USER MANUAL

DMC-41x3

Manual Rev. 1.3i

Galil Motion Control, Inc.

270 Technology Way
Rocklin, California

916.626.0101
support@galil.com
galil.com

10/2024

Using This Manual

This user manual provides information for proper operation of the DMC-41x3 controller. A separate supplemental manual, the Command Reference, contains a description of the commands available for use with this controller. It is recommended that the user download the latest version of the Command Reference and User Manual from the Galil Website.

http://galil.com/downloads/manuals-and-data-sheets

Your DMC-41x3 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.

Please note that many examples are written for the DMC-4143 4-axis controller or the DMC-4183 8-axis controller. Users of the DMC-4133 3-axis controller, DMC-4123 2-axis controller or DMC-4113 single axis controller should note that the DMC-4133 uses the axes denoted as ABC, the DMC-4123 uses the axes denoted as AB, and the DMC-4113 uses the A-axis only.

Examples for the DMC-4183 denote the axes as A,B,C,D,E,F,G,H. Users of the DMC-4153 5-axis controller, DMC-4163 6-axis controller, or DMC-4173 7-axis controller should note that the DMC-4153 denotes the axes as A,B,C,D,E, the DMC-4163 denotes the axes as A,B,C,D,E,F and the DMC-4173 denotes the axes as A,B,C,D,E,F,G. The axes A,B,C,D may be used interchangeably with X, Y, Z, W.

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

Chapter 1 Overview

Introduction

The DMC-41x3 Series are Galil's Econo motion controller that is a scaled-down version of the DMC-40x0 Accelera series controller. The controller series offers many enhanced features compared to prior generation Econo series controllers including high speed communications, non-volatile program memory, faster encoder speeds, and improved cabling for EMI reduction.

Each DMC-41x3 provides two communication channels: a high speed 100BaseT Ethernet connection and a USB programming port. There is an auxiliary RS-232 port that can be used to communicate to external devices such as HMI's. The controllers allow for high-speed servo control up to 15 million encoder counts/sec and step motor control up to 3 million steps per second. Sample rates as low as 62 µsec per axis are available.

A Flash EEPROM provides non-volatile memory for storing application programs, parameters, arrays and firmware. New firmware revisions are easily upgraded in the field.

The DMC-41x3 is available with up to eight axes in a single stand alone unit. The DMC-4113, 4123, 4133, 4143 are one through four axis controllers and the DMC-4153, 4163, 4173, 4183 are five through eight axis controllers. All eight axes have the ability to use Galil's integrated amplifiers or drivers and connections for integrating external devices.

Designed to solve complex motion problems, the DMC-41x3 can be used for applications involving jogging, point-to-point positioning, vector positioning, electronic gearing, multiple move sequences, contouring and a PVT mode. The controller eliminates jerk by programmable acceleration and deceleration with profile smoothing. For the smooth following of complex contours, the DMC-41x3 can receive a continuous stream of any number of linear and arc segments. The controller also features electronic gearing with multiple Master axes as well as gantry mode operation.

For synchronization with outside events, the DMC-41x3 provides uncommitted I/O including 8 optoisolated digital inputs (16 inputs for DMC-4153 through DMC-4183), 8 digital outputs (16 outputs for the DMC-4153 through DMC-4183), and 8 analog inputs for interface to joysticks, sensors, and pressure transducers. An additional 2 digital inputs (per axis) are available if the auxiliary encoder inputs are not in use. The DMC-41x3 also has digital inputs for home switches, abort signals, and forward and reverse limits (1 each per axis).

Commands are sent in ASCII. Additional software is available for automatic tuning, trajectory viewing on a PC screen, and program development using many environments such as VB.NET, C/C++, Python and others.

Part Numbers

The DMC-41x3 comes with two form factors: the 1-4 axis models (labeled A-D) and 5-8 axis models (labeled E-H). The number of axis is designated by x in the part number DMC-41x3. In addition, Axis A-D and Axis E-H have their own set of axis-specific options that can be ordered. For example, Axis A-D can have a different set of feedback options as Axis E-H even though they reside on the same DMC-41x3 board. The DMC-41x3 can also be ordered with optional internal amplifiers, labeled as AMP or SDM. These amplifiers are “sandwiched” to the back of the DMC-41x3 board. The abstract internal layout of a DMC-41x3 with optional AMP/SDM is shown for 1-4 axis in Figure 1.1.

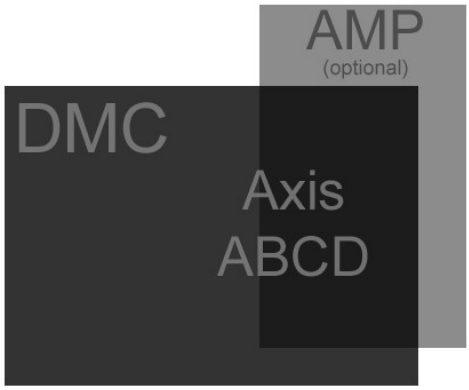


Figure 1.1: Abstract layout of a 1-4 axis DMC-41x3 and optional internal amp

The 5-8 axis models have room for an additional bank of internal amplifiers as shown in Figure 1.2.

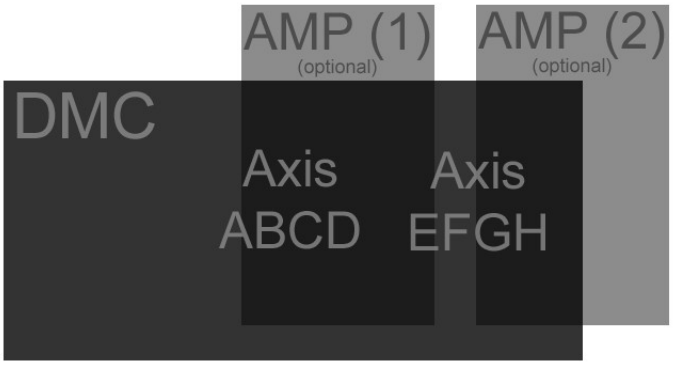


Figure 1.2: Abstract layout of a 5-8 axis DMC-41x3 and optional internal amps

It is important to note that if the DMC-41x3 is ordered *without* optional AMP/SDM options that it can come in either a card or box form factor, ordered as -CARD or -BOXn respectively; where n denotes 4 (for 1-4 axis models) or 8 (for 5-8 axis models). If the DMC-41x3 is ordered *with* an optional AMP/SDM then it *must* be ordered with a -BOXn. Table 1.1 below shows the different form factors and the appropriate part numbers to order them.

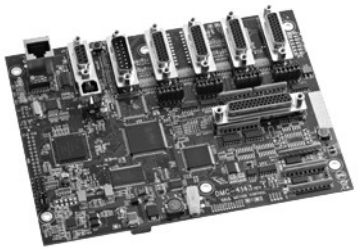
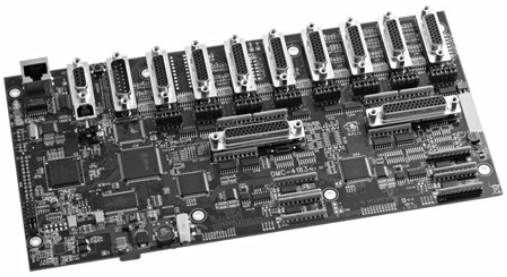
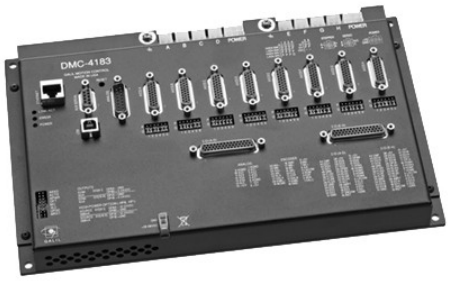
Form Factor (example)	Part Number	Description
	**-CARD**	DMC-41x3, 1-4 axis model ordered with the -CARD option.
	**-CARD**	DMC-41x3, 5-8 axis model ordered with the -CARD option.
	**-BOX4**	DMC-41x3, 1-4 axis model ordered with the -BOX4 option. This option is *required* if the DMC-41x3 is ordered with internal amplifiers, AMP or SDM. This example shows a DMC-41x3 with an internal AMP.
	**-BOX8**	DMC-41x3, 1-4 axis model ordered with the -BOX8 option. This option is *required* if the DMC-41x3 is ordered with internal amplifiers, AMP and/or SDM. This example shows a DMC-41x3 with internal AMPs.

Table 1.1: Explanation of different form factor options for the *DMC-41x3*

The full DMC-41x3 part number is a combination of the DMC controller part number (DMC-41x3), form factor ("-XXXX(Y)), axis-specific options (-ABCD(Y) and -EFGH(Y)), and optional amplifier types (-DXXXX(Y)), where Y is customization options for that specific board or set of axis. The layout of the full DMC-41x3 part number is shown in Figure 1.3 below.

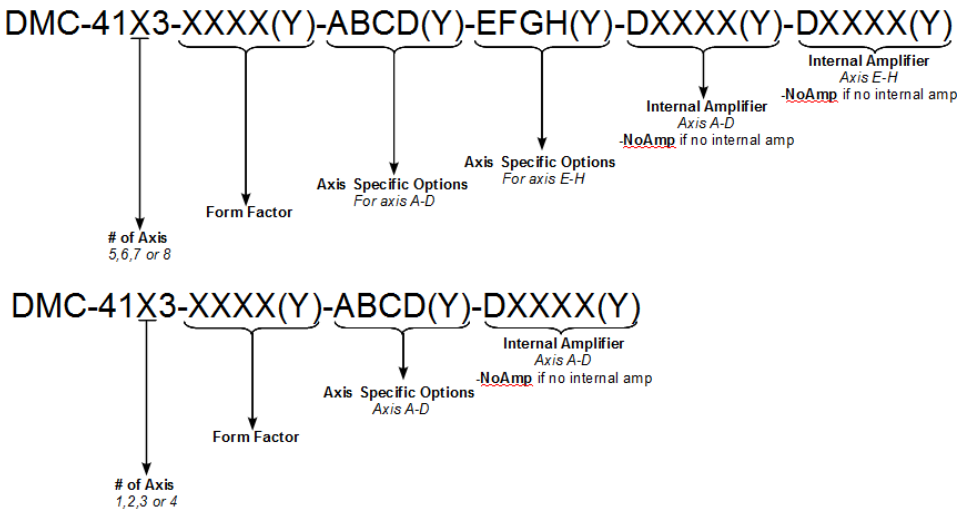


Figure 1.3: Layout of the *full DMC-41x3 part number*

If the part number is not readily available, you can determine the information by using the ID command. Issuing an ID command when connected to the controller will return your controller's internal hardware configuration.
The placement of the AMP/SDM options is extremely important for 5-8 axis models. Reading the DMC-41x3 part number in Figure 1.3 from left to right, the first AMP/SDM (Axis A-D) will be placed in the AMP(1) spot as shown in Figure 1.2 and the second AMP/SDM (Axis E-H) will be placed in the AMP(2) spot.
Many "Y" options can be ordered *per* board or set of axis if separated by a comma. Please use the DMC-41x3 part number generator to check the validity of all part numbers before ordering:
http://galil.com/order/part-number-generator/dmc-41x3

DMC and Form Factor, "DMC-41X3-XXXX(Y)" Options

Option Type	Options	Brief Description	Documentation
X	1, 2, 3, 4, 5, 6, 7, and 8	Number of controller axis	N/A
-XXXX	-CARD	See Table 1.1	See DMC and Form Factor , "DMC-41x3-XXXX(Y)" Options, starting on pg 172
	-BOX4		
	-BOX8		
Y	DIN	DIN Rail Mount clips (-BOXn only)	
	12 VDC	Power Controller with 12 VDC	
	16-bit	16-bit analog inputs	
	4-20mA	4-20mA analog inputs	
	TRES	Encoder terminating resistors	
	ISCNTL	Isolate Controller Power	
	422	RS-422 on Aux Serial Port	
	MO	Motor off jumpers added	

Table 1.2: Brief list of DMC and Form Factor options

Axis-specific options, “-ABCD(Y) and -EFGH(Y)”

Option Type	Options	Brief Description	Documentation
Y	SER	BiSS and SSI feedback	SER – BiSS and SSI Absolute Encoder Option, starting on pg 174
	HSRC	500mA Sourcing Outputs	500mA Sourcing Optoisolated Outputs (HSRC), pg 30
	LSNK	25mA Sinking Outputs	25mA Sinking Optoisolated Outputs (LSNK), pg 28
	LSRC	25mA Sourcing Outputs	25mA Sourcing Optoisolated Outputs (LSRC), pg 29

Table 3: Brief list of axis-specific options

It is important to note that 1-4 axis models come with a single bank (Bank 0) of eight optoisolated specified by -ABCD(Y). 5-8 axis models comes with an *additional* bank (Bank 1) of eight optoisolated outputs specified by -EFGH(Y). See Optoisolated Input Electrical Information, pg 23 for further details.

WARNING	If no option is specified, **the default optoisolated outputs for -ABCD and -EFGH are 4mA sinking**, see 4mA Sinking Optoisolated Outputs (Default), pg 27 for further details.

AMP/SDM, “-DXXXX(Y)” Options

Option Type	Options	Brief Description	Documentation
XXXX	3020/3040	500 W trapazoidal servo drive 2 and 4-axis models	A1 – AMP-430x0 (-D3040,-D3020), pg 189
	3140	20 W brush-type only drive	A2 – AMP-43140 (-D3140), pg 196
	3240	750 W trapazoidal servo drive	A3 – AMP-43240 (-D3240), pg 200
	3520/3540	600 W sinusoidal servo drive 2 and 4-axis models	A4 – AMP-43540 (-D3540, -D3520), pg 207
	3527/3547	600 W sinusoidal servo and microstepping drive	A5 – AMP-43547 (-D3547, -D3527), pg 214
	3640	20 W sinusoidal servo drive	A6 – AMP-43640 (-D3640), pg 222
	4040/4020	1.4 A with 1/16 microstepping drive	A8 – SDM-44040 (-D4040,-D4020), pg 235
	4140	3 A with 1/64 microstepping drive	A9 – SDM-44140 (-D4140), pg 239
Y	ISAMP	Isolates power between amplifiers Two banks of AMP/SDMs required	AMP/SDM, “-DXXXX(Y)” Internal Amplifier Options, starting on pg 175
	SR90	SR-49000 Shunt regulator option	
	SSR	Solid state relay[1]	
	100mA	100mA current -D3140 option only	[1] *Not available for all amplifier options, see* the proper *documentation.*
	HALLF	Filtered hall inputs	

Table 1.4: Brief list of amplifier options

Overview of Motor Types

The DMC-41x3 can provide the following types of motor control:

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

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

Standard Servo Motor with ±10 Volt Command Signal

The DMC-41x3 achieves superior precision through use of a 16-Bit motor command output DAC and a sophisticated PID filter that features velocity and acceleration feed-forward, an extra pole filter and integration limits.
The controller is configured by the factory for standard servo motor operation. In this configuration, the controller provides an analog signal (±10 volts) to connect to a servo amplifier. This connection is described in Chapter 2.

Stepper Motor with Step and Direction Signals

The DMC-41x3 can control stepper motors. In this mode, the controller provides two signals to the driver of 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.
If the stepper motor has encoder feedback, users can run the stepper motor in a closed loop fashion with a 2PB supported amplifier. See the A5 – AMP-43547 (-D3547, -D3527): 600W Configurable Servo Amps/Stepper Drives for more information.
Alternatively, if the stepper motor has encoder feedback, but the user prefers to run the motor in an open-loop configuration, Galil's Stepper Position Maintenance Mode may be used for automatic monitoring and correction of the stepper position. See Stepper Position Maintenance Mode (SPM) in Chapter 6 for more information.

Galil Internal Amplifiers and Drivers

Galil offers a variety of internal servo amplifiers and stepper drivers that are integrated directly into the enclosure of the DMC-41x3. By doing so, Galil can provide a simple, straightforward motion control solution in a single box. This saves space and prevents the hassle of configuring a separate device.
A full list of amplifier specifications and details can be found in the Integrated Components, starting on pg 187.

Overview of External Amplifiers

The amplifiers should be suitable for the motor and may be linear or pulse-width-modulated. An amplifier may have current feedback, voltage feedback or velocity feedback.

Amplifiers in Current Mode

Amplifiers in current mode should accept an analog command signal in the ±10 volt range. The amplifier gain should be set such that a +10V command will generate the maximum required current. For example, if the motor peak current is 10A, the amplifier gain should be 1 A/V.

Amplifiers in Velocity Mode

For velocity mode amplifiers, a command signal of 10 volts should run the motor at the maximum required speed. The velocity gain should be set such that an input signal of 10V runs the motor at the maximum required speed.

Stepper Motor Amplifiers

For step motors, the amplifiers should accept step and direction signals.

DMC-41x3 Functional Elements

The DMC-41x3 circuitry can be divided into the following functional groups as shown in Figure 1.4 and discussed below.

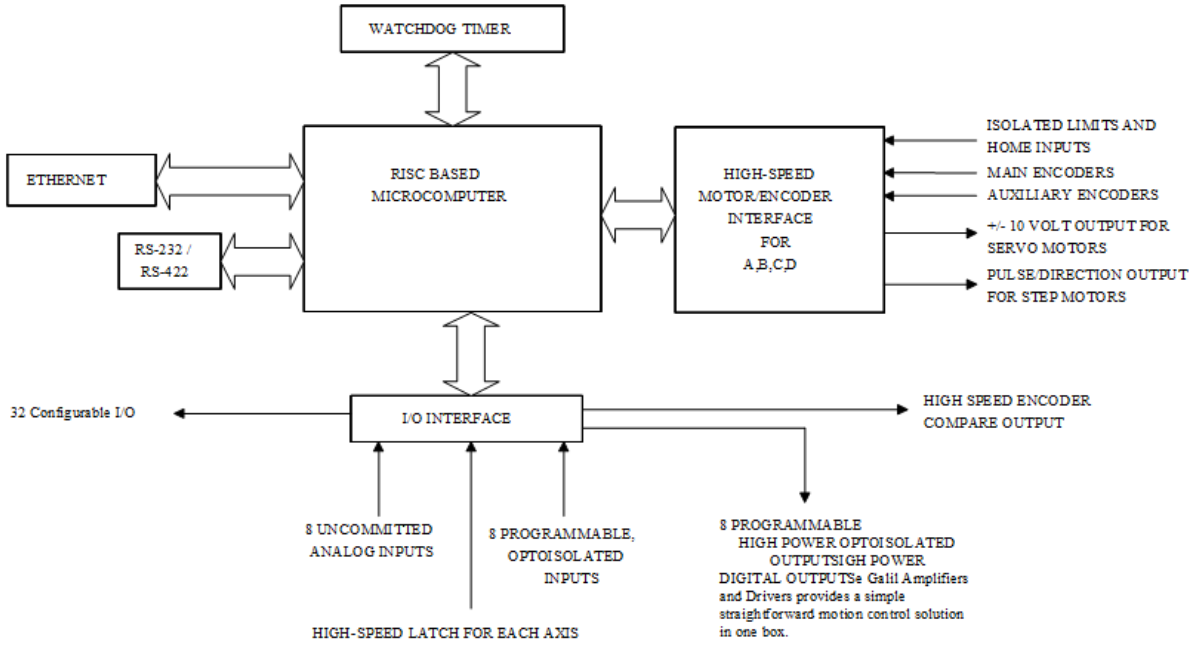


Figure 1.4: DMC-41x3 Functional Elements

Microcomputer Section

The main processing unit of the controller is a specialized Microcomputer with RAM and Flash EEPROM. The RAM provides memory for variables, array elements, and application programs. The flash EEPROM provides non-volatile storage of variables, programs, and arrays. The Flash also contains the firmware of the controller, which is field upgradeable.

Motor Interface

Galil's GL-1800 custom, sub-micron gate array performs quadrature decoding of each encoder at up to 15 MHz. For standard servo operation, the controller generates a ±10 volt analog signal (16 Bit DAC). For stepper motor operation, the controller generates a step and direction signal.

Communication

The communication interface with the DMC-41x3 consists of high speed 100bT Ethernet and a USB programming port.

General I/O

The DMC-41x3 provides interface circuitry for 8 bi-directional, optoisolated inputs, 8 optoisolated outputs and 8 analog inputs with 12-Bit ADC (16-Bit optional). The DMC-4153 through DMC-4183 controller provides an additional 8 optoisolated inputs and 8 optoisolated outputs. Unused auxiliary encoder inputs may also be used as additional inputs (2 inputs / each axis). The general inputs as well as the index pulse can also be used as high speed latches for each axis. A high speed encoder compare output is also provided.

System Elements

As shown in Figure 1.5, the DMC-41x3 is part of a motion control system which includes amplifiers, motors and encoders. These elements are described below.

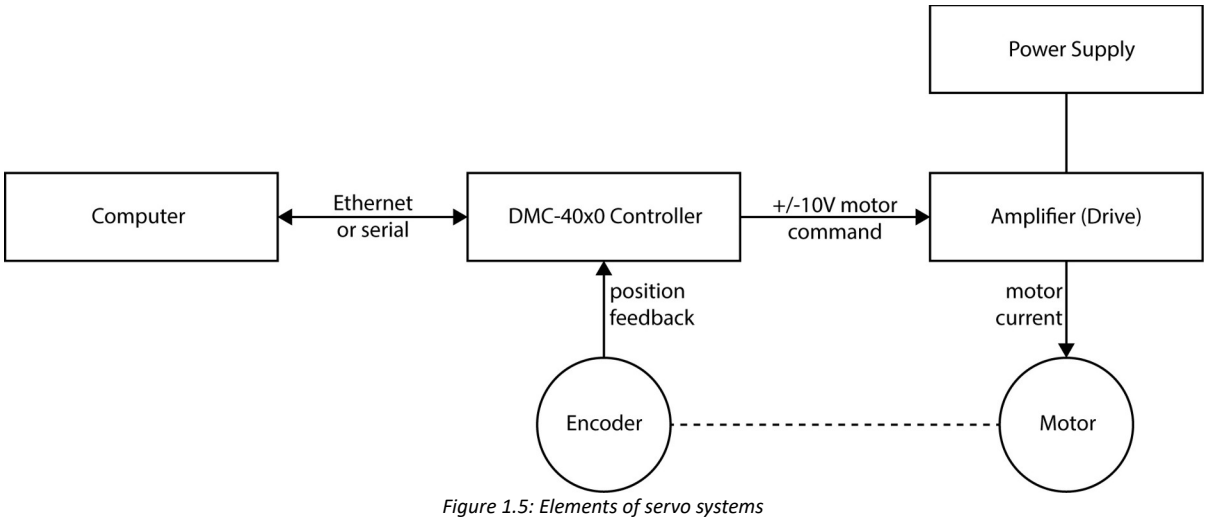


Figure 1.5: Elements of servo systems

Motor

A motor converts current into torque which produces motion. Each axis of motion requires a motor sized properly to move the load at the required speed and acceleration.
The motor may be a step or servo motor and can be brush-type or brushless, rotary or linear. For step motors, the controller can be configured to control full-step, half-step, or microstep drives. An encoder is not required when step motors are used.
Other motors and devices such as Ultrasonic Ceramic motors and voice coils can be controlled with the DMC-41x3.

Amplifier (Driver)

For each axis, the power amplifier converts a ±10 volt signal from the controller into current to drive the motor. For stepper motors, the amplifier converts step and direction signals into current. 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 or the controller must be configured to provide sinusoidal commutation. 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.
Galil offers amplifiers that are integrated into the same enclosure as the DMC-41x3. See the Integrated section in the Appendices or http://galil.com/motion-controllers/multi-axis/dmc-41x3 for more information.

Encoder

An encoder translates motion into electrical pulses which are fed back into the controller. The DMC-41x3 accepts feedback from either a rotary or linear encoder. Typical encoders provide two channels in quadrature, known as MA and MB. This type of encoder is known as a quadrature encoder. Quadrature encoders may be either single-ended (MA and MB) or differential (MA+, MA- and MB+, MB-). The DMC-41x3 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-41x3 can be ordered with 120 Ω termination resistors installed on the encoder inputs. See the Ordering Options in the Appendix for more information.

The DMC-41x3 can also interface to encoders with pulse and direction signals. Refer to the CE command in the command reference for details.

There is no limit on encoder line density; however, the input frequency to the controller must not exceed 3,750,000 full encoder cycles/second (15,000,000 quadrature counts/sec). For example, if the encoder line density is 10,000 cycles per inch, the maximum speed is 375 inches/second. If higher encoder frequency is required, please consult the factory.

The standard encoder voltage level is TTL (0-5v), however, voltage levels up to 12 Volts are acceptable. If using differential signals, 12 Volts can be input directly to the DMC-41x3. Single-ended 12 Volt signals require a bias voltage input to the complementary inputs.

The DMC-41x3 can accept analog feedback (±10v) instead of an encoder for any axis. For more information see the command AF in the command reference.

To interface with other types of position sensors such as absolute encoders, Galil can customize the controller and command set. Please contact Galil to talk to one of our applications engineers about your particular system requirements.

Watch Dog Timer

The DMC-41x3 provides an internal watch dog timer which checks for proper microprocessor operation. The timer toggles the Amplifier Enable Output (AMPEN) which can be used to switch the amplifiers off in the event of a serious DMC-41x3 failure. The AMPEN output is normally high. During power-up and if the microprocessor ceases to function properly, the AMPEN output will go low. The error light will also turn on at this stage. A reset is required to restore the DMC-41x3 to normal operation. Consult the factory for a Return Materials Authorization (RMA) Number if your DMC-41x3 is damaged.

Chapter 2 Getting Started

Elements You Need

For a complete system, Galil recommends the following elements:

1. DMC-41x3 series motion controller
2. Motor Amplifiers (integrated when using Galil amplifiers)
3. Power Supply for controller and amplifiers
4. Brushed or brushless servo motors with encoders or stepper motors
5. Cables for connecting motors and encoders to DMC-41x3
6. PC running Windows 7 or newer (64 bit) or Linux (64 bit)
7. Ethernet cable or USB cable
8. GDK software

GDK is recommended for first time users of the DMC-41x3. It provides interactive instructions for system connection, tuning, and analysis.

Installing the DMC, Amplifiers, and Motors

Installation of a complete, operational motion control system consists of the following steps:

WARNING	All wiring procedures and suggestions mentioned in the following sections should be done with the controller in a powered-off state. Failing to do so can cause harm to the user or to the controller.

NOTE	The following instructions are given for Galil products only. If wiring an non-Galil device, follow the instructions provided with that product. Galil shall not be liable or responsible for any incidental or consequential damages that occur to a 3rd party device.

Step 1. Connecting Encoder Feedback

The type of feedback the controller is capable of interfacing with depends on the additional options ordered for the controller. Table 2.1 shows the different encoder feedback types available for the DMC-41x3 including which options are required. Note that each feedback type has a different configuration command. See the Command Reference for full details on how to properly configure each axis.

NOTE	GDK's Step-By-Step tool provides an interactive walk-through for configuring motor encoders.

Feedback Type	Configuration Command	ICM/Part Number Required	Connection Location	Firmware Required
Standard Quadrature	CE	Standard on all units	Encoder	Standard
Pulse and Direction	CE	Standard on all units	Encoder	Standard
Analog [1]	AF	Standard on all units (12-bit Standard. 16-bit optional)	Analog	Standard
SSI	SI	SER option	Encoder	-SER
BiSS	SS	SER option	Encoder	-SER
None[2]	–	–	–	–
Other	Contact Galil			

Table 2.1: Configuration commands, ICM/Part numbers required for a given feedback type

[1] All wiring/electrical information regarding using analog inputs can be found in Analog Inputs, pg 33.

[2] Although stepper systems do not require feedback, Galil supports a feedback sensor on each stepper axis. Servo motors require a position sensor.

Different feedback types can be used on the same controller. For instance, one axis could be using standard quadrature and the next could be using SSI. By default, all axes are configured for standard quadrature.

When a stepper motor is used, the auxiliary encoder for the corresponding axis is unavailable for an external connection. If an encoder is used for position feedback with a stepper motor, connect the encoder to the main encoder input corresponding to that axis. The commanded position of the stepper can be interrogated with TD and the encoder position can be interrogated with TP.

See SER – BiSS and SSI Absolute Encoder Option, pg 174 for pin-outs and electrical information for SSI and BiSS options.

Encoder pin-outs can be found here:

Encoder 26 pin HD D-Sub Connector (Female), pg 180

Step 2a. Wiring Motors to Galil's Internal Amplifiers

If an external amplifier is being used, proceed to Step 2b. Connecting External Amplifiers and Motors. Table 2.2 below provides a general overview of the connections required for connecting different types of motors to Galil internal amplifiers.

Motor Type	Required Connections
Brushless servo motor	• Power to controller and internal amplifier • Motor power leads to internal amplifiers • Encoder feedback • Hall sensors (Optional for sinusoidal amplifiers)
Brushed servo motor	• Power to controller and internal amplifier • Motor power leads to internal amplifiers • Encoder feedback
Stepper motor	• Power to controller and internal drive • Motor power leads to internal drive • Encoder feedback (optional)

Table 2.2: Synopsis of connections required to connect a motor to Galil's internal amplifiers

Table 2.3 lists each of Galil's internal amplifiers and where to find pin-outs of the amplifier connections and electrical specifications. The commutation method and whether hall sensors are required are also listed.

Amplifier	Commutation	Hall Sensors Required
A1 – AMP-430x0 (-D3040,-D3020), pg 189	Trapezoidal or Brushed	Required for brushless motors
A2 – AMP-43140 (-D3140), pg 196	Brushed	No
A3 – AMP-43240 (-D3240), pg 187	Trapezoidal or Brushed	Required for brushless motors
A4 – AMP-43540 (-D3540, -D3520), pg 207	Sinusoidal or Brushed	Optional for brushless motors
A5 – AMP-43547 (-D3547, -D3527), pg 214	Sinusoidal, Stepper, or Brushed	Optional for brushless motors
A6 – AMP-43640 (-D3640), pg 222	Sinusoidal	Optional for brushless motors
A7 - AMP-43740 (-D3740), pg 228	Sinusoidal	Optional for brushless motors
A8 – SDM-44040 (-D4040,-D4020), pg 235	N/A, Stepper	No
A9 – SDM-44140 (-D4140), pg 239	N/A, Stepper	No

Table 2.3: Amplifier documentation location, commutation method(s), and hall sensor requirements for each internal amplifier.

Pin-outs for the hall sensor inputs can be found in the Encoder 26 pin HD D-Sub Connector (Female) section on page 180.

Step 2b. Connecting External Amplifiers and Motors

System connection procedures will depend on system components and motor types. Any combination of motor types can be used with the DMC-41x3. There can also be a combination of axes running from Galil integrated amplifiers and drivers and external amplifiers or drivers.
Table 2.4 below shows a brief synopsis of the connections required, the full walk through guide is provided below.

Motor Type	Connection Requirements
Servo motors (Brushed and Brushless)	• Power to controller and amplifier • Amplifier Enable • Motor Command • Encoder Feedback • See amplifier documentation for motor connections
Stepper motor	• Power to controller and amplifier • Amplifier enable • Step and Direction lines • Encoder feedback (optional) • See amplifier documentation for motor connections

Table 2.4: Synopsis of connections required to connect an external amplifier

Step A. Connect the motor to the amplifier
Initially do so 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, verify that the ground level of the amplifier is either floating or at the same potential as earth.

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

If you are not sure about the potential of the ground levels, connect the two ground signals (amplifier ground and earth) by a 10 kΩ resistor and measure the voltage across the resistor. Only if the voltage is zero, connect the two ground signals directly.
The amplifier enable signal is defaulted to 24V, high amp enable sinking (the amplifier enable signal will be high when the controller expects the amplifier to be enabled).
Pin-outs for the amplifier enable signal can be found here:
Encoder 26 pin HD D-Sub Connector (Female), pg 180
For full electrical specifications and wiring diagrams refer to:
External Amplifier Interface, pg 34
Once the amplifier enable signal is correctly wired, issuing an MO will disable the amplifier and an SH will enable it.

Step C. Connect the command signals

The DMC-41x3 has two ways of controlling amplifiers:

1. Using a motor command line (±10V analog output). The motor and the amplifier may be configured in torque or velocity mode. In the torque mode, the amplifier gain should be such that a 10V signal generates the maximum required current. In the velocity mode, a command signal of 10V should run the motor at the maximum required speed.
2. Using step (0-5V) and direction (0-5V toggling line), this is referred to as step/dir for short.

The pin-outs for the command signals can be found here:

Encoder 26 pin HD D-Sub Connector (Female), pg 180

See External Amplifier Interface, pg 34 for full electrical specifications.

Step D. Issue the appropriate configuration commands

The Motor Type (MT) command is used to configure each axis for a stepper or servo motor. The Configure Encoder (CE) command is used to configure the encoder for each axis.

Step 3. Power the Controller

WARNING	Exercise caution with the application of this equipment. Only qualified individuals should attempt to install, set up, and operate this equipment. Never open the controller box when DC power is applied.

Table 2.5 below shows which power connectors are required for powering the system based upon the options ordered. If multiple options are ordered, multiple power connections will be required. Different options may effect which connections and what bus voltages are appropriate. If using an internal amplifier, the ISCNTL – Isolate Controller Power, pg 173 option will require multiple connections, one to power the controller board and another to power the amplifiers. If using two banks of amplifiers, the ISAMP – Isolation of Power Between Each Amplifier, pg 175 option will require that the amplifiers are powered independently.

Options Ordered	**Controller Power Connector** (2-pin Molex on side)	**AMP/SDM Power Connector, Axis A-D** (6- or 4-pin Molex)	**AMP/SDM Power Connector, Axis E-H** (6- or 4-pin Molex)
None	X		
ISCNTL [1]	X		
AMP/SDM Axis A-D		X	
AMP/SDM Axis E-H			X
ISAMP [2]		X	X

Table 2.5: Available power connectors based upon options ordered

[1] If the ISCNTL option is ordered with an amplifier, the supplies that power the controller and amplifier need to share the same ground.

[2] If the ISAMP option is not ordered, the amplifiers are sharing power. Their bus voltages and grounds must be from the same source to prevent damage to the controller and amplifiers.

WARNING	Controller or amplifier power should never be plugged in HOT. Always power down the power supply before installing or removing power connector(s) to/from controller or amplifier.

For more information regarding connector type and part numbers see Power Connector Part Numbers, pg 176. The power specifications for the controller are provided in Power Requirements, pg 170 and the power specifications for each amplifier are found under their specific section in the appendix, see Integrated Components, pg 187.
Any emergency stop or disconnect switches should be installed on the AC input to the DC power supply. Relays and/or other switches should not be installed on the DC line between the Galil and the Power supply. An example system is shown in Figure 2.1 with a DMC-4183-D3040-D3040:

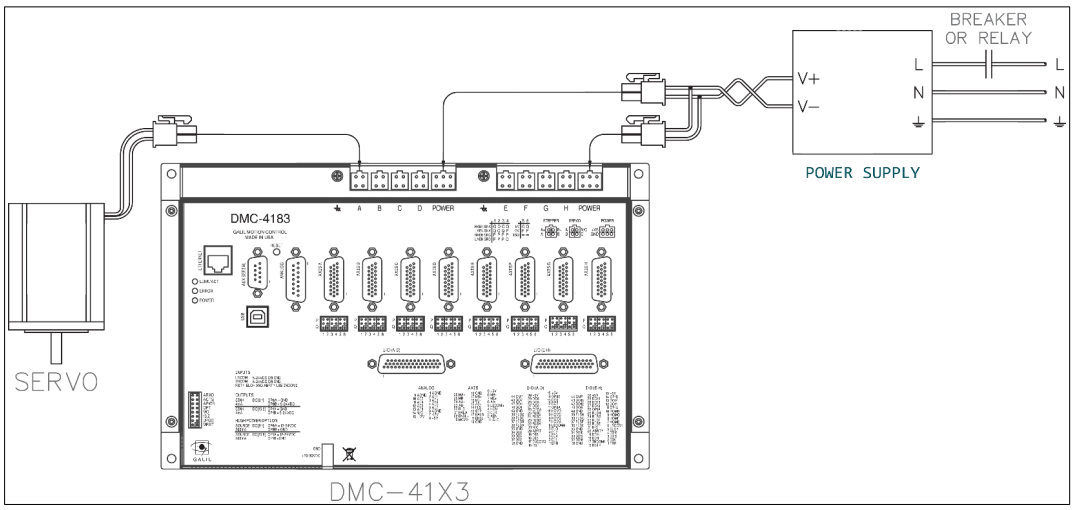


Figure 2.1: Wiring for DMC-4183 with Amplifiers

The green power light indicator should go on when power is applied.

Step 4. Install the Communications Software

After applying power to the controller, a PC is used for programming. Galil's development software enables communication between the controller and the host device. The current version of Galil's development software can be found here: http://galil.com/downloads/software/gdk

Step 5. Establish Communications with Galil Software

NOTE	GDK's Step-By-Step tool provides an interactive and easy to follow guide to setting up a Windows PC for communication with Galil controllers.

GDK will automatically setup the computer for serial communication. For Ethernet communication, see this video for how to configure your NIC card using Windows to connect to a DMC controller:
http://galil.com/learn/online-videos/basics-galil-design-kit-manager-terminal-and-editor
See the GDK manual for using the software to communicate: http://galil.com/sw/pub/all/doc/gdk/man/

Step 6. Motor Setup with GDK Step-By-Step

NOTE	GDK's Step-By-Step tool provides an interactive guide for the automatic set up of brushed, brushless, and stepper motors when using internal amplifiers.

For manual setup instructions, refer to the specific amplifier appendix section which were listed in Table 2.3.

Step 7. Setting Safety Features

Step A. Setting Amplifier Gains

A transconductance (current) amplifier takes a ±10V command signal and produces a current in the motor proportional to that signal. This ratio is the amplifier gain (amps/volts) and can be set via the Amplifier Gain (AG) command when using Galil's internal amplifiers. See the AG command in the Command Reference for proper current gain settings for Galil's internal amplifiers.

The Current Loop Gain (AU) command sets the current loop gain for the amplifier and is set based on the bus voltage powering the amplifier as well as the inductance of the motor. Consult the amplifier appendix section or command reference for more details.

Step B. Setting Torque Limits

The Torque Limit (TL) command will limit the output voltage of ±10V motor command line. This command should be used to avoid excessive torque and speed when initially setting up a servo system. The Peak Torque Limit (TK) command sets the peak torque limit of the motor output allowing the command line to momentarily exceed TL. See the TL and TK settings in the Command Reference for more information. Amplifier gain needs to be taken into account when setting both TL and TK. For example, if a particular motor has a continuous current rating of 2.0 A and a peak current rating of 5.0 A and the amplifier gain is 0.8A/V, TL and TK can be calculated as follows:

TL setting = (2.0A)/(0.8A/V) = 2.5V (TL n=2.5)

TK setting = (5.0A)/(0.8A/V) = 6.25V (TK n=6.25)

The user is responsible for determining the relationship between the motor command line and the amplifier torque/velocity using the documentation of the motor and/or amplifier.

Step C. Setting Error Limits

When an Error Limit (ER) and Off-on-Error (OE) is set, the controller will automatically shut down the motors when excess error (|TE| > ER) has occurred.

OE requires the amplifier enable signal to be connected from the controller to the amplifier. When using Galil's internal amplifiers, the amplifier enable is already configured. See Step 2b. Connecting External Amplifiers and Motors, pg 15 when using external amplifiers.

Step D. Other Safety Features

This section only provides a brief list of safety features that the DMC can provide. Other features include Encoder Failure Detection (OE, OT, OV), Automatic Subroutines to create an automated response to events such as limit switches toggling (**#LIMSWI**), large following error (**#POSERR**), amplifier errors (TA, **#AMPERR**), and more. For a full list of features and how to program each see Chapter 8 Hardware & Software Protection, pg 150.

Step 8. Tune the Servo System

Adjusting the tuning parameters is required when using servo motors. The controller's default set of PID's are not optimized and should not be used in practice.

For the theory of operation and a full explanation of all the PID and other filter parameters, see Chapter 10 Theory of Operation, pg 157.

For additional tuning resources and guides, see the following Application Notes:

Manual Tuning Methods: http://www.galil.com/download/application-note/note3413.pdf

Manual Tuning using the Velocity Zone method: http://www.galil.com/download/application-note/note5491.pdf

For information about the Autotuning Tool in GDK: http://galil.com/sw/pub/all/doc/gdk/man/tuner.html

Chapter 3 Connecting Hardware

Overview

The DMC-41x3 provides optoisolated digital inputs for **forward limit**, **reverse limit, home**, and **abort** signals. The controller also has **8 optoisolated, uncommitted inputs** (for general use) as well as **8 optoisolated outputs** and **8 analog inputs** configured for voltages between ±10 volts. Controllers with 5 or more axes have an additional 8 optoisolated inputs and an additional 8 optoisolated outputs.
This chapter describes the inputs and outputs and their proper connection.

Overview of Optoisolated Inputs

Limit Switch Input

The forward limit switch inhibits motion in the forward direction immediately upon activation of the switch. The reverse limit switch 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 **SD** command. The motor will remain on (in a servo state) after the limit switch has been activated and will hold motor position. The controller can be configured to disable the axis upon the activation of a limit switch, see the **OE** command in the command reference for further detail.
When a forward or reverse limit switch is activated, the current application program that is running in thread 0 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 for Monitoring Conditions are discussed in Chapter 7 Application Programming.

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. Any attempt at further motion before the logic state has been reset will result in the following error: "**22 Begin not possible due to limit switch**" error.

The operands, **_LF**m and **_LR**m, contain the state of the forward and reverse limit switches, respectively (m represents the axis, A, B, C, D etc.). The value of the operand is either a '0' or '1' corresponding to the logic state of the limit switch. Using a terminal program, the state of a limit switch can be printed to the screen with the command, MG_**LF**m or MG_**LR**m. This prints the value of the limit switch operands for the 'm' 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-41x3: Find Edge (FE), Find Index (FI), and Standard Home (HM).

The Find Edge routine is initiated by the command sequence: FEA; BGA. The Find Edge routine will cause the motor to accelerate, and 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. *When using the FE command, 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: FIA; BGA. 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 and then moves back to the index pulse and speed HV. 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 HMA; BGA. 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 HV 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, moves back to the index, and defines this position as 0. The logic state of the Home input can be interrogated with the command MG_HMA. 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.

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.

The effect of an Abort input is dependent on the state of the off-on-error function (OE Command) 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. This can be configured with the CN command. For information see the Command Reference, OE and CN.

ELO (Electronic Lock-Out) Input

Used in conjunction with Galil amplifiers, this input allows the user the shutdown the amplifier at a hardware level. For more detailed information on how specific Galil amplifiers behave when the ELO is triggered, see Integrated Components in the Appendices. The ELO input uses the INCOM pin of the I/O(A-D) connector (INCOM0). If using a 5-8 axis controller with two integrated amplifiers, the ELO input on the I/O(E-H) connector also uses INCOM0. If either ELO input is triggered, both amplifiers will be shut down at a hardware level.

Reset Input/Reset Button

When the Reset line is triggered the controller will be reset. The reset line and reset button will not Master Reset the controller unless the MRST jumper is installed during a controller reset.

Uncommitted Digital Inputs

The DMC-41x3 has 8 optoisolated inputs. These inputs can be read individually using the function @IN[n] where n specifies the input number (1 through 8). These inputs are uncommitted and can allow the user to create conditional statements related to events external to the controller. For example, the user may wish to have the A axis motor move 1000 counts in the positive direction when the logic state of Digital Input 1 goes high.

The Digital inputs can be used as high speed position latch inputs, see High Speed Position Capture (The Latch Function) for more information.

This can be accomplished by connecting a voltage in the range of +5V to +24V into INCOM of the input circuitry from a separate power supply.

Controllers with more than 4 axes have an additional 8 general optoisolated inputs (inputs 9-16). The INCOM for these inputs is found on the I/O (E-H) D-Sub connector.

Optoisolated Input Electrical Information

Electrical Specifications

INCOM/LSCOM Max Voltage	24 V_{DC}
INCOM/LSCOM Min Voltage	0 V_{DC}
Minimum current to turn on Inputs	1.2 mA
Minimum current to turn off Inputs once activated (hysteresis)	0.5 mA
Maximum current per input[1]	11 mA
Internal resistance of inputs	2.2 kΩ

[1] See the Input Current Limitations section on pg. 178 for more details.

The DMC-41x3's optoisolated inputs are rated to operate with a supply voltage of 5-24 VDC. The optoisolated inputs are powered in banks. For example, INCOM (Bank 0), located on the 44-pin I/O (A-D) D-sub connector, provides power to DI[8:1] (digital inputs), the abort input (ABRT), reset (RST), and electric lock-out (ELO). Table 3.1 shows all the input banks power commons and their corresponding inputs for 1-4 axis controllers and Table 3.2 shows the input banks for 5-8 axis controllers.

Common Signal	Common Signal Location	Powers Inputs Labeled
INCOM (Bank 0)	I/O (A-D) D-Sub Connector	DI[8:1], ABRT, RST, ELO
LSCOM (Bank 0)	I/O (A-D) D-Sub Connector	FLSA, RLSA, HOMA FLSB, RLSB, HOMB FLSC, RLSC, HOMC FLSD, RLSD, HOMD

Table 3.1: 1-4 axis controller INCOM and LSCOM banks and corresponding inputs powered

Common Signal	Common Signal Location	Powers Inputs
INCOM (Bank 0)	I/O (A-D) D-Sub Connector	DI[8:1], ABRT, RST, ELO
LSCOM (Bank 0)	I/O (A-D) D-Sub Connector	FLSA, RLSA, HOMA FLSB, RLSB, HOMB FLSC, RLSC, HOMC FLSD, RLSD, HOMD
INCOM (Bank 1)	I/O (E-H) D-Sub Connector	DI[16:9]
LSCOM (Bank 1)	I/O (E-H) D-Sub Connector	FLSE, RLSE, HOME FLSF, RLSF, HOMF FLSG, RLSG, HOMG FLSH, RLSH, HOMH

Table 3.2: 5-8 axis controller INCOM and LSCOM banks and corresponding inputs powered

The full pin-outs for each bank can be found in the Appendices under Pin-outs on page 179.

Wiring the Optoisolated Digital Inputs

To take full advantage of optoisolation, an isolated power supply should be used to provide the voltage at the input common connection. Connecting the ground of the isolated power to the ground of the controller will bypass optoisolation and is not recommended if true optoisolation is desired.

If there is not an isolated supply available, the 5 V_{DC}, 12 V_{DC}, and GND controller references may be used to power INCOM/LSCOM. The current supplied by the controller references are limited, see +5, ±12V Power Output Specifications, pg 170 in the Appendices for electrical specifications. Using the controller reference power completely bypasses optoisolation and is not recommended for most applications.

The optoisolators are bidirectional, meaning current can flow through them in either direction. Connecting $+V_s$ to INCOM/LSCOM will cause current to flow out of the limit switch pin when it is pulled to the isolated ground. Connecting the isolated ground to INCOM/LSCOM will cause current to flow into the limit switch pin when it is pulled up to $+V_s$. Regardless of the direction of current flow, the input is active low – the CPU will read a logical 0 when current is flowing and a logical 1 when no current is flowing.
Figure 3.1 and Figure 3.2 below illustrate the two possible wiring configurations for the limit switches.

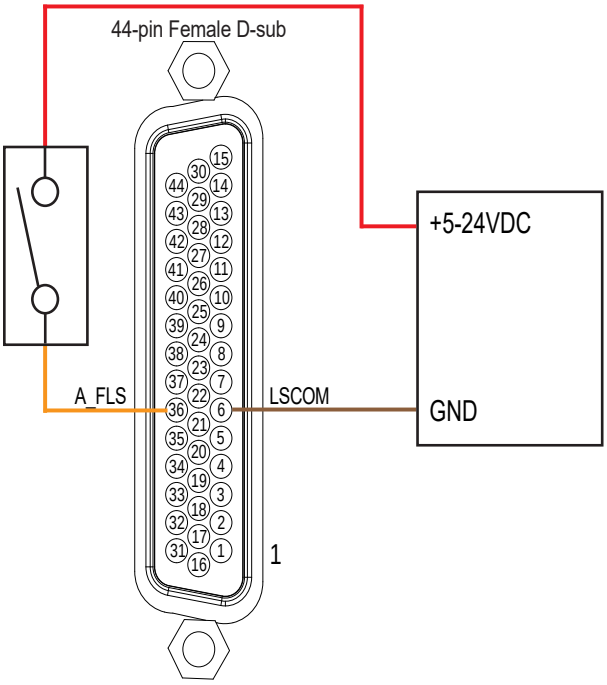


Figure 3.1: Wiring limit switch for sinking current

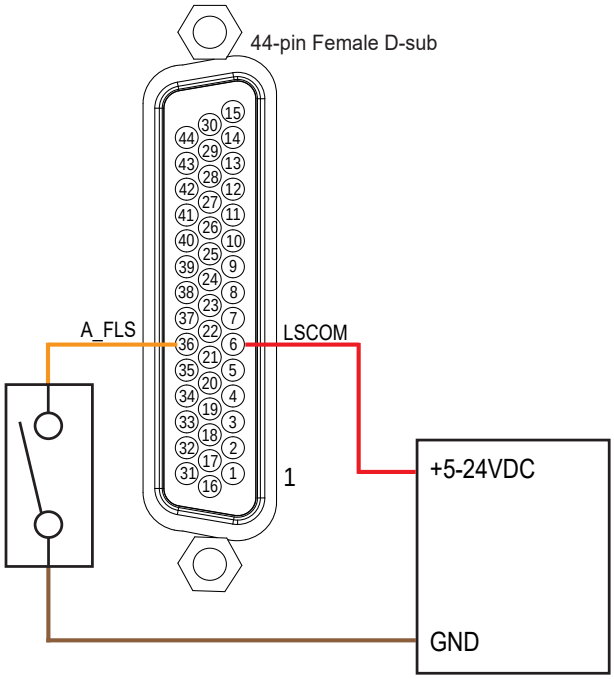


Figure 3.2: Wiring limit switch for sourcing current

Table 3.3 below shows the limit and home switch connections for axes A-D and axes E-H, respectively.

Axes A-D, left 44-pin Female D-sub			Axes E-H, right 44-pin Female D-sub		
Pin	**Label**	**Description**	**Pin**	**Label**	**Description**
6	LSCOM0	Limit Switch Common	6	LSCOM1	Limit Switch Common
7	HOMA	Axis A Home Input	7	HOME	Axis A Home Input
8	HOMB	Axis B Home Input	8	HOMF	Axis B Home Input
9	HOMC	Axis C Home Input	9	HOMG	Axis C Home Input
10	HOMD	Axis D Home Input	10	HOMH	Axis D Home Input
22	RLSA	Axis A Reverse Limit Input	22	RLSE	Axis A Reverse Limit Input
23	RLSB	Axis B Reverse Limit Input	23	RLSF	Axis B Reverse Limit Input
24	RLSC	Axis C Reverse Limit Input	24	RLSG	Axis C Reverse Limit Input
25	RLSD	Axis D Reverse Limit Input	25	RLSH	Axis D Reverse Limit Input
36	FLSA	Axis A Forward Limit Input	36	FLSE	Axis A Forward Limit Input
37	FLSB	Axis B Forward Limit Input	37	FLSF	Axis B Forward Limit Input
38	FLSC	Axis C Forward Limit Input	38	FLSG	Axis C Forward Limit Input
39	FLSD	Axis D Forward Limit Input	39	FLSH	Axis D Forward Limit Input

Table 3.3: Limit and home switch inputs for axes A-D and axes E-H, if present. Right side 44-pin female D-sub is absent on controllers with less than 5 axes.

Figures 3.3-3.7 show the schematics for INCOM/LSCOM (Bank 0) and INCOM/LSCOM (Bank 1) and their corresponding inputs.

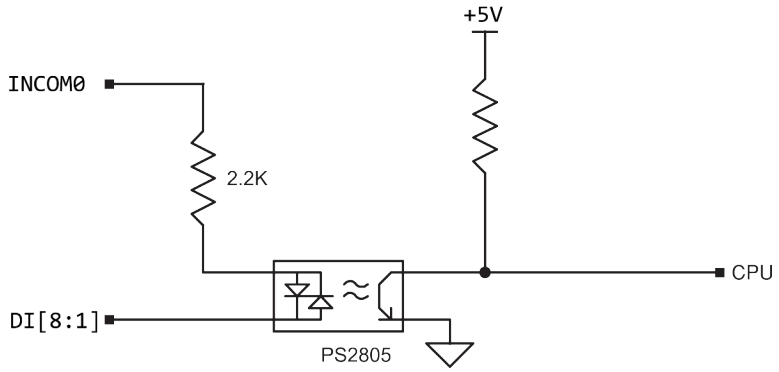


Figure 3.3: Digital Inputs 1-8 (DI[8:1])

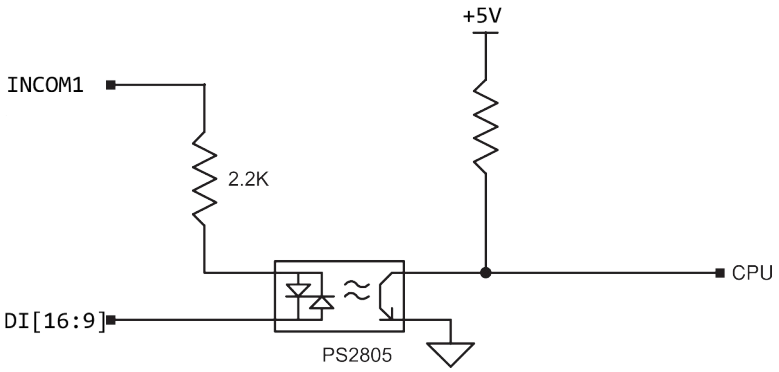


Figure 3.4: Digital Inputs 9-16 (DI[16:9])

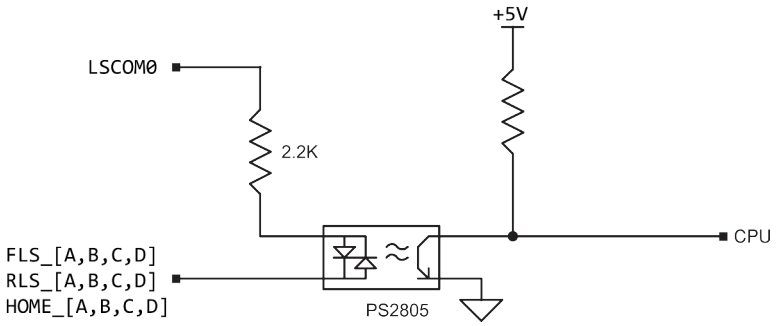


Figure 3.5: Limit Switch Inputs for Axes A-D

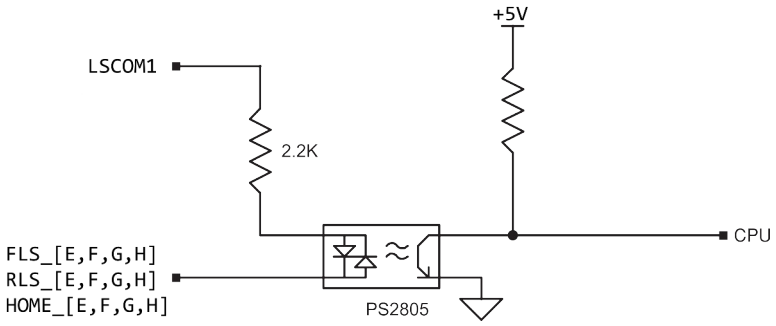


Figure 3.6: Limit Switch Inputs for Axes E-H

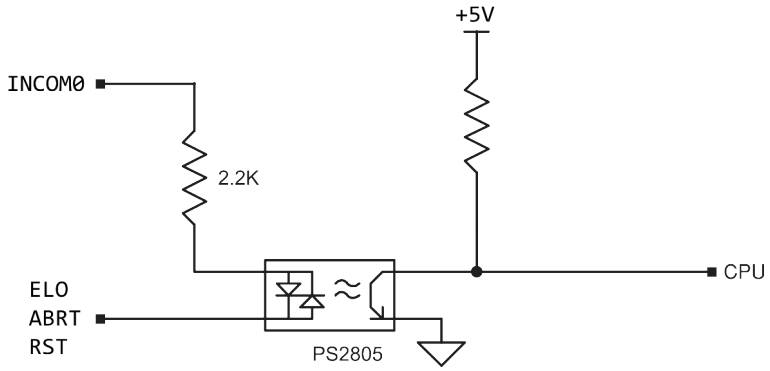


Figure 3.7: ELO, Abort and Reset Inputs

Optoisolated Outputs

The DMC-41x3 has several different options for the uncommitted digital outputs (labeled as DO). The default outputs are 4mA sinking which are ideal for interfacing to TTL level devices. Additional options include 25mA sinking (lower power sinking, LSNK), 25mA sourcing (low power sourcing, LSRC), and 500mA sourcing outputs (high power sourcing, HSRC). Please refer to your part number to determine which option you have.

