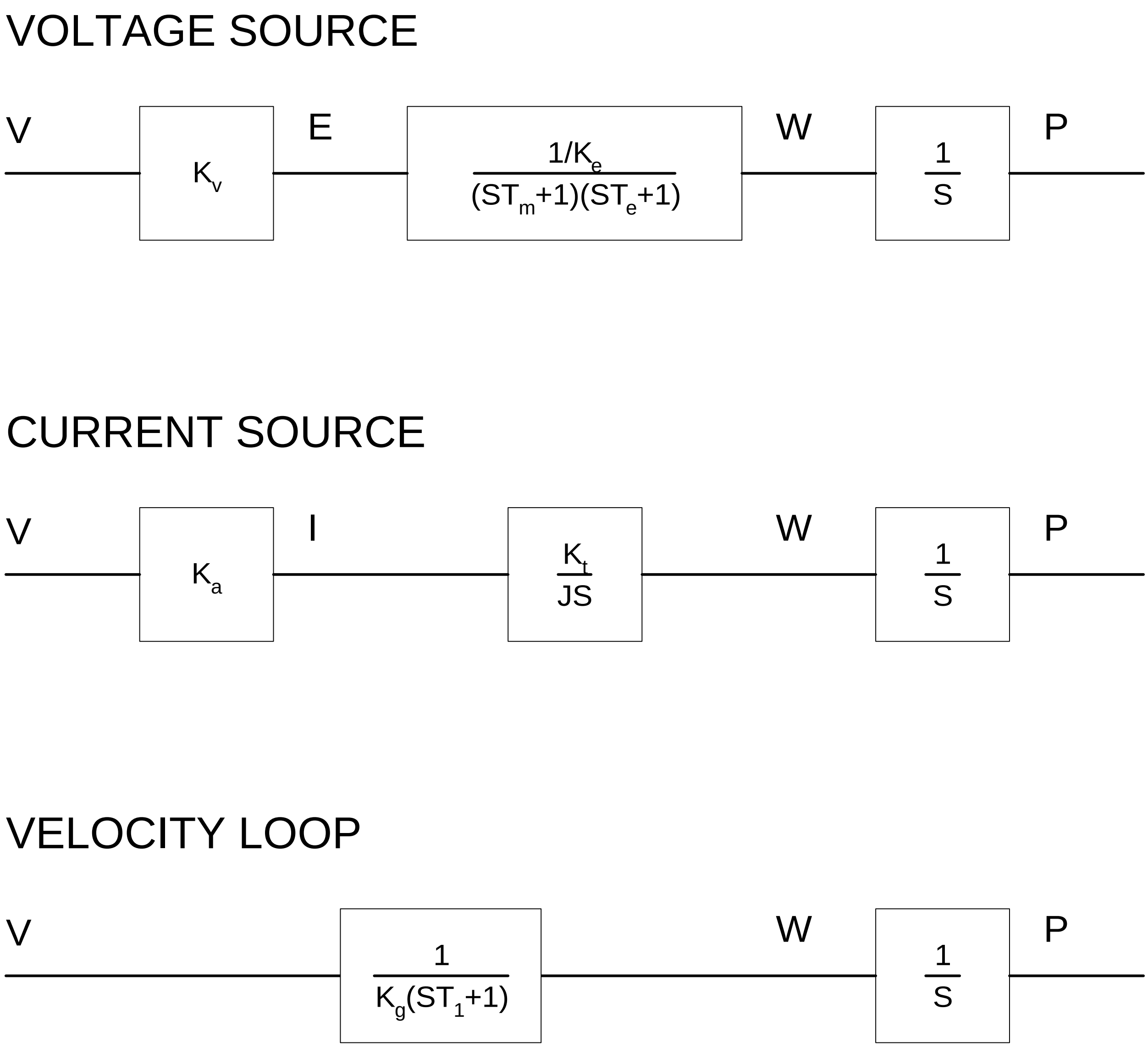


Figure 10.6 - Mathematical model of the motor and amplifier in three operational modes

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 modeled as

$K_f = 638$

DAC

The DAC or D-to-A converter converts a 16-bit number to an analog voltage. The input range of the numbers is 65,536 and the output voltage range is ±10 V or 20 V. Therefore, the effective gain of the DAC is

K= 20/65,536 = 0.0003 [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) • 4 or K = GN • 4

A = KD/(KP + KD) or A = ZR

C = KI/2

This filter includes a lead compensation and an integrator. It is equivalent to a continuous PID filter with a transfer function G(s).

G(s) = P + sD + I/s

P = 4KP

D = 4T • KD

I = KI/2T

For example, if the filter parameters of the DMC-141X are

KP = 4

KD = 36

KI = 0.5

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 + 0.144s + 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 modeled by the transfer function

H(s) = 1/(1+sT/2)

If the sampling period is T = 0.001, for example, H(s) becomes:

H(s) = 2000/(s+2000)

However, in most applications, H(s) may be approximated as one.

This completes the modeling of the system elements. Next, we discuss the system analysis.

System Analysis

To analyze the system, we start with a block diagram model of the system elements. The analysis procedure is illustrated in terms of the following example.

Consider a position control system with the DMC-141X 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

$$K_d = 0.0003 \ [V/count]$$

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:

$$P = 50$$

$$D = 0.98$$

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

$$G(s) = 50 + 0.98s = 0.98(s+51)$$

The system elements are shown in Fig. 10.7.

