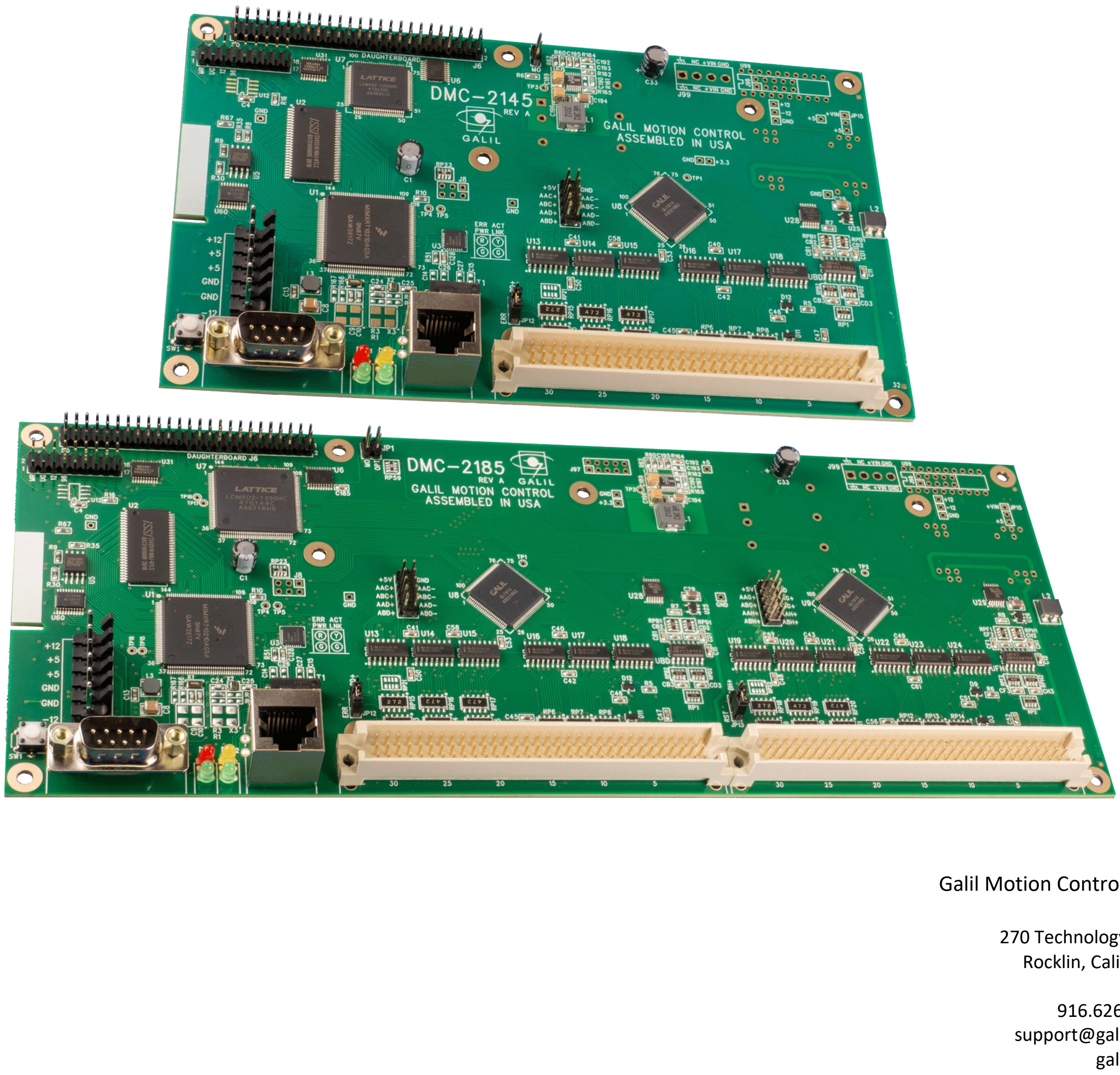
USER MANUAL

DMC-21x5

Manual Rev. 1.0a1



Galil Motion Control, Inc.

270 Technology Way
Rocklin, California

916.626.0101
support@galil.com
galil.com

12/2020

Using This Manual

This user manual provides information for proper operation of the DMC-21x5 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.

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

The DMC-21x5 motion controller has been designed to work with both servo and stepper type motors. Installation and system setup will vary depending upon whether the controller will be used with stepper motors or servo motors. To make finding the appropriate instructions faster and easier, icons will be next to any information that applies exclusively to one type of system. Otherwise, assume that the instructions apply to all types of systems. The icon legend is shown below.