The amount of uncommitted, optoisolated outputs the DMC-41x3 has depends on the number of axis. For instance, 1-4 axis models come with a single bank of 8 outputs, Bank 0 (DO[8:1]). 5-8 axis models come with an additional bank of 8 outputs, Bank 1 (DO[16:9]), for a total of 16 outputs.

The wiring pins for Bank 0 are located on I/O (A-D) 44 pin HD D-Sub Connector (Female) and the pins for wiring Bank 1 are located on I/O (E-H) 44 pin HD D-Sub Connector (Female). See the Pin-outs in the Appendix for more details.

Wiring diagrams, electrical specifications, and details for each bank and output type are provided below.

4mA Sinking Optoisolated Outputs (Default)

Description

The default outputs of the DMC-41x3 are capable of 4mA and are configured as sinking outputs. The voltage range for the outputs is 5-24 VDC. These outputs should not be used to drive inductive loads directly.

Electrical Specifications

Output PWR Max Voltage	24 V_{DC}
Output PWR Min Voltage	5 V_{DC}
ON Voltage (No Load, Output PWR= 5V_{DC})	0.1 V_{DC}
Max Drive Current per Output	4mA, Sinking

Wiring the 4mA Sinking outputs

With this configuration, the output power supply will be connected to Output PWR (labeled OPnB) and the power supply return will be connected to Output GND (labeled OPnA), where n denotes 0 or 1 referring to Bank 0 and Bank 1 respectively. Note that the load is wired between Output PWR and DO. The wiring diagram for Bank 0 is shown in Figure 3.9 and Bank 1 in Figure 3.8. Refer to Pin-outs in the Appendix for pin-out information.

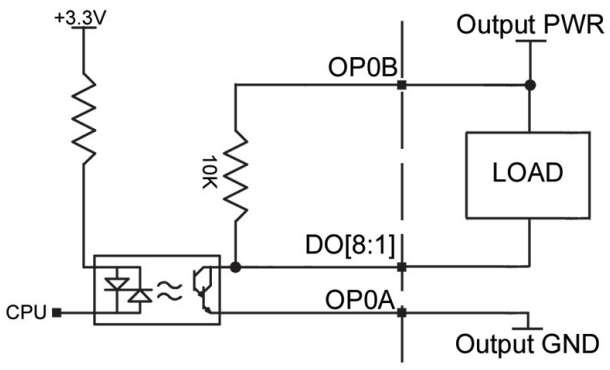


Figure 3.9: 4mA sinking wiring diagram for Bank 0, DO[8:1]

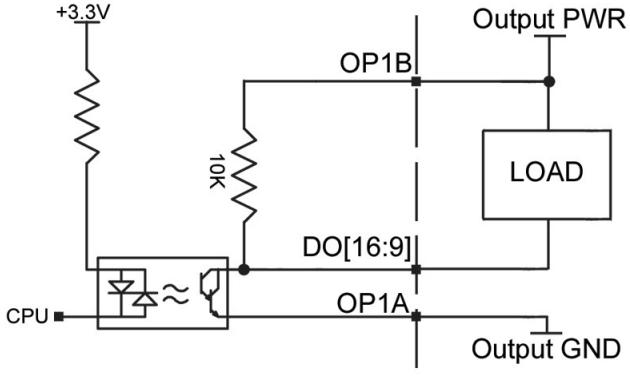


Figure 3.8: 4mA sinking wiring diagram for Bank 1, DO[16:9]

25mA Sinking Optoisolated Outputs (LSNK)

Description

The 25mA sinking option, referred to as lower power sinking (LSNK), are capable of sinking up to 25mA per output. The voltage range for the outputs is 5-24 VDC. These outputs should not be used to drive inductive loads directly.

Electrical Specifications

Output PWR Max Voltage	24 V_{DC}
Output PWR Min Voltage	5 V_{DC}
ON Voltage (No Load, Output PWR= 5V_{DC})	1.2 V_{DC}
Max Drive Current per Output	25mA, Sinking

Wiring the 25mA Sinking Outputs

When wiring the 25mA sinking outputs, the load is wired in the same fashion as the 4mA sinking outputs: The output power supply will be connected to Output PWR (labeled OPnB) and the power supply return will be connected to Output GND (labeled OPnA), where n denotes 0 or 1 referring to Bank 0 and Bank 1 respectively. Note that the load is wired between Output PWR and DO. The wiring diagram for Bank 0 is shown in Figure 3.10 and Bank 1 in Figure 3.11. Refer to Pin-outs in the Appendix for pin-out information.

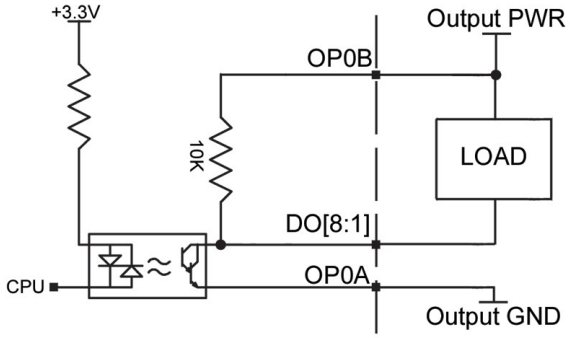


Figure 3.10: 25mA sinking wiring diagram for Bank 0, DO[8:1]

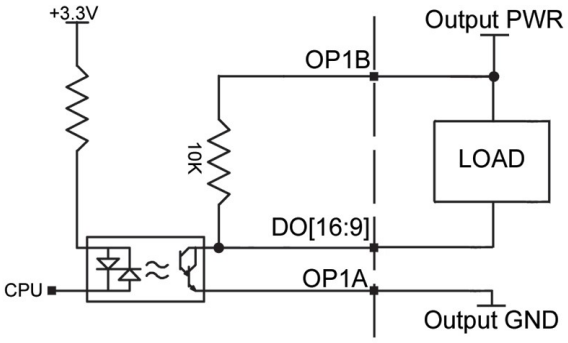


Figure 3.11: 25mA sinking wiring diagram for Bank 1, DO[16:9]

25mA Sourcing Optoisolated Outputs (LSRC)

Description

The 25mA sourcing option, referred to as lower power sourcing (LSRC), are capable of sourcing up to 25mA per output. The voltage range for the outputs is 5-24 VDC. These outputs should not be used to drive inductive loads directly.

Electrical Specifications

Output PWR Max Voltage	24 V_{DC}
Output PWR Min Voltage	5 V_{DC}
Max Drive Current per Output	25mA, Sourcing

Wiring the 25mA Sourcing Outputs

With this configuration, the output power supply will be connected to Output PWR (labeled OPnA) and the power supply return will be connected to Output GND (labeled OPnB), where n denotes 0 or 1 referring to Bank 0 and Bank 1 respectively. Note that the load is wired between DO and Output GND. The wiring diagram for Bank 0 is shown in Figure 3.12 and Bank 1 in Figure 3.13. Refer to Pin-outs in the Appendix for pin-out information.

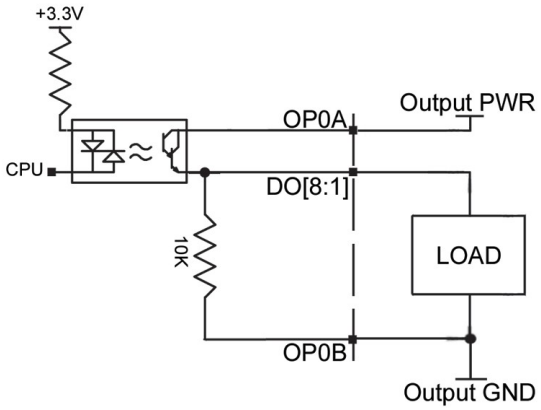


Figure 3.12: 25mA sourcing wiring diagram for Bank 0, DO[8:1]

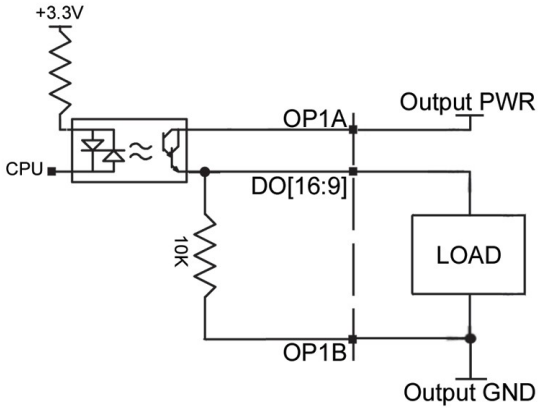


Figure 3.13: 25mA sourcing wiring diagram for Bank 1, DO[16:9]

500mA Sourcing Optoisolated Outputs (HSRC)

Description

The 500mA sourcing option, referred to as high power sourcing (HSRC), is capable of sourcing up to 500mA per output and up to 3A per **bank**. The voltage range for the outputs is 12-24 V_{DC}. These outputs are capable of driving inductive loads such as solenoids or relays. The outputs are configured for hi-side (sourcing) only.

Electrical Specifications

Output PWR Max Voltage	24 V_{DC}
Output PWR Min Voltage	12 V_{DC}
Max Drive Current per Output	0.5 A (not to exceed 3A per Bank)

Wiring the 500mA Sourcing Optoisolated Outputs

With this configuration, the output power supply will be connected to Output PWR (labeled OPnA) and the power supply return will be connected to Output GND (labeled OPnB), where n denotes 0 or 1 referring to Bank 0 and Bank 1 respectively. Note that the load is wired between DO and Output GND. The wiring diagram for Bank 0 is shown in Figure 3.14 and Bank 1 in Figure 3.15. Refer to Pin-outs in the Appendix for pin-out information.

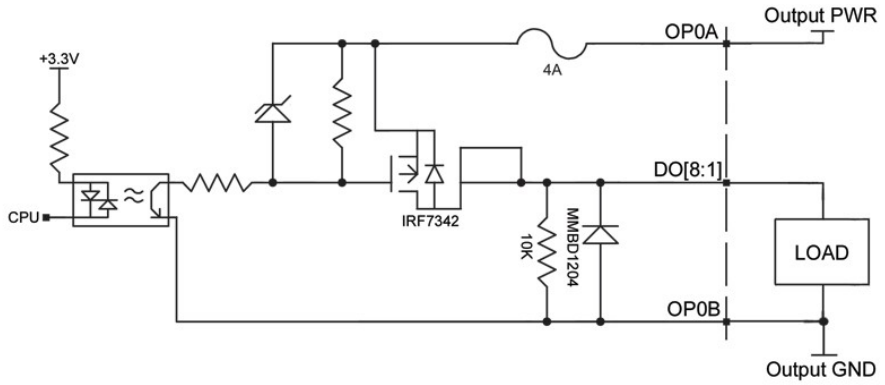


Figure 3.14: 500mA sourcing wiring diagrams for Bank 0, DO[8:1]

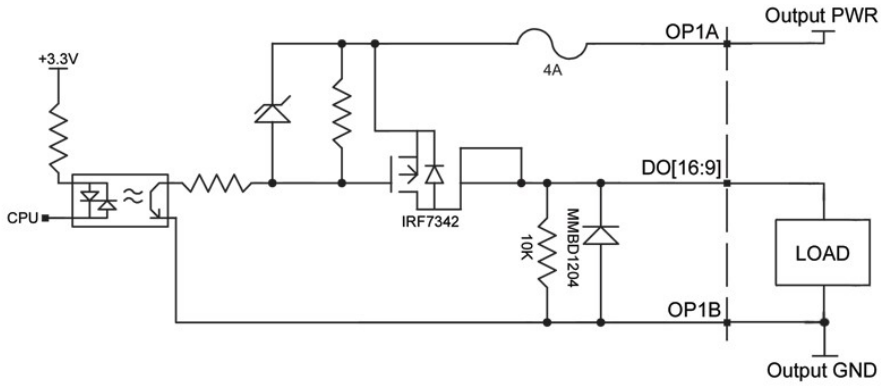


Figure 3.15: 500mA sourcing wiring diagram for Bank 1, DO[16:9]

TTL Inputs and Outputs

Main Encoder Inputs

The main encoder inputs can be configured for quadrature (default) or pulse and direction inputs. This configuration is set through the CE command. The encoder connections are found on the HD D-sub Encoder connectors and are labeled MA+, MA-, MB+, MB-. The '-' (negative) inputs are the differential inputs to the encoder inputs; if the encoder is a single ended 5V encoder, then the negative input should be left floating. If the encoder is a single ended and outputs a 0-12V signal then the negative input should be tied to the 5V line on the DMC-41x3. When the encoders are setup as step and direction inputs the MA channel will be the step or pulse input, and the MB channel will be the direction input.
The encoder inputs can be ordered with 120 Ω termination resistors installed. See TRES – Encoder Termination Resistors, pg 173 in the Appendix for more information.
If only a main encoder is used, the encoder should be on the motor shaft, not the load. Using only a load encoder may result in an unstable system and inefficient commutation when using sine amps.

Electrical Specifications

Maximum Voltage	12 V_{DC}
Minimum Voltage	-12 V_{DC}
Maximum Frequency (Quadrature)	15 MHz

'+' inputs are internally pulled-up to 5V through a 4.7 kΩ resistor
'-' inputs are internally biased to ~1.3V
pulled up to 5V through a 7.1 kΩ resistor
pulled down to GND through a 2.5 kΩ resistor

The Auxiliary Encoder Inputs

The auxiliary encoder inputs can be used for general use. For each axis, the controller has one auxiliary encoder and each auxiliary encoder consists of two inputs, channel A and channel B. The auxiliary encoder inputs are mapped to the inputs 81-96. The Aux encoder inputs are not available for any axis that is configured for step and direction outputs (stepper).
Each input from the auxiliary encoder is a differential line receiver and can accept voltage levels between ±12V. The inputs have been configured to accept TTL level signals. To connect TTL signals, simply connect the signal to the + input and leave the - input disconnected. For other signal levels, the '-' input should be connected to a voltage that is ~½ of the full voltage range (for example, connect the '-' input to the 5 volts on the Galil if the signal is 0 - 12V logic)

Electrical Specifications

Maximum Voltage	12 V_{DC}
Minimum Voltage	-12 V_{DC}

'+' inputs are internally pulled-up to 5V through a 4.7kΩ resistor
'-' inputs are internally biased to ~1.3V
pulled up to 5V through a 7.1kΩ resistor
pulled down to GND through a 2.5kΩ resistor

Example:
A DMC-4113 has one auxiliary encoder. This encoder has two inputs (channel A and channel B). Channel A input is mapped to input 81 and Channel B input is mapped to input 82. To use this input for 2 TTL signals, the first signal will be connected to AA+ and the second to AB+. AA- and AB- will be left unconnected. To access this input, use the function `@IN[81]` and `@IN[82]`.

NOTE: The auxiliary encoder inputs are not available for any axis that is configured for stepper motor.

Output Compare

The output compare signal is a TTL output signal and is available on the I/O (A-D) D-Sub connector labeled as CMP. An additional output compare signal is available for 5-8 axes controllers on the I/O (E-H) D-sub connector.

Output compare is controlled by the position of any of the main encoder inputs on the controller. The output can be programmed to produce either a brief, active low pulse (510 nsec) based on an incremental encoder value or to activate once ("one shot") when an axis position has been passed. When setup for a one shot, the output will stay low until the OC command is called again. For further information, see the command OC in the Command Reference.

NOTE	Output compare is not valid with sampled feedback types such as: SSI, BiSS, and Analog

Electrical Specifications

Output Voltage	0 – 5 VDC
Current Output	20 mA Sink/Source

Error Output

The controller provides a TTL signal, ERR, to indicate a controller error condition. When an error condition occurs, the ERR signal will go low and the controller LED will go on. An error occurs because of one of the following conditions:

1. At least one axis has a position error greater than the error limit. The error limit is set by using the command OC.
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 ERR signal is found on the I/O (A-D) D-Sub connector. For controllers with 5-8 axes, the ERR signal is duplicated on the I/O (E-H) D-Sub connector.
For additional information see Error Light (Red LED) in Chapter 9 Troubleshooting.

Electrical Specifications

Output Voltage	0 – 5 VDC
Current Output	20 mA Sink/Source

Analog Inputs

The DMC-41x3 has eight analog inputs configured for the range between -10V and 10V. The inputs are decoded by a 12-bit A/D decoder giving a voltage resolution of approximately .005V. A 16-bit ADC is available as an option (Example: *DMC-4123-CARD(-16bit)*). The analog inputs are specified as @AN[**n**] where **n** is a number 1 through 8.

AQ settings

The analog inputs can be set to a range of ±10V, ±5V, 0-5V or 0-10V, this allows for increased resolution when the full ±10V is not required. The inputs can also be set into a differential mode where analog inputs 2,4,6 and 8 can be set to the negative differential inputs for analog inputs 1,3,5 and 7 respectively. See the AQ command in the command reference for more information.

Electrical Specifications

Input Impedance (12 and 16 bit) –

Unipolar (0-5V, 0-10V)	42kΩ
Bipolar (±5V, ±10V)	31kΩ

External Amplifier Interface

External Stepper Control

The controller provides step and direction (STPn, DIRn) outputs for every axis available on the controller. Step and direction outputs need to be wired with respect to digital ground (GND). See the MT command for more details.

Step and Direction Electrical Specifications

Output Voltage	0 – 5 VDC
Current Output	20 mA Sink/Source

External Servo Control

The DMC-41x3 command voltage ranges between ±10V and is output on the motor command line - MCMn (where n is A-H). This signal, along with GND, provides the input to the motor amplifiers. The amplifiers must be sized to drive the motors and load. For best performance, the amplifiers should be configured for a torque (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.

Motor Command Line Electrical Specifications

Output Voltage	±10 VDC
Motor Command Output Impedance	500 Ω

The standard configuration of the amplifier enable signal is external voltage with active high amp enable (HAEN) sinking. 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 by configuring the Amplifier Enable Circuit on the DMC-41x3.

Note: Many amplifiers designate the enable input as ‘inhibit’.

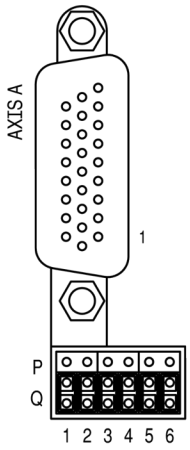
Amplifier Enable

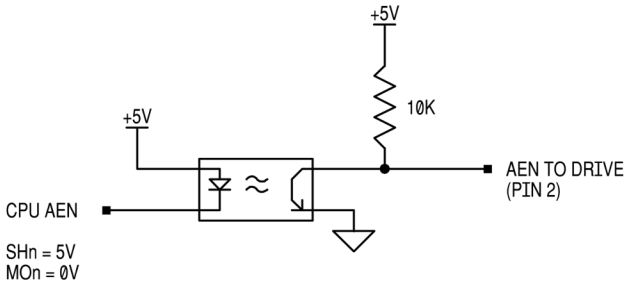
This section describes how to configure the DMC-41x3 for different Amplifier Enable configurations.

The default configuration of the amplifier enable signal is external voltage with active high amp enable (HAEN) sinking. In other words, the AEN signal will be high when the controller expects the amplifier to be enabled.

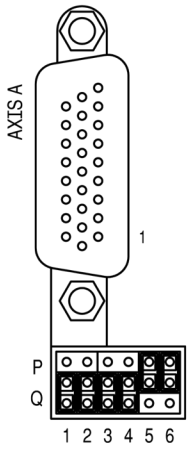
The DMC-41x3 is designed to be easily interfaced to multiple amplifier manufactures. As a result, the amplifier enable circuit for each axis is individually configurable through jumper settings. The user can choose between High-Amp-Enable (HAEN), Low-Amp-Enable (LAEN), 5V logic, 12V logic, external voltage supplies up to 24V, sinking, or sourcing. The different configurations are described below with jumper settings and a basic schematics of the circuit.

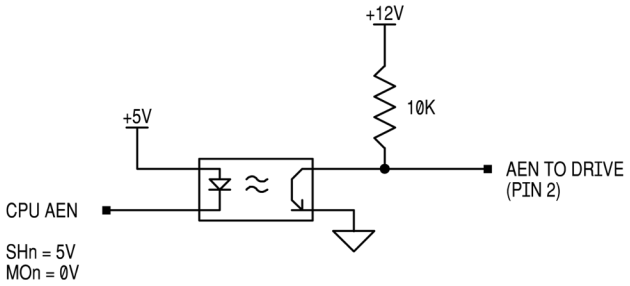
+5V
HIGH AMP ENABLE
SINKING



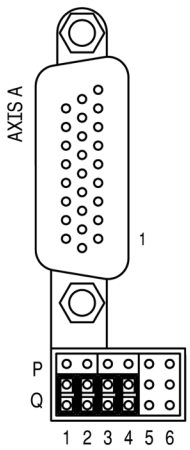


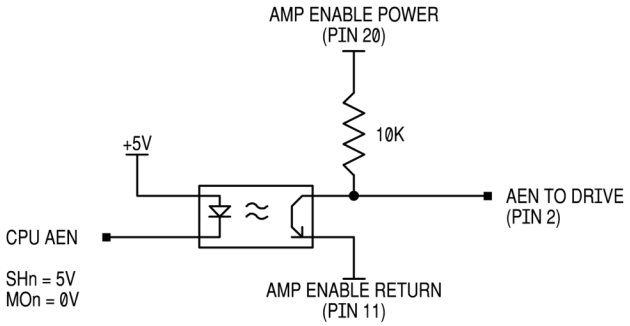
+12V
HIGH AMP ENABLE
SINKING



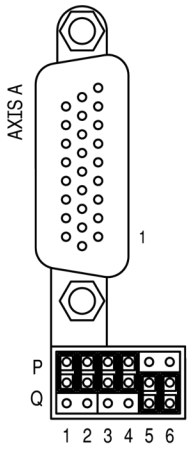


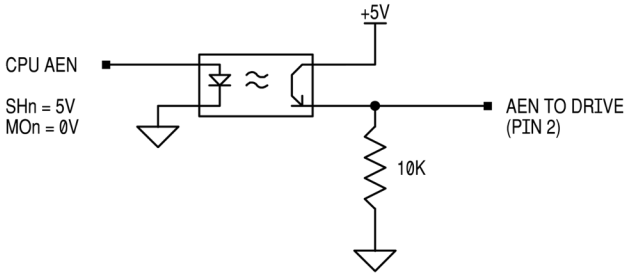
ISOLATED SUPPLY
HIGH AMP ENABLE
SINKING



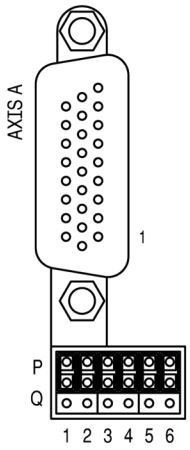


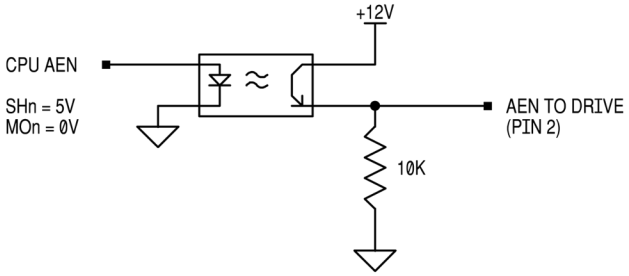
+5V
HIGH AMP ENABLE
SOURCING



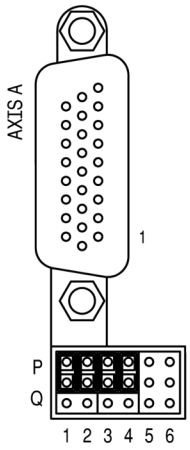


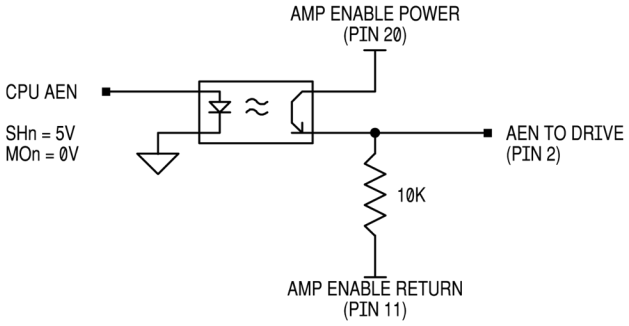
+12V
HIGH AMP ENABLE
SOURCING



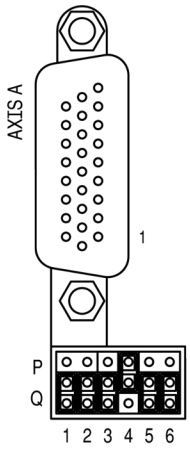


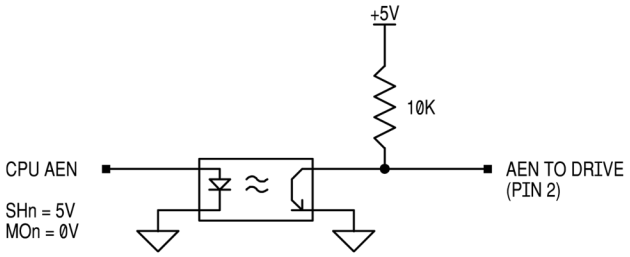
ISOLATED SUPPLY
HIGH AMP ENABLE
SOURCING



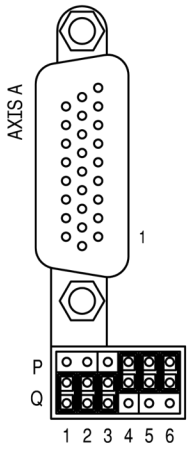


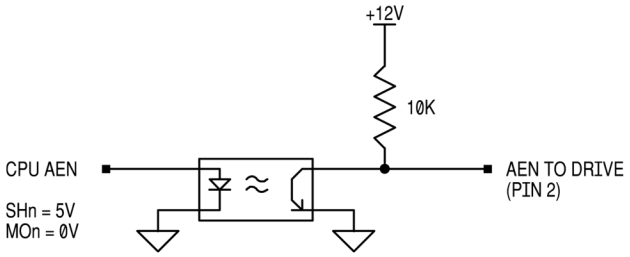
+5V
LOW AMP ENABLE
SINKING



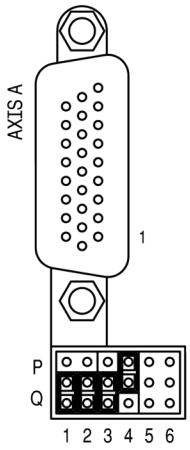


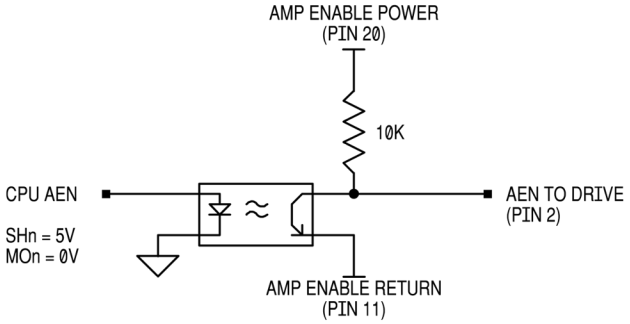
+12V
LOW AMP ENABLE
SINKING



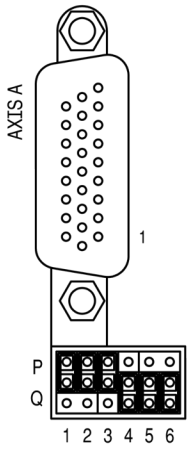


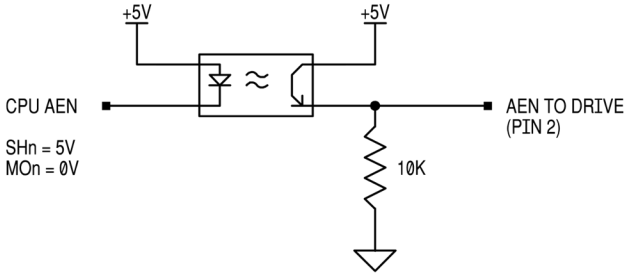
ISOLATED SUPPLY
LOW AMP ENABLE
SINKING



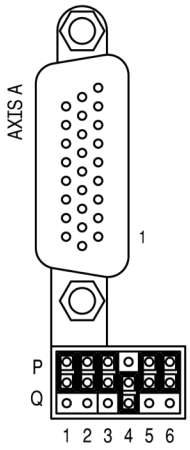


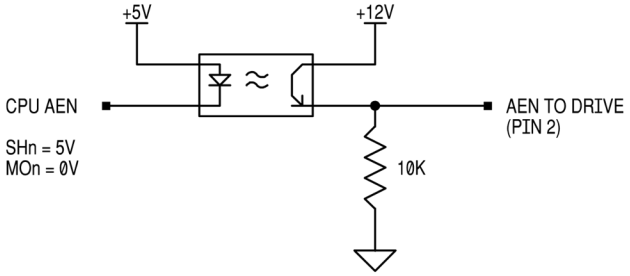
+5V
LOW AMP ENABLE
SOURCING



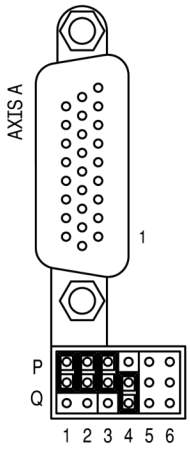


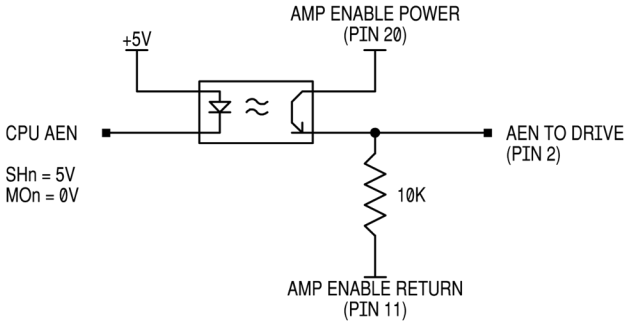
+12V
LOW AMP ENABLE
SOURCING





ISOLATED SUPPLY
LOW AMP ENABLE
SOURCING





Chapter 4 Software Tools and Communication

Introduction

The default configuration DMC-41x3 has one USB port, one RS-232 port and one Ethernet port. The auxiliary RS-232 port is the data term and can be configured with the software command CC. This configuration can be saved using the Burn (BN) instruction. The Ethernet port is a 10/100BASE-T connection that auto-negotiates the speed and half or full duplex.

The GDK software package is available for PC computers running Microsoft Windows or Linux to communicate with the DMC-41x3 controller. This software package has been developed to operate under Windows and Linux, and include all the necessary drivers to communicate to the controller. In addition, gclib, a C based software development communication library, is available which allows users to create their own application interfaces using programming environments such as C/C++, Python, .NET, and Java.

The following sections in this chapter are a description of the communications protocol, and a brief introduction to the software tools and communication techniques used by Galil. At the application level, GDK is the basic development software that the majority of users will need to communicate with the controller, to perform basic setup, and to develop application code (.dmc programs) that is downloaded to the controller. At the Galil API level, gclib is available for users who wish to develop their own custom application programs to communicate to the controller. Custom application programs can utilize API function calls directly to our DLL's. At the driver level, we provide fundamental hardware interface information for users who desire to create their own drivers.

Controller Response to Commands

Most DMC-41x3 instructions are represented by two characters followed by the appropriate parameters. Each instruction must be terminated by a carriage return. Multiple commands may be concatenated by inserting a semicolon between each command.

After the instruction is decoded, the DMC-41x3 returns a response to the port from which the command was generated. If the instruction was valid, the controller returns a colon (**:**) or the controller will respond with a question mark (**?**) if the instruction was not valid. For example, the controller will respond to commands which are sent via the USB port back through the USB port, and to commands which are sent via the Ethernet port back through the Ethernet port.

For instructions that return data, such as Tell Position (TP), the DMC-41x3 will return the data followed by a carriage return, line feed and colon.

It is good practice to check for colon after each command is sent to prevent errors. An echo function is provided to enable associating the DMC-41x3 response with the data sent. The echo is enabled by sending the command EO**1** to the controller.

Unsolicited Messages Generated by Controller

When the controller is executing a program, it may generate responses which will be sent via the USB port or Ethernet handles. This response could be generated as a result of messages using the MG command, or it could be the 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 the MG and CF commands in the Command Reference. If the port is not explicitly given or the default is not changed with the CF command, unsolicited messages will be sent to the default port. The default port is the USB port. When communicating via an Ethernet connection, the unsolicited messages must be sent through a handle that is not the main communication handle from the host. The GDK software automatically establishes this second communication handle.

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, CW**1** 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, CW**2**. For more information, see the CW command in the Command Reference.

When handshaking is used (hardware and/or software handshaking) 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 512 bytes. When this buffer becomes full, the controller must either stop executing commands or ignore additional characters generated for output. The command CW,**2** 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.

USB and RS-232 Ports

The USB port by default is an interpreted serial communication port for the DMC-41x3 and will receive and respond to DMC commands. The RS232 port is the auxiliary port by default and is used to connect to external devices that do not use DMC code, such as bar code readers or other RS232 sensors and displays. The controller can read and write generic data but the user must write their own communication routines. A full description of the USB and RS232 default configuration settings can be found in the following sections.

The US command can reverse the main and auxiliary port such that the RS232 port would interpret DMC commands and the USB port would be available for generic data. The following are several points to consider when switching the USB and RS232 ports with US.

- A null modem cable is required for interfacing to the RS232 port. Baud rates are set via the controller's jumpers.
- Firmware cannot be loaded from the RS232 port. The user must switch back to USB mode to load firmware via serial. Ethernet firmware updates are supported in both modes.
- When using the USB port in swapped mode, the remote device must interface to the USB to serial converter on the controller. On a PC this is done with a device driver. The USB to serial UART is an FT232 from FTDI.

For more information, see the US command in the DMC-41x3 command reference.

USB Port

The USB port on the DMC-41x3 is a USB to serial converter. It should be setup for 115.2kB, 8 Data bits, No Parity, 1 Stop Bit and Flow Control set for Hardware. The baud rate can be changed to 19200 baud by installing the 19.2 jumper on JP1, but this configuration is only recommended if a slower baud rate is required from the host communication. The USB port on the DMC-41x3 is a Female Type B USB port. The standard cable when communicating to a PC will be a Male Type A – Male Type B USB cable.
When connected to a PC, the USB connection will be available as a new serial port connection (ex. with GDK "COM3 115200").
The USB port is not recommended when using the GDK Scope. In this case the Ethernet connection is advised for higher performance.

Baud Rate Selection

JP1 JUMPER SETTINGS	
19.2	**BAUD RATE**
ON	19200
OFF (recommended)	115200

USB Driver

The USB port on the DMC-41x3 utilizes a USB to serial converter. The driver for this device is expected to be loaded automatically upon connection of the controller in most OS's. If the driver does not load, or is not installed automatically, it can be downloaded from the mfg website here:
http://www.ftdichip.com/Drivers/VCP.htm

RS-232 Port

The main purpose of the auxiliary RS232 port is to connect to external devices that cannot use DMC code to communicate. It is important to note that the Aux port is not an interpreted port and cannot receive DMC Galil commands directly. Instead, use CI, #COMINT, and the P2 operands to handle received data on this port.

Note: If you are connecting the RS-232 auxiliary port to a terminal or any device which is a DATASET, it is necessary to use a connector adapter, which changes a dataset to a dataterm. This cable is also known as a 'null' modem cable.

CC Command

The CC, or Configure Communications command, configures the auxiliary ports properties including: Baud rate, handshaking, enable/disabled port, and echo. See the CC command in the Command Reference for a full description and command syntax.

If the CC command is configured for hardware handshaking it is required to use the RTS and CTS lines. The RTS line will go high whenever the DMC-41x3 is not ready to receive additional characters. The CTS line will inhibit the DMC-41x3 from sending additional characters. Note, the CTS line goes high for inhibit.

RS-422 Configuration

The DMC-41x3 can be ordered with the auxiliary port configured for RS-422 communication. RS-422 communication is a differentially driven serial communication protocol that should be used when long distance serial communication is required in an application.
For more information see RS-422 – Auxiliary Serial Port Serial Communication in the in Appendix.

Ethernet Configuration

Communication Protocols

The Ethernet is a local area network through which information is transferred in units known as packets. Communication protocols are necessary to dictate how these packets are sent and received. The DMC-41x3 supports two industry standard protocols, TCP/IP and UDP/IP. The controller will automatically respond in the format in which it is contacted.

TCP/IP is a "connection" protocol. The Client connects to the Server through a series of packet handshakes in order to begin communicating. Each packet sent is acknowledged when received. If no acknowledgment is received, the information is assumed lost and is resent.

Unlike TCP/IP, UDP/IP does not require a "connection". If information is lost, the controller does not return a colon or question mark. Because UDP does not provide for lost information, the sender must re-send the packet.

It is recommended that the motion control network containing the controller and any other related devices be placed on a “closed” network. If this recommendation is followed, UDP/IP communication to the controller may be utilized instead of a TCP connection. With UDP there is less overhead, resulting in higher throughput. Also, there is no need to reconnect to the controller with a UDP connection. Because handshaking is built into the Galil communication protocol through the use of colon or question mark responses to commands sent to the controller, the TCP handshaking is not required.

Packets must be limited to 512 data bytes (including UDP/TCP IP Header) or less. Larger packets could cause the controller to lose communication.

Note: In order not to lose information in transit, the user must wait for the controller's response before sending the next packet.

Addressing

There are three levels of addresses that define Ethernet devices. The first is the MAC or hardware address. This is a unique and permanent 6 byte number. No other device will have the same MAC address. The DMC-41x3 MAC address is set by the factory and the last two bytes of the address are the serial number of the board. To find the Ethernet MAC address for a DMC-41x3 unit, use the TH command. A sample is shown here with a unit that has a serial number of 3:

Sample MAC Ethernet Address: 00-50-4C-38-00-03

The second level of addressing is the IP address. This is a 32-bit (or 4 byte) number that usually looks like this: 192.168.15.1. The IP address is constrained by each local network and must be assigned locally. Assigning an IP address to the DMC-41x3 controller can be done in a number of ways.

The first method for setting the IP address is using a DHCP server. The DH command controls whether the DMC-41x3 controller will get an IP address from the DHCP server. If the unit is set to DH**1** (default) and there is a DHCP server on the network, the controller will be dynamically assigned an IP address from the server. Setting the board to DH**0** will prevent the controller from being assigned an IP address from the server.

The second method to assign an IP address is to use the BOOT-P utility via the Ethernet connection. The BOOT-P functionality is only enabled when DH is set to 0. Either a BOOT-P server on the internal network or the Galil software may be used. When opening the Galil Software, it will respond with a list of all DMC-41x3’s and other controllers on the network that do not currently have IP addresses. The user must select the board and the software will assign the specified IP address to it. This address will be burned into the controller (BN) internally to save the IP address to the non-volatile memory.

NOTE: if multiple boards are on the network – use the serial numbers to differentiate them.

CAUTION	Be sure that there is only one BOOT-P or DHCP server running. If your network has DHCP or BOOT-P running, it may automatically assign an IP address to the DMC-41x3 controller upon linking it to the network. In order to ensure that the IP address is correct, please contact your system administrator before connecting the I/O board to the Ethernet network.

The third method for setting an IP address is to send the IA command through the USB port. The IA command is only valid if DH**0** is set. The IP address may be entered as a 4 byte number delimited by commas (industry standard uses periods) or a signed 32 bit number (e.g. IA **124,51,29,31** or IA **2083724575**). Type in BN to save the IP address to the DMC-41x3 non-volatile memory.
Note: Galil strongly recommends that the IP address selected is not one that can be accessed across the Gateway. The Gateway is an application that controls communication between an internal network and the outside world.
The third level of Ethernet addressing is the UDP or TCP port number. The Galil board does not require a specific port number. The port number is established by the Client each time it connects to the DMC-41x3 board. Typical port numbers for applications are:

Port 23: Telnet
Port 502: Modbus

Communicating with Multiple Devices

The DMC-41x3 is capable of supporting multiple Clients and Servers. The Clients may be multiple PC's that send commands to the controller. The Servers are typically peripheral I/O devices that receive commands from the controller.

An Ethernet handle is a communication resource within a device. The DMC-41x3 can have a maximum of 8 Ethernet handles open at any time. When using TCP/IP, each Client or Server uses an individual Ethernet handle. In UDP/IP, one handle may be used for all the Clients, but each Server uses one. If all 8 handles are in use and a 9th Client tries to connect, it will be sent a "reset packet" that generates the appropriate error in its windows application.
When the Galil controller acts as the Client, the IH command is used to assign handles and connect to its Servers. The IP address may be entered as a 4 byte number separated with commas (industry standard uses periods) or as a signed 32 bit number.
Which devices receive what information from the controller depends on a number of things. If a device queries the controller, it will receive the response unless it explicitly tells the controller to send it to another device. If the command that generates a response is part of a downloaded program, the response will route to whichever port is specified as the default (unless explicitly told to go to another port with the CF command). To designate a specific destination for the information, add {Ex} to the end of the command. (Ex. MG{EC}"Hello" will send the message "Hello" to handle #3.)

Multicasting

A multicast may only be used in UDP/IP and is similar to a broadcast (where everyone on the network gets the information) but specific to a group. In other words, all devices within a specified group will receive the information that is sent in a multicast. There can be many multicast groups on a network and are differentiated by their multicast IP address. To communicate with all the devices in a specific multicast group, the information can be sent to the multicast IP address rather than to each individual device IP address. All Galil controllers belong to a default multicast address of 239.255.19.56. The controller's multicast IP address can be changed by using the IA command.

Using Third Party Software

Galil supports DHCP, ARP, BOOT-P, and Ping which are utilities for establishing Ethernet connections. DHCP is a protocol used by networked devices (clients) to obtain the parameters necessary for operation in an Internet Protocol network. ARP is an application that determines the Ethernet (hardware) address of a device at a specific IP address. BOOT-P is an application that determines which devices on the network do not have an IP address and assigns the IP address you have chosen to it. Ping is used to check the communication between the device at a specific IP address and the host computer.
The DMC-41x3 can communicate with a host computer through any application that can send TCP/IP or UDP/IP packets. A good example of this is Telnet, a utility that comes with most Windows systems.

Modbus

An additional protocol layer is available for speaking to I/O devices. Modbus TCP is an application-layer protocol that sends and receives information in Modbus frames encapsulated in TCP/IP packets. In this protocol, each Server has a 1 byte Server address. The DMC-41x3 can use a specific Server address or default to the handle number. The port number for Modbus is 502. The DMC-41x3 can fill the role of either a Modbus Client or Modbus Server.
The Modbus protocol has a set of commands called function codes. The DMC-41x3 supports the 10 major function codes as a Modbus Client:

Function Code	Definition
01	Read Coil Status (Read Bits)
02	Read Input Status (Read Bits)
03	Read Holding Registers (Read Words)
04	Read Input Registers (Read Words)
05	Force Single Coil (Write One Bit)
06	Preset Single Register (Write One Word)
07	Read Exception Status (Read Error Code)
15	Force Multiple Coils (Write Multiple Bits)
16	Preset Multiple Registers (Write Words)
17	Report Server ID

The DMC-41x3 supports 2 major function codes as a Modbus Server:

Function Code	Definition
03	Read Holding Registers (Read Words)
16	Preset Multiple Registers (Write Words)

When the DMC-41x3 is setup as a Client, the DMC-41x3 can read from and write to the memory space of a Server by using the MB command. When the DMC-41x3 is setup as a Server, the Client can read from or write to 1000 elements of the array space of the DMC-41x3. Each element is accessible as a 16-bit unsigned integer (Modbus register 1xxx) or as a 32-bit floating point number (Modbus registers 2xxx). The ability to write to the Server array table is enabled by setting the ME command (ME is not needed to read array data). See the DMC-41x3 Command Reference for further details.
The DMC-41x3 provides three levels of Modbus communication. The first level allows the user to create a raw packet and receive raw data. It uses the MBm command with a function code of –1. The format of the command is

```
MBm=-1,len,array[]
```

where len is the number of bytes

array[] is the array with the data

The second level incorporates the Modbus structure. This is necessary for sending configuration and special commands to an I/O device. The formats vary depending on the function code that is called. For more information refer to the Command Reference.

The third level of Modbus communication uses standard Galil commands. Once the Server has been configured, the commands that may be used are @IN[], @AN[], SB, CB, OB, and AO. For example, AO 2020,8.2 would tell I/O register 2020 to output 8.2 volts.
If a specific Server address is not necessary, the I/O number to be used can be calculated with the following:

I/O Number = (HandleNum*1000) + ((Module-1)*4) + (BitNum-1)

Where HandleNum is the handle number from 1 (A) to 8 (H). Module is the position of the module in the rack from 1 to 16. BitNum is the I/O point in the module from 1 to 4.

Modbus Examples

Example #1

DMC-4143 connected as a Modbus Client to a RIO-47120 via Modbus. The DMC-4143 will set or clear all 16 of the RIO's digital outputs.

1. Begin by opening a connection to the RIO which in our example has IP address 192.168.1.120

   ```
   IHB=192,168,1,120<502>2      (Issued to DMC-4143)
   ```

2. Dimension an array to store the commanded values. Set array element 0 equal to 170 and array element 1 equal to 85. (array element 1 configures digital outputs 15-8 and array element 0 configures digital outputs 7-0)

   ```
   DM myarray[2]
   myarray[0]=170              (which is 10101010 in binary)
   myarray[1]=85                    (which is 01010101in binary)
   ```

3. Send the appropriate MB command. Use function code 15. Start at output 0 and set/clear all 16 outputs based on the data in myarray[]

   ```
   MBB=,15,0,16,myarray[]
   ```

   Set the outputs using the SB command.

   ```
   SB2001;SB2003;SB2005;SB2007;SB2008;SB2010;SB2012;SB2014;
   ```

Results:

Both steps 3a and 3b will result in outputs being activated as below. The only difference being that step 3a will set and clear all 16 bits where as step 3b will only set the specified bits and will have no affect on the others.

Bit Number	Status
0	0
1	1
2	0
3	1
4	0
5	1
6	0
7	1

Bit Number	Status
8	1
9	0
10	1
11	0
12	1
13	0
14	1
15	0

Example #2

DMC-4143 connected as a Modbus Client to a 3rd party PLC. The DMC-4143 will read the value of analog inputs 3 and 4 on the PLC located at addresses 40006 and 40008 respectively. The PLC stores values as 32-bit floating point numbers which is common.

1. Begin by opening a connection to the PLC which has an IP address of 192.168.1.10 in our example

```
IHB=192,168,1,120<502>2      (Issued to DMC-4143)
```

2. Dimension an array to store the results

```
DM myanalog[4]
```

3. Send the appropriate MB command. Use function code 4 (as specified per the PLC). Start at address 40006. Retrieve 4 modbus registers (2 modbus registers per 1 analog input, as specified by the PLC)

```
MBB=,4,40006,4,myanalog[]
```

Results:

Array elements 0 and 1 will make up the 32 bit floating point value for analog input 3 on the PLC and array elements 2 and 3 will combine for the value of analog input 4.

```
myarray[0]=16412=0x401C
myarray[1]=52429=0xCCCD
myarray[2]=49347=0xC0C3
myarray[3]=13107=0x3333
```

Analog input 3 = 0x401CCCCD = 2.45V

Analog input 4 = 0xC0C33333 = -6.1V

Example #3

DMC-4143 connected as a Modbus Client to a hydraulic pump. The DMC-4143 will set the pump pressure by writing to an analog output on the pump located at Modbus address 30000 and consisting of 2 Modbus registers forming a 32 bit floating point value.

1. Begin by opening a connection to the pump which has an IP address of 192.168.1.100 in our example

```
IHB=192,168,1,100<502>2      (Issued to DMC-4143)
```

2. Dimension and fill an array with values that will be written to the PLC

```
DM pump[2]
pump[0]=16531=0x4093
pump[1]=13107=0x3333
```

3. Send the appropriate MB command. Use function code 16. Start at address 30000 and write to 2 registers using the data in the array pump[]

```
MBB=,16,30000,2,pump[]
```

Results:

Analog output will be set to 0x40933333 which is 4.6V

Data Record

The DMC-41x3 can provide a binary block of status information with the use of the QR and DR commands. These commands, along with the QZ command can be very useful for accessing complete controller status. The QR command will return 4 bytes of header information and specific blocks of information as specified by the command arguments:

```
QR ABCDEFGHST
```

Each argument corresponds to a block of information according to the Data Record Map below. If no argument is given, the entire data record map will be returned. Note that the data record size will depend on the number of axes.

The following is the byte map for the binary data.

Data Record Map Key	
Acronym	**Meaning**
UB	Unsigned byte
UW	Unsigned word
SW	Signed word
SL	Signed long
UL	Unsigned long

General Controller Information and Status					
ADDR	**TYPE**	**ITEM**	**ADDR**	**TYPE**	**ITEM**
00	UB	1st Byte of Header	30-31	SW	Reserved
01	UB	2nd Byte of Header	32-33	SW	Reserved
02	UB	3rd Byte of Header	34-35	SW	Reserved
03	UB	4th Byte of Header	36-37	SW	Reserved
04-05	UW	sample number	38-39	SW	Reserved
06	UB	general input block 0 (inputs 1-8)	40-41	SW	Reserved
07	UB	general input block 1 (inputs 9-16)	42	UB	Ethernet Handle A Status
08	UB	general input block 2 (inputs 17-24)	43	UB	Ethernet Handle B Status
09	UB	general input block 3 (inputs 25-32)	44	UB	Ethernet Handle C Status
10	UB	general input block 4 (inputs 33-40)	45	UB	Ethernet Handle D Status
11	UB	general input block 5 (inputs 41-48)	46	UB	Ethernet Handle E Status
12	UB	general input block 6 (inputs 49-56)	47	UB	Ethernet Handle F Status
13	UB	general input block 7 (inputs 57-64)	48	UB	Ethernet Handle G Status
14	UB	general input block 8 (inputs 65-72)	49	UB	Ethernet Handle H Status
15	UB	general input block 9 (inputs 73-80)	50	UB	error code
16	UB	general output block 0 (outputs 1-8)	51	UB	thread status – see bit field map below
17	UB	general output block 1 (outputs 9-16)	52-55	UL	Amplifier Status
18	UB	general output block 2 (outputs 17-24)	56-59	UL	Segment Count for Contour Mode
19	UB	general output block 3 (outputs 25-32)	60-61	UW	Buffer space remaining – Contour Mode
20	UB	general output block 4 (outputs 33-40)	62-63	UW	segment count of coordinated move for S plane
21	UB	general output block 5 (outputs 41-48)	64-65	UW	coordinated move status for S plane – see bit field map below
22	UB	general output block 6 (outputs 49-56)	66-69	SL	distance traveled in coordinated move for S plane
23	UB	general output block 7 (outputs 57-64)	70-71	UW	Buffer space remaining – S Plane
24	UB	general output block 8 (outputs 65-72)	72-73	UW	segment count of coordinated move for T plane
25	UB	general output block 9 (outputs 73-80)	74-75	UW	Coordinated move status for T plane – see bit field map below
26-27	SW	Reserved	76-79	SL	distance traveled in coordinated move for T plane
28-29	SW	Reserved	80-81	UW	Buffer space remaining – T Plane

Axis Information					
ADDR	**TYPE**	**ITEM**	**ADDR**	**TYPE**	**ITEM**
82-83	UW	A axis status – see bit field map below	226-227	UW	E axis status – see bit field map below
84	UB	A axis switches – see bit field map below	228	UB	E axis switches – see bit field map below
85	UB	A axis stop code	229	UB	E axis stop code
86-89	SL	A axis reference position	230-233	SL	E axis reference position
90-93	SL	A axis motor position	234-237	SL	E axis motor position
94-97	SL	A axis position error	238-241	SL	E axis position error
98-101	SL	A axis auxiliary position	242-245	SL	E axis auxiliary position
102-105	SL	A axis velocity	246-249	SL	E axis velocity
106-109	SL	A axis torque	250-253	SL	E axis torque
110-111	SW or UW [1]	A axis analog input	254-255	SW or UW [1]	E axis analog input
112	UB	A Hall Input Status	256	UB	E Hall Input Status
113	UB	Reserved	257	UB	Reserved
114-117	SL	A User defined variable (ZAA)	258-261	SL	E User defined variable (ZAE)
118-119	UW	B axis status – see bit field map below	262-263	UW	F axis status – see bit field map below
120	UB	B axis switches – see bit field map below	264	UB	F axis switches – see bit field map below
121	UB	B axis stop code	265	UB	F axis stop code
122-125	SL	B axis reference position	266-269	SL	F axis reference position
126-129	SL	B axis motor position	270-273	SL	F axis motor position
130-133	SL	B axis position error	274-277	SL	F axis position error
134-137	SL	B axis auxiliary position	278-281	SL	F axis auxiliary position
138-141	SL	B axis velocity	282-285	SL	F axis velocity
142-145	SL	B axis torque	286-289	SL	F axis torque
146-147	SW or UW [1]	B axis analog input	290-291	SW or UW [1]	F axis analog input
148	UB	B Hall Input Status	292	UB	F Hall Input Status
149	UB	Reserved	293	UB	Reserved
150-153	SL	B User defined variable (ZAB)	294-297	SL	F User defined variable (ZAF)
154-155	UW	C axis status – see bit field map below	298-299	UW	G axis status – see bit field map below
156	UB	C axis switches – see bit field map below	300	UB	G axis switches – see bit field map below
157	UB	C axis stop code	301	UB	G axis stop code
158-161	SL	C axis reference position	302-305	SL	G axis reference position
162-165	SL	C axis motor position	306-309	SL	G axis motor position
166-169	SL	C axis position error	310-313	SL	G axis position error
170-173	SL	C axis auxiliary position	314-317	SL	G axis auxiliary position
174-177	SL	C axis velocity	318-321	SL	G axis velocity
178-181	SL	C axis torque	322-325	SL	G axis torque
182-183	SW or UW [1]	C axis analog input	326-327	SW or UW [1]	G axis analog input
184	UB	C Hall Input Status	328	UB	G Hall Input Status
185	UB	Reserved	329	UB	Reserved
186-189	SL	C User defined variable (ZAC)	330-333	SL	G User defined variable (ZAG)
190-191	UW	D axis status – see bit field map below	334-335	UW	H axis status – see bit field map below
192	UB	D axis switches – see bit field map below	336	UB	H axis switches – see bit field map below
193	UB	D axis stop code	337	UB	H axis stop code
194-197	SL	D axis reference position	338-341	SL	H axis reference position
198-201	SL	D axis motor position	342-345	SL	H axis motor position
202-205	SL	D axis position error	346-349	SL	H axis position error
206-209	SL	D axis auxiliary position	350-353	SL	H axis auxiliary position
210-213	SL	D axis velocity	354-357	SL	H axis velocity
214-217	SL	D axis torque	358-361	SL	H axis torque
218-219	SW or UW [1]	D axis analog input	362-363	SW or UW [1]	H axis analog input
220	UB	D Hall Input Status	364	UB	H Hall Input Status
221	UB	Reserved	365	UB	Reserved
222-225	SL	D User defined variable (ZAD)	366-369	SL	H User defined variable (ZAH)

[1] Will be either a Signed Word or Unsigned Word depending upon AQ setting. See AQ in the Command Reference for more information.