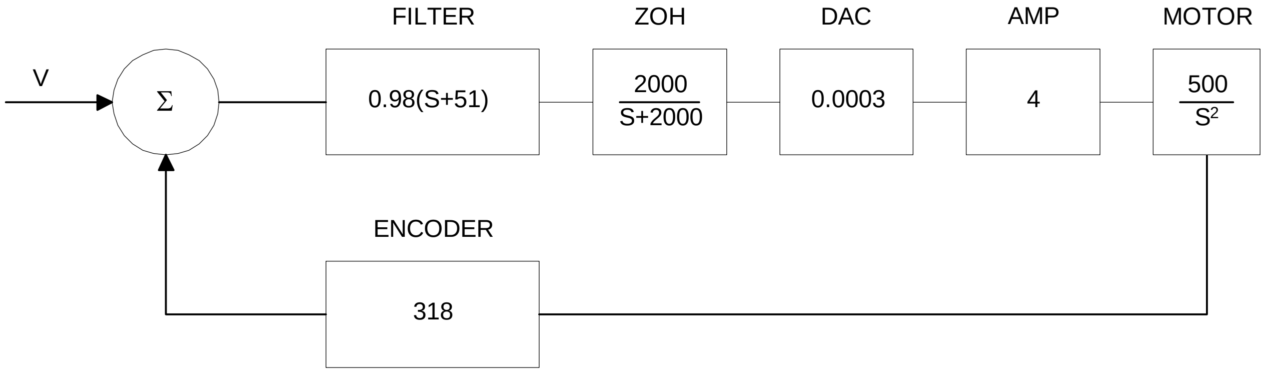


Figure 10.7 - Mathematical model of the control system

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

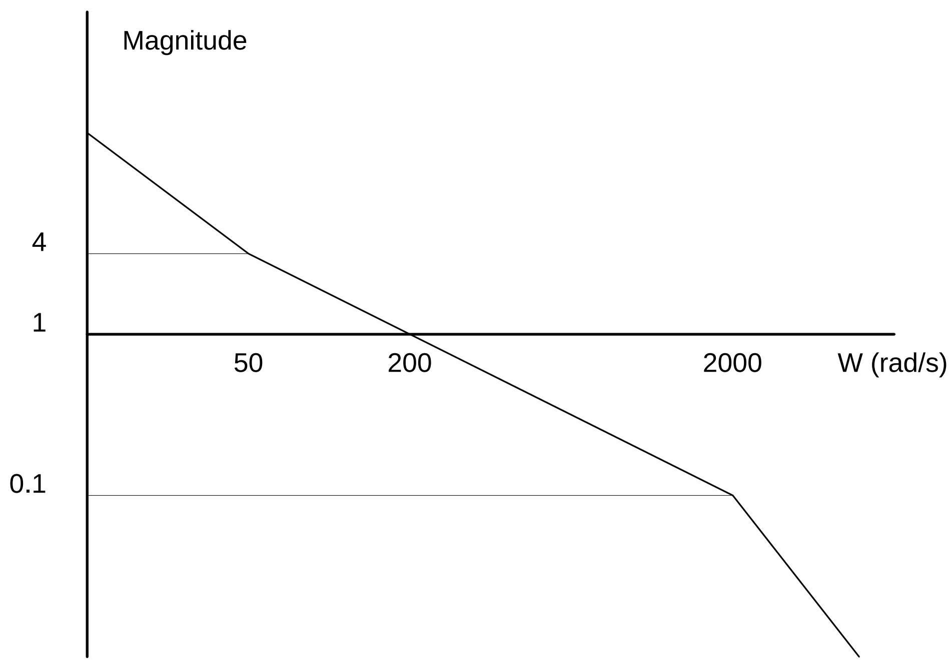


Figure 10.8 - Bode plot of the open loop transfer function

For the given example, the crossover frequency was computed numerically resulting in 200 rad/s.

Next, we determine the phase of A(s) at the crossover frequency.

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

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

$$\alpha = 76° - 180° - 6° = -110°$$

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-141X 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-141X outputs ±10 V for a 16-bit command of ±32,768 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

$$K_d = 10/32{,}768$$

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).

$$L(s) = M(s)\, K_a\, K_d\, K_f\, H(s) = 3.175*10^6/[s^2(s+2000)]$$

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 $\omega_c = 500$.

$$L(j500) = 3.175*10^6/[(j500)^2 (j500+2000)]$$

This function has a magnitude of

$$|L(j500)| = 0.00625$$

and a phase

$$Arg[L(j500)] = -180° - \tan^{-1}(500/2000) = -194°$$

G(s) is selected so that A(s) has a crossover frequency of 500 rad/s and a phase margin of 45 degrees. This requires that

$$|A(j500)| = 1$$

$$Arg [A(j500)] = -135°$$

However, since

$$A(s) = L(s) G(s)$$

then it follows that G(s) must have magnitude of

$$|G(j500)| = |A(j500)/L(j500)| = 160$$

and a phase

$$\arg [G(j500)] = \arg [A(j500)] - \arg [L(j500)] = -135° + 194° = 59°$$

In other words, we need to select a filter function G(s) of the form

$$G(s) = P + sD$$

so that at the frequency ω_c =500, the function would have a magnitude of 160 and a phase lead of 59 degrees.

These requirements may be expressed as:

$$|G(j500)| = |P + (j500D)| = 160$$

and

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

The solution of these equations leads to:

$$P = 160\cos 59° = 82.4$$

$$500D = 160\sin 59° = 137.2$$

Therefore,

$$D = 0.274$$

and

$$G = 82.4 + 0.274s$$

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

$$D(z) = 4*KP + 4* KD(1-z^{-1})$$

where

$$KP = P/4$$

and

$KD = D/4T$

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

KP = 20.6

KD = 68.6

The DMC-141X can be programmed with the instruction:

KP 20.6

KD 68.6

In a similar manner, other filters can be programmed. The procedure is simplified by the following table, which summarizes the relationship between the various filters.

Equivalent Filter Form

	DMC - 1410
Digital	$D(z) = K(z-A/z) + Cz/(z-1)$
Digital	$D(z) = 4\,KP + 4\,KD(1-z^{-1}) + KI/2(1-z^{-1})$
KP, KD, KI	$K = (KP + KD) \cdot 4$
	$A = KD/(KP+KD)$
	$C = KI/2$
Digital	$D(z) = 4\,GN(z-ZR)/z + KI\,z/2(z-1)$
GN, ZR, KI	$K = 4\,GN$
	$A = ZR$
	$C = KI/2$
Continuous	$G(s) = P + Ds + I/s$
PID, T	$P = 4\,KP$
	$D = 4\,T{*}KD$
	$I = KI/2T$

THIS PAGE LEFT BLANK INTENTIONALLY

Appendices

Electrical Specifications

Servo Control

ACMD Amplifier Command:	±10 volts analog signal. Resolution 16-bit, .0003 volts. 3 mA maximum
A+,A-,B+,B-,IDX+,IDX- Main Encoder Input A+, A-, B+, B- Aux Encoder input	TTL compatible, but can accept up to ±12 volts. Quadrature phase on CHA, CHB. Can accept single-ended (A+,B+ only) or differential (A+,A-,B+,B-). Maximum A,B edge rate: 8 MHz. Minimum IDX pulse width: 120 nsec.

Stepper Control

Pulse	TTL (0-5 volts) level at 50% duty cycle. 2,000,000 pulses/sec maximum frequency.
Direction	TTL (0-5 volts).

Input/Output

Limits, Home, Abort Inputs:	Line receiver inputs biased for 0-5v operation. Can accept up to +12 V signal.
OUT[1] thru OUT[3] Outputs:	TTL buffer output, 0-5 V.
IN[1] through IN[7] Inputs	Line receiver inputs biased for 0-5 V operation. Can accept up to +12 V signal.

Power Requirements

+5 V	400 mA
+12 V	20 mA
-12 V	20mA

Note: The +12 V DC-to-DC converter on the DMC-1414 is maxed out at 30mA. Do not try to draw any current out of the +12 V pins. The +5 V can supply 0.5A; the –12 V can supply 100mA.

Performance Specifications

Minimum Servo Loop Update Time:	250 μsec
Position Accuracy:	±1 quadrature count
Velocity Accuracy:	
Long Term	Phase-locked, better than 0.005%
Short Term	System dependent
Position Range:	±2147483647 counts per move
Velocity Range:	Up to 8,000,000 counts/sec
Velocity Resolution:	2 counts/sec
Motor Command Resolution:	16 bit DAC over ±10V range, 0.0003 V
Variable Range:	±2 billion. 4 bytes integer 32 bits, 2 bytes fraction 16 bits
Variable Resolution:	$1 \cdot 10^{-4}$. 4 bytes integer 32 bits, 2 bytes fraction 16 bits
Array Size:	1000 elements; 6 arrays
Program Size:	250 lines x 40 characters

Connectors

DMC-1412,1414: J3 General I/O; 37- PIN D-type

1 Reset*	20 Error*
2 Amp Enable	21 Amp Command for Servo motors
3 Output 3	22 Output 2
4 Output 1	23 Reserved
5 PWM or Step Out	24 Sign or Direction
6 Input 7	25 Input 6
7 Input 5	26 Input 4
8 Input 3	27 Input 2
9 Input 1 (and latch*)	28 Forward Limit*
10 + 5V	29 Reverse Limit*
11 Ground	30 Home
12 +12V	31 -12v
13 Ground	32 A+
14 A -	33 B+
15 B -	34 I+
16 I -	35 Auxiliary A +
17 Auxiliary A -	36 Auxiliary B +