⊖	Attention: Pertains to servo motor use.
⎍	Attention: Pertains to stepper motor use.
2185	Attention: Pertains to controllers with more than 4 axes.

Please note that many examples are written for the DMC-2145 four-axes controller or the DMC-2185 eight axes controller. Users of the DMC-2135 3-axis controller, DMC-2125 2-axes controller or DMC-2115 1-axis controller should note that the DMC-2135 uses the axes denoted as ABC, the DMC-2125 uses the axes denoted as AB, and the DMC-2115 uses the A-axis only.

Examples for the DMC-2185 denote the axes as A,B,C,D,E,F,G,H. Users of the DMC-2155 5-axes controller, DMC-2165 6-axes controller, or DMC-2175 7-axes controller should note that the DMC-2155 denotes the axes as A,B,C,D,E, the DMC-2165 denotes the axes as A,B,C,D,E,F and the DMC-2175 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-21x5 series is the latest in Galil's Econo line of motion controllers and is an upgraded version of the popular DMC-21x3 series controller. The controller series offers many enhanced features compared to prior generation controllers including high speed communications, non-volatile program memory, faster encoder speeds, and improved command processing.

The DMC-21x5 is available with up to eight axes in a single stand alone unit. The DMC-2115, 2125, 2135, 2145 are one thru four axes controllers and the DMC-2155, 2165, 2175, 2185 are five thru eight axes 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-21x5 can be used for applications involving jogging, point-to-point positioning, vector positioning, electronic gearing, multiple move sequences, and contouring. The controller eliminates jerk by programmable acceleration and deceleration with profile smoothing. For smooth following of complex contours, the DMC-21x5 provides continuous vector feed of an infinite 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-21x5 provides uncommitted I/O, including 8 digital inputs (16 inputs for DMC-2155 thru DMC-2185), 8 outputs (16 outputs for DMC-2155 thru DMC-2185), and 8 analog inputs for interface to joysticks, sensors, and pressure transducers. Further I/O is available if the auxiliary encoders are not being used (2 inputs / each axis). Dedicated inputs are provided for forward and reverse limits, Abort, home, and definable input interrupts.

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

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 Visual Basic, C, C++ etc.

Part Numbers

The DMC controller board comes in two sizes, 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-21x5.

The full DMC-21x5 part number is a combination of the DMC controller part number (DMC-21x5), controller board options, and optional accessory types, where Y is option for the controller or accessory.

The placement of the AMP/SDM options is extremely important for 5-8 axis models. The first AMP/SDM (Axis A-D) will be placed for the first bank of axes and the second AMP/SDM (Axis E-H) will be placed for the second bank.

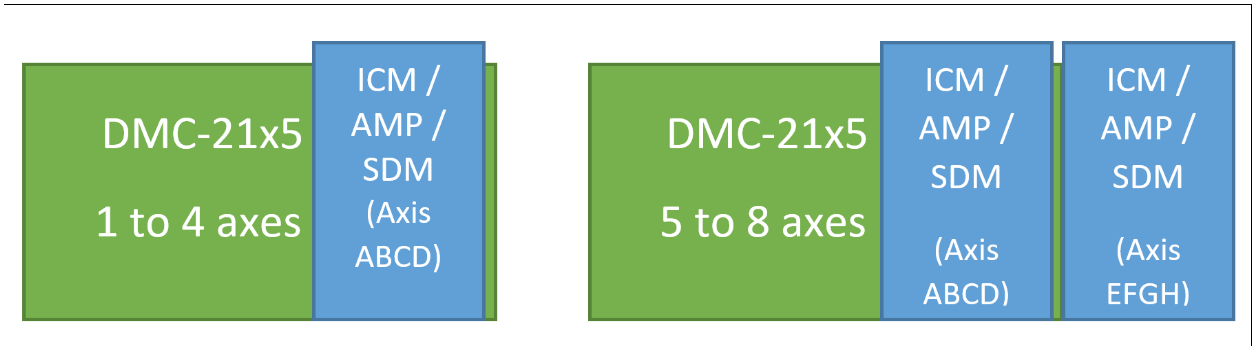

DMC-21x5
1 to 4 axes
ICM / AMP / SDM (Axis ABCD)
DMC-21x5
5 to 8 axes
ICM / AMP / SDM (Axis ABCD)
ICM / AMP / SDM (Axis EFGH)

Figure 1.1: Layout of DMC-21x5 with one bank of axes and layout of DMC-21x5 with two banks of axes