Data Record Bit Field Maps

Header Information - Byte 0, 1 of Header:

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
1	N/A	N/A	N/A	N/A	I Block Present in Data Record	T Block Present in Data Record	S Block Present in Data Record
BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
H Block Present in Data Record	G Block Present in Data Record	F Block Present in Data Record	E Block Present in Data Record	D Block Present in Data Record	C Block Present in Data Record	B Block Present in Data Record	A Block Present in Data Record

Bytes 2, 3 of Header:

Bytes 2 and 3 make a word which represents the Number of bytes in the data record, including the header. Byte 2 is the low byte and byte 3 is the high byte

Note: The header information of the data records is formatted in little endian (reversed network byte order).

Thread Status (1 Byte)

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
Thread 7 Running	Thread 6 Running	Thread 5 Running	Thread 4 Running	Thread 3 Running	Thread 2 Running	Thread 1 Running	Thread 0 Running

Coordinated Motion Status for S or T Plane (2 Byte)

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
Move in Progress	N/A	N/A	N/A	N/A	N/A	N/A	N/A

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
N/A	N/A	Motion is slewing	Motion is stopping due to ST or Limit Switch	Motion is making final decel.	N/A	N/A	N/A

Axis Status (1 Word)

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
Move in Progress	Mode of Motion PA or PR	Mode of Motion PA only	(FE) Find Edge in Progress	Home (HM) in Progress	1st Phase of HM complete	2nd Phase of HM complete or FI command issued	Mode of Motion Coord. Motion

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
Negative Direction Move	Mode of Motion Contour	Motion is slewing	Motion is stopping due to ST of Limit Switch	Motion is making final deceleration	Latch is armed	3rd Phase of HM in Progress	Motor Off

Axis Switches (1 Byte)

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
Latch Occurred	State of Latch Input	N/A	N/A	State of Forward Limit	State of Reverse Limit	State of Home Input	Stepper Mode

Amplifier Status (4 Bytes)

BIT 31	BIT 30	BIT 29	BIT 28	BIT 27	BIT 26	BIT 25	BIT 24
N/A	N/A	N/A	N/A	N/A	N/A	ELO Active (Axis E-H)	ELO Active (Axis A-D)

BIT 23	BIT 22	BIT 21	BIT 20	BIT 19	BIT 18	BIT 17	BIT 16
Peak Current H-axis	Peak Current G-axis	Peak Current F-axis	Peak Current E-axis	Peak Current D-axis	Peak Current C-axis	Peak Current B-axis	Peak current A-axis

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
Hall Error H-axis	Hall Error G-axis	Hall Error F-axis	Hall Error E-axis	Hall Error D-axis	Hall Error C-axis	Hall Error B-axis	Hall Error A-axis

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
Under Voltage Axis (E-H)	Over Temp. Axis (E-H)	Over Voltage Axis (E-H)	Over Current Axis (E-H)	Under Voltage Axis (A-D)	Over Temp. Axis (A-D)	Over Voltage Axis (A-D)	Over Current Axis (A-D)

Notes Regarding Velocity and Torque Information

The velocity information that is returned in the data record is 64 times larger than the value returned when using the command TV (Tell Velocity). See command reference for more information about TV.

The Torque information is represented as a number in the range of ±32767. Maximum negative torque is -32767. Maximum positive torque is 32767. Zero torque is 0.

QZ Command

The QZ command can be very useful when using the QR command, since it provides information about the controller and the data record. The QZ command returns the following 4 bytes of information.

BYTE #	INFORMATION
0	Number of axes present
1	number of bytes in general block of data record
2	number of bytes in coordinate plane block of data record
3	Number of Bytes in each axis block of data record

Galil Software

Galil provides a variety of software tools available to make communication and configuration easier for the user. Galil's latest generation software is available on the Galil website at:
http://galil.com/downloads/software

Creating Custom Software Interfaces

Galil provides programming tools so that users can develop their own custom software interfaces to a Galil controller. For new applications, Galil recommends the current generation communication libraries located on the Galil Website:
http://galil.com/downloads/api

Please visit the API examples page under the Learn section for details on getting started developing custom software interfaces for Galil controllers:

http://galil.com/learn/api-examples

Chapter 5 Command Basics

Introduction

The DMC-41x3 provides over 100 commands for specifying motion and machine parameters. Commands are included to initiate action, interrogate status and configure the digital filter. These commands are sent in ASCII.

The DMC-41x3 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 communications port for immediate execution by the DMC-41x3, or an entire group of commands can be downloaded into the DMC-41x3 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-41x3 instruction set and syntax. A summary of commands as well as a complete listing of all DMC-41x3 instructions is included in the *Command Reference*.

http://galil.com/downloads/manuals-and-data-sheets

Command Syntax - ASCII

DMC-41x3 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 <return> is used to terminate the instruction for processing by the DMC-41x3 command interpreter.

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

For example, the commands:

```
PR 4000; BG A <return>
```

will both be executed in order when the <return> is sent. PR is the two character instruction for position relative. **4000** is the argument which represents the required position value in counts. BG A begins motion. The <return> terminates the instructions. The space between PR and **4000** is optional.

Implicit Notation

For controllers with 5 or more axes, the axes are referred to as A, B, C, D, E, F, G, H. The specifiers X, Y, Z, and W may be used interchangeably with A, B, C, and D to maintain compatibility with legacy DMC code. The A, B, C, and D notation should be used for new applications.

When specifying values for commands that apply to multiple axes, commas are used to separate the values for each axis. If no data is specified for an axis, a comma is still needed as shown in the examples below. If no data is specified for an axis, the previous value is maintained.

To view the current values for each command, type the command followed by a ? for each axis requested.

PR 1000	Specify A only as 1000
PR ,2000	Specify B only as 2000
PR ,,3000	Specify C only as 3000
PR ,,,4000	Specify D only as 4000
PR 2000,4000,6000,8000	Specify A,B,C and D
PR ,8000,,9000	specify B and D only
PR ?,?,?,?	Request A,B,C,D values
PR ,?	Request B value only

Explicit Notation

The DMC-41x3 provides an alternative method for specifying data. Here data is specified individually using a single axis specifier such as A, B, C and D. An equals sign is used to assign data to that axis. There should be no space on either side of the equals sign. For example:

PRA=1000	Specify a position relative movement for the A axis of 1000
ACB=200000	Specify acceleration for the B axis as 200000

To issue a command for all axes simultaneously, the * character can be used in place of the axis name. For example, to set a position error limit of 1,500 counts for all the axes on the controller, issue

OE*=1500	Set error limit of 1500 counts for all axes

Some commands expect an “axis mask” as an argument. An axis mask is a list of axes to which the command will be applied, with no separating commas. For example, ST AB stops motion on both the A and B axes. Note that there is no comma between A and B. If no parameters follow the command, action will take place on all axes. Here are some examples of axis mask syntax:

BG B	Begin B only
BG ABCD	Begin all axes A-D
BG BD	Begin B and D only
BG	Begin all axes

Coordinated Motion with more than 1 axis

When requesting action for coordinated motion, the letter S or T is used to specify the coordinated motion. This allows for coordinated motion to be setup for two separate coordinate systems. Refer to the CA command in the Command Reference for more information on specifying a coordinate system. For example:

BG S	Begin coordinated sequence, S
BG TD	Begin coordinated sequence, T, and D axis

Controller Response to DATA

The DMC-41x3 returns a **:** for valid commands and a **?** for invalid commands.

When using GDK, if the command BG is sent in lower case, the DMC-41x3 will return a **?**. To handle this event, GDK will automatically send the TC**1** command.

:bg	invalid command, lower case
1 Unrecognized command	DMC-41x3 returns the error code and response
?	DMC-41x3 also returns a ?

The user can also request the error code and response using the TC**1** command. The error code will specify the reason for the invalid command response.

:TC**1**	Request the error code
1 Unrecognized command	Returned response

There are many reasons for receiving an invalid command response. The most common reasons are: unrecognized command (such as typographical entry or lower case), command given at improper time (such as during motion), or a command out of range (such as exceeding maximum speed). A complete listing of all codes is listed in the TC command in the Command Reference section.

Interrogating the Controller

Interrogation Commands

The DMC-41x3 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 Application Programming and the Command Reference.

COMMAND	DESCRIPTION
RP	Report Command Position
RL	Report Latch
SC	Stop Code
TA	Tell Amplifier Error
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

Table 5.1: List of interrogation commands to request information from the controller.

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

TP A	Tell position A
0	Controllers Response
TP AB	Tell position A and B
0,0	Controllers Response

Interrogating Current Commanded Values

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

PR ?,?,?,?	Request A,B,C,D values
PR ,?	Request B value only

The controller can also be interrogated with operands.

Operands

Most DMC-41x3 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 A axis can be assigned to the variable 'v' with the command:

v=**_TP**A

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

Command Summary

For a complete command summary, see *Command Reference* manual.

http://galil.com/downloads/manuals-and-data-sheets

Chapter 6 Programming Motion

Overview

The DMC-41x3 provides several modes of motion, including independent positioning and jogging, coordinated motion, electronic cam motion, and electronic gearing. Each one of these modes is discussed in the following sections.

The DMC-4113 is a single axis controller and uses A-axis motion only. Likewise, the DMC-4123 uses A and B, the DMC-4133 uses A, B and C, and the DMC-4143 uses A, B, C and D. The DMC-4153 use A, B, C, D and E. The DMC-4163 use A, B, C, D, E and F. The DMC-4173 uses A, B, C, D, E, F and G. The DMC-4183 use the axes A, B, C, D, E, F, G and H.

NOTE: X, Y, Z and W may be interchanged with A, B, C and D.

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

EXAMPLE APPLICATION	MODE OF MOTION	COMMANDS
Absolute or relative positioning where each axis is independent and follows prescribed velocity profile.	Independent Axis Positioning	PA, PR, SP, AC, DC
Velocity control where no final endpoint is prescribed. Motion stops on Stop command.	Independent Jogging	JG, AC, DC, ST
Absolute positioning mode where absolute position targets may be sent to the controller while the axis is in motion.	Position Tracking	PA, AC, DC, SP, PT
Motion Path described as incremental position points versus time.	Contour Mode	CM, CD, DT
Motion Path described as incremental position, velocity and delta time	PVT Mode	PV, BT
2 to 8 axis coordinated motion where path is described by linear segments.	Linear Interpolation Mode	LM, LI, VS, VR, VA, VD, LE
2-D motion path consisting of arc segments and linear segments, such as engraving or quilting.	Vector Mode: Linear and Circular Interpolation Motion	VM, VP, CR, VS, VR, VA, VD, VE
Third axis must remain tangent to 2-D motion path, such as knife cutting.	Coordinating motion with Tangent Motion	VM, VP, CR, VS, VA, VD, TN, VE
Electronic gearing where Follower axes are scaled to Master axis which can move in both directions.	Electronic Gearing	GA, GD, _GP, GR, GM (for gantry)
Master/Follower where Follower axes must follow a Master such as conveyer speed.	Electronic Gearing with Ramped Gearing	GA, GD, _GP, GR
Moving along arbitrary profiles or mathematically prescribed profiles such as sine or cosine trajectories.	Contour Mode	CM, CD, DT

Teaching or Record and Play Back	Contour Mode with Teach (Record and Play-Back)	CM, CD, DT, RA, RD, RC
Backlash Correction	Dual Loop (Auxiliary Encoder)	DV
Following a trajectory based on a Master encoder position	Electronic Cam	EA, EM, EP, ET, EB, EG, EQ
Smooth motion while operating in independent axis positioning	Independent Motion Smoothing	IT
Smooth motion while operating in vector or linear interpolation positioning	Motion Smoothing	IT
Smooth motion while operating with stepper motors	Using the KS Command (Step Motor Smoothing)	KS
Gantry - two axes are coupled by gantry	Electronic Gearing - Example - Gantry Mode	GR, GM

Independent Axis Positioning

In this mode, motion between the specified axes is independent, and each axis follows its own profile. The user specifies the desired absolute position (PA) or relative position (PR), slew speed (SP), acceleration ramp (AC), and deceleration ramp (DC), for each axis. On begin (BG), the DMC-41x3 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-41x3 profiler. The actual motor motion may not be complete when the profile has been completed, however, the next motion command may be specified.

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

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

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

Command Summary - Independent Axis

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

Table 6.1: List of commands related to independent motion

Operand Summary - Independent Axis

OPERAND	DESCRIPTION
_ACm	Return acceleration rate for the axis specified by 'm'
_DCm	Return deceleration rate for the axis specified by 'm'
_SPm	Returns the speed for the axis specified by 'm'
_PAm	Returns the last commanded position for the 'm' axis
_PRm	Returns current incremental distance specified for the 'm' axis
_TVm	Returns the actual velocity of the axis specified by 'm'

Table 6.2: List of operands related to independent motion that contain specific information about an individual axis

Example - Absolute Position Movement

```
DP 0,0;                    'define initial position at zero for both axes
SH AB;                     'servo both axes
PA 10000,20000;            'specify absolute A,B position
AC 1000000,1000000;        'acceleration for A,B
DC 1000000,1000000;        'deceleration for A,B
SP 50000,30000;            'speeds for A,B
BG AB;                     'begin motion
AM AB;                     'after motion disable A and B axes
MO AB;                     'disable both axes
EN;                        'end program
```

Example - Multiple Move Sequence

This example will specify a relative position movement on A, B and C axes. The movement on each axis will be separated by 20 msec Figure 6.1 shows the velocity profiles for the A,B and C axis.

```
#a                         'begin program
PR 2000,500,100;           'specify relative position move of 2000, 500 and 100
                           counts for A,B and C axes
SP 20000,10000,5000;       'specify speed of 20000, 10000, and 5000 counts /
                           sec
AC 500000,500000,500000;   'specify acceleration of 500000 counts / sec2 for
                           all axes
DC 500000,500000,500000;   'specify deceleration of 500000 counts/sec2 for all
                           axes
SH ABC;                    'enable A, B, and C axes
BG A;                      'begin motion on the A axis
WT 20;                     'wait 20 msec
BG B;                      'begin motion on the B axis
WT 20;                     'wait 20 msec
BG C;                      'begin motion on C axis
AM ABC;                    'wait for motion to complete on all 3 axes
MO ABC;                    'disable A, B, and C axes
EN;                        'end Program
```

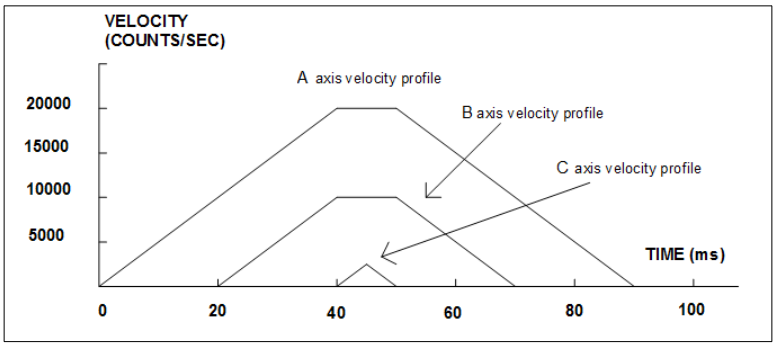


Figure 6.1: Velocity Profiles of ABC axes

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

Independent Jogging

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

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

Note that the controller operates as a closed-loop position controller while in the jog mode. The DMC-41x3 converts the velocity profile into a position trajectory and a new position target is generated every sample period. This method of control results in precise speed regulation with phase lock accuracy.

Command Summary - Jogging

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

Table 6.3: List of commands related to jogging mode of motion

Example - Jog in A only

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

```
#a                          'program label
SH AC;                      'servo A and C axes
AC 20000,,20000;            'specify A, C acceleration of 20000 counts/sec2
DC 20000,,20000;            'specify A, C deceleration of 20000 counts/sec2
JG 50000,,-25000;           'specify jog speed and direction for A and C axis
BG A;                       'begin A motion
AS A;                       'wait until A is at speed
BG C;                       'begin C motion
EN;                         'end program
```

Example - Joystick Jogging

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

```
#joy;                       'program label
SH A;                       'enable A axis
JG 0;                       'set in Jog Mode
BG A;                       'begin motion on A axis
#b;                         'label for loop
v1=@AN[1];                  'read analog input
vel=v1*(50000/10);          'compute speed
JG vel;                     'change JG speed
JP #b;                      'loop
EN;                         'end program
```

Position Tracking

The Galil controller may be placed in the position tracking mode to support changing the target of an absolute position move on the fly. New targets may be given in the same direction or the opposite direction of the current position target. The controller will then calculate a new trajectory based upon the new target and the acceleration, deceleration, and speed parameters that have been set. The motion profile in this mode is trapezoidal. There is not a set limit governing the rate at which the end point may be changed, however at the standard TM rate, the controller updates the position information at the rate of 1msec. The controller generates a profiled point every other sample, and linearly interpolates one sample between each profiled point. Some examples of applications that may use this mode are satellite tracking, missile tracking, random pattern polishing of mirrors or lenses, or any application that requires the ability to change the endpoint without completing the previous move.

The PA command is typically used to command an axis or multiple axes to a specific absolute position. For some applications such as tracking an object, the controller must proceed towards a target and have the ability to change the target during the move. In a tracking application, this could occur at any time during the move or at regularly scheduled intervals. For example if a robot was designed to follow a moving object at a specified distance and the path of the object wasn't known the robot would be required to constantly monitor the motion of the object that it was following. To remain within a specified distance it would also need to constantly update the position target it is moving towards. Galil motion controllers support this type of motion with the position tracking mode. This mode will allow scheduled or random updates to the current position target on the fly. Based on the new target the controller will either continue in the direction it is heading, change the direction it is moving, or decelerate to a stop.

The position tracking mode shouldn't be confused with the contour mode. The contour mode allows the user to generate custom profiles by updating the reference position at a specific time rate. In this mode, the position can be updated randomly or at a fixed time rate, but the velocity profile will always be trapezoidal with the parameters

specified by AC, DC, and SP. Updating the position target at a specific rate will not allow the user to create a custom profile.

The following example will demonstrate the possible different motions that may be commanded by the controller in the position tracking mode. In this example, there is a host program that will generate the absolute position targets. The absolute target is determined based on the current information the host program has gathered on the object that it is tracking. The position tracking mode does allow for all of the axes on the controller to be in this mode, but for the sake of discussion, it is assumed that the robot is tracking only in the X dimension.

The controller must be placed in the position tracking mode to allow on the fly absolute position changes. This is performed with the PT command. To place the X axis in this mode, the host would issue PT 1 to the controller if both X and Y axes were desired the command would be PT 1,1. The next step is to begin issuing PA command to the controller. The BG command isn't required in this mode, the AC, DC, and SP commands determine the shape of the trapezoidal velocity profile that the controller will use.

Example - Motion 1:

The host program determines that the first target for the controller to move to is located at 5000 encoder counts. The acceleration and deceleration should be set to 150,000 counts/sec^2 and the velocity is set to 50,000 counts/sec. The command sequence to perform this is listed below.

```
#ex1;                    'label
 DP 0;                   'define position as zero
SH A;                    'servo A axis
PT 1;                    'place the A axis in Position tracking mode
AC 150000;               'set the A axis acceleration to 150000 counts/sec2
DC 150000;               'set the A axis deceleration to 150000 counts/sec2
SP 50000;                'set the A axis speed to 50000 counts/sec
PA 5000;                 'command the A axis to absolute position 5000
                        encoder counts
AM A;                    'after motion
MO A;                    'disable A axis
EN;                      'end program
```

The output from this code can be seen in Figure 6.2, a screen capture from the GDK scope.

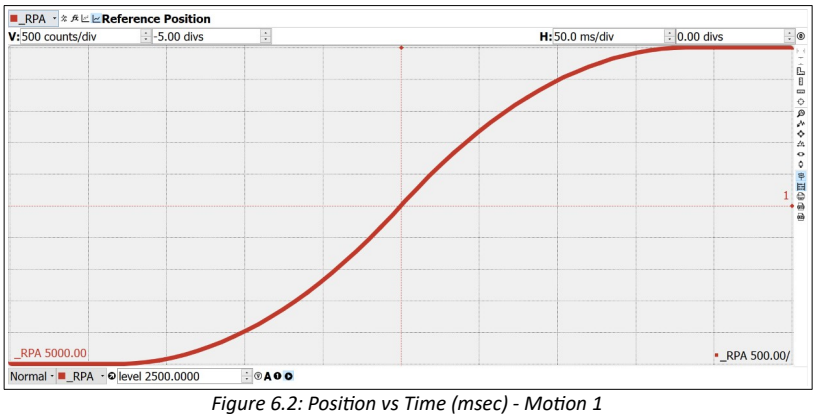


Figure 6.2: Position vs Time (msec) - Motion 1

Example - Motion 2:

The previous step showed the plot if the motion continued all the way to 5000, however partway through the motion, the object that was being tracked changed direction, so the host program determined that the actual target position should be 2000 counts at that time. Figure 6.2 shows what the position profile would look like if the move was allowed to complete to 5000 counts. The position was modified when the robot was at a position of 4200 counts (Figure 6.3). Note that the robot actually travels to a distance of almost 5000 counts before it turns around. This is a function of the deceleration rate set by the DC command. When a direction change is commanded, the controller decelerates at the rate specified by the DC command. The controller then ramps the velocity in up to the value set with SP in the opposite direction traveling to the new specified absolute position. Figure 6.3 the velocity profile is triangular because the controller doesn't have sufficient time to reach the set speed of 50000 counts/sec before it is commanded to change direction.

The below code is used to simulate this scenario:

```
#ex2;                    'label
 DP 0;                   'define position as zero
SH A;                    'servo A axis
PT 1;                    'place the A axis in Position tracking mode
AC 150000;               'set the A axis acceleration to 150000 counts/sec2
DC 150000;               'set the A axis deceleration to 150000 counts/sec2
SP 50000;                'set the A axis speed to 50000 counts/sec
PA 5000;                 'command the A axis to absolute position 5000
                        encoder counts
MF 4200;                 'move forward to absolute position 4200
PA 2000;                 'change end point position to position 2000
AM A;                    'after motion
MO A;                    'disable A axis
EN;                      'end program
```

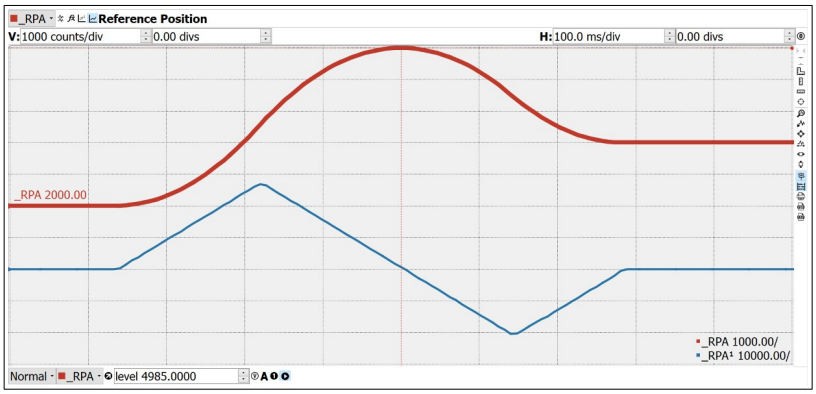


Figure 6.3: Position and Velocity vs Time(msec) for Motion 2

Example - Motion 3:

In this motion, the host program commands the controller to begin motion towards position 5000, changes the target to -2000, and then changes it again to 8000. Figure 6.4 shows the plot of position vs. time and velocity vs. time. Below is the code that is used to simulate this scenario:

```
#ex3;                    'label
 DP 0;                   'define position as zero
SH A;                    'servo A axis
PT 1;                    'place the A axis in Position tracking mode
AC 150000;               'set the A axis acceleration to 150000 counts/sec2
DC 150000;               'set the A axis deceleration to 150000 counts/sec2
SP 50000;                'set the A axis speed to 50000 counts/sec
PA 5000;                 'command the A axis to absolute position 5000
                        encoder counts
WT 300;                  'wait 300 ms
PA -2000;                'change end point position to -2000
WT 200;                  'wait 200 ms
PA 8000;                 'change end point position to 8000
AM A;                    'after motion
MO A;                    'disable A axis
EN;                      'end program
```

Figure 6.5 demonstrates the use of motion smoothing (IT) on the velocity profile in this mode. The jerk in the system is also affected by the values set for AC and DC.

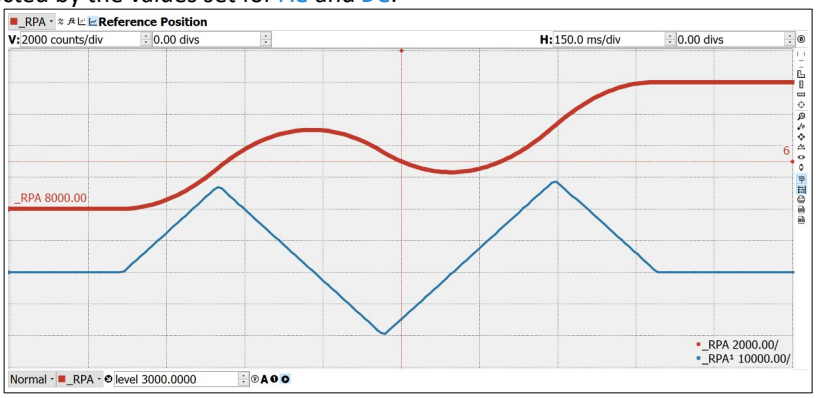


Figure 6.4: Position and Velocity vs Time (msec) for Motion 3

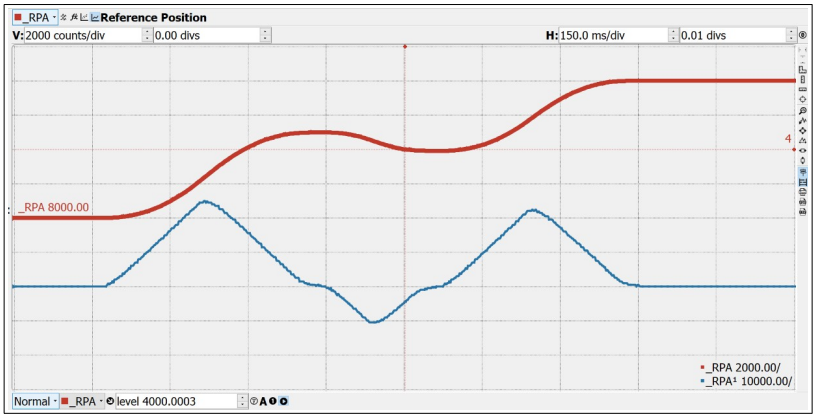


Figure 6.5: Position and Velocity vs Time (msec) for Motion 3 with IT 0.1

Note, the controller treats the point where the velocity passes through zero as the end of one move, and the beginning of another move. IT can be used to smooth the velocity profile, however the addition of IT will introduce some time delay.

Trippoints

Most trippoints are valid for use while in the position tracking mode. There are a few exceptions to this; the AM and MC commands may not be used while in this mode. It is recommended that AR, MF, MR, or AP be used, as they involve motion in a specified direction, or the passing of a specific absolute position.

Command Summary – Position Tracking Mode

COMMAND	DESCRIPTION
AC	Acceleration settings for the specified axes
AP	Trippoint that holds up program execution until an absolute position has been reached
DC	Deceleration settings for the specified axes
MF	Trippoint to hold up program execution until n number of counts have passed in the forward direction. Only one axis at a time may be specified.
MR	Trippoint to hold up program execution until n number of counts have passed in the reverse direction, only one axis at a time may be specified
PT	Command used to enter and exit Position Tracking Mode
PA	Command Used to specify the absolute position target
SP	Speed settings for the specified axes

Table 6.4: List of commands for Position Tracking Mode

Linear Interpolation Mode

The DMC-41x3 provides a linear interpolation mode for 2 or more axes. In linear interpolation mode, motion between the axes is coordinated to maintain the prescribed vector speed, acceleration, and deceleration along the specified path. The motion path is described in terms of incremental distances for each axis. An unlimited number of incremental segments may be given in a continuous move sequence, making the linear interpolation mode ideal for following a piece-wise linear path. There is no limit to the total move length.

The LM command selects the Linear Interpolation mode and axes for interpolation. For example, LM BC selects only the B and C axes for linear interpolation.

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

Specifying Linear Segments

The LI command specifies the incremental move distance for each specified axis. This means motion is prescribed with respect to the current axis position. Up to 511 incremental move segments may be given prior to the Begin Sequence (BGS) command. Once motion has begun, additional LI segments may be sent to the controller.

The Clear Sequence (CS) command can be used to remove LI segments stored in the buffer prior to the start of the motion. To stop the motion, use the instructions STS or AB. The command, ST, causes a decelerated stop. The Abort command, AB, causes an instantaneous stop and aborts the program, and the command AB**1** aborts the motion only.

The Linear End (LE) command must be used to specify the end of a linear move sequence. This command tells the controller to decelerate to a stop following the last LI command. If an LE command is not given, an Abort AB**1** must be used to abort the motion sequence.

It is the responsibility of the user to keep enough LI segments in the DMC-41x3 sequence buffer to ensure continuous motion. If the controller receives no additional LI segments and no LE command, the controller will stop motion instantly at the last vector. There will be no controlled deceleration. LM? or _**LM** returns the available

spaces for LI segments that can be sent to the buffer. 511 returned means the buffer is empty and 511 LI segments can be sent. A zero means the buffer is full and no additional segments can be sent. As long as the buffer is not full, additional LI segments can be sent at PC bus speeds.
The operand **_CS** returns the segment counter. As the segments are processed, _CS increases, starting at zero. This function allows the host computer to determine which segment is being processed.

Additional Commands

The commands VS, VA, and VD are used to specify the vector speed, acceleration and deceleration. The DMC-41x3 computes the vector speed based on the axes specified in the LM mode. For example, LM ABC designates linear interpolation for the A,B and C axes. The vector speed for this example would be computed using the equation: $VS^2=AS^2+BS^2+CS^2$, where AS, BS and CS are the speed of the A,B and C axes.
The controller always uses the axis specifications from LM, not LI, to compute the speed.
IT is used to set the S-curve smoothing constant for coordinated moves. The After Vector trippoint AV halts program execution until the specified vector distance has been reached.

An Example of Linear Interpolation Motion:

```
#lmove;                    'label
DP 0,0;                    'define position of A and B axes to be 0
SH AB;                     'servo A and B axes
LM AB;                     'define linear mode between A and B axes
LI 5000,0;                 'specify first linear segment
LI 0,5000;                 'specify second linear segment
LE;                        'end linear segments
VS 4000;                   'specify vector speed
BG S;                      'begin motion sequence
AV 4000;                   'set trippoint to wait until vector distance of 4000
                           is reached
VS 1000;                   'change vector speed
AV 5000;                   'set trippoint to wait until vector distance of 5000
                           is reached
VS 4000;                   'change vector speed
AM S;                      'after motion of S vector is completed
MO AB;                     'disable A and B axes
EN;                        'program end
```

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

Specifying Vector Speed for Each Segment

In some applications it is necessary to assign varying speeds to vector segments to produce optimal performance. This can be done by two functions: < n and > m

For example: LI **a,b,c,d<n>m**

The first speed related argument, <**n**, is equivalent to commanding a Vector Speed (VS **n**) at the start of the given segment and will cause an acceleration toward this newly commanded speed, subject to the other constraints.

The second argument, >**m**, requires the Vector Speed to reach the value **m** at the end of the segment. Note that the argument >**m** may start the deceleration within the given segment or during previous segments, as needed to meet the final speed requirement, under the given values of VA and VD.

Note, however, that the controller works with one > **m** argument at a time. As a consequence, one argument may be masked by another. For example, if the argument >100000 is followed by >5000, and the distance for deceleration is not sufficient, then the second condition will not be met. The controller will attempt to lower the speed to 5000, but will reach this speed in a later LI segment.

As an example, consider the following program.

```
#alt;                    'label for alternative program
DP 0,0;                  'define Position of A and Y axis to be 0
SH AB;                   'enable A and B axes
LM AB;                   'define linear mode between A and Y axes.
LI 4000,0<4000>1000;     'specify first linear segment with a vector speed of
                         4000 and end speed 1000
LI 1000,1000<4000>1000;  'specify second linear segment with a vector speed
                         of 4000 and end speed 1000
LI 0,5000<4000>1000;     'specify third linear segment with a vector speed of
                         4000 and end speed 1000
LE;                      'end linear segments
BG S;                    'begin motion sequence
AM S;                    'halt code until motion on the S plane is completed
MO AB;                   'disable A and B axes
EN;                      'program end
```

Changing Feed Rate:

The command VR allows the feed rate, VS, to be scaled between 0 and 10 with a resolution of .0001. This command takes effect immediately and causes VS to be scaled. VR also applies when the vector speed is specified with the '<' operator. This is a useful feature for feed rate override. VR does not ratio the accelerations. For example, VR **.5** results in the specification VS **2000** to be divided in half.

Command Summary - Linear Interpolation

COMMAND	DESCRIPTION
LM	Specify axes for linear interpolation
LI	Specify incremental distances relative to current position, and assign vector speed
VS	Specify vector speed
VA	Specify vector acceleration
VD	Specify vector deceleration
VR	Specify the vector speed ratio
BG	Begin Linear Sequence
CS	Clear sequence
LE	Linear End- Required at end of LI command sequence
AM	Trippoint for After Sequence complete
AV	Trippoint for After Relative Vector distance
IT	S curve smoothing constant for vector moves

Table 6.5: List of commands for Linear Interpolation Mode

Operand Summary - Linear Interpolation

OPERAND	DESCRIPTION
_AVm	Return distance traveled
_CSm	Returns number of the segment in the sequence, starting from 0
_LEm	Returns length of vector (resets after 2147483647)
_LMm	Returns number of available spaces for linear segments in the sequence buffer 0 means buffer full, 511 means buffer empty
_VPm	Return the absolute coordinate of the last data point along the trajectory

Table 6.6: List of operands related to Linear Interpolation Mode

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

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

Example - Linear Move

Make a coordinated linear move in the AB plane. Move to coordinates 5000,2000 counts at a vector speed of 100000 counts/sec and vector acceleration of 1000000 counts/sec^2.

```
#lmove;                 'label for program
SH AB;                  'enable A and B axes
LM AB;                  'specify axes for linear interpolation
LI 5000,2000;           'specify AB distances
LE;                     'specify end move
VS 100000;              'specify vector speed
VA 1000000;             'specify vector acceleration
VD 1000000;             'specify vector deceleration
BG S;                   'begin sequence
AM S;                   'wait for motion to finish
MO AB;                  'disable A and B axes
EN;                     'end program
```

Note that the above program specifies the vector speed, VS, and not the actual axis speeds VC and VD. The axis speeds are determined by the controller from:

$$VS = \sqrt{VC^2 + VD^2}$$

The result is shown in Figure 6.6: Linear Interpolation on the next page.

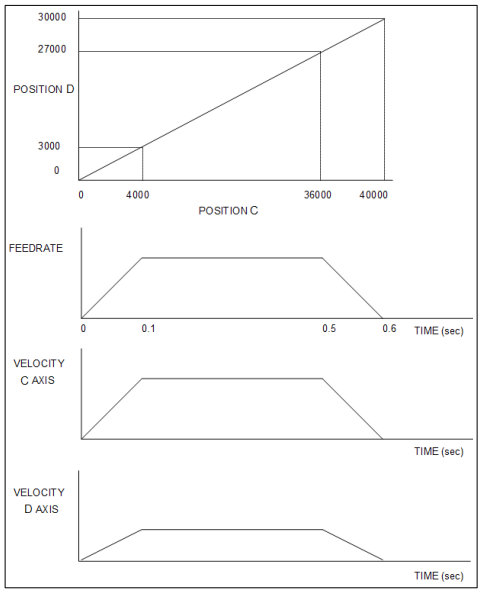


Figure 6.6: Linear Interpolation

Example - Multiple Moves

This example makes a coordinated linear move in the AB plane. The Arrays va and vb are used to store 750 incremental distances which are filled by the program #load.

```
#load;                    'label for load program
DM va [750],vb [750];     'define array
count=0;                  'initialize counter
n=0;                      'initialize position increment
#loop;                    'label for loop
va[count]=n;              'fill array element in va
vb[count]=n;              'fill array element in vb
n=n+10;                   'increment position
count=count+1;            'increment counter
JP#loop,(count<750);      'loop if array not full
#a;                       'label
SH AB;                    'enable A and B axes
LM AB;                    'specify linear mode for AB
count=0;                  'initialize array counter
#loop2;                   'if sequence buffer full, wait
WT 20
JP#loop2,(_LM=0);
JS#c,(count=500);         'begin motion on 500th segment
LI va[count],vb[count];   'specify linear segment
count=count+1;            'increment array counter
JP#loop2,(count<750);     'repeat until array done
LE;                       'end linear move
AM S;                     'after move sequence done
MG "DONE";                'send message
EN;                       'end program
#c;BG S;AM S;MO AB;EN;    'begin motion subroutine
```

Vector Mode: Linear and Circular Interpolation Motion

The DMC-41x3 allows a long 2-D path consisting of linear and arc segments to be prescribed. Motion along the path is continuous at the prescribed vector speed even at transitions between linear and circular segments. The DMC-41x3 performs all the complex computations of linear and circular interpolation, freeing the host PC from this time intensive task.

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

The VM command is used to specify the axes used in Vector Mode. For example, VM ACB selects the AC axes for coordinated motion and the B axis as the tangent.

Specifying the Coordinate Plane

The DMC-41x3 allows for 2 separate sets of coordinate axes for linear interpolation mode or vector mode. These two sets are identified by the letters S and T.

To specify vector commands the coordinate plane must first be identified. This is done by issuing the command CAS to identify the S plane or CAT to identify the T plane. All vector commands will be applied to the active coordinate system until changed with the CA command.

Specifying Vector Segments

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

The VP command specifies the coordinates of the end points of the vector movement with respect to the starting point. Non-sequential axis do not require comma delimitation. The CR command defines a circular arc with a radius, starting angle, and a traversed angle. The notation for a starting angle of zero corresponds to the positive horizontal direction, and a positive traverse angle corresponds to the counter-clockwise (CCW) direction.

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

The Clear Sequence (CS) command can be used to remove previous VP and CR commands which were stored in the buffer prior to the start of the motion. To stop the motion, use the instructions STS or AB**1**. ST stops motion at the specified deceleration. AB**1** aborts the motion instantaneously.

The Vector End (VE) command must be used to specify the end of the coordinated motion. This command requires the controller to decelerate to a stop following the last motion requirement. If a VE command is not given, an Abort (AB**1**) must be used to abort the coordinated motion sequence.

It is the responsibility of the user to keep enough motion segments in the DMC-41x3 sequence buffer to ensure continuous motion. If the controller receives no additional motion segments and no VE command, the controller will stop motion instantly at the last vector. There will be no controlled deceleration. **_LM** returns the available spaces for motion segments that can be sent to the buffer. **511** returned means the buffer is empty and 511 segments can be sent. Receiving a response of **0** implies the buffer is full and no additional segments can be sent. As long as the buffer is not full, additional segments can be sent at PC bus speeds.

The operand **_CS** can be used to determine the value of the segment counter.

Additional commands

The commands VS, VA, and VD are used for specifying the vector speed, acceleration, and deceleration.

IT is the s curve smoothing constant used with coordinated motion.

Specifying Vector Speed for Each Segment

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

VP n_1, n_2 <**o**>**p**

CR n_1, n_2, n_3<**o**>**p**

The first command, <**o**, is equivalent to commanding VS **o** at the start of the given segment and will cause an acceleration toward the new commanded speeds, subjects to the other constraints.

The second function, >p, requires the vector speed to reach the value at the end of the segment. Note that the function >p may start the deceleration within the given segment or during previous segments, as needed to meet the final speed requirement, under the given values of VA and VD.

Note, however, that the controller works with one >p command at a time. As a consequence, one function may be masked by another. For example, if the function >100000 is followed by >5000, and the distance for deceleration is not sufficient, the second condition will not be met. The controller will attempt to lower the speed to 5000, but will reach that at a different point.

Changing Feed Rate

The command VR allows the feed rate, VS, to be scaled between 0 and 10 with a resolution of .0001. This command takes effect immediately and causes VS scaled. VR also applies when the vector speed is specified with the < operator. This is a useful feature for feed rate override. VR does not ratio the accelerations. For example, VR 0.5 results in the specification VS 2000 to be divided by two.

Compensating for Differences in Encoder Resolution

By default, the DMC-41x3 uses a scale factor of 1:1 for the encoder resolution when used in vector mode. If this is not the case, the command, ES can be used to scale the encoder counts. The ES command accepts two arguments which represent the number of counts for the two encoders used for vector motion. The smaller ratio of the two numbers will be multiplied by the higher resolution encoder. For more information, see ES command in the Command Reference.

Trippoints

The AV command is the After Vector trippoint, which waits for the vector relative distance of n to occur before executing the next command in a program.

Tangent Motion

Several applications, such as cutting, require a third axis (i.e. a knife blade), to remain tangent to the coordinated motion path. To handle these applications, the DMC-41x3 allows one axis to be specified as the tangent axis. The VM command provides parameter specifications for describing the coordinated axes and the tangent axis.

Before the tangent mode can operate, it is necessary to assign an axis via the VM command and define its offset and scale factor via the TN command. The operand _TN can be used to return the initial position of the tangent axis.

Example:

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

```
#example;                'example program
SH ABC;                  'servo A, B, and C axes
VM ABC;                  'coordinate A and B axes, with C as tangent
TN 2000/360,-500;        '2000/360 counts/degree, position -500 is 0 degrees
                         in AB plane
CR 3000,0,180;           '3000 count radius, start at 0 and go to 180 CCW
VE;                      'end vector
CB 0;                    'disengage knife
```

```
PA 3000,0,_TN;          'move A and B to starting position, move C to initial
                        tangent position
BG ABC;                 'start the move to get into position
AM ABC;                 'when the move is complete
SB 0;                   'engage knife
WT 50;                  'wait 50 msec for the knife to engage
BG S;                   'do the circular cut
AM S;                   'after the coordinated move is complete
CB 0;                   'disengage knife
MO ABC;                 'disable A, B, C axes
MG "ALL DONE";          'send message
EN;                     'end program
```

Command Summary - Coordinated Motion Sequence

COMMAND	DESCRIPTION
VM	Specifies the axes for the planar motion
VP	Specifies target coordinates for a vector move
CR	Specifies arc segment, positive direction is CCW
VS	Specify vector speed or feed rate of sequence
VA	Specify vector acceleration along the sequence
VD	Specify vector deceleration along the sequence
VR	Specify vector speed ratio
BG	Begin motion sequence, S or T
CS	Clear sequence, S or T
AV	Trippoint for After Relative Vector distance
AM	Holds execution of next command until Motion Sequence is complete
TN	Tangent scale and offset
ES	Ellipse scale factor
IT	S curve smoothing constant for coordinated moves
CA	Specifies which coordinate system is to be active, S or T

Table 6.7: List of commands related to Vector Mode

Operand Summary - Coordinated Motion Sequence

OPERAND	DESCRIPTION
_VPm	The absolute coordinate of 'm' axis at the last intersection along the sequence
_AVm	Contains the vector distance from the start of the sequence for the 'm' coordinate system
_LMm	Number of available spaces for linear and circular segments in DMC-41x3 sequence buffer 0 means buffer is full, 511 means buffer is empty
_CSm	Number of the segment in the sequence for the 'm' coordinate system, starting at zero
_VEm	Vector length of coordinated move sequence for the 'm' coordinate system

Table 6.8: List of operands related to Vector Mode

Example:

Traverse the path shown in Figure 6.7. Feed rate is 20000 counts/sec. Plane of motion is AB

```
SH AB;                  'enable A and B axes
VM AB;                  'specify motion plane
VS 20000;               'specify vector speed
VA 1000000;             'specify vector acceleration
VD 1000000;             'specify vector deceleration
VP -4000,0;             'segment from point A to point B
```

```
CR 1500,270,-180;          'segment from point B to point C
VP 0,3000;                 'segment from point C to point D
CR 1500,90,-180;           'segment from point D to point A
VE;                        'end of sequence
BG S;                      'begin sequence
AM S;                      'wait until motion is completed
MO AB;                     'disable A and B axes
EN;                        'end program
```

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

The value of **_AVS** is **2000**.
The value of **_CSS** is **0**.
_VPA and **_VPB** contain the absolute coordinate of the point A.
Suppose that the interrogation is repeated at a point, halfway between the points C and D.
The value of **_AVS** is 4000+1500π+2000=**10712**.
The value of **_CSS** is **2**.
_VPA, **_VPB** contain the coordinates of the point C.

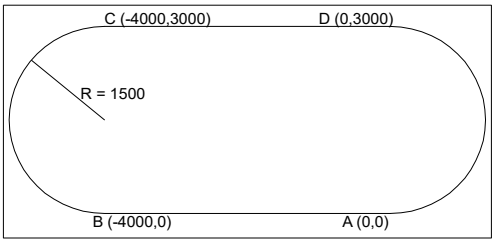


Figure 6.7: The Required Path

Vector Mode - Mathematical Analysis

The terms of coordinated motion are best explained in terms of the vector motion. The vector velocity, VS, which is also known as the feed rate, is the vector sum of the velocities along the A and B axes, VA and VB.

$$VS = \sqrt{VA^2 + VB^2}$$

The vector distance is the integral of VS, or the total distance traveled along the path. To illustrate this further, suppose that a string was placed along the path in the X-Y plane. The length of that string represents the distance traveled by the vector motion.

The vector velocity is specified independently of the path to allow continuous motion. The path is specified as a collection of segments. For the purpose of specifying the path, define a special X-Y coordinate system whose origin is the starting point of the sequence. Each linear segment is specified by the X-Y coordinate of the final point expressed in units of resolution, and each circular arc is defined by the arc radius, the starting angle, and the angular width of the arc. The zero angle corresponds to the positive direction of the X-axis and the CCW direction of rotation is positive. Angles are expressed in degrees, and the resolution is 1/256th of a degree. For example, the path shown in Figure 6.8 is specified by the instructions:

```
VP 0,10000
CR 10000, 180, -90
VP 20000, 20000
```

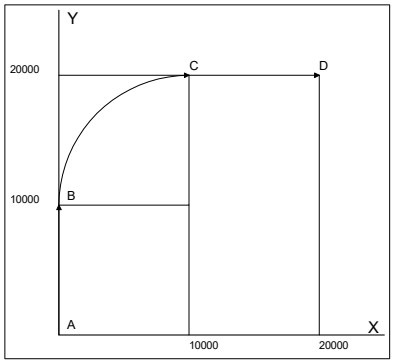


Figure 6.8: X-Y Motion Path

The first line describes the straight line vector segment between points A and B. The next segment is a circular arc, which starts at an angle of 180° and traverses -90°. Finally, the third line describes the linear segment between points C and D. Note that the total length of the motion consists of the segments:

A-B	Linear	10000 units
B-C	Circular	$\frac{R\|\Delta\theta 2\pi}{360}$ = 15708
C-D	Linear	10000
	Total	35708 counts

In general, the length of each linear segment is

$$L_k = \sqrt{Xk^2 + Yk^2}$$

Where Xk and Yk are the changes in X and Y positions along the linear segment. The length of the circular arc is

$$L_k = R_k|\Delta\Theta_k|2\pi/360$$

The total travel distance is given by

$$D = \sum_{k=1}^{n} L_k$$

The velocity profile may be specified independently in terms of the vector velocity and acceleration.

For example, the velocity profile corresponding to the path of Figure 6.8 may be specified in terms of the vector speed and acceleration.

```
VS 100000
VA 2000000
```

The resulting vector velocity is shown in Figure 6.9.

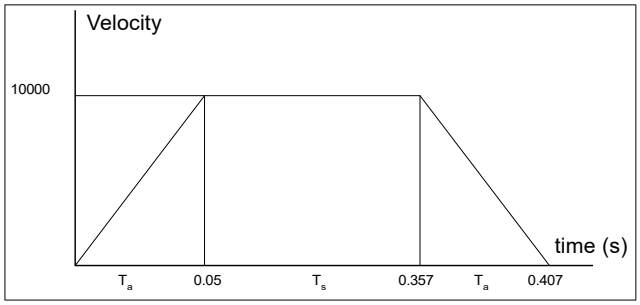


Figure 6.9: Vector Velocity Profile

The acceleration time, Ta, is given by

$$T_a = \frac{VS}{VA} = \frac{100000}{2000000} = 0.05s$$

The slew time, Ts, is given by

$$T_s = \frac{D}{VS} - T_a = \frac{35708}{100000} - 0.05 = 0.307s$$

The total motion time, Tt, is given by:

$$T_t = \frac{D}{VS} + T_a = 0.407s$$

The velocities along the X and Y axes are such that the direction of motion follows the specified path, yet the vector velocity fits the vector speed and acceleration requirements.

For example, the velocities along the X and Y axes for the path shown in Figure 6.8 are given in Figure 6.10.

Figure 6.10 on the next page shows the vector velocity. It also indicates the position point along the path starting at A and ending at D. Between the points A and B, the motion is along the Y axis. Therefore,

Vy = Vs

and

Vx = 0

Between the points B and C, the velocities vary gradually and finally, between the points C and D, the motion is in the X direction.

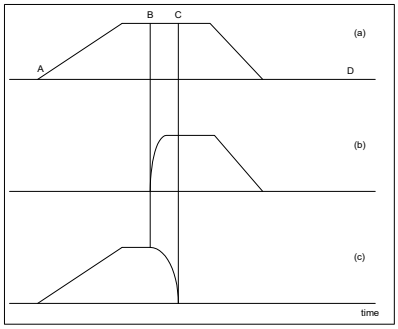


Figure 6.10: Vector and Axes Velocities

Electronic Gearing

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

The GA command specifies the master axes. GR specifies the gear ratios for the slaves where the ratio may be a number between ±127.9999 with a fractional resolution of .0001. There are two modes: standard gearing and gantry mode. The gantry mode (enabled with the command GM) allows the gearing to stay enabled even if a limit is hit or an ST command is issued on the slave axis. Under standard gearing mode, a limit switch condition or an ST command on the slave axis will end gearing mode. GR **0,0,0,0** turns off gearing in both modes.

The GM command select the axes to be controlled under the gantry mode. The parameter 1 enables gantry mode, and 0 disables it.

GR causes the specified axes to be geared to the actual position of the master. The master axis is commanded with motion commands such as PR, PA or JG.

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

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

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

Ramped Gearing

In some applications, especially when the master is traveling at high speeds, it is desirable to have the gear ratio ramp gradually to minimize large changes in velocity on the slave axis when the gearing is engaged. For example if the master axis is already traveling at 500,000 counts/sec and the slave will be geared at a ratio of 1:1 when the gearing is engaged, the slave will instantly develop following error, and command maximum current to the motor. This can be a large shock to the system. For many applications it is acceptable to slowly ramp the engagement of gearing over a greater time frame. Galil allows the user to specify an interval of the master axis over which the gearing will be engaged. For example, the same master A axis in this case travels at 500,000 counts/sec, and the gear ratio is 1:1, but the gearing is slowly engaged over 30,000 counts of the master axis, greatly diminishing the initial shock to the slave axis. Figure 6.11 below shows the velocity vs. time profile for instantaneous gearing. Figure 6.12 shows the velocity vs. time profile for the gradual gearing engagement.

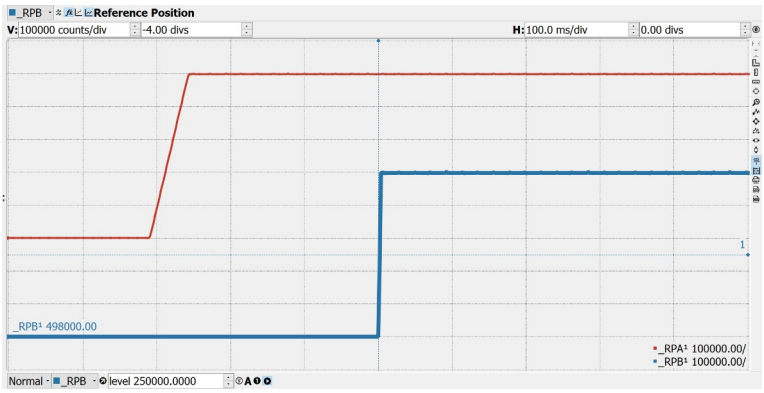


Figure 6.11: Velocity counts/sec vs. Time (msec) Instantaneous Gearing Engagement

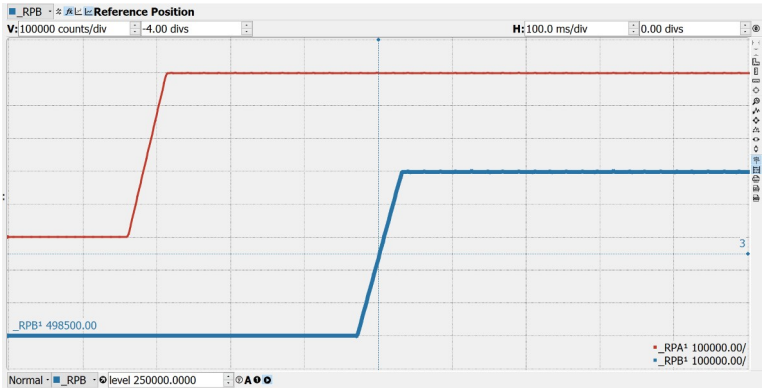


Figure 6.12: Velocity (counts/sec) vs. Time (msec) Ramped Gearing

The slave axis for each figure is shown on the bottom portion of the figure; the master axis is shown on the top portion. The shock to the slave axis will be significantly less in Figure 6.12 than in Figure 6.11. The ramped gearing does have one consequence. There isn't a true synchronization of the two axes, until the gearing ramp is complete. The slave will lag behind the true ratio during the ramp period. If exact position synchronization is required from the point gearing is initiated, then the position must be commanded in addition to the gearing. The controller keeps track of this position phase lag with the _GPm operand. The following example will demonstrate how the command is used.