18 Auxiliary B -	37 Abort*
19 Reserved	

DMC-1412 card: J5 Power: 7- PIN Molex:

1 -12V	5 +5V
2 Ground	6 +12V
3 Ground	7 Earth
4 +5V	

DMC-1414: J2 Power: 5-PIN Female

Terminal #	Label	Description
1	EARTH	Chassis Connection
2	GND	Input Power Return
3	MOTOR2	Motor Connection
4	MOTOR1	Motor Connection
5	AMP V+	Input Power

DMC-1412/1414: RS232 Main port: DB-9 Pin Male:

1 CTS- output	6 CTS - output
2 Transmit data-output	7 RTS - input
3 Receive Data-input	8 CTS - output
4 RTS – input	9 No connect
5 Ground	

DMC-1412: RS232 Auxiliary Port; DB-9 pin Female:

1 CTS – input	6 CTS - input
2 Transmit data-input	7 RTS - output
3 Receive data-output	8. CTS - input
4 RTS – output	9 5V
5 Ground	

DMC-1414: J3 General I/O Terminal Connections

Terminal #	Label	I/O	Description
1	GND		Signal Ground
2	5V	O	+ 5 volts
3	AB-	I	Auxiliary encoder B-

4	AB+	I	Auxiliary encoder B+
5	AA-	I	Auxiliary encoder A-
6	AA+	I	Auxiliary encoder A+
7	IDX-	I	Main encoder index -
8	IDX+	I	Main encoder index +
9	5V	O	+ 5 volts
10	MB-	I	Main encoder B-
11	MB+	I	Main encoder B+
12	MA-	I	Main encoder A-
13	MA+	I	Main encoder A+
14	GND		Signal Ground
15	ABORT	I	Abort Input
16	HOME	I	Home input
17	RLS	I	Reverse limit switch input
18	FLS	I	Forward limit switch input
19	IN1/LTCH	I	Input 1 / Input for Latch Function
20	IN2	I	Input 2
21	IN3	I	Input 3
22	IN4	I	Input 4
23	IN5	I	Input 5
24	IN6	I	Input 6
25	IN7	I	Input 7
26	GND		Signal Ground
27	5V	O	+ 5 volts
28	CMP	O	Circular Compare output
29	OUT1	O	Output 1
30	OUT2	O	Output 2
31	OUT3	O	Output 3
32	ERROR	O	Error signal
33	RESET	I	Reset
34	GND		Signal Ground
35	SIGN	O	Direction output for input to stepper motor amp
36	PWM	O	Pulse output for input to stepper motor amp
37	ACMD	O	Motor command to amp input (w / respect to ground)
38	AMPEN	O	Amplifier enable
39	-12V		-12 volts
40	+12V		+12 volts

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 16.7 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 step motors: The STEP OUT pin produces a series of pulses for input to a step motor driver. The pulses may either be low or high. The pulse width is 50%. Upon Reset, the output will be low if the SM jumper is on. If the SM jumper is not on, the output will be tristate.
Sign/Direction	Used with PWM signal to give the sign of the motor command for servo amplifiers or direction for step 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 3	These 3 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	
Main 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.
Main Encoder Index, I+	Once-Per-Revolution encoder pulse. Used in Homing sequence or Find Index command to define home on an encoder index.

Main 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.
Aux Encoder, A+, B+, A-, B-	Inputs for additional encoder. Used when an encoder on both the motor and the load is required.
Abort input	A low input stops commanded motion instantly without a controlled deceleration. Also aborts motion program.
Reset input	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 7	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 1is used for the high-speed latch.
Latch input	High speed position latch to capture axis position in less than 1 μsec on occurrence of latch signal. AL command arms latch. Input 1 is latch

Jumpers (DMC-1412/DMC-1414)

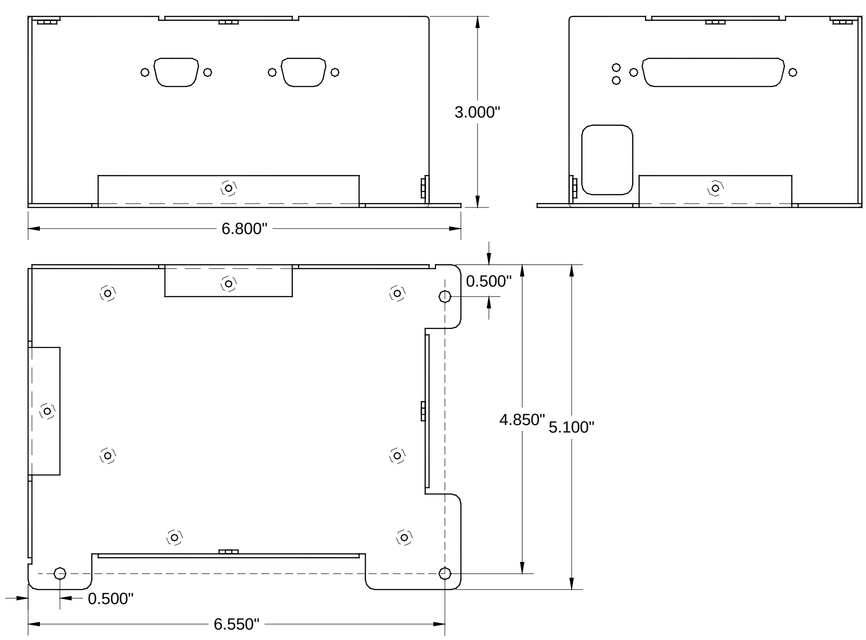
Label	**Function (If jumpered)**
SMX	The SM jumper selects the SM magnitude mode for servo motors or selects stepper motors. If you are using stepper motors, SM must always be jumpered. The Analog command is not valid with SM jumpered.
OPT	Reserved
9600 and 38.4	Selects baud rate. See getting started in chapter 2.
MRST	Master Reset enable. Returns controller to factory default settings and erases EEPROM. Requires power-on or RESET to be activated.