Please use the DMC-21x5 part number generator to check the validity of all part numbers before ordering:

https://galil.com/order/part-number-generator/dmc-21x5

DMC, "DMC-21x5(Y)" Options

Option Type	Options	Brief Description	Documentation
x	1, 2, 3, 4, 5, 6, 7, and 8	Number of controller axis	N/A
Y	DIN	DIN Rail Mounting	DMC, "DMC-21x5(Y)" Options starting on pg 138
	UP	Upward Facing 96 Pin DIN Connector	
	DOWN	Downward Facing 96 Pin DIN Connector	
	V	Vertical RS232, Status Lights, and Ethernet	
	H	Horizontal RS232, Status Lights, and Ethernet	
	VP	Vertical Power Connector	
	HP	Horizontal Power Connector	
	RA	Right Angle 96 Pin Connector	
	MO	Motor Off Jumper Installed	
	TRES	Termination Resistors	
	DC24, DC48	DC to DC Converter	

Table 1.1: List of DMC options

Accessory Options

Option Type	Options	Brief Description	Documentation
AMP	20341	20 W brush-type only drive	A1 – AMP-20341, pg 151
	20440	200 W brush-type only drive	A2 – AMP-20440, pg 155
	20540/20520	500 W trapazoidal servo drive 2 and 4-axis models	A3 – AMP-20545/20525, pg 161
SDM	20242	Configurable microstepping drive	A4 – SDM-20242, pg 169
	20640	1/64 microstepping drive	A5 – SDM-20645, pg 175
ICM	20100	I/O module	A6 – ICM-20100, pg 181
	20105	Optoisolated I/O module	A7 – ICM-20105, pg 183
DB	28040	Extended I/O daughter board	A8 – DB-28045, pg 190
Y	ISCNTL	Isolated controller power	Accessory Specific Options, starting on pg 139 Not available for all accessory options, see the proper documentation.
	SSR	Solid State Relay	
	LAEN	Low Amplifier Enable Configuration	
	24V	24V Amplifier Enable Configuration	
	BOX	Metal Enclosure	
	5V	5V Logic for Extended I/O	
	16BIT	16Bit Resolution on Analog Inputs	
	SHUNT	Shunt Regulator	A9 – SR-19900, pg 194

Table 1.2: List of accessories and options

Overview of Motor Types

The DMC-21x5 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-21x5 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-21x5 can control stepper motors. In this mode, the controller provides two signals to connect to the stepper motor: Step and Direction. For stepper motor operation, the controller does not require an encoder and operates the stepper motor in an open loop fashion. Chapter 2 describes the proper connection and procedure for using stepper motors.

If encoders are available on the stepper motor, 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.

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.

Overview of Galil Amplifiers and Drivers

With the DMC-21x5 Galil offers a variety of Servo Amplifiers and Stepper Drivers that are integrated with the controller. Using the Galil Amplifiers and Drivers provides a simple straightforward motion control solution . Instead of having to route a +/-10V motor command signal or STEP/DIR to some external box, all the wiring is taken care of. In addition, Galil's amplifiers reside on top of the controller, saving real estate space and the hassle of configuring a separate device.

A full list of amplifier specifications and details can be found in the Accessory Components, pg 149.

DMC-21x5 Functional Elements

The DMC-21x5 circuitry can be divided into the following functional groups as shown in Figure 1.2 and discussed below.

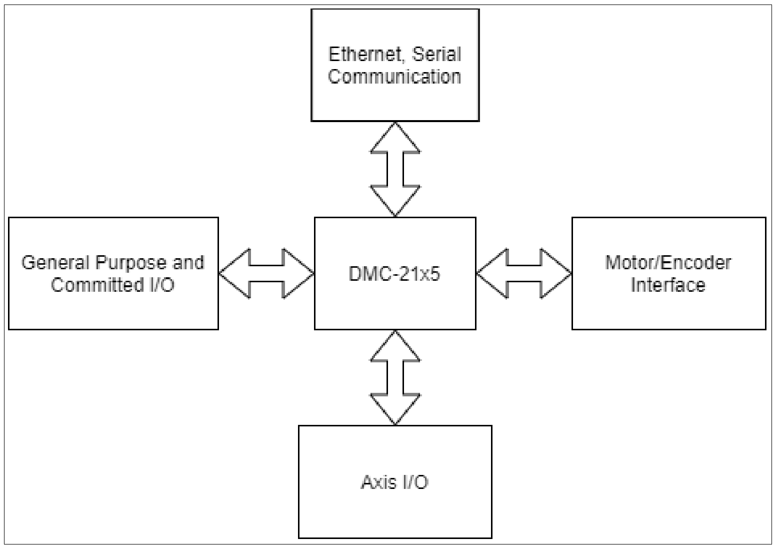

Ethernet, Serial Communication
General Purpose and Committed I/O
DMC-21x5
Motor/Encoder Interface
Axis I/O

Figure 1.2: DMC-21x5 Functional Elements

DMC-21x5

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/Encoder Interface

The controller provides an interface for 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.