Example – Electronic Gearing Over a Specified Interval

Objective Run two geared motors at speeds of 1.132 and -.045 times the speed of an external master. Because the master is traveling at high speeds, it is desirable for the speeds to change slowly.

The DMC-4133 where the C axis is the master and A and B are the geared axes can solve this problem. The gearing can be implemented to change over 6000 counts (3 revolutions) of the master axis.

```
MO C;                   'disable C axis for Master
GA C, C;                'specify C axis as the Master axis for both A and B
                        axes
GD 6000,6000;           'specify ramped gearing over 6000 counts of the
                        Master axis
GR 1.132,-.045;         'specify gear ratios
```

Using the ramped gearing, the slave axes will engage gearing gradually. Since the gearing is engaged over the interval of 6000 counts of the master, the slave axes will only travel ~3396 counts and ~135 counts respectively. The difference between these two values is stored in the _GPm operand. If exact position synchronization is required, the IP command is used to adjust for the difference.

Command Summary - Electronic Gearing

COMMAND	DESCRIPTION
GA	Specifies master axes for gearing CA, CB,CC,CD,CE,CF,CG,CH for commanded position DA, DB, DC, DD, DE, DF,DG,DH for auxiliary encoders S or T for gearing to coordinated motion
GD	Sets the distance the master will travel for the gearing change to take full effect
_GPm	Keeps track of the difference between the theoretical distance traveled if gearing changes took effect immediately, and the distance traveled since gearing changes take effect over a specified interval
GR	Sets gear ratio for slave axes, 0 disables electronic gearing for specified axis
GM	1 sets gantry mode, 0 disables gantry mode
MR	Trippoint for reverse motion past specified value
MF	Trippoint for forward motion past specified value

Table 6.9: List of commands for Electronic Gearing

Example - Simple Master / Slave

Master axis moves **10000** counts at slew speed of **100000** counts/sec. Axis B is defined as the master. The A, C, and D axes are geared to the master at ratios of **5**,**-.5** and **10** respectively.

```
#label;                 'label for program
GA B,,B,B;              'specify Master axes as B
GR 5,,-.5,10;           'set gear ratios
PR ,10000;              'specify B position
SP ,100000;             'specify B speed
SH ABCD;                'servo A, B, C, and D axes
BG B;                   'begin motion on B axis
AM B;                   'wait until motion is completed
MO ABCD;                'disable axes
EN;                     'end program
```

Example - Electronic Gearing

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

Solution: Use a DMC-4133 controller, where the C axis is the master and A and B are the geared axes.

```
MO C;                   'disable C axis
GA C,C;                 'specify C as the Master axis for both A and B
GR 1.132,-.045;         'specify gear ratios
```

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

```
GR 2;                   'specify gear ratio for A axis to be 2
```

Example - Gantry Mode

In applications where both the master and the slave are controlled by the DMC-41x3 controller, it may be desired to synchronize the slave with the commanded position of the master, rather than the actual position. This eliminates the coupling between the axes which may lead to oscillations.

For example, assume that a gantry is driven by A and B axes on both sides. This requires the gantry mode for strong coupling between the motors. The A axis is the master and the B axis is the slave. To synchronize B with the commanded position of A, use the instructions:

```
GA ,CA;                 'specify the commanded position of A as master for B
GR ,1;                  'set gear ratio for B as 1:1
GM ,1;                  'set gantry mode
```

Profiled position corrections can also be performed while in the electronic gearing mode. Suppose, for example, that the slave axis needs to be advanced by **10** counts:

```
IP ,10;                 'specify incremental position move of 10 on B axis
```

Under these conditions, this IP command is equivalent to:

```
PR ,10;                 'specify position relative movement of 10 on B axis
BG B;                   'begin motion on B axis
```

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

Example - Synchronize two axes with trapezoidal velocity correction

```
GA ,A;                  'define A as the master axis for B
GR ,2;                  'set gear ratio 2:1 for B
PR ,300;                'specify correction distance
SP ,5000;               'specify correction speed
AC ,100000;             'specify correction acceleration
DC ,100000;             'specify correction deceleration
SH AB;                  'servo A and B axes
BG B;                   'start correction
AM B;                   'wait until motion is completed
MO AB;                  'disable A and B axes
EN;                     'end program
```

Electronic Cam

The electronic cam is a motion control mode which enables the periodic synchronization of several axes of motion. Up to 7 axes can be slaved to one master axis. The master axis encoder must be input through a main encoder port.

The electronic cam is a more general type of electronic gearing which allows a table-based relationship between the axes. It allows synchronizing all the controller axes. For example, the DMC-4183 controllers may have one master and up to seven slaves.

To illustrate the procedure of setting the cam mode, consider the cam relationship for the Follower axis Y, when the master is X. Such a graphic relationship is shown in Figure 6.13.

Step 1. Selecting the Master Axis

The first step in the electronic cam mode is to select the master axis with the EA command.

```
EA A
```

Step 2. Specify the Master Cycle and the Change in the Slave Axis (or Axes).

In the electronic cam mode, the position of the master is always expressed modulo one cycle. In this example, the position of A 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 A and B are redefined as zero. The EM command is used to specify the master cycle and the slave cycle change.

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.

```
EM 6000,1500
```

Step 3. Specify the Master interval and starting point.

Now, the ECAM table needs to be constructed. 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 EP command.

For the given example, a table can be generated by specifying the position at the master points of 0, 2000, 4000 and 6000. The interval between each master point is 2000 counts.

```
EP 2000,0
```

Step 4. Specify the Slave Positions.

Next, the slave positions are specified with the ET command.

The table elements start at zero and may go up to 256. The parameters indicate the corresponding slave position.

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

The ECAM table has now been completed.

Step 5. Enable the ECAM

The EB command enables the ECAM mode.

Step 6. Engage the Slave Motion

The EG command is used to engage the slave axes.

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 7. Disengage the Slave motion

The EQ command disengages the slave motion.

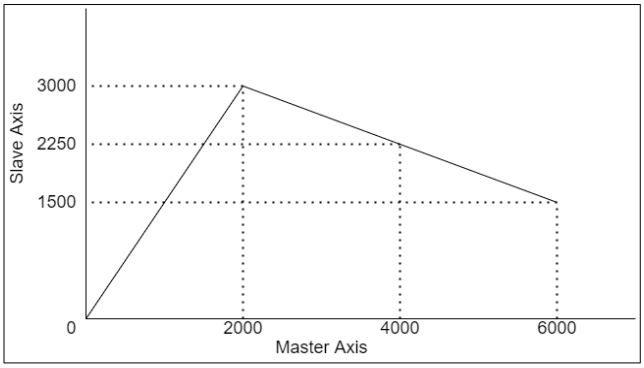


Figure 6.13: Electronic Cam Example

This disengages the Follower axis at a specified master position. If the parameter is outside the master cycle, the stopping is instantaneous. To illustrate the complete process, consider the cam relationship described by the equation:

$$B=\frac{1}{2}A+100\sin\left(0.18A\right)$$

where A 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, B varies by 1000.

A table of 100 elements will be defined, meaning increments of 20 counts each.

The following routine computes the table points. As the phase equals 0.18A and A varies in increments of 20, the phase varies by increments of 3.6°.

```
#setup;                    'label
EA A;                      'select A as Master
EM 2000,1000;              'cam cycles
EP 20,0;                   'Master position increments
n=0;                       'index
#loop;                     'loop to construct table from equation
p=n*3.6;                   '3.6 = 0.18 * 20
s=@SIN[p]*100;             'define sine position
y=n*10+s;                  'define Follower position
ET[n]= , y;                'define table
n=n+1;                     'increment index
JP#loop, (n<=100);         'repeat the process
EN;                        'end program
```

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 where A = 1000 and B = 500. This implies that B must be driven to that point to avoid a jump.

This is done with the program:

```
#run;                    'label
EB 1;                    'enable cam
PA ,500;                 'starting position
SP ,5000;                'set speed for B axis
SH AB;                   'servo A and B axes
BG B;                    'move B axis
AM B;                    'wait until B axis motion is completed
AI 1;                    'wait for start signal
EG ,1000;                'engage Follower
AI -1;                   'wait for stop signal
EQ ,1000;                'disengage Follower
MO AB;                   'disable A and B axes
EN;                      'end
```

Command Summary - Electronic CAM

COMMAND	DESCRIPTION
EA	Specifies master axes for electronic cam
EB	Enables the ECAM
EC	Sets the index into the ECAM table
EG	Engages ECAM
EM	Specifies the change in position for each axis of the CAM cycle
EP	Defines CAM table entry size and offset
EQ	Disengages ECAM at specified position
ET[n]	Defines the ECAM table entries
EW	Widen Segment (see Application Note #2444)
EY	Set ECAM cycle count

Table 6.10: List of commands related to Electronic CAM

Operand Summary - Electronic CAM

COMMAND	DESCRIPTION
_EB	Contains State of ECAM
_EC	Contains current ECAM index
_EGm	Contains ECAM status for each axis
_EMm	Contains size of cycle for each axis
_EP	Contains value of the ECAM table interval
_EQm	Contains ECAM status for each axis
_EY	Set ECAM cycle count

Table 6.11: List of operands related to Electronic CAM

Example - Electronic CAM

The following example illustrates a cam program with a master axis, C, and two slaves, A and B.

```
#a;v1=0;                    'label; initialize variable
SH ABC;                     'enable A and B axes
PA 0,0; BG AB; AM AB;       'go to position 0,0 on A and B axes
EA C;                       'C axis as the master for ECAM
EM 0,0,4000;                'change for C is 4000, 0 for A and B axes
EP 400,0;                   'ECAM interval is 400 counts with zero start
ET[0]=0,0;                  'when master is at 0 position; 1st point
ET[1]=0,0;                  '2nd point in the ECAM table
ET[2]=120,60;               '3rd point in the ECAM table
ET[3]=240,120;              '4th point in the ECAM table
ET[4]=360,180;              '5th point in the ECAM table
ET[5]=360,180;              '6th point in the ECAM table
ET[6]=360,180;              '7th point in the ECAM table
ET[7]=240,120;              '8th point in the ECAM table
ET[8]=120,60;               '9th point in the ECAM table
ET[9]=0,0;                  '10th point in the ECAM table
ET[10]=0,0;                 'starting point for next cycle
EB 1;                       'enable ECAM mode
JGC=4000;                   'set C to jog at 4000
EG 0,0;                     'engage both A and B when Master = 0
BG C;                       'begin jog on C axis
#loop;                      'loop until the variable is set
WT 20
JP #loop,(v1=0);
EQ 2000,2000;               'disengage A and B when master = 2000
MF ,, 2000;                 'wait until the master goes to 2000
ST C;                       'stop the C axis motion
AM C;                       'wait until motion is completed
EB 0                        'exit the ECAM mode
MO ABC;                     'disable axes
EN;                         'end program
```

The above example shows how the ECAM program is structured and how the commands can be given to the controller. Figure 6.14 shows the GDK scope capture of the ECAM profile. This shows how the motion will be seen during the ECAM cycles. The first trace is for the A axis, the second trace shows the cycle on the B axis and the third trace shows the cycle of the C axis.

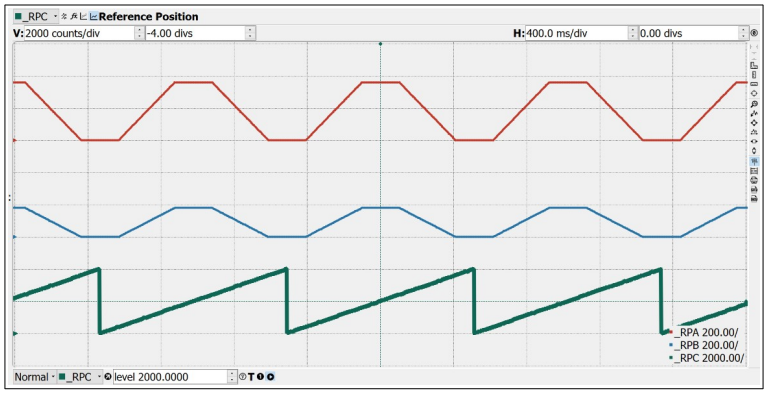


Figure 6.14: ECAM cycle with C axis as Master

PVT Mode

The DMC-41x3 controllers now supports a mode of motion referred to as “PVT.” This mode allows arbitrary motion profiles to be defined by position, velocity and time individually on all 8 axes. This motion is designed for systems where the load must traverse a series of coordinates with no discontinuities in velocity. By specifying the target position, velocity and time to achieve those parameters the user has control over the velocity profile. Taking advantage of the built in buffering the user can create virtually any profile including those with infinite path lengths.

Specifying PVT Segments

PVT segments must be entered one axis at a time using the PV command. The PV command includes the target distance to be moved and target velocity to be obtained over the specified timeframe. Positions are entered as relative moves, similar to the standard PR command, in units of encoder counts and velocity is entered in counts/second. The controller will interpolate the motion profile between subsequent PV commands using a 3rd order polynomial equation. During a PV segment, jerk is held constant, and accelerations, velocities, and positions will be calculated every other sample.

Motion will not begin until a BT command is issued, much like the standard BG command. This means that the user can fill the PVT buffer for each axis prior to motion beginning. The BT command will ensure that all axes begin motion simultaneously. It is not required for the “T” value for each axis to be the same, however if they are then the axes will remain coordinated. Each axis has a 255 segment buffer. This buffer is a FIFO and the available space can be queried with the operand **_PV**m. As the buffer empties the user can add more PVT segments.

Exiting PVT Mode

To exit PVT mode the user must send the segment command PVm=**0,0,0**. This will exit the mode once the segment is reached in the buffer. To avoid an abrupt stop the user should slow the motion to a zero velocity prior to executing this command. The controller will instantly command a zero velocity once a PVm=**0,0,0** is executed. In addition, a ST command will also exit PVT mode. When using ST, motion will come to a controlled stop using the DC value for deceleration. The same controlled stop will occur if a limit switch is activated in the direction of motion. As a result, the controller will be switched to a jog mode of motion.

Error Conditions and Stop Codes

If the buffer is allowed to empty while in PVT mode then the profiling will be aborted and the motor will come to a controlled stop on that axis with a deceleration specified by the DC command. Also, PVT mode will be exited and the stop code reported by SC will be set to **32**. During normal operation of PVT mode the stop code will be **30**. If PVT mode is exited normally (PVm=**0,0,0**), then the stop code will be set to **31**.

Additional PVT Information

It is the users’ responsibility to enter PVT data that the system’s mechanics and power system can respond to in a reasonable manner. Because this mode of motion is not constrained by the AC, DC or SP values, if a large velocity or position is entered with a short period to achieve it, the acceleration can be very high, beyond the capabilities of the system, resulting in excessive position error. The position and velocity at the end of the segment are guaranteed to be accurate but it is important to remember that the required path to obtain the position and velocity in the specified time may be different based on the PVT values. Mismatched values for PVT can result in different interpolated profiles than expected but the final velocity and position will be accurate.
The “T” value is entered in samples, which will depend on the TM setting. With the default TM of 1000, one sample is 976us. This means that a “T” value of 1024 will yield one second of motion. The velocity value, “V” will always be in units of counts per second, regardless of the TM setting. PVT mode is not available in the “-FAST” version of the firmware. If this is required please consult Galil.

Command Summary – PVT

COMMAND	DESCRIPTION
PVm=n_0,n_1,n_2	Specifies the segment of axis 'm' for a incremental PVT segment of 'n_0' counts, an end speed of 'n_1' counts/sec in a total time of 'n_2' samples.
_PVm	Contains the number of PV segments available in the PV buffer for a specified axes.
BT	Begin PVT mode
_BTm	Contains the number PV segments that have executed

Table 6.12: List of commands related to PVT mode

PVT Examples

Parabolic Velocity Profile

In this example we will assume that the user wants to start from zero velocity, accelerate to a maximum velocity of 1000 counts/second in 1 second and then back down to 0 counts/second within an additional second. The velocity profile would be described by the following equation and shown in Figure 6.15.

$$v(t) = -1000(t-1)^2 + 1000$$

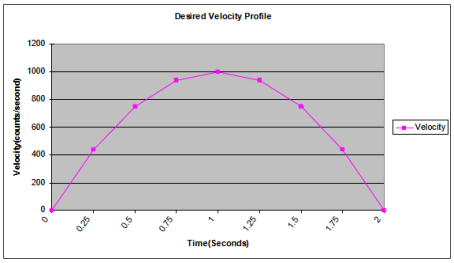


Figure 6.15: Parabolic Velocity Profile

To accomplish this we need to calculate the desired velocities and change in positions. In this example we will assume a delta time of ¼ of a second, which is 256 samples (1024 samples = 1 second with the default TM of 1000).

Velocity(counts/second)	Position(counts)
$v(t) = -1000(t-1)^2 + 1000$	$p(t) = \int (-1000(t-1)^2 + 1000)dt$
$v(.25) =$ 437.5	*p*(0 to .25) = 57
$v(.5) =$ 750	*p*(.25 to .5) = 151
$v(.75) =$ 937.5	*p*(.5 to .75) = 214
$v(1) =$ 1000	*p*(.75 to 1) = 245
$v(1.25) =$ 937.5	*p*(1 to 1.25) = 245
$v(1.5) =$ 750	*p*(1.25 to 1.5) = 214
$v(1.75) =$ 437.5	*p*(1.5 to 1.75) = 151
0	*p*(1.75 to 2) =57

The DMC program is shown below and the results can be seen in Figure 6.16.

```
#a;                          'label
PVA=57,437,256;              '1st PVT segment
PVA=151,750,256;             '2nd PVT segment
PVA=214,937,256;             '3rd PVT segment
PVA=245,1000,256;            '4th PVT segment
PVA=245,937,256;             '5th PVT segment
PVA=214,750,256;             '6th PVT segment
PVA=151,437,256;             '7th PVT segment
PVA=57,0,256;                '8th PVT segment
PVA=0,0,0;                   'termination of PVT buffer
BTA;                         'begin PVT mode
EN;                          'end program
```

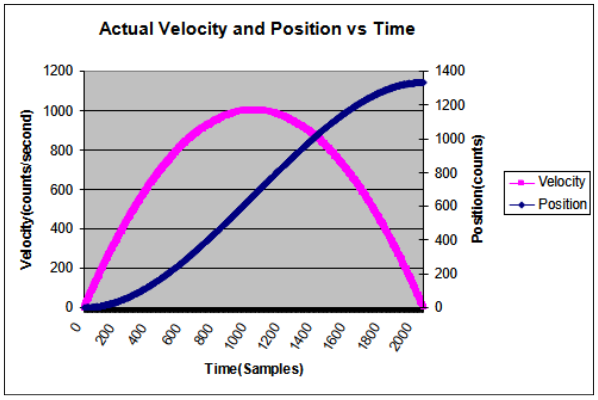


Figure 6.16: Actual Velocity and Position vs Time of Parabolic Velocity Profile

Multi-Axis Coordinated Move

Many applications require moving two or more axes in a coordinated move yet still require smooth motion at the same time. These applications are ideal candidates for PVT mode.

In this example we will have a 2 dimensional stage that needs to follow a specific profile. The application requires that the certain points be met however the path between points is not important. Smooth motion between points is critical.

The resultant DMC program is shown below. The position points are dictated by the application requirements and the velocities and times were chosen to create smooth yet quick motion. For example, in the second segment the B axis is slowed to 0 at the end of the move in anticipation of reversing direction during the next segment.

```
#a;                    'label
PVA=500,2000,500;      '1st A axis PVT segment
PVB=500,5000,500;      '1st B axis PVT segment
PVA=1000,4000,1200;    '2nd A axis PVT segment
PVB=4500,0,1200;       '2nd B axis PVT segment
PVA=1000,4000,750;     '3rd A axis PVT segment
PVB=-1000,1000,750;    '3rd B axis PVT segment
BTAB;                  'begin PVT mode on axes A and B
PVA=800,10000,250;     '4th A axis PVT segment
PVB=200,1000,250;      '4th B axis PVT segment
PVA=4000,0,1000;       '5th A axis PVT segment
PVB=-900,0,1000;       '5th B axis PVT segment
PVA=0,0,0;             'termination of PVT buffer for A axis
PVB=0,0,0;             'termination of PVT buffer for B axis
EN;                    'end program
```

NOTE: The BT command is issued prior to filling the PVT buffers and additional PV commands are added during motion for demonstration purposes only. The BT command could have been issued at the end of all the PVT points in this example.

The resultant X vs. Y position graph is shown in Figure 6.17, with the specified PVT points enlarged.

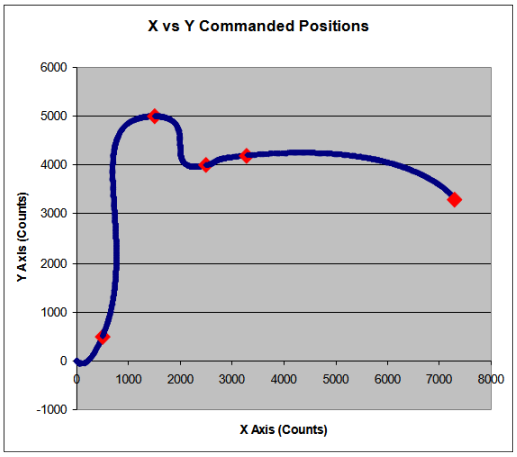


Figure 6.17: X vs Y Commanded Positions for Multi-Axis Coordinated Move

Contour Mode

The DMC-41x3 also provides a contouring mode. This mode allows any arbitrary position curve to be prescribed for 1 to 8 axes. This is ideal for following computer generated paths such as parabolic, spherical or user-defined profiles. The path is not limited to straight line and arc segments and the path length may be infinite.

Specifying Contour Segments

The Contour Mode is specified with the CM command. Any axes that are not being used in the contouring mode may be operated in other modes.

A contour is described by position increments which are described with the CD command over the time interval defined by the DT command. The time interval is defined as 2^n sample period (1 ms for TM1000), 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 sample. If the time interval changes for each segment, the CD command can be used to change the interval. Consider, for example, the trajectory shown in Figure 6.18. The position A may be described by the points:

Point 1	A=0 at T=0ms
Point 2	A=48 at T=4ms
Point 3	A=288 at T=12ms
Point 4	A=336 at T=28ms

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.

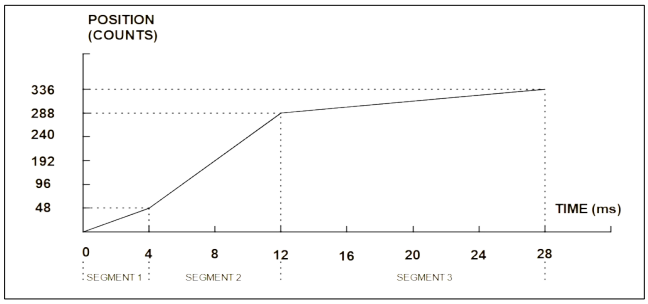


Figure 6.18: The Required Trajectory

The programmed commands to specify the above example are:

```
#main
CM A;                    'Enable Contour mode on the A axis
DT 2;                    'First segment will take 2^2=4ms
CD 48;                   'Move 48 counts in 4ms
CD 240=3;                'Move an additional 240 counts in 8ms
CD 48=4;                 'Move an additional 48 counts in 16ms
CD 0=0;                  'Issue special ending segment
#wait; JP#wait,(_CM<511); 'Loop until last segment is executed
MG “path complete”;      'at this point, profile is complete
EN;                      'End of program
```

Additional Comments

If no new data 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 question mark "?".

Command Summary - Contour Mode

COMMAND	DESCRIPTION
CM	Specifies which axes to run in Contour Mode
CD	Specifies position increment over time interval
DT	Specifies time interval

Table 6.13: List of commands related to Contour Mode

General Velocity Profiles

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. Contour mode can then be used by generating an array with data points or running a program.

Generating an Array - An Example

Consider the velocity and position profiles shown in Figure 6.19. The objective is to rotate a motor a distance of 6000 counts in 120 ms. The velocity profile is sinusoidal to reduce the jerk and the system vibration. If the position displacement can be described in terms of A counts in B milliseconds, than the motion can be described in the following manner:

$$\omega = \frac{A}{B}(1 - \cos(2\pi/B)) \qquad X = \frac{AT}{B} - \frac{A}{2\pi}\sin(2\pi/B)$$

NOTE: ω is the angular velocity; X is the position; and T is the variable, time, in milliseconds.
In the given example, A=6000 and B=120, the position and velocity profiles are:

X = 50T - (6000/2π) sin (2π T/120)

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

ω = 50 [1 - cos 2π T/120]

The DMC-41x3 can compute trigonometric functions. However, the argument must be expressed in degrees. Using our example, the equation for X is written as:

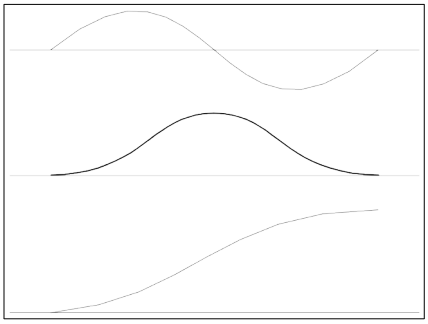


Figure 6.19: Velocity Profile with Sinusoidal Acceleration

X = 50T - 955 sin 3T

Teach (Record and Play-Back)

Several applications require teaching the machine a motion trajectory. Teaching can be accomplished using the DMC-41x3 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	DESCRIPTION
RA	Specifies which arrays to use for recording
RD	Specifies data to capture
RC	Specifies time interval
_RC	Contains if recording is in progress

Table 6.14: List of commands related to Record Array

Record and Playback Example:

```
#record;                    'begin program
DM apos[501];               'dimension array with 501 elements
RA apos[];                  'specify automatic record
RD _TPA;                    'specify A position to be captured
MO A;                       'turn A motor off
RC 2;                       'begin recording at 4 msec interval (at TM1000)
#a;                         'continue until done recording
WT 20
JP #a,(_RC=1);
#compute;                   'compute da
DM da[500];                 'dimension array for da
c=0;                        'initialize counter
#l;                         'label
d=c+1;
delta=apos[d]-apos[c];      'compute the difference
da[c]=delta;                'store difference in array
c=c+1;                      'increment index
JP #l,(c<500);              'repeat until done
#plyback;                   'begin playback
SH A;
CM A;                       'specify contour mode
DT 2;                       'specify time increment
i=0;                        'initialize array counter
#b;                         'loop counter
CD da[i]; i=i+1;            'specify contour data and increment array counter
JP #b,(i<500);              'loop until done
CD 0=0;                     'Issue ending segment
EN;                         'End of program, but profile will finish path
```

For additional information about automatic array capture, see Automatic Data Capture into Arrays in Chapter 7.

Virtual Axis

The DMC-41x3 controller has two additional virtual axes designated as the M and N axes. These axes have no encoder and no DAC. However, they can be commanded by the commands:

AC, DC, JG, SP, PR, PA, BG, IT, GA, VM, VP, CR, ST, DP, RP

The main use of the virtual axes is to serve as a virtual Master in ECAM modes, and to perform an unnecessary part of a vector mode. These applications are illustrated by the following examples.

ECAM Master Example

Suppose that the motion of the A and B axes is constrained along a path that can be described by an electronic cam table. Further assume that the ECAM Master is not an external encoder but has to be a controlled variable. This can be achieved by defining an imaginary axis as the Master with the EA command and setting the modulo of the Master. Next, the table is constructed. To move the constrained axes, simply command the N axis in the jog mode or with the PR and PA commands.

Sinusoidal Motion Example

The A axis must perform a sinusoidal motion of 10 cycles with an amplitude of 1000 counts and a frequency of 20 Hz. This can be performed by commanding the A and N axes to perform circular motion. Note that the value of VS must be:

$$VS = 2\pi * R * F$$

where R is the radius, or amplitude and F is the frequency in Hz.

Set VA and VD to maximum values for the fastest acceleration.

```
VM AN;                    'select axes for vector motion
VA 68000000;              'set acceleration
VD 68000000;              'set deceleration
VS 125664;                'VS for 20 Hz
CR 1000, -90, 3600;       'command 10 sine wave cycles
VE;                       'end vector sequence
SH A;                     'enable A axis
BG S;                     'begin motion
AM S;                     'wait until motion is completed
MO A;                     'disable A axis
EN;                       'end program
```

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

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

MT VALUE	DESCRIPTION
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, some amount of stepper motor smoothing will be applied with the KS command. 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 important when back and forth motion is commanded. 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:

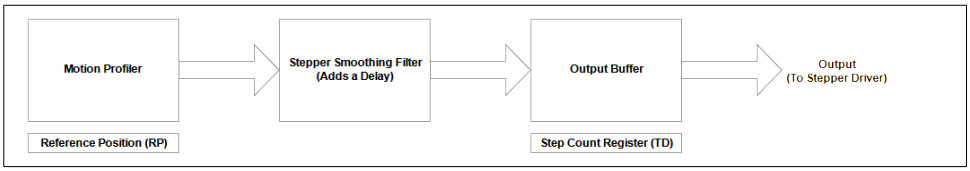


Figure 6.20: Flow Chart of how step pulses are generated

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.

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.

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.

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. 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: The auxiliary encoder is not available while operating with stepper motors.

NOTE: Closed loop operation with a stepper motor is possible with 2PB supported amplifiers. See the A5 – AMP-43547 (-D3547, -D3527) for more information.

Command Summary - Stepper Motor Operation

COMMAND	DESCRIPTION
DP	Define Reference Position and Step Count Register
DE	Define Encoder Position when using an encoder
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

Table 6.15: List of commands related to Stepper motor operation

Operand Summary - Stepper Motor Operation

OPERAND	DESCRIPTION
_TDm	Contains the value of the step count register for the 'm' axis
_TPm	Contains the value of the main encoder for the 'm' axis
_DEm	Contains the value of the step count register for the 'm' axis
_DPm	Contains the value of the main encoder for the 'm' axis
_RPm	Contains the commanded position generated by the profiler for the 'm' axis
_ITm	Contains the value of the Independent Time constant for the 'm' axis
_KSm	Contains the value of the Stepper Motor Smoothing Constant for the 'm' axis
_MTm	Contains the motor type value for the 'm' axis

Table 6.16: List of operands related to Stepper motor operation

Stepper Position Maintenance Mode (SPM)

The Galil controller can be set into the Stepper Position Maintenance (SPM) mode to handle the event of stepper motor position error. The mode looks at position feedback from the main encoder and compares it to the commanded step pulses. The position information is used to determine if there is any significant difference between the commanded and the actual motor positions. If such error is detected, it is updated into a command value for operator use. In addition, the SPM mode can be used as a method to correct for friction at the end of a microstepping move. This capability provides closed-loop control at the application program level. SPM mode can be used with Galil and non-Galil step drives.

SPM mode is configured, executed, and managed with seven commands. This mode also utilizes the **#POSERR** automatic subroutine allowing for automatic user-defined handling of an error event.

Command Summary - Stepper Motor Operation

COMMAND	DESCRIPTION
QS	Error Magnitude in pulses
YA	Step Drive Resolution in pulses/full motor step
YB	Step Motor Resolution in full motor steps/revolution
YC	Encoder resolution in counts/revolution
YR	Error Correction in pulses
YS	Enable Stepper Position Maintenance Mode
OE	Tell Position of Encoder

Table 6.17: List of operands related to Stepper Position Maintenance Mode

A pulse is defined by the resolution of the step drive being used. Therefore, one pulse could be a full step, a half step or a microstep.

When a Galil controller is configured for stepper motor operation, the step pulse output by the controller is internally fed back to the auxiliary encoder register. For SPM the feedback encoder on the stepper will connect to the main encoder port. Enabling the SPM mode on a controller with YS*m*=**1** executes an internal monitoring of the auxiliary and main encoder registers for that axis or axes. Position error is then tracked in step pulses between these two registers (QS command).

$$QS = TD - \frac{TP * YA * YB}{YC}$$

Where TD is the auxiliary encoder register(step pulses) and TP is the main encoder register(feedback encoder). Additionally, YA defines the step drive resolution where YA*m*=**1** for full stepping or YA*m*=**2** for half stepping. The YA command can be set from 1 to 9999 for microstepping drives.

Error Limit

The value of QS is internally monitored to determine if it exceeds a preset limit of three full motor steps. Once the value of QS exceeds this limit, the controller then performs the following actions:

1. The motion is maintained or is stopped, depending on the setting of the OE command. If OE*m*=**0** for an axis, it will stay in motion. If OE*m*=**1** for an axis, it will be stopped. Under SPM mode, the motor will not be disabled by position error.
2. YS is set to 2, which causes the automatic subroutine labeled **#POSERR** to be executed.

Correction

A correction move can be commanded by assigning the value of QS to the YR correction move command. The correction move is issued only after the axis has been stopped. After an error correction move has completed and QS is less than three full motor steps, the YS error status bit is automatically reset back to 1; indicating a cleared error.

Example: SPM Mode Setup

The following code demonstrates what is necessary to set up SPM mode for a 1/64th microstepping drive for an axis with a 1.8° step motor and 4000 count/rev encoder.

1/64th Step Microstepping Drive, A axis:

```
#setup
OE 1;                   'set the profiler to stop axis upon error
KS 16;                  'set step smoothing
MT -2;                  'motor type set to stepper
YA 64;                  'step resolution of the microstepping drive,
YB 200;                 'motor resolution (full steps per revolution)
YC 4000;                'encoder resolution (counts per revolution)
SH A;                   'enable axis
WT 50;                  'allow slight settle time
YS 1;                   'enable SPM mode
```

Example: Error Correction

The following code demonstrates what is necessary to set up SPM mode for the A axis, detect error, stop the motor, correct the error, and return to the main code. The drive is a full step drive, with a 1.8° step motor and 4000 count/rev encoder. When error occurs, the axis will stop due to OE1. In #POSERR, query the status YS and the error QS, correct, and return to the main code.

```
#setup
OE 1;                   'set the profiler to stop axis upon error
KS 16;                  'set step smoothing
MT -2,-2,-2,-2;         'motor type set to stepper
YA 2;                   'step resolution of the drive
YB 200;                 'motor resolution (full steps per revolution)
YC 4000;                'encoder resolution (counts per revolution)
YS 1;                   'enable stepper position maintenance mode
SH A;                   'enable axis
WT 100;                 'allow slight settle time

#motion;                'perform motion
SP 512;                 'set the speed
PR 1000;                'prepare mode of motion
BG A;                   'begin motion
EN;                     'end of program subroutine

#POSERR;                'automatic subroutine is called when _YS=2
ST A; AM A;             'stop and wait for motion to finish
WT 100;                 'wait helps user see the correction
spsave=_SPA;            'save current speed setting
JP #return,(_YSA<>2);   'return to thread zero if invalid error
SP 64;                  'set slow speed setting for correction
MG "ERROR= ",_QSA
YRA=_QSA;               'use QS for correction
MC A;                   'wait for motion to complete
```

```
MG "CORRECTED, ERROR NOW= ",_QSA
WT 100;                     'wait helps user see the correction

#return
SPA=spsave;                 'return the speed to previous setting
RE 0;                       'return from #POSERR
```

Example: Friction Correction

The following example illustrates how the SPM mode can be useful in correcting for A axis friction after each move when conducting a reciprocating motion. The drive is a 1/64th microstepping drive with a 1.8° step motor and 4000 count/rev encoder.

```
#setup;                     'set the profiler to continue upon error
KS 16;                      'set step smoothing
MT -2,-2,-2,-2;             'motor type set to stepper
YA 64;                      'step resolution of the microstepping drive
YB 200;                     'motor resolution (full steps per revolution)
YC 4000;                    'encoder resolution (counts per revolution)
SH A;                       'enable axis
WT 50;                      'allow slight settle time
YS 1;                       'enable SPM mode

#motion;                    'perform motion
SP 16384;                   'set the speed
PR 10000;                   'prepare mode of motion
BG A;                       'begin motion
MC A;                       'wait for motion to finish
JS #correct;                'move to correction
#motion2
SP 16384;                   'set the speed
PR -10000;                  'prepare mode of motion
BG A;                       'begin motion
MC A;                       'wait for motion to finish
JS #correct;                'move to correction
JP #motion

#correct;                   'correction code
spa=_SPA
#loop;                      'save speed value
SP 2048;                    'set a new slow correction speed
WT 100;                     'stabilize
JP #end,(@ABS[_QSA]<10);    'end correction if error is within defined tolerance
YRA=_QSA;                   'correction move
MC A
WT 100;                     'stabilize
JP #loop;                   'keep correcting until error is within tolerance
#end;                       'end #CORRECT subroutine, returning to code
SPA=spa
EN
```

Dual Loop (Auxiliary Encoder)

The DMC-41x3 provides an interface for a second encoder for each axis except for axes configured for stepper motor operation and axis used in circular compare. 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 a standard quadrature type, or it may provide pulse and direction. The controller also offers the provision for inverting the direction of the encoder rotation. The main and the auxiliary encoders are configured with the CE command. The DE command can be used to define the position of the auxiliary encoders while the the TD command returns the current position of the auxiliary encoders. The command DV configures the auxiliary encoder to be used for dual loop control.

Backlash Compensation

There are two methods for backlash compensation using the auxiliary encoders: continuous dual loop and sampled dual loop.
To illustrate the problem, consider a situation in which the coupling between the motor and the load has 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 controller's main encoder inputs and connect the motor encoder to the auxiliary encoder inputs. 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. 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 KPm=**0**, KIm=**0** and to maximize the value of KD under the condition DVm=**1**. Once KD is found, increase KP gradually to a maximum value, and finally, increase KI, if necessary. Dual loop can be enabled for a specific axis with the DVm=**1** command or disabled with the DVm=**0** command.

Sampled Dual Loop - Example

In this example, 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 A axis and connect the linear encoder to the auxiliary encoder of A. Assume that the required motion distance is one inch, which 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.

```
#dloop
CE 0;DE 0;                  'configure encoder and set the initial value
error=50;                   'define error limit for linear encoder
SH A;PR 40000;BG A;         'enable A axis, define move, begin motion


#correct;                   'correction loop
AM A;                       wait for motion completion
v1=10000-_TDA;              'find linear encoder error
v2=-_TEA/4+v1;              'compensate for motor error
JP #end,(@ABS[v2]<error);   'exit if error is small
PR v2*4;                    'correction move
BG A;                       'start correction
JP #correct;                'repeat correction move
#end;EN;                    'end program
```

Motion Smoothing

The DMC-41x3 controller allows the smoothing of the velocity profile to reduce the mechanical vibration of the system.

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

Using the IT Command:

When operating with servo motors, motion smoothing can be accomplished with the IT command. This command filters the acceleration and deceleration functions to produce a smooth velocity profile. The resulting velocity profile, has continuous acceleration and results in reduced mechanical vibrations.

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

The smoothing parameter used is a number between **0** and **1** to determine the degree of filtering. The maximum value of **1** implies no filtering, resulting in trapezoidal velocity profiles. Smaller values of the smoothing parameters imply heavier filtering and smoother moves.

The following example illustrates the effect of smoothing. Figure 6.21 shows the trapezoidal velocity profile and the modified acceleration and velocity.

NOTE: Enabling motion smoothing results in longer motion time.

Example – Smoothing

```
PR 20000;                'define position move
AC 100000;               'define acceleration
DC 100000;               'define deceleration
SP 5000;                 'define speed
IT .5;                   'filter for smoothing
SH A;                    'enable A axis
BG A;                    'begin motion
AM A;                    'wait until motion is finished
MO A;                    'disable A axis
EN;                      'end program
```

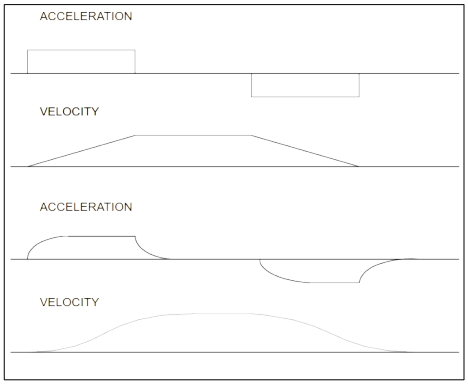


Figure 6.21: Acceleration and trapezoidal velocity without smoothing (above) and smoothed acceleration and trapezoidal velocity profiles (below)

Using the KS Command (Step Motor Smoothing)

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

The smoothing parameter used is a number between **0.5** and **128** and determines the degree of filtering. The minimum value of **0.5** implies the least filtering, resulting in trapezoidal velocity profiles. Larger values of the smoothing parameters imply heavier filtering and smoother moves.

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 _HMA reads **1** initially, the motor will go in the reverse direction first (direction of decreasing encoder counts). If _HMA 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** , a normally open switch will make _HMA read **1** initially, and a normally closed switch will make _HMA read 0. Furthermore, with CN,**1** a normally open switch will make _HMA read 0 initially, and a normally closed switch will make _HMA 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 HV 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 HV counts/sec until the encoder index pulse is detected. The motor then decelerates to a stop and goes back to the index. The DMC-41x3 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.

Home Switch Configuration			Direction of Motion		
Switch Type	CN Setting	Initial _HMA 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

Table 6.18: Possible directions of motions for each stage of the Home command

Example: Homing

```
#home;                  'label
CN ,-1;                 'configure the polarity of the home input
AC 1000000;             'acceleration rate
DC 1000000;             'deceleration rate
SP 5000;                'speed for stage 1
SH A;                   'enable A axis
HM A;                   'home A axis
BG A;                   'begin motion
AM A;                   'after complete
MG "AT HOME";           'send message
MO A;                   'disable A axis
EN;                     'end program
```

Figure 6.22 shows the velocity profile from the homing sequence of the example program above. For this profile, the switch is normally closed and CN,-1.

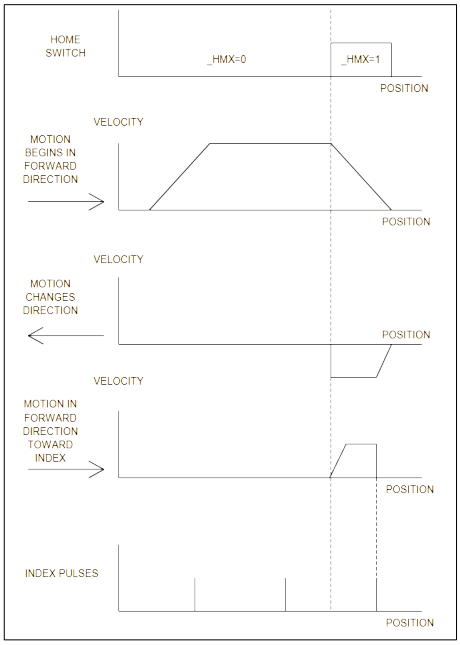


Figure 6.22: Homing Sequence for Normally Closed Switch and CN,-1

Example: Find Edge

```
#edge;                    'label
AC 2000000;               'acceleration rate
DC 2000000;               'deceleration rate
SP 8000;                  'speed
SH A;                     'enable A axis
FE A;                     'find edge command
BG A;                     'begin motion
AM A;                     'after complete
MG "FOUND HOME";          'send message
DP 0;                     'define position as 0
MO A;                     'disable A axis
EN;                       'end program
```

Command Summary - Homing Operation

COMMAND	DESCRIPTION
FE	Defines motion that searches for the Home Input
FI	Defines motion that searches for the Index Input
HM	Home routine that is a combination of FE and FI
SC	Stop Code that tells why the motor is not moving
TS	Tell Status of Switches and Inputs

Table 6.19: List of operands related to Homing

Operand Summary - Homing Operation

OPERAND	DESCRIPTION
_HMm	Contains the state of the Home Input for the 'm' axis
_SCm	Contains stop code for the 'm' axis
_TSm	Contains status of switches and inputs for the 'm' axis

Table 6.20: List of operands related to Homing

High Speed Position Capture (The Latch Function)

Often it is desirable to capture the position precisely for registration applications. Position capture can be programmed to latch on either a corresponding input (see Table 6.21) or on the index pulse for that axis. The position can be captured for either the main or auxiliary encoder within 25 microseconds of a high-to-low transition on the CPU's input. A high-to-low transition on the CPU's input will occur when current begins to flow through the digital input's optoisolator. The input voltage that causes current to flow through the optoisolator depends on the reference voltage supplied to INCOM. See Figure 3.1: Digital Inputs 1-8 (DI[8:1]).

Input 1	A-axis latch	Input 9	E-axis latch
Input 2	B-axis latch	Input 10	F-axis latch
Input 3	C-axis latch	Input 11	G-axis latch
Input 4	D-axis latch	Input 12	H-axis latch

Table 6.21: Inputs and corresponding axis latch

NOTE	Latching is not valid with sampled feedback types such as: SSI, BiSS, and Analog

To ensure a position capture within 25 microseconds, the input signal must be a transition from high to low. Low to high transitions may have greater delay.

The AL and RL commands are used to arm the latch and report the latched position respectively. The latch must be re-armed after each latching event. See the Command Reference for more details on these commands.

Chapter 7 Application Programming

Overview

The DMC-41x3 provides a powerful programming language that allows users to customize the controller for their particular application. Programs can be downloaded into the DMC-41x3 memory freeing the host computer for other tasks. However, the host computer can send commands to the controller at any time, even while a program is being executed. Only ASCII commands can be used for application programming.

In addition to standard motion commands, the DMC-41x3 provides commands that allow the DMC-41x3 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-41x3 provides user-defined variables, arrays and arithmetic functions. For example, with a cut-to-length operation, the length can be specified as a variable in a program which the operator can change as necessary.

The following sections in this chapter discuss all aspects of creating applications programs. The program memory size is 80 characters x 4000 lines.

Program Format

A DMC-41x3 program consists of DMC 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-41x3 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 80 characters. A carriage return enters the final command on a program line.

Using Labels in Programs

All DMC-41x3 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 in label.

The maximum number of labels which may be defined is 510.

Valid labels

```
#begin
#square
#a1
#begin1
```

Invalid labels

```
#1Square
#123
```

A Simple Example Program:

```
#start;                    'beginning of the program
SH AB;                     'enable A and B axes
#loop;                     'label for loop
PR 10000,20000;            'specify relative distances on A and B axes
BG AB;                     'begin motion
AM AB;                     'wait for motion complete
WT 2000;                   'wait 2 sec
JP #loop;                  'jump to loop label
EN;                        'end program
```

The above program moves A and B by **10000** and **20000** units respectively. After the motion is complete, the motors rest for 2 seconds. The cycle repeats indefinitely until the stop command is issued.

Special Labels

The DMC-41x3 have some special labels, which are used to define input interrupt subroutines, limit switch subroutines, error handling subroutines, and command error subroutines. See section on Auto-Start Routine.

LABEL	DESCRIPTION
#AMPERR	Amplifier error routine
#AUTO	Automatically run upon controller power up or reset
#AUTOERR	Automatically run if there is an EEPROM error out of reset
#CMDERR	Incorrect command subroutine
#COMINT	Communications Interrupt (See CC Command)
#FWERR	Firmware error subroutine, runs on sample overflow
#ININT	Input Interrupt subroutine (See II Command)
#LIMSWI	Limit Switch subroutine
#MCTIME	Timeout on Motion Complete trippoint
#POSERR	Position error subroutine
#TCPERR	TCP error subroutine

Table 7.1: List of labels of automatic subroutines

Commenting Programs

REM vs. NO or ' comments

There are 2 ways to add comments to a .dmc program. *REM* statements or **NO** and **'** comments. The main difference between the 2 is that *REM* statements are stripped from the program upon download to the controller and **NO** or **'** comments are left in the program. In most instances the reason for using *REM* statements instead of **NO** or **'** is to save program memory. The other benefit to using *REM* commands comes when command execution of a loop, thread or any section of code is critical. Although they do not take much time, **NO** and **'** comments still take time to process. So when command execution time is critical, *REM* statements should be used.
The GDK software will treat an apostrophe (**'**) comment different from an **NO** when the compression algorithm is activated upon a program download (line > 80 characters or program memory > 4000 lines). In this case the software will remove all (**'**) comments as part of the compression and it will download all **NO** comments to the controller.

The DMC-41x3 provides a command, **NO**, for commenting programs or single apostrophe. This command allows the user to include up to 78 characters on a single line after the **NO** command and can be used to include comments from the programmer as in the following example:

```
#path
'2D circular path
SH AB
VM AB
'vector motion on A and B
VS 10000
'vector speed is 10000
VP -4000,0;' bottom line
CR 1500,270,-180
REM half circle motion
VP 0,3000
NO top line
CR 1500,90,-180
NO half circle motion
VE; NO end vector sequence
REM end vector sequence
BG S
'begin sequence motion
EN
'end program
```

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

Executing Programs - Multitasking

The DMC-41x3 can run up to 8 independent threads simultaneously. These threads are numbered 0 through 7, where 0 is the main thread. Multitasking is useful for executing independent operations such as PLC functions that occur independently of motion.

The main thread differs from the others that when input interrupts are implemented for limit switches, position errors or command errors, the subroutines are executed as thread 0.

To begin execution of the various programs, use the XQ command. To halt the execution of any thread, use the HX command. Note that both the XQ and HX commands can be performed by an executing program.

The example below produces a waveform on Output 1 independent of a move.

```
#task1;                    'task1 label
AT 0;                      'initialize reference time
CB 1;                      'clear Output 1
#loop1;                    'loop1 label
AT 10;                     'wait 10 msec from ref time
SB 1;                      'set Output 1
AT -40;                    'wait 40 msec from ref time, reinitialize ref
CB 1;                      'clear Output 1
JP #loop1;                 'jump back to #loop1 label
#task2;                    'task2 label
XQ #task1,1;               'execute task1 on thread 1
SH A;                      'enable A axis
#loop2;                    'loop2 label
PR 1000;                   'define relative distance
BG A;                      'begin motion
AM A;                      'wait until motion is completed
WT 10                      'wait 10 msec
JP #loop2,(@IN[2]=1);      'jump to #loop2 label while Input 2 is not active
HX;                        'halt all threads
EN;                        'end program
```

The program above is executed with the command XQ #task2,0 which designates task2 as the main thread (i.e. Thread 0). #task1 is executed within #task2.

Debugging Programs

The DMC-41x3 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, TR**1**. TR**0** 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.

Error Code Command

The user can obtain information about the type of error condition that occurred by using the command, TC**1**. This command reports back a number and a text message which describes the error condition. The command, TC**0** 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.

RAM Interrogation Commands

For debugging the status of the program memory, array memory, or variable memory, the DMC-41x3 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-41x3 will have a maximum of 24000 array elements in up to 30 arrays. If an array of 100 elements is defined, the command DM? will return the value 23900 and the command DA? will return 29.

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, **_ED0** contains the last line of program execution, the command MG **_ED0** will display this line number.

OPERAND	DESCRIPTION
_ED0	Contains the last line of program execution
_DL	Contains the number of available labels
_UL	Contains the number of available variables
_DA	Contains the number of available arrays
_DM	Contains the number of available array elements

Table 7.2: List of operands related to memory on the DMC-41x3

Debugging Example:

The following program has an error. It attempts to specify a relative movement while the A axis is already in motion. When the program is executed, the controller stops at line 3. The user can then query the controller using the command, TC1. The controller responds with the corresponding explanation:

```
#a                          'program Label
SH A;                       'enable A axis
PR 1000;                    'position relative move of 1000 counts
BG A;                       'begin motion
PR 5000;                    'position relative move of 5000 counts
EN;                         'end program
```

```
From Terminal
:XQ #a
:?4 PR 5000
TC1
7 Command not valid while
running
```

Change line 4 of #a to BGA;AMA and re-download the program.

```
:XQ #a
```

Program Flow Commands

The DMC-41x3 provides instructions to control program flow. The controller program sequencer normally executes program instructions sequentially. The program flow can be altered with the use of event triggers, trippoints, and conditional jump statements.

Event Triggers & Trippoints

To function independently from the host computer, the DMC-41x3 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-41x3 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 controller can make decisions based on its own status or external events without intervention from a host computer.

DMC-41x3 Event Triggers

COMMAND	DESCRIPTION
AM	Pauses code until motion is complete on the specified axes or motion sequence
AD	Pauses code until reference position has reached the specified relative distance from the start of the move, only one axis may be specified at a time
AR	Pauses code until after specified distance from the last AR or AD command has elapsed, only one axis may be specified at a time
AP	Pauses code until after absolute position occurs, only one axis may be specified at a time
MF	Pauses code until after forward motion reached absolute position, only one axis may be specified
MR	Pauses code until after reverse motion reached absolute position, only one axis may be specified
MC	Pauses code until after the reference position and the encoder has passed the specified position
AI	Pauses code until after specified input is at specified logic level
AS	Pauses code until specified axis has reached its slew speed
AT	Pauses code until specified time has elapsed from the last time reference created
AV	Pauses code until specified distance along a coordinated path has occurred
WT	Pauses code until specified time in msec or samples has elapsed

Table 7.3: List of trippoint commands that halt code execution until an event occurs

Event Trigger - Multiple Move Sequence

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

```
#twomove;                    'program label
SH A;                        'enable A axis
PR 2000;                     'position relative command
BG A;                        'begin motion
AM A;                        'wait for motion to finish
PR 4000;                     'next position relative command
BG A;                        'begin 2nd move
AM A;                        'wait for motion to finish
MO A;                        'disable A axis
EN;                          'end program
```

Event Trigger - Set Output after Distance

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

```
#setbit;                'program label
SH A;                   'enable A axis
SP 10000;               'speed is 10000 cts/sec
PA 20000;               'specify absolute position
BG A;                   'begin motion
AD 1000;                'wait until 1000 counts
SB 1;                   'set Output 1
AM A;                   'wait for motion to finish
MO A;                   'disable A axis
EN;                     'end program
```

Event Trigger - Repetitive Position Trigger

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

```
#trip;                  'program label
JG 50000;               'specify jog speed
SH A;                   'enable A axis
BG A;n=0;               'begin motion, create counter variable
#loop;                  'label for loop
AR 10000;               'wait 10000 counts
TP A;                   'tell position
SB 1;                   'set Output 1
WT 50;                  'wait 50 msec
CB 1;                   'clear Output 1
n=n+1;                  'increment counter
JP #loop,(n<5);         'repeat 5 times
ST A;                   'stop motion
AM A;                   'wait for motion to finish
EN;                     'end program
```

Event Trigger - Start Motion on Input

This example waits for input 1 to become active and then starts motion. The AI command actually halts execution of the program until the input occurs. Input Interrupt command (II) or a conditional jump on an input can be used if code execution should not be paused.

```
#input;                 'program label
AI -1;                  'wait for input 1 to be active
SH A;                   'enable A axis
PR 10000;               'position relative command
BG A;                   'begin motion
AM A;                   'wait for motion to finish
MO A;                   'disable A axis
EN;                     'end program
```

Event Trigger - Set Output when at speed

```
#atspeed;                'program label
JG 50000;                'specify jog speed
AC 10000;                'set acceleration rate
SH A;                    'enable A axis
BG A;                    'begin motion
AS A;                    'wait for at slew speed 50000
SB 1;                    'set Output 1
ST A;                    'stop motion
AM A;                    'wait for motion to finish
MO A;                    'disable A axis
EN;                      'end program
```

Event Trigger - Change Speed along Vector Path

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

```
#vector;                 'program label
VM AB;                   'define S vector plane as A and B axes
VS 5000;                 'set vector speed
VP 10000,20000;          'define vector position point
VP 20000,30000;
VE;                      'end vector sequence
SH AB;                   'enable A and B axes
BG S;                    'begin sequence on S plane
AV 5000;                 'wait until passed vector distance
VS 1000;                 'reduce speed
AM S;                    'wait for motion to finish
MO AB;                   'disable A and B axis
EN;                      'end program
```

Event Trigger - Multiple Move with Wait

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

```
#moves;                  'program label
PR 12000;                'define position relative move
SP 20000;                'set speed
AC 100000;               'set acceleration
SH A;                    'enable A axis
BG A;                    'begin motion
AD 10000;                'wait until position is reached
SP 5000;                 'set new speed
AM A;                    'wait for motion to finish
WT 200;                  wait 200 ms
PR -10000;               'define new position move
SP 30000;                'set new speed
AC 150000;               'set new acceleration
BG A;                    'begin motion
AM A;                    'wait for motion to finish
MO A;                    'disable A axis
EN;                      'end program
```

Define Output Waveform Using AT

The following program causes Output 1 active for 10 msec and inactive for 40 msec. This cycle repeats indefinitely.

```
#output;                'program label
AT 0;                   'initialize time reference
SB 1;                   'set Output 1
#loop;                  'loop label
AT 10;                  'wait for 10 samples from ref
CB 1;                   'clear Output 1
AT -40;                 'wait for 40 samples from ref, reset ref
SB 1;                   'set Output 1
JP #loop;               'jump back to loop label
EN;                     'end program
```

Conditional Jumps

The DMC-41x3 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 controller to make decisions without a host computer. For example, the DMC-41x3 can decide between two motion profiles based on the state of an input line.