Accessories and Options

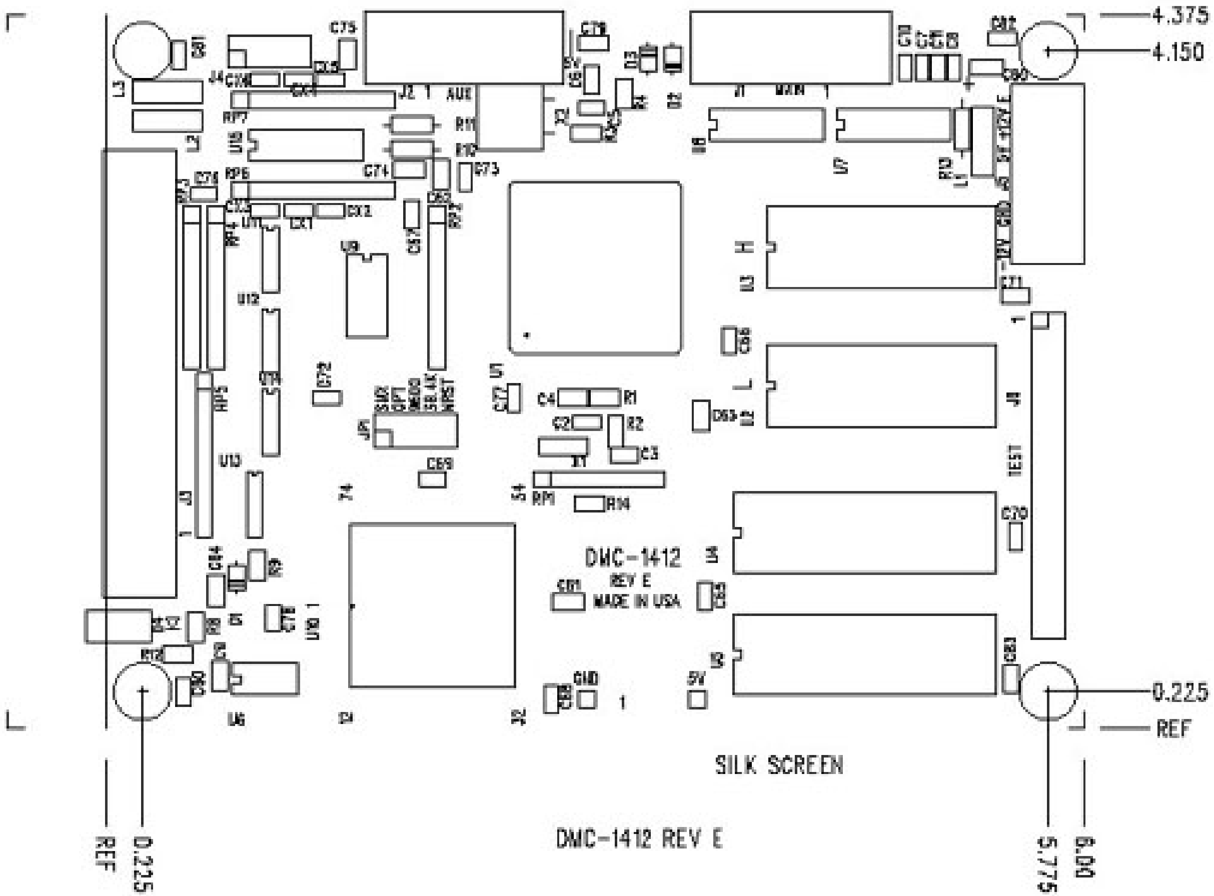
Part #	Description
DMC-1412	1-axis motion controller with RS232
DMC-1414	1-axis integrated RS232 motion controller/DC brush-type amplifier
ICM-1460	Interconnect module
AMP-1460	Interconnect module with 1-axis power amplifier
Cable 37-pin D	37 - pin cable for DMC-1410 & DMC-1412
Cable 9-pin D	9-pin RS232 cable for DMC-1412/1414
Galil Software CD	Terminal emulation and communication drivers and DLL for Windows ™
WSDK-16 bit	Servo Design Kit for Windows 3.X
WSDK-32 bit	Servo Design Kit for Windows 95, 98, NT4, ME, 2000, and XP.
VB Toolkit	Visual Basic™ Tool Kit
TERM-1500H	Handheld terminal
TERM-1500P	Panel mount terminal

DMC-1412 – Box Dimensions

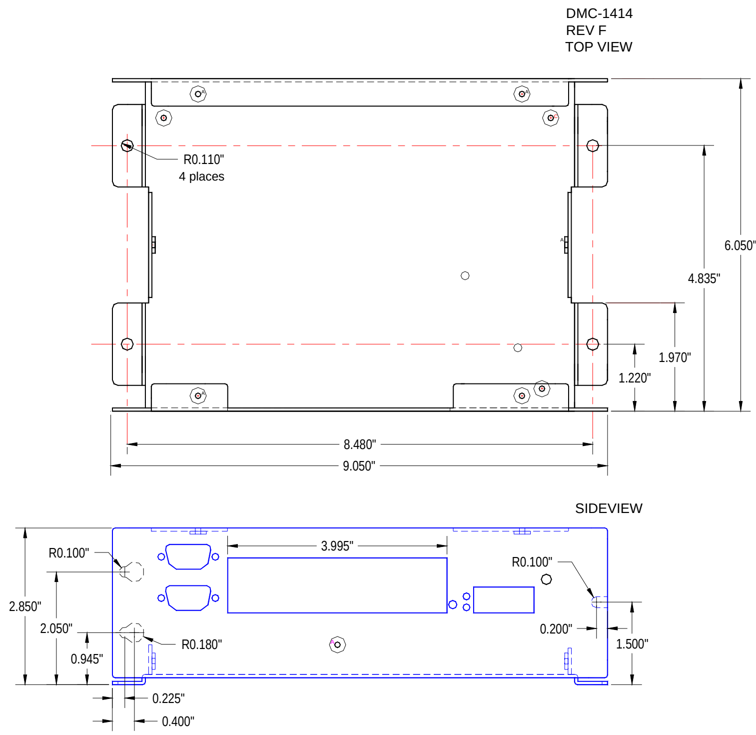
DMC-1412 Box



DMC-1412 – Card Dimensions



DMC-1414 – Dimensions



ICM-1460 Interconnect Module

The ICM-1460 Interconnect Module provides easy connections between the DMC-141X series controllers and other system elements, such as amplifiers, encoders, and external switches. The ICM-1460 accepts the 37-pin cable from the DMC-1410 or the 40 pin to 37-pin cable from the DMC-1411 and breaks it into screw-type terminals. Each screw terminal is labeled for quick connection of system elements.

The ICM-1460 is packaged as a circuit board mounted to a metal enclosure. A version of the ICM-1460 is also available with a servo amplifier (see AMP-1460).

Features:

- Breaks out 37-pin ribbon cable into individual screw-type terminals.
- Clearly identifies all terminals

- Available with on-board servo drive (see AMP-1460).
- 10-pin IDC connectors for encoders.

Specifications:

Dimensions: 6.9" x 4.9" x 2.6" Weight: 1 pound

Rev A-F Terminal #	Rev G Terminal #	Label	I/O	Description
1	1	+12V[4]	O	+12 Volts
2	2	-12V[4]	O	-12 Volts
3	3	AMPEN/SIGNY[5]	O	Amplifier enable X axis or Y Axis Sign Output for Stepper
4	4	ACMDX/PULSE(X)	O	X Axis Motor command or Pulse Output for Stepper
5	5	AN1	O	Analog Input 1
6	6	AI2	O	Analog Input 2
7	7	GND	--	Signal Ground
8	8	RESET	I	Reset
9	9	ERROR/PULSE(Y)[6]	O	Error signal or Y Axis Pulse Output for Stepper
10	10	OUT3	O	Output 3
11	11	OUT2	O	Output 2
12	12	OUT1	O	Output 1
13	13	CMP/ICOM[7]	O	Circular Compare / Input common for Opto option
14	14	5V	O	+ 5 Volts
15	15	GND	--	Signal Ground
16	16	IN7/INDY+	I	Input 7 (Y Axis Main Encoder Index + for DMC-1425)
17	17	IN6/HOMY	I	Input 6 (Y Axis Home input for DMC-1425)
18	18	IN5/RLSY	I	Input 5 (Y axis reverse limit on DMC-1425)
19	19	IN4/FLSY	I	Input 4 (Y axis forward limit on DMC-1425)
20	20	IN3/IDY-	I	Input 3 (Y axis main encoder index for DMC-1425)
21	21	IN2	I	Input 2
22	22	IN1/LTCH	I	Input 1 / Input for Latch Function