Axis I/O

The controller provided dedicated I/O for each axis such as forward and reverse limit switches, home switches, and a high speed latch.

Communication

The communication interface with the DMC-21x5 consists of high speed 100bT Ethernet and an RS232 serial port.

General Purpose and Committed I/O

The DMC-21x5 provides interface circuitry for 8 TTL inputs, 8 TTL outputs, as well as 40 extended IO bits and 8 analog inputs with 12-Bit or 16-Bit ADC with certain accessories. 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.

2185 The DMC-2155 through DMC-2185 controller provides an additional 8 inputs and 8 outputs.

System Elements

As shown in Figure 1.3, the DMC-21x5 is part of a motion control system which includes amplifiers, motors and encoders. These elements are described below.

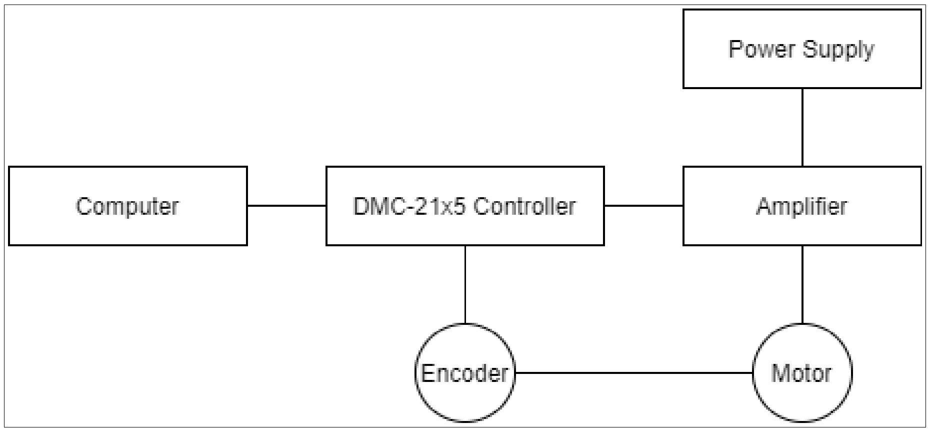


Figure 1.3: Elements of Servo Systems

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-21x5.

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 with the DMC-21x5. See the Accessory Components section in the Appendices or https://galil.com/motion-controllers/multi-axis/dmc-21x5 for more information.

Encoder

An encoder translates motion into electrical pulses which are fed back into the controller. The DMC-21x5 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-21x5 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-21x5 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-21x5. Single-ended 12 Volt signals require a bias voltage input to the complementary inputs.

The DMC-21x5 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 about particular system requirements.

Watch Dog Timer

The DMC-21x5 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-21x5 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-21x5 to normal operation. Consult the factory for a Return Materials Authorization (RMA) Number if the controller has been damaged.

Chapter 2 Getting Started

Elements Needed

For a complete system, Galil recommends the following elements:

1. DMC-21x5 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-21x5
6. PC running Windows 7 or newer (64 bit) or Linux (64 bit)
7. Ethernet cable or straight-through serial cable
8. GDK software

GDK is recommended for first time users of the DMC-21x5. It provides interactive instructions for system connection, tuning, and analysis.