Command Format - JP and JS

FORMAT	DESCRIPTION
JS #label, (logical condition)	Jump to label if logical condition is satisfied, return when routine is finished
JP #label, (logical condition)	Jump to location if logical condition is satisfied

Table 7.4: Format and description for conditional jump commands

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

Logical operators:

OPERATOR	DESCRIPTION
<	Less than
>	Greater than
=	Equal to
<=	Less than or equal to
>=	Greater than or equal to
<>	Not equal

Table 7.5: List of operators available for conditional statements

Conditional Statements

The conditional statement is satisfied if it evaluates to any value other than zero. The conditional statement can be any valid DMC-41x3 numeric operand, including variables, array elements, numeric values, functions, keywords, and arithmetic expressions. If no conditional statement is given, the jump will always occur.

Number	`v1=6`
Numeric Expression	`v1=v7*6`
	`@ABS[v1]>10`
Array Element	`v1<count[2]`
Variable	`v1<v2`
Operand	`_TPA=0`
	`_TVA>500`
I/O	`v1>@AN[2]`
	`@IN[1]=0`

Multiple Conditional Statements

The DMC-41x3 will accept multiple conditions in a single jump statement. The conditional statements are combined in pairs using the operands & and |. 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-41x3 executes operations from left to right. See Mathematical and Functional Expressions for more information.

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
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.

Example Using JP command:

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

```
#begin;                 'program label
count=10;               'initialize loop counter
SH A;                   'enable A axis
#loop;                  'label for loop
PA 1000;                'position absolute move to 1000
BG A;                   'begin move
AM A;                   'wait for motion to finish
WT 100;                 'wait 100 msec
PA 0;                   'position absolute move to 0
BG A;                   'begin move
AM A;                   'wait for motion complete
```

```
WT 100;                 'wait 100 msec
count=count-1;          'decrement loop counter
JP #loop,(count>0);     'jump to #loop while count is greater than 0
MO A;                   'disable A axis
EN;                     'end program
```

Using IF, ELSE, and ENDIF Commands

The DMC-41x3 provides a structured approach to conditional statements using **IF**, **ELSE** and **ENDIF** commands.

Using the IF and ENDIF Commands

An IF conditional statement is formed by the combination of an **IF** and **ENDIF** command. The **IF** command can have one or more conditional statements. If the conditional statement(s) evaluate true, the command interpreter will continue executing commands which follow the **IF** command. If the conditional statement evaluates false, the controller will ignore commands until the associated **ENDIF** command is executed or an **ELSE** command occurs in the program (see discussion of **ELSE** command below).

NOTE: An **ENDIF** command must always be executed for every **IF** command that has been executed. It is recommended that the user not include jump commands inside **IF** conditional statements since this causes re-direction of command execution. In this case, the command interpreter may not execute an **ENDIF** command.

Using the ELSE Command

The **ELSE** command is an optional part of an **IF** conditional statement and allows for the execution of command only when the argument of the **IF** command evaluates False. The **ELSE** command must occur after an **IF** command and has no arguments. If the argument of the **IF** command evaluates false, the controller will skip commands until the **ELSE** command. If the argument for the **IF** command evaluates true, the controller will execute the commands between the **IF** and **ELSE** command.

Nesting IF Conditional Statements

The DMC-41x3 allows for **IF** conditional statements to be included within other **IF** conditional statements. This technique is known as 'nesting' and the DMC-41x3 allows up to 255 **IF** conditional statements to be nested. This is a very powerful technique allowing the user to specify a variety of different cases for branching.

Command Format - IF, ELSE and ENDIF

COMMAND	DESCRIPTION
IF (conditional statements)	Execute following commands if conditional statements are true, otherwise continue executing at **ELSE** or **ENDIF** command
ELSE	Execute following commands if conditional statements are false, optional command
ENDIF	Command to end IF conditional statement

Example using IF, ELSE and ENDIF:

```
#test;                  'program label
II,,3;                  'enable input interrupts on Inputs 1 and 2
MG "WAITING FOR INPUT 1,  'message
INPUT 2";
#loop;                  'label for endless loop
WT 10;                  'wait
JP #loop;               'endless loop
EN;                     'end of main program
```

```
#ININT;                    'Input Interrupt Subroutine
WT 50;                     'wait
IF (@IN[1]=0);             'IF statement based on input 1
IF (@IN[2]=0);             '2nd IF statement if 1st IF is true
MG "INPUT 1 AND INPUT 2    'message if 2nd IF is true
ARE ACTIVE";
ELSE;                      'ELSE for 2nd IF statement
MG "ONLY INPUT 1 IS        'message if 2nd IF is false
ACTIVE";
ENDIF;                     'end of 2nd IF statement
ELSE;                      'ELSE for 1st IF statement
MG "ONLY INPUT 2 IS        'message if 1st IF statement is false
ACTIVE";
ENDIF;                     'end of 1st IF statement
#wait;                     'label to be used for a loop
JP #wait,((@IN[1]=0) |     'loop until both inputs are not active
(@IN[2]=0));
RI 0;                      'return to #test without restoring trippoints
```

Subroutines

A subroutine is a group of instructions beginning with a label and ending with an EN command. Subroutines are called from the main program with the jump subroutine instruction JS, followed by a label or line number, and conditional statement. 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. An example of a subroutine to draw a square 500 counts per side is given below. The square is drawn at vector position 1000,1000.

```
#main;                     'begin main program
CB 1;                      'clear Output 1, pick up pen
SH AB;                     'enable A and B axes
VM AB;                     'define S vector plane as A and B axes
VP 1000,1000;              'define vector position, move pen
LE;                        'finish vector sequence
BG S;                      'begin motion
AM S;                      'wait for motion to finish
SB 1;                      'set Output 1, put down pen
JS #square;                'jump to square subroutine
CB 1;                      'clear Out#put 1, pick up pen
EN;                        'end main program
#square;                   'square subroutine
v1=500;                    'define length of side
JS #l;                     'jump to #l subroutine
v1=-v1;                    'switch direction
JS #l;                     'jump to #l subroutine
EN;                        'end subroutine
#l;                        'start of #l subroutine
PR v1,v1;                  'define move for length of side
BG A;                      'begin motion on A axis only
AM A;                      'wait for motion to finish on A axis only
BG B;                      'begin motion on B axis only
AM B;                      'wait for motion to finish on B axis only
EN;                        'end subroutine
```

Stack Manipulation

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

Auto-Start Routine

The DMC-41x3 has a special label for automatic program execution. A program which has been saved into the controller's 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-41x3 program sequences. The controller 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:

LABEL	DESCRIPTION
#AMPERR	Amplifier error routine
#AUTO	Automatically run upon controller power up or reset
#AUTOERR	Automatically run if there is an EEPROM error out of reset
#CMDERR	Incorrect command subroutine
#COMINT	Communications Interrupt (See CC Command)
#FWERR	Firmware error subroutine, runs on sample overflow
#ININT	Input Interrupt subroutine (See II Command)
#LIMSWI	Limit Switch subroutine
#MCTIME	Timeout on Motion Complete trippoint
#POSERR	Position error subroutine
#TCPERR	TCP error subroutine

Table 7.6: List of labels of automatic subroutines

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 #CMDERR to function.

Example using Limit Switches:

This program prints a message upon the occurrence of a limit switch. For the #LIMSWI routine to function, the DMC-41x3 must be executing an applications program from memory. This can be a very simple program that does

nothing but loop on a statement, such as #LOOP;JP #LOOP;EN. Motion commands, such as JG 5000 can still be sent from the PC even while the "dummy" applications program is being executed.

```
#loop;                    'dummy program
WT 20
JP #loop;EN;              'jump to loop
#LIMSWI;                  'limit switch label
WT 1000;
MG "LIMIT OCCURRED";      'print message
RE;                       'return to main program
                          'download program
:XQ #LOOP                 execute dummy program
:JG 5000                  jog
:BG A                     begin motion
```

Now, when a forward limit switch occurs on the A axis, the #LIMSWI subroutine will be executed.
Notes regarding the #LIMSWI Routine:

1) The RE command is used to return from the #LIMSWI subroutine.
2) The #LIMSWI subroutine will be re-executed if the limit switch remains active.

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

Example - Position Error

```
#loop;                    'dummy program
WT 10;
JP #loop;                 'loop
EN;
#POSERR;                  'position error routine
v1=_TEA;                  'read position error
ST A;
AM A;
MO A;
MG "EXCESS POSITION       'print message
ERROR";
MG "ERROR=",v1;           'print error
WT 1000;
RE;                       'return from error
                          download program
:XQ #LOOP                 execute dummy program
:JG 100000                jog at high speed
:BGA                      begin motion
```

Example - Input Interrupt

```
#a;                         'label
II 1;                       'input interrupt on 1
JG 30000,,,60000;           'jog
SH AD;                      'enable A and D axes
BG AD;                      'begin motion
#loop;                      'loop
WT 10;
JP#loop;
EN;
#ININT;                     'input interrupt
ST AD;AM AD;                'stop motion
#test;                      'test for input 1 still low
JP #test, (@IN[1]=0);
JG 30000,,,6000;            'restore velocities
BG AD;                      'begin motion
RI 0;                       'return from interrupt routine, do not re-enable
                            trippoints
```

Example - Motion Complete Timeout

```
#begin                      'begin main program
TW 1000;                    'set the time out to 1000 ms
PA 10000;                   'position absolute command
SH A;                       'enable A axis
BG A;                       'begin motion
MC A;                       'motion complete trippoint
EN;                         'end main program
#MCTIME;                    'motion complete subroutine
MG "A axis fell short";     'send out a message if move doesn't finish in 1000ms
EN;                         'end subroutine
```

Example - Command Error

```
#main;                      'begin main program
speed=2000;                 'set variable for speed
JG speed;                   'define jog speed
SH A;                       'enable A axis
BG A;                       'begin motion
#loop;                      'label for loop
JG speed;                   'update jog speed based upon speed variable
WT 100;                     'wait
JP #loop;                   'jump to #loop label
EN;                         'end main program
#CMDERR;                    'command error subroutine
JP #done,(_ED0<>2);         'check if error on line 2
JP #done,(_TC<>6);          'check if out of range for JG command
MG "SPEED TOO HIGH";        'send message
MG "TRY AGAIN";
speed=_JGA;                 'keep current jog speed
#done;
ZS 1;                       'zero stack
JP #loop;                   'return to main loop
EN;                         'end program
```

The program on the previous page 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.

In multitasking applications, there is an alternate method for handling command errors from different threads. Using the XQ command along with the special operands described below allows the controller to either skip or retry invalid commands.

The following example shows an error correction routine which uses the operands.

Example - Command Error with Multitasking

```
#a;                     'begin thread 0 as a loop
XQ #b,1;                'execute #b on another thread
#loop;                  'beginning of loop
WT 500;                 'wait
JP #loop;               'jump to #loop label
EN;                     'end of thread 0

#b;                     'begin thread 1
n=-1;                   'create new variable
KP n;                   'set KP to value of n, an invalid value
EN;                     'end of thread 1

#CMDERR;                'begin command error subroutine
error=_TC;              'log error in variable
IF (error=6);           'if error is out of range (KP -1)
n=1;                    'set n to a valid number
XQ _ED0,_ED1;           'retry KP command
error=0;                'reset error variable
ENDIF;
EN;                     'end of command error routine
```

Example - Communication Interrupt

A DMC-4113 is used to move the A axis back and forth from 0 to 10000. This motion can be paused, resumed and stopped via input from the USB port.

```
#main;                  'label for main of program
CI 2;                   'setup communication interrupt for USB port
MG {P1}"Type 0 to stop  'send message to USB port
motion";
MG {P1}"Type 1 to pause 'send message to USB port
motion";
MG {P1}"Type 2 to resume 'send message to USB port
motion";
rate=2000;              'variable to remember speed
SPA=rate;               'set speed of a axis motion
SH A;                   'enable A axis
#loop;                  'label for loop
PAA=10000;              'move to absolute position 10000
BG A;                   'begin motion on a axis
AM A;                   'wait for motion to be complete
PAA=0;                  'move to absolute position 0
BG A;                   'begin motion on a axis
AM A;                   'wait for motion to be complete
JP #loop;               'continually loop to make back and forth motion
EN;                     'end main program
```

```
#COMINT;                    'interrupt routine
JP #stop,(P1CH="0");        'check for s (stop motion)
JP #pause,(P1CH="1");       'check for p (pause motion)
JP #resume,(P1CH="2");      'check for r (resume motion)
EN 1,1;                     'do nothing
#stop;                      'routine for stopping motion
ST A;                       'stop motion
ZS;                         'zero program stack
EN;                         'end program
#pause;                     'routine for pausing motion
rate=_SPA;                  'save current speed setting of a axis motion
SPA=0;                      'set speed of a axis to zero (allows for pause)
EN 1,1;                     're-enable trippoint and communication interrupt
#resume;                    'routine for resuming motion
SPA=rate;                   'set speed on a axis to original speed
EN 1,1;                     're-enable trippoint and communication interrupt
```

For additional information, see section on Using Communication Interrupt.

JS Subroutine Stack Variables (^a, ^b, ^c, ^d, ^e, ^f, ^g, ^h)

There are 8 variables that may be passed on the subroutine stack when using the JS command. Passing values on the stack is advanced DMC programming, and is recommended for experienced DMC programmers familiar with the concept of passing arguments by value and by reference.

1. Passing parameters has no type checking, so it is important to exercise good programming style when passing parameters. See examples below for recommended syntax.
2. Do not use spaces in expressions containing ^.
3. Global variables MUST be assigned prior to any use in subroutines where variables are passed by reference.
4. Please refer to the JS command in the controller's command reference for further important information.

Example: A Simple Adding Function

```
#add;                           'label for main program
JS #sum(1,2,3,4,5,6,7,8)        'call subroutine and pass values
MG_JS                           'print return value
EN
#sum                            '(^a,^b,^c,^d,^e,^f,^g,^h) syntax note for use
EN,,(^a+^b+^c+^d+^e+^f+^g+^h)   'return sum

:XQ                             'loop for main program
36.0000
```

Example: Variable, and an Important Note about Creating Global Variables

```
#var;                           'label for main program
value=5;                        'value to be passed by reference
global=8;                       'global variable
JS #sum(&value,1,2,3,4,5,6,7);  'first argument passed by reference
MG value;                       'message value after subroutine
MG _JS;                         'message returned value
EN;
```

```
#sum;
^a=^b+^c+^d+^e+^f+^g+^h+
global
EN,,^a

:XQ                              'loop for main program
36.0000
36.0000
```

Example: Working with Arrays

```
#array;                     'label for main program
DM speeds[8];               'dimension arrays
DM other[256];
JS #initAry("speeds",0);    'set elements in speeds[] starting at index 0
JS #initAry("other",0);     'set elements in other[] starting at index 0
EN;

#initAry;                   '(array ^a, index ^b) set elements starting at index
^a[^b]=1;
^b=^b+1;                    'increment index
JP #initAry(^b<^a[-1]);
EN,,^a
```

Example: Local Scope

```
#main;                      'label for main program
JS #power(2,2);             'call power subroutine to calculate 2^2
MG _JS;                     'message returned value
JS #power(2,16);            'call power subroutine to calculate 2^2
MG _JS;                     'message returned value
JS #power(2,-8);            'call power subroutine to calculate 2^2
MG _JS;                     'message returned value
EN;

#power;                     '(base ^a,exponent^b) returns base^exponent power
^c=1;
IF (^b=0);                  'if exponent is 0
EN,,1;                      'return 1
ENDIF;
IF (^b<0);                  'if exponent is negative
^d=1;                       'create a variable for later
^b=@ABS[^b];                'find absolute value of exponent
ELSE;                       'if exponent is positive
^d=0;
ENDIF;
#pwrhlpr;                   'label for loop for calculations
^c=^c*^a;                   'multiply through
^b=^b-1;                    'decrement index
JP #pwrhlpr,(^b>0);         'loop while index is above 0
IF (^d=1);                  'if exponent is negative
^c=(1/^c);                  'invert value
ENDIF;
EN,,^c;                     'return value
```

Example: Recursion

```
JS #axsInfo(0);            'jump to axsInfo subroutine with recursion
MG {Z2.0}"Recursed through'message number of stacks
",_JS," stacks";
EN;

#axsInfo;
~h=^a;                     'assign variable axis designator
^b=(^a+$41)*$1000000;      'convert variable axis to ASCII
MG^b{S1}, " Axis: "{N}     'message axis, suppress carriage return
MG{Z8.0}"Position:         'message info
",_TP~h," Error:",_TE~h,"
Torque:",_TT~h{F1.4}
IF (^a=(_BV-1));           'if last axis info has printed
EN,,1;                     'end routine
ENDIF;
JS #axsInfo(^a+1);         'jump back into axsInfo, increment axis
EN,,_JS+1;                 'end routine
```

General Program Flow and Timing information

This section will discuss general programming flow and timing information for Galil programming.

WT vs AT and coding deterministic loops

The main difference between WT and AT is that WT will hold up execution of the next command for the specified time from the execution of the WT command, AT will hold up execution of the next command for the specified time from the last time reference set with the AT command.

```
#main;                     'label for main program
AT 0;                      'set initial AT time reference
t=TIME                     'record initial time
WT 1000;                   'wait 1000 samples
t1=TIME-t;t=TIME;          'find time since initial time, record current time
AT 4000;                   'wait 4000 samples from last time reference
t2=TIME-t;                 'find time since TIME t
MG t,t2;                   'this should output 1000,3000
EN;                        'end program
```

Where the functionality of the operation of the AT command is very useful is when it is required to have a deterministic loop operating on the controller. These instances range from writing PLC-type scan threads to writing custom control algorithms. The key to having a deterministic loop time is to have a trippoint that will wait a specified time independent of the time it took to execute the loop code. In this definition, the AT command is a perfect fit.

Mathematical and Functional Expressions

Mathematical Operators

For manipulation of data, the DMC-41x3 provides the use of the following mathematical operators:

OPERATOR	DESCRIPTION
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo Division
&	Logical And (Bit-wise)
|	Logical Or (Bit-wise)
( )	Parenthesis

Table 7.7: List of mathematical operators

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

```
speed=7.5*v1/2;            'speed equals 7.5 multiplied by v1 and divided by 2
count=count+2;             'count equals current count value plus 2
result=_TPA-(@COS[45]*40);'result equals current A axis position minus 28.28,
                           cosine of 45deg *40 is 28.28
temp=@IN[1]&@IN[2];        'temp equals 1 only if Input 1 and Input 2 are high
```

Mathematical Operation Precision and Range

The controller stores non-integers in a fixed point representation (not floating point). Numbers are stored as 4 bytes of integer and 2 bytes of fraction within the range of ±2,147,483,647.9999. The precision, or the smallest change in value, is 1/65536 or approximately 0.000015.

Using basic mathematics it is known that 1.4*(80,000) = 112,000. However, using a terminal, a DMC controller would calculate the following:

```
:var=1.4*80000
:MG var
111999.5117
:
```

The reason for this error relies in the precision of the controller. 1.4 must be stored to the nearest multiple of 1/65536, which is 91750/65536 = 1.3999. Thus, (91750/65536)*80000 = 111999.5117 and reveals the source of the error.

By ignoring decimals and multiplying by integers first (since they carry no error), and then adding the decimal back in by dividing by a factor of 10 will allow the user to avoid any errors caused by the limitations of precision of the controller. Continuing from the example above:

```
:var=14*80000
:MG var
1120000.0000
:var=var/10
:MG var
112000.0000
:
```

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-41x3 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:

```
#main;                    'main program
len="TESTME";             'set len to a string value
Flen=@FRAC[len];          'set variable to fractional part of string
Flen=Flen*$10000;         'shift Flen by 32 bits, convert fraction to integer
len1=(Flen&$00FF);        'mask first byte of Flen and set new variable
len2=(Flen&$FF00)/$100;   'set variable to top byte of Flen
len3=len&$000000FF;       'set variable to bottom byte of len
len4=(len&$0000FF00)/$100;'set variable to second byte of len
len5=(len&$00FF0000)/     'set variable to third byte of len
$10000;
len6=(len&$FF000000)/     'set variable to fourth byte of len
$1000000;
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;                       'end program
```

```
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};
```

This program will accept a string 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.

Functions

FUNCTION	DESCRIPTION
@SIN[n]	Sine of 'n' degrees
@COS[n]	Cosine of 'n' degrees
@TAN[n]	Tangent of 'n' degrees
@ASIN[n]	Arc Sine of 'n' degrees
@ACOS[n]	Arc Cosine of 'n' degrees
@ATAN[n]	Arc Tangent of 'n' degrees
@COM[n]	Bitwise Complement of 'n'
@ABS[n]	Absolute value of 'n'
@FRAC[n]	Fraction portion of 'n'
@INT[n]	Integer portion of 'n'
@RND[n]	Rounds 'n' up if the fractional part is .5 or greater
@SQR[n]	Square root of 'n'
@IN[n]	Return digital input 'n'
@OUT[n]	Return digital output 'n'
@AN[n]	Return analog input 'n'

Table 7.8: List of labels of automatic subroutines

NOTE: that these functions are multi-valued. An application program may be used to find the correct band.

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

```
v1=@ABS[v7];              'set v1 equal to the absolute value of variable v7
v2=5*@SIN[pos];           'set v2 equal to five * sine of the variable pos
v3=@IN[1];                'set v3 equal to value of Digital Input 1
v4=2*(5+@AN[5]);          'set v4 equal to value of Analog Input 5 plus 5,
                          then multiplied by 2
```

Variables

For applications that require a parameter that is variable, the DMC-41x3 provides 510 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 program calculations. For example, a cut-to-length application may require that a cut length be variable.

```
posa=5000;                  'set posa equal to 5000
PR posa;                    'set posa as argument in PR command
JG rpmB*70;                 'set rpmB*70 as argument in JG command
```

Programmable Variables

The DMC-41x3 allows the user to create up to 510 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-41x3 instructions. For example, PR is not a good choice for a variable name.

NOTE: It is generally a good idea to use lower-case variable names so there is no confusion between Galil commands and variable names.

Valid Variable Names

```
posa
pos1
speedC
```

Invalid Variable Names

RealLongName - Cannot have more than 8 characters
123 - Cannot begin variable name with a number
speed C - Cannot have spaces in the name

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-41x3 function can be used to assign a value to a variable. 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.

```
posA=_TPA;                  'set posA to current A axis motor position
speed=5.75;                 'set speed to 5.75
input=@IN[2];               'set input to state of Digital Input 2
v2=v1+v3*v4;                'set v2 equal to v1 plus v2, then multiplied by v4
var="CAT";                  'set var to CAT string
MG var{S3};                 'message variable as a string
```

Assigning Variable Values to Controller Arguments

Variable values may be used as arguments for controller commands such as PR or SP.

```
PR v1;                    'set v1 as argument in PR command
SP vS*20000;              'set vS*20000 as argument in SP command
```

Displaying the value of variables at the terminal

Variables may be sent to the screen using the format, var=. For example, v1= , returns the value of the variable v1.

Example - Using Variables for Joystick

The example below reads the voltage of an X-Y joystick and assigns it to variables vA and vB to drive the motors at proportional velocities, where:
10 Volts = 3000 rpm = 200000 c/sec
Speed/Analog input = 200000/10 = 20000

```
#joystik;                 'main program label
JG 0,0;                   'set A and B axes in Jog mode
SH AB;                    'enable A and B axes
BG AB;                    'begin motion
AT 0;                     'set AT time reference
#loop;                    'loop label
vA=@AN[1]*20000;          'read joystick X and calculate speed
vB=@AN[2]*20000;          'read joystick Y and calculate speed
JG vA,vB;                 'jog at calculated speeds vA and vB
AT -20;                   'wait 20ms from last time reference, creates a
                          deterministic loop time
JP #loop;                 'jump back to loop label
EN;                       'end program
```

Operands

Operands allow motion or status parameters of the DMC-41x3 to be incorporated into programmable variables and expressions. Most DMC commands have an equivalent operand - which are designated by adding an underscore (_) prior to the DMC-41x3 command. The command reference indicates which commands have an associated operand.

Status commands such as Tell Position (TP) return actual values, whereas action commands such as KP or SP return the values in the DMC-41x3 registers. The axis designation is required following the command.

```
posA=_TPA;                'set posA to current A axis motor position
deriv=_KDC*2;             'set deriv to C axis KD parameter multiplied by 2
JP #loop,(_TEA>5);        'jump to #loop if A axis 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: _KDA=2 is invalid.

Special Operands

The DMC-41x3 provides a few additional operands which give access to internal variables that are not accessible by standard DMC-41x3 commands.

OPERAND	DESCRIPTION
_BGm	Contains the status of motion on the 'm' axis
_BN	Contains the serial number of the controller
_DA	Contains the number of available array names
_DL	Contains the number of available labels
_DM	Contains the number of available array elements
_HMm	Contains the status of the Home Switch for the 'm' axis
_LFm	Contains the status of the Forward Limit Switch for the 'm' axis
_LRm	Contains the status of the Reverse Limit Switch for the 'm' axis
_UL	Contains the number of available variables
TIME	Free-Running Real Time Clock, resets with power-on

Some operands have corresponding commands while others like _LFm, _LRm, and TIME do not. All operands are listed in the Command Reference.

Arrays

For storing and collecting numerical data, the DMC-41x3 provides array space for 24000 elements. The arrays are one dimensional and up to 30 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 analog input values. In the contouring mode, arrays are convenient for holding the points of a position trajectory in a record and playback application.

Defining Arrays

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

```
DM posA[7];              'defines an array named posA with 7 elements
DM speed[100];           'defines an array named speed with 100 elements
DA posA[];               'deallocates the posA array
```

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.
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.

```
DM speed[10];            'dimension speed Array
speed[0]=7650.2;         'set 1st element of the array to 7650.2
speed[0]=?;              'return array element value
posA[9]=_TPA;            'set 10th element of the array 'posA' to the current
                         A axis motor position
con[1]=@COS[pos]*2;      'set 2nd element of the array 'con' using the 'pos'
                         variable
timer[0]=TIME;           'set 1st element of the array 'timer' to the current
                         TIME value
```

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.

```
#main;                   'program labe
count=0;                 'initialize counter variable
DM pos[10];              'dimension array
#loop;                   'loop label
WT 10;                   'wait 10 msec
pos[count]=_TPA;         'record current A axis motor position
pos[count]=?;            'report position
count=count+1;           'increment counter
JP #loop,(count<10);     'jump to #loop while count is less than 10
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

The GDK software is recommended for downloading and uploading array data from the controller. The gclib programming library also provides function calls for downloading and uploading array data from the controller to/from a buffer or a file. Arrays may uploaded and downloaded when using non Galil software or programming library by using the QU and QD commands.

Automatic Data Capture into Arrays

The DMC-41x3 provides a special feature for automatic capture of data such as position, position error, inputs or torque. This is useful for teaching motion trajectories or observing system performance. Up to eight types of data can be captured and stored in eight arrays. The capture rate or time interval may be specified. Recording can done as a one time event or as a circular continuous recording.

Command Summary - Automatic Data Capture

COMMAND	DESCRIPTION
RA	Selects predefined arrays for data capture
RD	Selects the type of data to be recorded
RC	Determines the recording rate and begins recording

Data Types for Recording:

DATA TYPE	DESCRIPTION
TIME	Controller sample time
_AFm	Analog Input digital value corresponding to 'm' axis
_DEm	Auxiliary encoder position of the'm' axis
_OP	Digital output states as bitmask
_RLm	Latched position of the 'm' axis
_RPm	Commanded position of the 'm' axis
_SCm	Stop code of the 'm' axis
_TDm	Auxiliary encoder position of the 'm' axis
_TEm	Position error of the 'm' axis
_TI	Digital input states as bitmask
_TPm	Motor encoder position of 'm' axis
_TSm	Status of switches of the 'm' axis
_TTm	Commanded torque of the 'm' axis

Operand Summary - Automatic Data Capture

OPERAND	DESCRIPTION
_RC	Contains recording status
_RD	Contains the address of the next element for recording

Example - Recording into An Array

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

```
#record;                                    'label for program
DM posA[300],posB[300];                     'define A,B position arrays
DM erA[300],erB[300];                       'define A,B error arrays
RA posA[],erA[],posB[],erB[];               'select arrays for capture
RD _TPA,_TEA,_TPB,_TEB;                     'select data types
PR 10000,20000;                             'specify a relative position move
SH AB;                                      'enable A and B axes
RC 1;                                       'start recording at rate of 2 msec
BG AB;                                      'begin motion
#loop;                                      'loop label
WT 10;
JP #loop,(_RC=1);                           'Jump to #loop while recording is in progress
AM AB; MO AB;                               'wait for motion to finish, disable A and B axes
MG "Recording finished";                    'print message
i=0;                                        'intialize counter
#message;                                   'label to loop for messaging
MG “Motor position: “ ,posA[i],posB[i];'send message of recorded motor positions
MG “Position Error: “ ,erA[i],erB[i];   'send message of recorded position errors
i=i+1;                                      'increment counter
JP #message,(i<300);                        'loop while counting through elements in arrays
EN;                                         'end program
```

De-allocating Array Space

Array space may be de-allocated using the DA command followed by the array name.

Input of Data (Numeric and String)

Sending Data from a Host

The DMC-41x3 can accept ASCII strings from a host. This is the most common way to send data to the controller such as setting variables to numbers or strings. Any variable can be stored in a string format up to 6 characters by simply specifying defining that variable to the string value with quotes. To assign a variable a numerical value, the direct number is used.
All variables on the DMC-41x3 controller are stored with 4 bytes of integer and 2 bytes of fractional data.

Operator Data Entry Mode

The Operator Data Entry Mode provides for un-buffered data entry through the auxiliary RS-232 port. In this mode, the DMC-41x3 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 Port 2 only. This mode may be exited with the \ or <escape> key. See the CC command for more details.

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

To capture and decode characters in the Operator Data Mode, the DMC-41x3 provides special the following keywords:

Keyword	DESCRIPTION
P2CH	Contains the last character received
P2ST	Contains the received string
P2NM	Contains the received number
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 a number

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

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

Example – Recording into An Array

```
#AUTO;                    'automatic subroutine label
JP #loop,P2CD<>3;         'check to see if status code is 3 (number received)
JP #p,P2CH=”v";           'check if last character received was a V
PR P2NM;                  'assign received number to position
JS #xaxis,P2ST=”x";       'check to see if received string is x
```

Using Communication Interrupt

The DMC-41x3 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 #COMINT label is used for the communication interrupt. For example, the DMC-41x3 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 – Communication Interrupts

A DMC-41x3 is used to jog the A and B axis. This program automatically begins upon power-up and allows the user to input values from the main USB port terminal. The speed of either axis may be changed during motion by specifying the axis letter followed by the new speed value. An S stops motion on both axes.

```
#AUTO;                    'label to automatically execute
speedA=10000;             'initial A axis speed
speedB=10000;             'initial B axis speed
CI 2;                     'set Port 2 for Character Interrupt
JG speedA, speedB;        'specify jog mode speed for A and B axes
SH AB;                    'enable A and B axes
BG AB;                    'begin motion
#print;                   'label for routine to print to terminal;
MG{P2}"TO CHANGE SPEEDS";
MG{P2}"TYPE A OR B";
MG{P2}"TYPE S TO STOP";
#jogloop;                 'label for loop to change jog speeds
JG speedA, speedB;        'set new jog speed
WT 50;
JP #jogloop;              'jump to loop label
EN;                       'end of main program
#COMINT;                  'interrupt routine
JP #a,(P2CH="A");         'check for A
JP #b,(P2CH="B");         'check for B
JP #s,(P2CH="S");         'check for S
ZS 1;CI 2;JP #jogloop;    'jump back if not A,B, or S
#a;JS #num;
speedA=val;               'new A axis speed
ZS 1;CI 2;JP#print;       'jump to print
#b;JS#num;
speedB=val;               'new B axis speed
ZS 1;CI 2;JP#print;       'jump to print
#s;ST AB;AM AB;CI-1;      'stop motion
MG{^8}, "THE END";
ZS;EN,1;                  'end routine, re-enable interrupt
#num;                     'routine for entering new jog speed
MG "ENTER",P2CH{S},"AXIS  'prompt for value
SPEED" {N};
#numloop; CI-1;           'check for enter
#nmlp;                    'routine to check input from terminal
JP #nmlp,(P2CD<2);
JP #error,(P2CD=2);       'jump to error if string
val=P2NM;
EN;                       'end subroutine
#error;CI-1;              'error Routine
MG "INVALID, TRY AGAIN";  'error message
JP #nmlp;                 'jump to loop for new value
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.

The DMC-41x3, stores all variables as 6 bytes of information. When a variable is specified as a number, the value of the variable is represented as 4 bytes of integer and 2 bytes of fraction. When a variable is specified as a string, the variable can hold up to 6 characters (each ASCII character is 1 byte). The characters can be individually separated by using bit-wise operations, see section Bit-wise Operators.

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.

```
MG "Final Value = ",      'message containing result variable
result;
```

In addition to variables, functions and operands can be used in the message command.

```
MG "Analog input is ",    'message containing value of Analog Input 1
@AN[1];
MG "A axis position is ", 'message containing A axis motor position
_TPA;
```

Specifying the Port for Messages:

The port can be specified with the specifier, {P1} for the main USB port or {En} for the Ethernet port.

```
MG {P1} "Hello World";    'message sent through the USB port
```

Formatting Messages

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

```
MG str {S3};              'message with first 3 characters of str variable
```

Numeric data may be formatted using the {Fm.n} expression following the completed MG statement. {$m.n} 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.

```
MG "Final Value = ", result {F5.2}
```

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

Final Value = 00004.10

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

Final Value = 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.

```
#main;                      'label for main program
SH A;                       'enable A axis
JG 50000;                   'set jog speed to 50000 cts/s on A axis
BG A;                       'begin motion
AS A;                       'wait until A axis is at target speed
MG "The Speed is ",_TVA     'send message with A axis speed
{F5.0}{N}
MG"counts/sec";
EN;                         'end program
```

Using the MG Command to Configure Terminals

The MG command can 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.

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

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

Summary of Message Formatters

FORMATTER	DESCRIPTION
" "	Surrounds text string
{Fm.n}	Formats numeric values in decimal n digits to the left of the decimal point and m digits to the right
{P1} or {E}	Send message to Main USB Port or Ethernet Port
{$m.n}	Formats numeric values in hexadecimal
{^n}	Sends ASCII character specified by integer n
{N}	Suppresses carriage return/line feed
{Sn}	Sends the first n characters of a string variable, where n is 1 thru 6

Table 7.9: List of different formatters to display information with messages

Displaying Variables and Arrays

Variables and arrays may be sent to the screen using the format, variable= or array[x]=?.

Example - Printing a Variable and an Array element

```
#main;                      'label for main program
DM posA[7];                 'define posA array with 7 elements
SH A;                       'enable A axis
PR 1000;                    'define position relative move
BG A;                       'begin motion
AM A;                       'wait for motion to finish
MO A;                       'disable A axis
v1=_TPA;                    'set variable to A axis motor position
posA[1]=_RPA;               'assign array element to A axis reference position
v1=?;                       'print v1
posA[1]=;                   'print posA[1]
EN;                         'end program
```

Interrogation Commands

The DMC-41x3 has a set of commands that directly interrogate the controller. When these command 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 Command Basics.

Using the PF Command to Format Response from Interrogation Commands

The PF command can change format of the values returned by interrogation commands. 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. If the number of decimal places specified by PF is less than the actual value, a nine appears in all the decimal places.
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 PF10.

```
:LZ 0                    'disable suppressing leading zeroes
:PF 10                   'set format to 10 places
:DP 21                   'define position
:TPA                     'tell position
0000000021
:PF4                     'change format to 4 places
:TPA                     'tell position
0021
:PF-4                    'change to hexadecimal format
:TPA                     'tell position
$0015
:PF2                     'change format to 2 places
:DP 121                  'define new motor position
:TPA                     'reports 99 since value is more than 2 places
99
```

Adding Leading Zeros from Response to Interrogation Commands

The leading zeros on data returned as a response to interrogation commands can be added by the use of the command, LZ. The LZ command is set to a default of **1**.

```
:LZ0                     'disables the LZ function
:TP                      'tell position
-0000000009, 0000000005
:LZ1                     'enables LZ
:TP                      'tell position
-9, 5
```

Local Formatting of Response of Interrogation Commands

The response of interrogation commands may be formatted locally. To format locally, use the command, {Fm.n} or {$m.n} on the same line as the interrogation command. The symbol F specifies that the response should be returned in decimal format and $ specifies hexadecimal. m is the number of digits to the left of the decimal, and n is the number of digits to the right of the decimal.

```
:TP {F2.2}               'tell position in decimal format 2.2
-05.00, 05.00, 00.00,
07.00
:TP {$4.2}               'tell position in hexadecimal format 4.2
FFFB.00,$0005.00,$0000.00,
$0007.00
```

Formatting Variables and Array Elements

The VF command is used to format variables and array elements. A negative sign can be used to specify hexadecimal format. The default format for VF is VF10.4. Hex values that are returned are preceded by a $ and in 2's complement.

```
:v1=10                    'assign v1 variable
:v1=?                     'return v1
10.0000
:VF2.2                    'change format
:v1=?                     'return v1
10.00
:VF-2.2                   'specify hex format
:v1=?                     'return v1
$0A.00
:VF1                      'change format
:v1=?                     'return v1
9
```

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.

```
:VF 10.4                  'set default variable format
:v1=10                    'assign v1 variable
:v1=                      'return v1
10.0000
:v1={F4.2}                'specify local format
0010.00
:v1={$4.2}                'specify hex format
$000A.00
: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-41x3 position parameters such as PR, PA, and VP have units of quadrature counts. Speed parameters such as SP, JG, and VS have units of counts/sec. Acceleration parameters such as AC, DC, VA, and VD have units of counts/sec^2. 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.

```
#run;                     'main program label
cts=2000;                 'counts per rev conversion factor
MG “Enter number of       'prompt for number of revs
revolutions”;
MG “rev=?”
rev=0;                    'initialize rev variable
#rev;                     'label for loop
WT 20
JP #rev,(rev=0);          'jump back to rev label if variable is unchanged
PR rev*cts;               'define relative position move in counts
MG "Enter speed in RPM";  'prompt for speed
```

```
MG “rpm=?”
rpm=0;                      'initialize rpm variable
#spd;                       'label for loop
WT 20
JP #spd,(rpm=0);            'jump back to spd label if variable is unchanged
SP rpm*cts/60;              'convert to counts/sec and use as speed
MG "Enter acceleration in   'prompt for acceleration
Rad/sec2";
MG “acc=?”
acc=0;                      'initialize acc variable
#acc;                       'label for loop
WT 20
JP #acc,(acc=0);            'jump back to acc label if variable is unchanged
AC acc*cts/(2*3.14);        'convert to counts/sec²
SH A;                       'enable A axis
BG A;                       'begin motion
AM A;                       'wait for motion to finish
MO A;                       'disable A axis
EN;                         'end program
```

Hardware I/O

Digital Outputs

The DMC-41x3 has an 8-bit uncommitted output port, the DMC-4153 through DMC-4183 has an additional 8 outputs. Each bit on the output port may be set and cleared with the software instructions SB (Set Bit) and CB (Clear Bit), or OB (define output bit).

```
SB 6;                       'set Output 6
CB 4;                       'clear Output 4
```

The OB command 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. Any zero results in a clear bit.

```
OB 1,(pos);                 'set Output 1 if the variable POS is non-zero
OB 2,(@IN[1]);              'set Output 2 if @IN[1]=1
OB 3,(@IN[1]&@IN[2]);       'set Output 3 if @IN[1] and @IN[2] equal 1
OB 4,(count[1]);            'set Output 4 if count array element 1 is not zero
```

The outputs can be set by specifying an 16-bit word using the OP command. This command allows a single command to define the state of the entire 16-bit output port, where bit 0 is Output 1, bit 1 is Output 2 and so on. A 1 designates that the output is on.

```
OP 6;                       'sets outputs 2 and 3 of output port to high, all
                            other bits are 0 (2^1 + 2^2 = 6)
OP 0;                       'clears all bits of output port
OP 255;                     'sets all bits of output port
                            (2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7)
```

Example - Turn on output after move

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

```
#main;                    'main program label
PR 2000;                  'specify position relative command
SH A;                     'enable A axis
BG A;                     'begin motion
AM A;                     'after motion is finished
SB 1;                     'set Output 1
WT 1000;                  'wait 1000 msec
CB 1;                     'clear Output 1
MO A;
EN;                       'end program
```

Digital Inputs

The general digital inputs for are accessed by using the @IN[n] function or the TI command. The @IN[n] function returns the logic level of the specified input, n, where n is a number 1 through 16.

Example - Using Inputs to control program flow

```
JP #main,(@IN[1]=0);      'jump to main label if input 1 is active
JP #input,(@IN[2]=1);     'jump to input label if input 2 is inactive
AI 7;                     'wait until input 7 is inactive
AI -6;                    'wait until input 6 is active
```

Example - Start Motion on Switch

Motor A must turn at 4000 counts/sec when the user flips a panel switch to on. When panel switch is turned to off position, motor A must stop turning. The switch is tied to Input 1 and the input is active when the switch is turned on.

```
#main;                    'main program label
SH A;                     'enable A axis
JG 4000;                  'define jog speed
#speed;                   'label for loop
AI -1;                    'wait for switch to be on, Input 1 active, @IN[1]=0
BG A;                     'begin motion
AI 1;                     'wait for switch to be off, Input 1 inactive,
                          @IN[1]=1
ST A;                     'stop motion on A axis
AM A;                     'wait for motion to finish
JP #speed;                'jump back to speed label to repeat
EN;                       'end program
```

The Auxiliary Encoder Inputs

The auxiliary encoder inputs can be used for general use. For each axis, the controller has one auxiliary encoder and each auxiliary encoder consists of two inputs, channel A and channel B. The auxiliary encoder inputs are mapped to the inputs 81-96.

Each input from the auxiliary encoder is a differential line receiver and can accept voltage levels between ±12 volts. The inputs have been configured to accept TTL level signals. To connect TTL signals, simply connect the signal to the + input and leave the - input disconnected. For other signal levels, the - input should be connected to a voltage that is ½ of the full voltage range (for example, connect the - input to 5 volts if the signal is a 0 - 12 volt logic).

A DMC-4113 has one auxiliary encoder. This encoder has two inputs (channel A and channel B). Channel A input is mapped to input 81 and Channel B input is mapped to input 82. To use this input for 2 TTL signals, the first signal will be connected to AA+ and the second to AB+. AA- and AB- will be left unconnected. To access this input, use the function @IN[81] and @IN[82].

NOTE: The auxiliary encoder inputs are not available for any axis that is configured for stepper motor.

Input Interrupt Function

The DMC-41x3 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 command.

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.

NOTE: Use the RI command, not EN, to return from the #ININT subroutine.

Example - Input Interrupt

```
#main;                    'main program label
II 1;                     'enable Input 1 for interrupt function
JG 30000,-20000;          'set jog speeds on A and B axes
SH AB;                    'enable A and B axes
BG AB;                    'begin motion
#loop;                    'label for loop
TP AB;                    'report A and B axes motor positions
WT 1000;                  'wait 1000 milliseconds
JP #loop;                 'jump back to loop label
EN;                       'end program
#ININT;                   'Input Interrupt subroutine
MG "Interrupt occurred";  'displays the message
ST AB;                    'stop motion on A and B axes
AM AB;                    'wait for motion to finish
#loopInt;                 'label for loop
WT 20
JP #loopInt,(@IN[1]=0);   'jump up to loopInt label while input is active
JG 15000,10000;           'specify new speeds
WT 300;                   'wait 300 milliseconds
BG AB;                    'begin motion on A and B axes
RI;                       'return from Interrupt subroutine
```

Jumping back to main program with #ININT

To jump back to the main program using the JP command, the RI command must be issued in a subroutine and then the ZS command must be issued prior to the JP command. See Application Note # 2418 for more information.

https://galil.com/download/application-note/note2418.pdf

Analog Inputs

The DMC-41x3 provides eight analog inputs. See the Integrated Components section for more details. The value of these inputs in volts may be read using the @AN[n] function where n is the analog input 1 through 8. The resolution of the Analog-to-Digital conversion is 12 bits (16-bit ADC is available as an option). Analog inputs are useful for reading special sensors such as temperature, tension or pressure.

The following examples show programs which cause the motor to follow an analog signal. The first example is a point-to-point move. The second example shows a continuous move.

Example - Position Follower (Point-to-Point)

The motor must follow an analog signal. When the analog signal varies by 10V, motor must move 10000 counts. Read the analog input and command A to move to that point.

```
#points;                    'label for main program
SP 7000;                    'define speed
AC 80000;                   'define acceleration
DC 80000;                   'define deceleration
SH A;                       'enable A axis
#loop;                      'label for loop
vp=@AN[1]*1000;             'read analog input, compute position and store
PA vp;                      'define absolute position move
BG A;                       'start motion
AM A;                       'wait for motion to finish
JP #loop;                   'jump back to loop label
EN;                         'end program
```

Example - Position Follower (Continuous Move)

Read the analog input, compute the commanded position and the position error. Command the motor to run at a speed in proportions to the position error.

```
#cont;                      'label for main program
AC 80000;                   'define acceleration
DC 80000;                   'define deceleration
SH A;                       'enable A axis
JG 0;                       'define jog speed of 0 cts/s
BG A;                       'begin motion, motion is commanded to speed of 0
#loop;                      'label for loop
vp=@AN[1]*1000;             'compute desired position
ve=vp-_TPA;                 'calculate error
vel=ve*20;                  'compute velocity
JG vel;                     'set new speed
JP #loop;                   'jump back to loop label
EN;                         'end program
```

Example – Low Pass Digital Filter for the Analog inputs

Because the analog inputs on the Galil controller can be used to close a position loop, they have a very high bandwidth and will therefor read noise that comes in on the analog input. Often when an analog input is used in a motion control system, but not for closed loop control, the higher bandwidth is not required. In this case a simple digital filter may be applied to the analog input, and the output of the filter can be used for in the motion control application. This example shows how to apply a simple single pole low-pass digital filter to an analog input. This code is commonly run in a separate thread (XQ#filt,1 – example of executing in thread 1).

```
#filt;                      'label for main program
an1=@AN[1];                 'set initial value
k1=32/64;                   'increase k1 for less filtering
k2=32/64;                   'increase k2 for more filtering
AT 0;                       'set initial time reference
#loop;                      'label for loop
an1=(k1*@AN[1])+(k2*an1);   'calculate filtered input
AT -2,1;                    'wait 2 samples from last reference
JP #loop;                   'jump back to loop label
EN;                         'end program
```

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:

4000/2π = 637 count/inch

This implies that a distance of 10 inches equals 6370 counts, and a slew speed of 5 inches per second, for example, equals 3185 count/sec.

The input signal may be applied to Input 1, 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 Figure 7.1.

The program starts at a state that defined as #main. Here the controller waits for the input pulse on Input 1. 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 for a new cycle.

```
#main;                  'main program label
PR 6370;                'define relative position move
SP 3185;                'define speed
SH A;                   'enable A axis
#loop;                  'label for loop
AI -1;                  'wait until Input 1 is active
BG A;                   'begin motion
AM A;                   'wait for motion to finish
SB 1;                   'set Output 1
WT 20;                  'wait 20 ms
CB 1;                   'clear Output 1
WT 80;                  'wait 80 ms
JP #loop;               'jump to loop label
EN;                     'end program
```

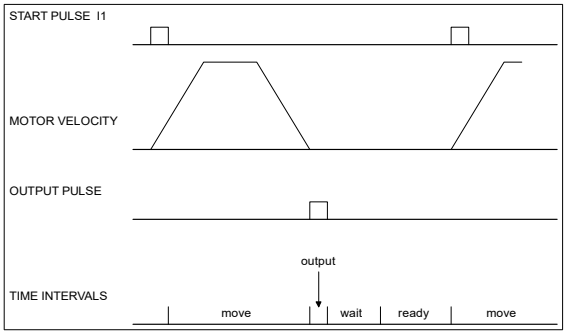


Figure 7.1: Motor Velocity and the Associated Input/Output signals

X-Y Table Controller

An X-Y-Z system must cut the pattern shown in Figure 7.2. The X-Y table moves the plate while the Z motor raises and lowers the cutting tool.

The solid curves in Figure 7.2 indicate sections where cutting takes place. Those must be performed at a feed rate of 1 inch per second. The dashed line corresponds to non-cutting moves and should be performed at 5 inch per second. The acceleration rate is 0.1 g.

The motion starts at point A, with the Z motor raised. An X-Y motion to point B is followed by lowering the Z motor and performing a cut along the circle. Once the circular motion is completed, the Z motor is raised and the motion continues to point C, etc.

Assume that all of the 3 axes are driven by lead screws with 10 turns-per-inch pitch. Also assume encoder resolution of 1000 lines per revolution. This results in the relationship:

1 inch = 40,000 counts

and the speeds of

1 in/sec = 40,000 count/sec

5 in/sec = 200,000 count/sec

an acceleration rate of 0.1g equals

0.1g = 38.6 in/s2 = 1,544,000 count/s^2

Note that the circular path has a radius of 2" or 80000 counts, and the motion starts at the angle of 270° and traverses 360° in the CW (negative direction).

Further assume that the Z must move 2" at a linear speed of 2" per second. The X-Y motion is tied to A and B axes while the Z motor is connected to C axis.

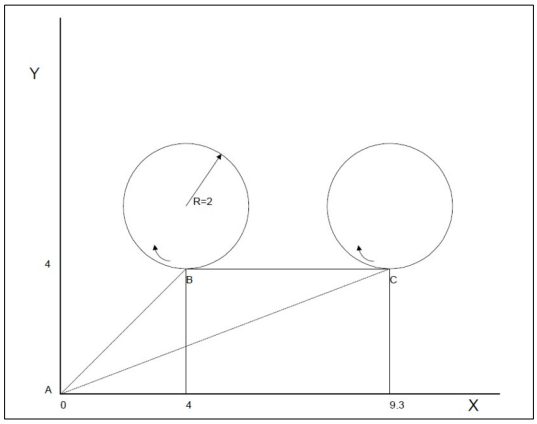


Figure 7.2: Motor Velocity and the Associated Input/Output signals

Example – Vector Mode

```
#main;                  'main program label
SH ABC;                 'enable A, B, and C axes
VM AB;                  'select A and B axes for Vector Mode
VP 160000,160000;       'define vector position
VE;                     'end vector sequence
VS 200000;              'set vector speed
VA 1544000;             'set vector acceleration
BG S;                   'begin vector motion
AM S;                   'wait for motion to finish
PRC=-80000;             'define move for C axis to move down
SPC=80000;              'set speed for C axis
BG C;                   'begin motion
AM C;                   'wait for motion to finish
CR 80000,270,-360;      'define a circle in vector mode
VE;                     'end vector sequence
VS 40000;               'set vector speed
BG S;                   'begin vector motion
AM S;                   'wait for motion to finish
PRC=80000;              'define move for C axis to move up
BG C;                   'begin motion
AM C;                   'wait for motion to finish
PRA=-21600;             'define relative position move for A axis
SPA=20000;              'set speed for A axis
BG A;                   'begin motion
AM A;                   'wait for motion to finish
PRC=-80000;             'define move for C axis to move down
BG C;                   'begin motion
AM C;                   'wait for motion to finish
CR 80000,270,-360;      'define a circle in vector mode
VE;                     'end vector sequence
VS 40000;               'set vector speed
BG S;                   'begin vector motion
AM S;                   'wait for motion to finish
PRC=80000;              'define move for C axis to move up
BG C;                   'begin motion
AM C;                   'wait for motion to finish
VP -37600,-16000;       'move to return A and B axes to initial position
VE;                     'end vector sequence
VS 200000;              'set vector speed
BG S;                   'begin vector motion
AM S;                   'wait for motion to finish
MO ABC;                 'disable A, B, and C axes
EN;                     'end program
```

Speed Control by Joystick

The speed of a motor is controlled by a joystick. The joystick produces a signal in the range between -10V and +10V. The objective is to drive the motor at a speed proportional to the input voltage. Assume that a full voltage of 10 Volts must produce a motor speed of 3000 rpm with an encoder resolution of 1000 lines or 4000 count/rev. This speed equals:

3000 rpm = 50 rev/sec = 200000 counts/sec

To equate 10V to 200000 counts/sec, the program will use the following equation:

Speed = 20000 x @AN[n]

To ensure the speed is always proportional to the analog input, looping DMC code will be used:

```
#main;                    'label for main program
JG 0;                     'set starting jog speed
SH A;                     'enable A axis
BG A;                     'begin motion
#loop;                    'label for loop
vel=@AN[1]*20000;         'read Analog Input 1 and calculate new speed
JG vel;                   'set new jog speed
WT 20;
JP #loop;                 'jump back to loop label
EN;                       'end program
```

Position Control by Joystick

This system requires the position of the motor to be proportional to the joystick angle. Furthermore, the ratio between the two positions must be programmable. For example, if the control ratio is 5:1, it implies that when the joystick voltage is 5 Volts, corresponding to 1028 counts, the required motor position must be 5120 counts. The variable changes the position ratio.

```
#program;                 'label for main program
ratio=5;                  'define control ratio
DP 0;                     'define the starting position
JG 0;                     'set starting jog speed
SH A;                     'enable A axis
BG A;                     'begin motion
#loop;                    'label for loop
vin=@AN[1];               'read and store Analog Input 1
target=vin*ratio;         'compute desired position
error=target-_TPA-_TEA;   'compute the following error
speed=error*20;           'compute a proportional speed
JG speed;                 'set new jog speed
WT 50;                    'wait
JP #loop;                 'jump back to loop label
EN;                       'end program
```

Backlash Compensation by Sampled Dual-Loop

The continuous dual loop, enabled by the DV command is an effective way to compensate for backlash. In some cases, however, when the backlash magnitude is large, it may be difficult to stabilize the system. In those cases, it may be easier to use the sampled dual loop method described below.