23	23	FLSX	I	Forward limit switch input
24	24	RLSX	I	Reverse limit switch input
25	25	HOMX	I	Home input
26	26	ABORT	I	Abort Input
27	27	GND	--	Signal Ground
28	28	MA+	I	X Axis Main Encoder A+ [5]
29	29	MA-	I	X Axis Main Encoder A- [5]
30	30	MB+	I	X Axis Main Encoder B+ [5]
31	31	MB-	I	X Axis Main Encoder B- [5]
32	32	IDX+	I	X Axis Main Encoder Index + [5]
33	33	IDX-	I	X Axis Main Encoder Index – [5]
34	34	AA+	I	X Axis Auxiliary Encoder A+ (Y Axis Main Encoder A+ for DMC-1425)
35	35	AA-	I	X Axis Auxiliary Encoder A- (Y Axis Main Encoder A- for DMC-1425)
36	36	AB+	I	X Axis Auxiliary Encoder B+ (Y Axis Main Encoder B+ for DMC-1425)
37	37	AB-	I	X Axis Auxiliary Encoder B- (Y Axis Main Encoder B- for DMC-1425)
38	38	ACMD2/SIGNX	O	2nd Motor command Signal for Sine Amplifier or SIGNX for stepper
39	39	5V	O	+ 5 Volts
40	40	GND	--	Signal Ground

1 The screw terminals for +/-12V can be configured as opto-input/output common. See next section for detail.
2 The screw terminal for amplifier enable output can be configured as the stepper motor direction output for Y axis for DMC1425 controller. This needs to be specified when ordering the controller. Please contact Galil for detailed info.
3 The screw terminal for ERROR Output can be configured as the stepper motor pulse output for Y axis for DMC1425 controller. This needs to be specified when ordering the controller. Please contact Galil for detailed info.
4 The screw terminal for CMP can be configured as input/output common for opto-isolated I/O. Please see next section for detail.

J8, 9 Encoder -10pin header

1	Main Encoder A+	2	+5 VDC
3	GND	4	NC
5	NC	6	NC
7	NC	8	Main encoder B+
9	NC	10	Main encoder I+

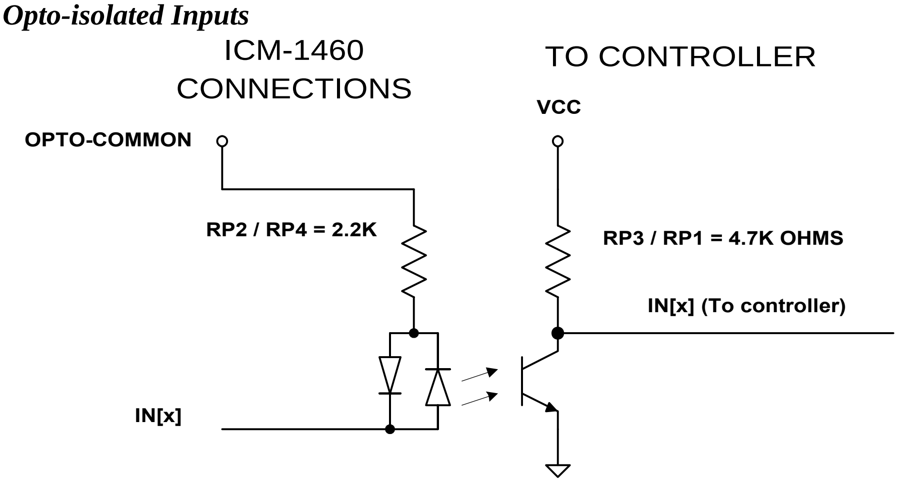
Opto-Isolation Option for ICM-1460 (rev F and above only)

The ICM-1460 module from Galil has an option for opto-isolated inputs and outputs. Any of the following pins can be chosen to be the input/output common: pin 1 (labeled as +12V), pin 2 (labeled as –12V) and pin 13 (labeled as CMP/ICOM). When pin 1 is used as input/output common, the +12V output be comes inaccessible, when pin 2 is used, the –12V becomes inaccessible, and when pin13 is used, the output compare function is not available. The common point needs to be specified at the time of ordering.

The ICM-1460 can also be configured so that the opto common is jumped with Vcc (+5V). In this case, no screw connections is needed, and the internal 5V will be used for powering the input/output.

Option for separate input/output commons is also available. This will require the use of both pin 1 and pin 2. When selecting this option, both +12V and –12V become inaccessible.

Opto-isolated Inputs

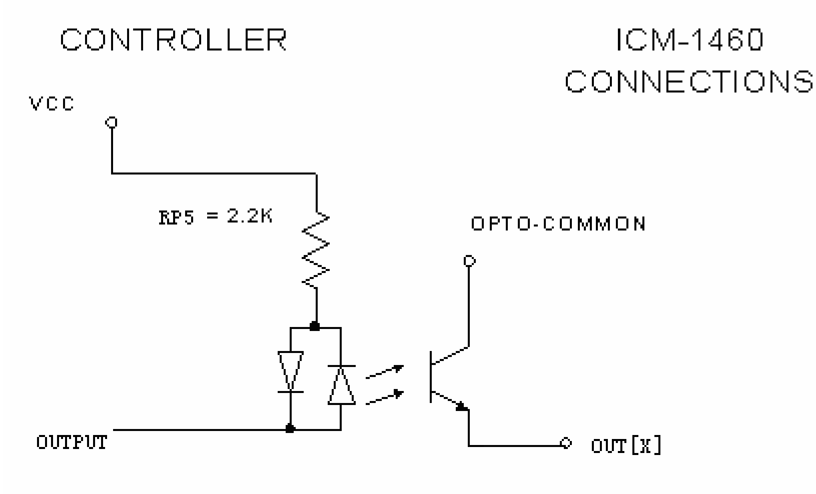


The signal "IN[x]" is one of the isolated digital inputs where x stands for the digital input terminal.

The OPTO COMMON signal is available on TERMINAL 13 labeled CMP/ICOM. The OPTO COMMON point should be connected to an isolated power supply in order to obtain isolation from the controller. By connecting the OPTO-COMMON to the + side of the power supply, the inputs will be activated by sinking current. By connecting the OPTO-COMMON to the GND side of the power supply, the inputs will be activated by sourcing current.

The opto-isolation circuit requires 1ma drive current with approximately 400 µsec response time. The voltage should not exceed 24 V without placing additional resistance to limit the current to 11 mA.

Opto-isolated Outputs



The signal “OUT[x]" is one of the isolated digital outputs where x stands for the digital output terminal.

The OPTO-COMMON needs to be connected to an isolated power supply. The OUT[x] can be used to source current from the power supply. The maximum sourcing current for the OUT[x] is 25 ma. Sinking configuration can also be specified. Please contact Galil for details.

When opto-isolated outputs are used, either a pull-up or pull-down resistor needs to be provided by the user depending upon whether the signal is sinking or sourcing.

AMP-1460 Mating Power Amplifiers

The AMP-1460 provides the features of the ICM-1460, with the addition of a brush-type servo amplifier. The amplifier is rated for 7 amps continuous, 10 amps peak at up to 80 volts. The gain of the AMP-1460 is 1 amp per volt.

The AMP-1460 requires an external DC supply. The AMP-1460 connects to the controller with a cable 37 pin cable, and screw-type terminals are provided for connecting to motors, encoders and external switches.

- 7 amps continuous, 10 amps peak; 20 to 80 volts. DC supply.
- Connects directly to DMC-141X series controllers via 37 pin cable.
- Screw-type terminals for easy connection to motors, encoders and switches.

Specifications

Minimum motor inductance:	1 mH
PWM frequency	30 kHz
Ambient operating temperature	0-70° C
Dimensions	6.9" x 4.9" x 2.6"
Weight	1 pound
Mounting	Keyholes - .2”Φ
Gain	1 amp/volt

The DMC-141X generates a ±10 volt range analog signal, ACMD, and ground (pin 21) for input to power amplifiers which have been sized to drive the motor and load. For best performance, the amplifier 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-1460 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.