Layout

DMC-2145 Dimensions

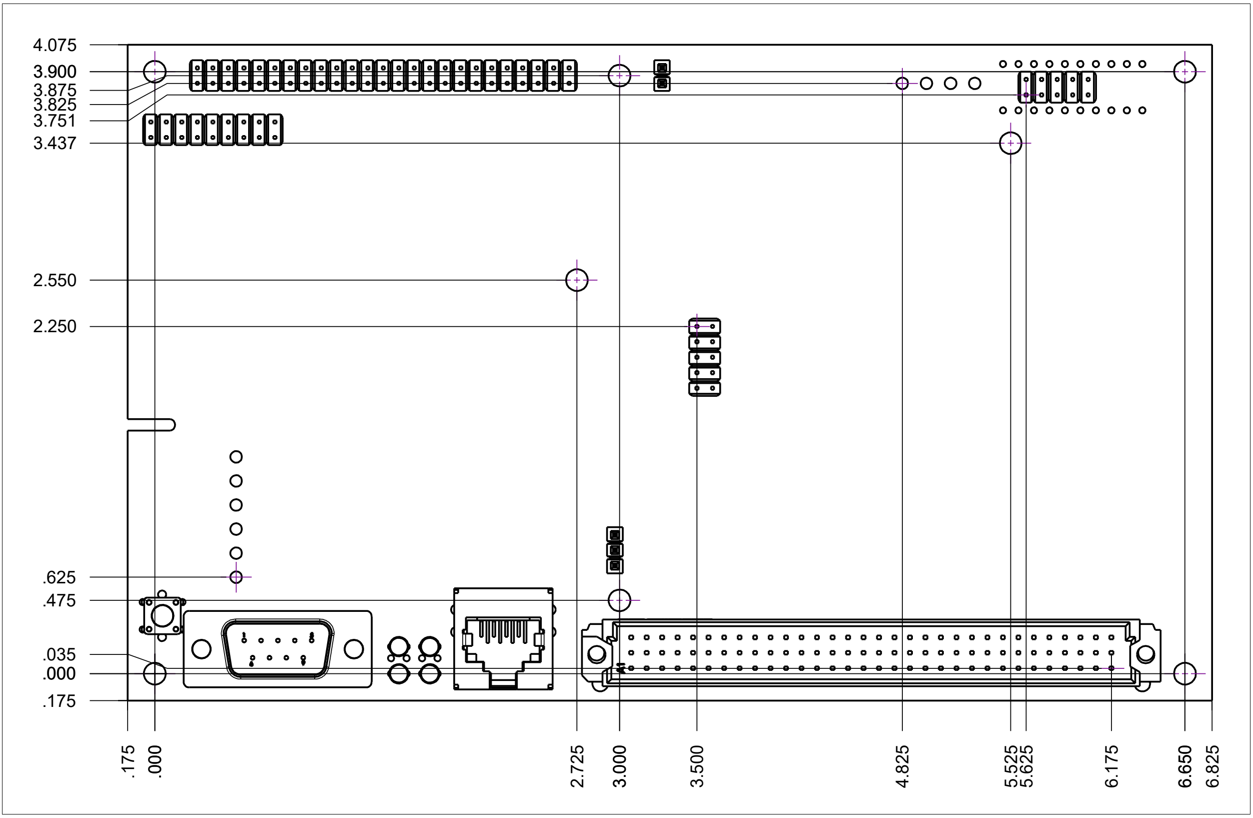


Figure 2.1: DMC-21x5 Dimensions (inches) where x is 1 through 4 axes

DMC-2185 Dimensions

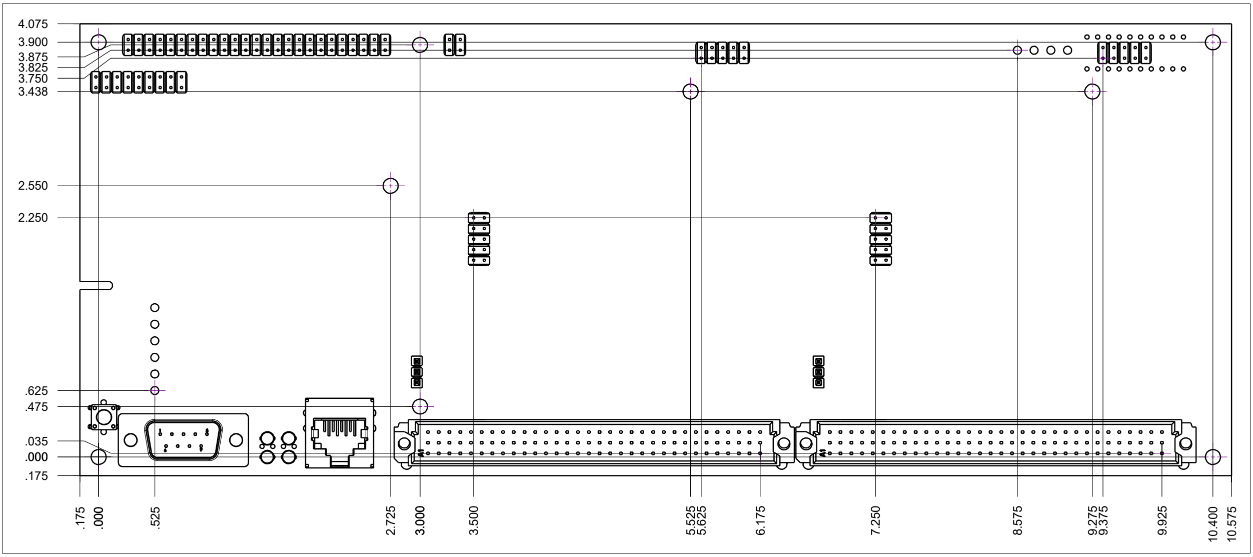


Figure 2.2: DMC-21x5 Dimensions (inches) where x is 5 through 8 axes

Installing the DMC, Amplifiers, and Motors

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

Step 1. Connecting Encoder Feedback, pg 11	*Optional for steppers*
Step 2a. Wiring Motors to Galil's Amplifiers, pg 12	*Internal Amplifiers only*
Step 2b. Connecting External Amplifiers and Motors, pg 13	*External Amplifiers only*
Step 3. Power the Controller, pg 14	
Step 4. Install the Communications Software, pg 15	
Step 5. Establish Communications with Galil Software, pg 15	
Step 6. Setting Safety Features, pg 16	
Step 7. Tune the Servo System, pg 17	*Servo motors only*

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-21x5 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
None[1]	–	–	–	–
Other	Contact Galil			

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

[1] 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 pulse and direction. 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.

Encoder pin-outs can be found here:

A1 – AMP-20341, pg 153
A2 – AMP-20440, pg 157
A3 – AMP-20545/20525, pg 163
A4 – SDM-20242, pg 171
A5 – SDM-20645, pg 177
A6 – ICM-20100, pg 182
A7 – ICM-20105, pg 184

Step 2a. Wiring Motors to Galil's 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
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-20341, pg 151	Brushed	No
A2 – AMP-20440, pg 155	Brushed	No
A3 – AMP-20545/20525, pg 161	Trapezoidal or Brushed	Required for brushless motors
A4 – SDM-20242, pg 169	N/A, Stepper	No
A5 – SDM-20645, pg 175	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 here:

A3 – AMP-20545/20525, pg 163

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-21x5. 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 walkthrough 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 there is uncertainty 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:

External Amplifier Interface, pg 142

A6 – ICM-20100, pg 182

A7 – ICM-20105, pg 184

For see External Amplifier Interface, pg 23 for full electrical specifications .

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-21x5 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:

External Amplifier Interface, pg 142

A6 – ICM-20100, pg 182

A7 – ICM-20105, pg 184