This design example addresses the basic problems of backlash in motion control systems. The objective is to control the position of a linear slide precisely. The slide is to be controlled by a rotary motor, which is coupled to the slide by a 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 a rotary sensor is used, there will be a 4 micron backlash error. On the other hand, if a linear encoder is used, the backlash in the feedback loop will cause oscillations due to instability.

An alternative approach is the dual-loop, where two sensors are used, 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 A-axis, and that the linear sensor is read and stored in a variable. Further assume that at the start, both the position of A 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 A motor to move to the rotary position of 1000. Once it arrives, the position of the load is checked. If, for example, the load position is 980 counts, it implies that a correction of 20 counts must be made. However, when the A-axis is commanded to be at the position of 1000, suppose that the actual position is only 995, implying that A 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 A-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.

```
#main;                'label for main program
DP 0;                 'define starting position
linpos=0;             'initialize variable for linear position
PR 1000               'define relative position move
SH A;                 'enable A axis
BG A;                 'begin motion
#loop;                'label for loop
AM A;                 'wait for motion to finish
WT 50;                'wait 50 msec
linpos=_DEA;          'read linear position from Auxiliary encoder
er=1000-linpos-_TEA;  'calculate correction
JP #end,(@ABS[er]<2); 'jump to end label if error is within acceptable margin
PR er;                'command correction move
BG A;                 'begin motion
JP #loop;             'jump to loop label to repeat the process
#end;                 'label to end program
EN;                   'end program
```

Using the DMC Editor to Enter Programs (Advanced)

The GDK software package provides an editor and utilities that allow the upload and download of DMC programs to the motion controller. In most instances the user will use Galil software or a host application to download programs to the Galil controller rather than using the ED command.

Chapter 8 Hardware & Software Protection

Introduction

The DMC-41x3 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-41x3 is an integral part of the machine, the engineer should design his overall system with protection against a possible component failure on the DMC-41x3. Galil shall not be liable or responsible for any incidental orf consequential damages.

Hardware Protection

The DMC-41x3 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 (OE**1**) and the abort command is given. Each axis amplifier has separate amplifier enable lines. This signal also goes low when the watch-dog timer is activated, or upon reset.

NOTE: The standard configuration of the AEN signal is high amplifier enable (HAEN). Both the polarity and the amplitude can be changed. To make these changes, see section entitled Amplifier Enable in Chapter 3.

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 ERR. When the error signal is low, this indicates an error condition and the Error Light on the controller will be illuminated. For details on the reasons why the error output would be active see Error Light (Red LED) in Chapter 9.

Input Protection Lines

General Abort

A low input stops commanded motion instantly without a controlled deceleration. For any axis in which the Off-On-Error function is enabled, the amplifiers will be disabled. This could cause the motor to 'coast' to a stop. If the Off-On-Error function is not enabled, the motor will instantaneously stop and servo at the current position. The Off-On-Error function is further discussed in this chapter.

The Abort input by default will also halt program execution; this can be changed by changing the 5th field of the CN command. See the CN command in the command reference for more information.

Selective Abort

The controller can be configured to provide an individual abort for each axis. Activation of the selective abort signal will act the same as the Abort Input but only on the specific axis. To configure the Digital Inputs for selective abort, use the CN command. See the CN command in the command reference for more information.

ELO (Electronic Lock Out)

Used in conjunction with Galil amplifiers, this input allows the user the shutdown the amplifier at a hardware level. For more detailed information on how each Galil amplifier behaves when the ELO is triggered, see Integrated Components in the Appendices.

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. The OE command can also be configured so that the axis will be disabled upon the activation of a limit switch.

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. The OE command can also be configured so that the axis will be disabled upon the activation of a limit switch.

Software Protection

The DMC-41x3 provides a programmable error limit as well as encoder failure detection. It is recommended that both the position error and encoder failure detection be used when running servo motors with the DMC-41x3. Along with position error and encoder failure detection, then DMC-41x3 has the ability to have programmable software limit.

Position Error

The error limit can be set for each axis using the ER command. 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 controller 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 OE**1** or OE**3**
AEN Output Line	Switches to Motor Off state

The position error can be monitored during execution using the TE command.

Encoder Failure detection

The encoder failure detection on the controller operates based upon two factors that are user settable, a threshold of motor command output (OV), a time above that threshold (OT) in which there is no more than 4 counts of change on the encoder input for that axis. The encoder failure detection is activated with the OA command. When an encoder failure is detected and OA is set to **1** for that axis, the same conditions will occur as a position error. For more information, refer to the command reference for the DMC-41x3.

Using Encoder Failure to detect a hard stop or stalled motor

The encoder failure detection can also be used to detect when an axis is up against a hard stop. In this scenario the motor command will be commanded above the OV threshold, but because the motor is not moving the controller will detect this scenario as an encoder failure.

Programmable Position Limits

The DMC-41x3 provides programmable forward and reverse position limits. These are set by the BL and FL commands. Once a position limit is specified, the DMC-41x3 will not accept position commands beyond the limit. Motion beyond the limit is also prevented.

```
#main;                    'program label
DP 0,0,0;                 'define position
BL -2000,-4000,-8000;     'set reverse position limit
FL 2000,4000,8000;        'set forward position limit
JG 2000,2000,2000;        'set jog speeds on A, B, and C axes
SH ABC;                   'enable A, B, and C axes
BG ABC;                   'begin motion, all axes will stop after 3 seconds
EN;                       'end program
```

Off-On-Error

The DMC-41x3 controller has a built in function which can turn off the motors under certain error conditions. This function is known as Off-On-Error and is enabled with the OE command. When this function is enabled, the specified motor will be disabled under the following conditions depending on the parameter used in the OE command:

1. The position error for the specified axis exceeds the limit set with the ER command
2. A hardware limit is reached
3. The abort command is given
4. 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 Servo Here (SH) command once it is deemed safe to do so.

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, a encoder failure is detected, or the Abort input is triggered. 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 cleared.

```
#main;                    'label for main dummy program
ERA=1000;                 'set error limit for A axis
#loop                     'label for loop
WT 20;
JP #loop;EN;              'jump back to loop label

#POSERR;                  'start of position error routine
MG "A Axis Error";        'send message
SB 1;                     'set Output 1
ST A;                     'stop motion
AM A;                     'wait for motion to finish
SH A;                     'servo motor to clear error
RE;                       'return to main program
```

Limit Switch Routine

The DMC-41x3 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 evaluated with the **JP** command. The **_LR**m operand contains the state of the reverse limit and **_LF**m operand contains the state of the forward limit. The CN command can be used to configure the polarity of the limit switches.

```
#main;                    'label for main program
WT 20;
JP #main; EN;             'jump back to main label

#LIMSWI;                  'start of limit switch routine
ST A; AM A;               'stop motion, wait for motion to finish
JP #lf,(_LFA=0);          'jump to #LF if forward limit
JP #lr,(_LRA=0);          'jump to #LR if reverse limit
#lf;                      'forward limit label
MG "A Axis Forward Limit";'send message
PR -1000;BG A;AM A;       'move off limit in reverse direction
JP #end;                  'jump to end label
#lr;                      'reverse limit label
MG "A Axis Reverse Limit";'send message
PR 1000;BG A;AM A;        'move off limit in forward direction
#end;                     'end label
RE;                       'return to main program
```

Chapter 9 Troubleshooting

Overview

Potential problems have been divided into groups as follows:

1. Installation
2. Stability and Compensation
3. Operation
4. Error Light (Red LED)

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 a different Amplifier Enable setting on the Interconnect Module	Refer to Chapter 3 or contact Galil.
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 (MA+ and MB+ only) do not make any connections to the MA- and MB- inputs. Check CE command Contact Galil
Encoder Position Drifts	Swapping cables fixes the problem	1. Poor Connections / intermittent cable	Review all connections and connector contacts.
Encoder Position Drifts	Significant noise can be seen on MA+ and / or MB+ encoder signals	1. Noise	Shield encoder cables Avoid placing power cables near encoder cables Avoid Ground Loops Use differential encoders Use ±12V encoders

Stability

SYMPTOM	DIAGNOSIS	CAUSE	REMEDY
Servo motor runs away when the loop is closed.	Reversed Motor Type corrects situation (MT -1)	Wrong feedback polarity.	Reverse Motor or Encoder Wiring (remember to set Motor Type back to default value: MT 1)
Motor oscillates.		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.	Anything	Correct problem reported by TC1
Motor Doesn't Move	Response of controller from TC1 diagnoses error.	Anything	Correct problem reported by SC

Error Light (Red LED)

The red error LED has multiple meanings for Galil controllers. Here is a list of reasons the error light will come on and possible solutions:

Under Voltage

If the controller is not receiving enough voltage to power up.

Under Current

If the power supply does not have enough current, the red LED will cycle on and off along with the green power LED.

Position Error

If any axis that is set up as a servo (MT command) has a position error value (TE) that exceeds the error limit (ER) - the error light will come on to signify there is an axis that has exceeded the position error limit. Use a DP*=0 to set all encoder positions to zero or a SH (Servo Here) command to eliminate position error.

Invalid Firmware

If the controller is interrupted during a firmware update or an incorrect version of firmware is installed - the error light will come on. The prompt will show up as a greater than sign “>” instead of the standard colon “:” prompt. Use GDK software to install the correct version of firmware to fix this problem.

Self Test

During the first few seconds of power up, it is normal for the red LED to turn on while it is performing a self test. If the self test detects a problem such as corrupted memory or damaged hardware - the error light will stay on to signal a problem with the board. To fix this problem, a Master Reset may be required. The Master Reset will set the controller back to factory default conditions so it is recommended that all motor and I/O cables be removed for safety while performing the Master Reset. Cables can be plugged back in after the correct settings have been loaded back to the controller (when necessary). To perform a Master Reset - find the jumper location labeled MR or MRST on the controller and put a jumper across the two pins. Power up with the jumper installed. The Self-Test will take slightly longer - up to 5seconds. After the error light shuts off, it is safe to power down and remove the Master Reset jumper. If performing a Master Reset does not get rid of the error light, the controller may need to be sent back to the factory to be repaired. Contact Galil for more information.

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 Figure 10.1.

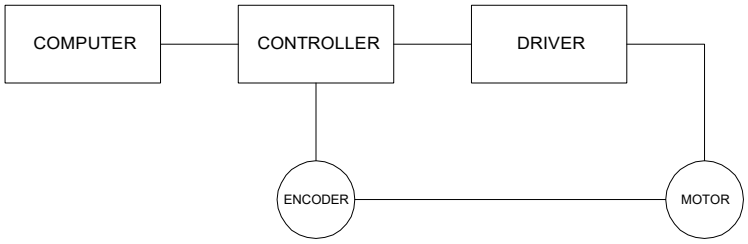


Figure 10.1: Elements of Servo Systems

The operation of such a system can be divided into three levels, defined below:

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.

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

```
PR 6000, 4000
SP 20000, 20000
AC 200000, 0
BG A
AD 2000
BG B
EN
```

This program corresponds to the velocity profiles shown in Figure 10.2. Note that the profiled positions show where the motors must be at any instant of time.

Finally, it remains up to the servo system to verify that the motor follows the profiled position by closing the servo loop.

The following section explains the operation of the servo system. First, it is explained qualitatively, and then the explanation is repeated using analytical tools for those who are more theoretically inclined.

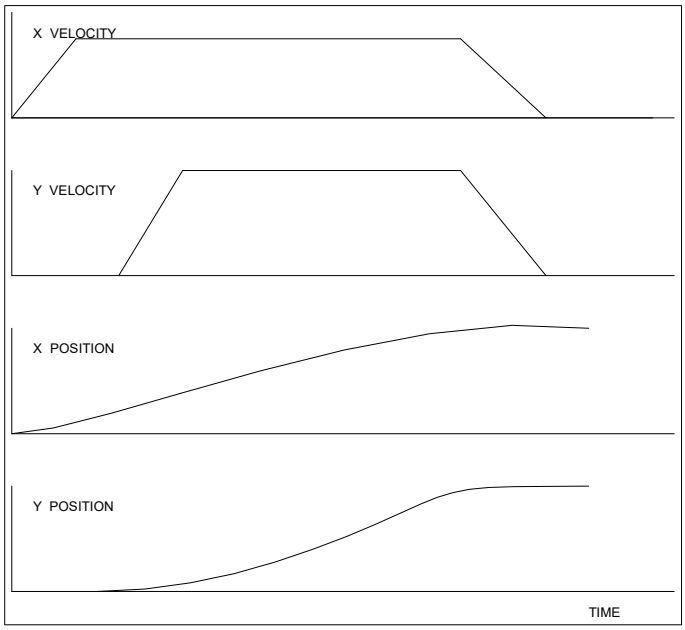


Figure 10.2: Velocity and Position Profiles

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 over-damped 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 Figure 10.3. The mathematical model of the various components is given below.

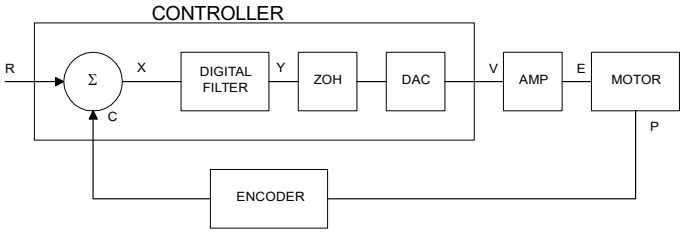


Figure 10.3: Functional Elements of a Motion Control System

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 Drive

The amplifier is a voltage source with a gain of K_v [V/V]. The transfer function relating the input voltage, V, to the motor position, P, is

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

where

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

and

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

and the motor parameters and units are

K_t	Torque constant [Nm/A]
R	Armature Resistance Ω
J	Combined inertia of motor and load [kg m^2]
L	Armature Inductance [H]

When the motor parameters are given in English units, it is necessary to convert the quantities to MKS units. For example, consider a motor with the parameters:

K_t = 14.16 oz - in/A = 0.1 Nm/A

R = 2 Ω

J = 0.0283 oz-in-s^2 = 2 * 10^{-4} kg . m^2

L = 0.004H

Then the corresponding time constants are

$T_m = 0.04$ sec

and

$T_e = 0.002$ sec

Assuming that the amplifier gain is $K_v = 4$, 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 K_a. The resulting transfer function in this case is

$P/V = K_a K_t / Js^2$

where Kt and J are as defined previously. For example, a current amplifier with $K_a = 2$ A/V with the motor described by the previous example will have the transfer function:

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

If the motor is a DC brushless motor, it is driven by an amplifier that performs the commutation. The combined transfer function of motor amplifier combination is the same as that of a similar brush motor, as described by the previous equations.

Velocity Loop

The motor driver system may include a velocity loop where the motor velocity is sensed by a tachometer and is fed back to the amplifier. Such a system is illustrated in Figure 10.4. 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, T_1, equals

$T_1 = J/K_a K_t K_g$

This leads to the transfer function

$P/V = 1/[K_g s(sT_1+1)]$

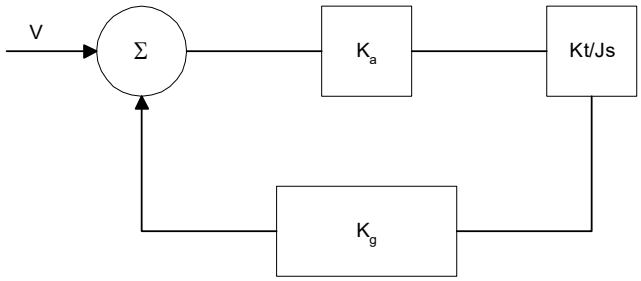


Figure 10.4: Elements of velocity loops

The resulting functions derived above are illustrated by the block diagram of Figure 10.5.

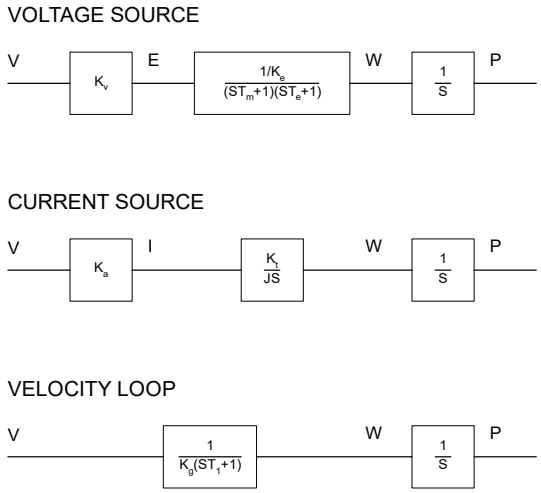


Figure 10.5: Mathematical model of the motor and amplifier in three operational modes

Encoder

The encoder generates N pulses per revolution. It outputs two signals, Channel A and B, which are in quadrature. Due to the quadrature relationship between the encoder channels, the position resolution is increased to 4N quadrature counts/rev.

The model of the encoder can be represented by a gain of

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

For example, a 1000 lines/rev encoder is modeled as

$K_f = 638$

DAC

The DAC or D-to-A converter converts a 16-bit number to an analog voltage. The input range of the numbers is 65536 and the output voltage range is ±10V or 20V. Therefore, the effective gain of the DAC is

K= 20/65536 = 0.0003 [V/count]

Digital Filter

The digital filter has three element in series: PID, low-pass and a notch filter. The transfer function of the filter. The transfer function of the filter elements are:

PID D(z) = $\frac{K(Z-A)}{Z} + \frac{CZ}{Z-1}$

Low-pass L(z) = P_m

Notch N(z) = $\frac{(Z-z)(Z-\bar{z})}{(Z-p)(Z-\bar{p})}$

The filter parameters, K, A, C and B are selected by the instructions KP, KD, KI and PL, respectively. The relationship between the filter coefficients and the instructions are:

K = (KP + KD)
A = KD/(KP + KD)
C = KI
B = PL

The PID and low-pass elements are equivalent to the continuous transfer function G(s).

G(s) = (P + sD + I/s) · a / (s + a)

where,

T = TM
P = KP
D = T·KD
I = KI/T

a = $\frac{1}{T}\ln\left(\frac{1}{B}\right)$

where T is the sampling period, and B is the pole setting

For example, if the filter parameters of the DMC-41x3 are

KP **16**
KD **144**
KI **1**
PL **0.75**
TM **1000** (or 0.001 s)

the digital filter coefficients are

K = 160
A = 0.9
C = 2
a = 250 rad/s

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

G(s) = [16 + 0.144s + 2000/s] · 250/ (s+250)

The notch filter has two complex zeros, z and $\bar{z}$, and two complex poles, p and $\bar{p}$.

The purpose of the notch filter is to cancel the resonance effect by placing the complex zeros on top of the resonance poles. The notch poles, p and $\bar{p}$, are programmable and are selected to have sufficient damping. It is best to select the notch parameters by the frequency terms. The poles and zeros have a frequency in Hz, selected by the command NF. The real part of the poles is set by NB and the real part of the zeros is set by NZ.

The most simple procedure for setting the notch filter is to identify the resonant frequency and set NF to the same value. Set NB to about one half of NF and set NZ to a low value between zero and 5.

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-41x3 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 = 50		Digital filter gain
KD = 980		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

$M(s) = P/I = K_t/Js^2 = 500/s^2$ [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) = 1030 (z-0.95)/Z

Accordingly, the coefficients of the continuous filter are:

P = 50

D = 0.98

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

$$G(s) = 50 + 0.98s = .098\ (s+51)$$

The system elements are shown in Figure 10.6.

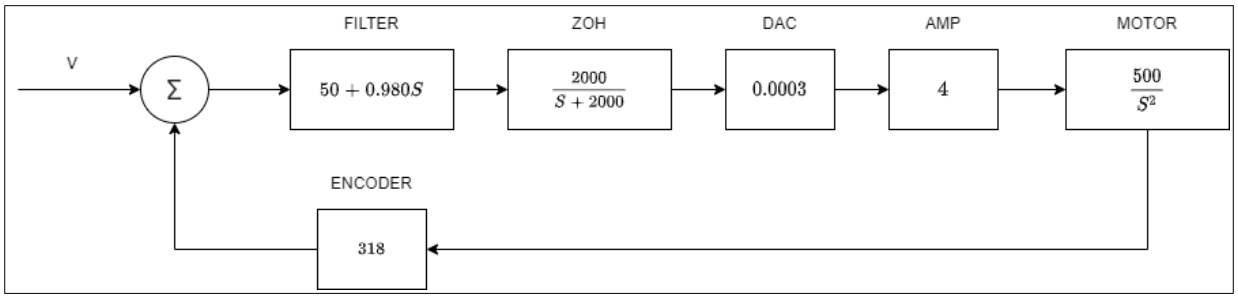


Figure 10.6: Mathematical model of the control system

The open loop transfer function, A(s), is the product of all the elements in the loop.

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

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 Figure 10.7.

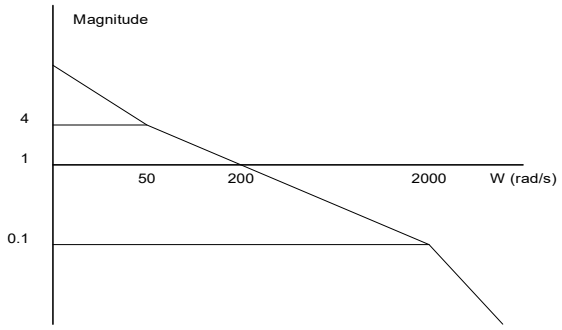


Figure 10.7: Bode plot of the open loop transfer function

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

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

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

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

$$\alpha = 76° - 180° - 6° = -110°$$

Finally, the phase margin, PM, equals

$$PM = 180° + \alpha = 70°$$

As long as PM is positive, the system is stable. However, for a well damped system, PM should be between 30° and 45°. The phase margin of 70° given above indicated over-damped 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-41x3 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= 0.2	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 theDMC-41x3 outputs ±10V for a 16-bit command of ±32768 counts.

The design objective is to select the filter parameters in order to close a position loop with a crossover frequency of ω_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

$M(s) = P/I = Kt/Js^2 = 1000/s^2$

Amp

$K_a = 2$ [Amp/V]

DAC

$K_d = 10/32768 = .0003$

Encoder

$K_f = 4N/2\pi = 636$

ZOH

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

Compensation Filter

$G(s) = P + sD$

The next step is to combine all the system elements, with the exception of G(s), into one function, L(s).

$$L(s) = M(s)\, K_a\, K_d\, K_f\, H(s) = 3.17 \cdot 10^6/[s^2(s+2000)]$$

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 ω_c = 500.

$$L(j500) = 3.17 \cdot 10^6/[(j500)^2\,(j500+2000)]$$

This function has a magnitude of:

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

and a phase:

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

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

$$|A(j500)| = 1$$
$$\text{Arg}\,[A(j500)] = -135°$$

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:

$$P = 160\cos 59° = 82.4$$
$$500D = 160\sin 59° = 137$$

Therefore,

$$D = 0.274$$

and

$$G = 82.4 + 0.274s$$

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

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

where

$$P = KP$$
$$D = KD \cdot T$$

and

$$KD = D/T$$

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

$$KP = 82.4$$
$$KD = 274$$

The DMC-41x3 can be programmed with the instruction:

```
KP 82.4;KD 274
```

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-41x3

Digital	$D(z) = [K(z-A/z) + Cz/(z-1)] \cdot (1-B)/(Z-B)$
KP, KD, KI, PL	$K = (KP + KD)$
	$A = KD/(KP+KD)$
	$C = KI$
	$B = PL$
Digital	$D(z) = [KP + KD(1-z^{-1}) + KI/2(1-z^{-1})] \cdot (1-PL)/(Z-PL)$
Continuous	$G(s) = (P + Ds + I/s) \cdot a/(s+a)$
PID, T	$P = KP$
	$D = T * KD$
	$I = KI / T$
	$a = 1/T \ln(1/PL)$

Appendices

Electrical Specifications

NOTE	Electrical specifications are only valid once controller is out of reset.

Servo Control

Motor command line	±10 V analog signal Resolution: 16-bit DAC or 0.0003 volts 3 mA maximum. Output impedance – 500 Ω
Main and auxiliary encoder inputs	TTL compatible, but can accept up to ±12 volts Quadrature phase on CHA, CHB Single-ended or differential Maximum A, B edge rate: 15 MHz Minimum IDX pulse width: 39 nsec

Stepper Control

STPn (Step)	TTL (0-5 volts) level at 50% duty cycle. 3,000,000 pulses/sec maximum frequency
DIRn (Direction)	TTL (0-5 volts)

Input / Output

Opto-isolated Inputs: DI[16:1]*, Limit switches, home, abort, reset	2.2 kΩ in series with opto-isolator Active high or low requires at least 1mA to activate. Once activated, the input requires the current to go below 0.5mA. All Limit Switch and Home inputs use one common voltage (LSCOM) which can accept up to 24 volts. Voltages above 24 volts require an additional resistor. ≥ 1 mA = ON; ≤ 0.5 mA = OFF *[8:1] for 1-4 axes models, [16:1] for 5-8 axes models
Analog Inputs: AI[8:1]	±10 volts 12-Bit Analog-to-Digital converter 16-bit optional
Digital Outputs: DO[16:1]*	4mA sinking (25mA sinking/sourcing and 500mA sourcing options) *[8:1] for 1-4 axes models, [16:1] for 5-8 axes models
Auxiliary Inputs as Uncommitted Inputs: DI[96:81]*	The auxiliary pins can be used as uncommitted inputs and are assigned to the following bits: Axis A: DI81, DI82 Axis B: DI83, DI84 Axis C: DI85, DI86 Axis D: DI87, DI88 Axis E: DI89, DI90 Axis F: DI91, DI92 Axis G: DI93, DI94 Axis H: DI95, DI96 These inputs have the same specifications as listed above for encoder inputs. *The number of auxiliary inputs is dependent on the number of axes ordered

Power Requirements

20-80 V_{DC}	10 W at 25° C

+5, ±12V Power Output Specifications

Output Voltage	Tolerance	Max Current Output
+5V	±5%	1.1A
+12V	±5%	40mA
-12V	±5%	40mA

Certifications

The DMC-4103 is certified for the following when the product or package is marked.

UL

https://www.galil.com/about/ul-certification

Performance Specifications

Minimum Servo Loop Update Time/Memory:

Minimum Servo Loop Update Time	
DMC-4113	125 µsec
DMC-4123	125 µsec
DMC-4133	250 µsec
DMC-4143	250 µsec
DMC-4153	375 µsec
DMC-4163	375 µsec
DMC-4173	500 µsec
DMC-4183	500 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 15,000,000 counts/sec servo; 3,000,000 pulses/sec-stepper
Velocity Resolution	2 counts/sec
Motor Command Resolution	16 bit or 0.0003 V
Variable Range	±2 billion
Variable Resolution	1×10^{-4}
Number of Variables	510
Array Size	24000 elements, 30 arrays
Program Size	4000 lines x 80 characters

Ordering Options

Overview

The DMC-41x3 can be ordered in many different configurations and with different options. This section provides information regarding the different options available on the DMC-41x3 motion controller, axis-specific options, and internal amplifiers. For information on pricing and how to order your controller with these options, see our DMC-41x3 part number generator on our website.

http://galil.com/order/part-number-generator/dmc-41x3

DMC and Form Factor , "DMC-41x3-XXXX(Y)" Options

The following options are the "-XXXX" form factor options that determine how the DMC-41x3 product will be delivered. See TABLE for example images.

CARD

This is the default option if none is specified and no internal amplifier is being used. If an internal amplifier is being used, the BOXn (where n stands for 4 or 8) option is required.
Part number ordering example: DMC-4153-CARD

BOX4, BOX8

The BOX4 or BOX8 option on the DMC-41x3 provides a metal enclosure for the controller. BOX4 is the box for the 1-4 axis controllers, the BOX8 is for 5-8 axis controllers. The BOX*n* option is required when using Galil internal amplifiers and is recommended for any application that requires CE certification.
Part number ordering example: DMC-4113-BOX4
The following options are the "Y" configuration options that can be added to the DMC part number. Often, multiple Y-options can be ordered per board as long as they're separated by a comma:

DIN – DIN Rail Mounting

The DIN option on the DMC-41x3 motion controller provides DIN rail mounts on the base of the controller. This will allow the controller to be mounted to any standard DIN rail. Requires BOX4 or BOX8 option.

Part number ordering example: DMC-4113-BOX4-DIN

12V – Power Controller with 12VDC

The 12V option allows the controller to be powered with a regulated 12V supply. The tolerance of the 12V input must be within ±5%. If ordered with an internal amplifier, the 12V will automatically be upgraded to the ISCNTL option, see ISCNTL – Isolate Controller Power, pg 173. The controller will be powered through the 2-pin Molex connector on the front face of the controller. Molex connector part numbers and power connector pin-outs can be found here: Power Connector Part Numbers, pg 176.

Part number ordering example: DMC-4113-CARD(12V)

16 bit – 16 bit Analog Inputs

The 16 bit option provides 16 bit analog inputs on the DMC-41x3 motion controller. The standard resolution of the analog inputs is 12 bits.

Part number ordering example: DMC-4113-CARD(-16bit)

4-20mA – 4-20mA Analog Inputs

The 4-20mA option converts all 8 analog inputs into 4-20mA analog inputs. This is accomplished by installing 475Ω precision resistors between the analog inputs and ground. When using this option the analog inputs should be configured for 0-10V analog inputs using the AQ command (AQ n,4). The equation for calculating the current is:

$$I_{ma} = 2.105*V$$

Where I_{ma} = current in mA
V = Voltage reading from DMC-41x3

Part number ordering example: DMC-4113-CARD(4-20mA)

TRES – Encoder Termination Resistors

The TRES option provides termination resistors on all of the main and auxiliary encoder inputs on the DMC-41x3 motion controller. The termination resistors are 120 Ω, and are placed between the positive and negative differential inputs on the Main A, B, Index channels as well as the Auxiliary A and B channels as in Figure A.1.

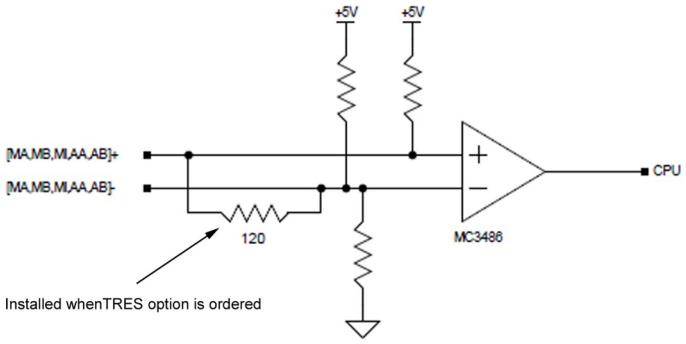


Figure A.1: Encoder Inputs with -TRES option

Note: Single ended encoders will not operate correctly with the termination resistors installed. If a combination of differential encoder inputs with termination resistors and single ended encoders is required on the same controller, contact Galil directly.

Part number ordering example: DMC-4113-CARD(TRES)

ISCNTL – Isolate Controller Power

The ISCNTL option isolates the power input for the controller from the power input of the amplifiers. When using this option, it is important that users connect the returns of the power supplies together. With this option, the power is brought in through the 2 pin Molex connector on the front face of the controller. This option is only valid when Galil amplifiers are ordered with the DMC-41x3. Molex connector part numbers and power connector pin-outs can be found here: Power Connector Part Numbers, pg 176.

Part number ordering example: DMC-4113-BOX4(ISCNTL)-D3020

RS-422 – Auxiliary Serial Port Serial Communication

The default serial configuration for the DMC-41x3 is to have RS-232 communication on the Aux (P2) serial port. The controller can be ordered to have RS-422 for this port. RS-422 communication is a differentially driven serial communication protocol that should be used when long distance serial communication is required in an application.

Part number ordering example: DMC-4113-CARD(P2422)

See the Jumper Description for DMC-41x3 section for termination resistor options for CTS and RXD signals.

See the DB-9 Female Serial Connector (Aux Serial Port) section for pin-outs.

MO – Motor Off Jumpers Installed

When a jumper is installed on the "MO" pins, the controller will be powered up in the "motor off" state. This option will cause jumper to be installed at the factory.

Part number ordering example: DMC-4123-BOX4(MO)-D3520

INVELO – Inverted ELO (Electronic Lock-Out) Logic

This option will invert the logic for the ELO input. With this option, current flowing through the ELO circuit (normally closed) is normal operation and will allow Galil internal amplifiers to be powered. An open circuit or no current flowing will activate the ELO and cut power to the amplifier. If a 5 through 8 axis unit is ordered, the inverted logic will apply to both amplifiers.

Part number ordering example: DMC-4123-BOX4(INVELO)-D3520

Axis-specific, "-ABDC(Y), -EFGH(Y)" Options

The following options are the "Y" configuration options that can be added to the axis-specific part numbers. Often, multiple Y-options can be ordered per bank of 4 axis as long as they're separated by a comma.

SER – BiSS and SSI Absolute Encoder Option

The DMC-41x3 controller can be configured to support BiSS and SSI encoders.

For the BiSS protocol, this option configures the DMC-41x3 with hardware and firmware that supports gathering position data from BiSS-B and BiSS-C absolute encoders. Galil supports point-to-point configuration and one slave device per axis of the DMC-41x3. Galil does not support bus configuration. Encoder resolutions up to 31 bits are supported. For more information, please contact Support at Galil.

For the SSI protocol, this option configures the DMC-41x3 with hardware and firmware that supports gathering position data from SSI absolute encoders. Galil supports one slave device per axis of the DMC-41x3. Encoder resolutions up to 31 bits are supported. For more information, please contact Support at Galil.

For more information, see the SS command for BiSS and the SI command for SSI in the DMC-41x3 Command Reference. For pin-outs, refer to Encoder 26 pin HD D-Sub Connector (SER option).

LSNK – 25mA Sinking Outputs

The LSNK option modifies the digital outputs on the DMC-41x3 to be capable of sinking up to 25mA per output. For detailed information see the 25mA Sinking Optoisolated Outputs (LSNK) section in Chapter 3 Connecting Hardware.

Part number ordering example: DMC-4113-CARD-ABCD(LSNK)

LSRC – 25mA Sourcing Outputs

The LSRC option modifies the digital outputs on the DMC-41x3 to be capable of sourcing up to 25mA per output. For detailed information see the 25mA Sourcing Optoisolated Outputs (LSRC) section in Chapter 3 Connecting Hardware.

Part number ordering example: DMC-4113-CARD-ABCD(LSRC)

HSRC – 500mA Sourcing Outputs

The HSRC option modifies the digital outputs on the DMC-41x3 to be capable of sourcing up to 500mA per output. For detailed information see the 500mA Sourcing Optoisolated Outputs (HSRC) section in Chapter 3 Connecting Hardware.

Part number ordering example: DMC-4113-CARD-ABCD(HSRC)

AMP/SDM, "-DXXXX(Y)" Internal Amplifier Options

The following options are the "Y" configuration options that can be added to the -DXXXX(Y) amplifier part number. Often, multiple Y-options can be ordered per board as long as they're separated by a comma.

ISAMP – Isolation of Power Between Each Amplifier

The ISAMP option separates the power pass-through between the Axes 1-4 amplifier and the Axes 5-8 amplifier. This allows the 2 internal amplifiers to be powered at separate voltages. When using this option, it is important that users connect the returns of the power supplies together.

If the ISCNTL option is NOT ordered on the DMC-41x3, the amplifier with the higher bus voltage will automatically power the controller. The amplifier with the higher voltage, and the voltage level does not have to be specified during time of purchase as long as the voltage falls within the range of 20-80VDC.

This option is only valid on the 5-8 Axes amplifier board.

Part number ordering example: DMC-4183-BOX8-D3040-D3040-ISAMP

SR90 – SR-49000 Shunt Regulator Option

The SR-49000 is a shunt regulator for the DMC-41x3 controller and internal amplifiers. This option is highly recommended for any application where there is a large inertial load, or a gravitational load. To calculate if your system requires a Shut Regulator, see Application Note #5448: "Shunt Regulator Operation" linked below:

http://galil.com/download/application-note/note5448.pdf

The SR-49000 is installed inside the box of the DMC-41x3 controller, so it does not effect the form of the unit.

As a functional example, -SR90 shunt regulator activates when the voltage supplied to the amplifier rises above 90V. When activated, the power from the power supply is dissipated through a 5 Ω , 20W power resistor.

When used a 5-8 axis controller with two internal amplifiers, the -SR90 option can protect both internal amplifiers. However, with an -ISAMP (ISAMP – Isolation of Power Between Each Amplifier) option, the -SR90 option will only protect the first four axis with internal amplifiers.

Part number ordering example: DMC-4143-BOX4-D3040-SR90

SSR – Solid State Relay Option for AMP-43140

The SSR option configures the AMP-43140 (-D3140) with Solid State Relays on the motor power leads that are engaged and disengaged when the amplifier is enabled and disabled. See the -SSR Option in the AMP-43140 section of the Appendix for more information.

This option is only valid with the AMP-43140.

Part number ordering example: DMC-4143-BOX4-D3140(SSR)

100mA – 100mA Maximum Current Output for AMP-43140

The 100mA option configures the AMP-43140 (-D3140) for 10mA/V gain with a maximum current output of 100mA. This option is only valid with the AMP-43140.

Part number ordering example: DMC-4143-BOX4(ISCNTL)-D3140(100mA)

HALLF – Filtered Hall Sensor Inputs

The HALLF option will place a capacitor between the hall input and digital GND to filter unwanted noise. This results in cleaner, more reliable hall sensor reads. The HALLF option is only available for Galil's internal PWM amplifiers.

Part number ordering example: DMC-4143-BOX4-D3040(HALLF)

Power Connector Part Numbers

Overview

The DMC-41x3 uses Molex Mini-Fit, Jr.™ Receptacle Housing connectors for connecting DC Power to the Amplifiers, Controller, and Motors. This section gives the specifications of these connectors. For information specific to your Galil amplifier or driver, refer to the specific amplifier/driver in the Integrated Components section.

Molex Part Numbers

There are 3 different Molex connectors used with the DMC-41x3. The type of connectors on any given controller will be determined be the Amplifiers/Drivers that were ordered. Below are tables indicating the type of Molex Connectors used and the specific part numbers used on each Amplifier or Driver. For more information on the connectors, go to http://www.molex.com/

On Board Connector	Common Mating Connectors*	Crimp Part Number	Type
MOLEX# 39-31-0060	MOLEX# 39-01-2065	MOLEX# 44476-3112	6 Position
MOLEX# 39-31-0040	MOLEX# 39-01-2045	MOLEX# 44476-3112	4 Position
MOLEX# 39-31-0020	MOLEX# 39-01-2025	MOLEX# 44476-3112	2 Position

*The mating connectors listed are not the only mating connectors available from Molex. See http://www.molex.com/ for the full list of available mating connectors.

Galil Amplifier / Driver		On Board Connector	Type
None	-	MOLEX# 39-31-0020	2 Position
AMP-43040	Power	MOLEX# 39-31-0060	6 Position
	Motor	MOLEX# 39-31-0040	4 Position
AMP-43140	Power	MOLEX# 39-31-0040	4 Position
	Motor	MOLEX# 39-31-0020	2 Position
SDM-44040	Power	MOLEX# 39-31-0060	6 Position
	Motor	MOLEX# 39-31-0040	4 Position
SMD-44140	Power	MOLEX# 39-31-0060	6 Position
	Motor	MOLEX# 39-31-0040	4 Position

Cable Shielding

Electrical noise can negatively impact system performance by affecting different machine components. For example, it can cause the incorrect reporting of digital input states or position data. One simple and effective way to mitigate electrical noise is to add a shield around the encoder and motor phase cables. For proper shielding, the shield's drain wire should be terminated on only one side of the cable. The following diagrams illustrate the best means of achieving this with the DMC-41x3, with the expectation that the controller's baseplate is mounted to something with a secure connection to Earth Ground.

Figure A.2 shows how to connect the encoder cable's shield drain wire. This side of the cable should then go to the controller's 26 pin encoder connector, making a connection to Earth Ground.

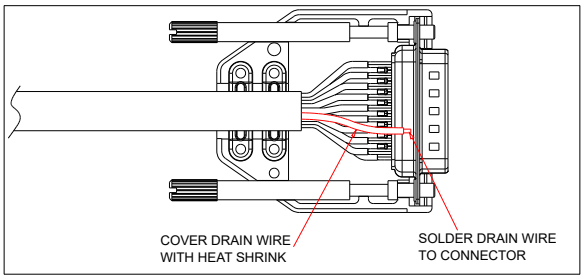


Figure A.2: Drain wire of the encoder cable shield

Figure A.3 shows how to connect the drain wire to the motor phase shield.

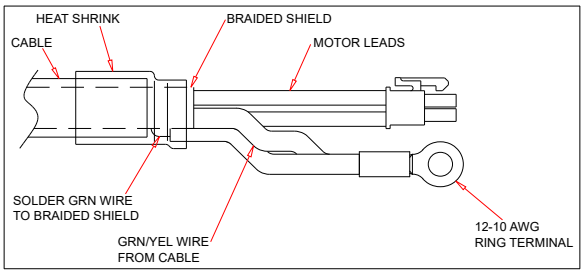


Figure A.3: Drain wire of the motor phase shield

The motor phase drain wire is fastened to Earth ground on the controller box, shown in Figure A.4.

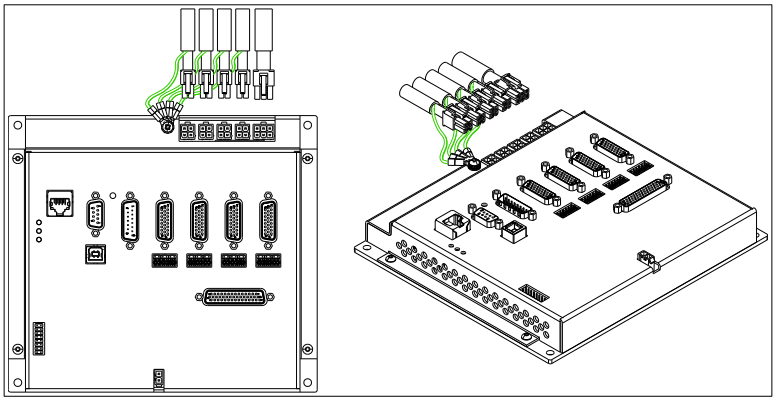


Figure A.4: Ring terminal of motor phase drain wire connected to box of DMC-4143

For more information on the theory behind electrical noise and its different sources, see Application Note #5438, https://www.galil.com/download/application-note/note5438.pdf

Input Current Limitations

The current for an optoisolated input shall not exceed 11mA. Some applications may require the use of an external resistor (R) to limit the amount of current for an input. These external resistors can be placed in series between the inputs and their power supply (Vs). To determine if an additional resistor (R) is required, follow Equation A1 below for guidance.

$$1\,mA < \frac{Vs}{R + 2200\,\Omega} < 11\,mA$$

Equation A1: Current limitation requirements for each input.

Pin-outs

I/O (A-D) 44 pin HD D-Sub Connector (Female)

Pin#	Label	Description	Pin#	Label	Description	Pin#	Label	Description
1	ERR	Error Output	16	RST	Reset Input	31	GND	Digital Ground
2	DI1	Digital Input 1/ A latch	17	INCOM0	Input Common (DI 1-8)	32	DI2	Digital Input 2 / B latch
3	DI4	Digital Input 4 / D latch	18	DI3	Digital Input 3 / C latch	33	DI5	Digital Input 5
4	DI7	Digital Input 7	19	DI6	Digital Input 6	34	DI8	Digital Input 8
5	ELO	Electronic Lock Out	20	ABRT	Abort Input	35	GND	Digital Ground
6	LSCOM0	Limit Switch Com (A-D)	21	N/C	No Connect	36	FLSA	Forward Limit Switch A
7	HOMA	Home Switch A	22	RLSA	Reverse Limit Switch A	37	FLSB	Forward Limit Switch B
8	HOMB	Home Switch B	23	RLSB	Reverse Limit Switch B	38	FLSC	Forward Limit Switch C
9	HOMC	Home Switch C	24	RLSC	Reverse Limit Switch C	39	FLSD	Forward Limit Switch D
10	HOMD	Home Switch D	25	RLSD	Reverse Limit Switch D	40	GND	Digital Ground
11	OP0A	Output GND/PWR (Bank 0)	26	OP0A	Output GND/PWR (Bank 0)	41	DO1	Digital Output 1
12	DO3	Digital Output 3	27	DO2	Digital Output 2	42	DO4	Digital Output 4
13	DO6	Digital Output 6	28	DO5	Digital Output 5	43	DO7	Digital Output 7
14	OP0B	Output PWR/GND (Bank 0)	29	DO8	Digital Output 8	44	CMP	Output Compare (A-D)
15	+5V	+5V	30	+5V	+5V			

I/O (E-H) 44 pin HD D-Sub Connector (Female)

For DMC-4153 through DMC-4183 controllers only.

Pin#	Label	Description	Pin#	Label	Description	Pin#	Label	Description
1	ERR	Error Output	16	RST*	Reset Input	31	GND	Digital Ground
2	DI9	Digital Input 9 / E latch	17	INCOM1	Input Common (DI 9-16)	32	DI10	Digital Input 10 / F latch
3	DI12	Digital Input 12/H latch	18	DI11	Digital Input 11 / G latch	33	DI13	Digital Input 13
4	DI15	Digital Input 15	19	DI14	Digital Input 14	34	DI16	Digital Input 16
5	ELO*	Electronic Lock Out	20	ABRT*	Abort Input	35	GND	Digital Ground
6	LSCOM1	Limit Switch Com (E-H)	21	N/C	No Connect	36	FLSE	Forward Limit Switch E
7	HOME	Home Switch E	22	RLSE	Reverse Limit Switch E	37	FLSF	Forward Limit Switch F
8	HOMF	Home Switch F	23	RLSF	Reverse Limit Switch F	38	FLSG	Forward Limit Switch G
9	HOMG	Home Switch G	24	RLSG	Reverse Limit Switch G	39	FLSH	Forward Limit Switch H
10	HOMH	Home Switch H	25	RLSH	Reverse Limit Switch H	40	GND	Digital Ground
11	OP1A	Output GND/PWR (Bank 1)	26	OP1A	Output GND/PWR (Bank 1)	41	DO9	Digital Output 9
12	DO11	Digital Output 11	27	DO10	Digital Output 10	42	DO12	Digital Output 12
13	DO14	Digital Output 14	28	DO13	Digital Output 13	43	DO15	Digital Output 15
14	OP1B	Output PWR/GND (Bank 1)	29	DO16	Digital Output 16	44	CMP	Output Compare (E-H)
15	+5V	+5V	30	+5V	+5V			

* ABRT, RST and ELO use INCOM0

Encoder 26 pin HD D-Sub Connector (Female)

Pin #	Label	Description	Pin #	Label	Description
1	HALC	Hall C	14	FLS	Forward Limit Switch Input
2	AEN	Amplifier Enable	15	AB+	B+ Aux Encoder Input
3	DIR	Direction	16	MI-	I- Index Pulse Input
4	HOM	Home	17	MB+	B+ Main Encoder Input
5	LSCOMn	Limit Switch Common*	18	GND	Digital Ground
6	AA-	A- Aux Encoder Input	19	MCMD	Motor Command
7	MI+	I+ Index Pulse Input	20	ENBL+	Amp Enable Power
8	MA-	A- Main Encoder Input	21	HALA	Hall A
9	+5V	+5V	22	RLS	Reverse Limit Switch Input
10	GND	Digital Ground	23	AB-	B- Aux Encoder Input
11	ENBL-	Amp Enable Return	24	AA+	A+ Aux Encoder Input
12	HALB	Hall B	25	MB-	B- Main Encoder Input
13	STP	Step	26	MA+	A+ Main Encoder Input

* LSCOMn on JA1, JB1, JC1 and JD1 is common to LSCOM0 on J5
LSCOMn on JE1, JF1, JG1 and JH1 is common to LSCOM1 on J8

Encoder 26 pin HD D-Sub Connector (SER option)

Pin #	Label	Description	Pin #	Label	Description
1	HALC	Hall C	14	FLS	Forward Limit Switch Input
2	AEN	Amplifier Enable	**15**	**AB+**	**Data + (Dn+ or SLO+)**
3	DIR	Direction	16	MI-	I- Index Pulse Input
4	HOM	Home	17	MB+	B+ Main Encoder Input
5	LSCOMn	Limit Switch Common [2]	**18**	**GND**	**Digital Ground** [1]
6	**AA-**	**Clock - (Cn- or MA-)**	19	MCMD	Motor Command
7	MI+	I+ Index Pulse Input	20	ENBL+	Amp Enable Power
8	MA-	A- Main Encoder Input	21	HALA	Hall A
9	+5V	+5V	22	RLS	Reverse Limit Switch Input
10	**GND**	**Digital Ground** [1]	**23**	**AB-**	**Data - (Dn- or SLO-)**
11	ENBL-	Amp Enable Return	**24**	**AA+**	**Clock + (Cn+ or MA+)**
12	HALB	Hall B	25	MB-	B- Main Encoder Input
13	STP	Step	26	MA+	A+ Main Encoder Input

[1] Only one ground pin must be connected to the encoders digital ground signal.

[2] LSCOMn on JA1, JB1, JC1 and JD1 is common to LSCOM0 on J5
LSCOMn on JE1, JF1, JG1 and JH1 is common to LSCOM1 on J8

Analog 15 pin D-sub Connector (Male)

Pin #	Label	Description
1	AGND	Analog Ground
2	AI1	Analog Input 1
3	AI3	Analog Input 3
4	AI5	Analog Input 5
5	AI7	Analog Input 7
6	AGND	Analog Ground
7	-12V	-12V
8	+5V	+5V
9	AGND	Analog Ground
10	AI2	Analog Input 2
11	AI4	Analog Input 4
12	AI6	Analog Input 6
13	AI8	Analog Input 8
14	N/C	No Connect
15	+12V	+12V

Amplifier Enable Jumper Description for DMC-41x3

Jumper	Label	Function (If jumpered)
Q and P	1	Sink/Source Selection
	2	Sink/Source Selection
	3	Sink/Source Selection
	4	HAEN/LAEN Selection
	5	5V/12V/External Power Selection
	6	5V/12V/External Power Selection

Note: See Amplifier Enable in Chapter 3 for detailed information.

RJ45 Connector - Ethernet

Pin #	Signal
1	TXP
2	TXN
3	RXP
4	NC
5	NC
6	RXN
7	NC
8	NC

The Ethernet connection is Auto MDIX, 100bT/10bT.

USB

The USB port on the DMC-41x3 is a Female Type B USB port. The standard cable when communicating to a PC will be a Male Type A – Male Type B USB cable.

DB-9 Female Serial Connector (Aux Serial Port)

Pin #	Signal Direction	Signal Name	Signal Function
1	-	-	-
2	Input	RxD	Receive Data
3	Output	TxD	Transmit Data
4	-	-	-
5	-	GND	Ground
6	-	-	-
7	Input	RTS	Request To Send
8	Output	CTS	Clear To Send
9	-	-	-

RS-422-Auxiliary Port (Non-Standard Option)

Standard connector and cable when DMC-41x3 is ordered with RS-422 Option. For detailed information on the RS-422 option see RS-422 – Auxiliary Serial Port Serial Communication in the Appendices.

Pin #	Signal Direction	Signal Name	Signal Function
1	Output	CTS-	Clear To Send
2	Input	RXD-	Receive Data
3	Output	TXD-	Transmit Data
4	Input	RTS-	Request To Send
5	-	GND	Ground
6	Output	CTS+	Clear To Send
7	Input	RXD+	Receive Data
8	Output	TXD+	Transmit Data
9	Input	RTS+	Request To Send

Jumper Description for DMC-41x3

Label	Function (If jumpered)
ARXD	RS-422 Option Only: Connects a 120 Ω Termination resistor between the differential "Receive" inputs on the Aux Serial port. Pins 2 and 7 on RS-422 Auxiliary Port.
ACTS	RS-422 Option Only: Connects a 120 Ω Termination resistor between the differential "Clear To Send" inputs on the Aux Serial port. Pins 1 and 6 on RS-422 Auxiliary Port.
APWR	Connects 5V to pin 9 on the Aux serial port connector (J3)
OPT	Reserved
MO	When controller is powered on or reset, Amplifier Enable lines will be in a Motor Off state. A SH will be required to re-enable the motors.
19.2K	Baud Rate setting – see table below
UPGD	Applied for recovery from a failed firmware upgrade.
MRST	Master Reset enable. Returns controller to factory default settings and erases EEPROM. Requires power-on or RESET to be activated.

Note: The ARXD and ACTS jumpers should be installed for single-drop RS-422. For multi-drop, the jumpers should be installed on the last device.

Baud Rate Jumper Settings

19.2	BAUD RATE
ON	19200
OFF	115200 (Recommended)

Signal Descriptions

Inputs

Encoder, MA+, MB+	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 15,000,000 quadrature states/sec. The controller performs quadrature decoding of the encoder signals resulting in a resolution of quadrature counts (4 x encoder cycles). Encoders that produce outputs in the format of pulses and direction may also be used by inputting the pulses into MA+ and direction into MB+ and using the CE command to configure this mode.
Encoder Index, MI+	Once-Per-Revolution encoder pulse. Used in Homing sequence or Find Index command to define home on an encoder index.
Encoder, MA-, MB-, MI-	Differential inputs from encoder. May be input along with MA+, MB+ for noise immunity of encoder signals. The MA- and MB- inputs are optional.
Auxiliary Encoder, AA+, AB+, Aux A-, Aux B-	Inputs for additional encoder. Used when an encoder on both the motor and the load is required. Not available on axes configured for step motors.
Abort	A low input stops commanded motion instantly without a controlled deceleration. Also aborts motion program.
Reset	A low input resets the state of the processor to its power-on condition. The previously saved state of the controller, along with parameter values, and saved sequences are restored.
Electronic Lock Out	Input that when triggered will shut down the amplifiers at a hardware level. Useful for safety applications where amplifiers must be shut down at a hardware level.
Forward Limit Switch	When active, inhibits motion in forward direction. Also causes execution of limit switch subroutine, #LIMSWI. The polarity of the limit switch may be set with the CN command.
Reverse Limit Switch	When active, inhibits motion in reverse direction. Also causes execution of limit switch subroutine, #LIMSWI. The polarity of the limit switch may be set with the CN command.
Home Switch	Input for Homing (HM) and Find Edge (FE) instructions. Upon BG following HM or FE, the motor accelerates to slew speed. A transition on this input will cause the motor to decelerate to a stop. The polarity of the Home Switch may be set with the CN command.
Input 1 - Input 8 isolated Input 9 - Input 16 isolated	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.
Latch	High speed position latch to capture axis position on occurrence of latch signal. AL command arms latch. Input 1 is latch A, Input 2 is latch B, Input 3 is latch C and Input 4 is latch D. Input 9 is latch E, input 10 is latch F, input 11 is latch G, input 12 is latch H.