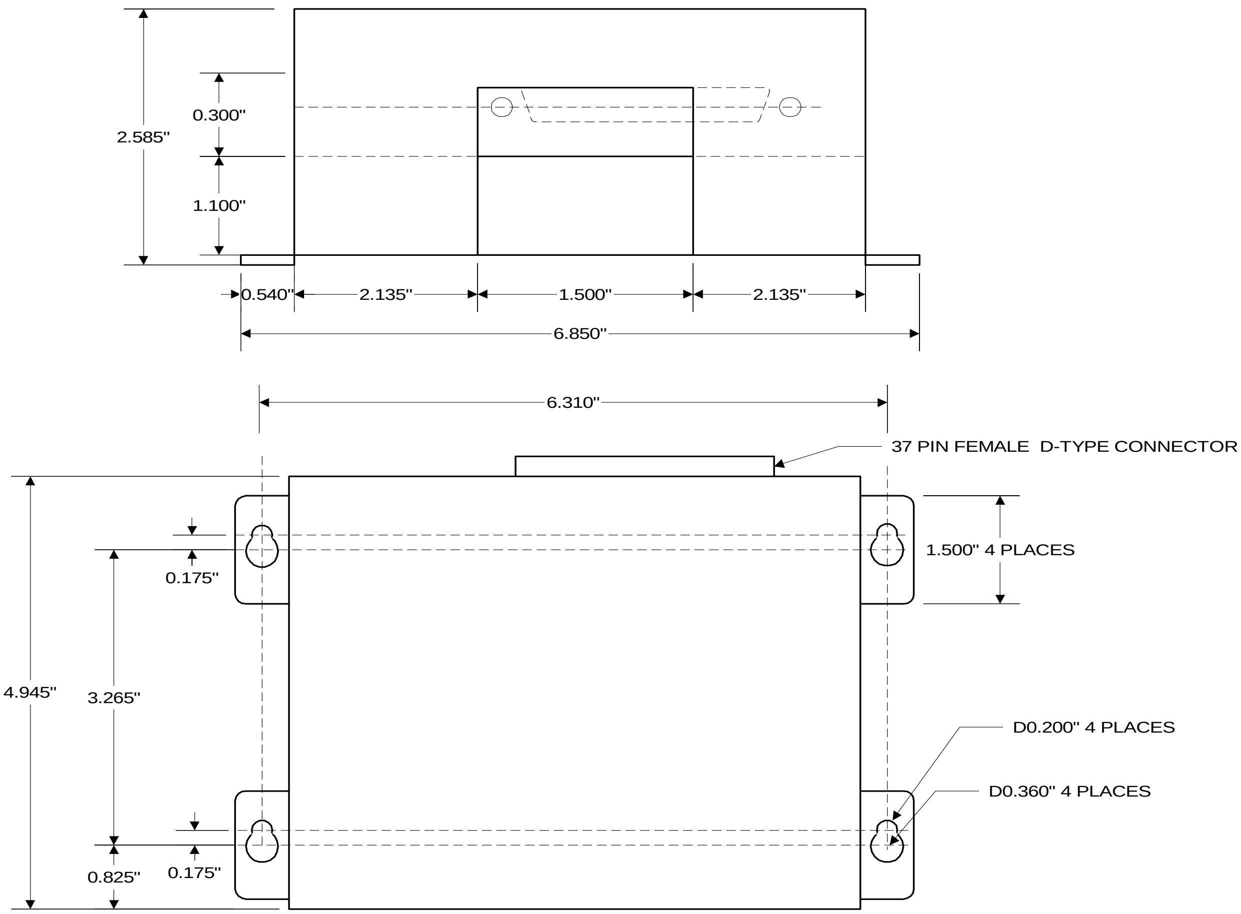
The standard configuration of the AEN signal is TTL active low. Both the polarity and the amplitude can be changed if you are using the ICM-1460 interface board. To change the polarity from active low (zero volts = disable) to active high replace the 7407 IC with a 7406.

To change the voltage level, note the state of the jumper on the ICM-1460. When JP4 has a jumper from “AEN” to “5V” (default setting), the output voltage is 0-5V. To change to 12 volts, pull the jumper and rotate it so that it connects the pins marked “AEN” and “+12V”. If the jumper is removed entirely, the output is an open collector signal, allowing the user to connect to external supplies with voltages up to 24 V.

AMP-1460 20 Watt Linear Amplifier Option

The ICM-1460 Interconnect Module can be purchased with a 20 watt linear amplifier suitable for driving small motors. This amplifier requires an external supply of ±10 V to ±35 V. Care should be taken to ensure the average power dissipation across the amplifier is less than 20watts.

ICM-1460 Drawing



TERM-1500 Operator Terminal

General Description:

Galil offers two terminals for interface to Galil Stand Alone, RS-232 controllers:

- TERM-P panel mount unit
- TERM-H hand held unit.

Both units have the same programming characteristics.

The TERM is a compact ASCII terminal for use with Galil RS-232 based motion controllers. Its numeric keypad allows easy data entry from an operator. The TERM is available with a male adapter for connection to the auxiliary serial port (Dataset).

NOTE: Since the TERM-1500 requires +5 V on pin 9 of RS-232, it can only work with port 2 of the DMC-1412/1414.