See External Amplifier Interface, pg 23 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.

Part Number	**+/-12, +5**	**+VDC, Top Right Connector**	**AMP/SDM Power Connector, Axis A-D** (6- or 4-pin Molex)	**AMP/SDM Power Connector, Axis E-H** (6- or 4-pin Molex)
DMC-21x5	X			
DMC-21x5-DC		x		
DMC-21x5-AMP/SDM	x		x	
DMC-21x5-DC-AMP/SDM			x	
DMC-21x5-DC-AMP/SDM-ISCNTL		x	x	
DMC-21x5-AMP/SDM-AMP/SDM	x		x	x
DMC-21x5-DC-AMP/SDM-AMP/SDM			x	x
DMC-21x5-DC-AMP/SDM-AMP/SDM-ISCNTL		x	x	x

Table 2.5: Required power connections based upon options ordered

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 141. The power specifications for the controller are provided in Power Requirements, pg 136 and the power specifications for each amplifier are found under their specific section in the appendix, see Accessory Components, pg 149.

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.

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: https://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 a NIC card using Windows to connect to a DMC controller:

https://galil.com/learn/online-videos/basics-galil-design-kit-manager-terminal-and-editor

See the GDK manual for using the software to communicate: https://galil.com/sw/pub/all/doc/gdk/man/

Step 6. 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 (`TLm=2.5`)

TK setting = (5.0A)/(0.8A/V) = 6.25V (`TKm=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 13 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 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 118.

Step 7. 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 124.

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

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

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

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

Chapter 3 Connecting Hardware

Overview

The DMC-21x5 provides a variety of dedicated inputs such as forward limit, reverse limit, home, and Abort signals. as well as general purpose

The DMC-21x5 provides digital inputs for forward limit, reverse limit, home, and Abort signals. The controller also has 8 general purpose uncommitted inputs as well as 8 outputs.

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

The controller can be ordered with different accessories that provide additional features. For details on connecting to specific DMC-21x5 accessory, reference the Accessory Components section or Table 3.1 below.