Outputs

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.
Amplifier Enable	Signal to disable and enable an amplifier. Amp Enable goes low on Abort and OE**1**.
Step Output	For stepper 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%. For sign/magnitude mode, please contact a Galil Applications Engineer.
Direction	Provides the direction signal for step motors. For sign/magnitude mode, please contact a Galil Applications Engineer.
Error	The signal goes low when the position error on any axis exceeds the value specified by the error limit command, ER.
Output 1-Output 8 Output 9-Output 16 (DMC-4153 through 4183)	The high power optically isolated 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.

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-12:30 pm)

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 (8:00 am-5:00 pm)

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)
http://galil.com/learn/classes

Contacting Us

Galil Motion Control

270 Technology Way

Rocklin, CA 95765

Phone: 916-626-0101

Fax: 916-626-0102

E-Mail Address: support@galil.com

Web: http://galil.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.

Integrated Components

Overview

When ordered, the following components will reside inside the box of the DMC-41x3 motion controller. The amplifiers and stepper drivers provide power to the motors in the system, and the interconnect modules and communication boards provide the connections for the signals and communications.

A1 – AMP-430x0 (-D3040,-D3020): 500W Trapezoidally Commutated Amplifiers

The AMP-43040 (4-axis) and AMP-43020 (2-axis) are trapezoidally commutated transconductance, PWM amplifiers for driving brushless or brush-type servo motors. They are capable of providing 500 watts of continuous power per axis. The AMP-43040/43020 drive modules are enclosed in the metal enclosure of the DMC-41x3. The standard amplifier accepts DC supply voltages from 20-80 VDC.

A2 – AMP-43140 (-D3140): 20W Linear Brush-type Amplifiers

The AMP-43140 (4-axis) provides four linear drives for operating small brush-type servo motors. It is capable of providing 20 watts per axis, 60 watts total. The AMP-43140 drive modules are enclosed in the metal enclosure of the DMC-41x3. The standard amplifier accepts DC supply voltages from ±12-30 VDC.

A3 – AMP-43240 (-D3240): 750W Trapezoidally Commutated Amplifiers

The AMP-43240 (4-axis) is a trapezoidally commutated transconductance, PWM amplifier for driving brushless or brush-type servo motors. It is capable of providing 750 watts of continuous power per axis. The AMP-43240 Brushless drive modules are enclosed in the metal enclosure of the DMC-41x3. The standard amplifier accepts DC supply voltages from 20-80 VDC.

A4 – AMP-43540 (-D3540, -D3520): 600W Sinusoidally Commutated Amplifiers

The AMP-43540 (4-axis) and AMP-43520 (2-axis) are sinusoidally commutated, 16 bit PWM amplifiers for driving brushless and brush-type servo motors. They are capable of providing 600 watts of continuous power per axis. The AMP-43540/43520 drive modules are enclosed in the metal enclosure of the DMC-41x3. The standard amplifier accepts DC supply voltages from 20-80 VDC.

A5 – AMP-43547 (-D3547, -D3527): 600W Configurable Servo Amps/Stepper Drives

This amplifier is unique in that it can run either servo or stepper motors on any axis.
The AMP-43547 (4-axis) and AMP-43527 (2-axis) are sinusoidally commutated, 16 bit PWM amplifiers for driving brushless and brush-type servo motors. The AMP-43547 can also be configured by the user as 1/256 microstepping drives for operating two-phase bipolar stepper motors.
The AMP-43540/43527 are capable of providing 600 watts of continuous power per axis. The AMP-43547/43527 drive modules are enclosed in the metal enclosure of the DMC-41x3. The standard amplifier accepts DC supply voltages from 20-80 VDC.

A6 – AMP-43640 (-D3640): 20W Sinusoidally Commutated Linear Amplifiers

The AMP-43640 (4-axis) provides four linear drives for sinusoidally commutating brushless motors. It is capable of providing 20 watts per axis, 80 watts total. The AMP-43640 drive modules are enclosed in the metal enclosure of the DMC-41x3. The standard amplifier accepts DC supply voltages from 15–40VDC.

A7 - AMP-43740 (-D3740): 1200W Sinusoidally Commutated Amplifiers

The AMP-43740 (4-axis) is a sinusoidally commutated, 16 bit PWM amplifier for driving brushless and brush-type servo motors. It is capable of providing 1200 watts of continuous power per axis. The AMP-43740 drive modules are enclosed in the metal enclosure of the DMC-41x3. The standard amplifier accepts DC supply voltages from 20-80 VDC.

A8 – SDM-44040 (-D4040,-D4020): Microstepping Stepper Drives

The SDM-44040 (4-axis) and SDM-44020 (2-axis) are configurable (full, half, 1/4 and 1/16) microstepping drives for operating two-phase bipolar stepper motors. They are capable of providing a selectable current of 0.5, 0.75, 1.0, and 1.4 amps per axis. The AMP-44040/44020 drive modules are enclosed in the metal enclosure of the DMC-41x3. The standard amplifier accepts DC supply voltages from 12-30 VDC.

A9 – SDM-44140 (-D4140): Microstepping Stepper Drives

The SDM-44140 (4-axis) is a 1/64 microstepping drive for operating two-phase bipolar stepper motors. It is capable of providing a selectable current of 0.5, 1.0, 2.0 and 3.0 amps per axis. The AMP-44140 drive modules are enclosed in the metal enclosure of the DMC-41x3. The standard amplifier accepts DC supply voltages from 20-60 VDC.

A1 – AMP-430x0 (-D3040,-D3020)

Description

The AMP-43040 resides inside the DMC-41x3 enclosure and contains four trapezoidally commutated transconductance, PWM amplifiers for driving brushless or brush-type servo motors. Each amplifier drives motors operating at up to 7 Amps continuous, 10 Amps peak, 20–80 VDC. The gain settings of the amplifier are user-programmable at 0.4 Amp/Volt, 0.7 Amp/Volt and 1 Amp/Volt. The switching frequency is 66 kHz.
The drive for each axis is software configurable to operate in either a chopper or inverter mode. The chopper mode is intended for operating low inductance motors. The amplifier offers protection for Over-Voltage, under-voltage, Over-Current, short-circuit and over-temperature. Two AMP-43040s can be used in 5- through 8-axis controllers. A shunt regulator option is available. A two-axis version, the AMP-43020 is also available. If higher voltages are required, please contact Galil.
GDK's Step-By-Step tool provides an interactive guide for configuring brushless and brushed motors.

WARNING	Do not “hot swap” the motor power or supply voltage power input connections. If the amplifier is enabled when the motor connector is connected or disconnected, damage to the amplifier can occur. Galil recommends powering the controller and amplifier down before changing the connector, and breaking the AC side of the power supply connection in order to power down the amplifier.

Electrical Specifications

	Servo (Brushless/Brush)
Supply Voltage	20-80 VDC
Continuous Current	7 Amps
Peak Current	10 Amps
Switching Frequency	66 kHz (up to 140 kHz available)
Amplifier Gains	0.4, 0.7, 1.0 Amps/Volt
Brushless Commutation Angle	120° (60° option available)
PWM Output Operation	Inverter or Chopper (user configurable)
Minimum Load Inductance	$L(mH) = \frac{Vs(V)}{480 * I_{Ripple}(A)}$

Vs = Supply Voltage, I_{ripple} = 10% of the maximum current at chosen gain

Table A1.1: Amplifier Electrical Specifications

Mating Connectors

	Cable Connectors	Terminal Pins	On board Connector
POWER	6-pin Mini-Fit, Jr.™ (Male) Molex #0039012065	Molex #0444761112	6-pin Mini-Fit, Jr.™ (Female) Molex #0039310060
A,B,C,D: 4-pin Motor Power Connectors	4-pin Mini-Fit, Jr.™ (Male) Molex #0039012045	Molex #0444761112	4-pin Mini-Fit, Jr.™ (Female) Molex #0039310040

Table A1.2: Molex part numbers for connectors and terminal pin

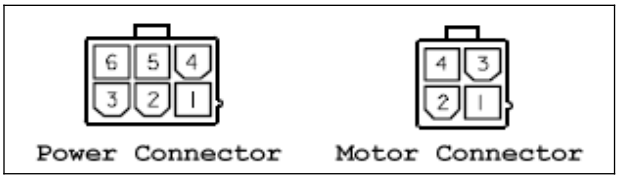


Figure A1.1: Power and Motor Connector Diagram

Power Connector		
Pin Number	**Connection**	
1,2,3	DC Power Supply Ground	
4,5,6	+VS (DC Power)	
Motor Connector		
Pin Number	**3-phase brushless**	**Brushed**
1	Phase C	No Connect
2	Phase B	Phase A-
3	No Connect	No Connect
4	Phase A	Phase A+

Table A1.3: Power and Motor Connector Pin-outs

Servo Motor Operation

Brushless Motor Setup

The AMP-430x0 supports brushless motor operation. To configure an axis for a brushless motor, connect the 3 motor leads as prescribed in Table A1.3. Connect the encoders, homes, and limits as required. Trapezoidal commutation for brushless motors requires the motors have Hall-effect sensors. Wire the Hall-effect sensors to the dedicated inputs on the encoder connector. Once complete, use GDK's Step-By-Step tool for initializing commutation.

Brushed Motor Operation

The AMP-430x0 also supports brushed motor operation. To configure an axis for a brushed motor, connect the 2 motor leads as prescribed in Table A1.3. Connect the encoders, homes, and limits as required. Then, use GDK's Step-By-Step tool for applying settings for brush-type operation.

Using External Amplifiers

The BR command must be set to a **1** for any axis where an AMP-43040 is installed but the use of an external axis is required. Doing so will this will disable the Hall Error Protection. Setting BR*m*=**1** (where *m* is the axis designator) is required for both external servo amplifiers and stepper drivers. In order to use the full torque limit, make sure the AG setting for the axes using external amplifiers are set to **0** or **1**.

Use D-Sub connectors on the top face of the controller to access necessary signals to run external amplifiers. For more information on connecting external amplifiers, see Step 2b. Connecting External Amplifiers and Motors in Chapter 2.

Setting Amplifier Gain and Current Loop Bandwidth

The AG command will set the amplifier gain (Amps/Volt). The AU command will set the current loop gain.

AG Command:

The AMP-43040 has 3 amplifier gain settings. The gain is set with the AG command. When configuring the amplifier gain, the axis must be in a motor off (MO) state prior to execution of the AG command. For more information on the AG command, refer to the command reference for the DMC-41x3.

AG setting	Gain Value
n = 0	0.4 A/V
n = 1	0.7 A/V
n = 2	1.0 A/V

Table A1.4: Amplifier Gain Settings

AU Command:

Proper configuration of the AU command is essential to optimize the operation of the AMP-43040. This command sets the gain for the current loop on the amplifier. Table A1.5 indicates the recommended AU settings for 24 and 48 VDC power supplies based on motor inductance. When configuring the current loop gain, the axis must be in a motor off (MO) state prior to execution of the AU command. For more information on the AU command, refer to the command reference for the DMC-41x3.

Vsupply VDC	Inductance L (mH)	n
24	0.5 ≤ L < 5.0	0
24	0.2 ≤ L < 0.5	0.5
24	5.0 ≤ L	1
24	5.0 ≤ L	1.5
48	0.5 ≤ L < 10.0	0
48	0.2 ≤ L < 0.5	0.5
48	10.0 ≤ L	1
48	10.0 ≤ L	1.5

Table A1.5: Amplifier Current Loop Gain Settings

Chopper Mode

The AMP-430x0 can be put into what is called a “Chopper” mode. The chopper mode is in contrast to the normal inverter mode in which the amplifier sends PWM power to the motor of ±VS. In chopper mode, the amplifier sends a 0 to +VS PWM to the motor when moving in the forward direction, and a 0 to –VS PWM to the motor when moving in the negative direction. This mode is useful when using low inductance motors because it reduces the losses due to switching voltages across the motor windings. It is recommended to use chopper mode when using motors with 200-500 ↄH inductance.

This mode is set with the AU command. A setting of 0.5 is Chopper mode with normal current bandwidth. A setting of 1.5 is Chopper mode with high current bandwidth.

Setting Peak and Continuous Torque Limits

TK and TL Commands:

The peak and continuous torque limits can be set through the TK and TL commands respectively. The controller will command the torque signal (TT) up to the value specified by TK for a maximum of one second, as shown in Figure A1.2. It will then limit the torque to the value specified by TL. The more time the controller commands the torque below TL, the longer it will command it to TK when necessary. The DMC code example below exhibits the behavior of TT being limited by TL and TK, as shown in Figure A1.2.

```
#tk;          'begin program
MO;           'disable all axes
AGA=2;        'set max amplifier gain of 1.0 A/V
TLA=5;        'set continuous torque limit of 5V
TKA=9.9982;   'set max peak torque limit to 9.9982V
SHA;          'enable A axis
OFA=9.9982;   'command torque signal to 9.9982V
EN;           'end program
```

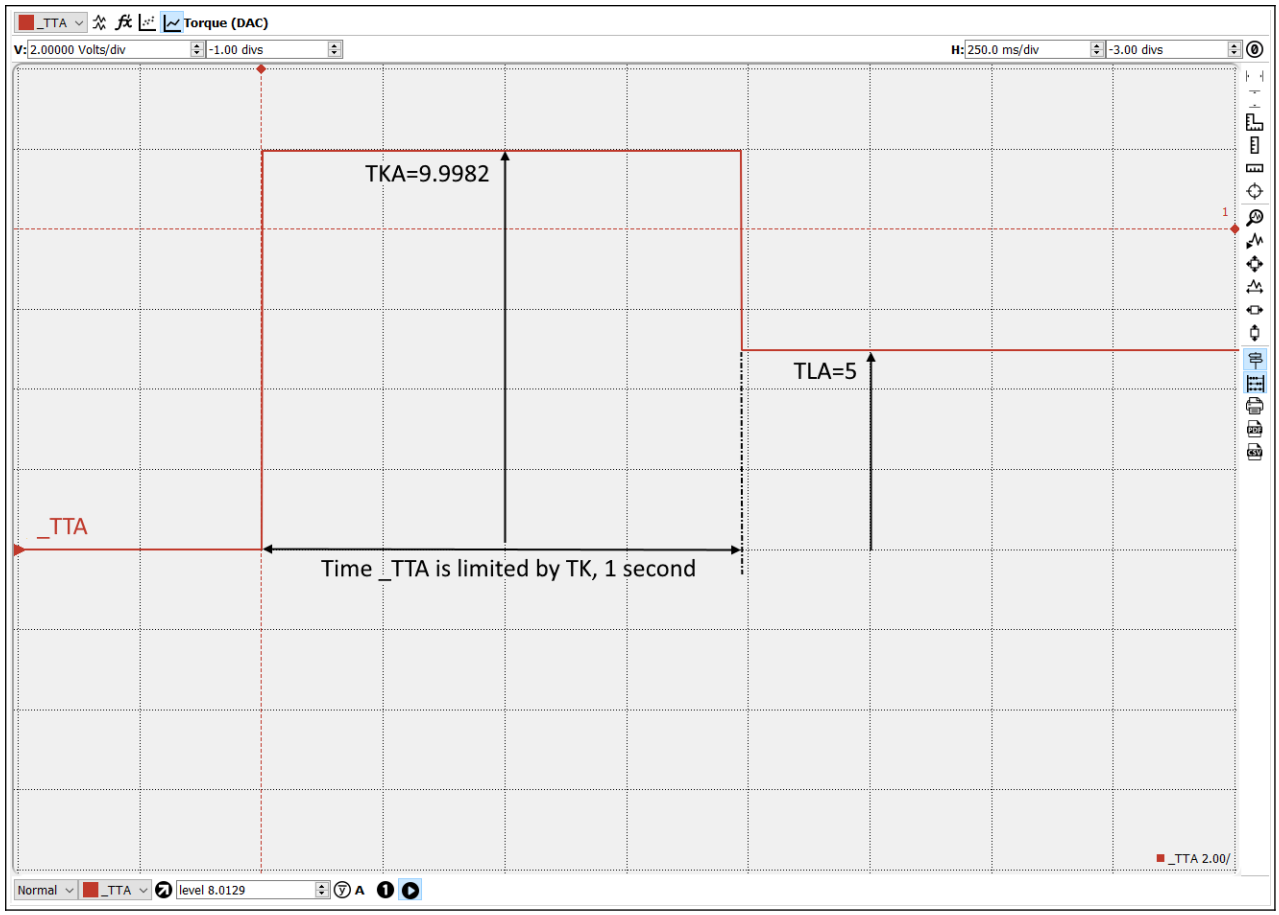


Figure A1.2: Peak Current Operation

Error Monitoring and Protection

This amplifier is protected against Over-Current, Over-Voltage, Over-Temperature, and Under-Voltage conditions. The amplifier errors can be monitored using the TA command. This command will return a bit mask number representing specific error conditions. For more information, refer to the TA command in the Command Reference. The controller will also monitor for illegal hall states using the QH command. For more information, refer to the QH command in the Command Reference.
To handle amplifier error conditions, users can include the #AMPERR automatic subroutine in their DMC program. This subroutine is user defined. Two conditions must be met for the #AMPERR subroutine to run: the #AMPERR subroutine must be defined in the DMC program; and an amplifier error must be detected. When these conditions are met, the program will automatically interrupt thread 0 and jump to the #AMPERR subroutine.

Hall Error Protection

During normal operation, the controller should not have any Hall errors. Hall errors can be caused by a faulty Hall-effect sensor or a noisy environment. The state of the Hall inputs can also be monitored through the QH command. Hall errors will cause the amplifier to be disabled if OEm=**1** is set, and will cause the controller to enter the #AMPERR subroutine if it is included in a running program.

Over-Current Protection

The amplifier also has circuitry to protect against Over-Current. If the total current from a set of 2 axes (ie A and B or C and D) exceeds 20 A, the amplifier will be disabled. The amplifier can be re-enabled with the SH command when there is no longer an Over-Current condition.

WARNING	If this error occurs, it is indicative of a problem at the system level. An Over-Current error is usually due to a short across the motor leads or a short from a motor lead to ground.

Over-Voltage Protection

If the voltage supply to the amplifier rises above 94 VDC, then the amplifier will automatically disable. The amplifier will re-enable when the supply drops below 90 V.
The over voltage condition will not permanently shut down the amplifier and will not trigger the #AMPERR routine. The amplifier will be momentarily disabled; when the condition goes away, the amplifier will continue normal operation assuming it did not cause the position error to exceed the error limit.

WARNING	If this error occurs, it is indicative of a problem at the system level. An Over-Voltage error is usually due to regeneration from the motor and load. Consider using a shunt regulator if this occurs during normal operation.

Over-Temperature Protection

The amplifier is also equipped with over-temperature protection.
Rev A and Rev B amplifiers:
If the average heat sink temperature rises above 100°C, then the amplifier will be disabled. The over-temperature condition will trigger the #AMPERR routine if included in the program on the controller.
The amplifier will re-enable when the temperature drops below 100 °C.
Rev C and newer amplifiers:
If the average heat sink temperature rises above 80°C, then the amplifier will be disabled. The over-temperature condition will trigger the #AMPERR routine if included in the program on the controller.
The amplifier can be re-enabled with the SH command once the temperature drops below 80°C.
Rev C Amplifiers began shipping in December 2008.

WARNING	An Over-Temperature error is usually due to insufficient heat dissipation from the amplifier. Heat must be drawn away from the base of the controller with forced air flow or a heat sink.

Under-Voltage Protection

If the voltage supplied to the amplifier drops below its rated supply voltage, an Under-Voltage error will be reported and the amplifier will be disabled.
If an #AMPERR routine is defined and the controller is powered before the amplifier, then the #AMPERR routine will automatically be triggered. The amplifier will return to normal operation once the supply is raised above the rated supply voltage assuming it did not cause the position error to exceed the error limit.

Peak Current

If the commanded torque value (TT) on any axis equals the peak torque limit set by TK, a Peak Current condition will be reported for that axis, and the axis bit in the TA**2** bit mask will be set. Peak Current does not cause #AMPERR to run. The Peak Current condition reports live and is not latching.

ELO Input

If the ELO input on the controller is asserted, the power stage of the amplifier will be disabled. When the ELO input is triggered, TA**3** will change state and the #AMPERR routine will run on Thread 0. To recover from an ELO, the user must issue the MO command followed by the SH command. To recover from an ELO in a DMC program, use MO;WT**2**;SH.
It is recommended that OE**1** be used for all axes when the ELO is used in an application.
See the Optoisolated Input Electrical Information section in Chapter 3 Connecting Hardware for information on connecting the ELO input.

A2 – AMP-43140 (-D3140)

Description

The AMP-43140 resides inside the DMC-41x3 enclosure and contains four linear drives for operating small, brush-type servo motors. The AMP-43140 requires a ± 12-30 VDC input. Output power is 20 W per amplifier or 60 W total. The gain of each transconductance linear amplifier is 0.1 A/V at 1 A maximum current. The typical current loop bandwidth is 4 kHz. The AMP-43140 can be ordered to have a 100mA maximum current output where the gain of the amplifier is 10mA/V. Order as '-D3140(100mA)'.
GDK's Step-By-Step tool provides an interactive guide for configuring brushed motors.

WARNING	Do not "hot swap" the motor power or supply voltage power input connections. If the amplifier is enabled when the motor connector is connected or disconnected, damage to the amplifier can occur. Galil recommends powering the controller and amplifier down before changing the connector, and breaking the AC side of the power supply connection in order to power down the amplifier.

Electrical Specifications

	Servo (Brush)
Supply Voltage	±20-30 VDC (±12-30 available with ISCNTL)
Continuous Current	1.0 Amps (100mA option available)
Amplifier Gains	0.1 A/V (10mA A/V option available)
Power Output (per channel)	20 W
Total Max Power Output	60 W

Table A2.1: Amplifier Electrical Specifications

Mating Connectors

	Cable Connectors	Terminal Pins	On board Connector
POWER	4-pin Mini-Fit, Jr.™ (Male) Molex #0039012045	Molex #0444761112	4-pin Mini-Fit, Jr.™ (Female) Molex #0039310040
Motor Power Connectors	2-pin Mini-Fit, Jr.™ (Male) Molex #0039012025	Molex #0444761112	2-pin Mini-Fit, Jr.™ (Female) Molex #0039310020

Table A2.2: Molex part numbers for connectors and terminal pin

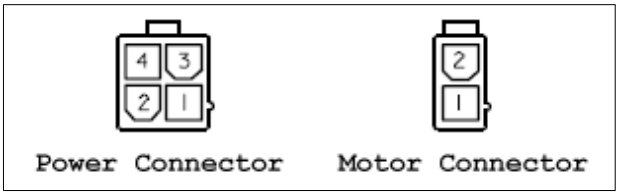


Figure A2.1: Power and Motor Connector Diagram

Power Connector	
Pin Number	**Connection**
1, 2	DC Power Supply Ground
3	-VS (-DC Power)
4	+VS (DC Power)
Motor Connector	
Pin Number	**Brushed**
1	Phase A-
2	Phase A+

Table A2.3: Power and Motor Connector Pin-outs

Servo Motor Operation

Using External Amplifiers

Use the D-Sub connectors on the top face of controller to access the necessary signals to control external amplifiers. For more information on connecting external amplifiers, see Step 2b. Connecting External Amplifiers and Motors in Chapter 2.

-SSR Option

The AMP-43140 linear amplifier require a bipolar power supply. It is possible that the plus and minus (V+ and V-) rise to nominal voltage at different rates during power up and any difference between voltage levels will be seen as an offset in the amplifier. This offset may cause a slight jump during power up prior to the controller establishing closed-loop control. When ordered with the –SSR option a solid state relay is added to the amplifier. This relay disconnects the amplifier power from the motor power leads when the controller is placed in the motor-off state. If the MO jumper is installed, or the MO command is burned into memory, the addition of the –SSR option will eliminate any jump due to the power supply.

Error Monitoring and Protection

Peak Current

If the commanded torque value (TT) on any axis equals the peak torque limit set by TK, a Peak Current condition will be reported for that axis, and the axis bit in the TA**2** bit mask will be set. Peak Current does not cause **#AMPERR** to run. The Peak Current condition reports live and is not latching.

ELO Input

If the ELO input on the controller is asserted, the power stage of the amplifier will be disabled. When the ELO input is triggered, TA**3** will change state and the **#AMPERR** routine will run on Thread 0. To recover from an ELO, the user must issue the MO command followed by the SH command. To recover from an ELO in a DMC program, use MO;WT**2**;SH.
It is recommended that OE**1** be used for all axes when the ELO is used in an application.
See the Optoisolated Input Electrical Information section in Chapter 3 Connecting Hardware for information on connecting the ELO input.

-SSR Option

The AMP-43140 linear amplifier require a bipolar power supply. It is possible that the plus and minus (V+ and V-) rise to nominal voltage at different rates during power up and any difference between voltage levels will be seen as an offset in the amplifier. This offset may cause a slight jump during power up prior to the controller establishing closed-loop control. When ordered with the –SSR option a solid state relay is added to the amplifier. This relay disconnects the amplifier power from the motor power leads when the controller is placed in the motor-off state. If the MO jumper is installed, or the MO command is burned into memory, the addition of the –SSR option will eliminate any jump due to the power supply.

A3 – AMP-43240 (-D3240)

Description

The AMP-43240 resides inside the DMC-41x3 enclosure and contains four trapezoidally commutated transconductance, PWM amplifiers for driving brushless or brush-type servo motors. Each amplifier drives motors operating at up to 10 Amps continuous, 20 Amps peak, 20–80 VDC. The gain settings of the amplifier are user-programmable at 0.5 Amp/Volt, 1.0 Amp/Volt and 2.0 Amp/Volt. The switching frequency is 24 kHz.
The drive operates in a Chopper Mode. The amplifier offers protection for Over-Voltage, under-voltage, Over-Current, short-circuit and over-temperature. Two AMP-43240s can be used in 5- through 8-axis controllers. A shunt regulator option is available. If higher voltages are required, please contact Galil.
GDK's Step-By-Step tool provides an interactive guide for configuring brushless and brushed motors.

WARNING	Do not “hot swap” the motor power or supply voltage power input connections. If the amplifier is enabled when the motor connector is connected or disconnected, damage to the amplifier can occur. Galil recommends powering the controller and amplifier down before changing the connector, and breaking the AC side of the power supply connection in order to power down the amplifier.

Electrical Specifications

	Servo (Brushless/Brush)
Supply Voltage	20-80 VDC
Continuous Current	10 Amps
Peak Current	20 Amps
Switching Frequency	24 kHz
Amplifier Gains	0.5, 1.0, 2.0 Amps/Volt
Brushless Commutation Angle	120° (60° option available)
PWM Output Operation	Chopper
Minimum Load Inductance	$L(mH) = \frac{Vs(V)}{192 * I_{Ripple}(A)}$

Vs = Supply Voltage, I_{ripple} = 10% of the maximum current at chosen gain

Table A3.1: Amplifier Electrical Specifications

Mating Connectors

	Cable Connectors	Terminal Pins	On board Connector
POWER	6-pin Mini-Fit, Jr.™ (Male) Molex #0039012065	Molex #0444761112	6-pin Mini-Fit, Jr.™ (Female) Molex #0039310060
A,B,C,D: 4-pin Motor Power Connectors	4-pin Mini-Fit, Jr.™ (Male) Molex #0039012045	Molex #0444761112	4-pin Mini-Fit, Jr.™ (Female) Molex #0039310040

Table A3.2: Molex part numbers for connectors and terminal pin

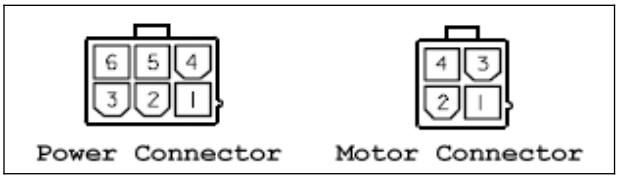


Figure A3.1: Power and Motor Connector Diagram

Power Connector		
Pin Number	**Connection**	
1,2,3	DC Power Supply Ground	
4,5,6	+VS (DC Power)	
Motor Connector		
Pin Number	**3-phase brushless**	**Brushed**
1	Phase C	No Connect
2	Phase B	Phase A-
3	No Connect	No Connect
4	Phase A	Phase A+

Table A3.3: Power and Motor Connector Pin-outs

Servo Motor Operation

Brushless Motor Setup

The AMP-43240 supports brushless motor operation. To configure an axis for a brushless motor, connect the 3 motor leads as prescribed in Table A3.3. Connect the encoders, homes, and limits as required. Trapezoidal commutation for brushless motors requires the motors have Hall-effect sensors. Wire the Hall-effect sensors to the dedicated inputs on the encoder connector. Once complete, use GDK's Step-By-Step tool for initializing commutation.

Brushed Motor Operation

The AMP-43240 also supports brushed motor operation. To configure an axis for a brushed motor, connect the 2 motor leads as prescribed in Table A3.3. Connect the encoders, homes, and limits as required. Then, use GDK's Step-By-Step tool for applying settings for brush-type operation.

Using External Amplifiers

The BR command must be set to a **1** for any axis where an AMP-43240 is installed but the use of an external axis is required. Doing so will this will disable the Hall Error Protection. Setting BR**m**=**1** (where **m** is the axis designator) is required for both external servo amplifiers and stepper drivers. In order to use the full torque limit, make sure the AG setting for the axes using external amplifiers are set to **0** or **1**.

Use D-Sub connectors on the top face of the controller to access necessary signals to run external amplifiers. For more information on connecting external amplifiers, see Step 2b. Connecting External Amplifiers and Motors in Chapter 2.

Setting Amplifier Gain and Current Loop Bandwidth

The AG command will set the amplifier gain (Amps/Volt). The AU command will set the current loop gain.

AG Command:

The AMP-43240 has 3 amplifier gain settings. The gain is set with the AG command. When configuring the amplifier gain, the axis must be in a motor off (MO) state prior to execution of the AG command. For more information on the AG command, refer to the command reference for the DMC-41x3

AG setting	Gain Value
n = 0	0.5 A/V
n = 1	1.0 A/V
n = 2	2.0 A/V

Table A3.4: Amplifier Gain Settings

AU Command:

Proper configuration of the AU command is essential to optimize the operation of the AMP-43240. This command sets the gain for the current loop on the amplifier. Table A3.5 indicates the recommended AU settings for 24 and 48 VDC power supplies based on motor inductance. When configuring the current loop gain, the axis must be in a motor off (MO) state prior to execution of the AU command. For more information on the AU command, refer to the command reference for the DMC-41x3.

Vsupply VDC	Inductance L (mH)	n
24	0.5 ≤ L < 5.0	0
24	5.0 ≤ L	1
48	0.5 ≤ L < 10.0	0
48	10.0 ≤ L	1

Table A3.5: Amplifier Current Loop Gain Settings

Chopper Mode

The AMP-43240 runs in what is called a “Chopper” mode. The chopper mode is in contrast to the normal inverter mode (AMP-43040) in which the amplifier sends PWM power to the motor of ±VS. In chopper mode, the amplifier sends a 0 to +VS PWM to the motor when moving in the forward direction, and a 0 to –VS PWM to the motor when moving in the negative direction.

Setting Peak and Continuous Torque Limits

TK and TL Commands:

The peak and continuous torque limits can be set through the TK and TL commands respectively. The controller will command the torque signal (TT) up to the value specified by TK for a maximum of one second, as shown in Figure A3.2. It will then limit the torque to the value specified by TL. The more time the controller commands the torque below TL, the longer it will command it to TK when necessary. The DMC code example below exhibits the behavior of TT being limited by TL and TK, as shown in Figure A3.2.

```
#tk;          'begin program
MO;           'disable all axes
AGA=2;        'set max amplifier gain of 2.0 A/V
TLA=5;        'set continuous torque limit of 5V
TKA=9.9982;   'set max peak torque limit to 9.9982V
SHA;          'enable A axis
OFA=9.9982;   'command torque signal to 9.9982V
EN;           'end program
```

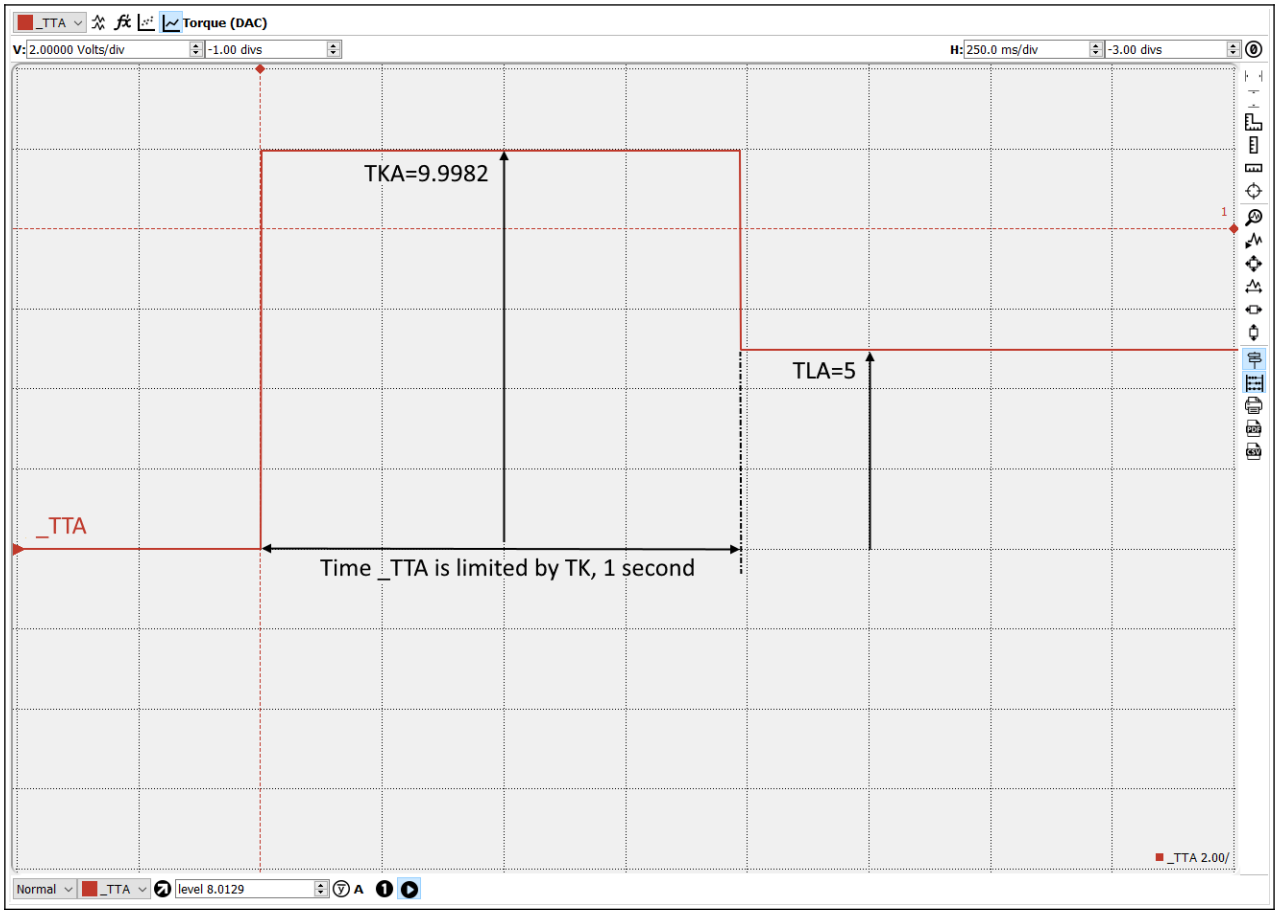


Figure A3.2: Peak Current Operation

Error Monitoring and Protection

This amplifier is protected against Over-Current, Over-Voltage, Over-Temperature, and Under-Voltage conditions. The amplifier errors can be monitored using the TA command. This command will return a bit mask number representing specific error conditions. For more information, refer to the TA command in the Command Reference. The controller will also monitor for illegal hall states using the QH command. For more information, refer to the QH command in the Command Reference.
To handle amplifier error conditions, users can include the **#AMPERR** automatic subroutine in their DMC program. This subroutine is user defined. Two conditions must be met for the **#AMPERR** subroutine to run: the **#AMPERR** subroutine must be defined in the DMC program; and an amplifier error must be detected. When these conditions are met, the program will automatically interrupt thread 0 and jump to the **#AMPERR** subroutine.

Hall Error Protection

During normal operation, the controller should not have any Hall errors. Hall errors can be caused by a faulty Hall-effect sensor or a noisy environment. The state of the Hall inputs can also be monitored through the QH command. Hall errors will cause the amplifier to be disabled if OEm=**1** is set, and will cause the controller to enter the **#AMPERR** subroutine if it is included in a running program.

Over-Current Protection

The amplifier also has circuitry to protect against Over-Current. If the total current from a set of 2 axes (ie A and B or C and D) exceeds 40 A, the amplifier will be disabled. The amplifier can be re-enabled with the SH command when there is no longer an Over-Current condition.

WARNING	If this error occurs, it is indicative of a problem at the system level. An Over-Current error is usually due to a short across the motor leads or a short from a motor lead to ground.

Over-Voltage Protection

If the voltage supply to the amplifier rises above 94 VDC, then the amplifier will automatically disable. The amplifier will re-enable when the supply drops below 90 V.
The over voltage condition will not permanently shut down the amplifier and will not trigger the **#AMPERR** routine. The amplifier will be momentarily disabled; when the condition goes away, the amplifier will continue normal operation assuming it did not cause the position error to exceed the error limit.

WARNING	If this error occurs, it is indicative of a problem at the system level. An Over-Voltage error is usually due to regeneration from the motor and load. Consider using a shunt regulator if this occurs during normal operation.

Over-Temperature Protection

The amplifier is also equipped with over-temperature protection.
If the average heat sink temperature rises above 80°C, then the amplifier will be disabled. The over-temperature condition will trigger the **#AMPERR** routine if included in the program on the controller.
The amplifier can be re-enabled with the SH command once the temperature drops below 80°C.

WARNING	An Over-Temperature error is usually due to insufficient heat dissipation from the amplifier. Heat must be drawn away from the base of the controller with forced air flow or a heat sink.

Under-Voltage Protection

If the voltage supplied to the amplifier drops below its rated supply voltage, an Under-Voltage error will be reported and the amplifier will be disabled.
If an **#AMPERR** routine is defined and the controller is powered before the amplifier, then the **#AMPERR** routine will automatically be triggered. The amplifier will return to normal operation once the supply is raised above the rated supply voltage assuming it did not cause the position error to exceed the error limit.

Peak Current

If the commanded torque value (TT) on any axis equals the peak torque limit set by TK, a Peak Current condition will be reported for that axis, and the axis bit in the TA**2** bit mask will be set. Peak Current does not cause **#AMPERR** to run. The Peak Current condition reports live and is not latching.

ELO Input

If the ELO input on the controller is asserted, the power stage of the amplifier will be disabled. When the ELO input is triggered, TA**3** will change state and the **#AMPERR** routine will run on Thread 0. To recover from an ELO, the user must issue the MO command followed by the SH command. To recover from an ELO in a DMC program, use MO;WT**2**;SH.
It is recommended that OE**1** be used for all axes when the ELO is used in an application.
See the Optoisolated Input Electrical Information section in Chapter 3 Connecting Hardware for information on connecting the ELO input.

A4 – AMP-43540 (-D3540, -D3520)

Description

The AMP-43540 is a sinusoidally commutated, 16 bit PWM amplifier. It can also be ordered in a two axis configuration specified as -D3520. The AMP-43540 can be configured for the motors below with just a few commands.

- 3-phase Brushless Motors
- Brushed Motors

GDK's Step-By-Step tool provides an interactive guide for configuring brushless and brushed motors.

WARNING	Do not “hot swap” the motor power or supply voltage power input connections. If the amplifier is enabled when the motor connector is connected or disconnected, damage to the amplifier can occur. Galil recommends powering the controller and amplifier down before changing the connector, and breaking the AC side of the power supply connection in order to power down the amplifier.

Electrical Specifications

	Servo (Brushless/Brushed)
Supply Voltage	20-80 V
Continuous Current	8 Amps
Peak Current	15 Amps
Switching Frequency	33 kHz
Amplifier Gains	0.4, 0.8, 1.6 Amps/Volt
Minimum Load Inductance	$L(mH) = \frac{Vs(V)}{264 * I_{Ripple}(A)}$

Vs = Supply Voltage, I_{ripple} = 10% of the maximum current at chosen gain

Table A4.1: Amplifier Electrical Specifications

Mating Connectors

	Cable Connectors	Terminal Pins	On board Connector
POWER	6-pin Mini-Fit, Jr.™ (Male) Molex #0039012065	Molex #0444761112	6-pin Mini-Fit, Jr.™ (Female) Molex #0039310060
A,B,C,D: 4-pin Motor Power Connectors	4-pin Mini-Fit, Jr.™ (Male) Molex #0039012045	Molex #0444761112	4-pin Mini-Fit, Jr.™ (Female) Molex #0039310040

Table A4.2: Molex part numbers for connectors and terminal pin

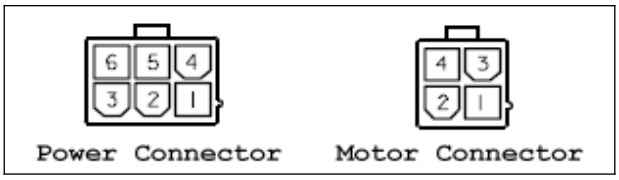


Figure A4.1: Power and Motor Connector Diagram

Power Connector		
Pin Number	**Connection**	
1,2,3	DC Power Supply Ground	
4,5,6	+VS (DC Power)	
Motor Connector		
Pin Number	**3-phase brushless**	**Brushed**
1	Phase C	Phase A-
2	Phase B	No Connect
3	No Connect	No Connect
4	Phase A	Phase A+

Table A4.3: Power and Motor Connector Pin-outs

Servo Motor Operation

Brushless Motor Operation

For Brushless motor operation, MT must be set to **1** for the axis. The 6 commands used for setup are the BA, BM, BX, BZ, BC and BI. Please refer to the Command Reference for details. Below is the setup procedure for configuring an axis to operate a Brushless motor.

1. Issue the BA command to specify on which axis to configure sinusoidal commutation.
2. The brushless modulus is the number of encoder counts per magnetic cycle and is set with the BM command. For example, in a rotary motor that has 2 pole pairs and 10000 counts per revolution, the number of encoder counts per magnetic cycle would be 10000/2. BM should be assigned to a value of **5000**.
3. Issue either the BI and BC, BZ, or BX command to initialize the motor for sinusoidal commutation. The command must be executed on every reset or power-up of the controller.

Setup Commands

1. BI/BC Commands – If the motor includes hall sensors, Galil recommends using the BI and BC commands. The BI command uses the hall sensors to set the approximate location of the motor in the magnetic cycle. The BC command will precisely set the commutation angle upon detecting a hall state transition.
2. BZ Command – If the motor does not include hall sensors, Galil recommends using the BZ command. The BZ command forces the motor to a known location in the magnetic cycle. Note that this can cause up to ½ a magnetic cycle of motion.
3. BX Command – If the motor does not have hall sensors and the BZ command generates too much motion for the application, Galil recommends using the BX command. This command utilizes a minimal movement algorithm in order to determine the correct commutation of the motor.

Brushed Motor Operation

For brushed motor operation, set MT and BR to **1** for the axis.

Using External Amplifiers

The BR command must be set to a **-1** for any axis where an AMP-43540 is installed but the use of an external axis is required. This setting will disable the requirement to have the BA, BM and BX or BZ commands executed prior to being able to issue the SH command for that axis. Setting BR*m*=**-1** (where *m* is the axis designator) is required for both external servo amplifiers and stepper drivers. In order to use the full torque limit, make sure the AG setting for the axes using external amplifiers are set to **0** or **1**.

Use D-Sub connectors on the top face of the controller to access necessary signals to run external amplifiers. For more information on connecting external amplifiers, see Step 2b. Connecting External Amplifiers and Motors in Chapter 2.

Setting Amplifier Gain and Current Loop Gain

The AG command will set the amplifier gain (Amps/Volt). The AU command will set the current loop gain.

AG Command:

The AMP-43540 has 3 amplifier gain settings. The gain is set with the AG command. When configuring the amplifier gain, the axis must be in a motor off (MO) state prior to execution of the AG command. For more information on the AG command, refer to the command reference for the DMC-41x3.

AG setting	Gain Value
n = 0	0.4 A/V
n = 1	0.8 A/V
n = 2	1.6 A/V

Table A4.4: Amplifier Gain Settings

AU Command:

Proper configuration of the AU command is essential to optimize the operation of the AMP-43540. This command sets the gain for the current loop on the amplifier. Table A4.5 indicates the recommended AU settings for 24 and 48 VDC power supplies based on motor inductance. When configuring the current loop gain, the axis must be in a motor off (MO) state prior to execution of the AU command. For more information on the AU command, refer to the command reference for the DMC-41x3.

Vsupply VDC	Inductance L (mH)	n
24	L < 1	9
24	1 ≤ L < 2.3	10
24	2.3 ≤ L < 4.2	11
24	4.2 ≤ L	12
48	L < 2.4	9
48	2.4 ≤ L < 4.2	10
48	4.2 ≤ L < 7	11
48	7 ≤ L	12

Table A4.5: Amplifier Current Loop Gain Settings

Setting Peak and Continuous Torque Limits

TK and TL Commands:

The peak and continuous torque limits can be set through the TK and TL commands respectively. The controller will command the torque signal (TT) up to the value specified by TK for a maximum of one second, as shown in Figure A4.2. It will then limit the torque to the value specified by TL. The more time the controller commands the torque below TL, the longer it will command it to TK when necessary. The DMC code example below exhibits the behavior of TT being limited by TL and TK, as shown in Figure A4.2.

```
#tk;          'begin program
MO;           'disable all axes
AGA=2;        'set max amplifier gain of 1.6 A/V
TLA=5;        'set continuous torque limit of 5V
TKA=9.9982;   'set max peak torque limit to 9.9982V
SHA;          'enable A axis
OFA=9.9982;   'command torque signal to 9.9982V
EN;           'end program
```

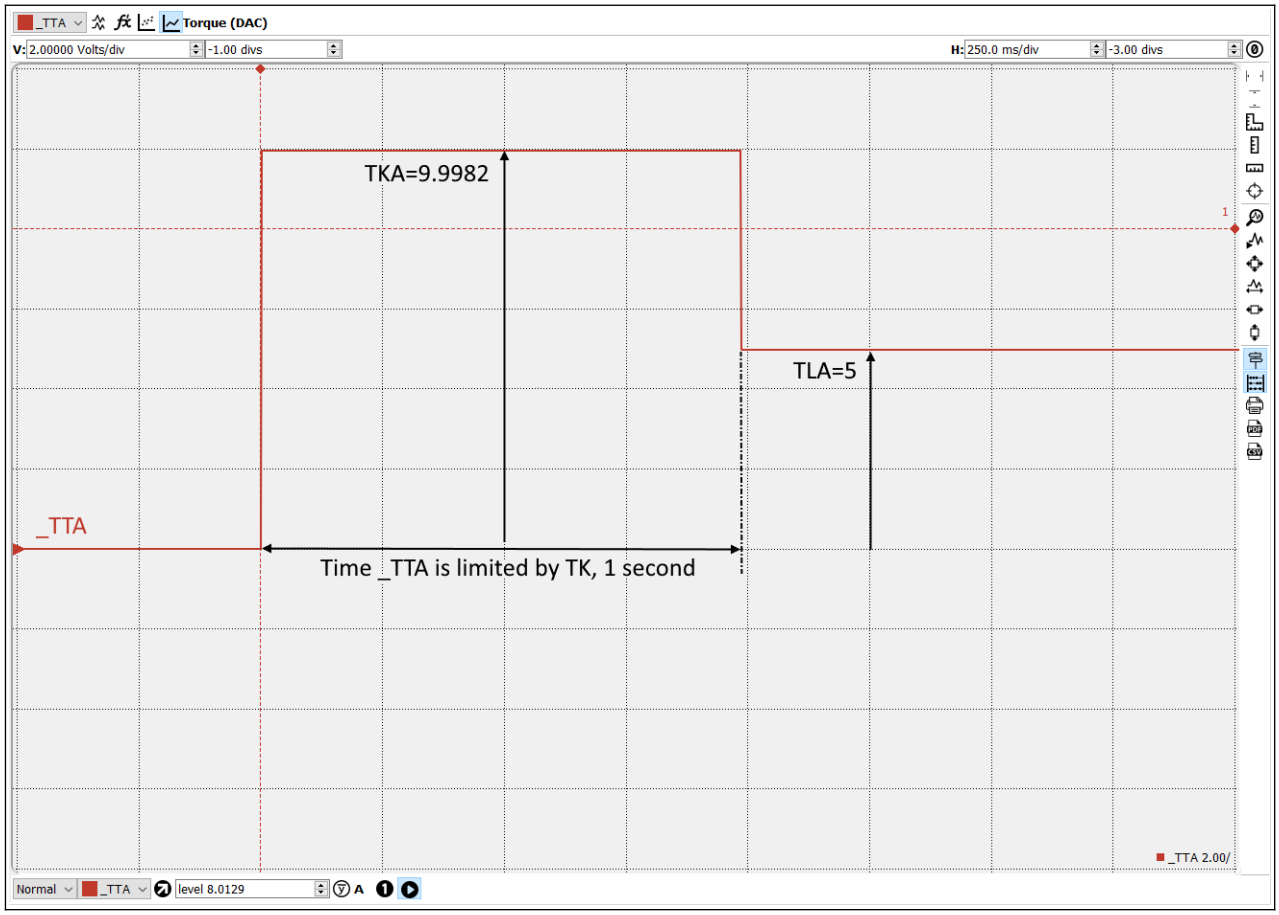


Figure A4.2: Peak Current Operation

Error Monitoring and Protection

This amplifier is protected against Over-Current, Over-Voltage, Over-Temperature, and Under-Voltage conditions. The amplifier errors can be monitored using the TA command. This command will return a bit mask number representing specific error conditions. It will also report whether or not an amplifier error is latched. For more information, refer to the TA command in the Command Reference. Latched errors can be cleared with the AZ command. For more details on clearing latched errors, refer to the Clearing Latched Amplifier Errors section.
To handle amplifier error conditions, users can include the **#AMPERR** automatic subroutine in their DMC program. This subroutine is user defined. Two conditions must be met for the **#AMPERR** subroutine to run: the **#AMPERR** subroutine must be defined in the DMC program; and an amplifier error must be detected. When these conditions are met, the program will automatically interrupt thread 0 and jump to the **#AMPERR** subroutine.

Over-Current Protection

If there is a short circuit in the amplifier or motor, an Over-Current error will be reported and the amplifier will be disabled. The Over-Current amplifier error is a latching error, see the Clearing Latched Amplifier Errors section for details.

WARNING	If this error occurs, it is indicative of a problem at the system level. An Over-Current error is usually due to a short across the motor leads or a short from a motor lead to ground.

Over-Voltage Protection

If the voltage supplied to the amplifier rises above its rated supply voltage, an Over-Voltage error will be reported. While the amplifier is experiencing this condition, current is no longer commanded and the motor phases are shorted together. This results in a drag on the motors causing them to coast to a stop. The amplifier will resume commanding current when the voltage drops below the maximum rated supply voltage. The Over-Voltage amplifier error is *not* a latching error.

WARNING	If this error occurs, it is indicative of a problem at the system level. An Over-Voltage error is usually due to regeneration from the motor and load. Consider using a shunt regulator if this occurs during normal operation.

Over-Temperature Protection

If the average amplifier temperature rises above 80°C, an Over-Temperature error will be reported and the amplifier will be disabled. The error cannot be cleared until the temperature drops below 80°C. The Over-Temperature amplifier error is a latching error, see the Clearing Latched Amplifier Errors section for details.

WARNING	An Over-Temperature error is usually due to insufficient heat dissipation from the amplifier. Heat must be drawn away from the base of the controller with forced air flow or a heat sink.

Under-Voltage Protection

If the voltage supplied to the amplifier drops below its rated supply voltage, an Under-Voltage error will be reported and the amplifier will be disabled. The error cannot be cleared until the supply voltage raises above its minimum rated supply voltage. Under-Voltage does not latch by default. It will become a latching error after AZ**2** is issued to the controller. See the Clearing Latched Amplifier Errors section for details.

Peak Current

If the commanded torque value (TT) on any axis equals the peak torque limit set by TK, a Peak Current condition will be reported for that axis, and the axis bit in the TA**2** bit mask will be set. Peak Current does not cause **#AMPERR** to run. The Peak Current condition reports live and is not latching.

ELO Input

If the ELO input on the controller is asserted, the power stage of the amplifier will be disabled. The ELO input is a latching input, see the Clearing Latched Amplifier Errors section for details. Reference the Optoisolated Input Electrical Information section in Chapter 3 Connecting Hardware for information on connecting the ELO input.

Clearing Latched Amplifier Errors

Default Error Clearing

Amplifier errors that do not latch will be reported in real-time by the TA command.
For latching errors to report and latch for a bank of axes, at least one axis in that bank must be in a servo here state (SH).
To clear latched amplifier errors, users must issue the MO command followed by the SH command. To clear latched amplifier errors in a DMC program, use MO**;**WT**2;**SH. If an axis is moving, motion must be halted before attempting to clear amplifier errors.

Enhanced Error Clearing

Enhanced error clearing and reporting can be enabled by issuing AZ**2** to the controller.
While all axes are in a motor off state (MO) and there are no latched amplifier errors, all amplifier errors will be reported in real-time by the TA command. For an amplifier error to latch for a bank of axes, at least one axis in that bank must be in a servo here state (SH).
To clear latched amplifier errors, issue the MO command followed by AZ**1**. Use the TA command to report if an amplifier error condition is still present. If an axis is moving, motion must be halted before attempting to clear amplifier errors. See the **#AMPERR** label and AZ command in the Command Reference for more information and examples.
NOTE: Enhanced Error Clearing is supported on the DMC-41x3 as of firmware revision 1.3e and amplifier hardware revision 4.