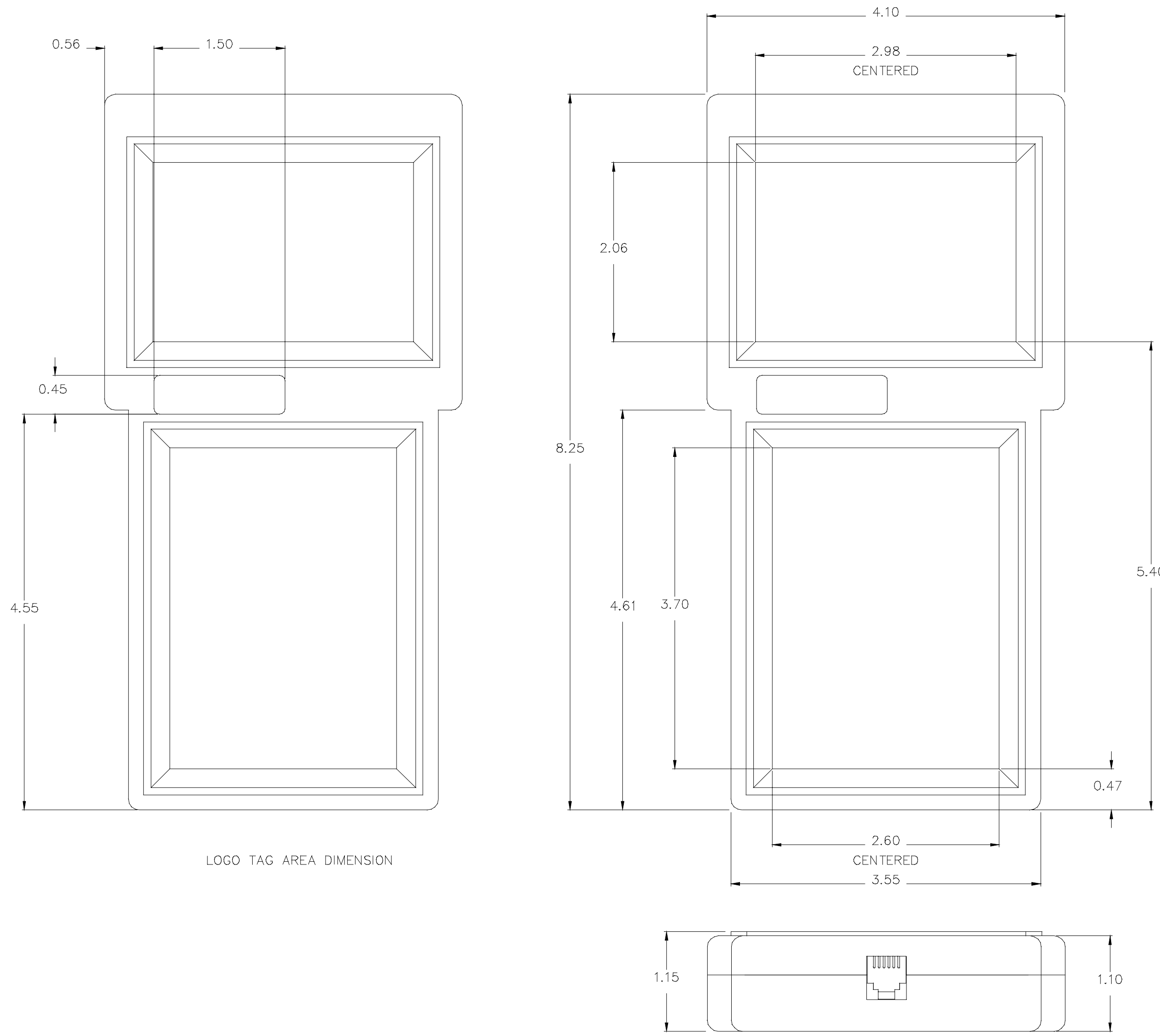


Figure 1. Hand Held Terminal

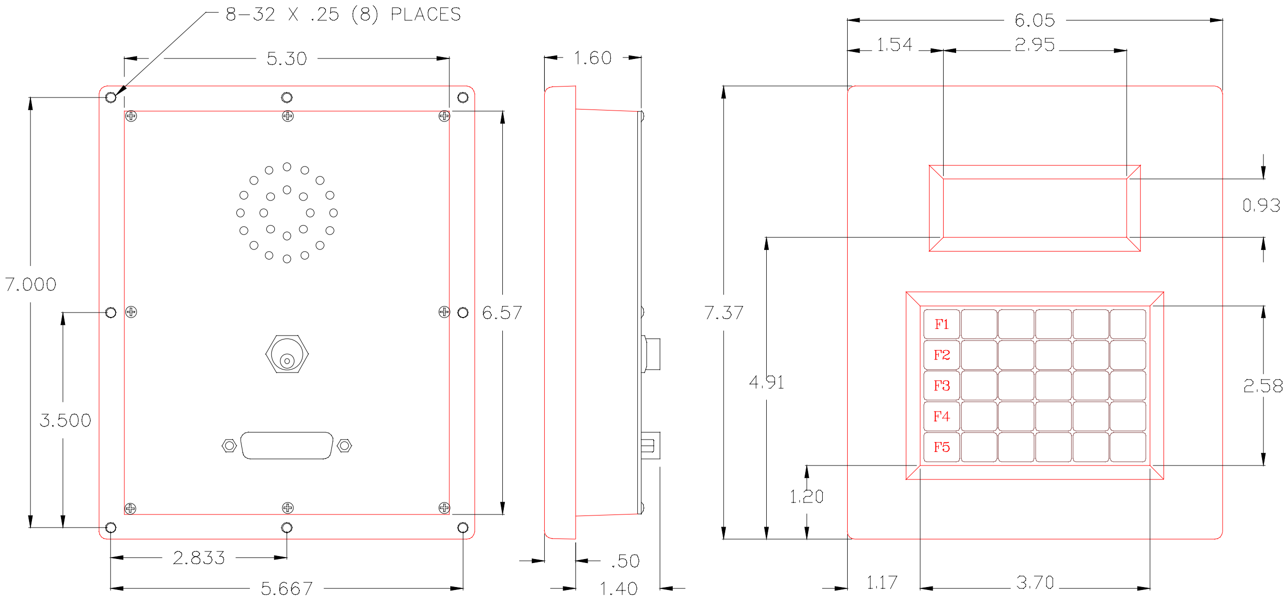


Figure 2. Panel Mount Terminal

Features

- For easy data entry to DMC-1412/1414 motion controller
- 4 line x 20 character Liquid Crystal Display
- Full numeric keypad
- Five programmable function keys
- Available in Hand-held or Panel Mount
- No external power supply required
- Connects directly to RS232 port P2 via coiled cable

<u>Specifications - Hand-Held</u>

Keypad	30-Key: 6 rows x 5 columns
Display	4 row x 20 character LCD
Power	5 volts, 30mA (from DMC-1412/1414)

<u>Specifications - Panel Mount</u>

Keypad	30-Key; 5 rows x 6 columns
Display	4 row x 20 character LCD
Power	5 volts, 30mA (from DMC-1412/1414)

Keypad Maps - Hand-Held -- 30 Keys: 6 rows by 5 columns

Single Key Output	6	F1 (22)	F2 (23)	F3 (24)	F4 (25)	F5 (26)
	5		1	2	3	
	4		4	5	6	
	3		7	8	9	
	2			0		
	1	CTRL	SHIFT	SPACE	BKSPC	ENTER

Shift Key Output	6	A	B	C	D	E
	5	F	G	H	I	J
	4	K	L	M	N	O
	3	P	Q	R	S	T
	2	U	V	W	X	Y
	1	CTRL	SHIFT	Z	,	?

CTRL Key Output	6	(18)	(16)	(9)	(4)	(17)
	5	(19)	(2)	!	"	%
	4	*	+	/	$	;
	3	<	>	\	[	]
	2	^	-	@	{	}
	1	CTRL	SHIFT	ESC	=	#
		1	2	3	4	5

Note: Values in parentheses are ASCII decimal values. Key locations are represented by [m,n] where m is element column, n is element row. The first column in the above tables is for numbering the rows and is not a column of buttons on the TERM keypad.

Example:

U is <Shift>[1,2]

is <Ctrl>[5,1]

Keypad Map - Panel Mount – 5 rows by 6 columns

Single Key Output

5	F1		1	2	3	
4	F2		4	5	6	
3	F3		7	8	9	
2	F4		-	0	.	
1	F5	CTRL	SHIFT	SPACE	BKSP	ENTER

Shift Key Output

5	A	F	G	H	I	J
4	B	K	L	M	N	O
3	C	P	Q	R	S	T
2	D	U	V	W	X	Y
1	E	CTRL	SHIFT	Z	,	?

CTRL Key Output

5	(18)	(19)	(2)	!	"	%
4	(16)	*	+	/	$	;
3	(9)	<	>	\	[	]
2	(4)	^	-	@	{	}
1	(17)	CTRL	SHIFT	ESC	=	#
	1	2	3	4	5	6

Note: Values in parentheses are ASCII decimal values. Key locations are represented by [m,n] where m is element column, n is element row. The first column in the above tables is for numbering the rows and is not a column of buttons on the TERM keypad.

Escape Commands

Escape codes can be used to control the TERM display, cursor style, and position, and sound settings. The controller syntax for the escape character is {^27}, so the command MG{P2},{^27},"H" sends ESC H to the TERM. Twenty-seven is the ASCII decimal value for the Escape command. See the controller Command Reference for more information on the MG command.

The same command can be sent from the TERM keypad by pushing <CTRL>SPACE then <SHIFT>[3,5].