Accessory	Pin-outs	Digital Inputs	Digital Outputs	External Amplifier Interface	Analog Inputs	Extended I/O
None	pg 142	TTL, pg 19	TTL, pg 22	pg 23	NA	NA
A1 – AMP-20341	pg 153	TTL, pg 19	TTL, pg 22	pg 23	NA	NA
A2 – AMP-20440	pg 155	TTL, pg 19	TTL, pg 22	pg 23	NA	NA
A3 – AMP-20545/20525	pg 163	TTL, pg 19	TTL, pg 22	pg 167	pg 168	NA
A4 – SDM-20242	pg 171	TTL, pg 19	TTL, pg 22	pg 23	NA	NA
A5 – SDM-20645	pg 177	TTL, pg 19	TTL, pg 22	pg 23	pg 180	NA
A6 – ICM-20100	pg 182	TTL, pg 19	TTL, pg 22	pg 23	NA	NA
A7 – ICM-20105	pg 184	Optoisolated, pg 185	Optoisolated, pg 187	pg 188	NA	NA
A8 – DB-28045	pg 191	NA	NA	NA	pg 193	pg 192

Table 3.1: Hardware specification page locations for each controller accessory

Overview of Inputs

General Purpose Digital Inputs

The DMC-21x5 inputs are rated for 5V. The amount of inputs available on the DMC-21x5 depends on the number of axes on the controller. For example, 1-4 axis models come with a single bank of 8 inputs, Bank 0 (DI[8:1]). 5-8 axis models come with an additional bank of 8 inputs, Bank 1 (DI[16:9]), for a total of 16 inputs. These inputs can be read individually by using the `@IN[n]` operand where n specifies the input number. 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 DI1 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.

Electrical Specifications

Maximum Voltage	5 V_{DC}
Minimum Voltage	0 V_{DC}

Inputs are pulled up to 5V through a 4.7kΩ resistor

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-21x5.

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.

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

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 ±12 V. 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).

Example:

A DMC-2115 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 operand `@IN[81]` and `@IN[82]`.

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

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

Limit Switch Inputs

The forward limit switch (FLSx) inhibits motion in the forward direction immediately upon activation of the switch. The reverse limit switch (RLSx) inhibits motion in the reverse direction immediately upon activation of the switch. 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 zero 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, `_LFm` and `_LRm`, 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_LFm` or `MG_LRm`. 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 Inputs

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.

There are three homing routines supported by the DMC-21x5: 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. 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 Abort input is an active low signal.

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.

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

NOTE: The effect of an Abort input is dependent on the state of the off-on-error function (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.

Reset Input/Reset Button

When the Reset input is triggered the controller will be reset. The Reset input and reset button will not master reset the controller unless the MRST jumper is installed during a controller reset. The Reset input is an active low signal.

Overview of Outputs

General Purpose Digital Outputs

The DMC-21x5 outputs are rated for 5V. The amount of outputs depends on the number of axes on the controller. 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. These inputs can be read individually by using the `@OUT[n]` operand where n specifies the input number.

Electrical Specifications

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

Output Compare

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 (624 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

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 ER.
2. The reset line on the controller is held low or is being affected by noise.
3. There is a failure on the controller and the processor is resetting itself.
4. There is a failure with the output IC which drives the error signal.

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

External Amplifier Interface

External Servo Control

The DMC-21x5 command voltage ranges between ±10V and is output on the motor command line. 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

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). These outputs are typically used for interfacing to external stepper drivers. See the `MT` command for more details.

Step and Direction Electrical Specifications

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

Amplifier Enable

The DMC-21x5 has an amplifier enable signal which changes under the following conditions:

- The motor off command `MO` is given or the watchdog timer activates.
- The OE command (Off-On-Error) is set and the position error exceeds the error limit (set by ER).
- A limit switch is reached (see OE command in the Command Reference for more information).

The default configuration of the amplifier enable signal is 5V 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.

Note: Many amplifiers designate the enable input as 'inhibit'.

Chapter 4 Software Tools and Communication

Introduction

The default configuration DMC-21x5 has one one RS-232 port and one Ethernet port. 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-21x5 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.

Controller Response to Commands

Most DMC-21x5 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-21x5 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 RS-232 port back through the RS-232 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-21x5 will return the data followed by a carriage return, line feed and : .

It is good practice to check for : after each command is sent to prevent errors. An echo function is provided to enable associating the DMC-21x5 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 RS-232 port or Ethernet handles. This response could be generated as a result of messages using the MG command or as a result of a command error. These responses are known as unsolicited messages since they are not generated as the direct response to a command.

Messages can be directed to a specific port using the specific Port arguments – see 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 serial 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.