A5 – AMP-43547 (-D3547, -D3527)

Description

The AMP-43547 is a sinusoidally commutated, 16 bit PWM amplifier that can also be configured as a microstepping drive. It can also be ordered in a two axis configuration specified as -D3527. This amplifier is unique in that it can run either servo or stepper motors on any axis. The AMP-43547 can be configured for all of the motors below with just a few commands.

- 3-phase Brushless Motors
- Brushed Motors
- Stepper Motors
- 2-phase Brushless Motors

GDK's Step-By-Step tool provides an interactive guide for configuring brushless, brushed, and stepper motors.

WARNING	Do not “hot swap” the motor power or supply voltage power input connections. If the amplifier is enabled when the motor connector is connected or disconnected, damage to the amplifier can occur. Galil recommends powering the controller and amplifier down before changing the connector, and breaking the AC side of the power supply connection in order to power down the amplifier.

Electrical Specifications

	Servo (Brushless/Brushed)	Stepper
Supply Voltage	20-80 V	
Continuous Current	8 A	-
Peak Current	15 A	-
Maximum Current per Phase	-	6 A
Step Resolution	-	256 microsteps/full step
Switching Frequency	33 kHz	
Amplifier Gains	0.4, 0.8, 1.6 Amps/Volt	0.75, 1.5, 3, 6 A
Minimum Load Inductance	$L(mH) = \frac{Vs(V)}{264 * I_{Ripple}(A)}$	

Vs = Supply Voltage, I_{ripple} = 10% of the maximum current at chosen gain

Table A5.1: Amplifier Electrical Specifications

Mating Connectors

	Cable Connectors	Terminal Pins	On board Connector
POWER	6-pin Mini-Fit, Jr.™ (Male) Molex #0039012065	Molex #0444761112	6-pin Mini-Fit, Jr.™ (Female) Molex #0039310060
A,B,C,D: 4-pin Motor Power Connectors	4-pin Mini-Fit, Jr.™ (Male) Molex #0039012045	Molex #0444761112	4-pin Mini-Fit, Jr.™ (Female) Molex #0039310040

Table A5.2: Molex part numbers for connectors and terminal pin

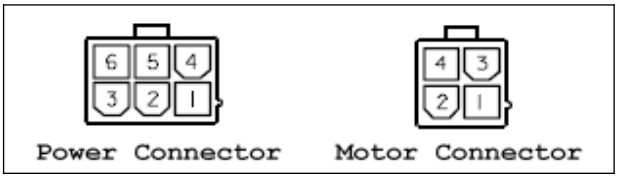


Figure A5.1: Power and Motor Connector Diagram

Power Connector			
Pin Number	**Connection**		
1,2,3	DC Power Supply Ground		
4,5,6	+VS (DC Power)		
Motor Connector			
Pin Number	**3-phase brushless**	**Brushed**	**2-phase brushless or Stepper**
1	Phase C	Phase A-	Phase B-
2	Phase B	No Connect	Phase B+
3	No Connect	No Connect	Phase A-
4	Phase A	Phase A+	Phase A+

Table A5.3: Power and Motor Connector Pin-outs

Servo Motor Operation

Brushless Motor Operation

For Brushless motor operation, MT must be set to **1** for the axis. The 6 commands used for setup are the BA, BM, BX, BZ, BC and BI. Please refer to the Command Reference for details. Below is the setup procedure for configuring an axis to operate a Brushless motor.

1. Issue the BA command to specify on which axis to configure sinusoidal commutation.
2. The brushless modulus is the number of encoder counts per magnetic cycle and is set with the BM command. For example, in a rotary motor that has 2 pole pairs and 10000 counts per revolution, the number of encoder counts per magnetic cycle would be 10000/2. BM should be assigned to a value of **5000**.
3. Issue either the BI and BC, BZ, or BX command to initialize the motor for sinusoidal commutation. The command must be executed on every reset or power-up of the controller.

Setup Commands

1. BI/BC Commands – If the motor includes hall sensors, Galil recommends using the BI and BC commands. The BI command uses the hall sensors to set the approximate location of the motor in the magnetic cycle. The BC command will precisely set the commutation angle upon detecting a hall state transition.
2. BZ Command – If the motor does not include hall sensors, Galil recommends using the BZ command. The BZ command forces the motor to a known location in the magnetic cycle. Note that this can cause up to ½ a magnetic cycle of motion.
3. BX Command – If the motor does not have hall sensors and the BZ command generates too much motion for the application, Galil recommends using the BX command. This command utilizes a minimal movement algorithm in order to determine the correct commutation of the motor.

2PB Motor Operation

For 2PB motor operation, MT must be set to **4** for the axis. Initialization and setup of a 2PB motor is similar to the Brushless setup described above. The BZ command must be used to initialize commutation.

Minimum Encoder Resolution

Due to the density of the magnetic cycle in 2PB motors, operation requires the use of an encoder with a high enough resolution resulting in a minimum BM value of **80** encoder counts per magnetic cycle. Contact a Galil Applications Engineer to review minimum encoder resolution requirements.

Brushed Motor Operation

For brushed motor operation, set MT and BR to **1** for the axis.

Setting Amplifier Gain and Current Loop Gain

The AG command will set the amplifier gain (Amps/Volt). The AU command will set the current loop gain.

AG Command:

The AMP-43547 has 3 amplifier gain settings when configured for servo motors. The gain is set with the AG command. When configuring the amplifier gain, the axis must be in a motor off (MO) state prior to execution of the AG command. For more information on the AG command, refer to the command reference for the DMC-41x3.

AG setting	Gain Value
n = 0	0.4 A/V
n = 1	0.8 A/V
n = 2	1.6 A/V

Table A5.4: Amplifier Gain Settings

AU Command:

Proper configuration of the AU command is essential to optimize the operation of the AMP-43547. This command sets the gain for the current loop on the amplifier. Table A5.5 indicates the recommended AU settings for 24 and 48 VDC power supplies based on motor inductance. When configuring the current loop gain, the axis must be in a motor off (MO) state prior to execution of the AU command. For more information on the AU command, refer to the command reference for the DMC-41x3.

Vsupply VDC	Inductance L (mH)	n
24	L < 1	9
24	1 ≤ L < 2.3	10
24	2.3 ≤ L < 4.2	11
24	4.2 ≤ L	12
48	L < 2.4	9
48	2.4 ≤ L < 4.2	10
48	4.2 ≤ L < 7	11
48	7 ≤ L	12

Table A5.5: Amplifier Current Loop Gain Settings

Setting Peak and Continuous Torque Limits

TK and TL Commands:

The peak and continuous torque limits can be set through the TK and TL commands respectively. The controller will command the torque signal (TT) up to the value specified by TK for a maximum of one second, as shown in Figure A5.2. It will then limit the torque to the value specified by TL. The more time the controller commands the torque below TL, the longer it will command it to TK when necessary. The DMC code example below exhibits the behavior of TT being limited by TL and TK, as shown in Figure A5.2.

```
#tk;          'begin program
MO;           'disable all axes
AGA=2;        'set max amplifier gain of 1.6 A/V
TLA=5;        'set continuous torque limit of 5V
TKA=9.9982;   'set max peak torque limit to 9.9982V
SHA;          'enable A axis
OFA=9.9982;   'command torque signal to 9.9982V
EN;           'end program
```

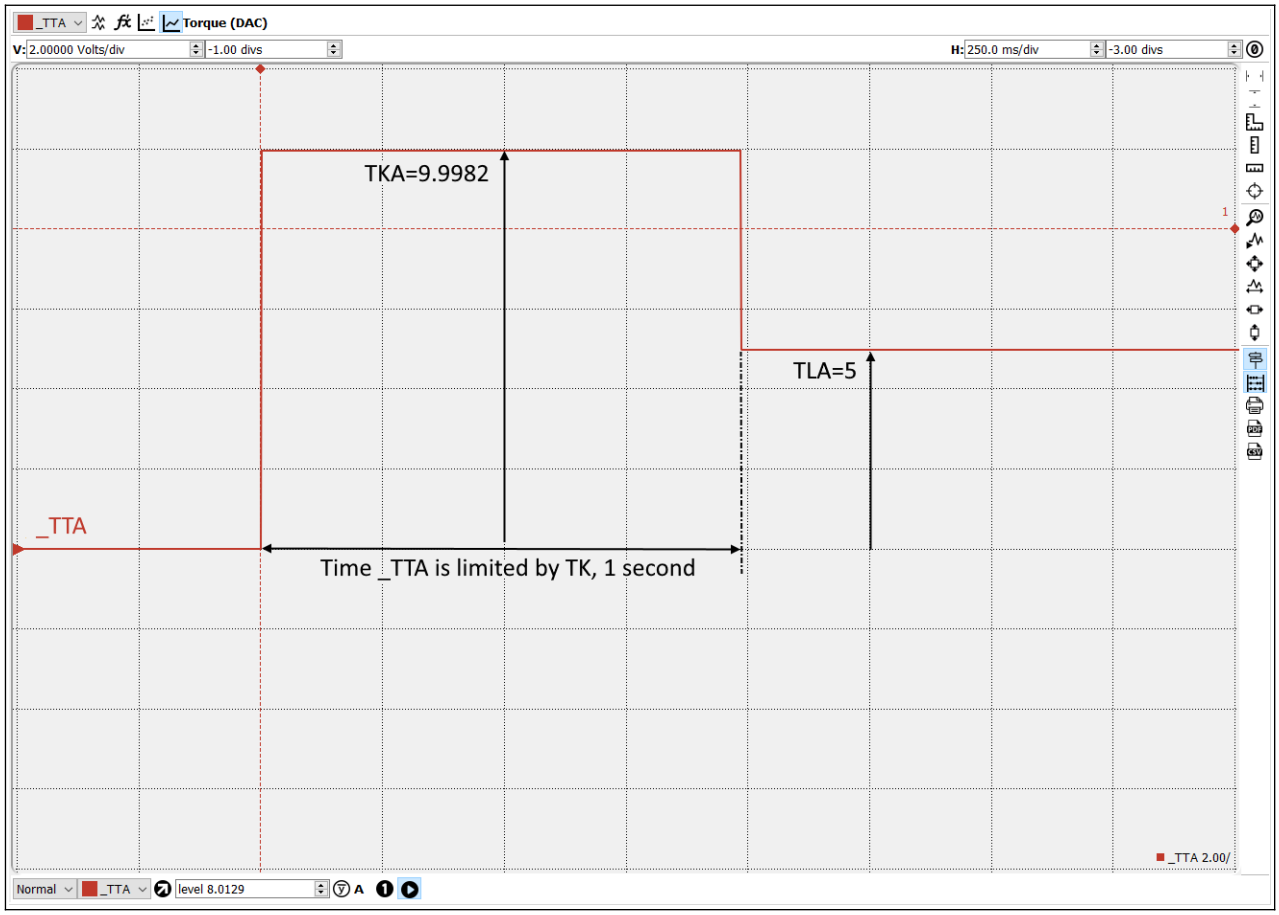


Figure A5.2: Peak Current Operation

Stepper Motor Operation

To configure an axis for a bipolar stepper motor, set MT to **-2** for that axis.

Setting Amplifier Gain and Current Loop Gain

The AG command will set the amplifier gain (Amps per phase). The AU command will set the current loop gain. Refer to Table A5.5 for more information.

AG Command:

The AMP-43547 has 4 amplifier gain settings when configured for stepper motors. The gain is set with the AG command. When configuring the amplifier gain, the axis must be in a motor off (MO) state prior to execution of the AG command. For more information on the AG command, refer to the command reference for the DMC-41x3.

AG setting	Gain Value
n = 0	0.75 A
n = 1	1.5 A
n = 2	3 A
n = 3	6 A

Table A5.6: Amplifier Gain Settings

AU Command:

Proper configuration of the AU command is essential to optimize the operation of the AMP-43547. This command sets the gain for the current loop on the amplifier. Table A5.5 indicates the recommended AU settings for 24 and 48 VDC power supplies based on motor inductance. When configuring the current loop gain, the axis must be in a motor off (MO) state prior to execution of the AU command. For more information on the AU command, refer to the command reference for the DMC-41x3.

Low Current Mode

By default, 100% of the current set by the AG command will be delivered to the motor while the axis is holding position. An axis can be configured for Low Current mode so 25% of the current will be sent to the motor while holding position. Low Current mode can also be configured to wait a specified number of samples after a move is completed before lowering the current to 0%. For more information, refer to the LC command in the command reference for the DMC-41x3.

LC Setting	Description
n = 0	100% holding current
n > 1	100% current n samples after move finishes before going to 0% holding current
n = 1	25% holding current immediately after move finishes current
n < 0	100% current n samples after move finishes before going to 25% holding current

Table A5.7: Low Current Mode Configurations

Error Monitoring and Protection

This amplifier is protected against Over-Current, Over-Voltage, Over-Temperature, and Under-Voltage conditions. The amplifier errors can be monitored using the TA command. This command will return a bit mask number representing specific error conditions. It will also report whether or not an amplifier error is latched. For more information, refer to the TA command in the Command Reference. Latched errors can be cleared with the AZ command. For more details on clearing latched errors, refer to the Clearing Latched Amplifier Errors section.
To handle amplifier error conditions, users can include the **#AMPERR** automatic subroutine in their DMC program. This subroutine is user defined. Two conditions must be met for the **#AMPERR** subroutine to run: the **#AMPERR** subroutine must be defined in the DMC program; and an amplifier error must be detected. When these conditions are met, the program will automatically interrupt thread 0 and jump to the **#AMPERR** subroutine.

Over-Current Protection

If there is a short circuit in the amplifier or motor, an Over-Current error will be reported and the amplifier will be disabled. The Over-Current amplifier error is a latching error, see the Clearing Latched Amplifier Errors section for details.

WARNING	If this error occurs, it is indicative of a problem at the system level. An Over-Current error is usually due to a short across the motor leads or a short from a motor lead to ground.

Over-Voltage Protection

If the voltage supplied to the amplifier rises above its rated supply voltage, an Over-Voltage error will be reported. While the amplifier is experiencing this condition, current is no longer commanded and the motor phases are shorted together. This results in a drag on the motors causing them to coast to a stop. The amplifier will resume commanding current when the voltage drops below the maximum rated supply voltage. The Over-Voltage amplifier error is *not* a latching error.

WARNING	If this error occurs, it is indicative of a problem at the system level. An Over-Voltage error is usually due to regeneration from the motor and load. Consider using a shunt regulator if this occurs during normal operation.

Over-Temperature Protection

If the average amplifier temperature rises above 80°C, an Over-Temperature error will be reported and the amplifier will be disabled. The error cannot be cleared until the temperature drops below 80°C. The Over-Temperature amplifier error is a latching error, see the Clearing Latched Amplifier Errors section for details.

WARNING	An Over-Temperature error is usually due to insufficient heat dissipation from the amplifier. Heat must be drawn away from the base of the controller with forced air flow or a heat sink.

Under-Voltage Protection

If the voltage supplied to the amplifier drops below its rated supply voltage, an Under-Voltage error will be reported and the amplifier will be disabled. The error cannot be cleared until the supply voltage raises above its minimum rated supply voltage. The Under-Voltage amplifier error is a latching error, see the Clearing Latched Amplifier Errors section for details.
Note: If there is an #AMPERR routine and the controller is powered before the amplifier, then the #AMPERR routine will automatically be triggered.

Peak Current

If the commanded torque value (TT) on any axis equals the peak torque limit set by TK, a Peak Current condition will be reported for that axis, and the axis bit in the TA**2** bit mask will be set. Peak Current does not cause **#AMPERR** to run. The Peak Current condition reports live and is not latching.

ELO Input

If the ELO input on the controller is asserted, the power stage of the amplifier will be disabled. The ELO input is a latching input, see the Clearing Latched Amplifier Errors section for details. Reference the Optoisolated Input Electrical Information section in Chapter 3 Connecting Hardware for information on connecting the ELO input.

Clearing Latched Amplifier Errors

While all axes are in a motor off state (MO) and there are no latched amplifier errors, all amplifier errors will be reported in real-time by the TA command. For an amplifier error to latch for a bank of axes, at least one axis in that bank must be in a servo here state (SH).
To clear latched amplifier errors, issue the MO command followed by AZ**1**. Use the TA command to report if an amplifier error condition is still present. If an axis is moving, motion must be halted before attempting to clear amplifier errors. See the **#AMPERR** label and AZ command in the Command Reference for more information and examples.
NOTE: Enhanced Error Clearing is supported on the DMC-41x3 as of firmware revision 1.3e and later.

A6 – AMP-43640 (-D3640)

Description

The AMP-43640 contains four linear drives for sinusoidally commutating brushless motors. The AMP-43640 requires a single 15–40VDC input. Output power delivered is typically 20 W per amplifier or 80 W total. The gain of each transconductance linear amplifier is 0.2 A/V. When used with a 24VDC power supply, the amplifier will deliver 1A continuous and 2A peak. The current loop bandwidth is approximately 4 kHz. The amplifier will use 12 bit DACs. There is an option for 16 bit DACs to increase the current resolution for systems with high feedback gain.
GDK's Step-By-Step tool provides an interactive guide for configuring brushless motors

WARNING	Do not “hot swap” the motor power or supply voltage power input connections. If the amplifier is enabled when the motor connector is connected or disconnected, damage to the amplifier can occur. Galil recommends powering the controller and amplifier down before changing the connector, and breaking the AC side of the power supply connection in order to power down the amplifier.

Electrical Specifications

	Servo (Brushless/Brushed)
Supply Voltage	15-40 VDC (15-20 VDC requires ISCNTL option)
Continuous Current	1.0 Amps
Peak Current	2.0 Amps
Amplifier Gains	0.2 Amps/Volt
Power Output (per channel)	20 W
Total Max Power Output	80 W

Table A6.1: Amplifier Electrical Specifications

Mating Connectors

	Cable Connectors	Terminal Pins	On board Connector
POWER	6-pin Mini-Fit, Jr.™ (Male) Molex #0039012065	Molex #0444761112	6-pin Mini-Fit, Jr.™ (Female) Molex #0039310060
A,B,C,D: 4-pin Motor Power Connectors	4-pin Mini-Fit, Jr.™ (Male) Molex #0039012045	Molex #0444761112	4-pin Mini-Fit, Jr.™ (Female) Molex #0039310040

Table A6.2: Molex part numbers for connectors and terminal pin

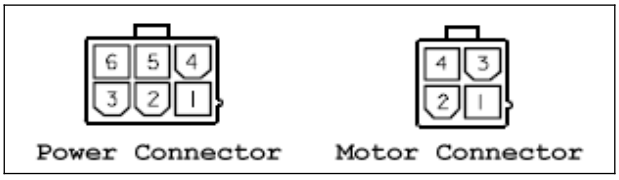


Figure A6.1: Power and Motor Connector Diagram

Power Connector	
Pin Number	**Connection**
1,2,3	DC Power Supply Ground
4,5,6	+VS (DC Power)
Motor Connector	
Pin Number	**Brushless**
1	Phase C
2	Phase B
3	No Connect
4	Phase A

Table A6.3: Power and Motor Connector Pin-outs

Power

Unlike a switching amplifier a linear amplifier does not have a straightforward relationship between the power delivered to the motor and the power lost in the amplifier. Therefore, determining the available power to the motor is dependent on the supply voltage, the characteristics of the load motor, and the required velocity and current.

All of the power delivered by the power supply is either used in the motor or lost in the amplifier.

Power of Power Supply $P_{ps} = P_m + P_A$

The power to the motor is both the power used to provide motion and the power lost to heat.

Power of the motor = Work + Power Lost in Motor $P_m = K_e * Velocity * i + i^2 R_m$

Power of amplifier $P_A = \left(V_s - i * R_m - K_e * Velocity\right) * i$

In addition there is a minimum power dissipated by the amplifier when powered regardless of load. The minimum power that the amplifier will consume is roughly

$P_{A,\min} \approx$ drop across op amp power stages + drop across sense resistor + op amp supply

$P_{A,\min} \approx 4 * i + i^2 * .5 + N$

Where N =1.5W for 24V and N = 3W for 48V

For example: assume a 24VDC supply and a motor with $R_m = 4$ohms and $K_e = 5V / RPM$ and desired output currents of 1 and .5 amps.

First calculate the minimum power used in the amplifier.

$P_{A,\min}(1amp) \approx 4 * i + i^2 * .5 + 1.5 = 6W$

$P_{A,\min}(.5amp) \approx 4 * .5 + .5^2 * .5 + 3 = 5.125W$

The power used by the motor will vary by its velocity even though the power lost in the motor is a constant for each value of current. The more power sent to the motor, the less power will be dissipated by the amplifier as heat.

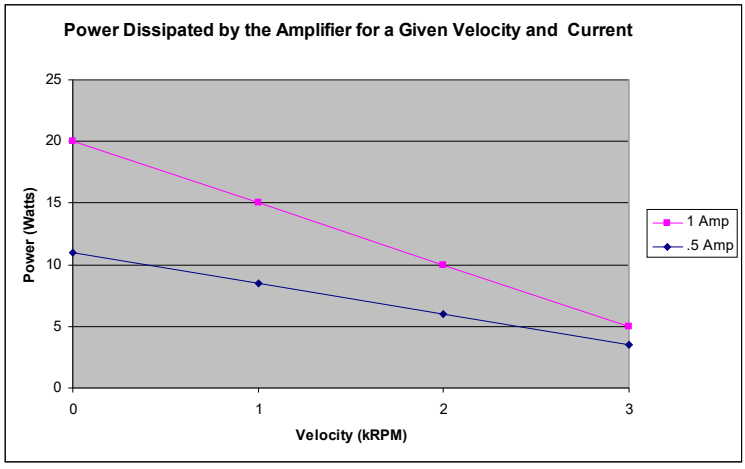


Figure A6.2: Power Dissipation for Velocity and Current

Servo Motor Operation

Brushless Motor Operation

For Brushless motor operation, MT must be set to **1** for the axis. The 6 commands used for setup are the BA, BM, BX, BZ, BC and BI. Please refer to the Command Reference for details. Below is the setup procedure for configuring an axis to operate a Brushless motor.

1. Issue the BA command to specify on which axis to configure sinusoidal commutation.
2. The brushless modulus is the number of encoder counts per magnetic cycle and is set with the BM command. For example, in a rotary motor that has 2 pole pairs and 10000 counts per revolution, the number of encoder counts per magnetic cycle would be 10000/2. BM should be assigned to a value of **5000**.
3. Issue either the BI and BC, BZ, or BX command to initialize the motor for sinusoidal commutation. The command must be executed on every reset or power-up of the controller.

Setup Commands

1. BI/BC Commands – If the motor includes hall sensors, Galil recommends using the BI and BC commands. The BI command uses the hall sensors to set the approximate location of the motor in the magnetic cycle. The BC command will precisely set the commutation angle upon detecting a hall state transition.
2. BZ Command – If the motor does not include hall sensors, Galil recommends using the BZ command. The BZ command forces the motor to a known location in the magnetic cycle. Note that this can cause up to ½ a magnetic cycle of motion.
3. BX Command – If the motor does not have hall sensors and the BZ command generates too much motion for the application, Galil recommends using the BX command. This command utilizes a minimal movement algorithm in order to determine the correct commutation of the motor.

Brushed Motor Operation

The AMP-43640 must be configured for brushed motor operation at the factory. Contact Galil prior to placing the order. Once the amplifier is configured for a brushed motor, the controller needs to be set for brushed mode by setting the BR command to a value of **1**. The A and C motor phases are used for connecting to the brushed motor (B phase is a no connect).

Using External Amplifiers

The BR command must be set to a **-1** for any axis where an AMP-43640 is installed but the use of an external axis is required. This setting will disable the requirement to have the BA, BM and BX or BZ commands executed prior to being able to issue the SH command for that axis. Setting BR**m**=**-1** (where **m** is the axis designator) is required for both external servo amplifiers and stepper drivers. In order to use the full torque limit, make sure the AG setting for the axes using external amplifiers are set to **0** or **1**.
Use D-Sub connectors on the top face of the controller to access necessary signals to run external amplifiers. For more information on connecting external amplifiers, see Step 2b. Connecting External Amplifiers and Motors in Chapter 2.

Setting Peak and Continuous Torque Limits

TK and TL Commands:

The peak and continuous torque limits can be set through the TK and TL commands respectively. The controller will command the torque signal (TT) up to the value specified by TK for a maximum of one second, as shown in Figure A6.3. It will then limit the torque to the value specified by TL. The more time the controller commands the torque below TL, the longer it will command it to TK when necessary. The DMC code example below exhibits the behavior of TT being limited by TL and TK, as shown in Figure A6.3.

```
#tk;          'begin program
MO;           'disable all axes
TLA=5;        'set continuous torque limit of 5V
TKA=9.9982;   'set max peak torque limit to 9.9982V
SHA;          'enable A axis
OFA=9.9982;   'command torque signal to 9.9982V
EN;           'end program
```

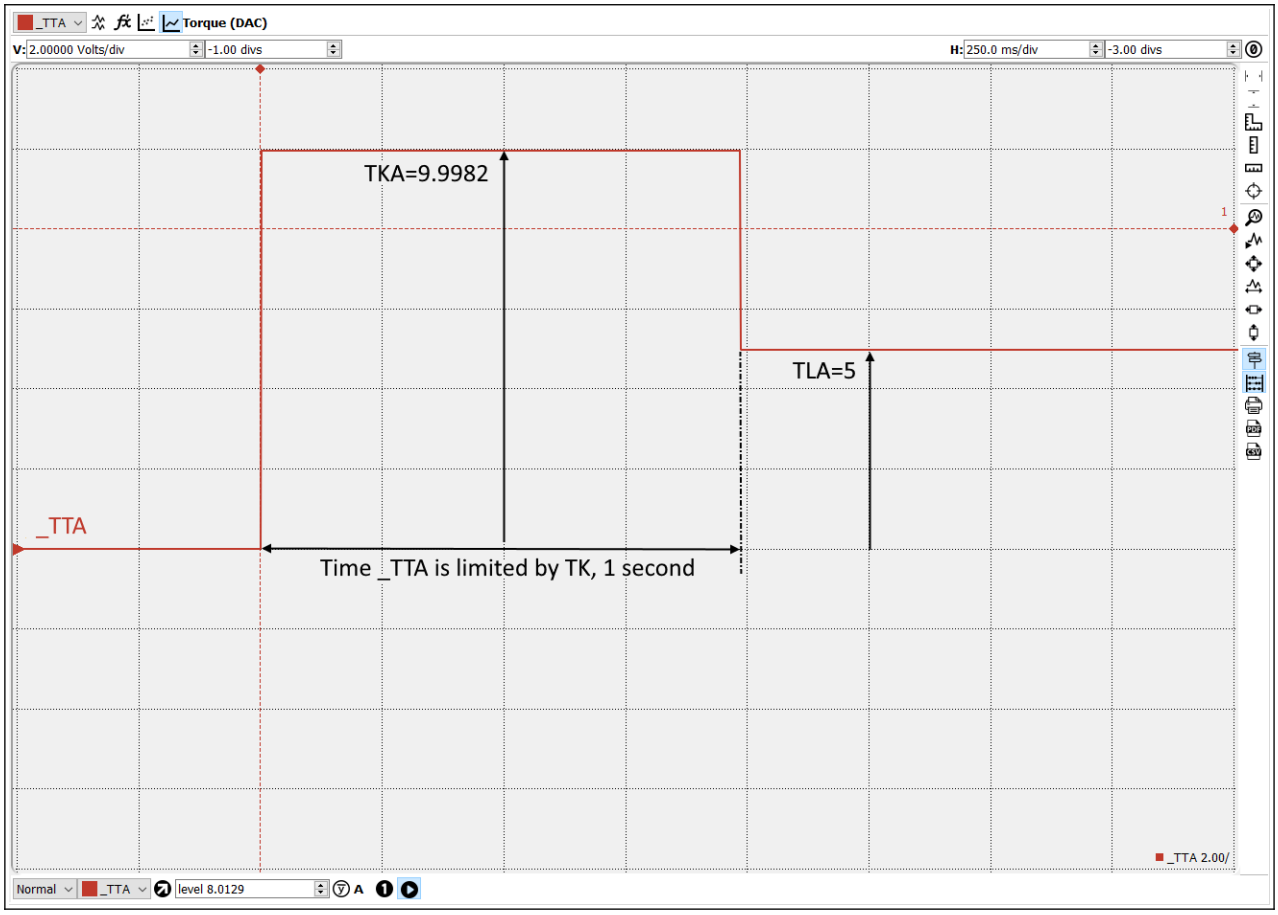


Figure A6.3: Peak Current Operation

Error Monitoring and Protection

This amplifier is protected against Over-Temperature conditions. The amplifier errors can be monitored using the TA command. This command will return a bit mask number representing specific error conditions. For more information, refer to the TA command in the Command Reference.
To handle amplifier error conditions, users can include the **#AMPERR** automatic subroutine in their DMC program. This subroutine is user defined. Two conditions must be met for the **#AMPERR** subroutine to run: the **#AMPERR** subroutine must be defined in the DMC program; and an amplifier error must be detected. When these conditions are met, the program will automatically interrupt thread 0 and jump to the **#AMPERR** subroutine.

Over-Temperature Protection

The amplifier is also equipped with over-temperature protection.
If the average heat sink temperature rises above 80°C, then the amplifier will be disabled. The over-temperature condition will trigger the **#AMPERR** routine if included in the program on the controller.
The amplifier can be re-enabled with the SH command once the temperature drops below 80°C.

WARNING	An Over-Temperature error is usually due to insufficient heat dissipation from the amplifier. Heat must be drawn away from the base of the controller with forced air flow or a heat sink.

Peak Current

If the commanded torque value (TT) on any axis equals the peak torque limit set by TK, a Peak Current condition will be reported for that axis, and the axis bit in the TA**2** bit mask will be set. Peak Current does not cause **#AMPERR** to run. The Peak Current condition reports live and is not latching.

ELO Input

If the ELO input on the controller is asserted, the power stage of the amplifier will be disabled. When the ELO input is triggered, TA**3** will change state and the **#AMPERR** routine will run on Thread 0. To recover from an ELO, the user must issue the MO command followed by the SH command. To recover from an ELO in a DMC program, use MO;WT**2**;SH.
It is recommended that OE**1** be used for all axes when the ELO is used in an application.
See the Optoisolated Input Electrical Information section in Chapter 3 Connecting Hardware for information on connecting the ELO input.

A7 - AMP-43740 (-D3740)

Description

The AMP-43740 is a sinusoidally commutated, 16 bit PWM amplifier. The AMP-43740 can be configured for the motors below with just a few commands.

- Brushless Motors
- Brushed Motors

GDK's Step-By-Step tool provides an interactive guide for configuring brushless and brushed motors.

WARNING	Do not “hot swap” the motor power or supply voltage power input connections. If the amplifier is enabled when the motor connector is connected or disconnected, damage to the amplifier can occur. Galil recommends powering the controller and amplifier down before changing the connector, and breaking the AC side of the power supply connection in order to power down the amplifier.

Electrical Specifications

	Servo (Brushless/Brushed)
Supply Voltage	20-80 VDC
Continuous Current	16 Amps
Peak Current	30 Amps
Switching Frequency	20 kHz
Amplifier Gains	0.8, 1.6, 3.2 Amps/Volt
Minimum Load Inductance	$L(mH) = \frac{Vs(V)}{160 * I_{Ripple}(A)}$

Vs = Supply Voltage, I_{ripple} = 10% of the maximum current at chosen gain

Table A7.1: Amplifier Electrical Specifications

Mating Connectors

	Cable Connectors	Terminal Pins	On board Connector
POWER	6-pin Mega-Fit (Male) Molex #1716920106	Molex #1720630312	6-pin Mega-Fit (Female) Molex #1720650006
A,B,C,D: 4-pin Motor Power Connectors	4-pin Mega-Fit (Male) Molex #1716920104	Molex #1720630312	4-pin Mega-Fit (Female) Molex #1720650004

Table A7.2: Molex part numbers for connectors and terminal pin

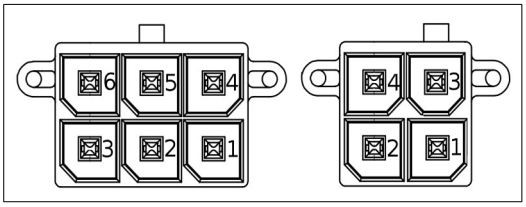


Figure A7.1: Power and Motor Connector Diagram

Power Connector		
Pin Number	**Connection**	
1,2,3	DC Power Supply Ground	
4,5,6	+VS (DC Power)	
Motor Connector		
Pin Number	**Brushless**	**Brushed**
1	Phase C	Phase A-
2	Phase B	No Connect
3	No Connect	No Connect
4	Phase A	Phase A+

Table A7.3: Power and Motor Connector Pin-outs

Servo Motor Operation

Brushless Motor Operation

For Brushless motor operation, MT must be set to **1** for the axis. The 6 commands used for setup are the BA, BM, BX, BZ, BC and BI. Please refer to the Command Reference for details. Below is the setup procedure for configuring an axis to operate a Brushless motor.

1. Issue the BA command to specify on which axis to configure sinusoidal commutation.
2. The brushless modulus is the number of encoder counts per magnetic cycle and is set with the BM command. For example, in a rotary motor that has 2 pole pairs and 10000 counts per revolution, the number of encoder counts per magnetic cycle would be 10000/2. BM should be assigned to a value of **5000**.
3. Issue either the BI and BC, BZ, or BX command to initialize the motor for sinusoidal commutation. The command must be executed on every reset or power-up of the controller.

Setup Commands

1. BI/BC Commands – If the motor includes hall sensors, Galil recommends using the BI and BC commands. The BI command uses the hall sensors to set the approximate location of the motor in the magnetic cycle. The BC command will precisely set the commutation angle upon detecting a hall state transition.
2. BZ Command – If the motor does not include hall sensors, Galil recommends using the BZ command. The BZ command forces the motor to a known location in the magnetic cycle. Note that this can cause up to ½ a magnetic cycle of motion.
3. BX Command – If the motor does not have hall sensors and the BZ command generates too much motion for the application, Galil recommends using the BX command. This command utilizes a minimal movement algorithm in order to determine the correct commutation of the motor.

Brushed Motor Operation

For brushed motor operation, set MT and BR to **1** for the axis.

Using External Amplifiers

The BR command must be set to a **-1** for any axis where an AMP-43740 is installed but the use of an external axis is required. This setting will disable the requirement to have the BA, BM and BX or BZ commands executed prior to being able to issue the SH command for that axis. Setting BR*m*=**-1** (where *m* is the axis designator) is required for both external servo amplifiers and stepper drivers. In order to use the full torque limit, make sure the AG setting for the axes using external amplifiers are set to **0** or **1**.

Use D-Sub connectors on the top face of the controller to access necessary signals to run external amplifiers. For more information on connecting external amplifiers, see Step 2b. Connecting External Amplifiers and Motors in Chapter 2.

Setting Amplifier Gain and Current Loop Gain

The AG command will set the amplifier gain (Amps/Volt). The AU command will set the current loop gain.

AG Command:

The AMP-43740 has 3 amplifier gain settings when configured for servo motors. The gain is set with the AG command. When configuring the amplifier gain, the axis must be in a motor off (MO) state prior to execution of the AG command. For more information on the AG command, refer to the command reference for the DMC-41x3.

AG setting	Gain Value
n = 0	0.8 A/V
n = 1	1.6 A/V
n = 2	3.2 A/V

Table A7.4: Amplifier Gain Settings

AU Command:

Proper configuration of the AU command is essential to optimize the operation of the AMP-43740. This command sets the gain for the current loop on the amplifier. Table A7.5 indicates the recommended AU settings for 24 and 48 VDC power supplies based on motor inductance. When configuring the current loop gain, the axis must be in a motor off (MO) state prior to execution of the AU command. For more information on the AU command, refer to the command reference for the DMC-41x3.

Vsupply VDC	Inductance L (mH)	n
24	L < 1.0	9
24	1 ≤ L < 2.3	10
24	2.3 ≤ L < 4.2	11
24	4.2 ≤ L	12
48	L < 2.4	9
48	2.4 ≤ L < 4.2	10
48	4.2 ≤ L < 7.0	11
48	7.0 ≤ L	12

Table A7.5: Amplifier Current Loop Gain Settings

Setting Peak and Continuous Torque Limits

TK and TL Commands:

The peak and continuous torque limits can be set through the TK and TL commands respectively. The controller will command the torque signal (TT) up to the value specified by TK for a maximum of one second, as shown in Figure A7.2. It will then limit the torque to the value specified by TL. The more time the controller commands the torque below TL, the longer it will command it to TK when necessary. The DMC code example below exhibits the behavior of TT being limited by TL and TK, as shown in Figure A7.2.

```
#tk;          'begin program
MO;           'disable all axes
AGA=2;        'set max amplifier gain of 3.2 A/V
TLA=5;        'set continuous torque limit of 5V
TKA=9.9982;   'set max peak torque limit to 9.9982V
SHA;          'enable A axis
OFA=9.9982;   'command torque signal to 9.9982V
EN;           'end program
```

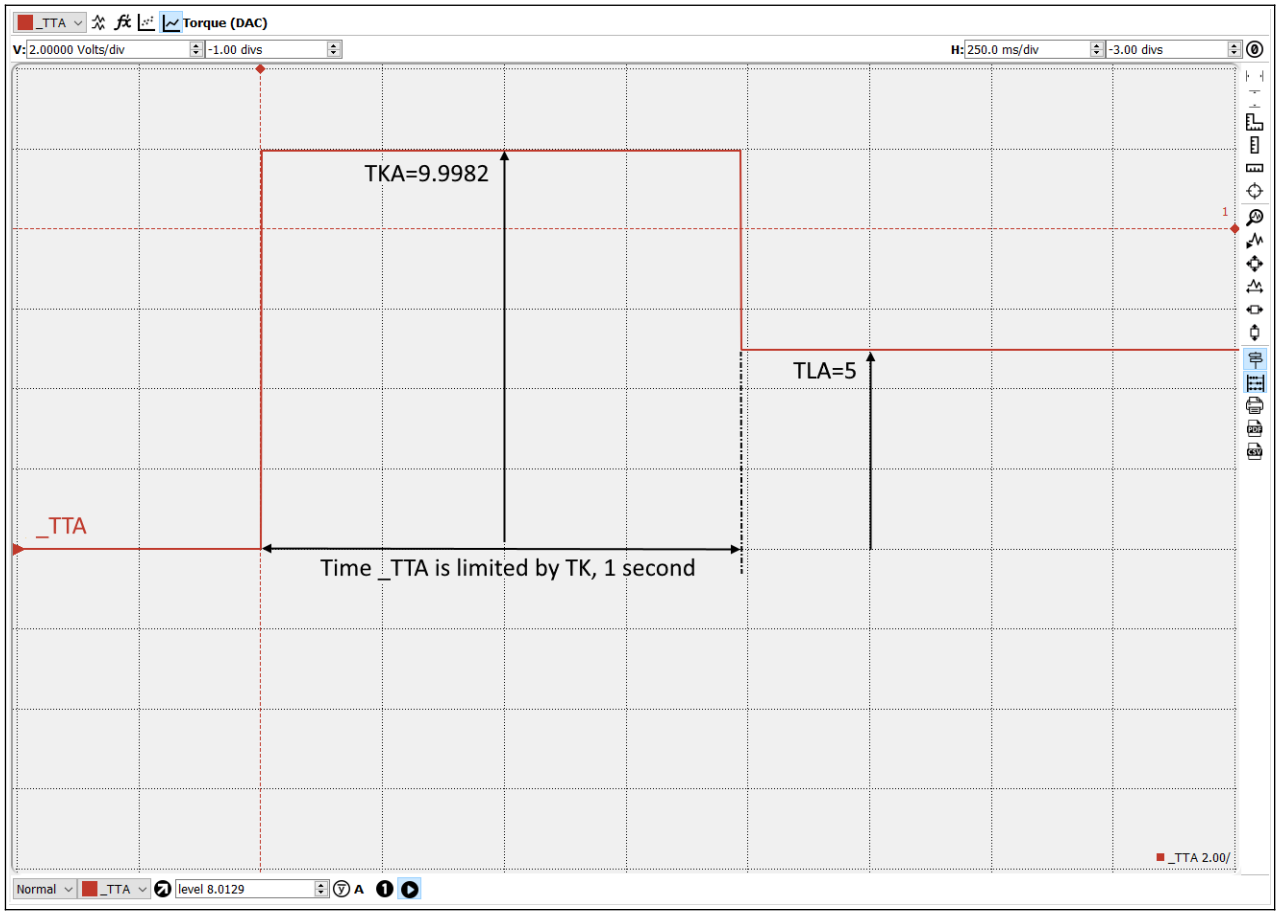


Figure A7.2: Peak Current Operation

Error Monitoring and Protection

This amplifier is protected against Over-Current, Over-Voltage, Over-Temperature, and Under-Voltage conditions. The amplifier errors can be monitored using the TA command. This command will return a bit mask number representing specific error conditions. It will also report whether or not an amplifier error is latched. For more information, refer to the TA command in the Command Reference. Latched errors can be cleared with the AZ command. For more details on clearing latched errors, refer to the Clearing Latched Amplifier Errors section.
To handle amplifier error conditions, users can include the **#AMPERR** automatic subroutine in their DMC program. This subroutine is user defined. Two conditions must be met for the **#AMPERR** subroutine to run: the **#AMPERR** subroutine must be defined in the DMC program; and an amplifier error must be detected. When these conditions are met, the program will automatically interrupt thread 0 and jump to the **#AMPERR** subroutine.

Over-Current Protection

If there is a short circuit in the amplifier or motor, an Over-Current error will be reported and the amplifier will be disabled. The Over-Current amplifier error is a latching error, see the Clearing Latched Amplifier Errors section for details.

WARNING	If this error occurs, it is indicative of a problem at the system level. An Over-Current error is usually due to a short across the motor leads or a short from a motor lead to ground.

Over-Voltage Protection

If the voltage supplied to the amplifier rises above its rated supply voltage, an Over-Voltage error will be reported. While the amplifier is experiencing this condition, current is no longer commanded and the motor phases are shorted together. This results in a drag on the motors causing them to coast to a stop. The amplifier will resume commanding current when the voltage drops below the maximum rated supply voltage. The Over-Voltage amplifier error is *not* a latching error.

WARNING	If this error occurs, it is indicative of a problem at the system level. An Over-Voltage error is usually due to regeneration from the motor and load. Consider using a shunt regulator if this occurs during normal operation.

Over-Temperature Protection

If the average amplifier temperature rises above 80°C, an Over-Temperature error will be reported and the amplifier will be disabled. The error cannot be cleared until the temperature drops below 80°C. The Over-Temperature amplifier error is a latching error, see the Clearing Latched Amplifier Errors section for details.

WARNING	An Over-Temperature error is usually due to insufficient heat dissipation from the amplifier. Heat must be drawn away from the base of the controller with forced air flow or a heat sink.

Under-Voltage Protection

If the voltage supplied to the amplifier drops below its rated supply voltage, an Under-Voltage error will be reported and the amplifier will be disabled. The error cannot be cleared until the supply voltage raises above its minimum rated supply voltage. Under-Voltage does not latch by default. It will become a latching error after AZ**2** is issued to the controller. See the Clearing Latched Amplifier Errors section for details.

Peak Current

If the commanded torque value (TT) on any axis equals the peak torque limit set by TK, a Peak Current condition will be reported for that axis, and the axis bit in the TA**2** bit mask will be set. Peak Current does not cause **#AMPERR** to run. The Peak Current condition reports live and is not latching.

ELO Input

If the ELO input on the controller is asserted, the power stage of the amplifier will be disabled. The ELO input is a latching input, see the Clearing Latched Amplifier Errors section for details. Reference the Optoisolated Input Electrical Information section in Chapter 3 Connecting Hardware for information on connecting the ELO input.

Clearing Latched Amplifier Errors

Amplifier errors that do not latch will be reported in real-time by the TA command.
For latching errors to report and latch for a bank of axes, at least one axis in that bank must be in a servo here state (SH).
To clear latched amplifier errors, users must issue the MO command followed by the SH command. To clear latched amplifier errors in a DMC program, use MO**;**WT**2;**SH. If an axis is moving, motion must be halted before attempting to clear amplifier errors.

A8 – SDM-44040 (-D4040,-D4020)

Description

The SDM-44040 resides inside the DMC-41x3 enclosure and contains four drives for operating two-phase bipolar step motors. The SDM-44040 requires a single 12-30 VDC input. The unit is user-configurable for 1.4 A, 1.0 A, 0.75 A, or 0.5 A per phase and for full-step, half-step, 1/4 step or 1/16 step. A two-axis version, the SDM-44020 is also available.

WARNING	Do not “hot swap” the motor power or supply voltage power input connections. If the amplifier is enabled when the motor connector is connected or disconnected, damage to the amplifier can occur. Galil recommends powering the controller and amplifier down before changing the connector, and breaking the AC side of the power supply connection in order to power down the amplifier.

Electrical Specifications

	Stepper
Supply Voltage	12-30 VDC (12-20 VDC available with ISCNTL)
Max Current (Per Axis)	1.4 A (user-configurable)
Amplifier Gains (Amps/Phase)	0.5, 0.75, 1.0, 1.4 A
Max Step Frequency	3 MHz
Motor Type	Bipolar 2-Phase

Table A8.1: Amplifier Electrical Specifications

Mating Connectors

	Cable Connectors	Terminal Pins	On board Connector
POWER	6-pin Mini-Fit, Jr.™ (Male) Molex #0039012065	Molex #0444761112	6-pin Mini-Fit, Jr.™ (Female) Molex #0039310060
A,B,C,D: 4-pin Motor Power Connectors	4-pin Mini-Fit, Jr.™ (Male) Molex #0039012045	Molex #0444761112	4-pin Mini-Fit, Jr.™ (Female) Molex #0039310040

Table A8.2: Molex part numbers for connectors and terminal pin

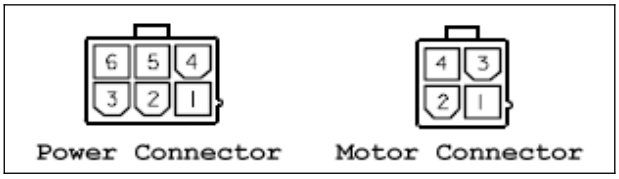


Figure A8.1: Power and Motor Connector Diagram

Power Connector	
Pin Number	**Connection**
1,2,3	DC Power Supply Ground
4,5,6	+VS (DC Power)
Motor Connector	
Pin Number	**Stepper**
1	B-
2	A-
3	B+
4	A+

Table A8.3: Power and Motor Connector Pin-outs

Stepper Motor Operation

Current Level Setup (AG Command)

The AMP-44040 has 4 amplifier gain settings. The gain is set with the AG command. When configuring the amplifier gain, the axis must be in a motor off (MO) state prior to execution of the AG command. For more information on the AG command, refer to the command reference for the DMC-41x3.

AG setting	Gain Value
n = 0	0.5 A
n = 1	0.75 A
n = 2	1.0 A
n = 3	1.4 A

Table A8.4: Amplifier Gain Settings

Low Current Setting (LC Command)

By default, 100% of the current set by the AG command will be delivered to the motor while the axis is holding position. An axis can be configured for Low Current mode so 25% of the current will be sent to the motor while holding position. Low Current mode can also be configured to wait a specified number of samples after a move is completed before lowering the current to 0%. For more information, refer to the LC command in the command reference for the DMC-41x3.

LC setting	Effect on Stepper Phase Current
n > 1	n samples after move finishes before going to 0% holding current
n = 1	25% holding current immediately after move finishes
n = 0	100% holding current
n < 0	n samples after move finishes before going to 25% holding current

Table A8.5: Low Current Mode configurations

Step Drive Resolution Setting (YA command)

When using the SDM-44040, the step drive resolution can be set with the YA command. Note, when running in full step mode – the current to the motor is 70% of maximum. All microstep settings are able to deliver full current.

AG setting	Gain Value
n = 1	Full
n = 2	Half
n = 4	1/4
n = 16	1/16

Table A8.6: Amplifier Gain Settings

Using External Amplifiers

Use D-Sub connectors on the top face of the controller to access necessary signals to run external amplifiers. For more information on connecting external amplifiers, see Step 2b. Connecting External Amplifiers and Motors in Chapter 2.

Error Monitoring and Protection

This amplifier is protected against Over-Current conditions. This amplifier error can be monitored using the TA command. This command will return a bit mask number representing specific error conditions. For more information, refer to the TA command in the Command Reference.
To handle amplifier error conditions, users can include the **#AMPERR** automatic subroutine in their DMC program. This subroutine is user defined. Two conditions must be met for the **#AMPERR** subroutine to run: the **#AMPERR** subroutine must be defined in the DMC program; and an amplifier error must be detected. When these conditions are met, the program will automatically interrupt thread 0 and jump to the **#AMPERR** subroutine.

Over-Current Protection

If there is a short circuit in the amplifier or motor, an Over-Current error will be reported and the amplifier will be disabled. The amplifier can be re-enabled with the SH command when there is no longer an Over-Current condition.

WARNING	If this error occurs, it is indicative of a problem at the system level. An Over-Current error is usually due to a short across the motor leads or a short from a motor lead to ground.

ELO Input

If the ELO input on the controller is asserted, the power stage of the amplifier will be disabled. When the ELO input is triggered, TA**3** will change state and the **#AMPERR** routine will run on Thread 0. To recover from an ELO, the user must issue the MO command followed by the SH command. To recover from an ELO in a DMC program, use MO;WT**2**;SH.
It is recommended that OE**1** be used for all axes when the ELO is used in an application.
See the Optoisolated Input Electrical Information section for information on connecting the ELO input.

A9 – SDM-44140 (-D4140)

Description

The SDM-44140 resides inside the DMC-41x3 enclosure and contains four microstepping drives for operating two-phase bipolar stepper motors. The drives produce 64 microsteps per full step which results in 12,800 steps/rev for a standard 200-step motor. The maximum step rate generated by the controller is 3,000,000 microsteps/second. The SDM-44140 drives motors operating at up to 3 Amps at 20 to 60 VDC (available voltage at motor is 10% less). There are four software selectable current settings: 0.5 A, 1 A, 2 A and 3 A. Plus, a selectable low current mode reduces the current by 75% when the motor is not in motion. No external heatsink is required.

WARNING	Do not “hot swap” the motor power or supply voltage power input connections. If the amplifier is enabled when the motor connector is connected or disconnected, damage to the amplifier can occur. Galil recommends powering the controller and amplifier down before changing the connector, and breaking the AC side of the power supply connection in order to power down the amplifier.

Electrical Specifications

	Stepper
Supply Voltage	20-60 VDC
Max Current (Per Axis)	3.0 Amps (user-configurable)
Amplifier Gains (Amps/Phase)	0.5, 1.0, 2.0, 3.0 A
Max Step Frequency	3 MHz
Minimum Load Inductance	0.5 mH
Motor Type	Bipolar 2-Phase

Table A9.1: Amplifier Electrical Specifications

Mating Connectors

	Cable Connectors	Terminal Pins	On board Connector
POWER	6-pin Mini-Fit, Jr.™ (Male) Molex #0039012065	Molex #0444761112	6-pin Mini-Fit, Jr.™ (Female) Molex #0039310060
A,B,C,D: 4-pin Motor Power Connectors	4-pin Mini-Fit, Jr.™ (Male) Molex #0039012045	Molex #0444761112	4-pin Mini-Fit, Jr.™ (Female) Molex #0039310040

Table A9.2: Molex part numbers for connectors and terminal pin

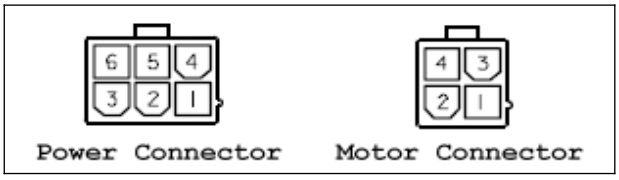


Figure A9.1: Power and Motor Connector Diagram

Power Connector	
Pin Number	**Connection**
1,2,3	DC Power Supply Ground
4,5,6	+VS (DC Power)
Motor Connector	
Pin Number	**Stepper**
1	B-
2	A-
3	B+
4	A+

Table A9.3: Power and Motor Connector Pin-outs

Stepper Motor Operation

Current Level Setup (AG Command)

The AMP-44140 has 4 amplifier gain settings. The gain is set with the AG command. When configuring the amplifier gain, the axis must be in a motor off (MO) state prior to execution of the AG command. For more information on the AG command, refer to the command reference for the DMC-41x3.

AG setting	Gain Value
n = 0	0.5 A
n = 1	1.0 A
n = 2	2.0 A
n = 3	3.0 A

Table A9.4: Amplifier Gain Settings

Low Current Setting (LC Command)

By default, 100% of the current set by the AG command will be delivered to the motor while the axis is holding position. An axis can be configured for Low Current mode so 25% of the current will be sent to the motor while holding position. Low Current mode can also be configured to wait a specified number of samples after a move is completed before lowering the current to 0%. For more information, refer to the LC command in the command reference for the DMC-41x3.

LC setting	Effect on Stepper Phase Current
n > 1	n samples after move finishes before going to 0% holding current
n = 1	25% holding current immediately after move finishes
n = 0	100% holding current
n < 0	n samples after move finishes before going to 25% holding current

Table A9.5: Low Current Mode configurations

Using External Amplifiers

Use D-Sub connectors on the top face of the controller to access necessary signals to run external amplifiers. For more information on connecting external amplifiers, see Step 2b. Connecting External Amplifiers and Motors in Chapter 2.

Error Monitoring and Protection

This amplifier is protected against Over-Current and Under-Voltage conditions. This amplifier error can be monitored using the TA command. This command will return a bit mask number representing specific error conditions. For more information, refer to the TA command in the Command Reference.
To handle amplifier error conditions, users can include the #AMPERR automatic subroutine in their DMC program. This subroutine is user defined. Two conditions must be met for the #AMPERR subroutine to run: the #AMPERR subroutine must be defined in the DMC program; and an amplifier error must be detected. When these conditions are met, the program will automatically interrupt thread 0 and jump to the #AMPERR subroutine.

Over-Current Protection

The stepper driver has circuitry to protect against Over-Current. If the total current from a set of 2 axes (ie A and B or C and D) exceeds 10 A, the SDM-44140 will be disabled. The amplifier can be re-enabled with the SH command when there is no longer an Over-Current condition.

WARNING	If this error occurs, it is indicative of a problem at the system level. An Over-Current error is usually due to a short across the motor leads or a short from a motor lead to ground.

Under-Voltage Protection

If the voltage supplied to the amplifier drops below its rated supply voltage, an Under-Voltage error will be reported and the amplifier will be disabled.
If there is an #AMPERR routine and the controller is powered before the amplifier, then the #AMPERR routine will automatically be triggered. The amplifier will return to normal operation once the supply is raised above the rated supply voltage assuming it did not cause the position error to exceed the error limit.

ELO Input

If the ELO input on the controller is asserted, the power stage of the amplifier will be disabled. When the ELO input is triggered, TA3 will change state and the #AMPERR routine will run on Thread 0. To recover from an ELO, the user must issue the MO command followed by the SH command. To recover from an ELO in a DMC program, use MO;WT2;SH.
It is recommended that OE1 be used for all axes when the ELO is used in an application.
See the Optoisolated Input Electrical Information section for information on connecting the ELO input.