Cursor Movement Commands

ESC A	Cursor Up
ESC B	Cursor Down
ESC C	Cursor Right
ESC D	Cursor Left

In the above sequences, the cursor moves one position in the indicated direction. The cursor will not move beyond the start or end of a line, and will not cause the display to scroll.

ESC Y Pr Pc   Cursor Position

In the above sequence, Pr is the row number and Pc is the column number of the target cursor location. These parameters are formed by adding hexadecimal 1F to the row and column numbers. Row and column numbers are absolute, with row 1, column 20 (Pr = H20, Pc = H3F) representing the upper right corner of the display. The notation Hnn indicates hexadecimal representation. When using the ESC Y command from the TERM, the Pr and Pc values need to be specified by their ASCII values. The ASCII value for 20 hex is the Space and 3F hex is the question mark '?'. The question mark for the TERM-H is <SHIFT>[5,6], and for the TERM-P it's <SHIFT>[6,5].

The same command can be sent from the Galil controller as follows:
MG{P2},{^27},"Y",{^32},{^63},{N}. Thirty-two is the decimal value for 20 hex, and 63 is the decimal value for 3F hex. Also, the {N} is used to suppress the carriage return/line feed after the command is sent. The commas in between the fields are necessary.

ESC H   Cursor Home

Note: This command is functionally equivalent to the Cursor Position command with Pr = H20 and Pc = H20

Erasing Display

ESC E	Clear Display and Home
ESC I	Clear Display
ESC J	Cursor to End of Display
ESC K	Cursor to End of Line
ESC M	Line Containing Cursor

Sounds

ESC T	Short Bell
ESC L	Long Bell
ESC P	Click
ESC Q	Alert

Cursor Style

ESC F	Underscore Cursor On
ESC G	Underscore Cursor Off
ESC R	Blinking Cursor On
ESC S	Blinking Cursor Off

Key Clicks (audible sounds from terminal)

ESC U	Key Click Enable
ESC V	Key Click Disable

Identify (sends "TT1" then terminal firmware version)

ESC Z	Send Terminal ID

Configuration

The key<CNTRL><SHIFT>F1 allows the user to configure the TERM. Follow the display prompts to change configuration. Likewise, the Galil controller's auxiliary serial port is configured with the CC command. See the controller Command Reference for more details on CC.

Recommended TERM Configuration:

Baud Rate	9600
Data bits	7
Parity	Space
Display PE	Enabled
Repeat	Fast
Echo	Disabled
Handshake	Disabled
Self Test	Disabled

Corresponding CC setting: CC 9600,0,0,1

Function Keys

The function commands on the TERM have ASCII decimal values assigned to them. These number assignments are shown below

Default Function Keys

F1 22 decimal

F2 23 decimal

F3 24 decimal

F4 25 decimal

F5 26 decimal

Therefore, to send F1 to the TERM, use the command MG{P2}{^22}

Example 1:

CC 9600,0,0,1	Configures P2
MG{P2} "Hello There", V1{F2.1}	Send message to P2
IN{P2} "Enter Value", NUM	Prompts operator for value

Example 2:

#A	
CI 0,2,1;CC 9600,0,0,1	#A Interrupt on any key; Configure P2
MG {P2} "press F1 to start X”	Print Message to P2
MG {P2} "Press F2 to start Y"	Print Message to P2
#B; JP#B;EN	End Program
#COMINT	Interrupt Routine
JS #XMOVE,P2CH=F1	Jump to X move if F1
JS #YMOVE,P2CH=F2	Jump to Y move if F2
EN0,1	End, restore comm interrupt
#XMOVE;PR1000;BGX;EN	Move X routine
#YMOVE;PR,1000;BGY;EN	Move Y routine

Note 1: F1 through F5 are used as dedicated internal keywords for testing function keys. Do not use these as variables.

Note 2: The syntax for the CI command above is for the DMC-2xx0 controllers only. See the Command Reference of your controller for more information on this command. Also, the ‘Operator Data Entry Mode’ section of the controller manual discusses the use of the CI and P2CH commands.

Pin-Out for TERM-H

6-Pin RJ-11 Connector into the TERM

1 +5 volts

2 Handshake in

3 Handshake out

4 RTS - Input

5 CTS - Output

6 Ground

6-Pin RJ11 Connector into 9-Pin D Adapter

1 Ground

2 Transmit Data - Output

3 Receive Data - Input

4 CTS - output

5 RTS - input

6 5 V

9-Pin D Adapter - Male (For connection to Aux Serial port on controller)

1. CTS - output
2. Transmit Data - output
3. Receive Data - input
4. RTS - output
5. Ground
6. NC
7. NC
8. NC
9. 5 V

Pin-Out for TERM-P

9-Pin D Adapter - Female

1. CTS - output
2. Transmit Data - output
3. Receive Data - input
4. RTS - output
5. Ground
6. NC
7. NC
8. NC
9. 5 V

Note: Out and in are referenced to the TERM-H and TERM-P. For the TERM-P, use a straight through, male to male, RS-232 cable to connect to P2 on the controller.

Ordering Information

TERM-1500H-P2	Hand-held
TERM-1500P-P2	Panel Mount

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

Training Seminars

Galil, a leader in motion control with over 500,000 controllers working worldwide, has a proud reputation for anticipating and setting the trends in motion control. Galil understands your need to keep abreast with these trends in order to remain resourceful and competitive. Through a series of seminars and workshops held over the past 20 years, Galil has actively shared their market insights in a no-nonsense way for a world of engineers on the move. In fact, over 10,000 engineers have attended Galil seminars. The tradition continues with three different seminars, each designed for your particular skill set-from beginner to the most advanced.

MOTION CONTROL MADE EASY

WHO SHOULD ATTEND

Those who need a basic introduction or refresher on how to successfully implement servo motion control systems.

TIME: 4 hours (8:30 am-12:30pm)

ADVANCED MOTION CONTROL

WHO SHOULD ATTEND

Those who consider themselves a “servo specialist” and require an in-depth knowledge of motion control systems to ensure outstanding controller performance. Also, prior completion of “Motion Control Made Easy” or equivalent is required. Analysis and design tools as well as several design examples will be provided.

TIME: 8 hours (8-5pm)

PRODUCT WORKSHOP

WHO SHOULD ATTEND

Current users of Galil motion controllers. Conducted at Galil’s headquarters in Rocklin, CA, students will gain detailed understanding about connecting systems elements, system tuning and motion programming. This is a “hands-on” seminar and students can test their application on actual hardware and review it with Galil specialists.

Attendees must have a current application and recently purchased a Galil controller to attend this course.

TIME: Two days (8:30-4:30pm)

Contacting Us

Galil Motion Control

270 Technology Way

Rocklin, California 95765

Phone: 916-626-0101

Fax: 916-626-0102

Internet address: www.galilmc.com

WARRANTY

All controllers manufactured by Galil Motion Control are warranted against defects in materials and workmanship for a period of 18 months after shipment. Motors, and Power supplies are warranted for 1 year. Extended warranties are available.

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 and for products within warranty.

Call Galil to receive a Return Materials Authorization (RMA) number prior to returning product to Galil.

Any defect in materials or workmanship determined by Galil Motion Control to be attributable to customer alteration, modification, negligence or misuse is not covered by this warranty.

EXCEPT AS SET FORTH ABOVE, GALIL MOTION CONTROL WILL MAKE NO WARRANTIES EITHER EXPRESSED OR IMPLIED, WITH RESPECT TO SUCH PRODUCTS, AND SHALL NOT BE LIABLE OR RESPONSIBLE FOR ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES.

Index