RS-232 Port

The RS-232 port by default is an interpreted serial communication port for the DMC-21x5 and will receive and respond to DMC commands. The controller can read and write generic data but the user must write their own communication routines. A full description of the RS-232 default configuration settings can be found in the following sections.

- A straight through cable is required for interfacing to the RS-232 port. Baud rates are set via the controller's jumpers.
- Firmware can only be updated from the RS-232 port.

Baud Rate Selection

JP2 JUMPER SETTINGS		
1200	**9600**	**BAUD RATE**
ON	OFF	1200
OFF	ON	9600
OFF	OFF	19200

Table 4.1: Baud Rate Selection

Handshaking

The RS-232 main port is set for hardware handshaking. Hardware Handshaking uses the RTS and CTS lines. The CTS line will go high whenever the DMC-40x0 is not ready to receive additional characters. The RTS line will inhibit the DMC-40x0 from sending additional characters. Note, the RTS line goes high for inhibit.

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-21x5 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 master, or client, connects to the slave, or 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.

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-21x5 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-21x5 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-58-00-03

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-21x5 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-21x5 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 do so manually through Galil software. When opening the Galil Software, it will respond with a list of all DMC-21x5'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.

WARNING	Be sure that there is only one DHCP server running. If the network has DHCP running, it may automatically assign an IP address to the DMC controller upon linking it to the network. In order to ensure that the IP address is correct, please contact the 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 serial port. The `IA` command will automatically change the `DH` setting from 1 to 0. 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-21x5 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. Communication over TCP to the controller uses port 23. Communication over UDP to the controller uses port 23 and 60007.

Communicating with Multiple Devices

The DMC-21x5 is capable of supporting multiple masters and slaves. The masters may be multiple PC's that send commands to the controller. The slaves are typically peripheral I/O devices that receive commands from the controller. The term "Master" is equivalent to the internet "client". The term "Slave" is equivalent to the internet "server".

An Ethernet handle is a communication resource within a device. The DMC-21x5 can have a maximum of 8 Ethernet handles open at any time. When using TCP/IP, each master or slave uses an individual Ethernet handle. In UDP/IP, one handle may be used for all the masters, but each slave uses one. Pings and ARPs do not occupy handles.

When the Galil controller acts as the master, the `IH` command is used to assign handles and connect to its slaves. 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> u command.

Using Third Party Software

Galil supports DHCP, ARP, 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. Ping is used to check the communication between the device at a specific IP address and the host computer.

The DMC-21x5 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.

Data Record

The DMC-21x5 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
00	UB	1st Byte of Header
01	UB	2nd Byte of Header
02	UB	3rd Byte of Header
03	UB	4th Byte of Header
04-05	UW	sample number
06	UB	general input block 0 (inputs 1-8)
07	UB	general input block 1 (inputs 9-16)
08	UB	general input block 2 (inputs 17-24)
09	UB	general input block 3 (inputs 25-32)
10	UB	general input block 4 (inputs 33-40)
11	UB	general input block 5 (inputs 41-48)
12	UB	general input block 6 (inputs 49-56)
13	UB	general input block 7 (inputs 57-64)
14	UB	general input block 8 (inputs 65-72)
15	UB	general input block 9 (inputs 73-80)
16	UB	general output block 0 (outputs 1-8)
17	UB	general output block 1 (outputs 9-16)
18	UB	general output block 2 (outputs 17-24)
19	UB	general output block 3 (outputs 25-32)
20	UB	general output block 4 (outputs 33-40)
21	UB	general output block 5 (outputs 41-48)
22	UB	general output block 6 (outputs 49-56)
23	UB	general output block 7 (outputs 57-64)
24	UB	general output block 8 (outputs 65-72)
25	UB	general output block 9 (outputs 73-80)
26	UB	error code
27	UB	general status
28-29	UW	segment count of coordinated move for S plane
30-31	UW	coordinated move status for S plane – see bit field map below
32-35	SL	distance traveled in coordinated move for S plane
36-37	UW	segment count of coordinated move for T plane
38-39	UW	coordinated move status for T plane – see bit field map below
40-43	SL	distance traveled in coordinated move for T plane

Axis Information

ADDR	TYPE	ITEM	ADDR	TYPE	ITEM
44-45	UW	A axis status – see bit field map below	156-157	UW	E axis status – see bit field map below
46	UB	A axis switches – see bit field map below	158	UB	E axis switches – see bit field map below
47	UB	A axis stop code	159	UB	E axis stop code
48-51	SL	A axis reference position	160-163	SL	E axis reference position
52-55	SL	A axis motor position	164-167	SL	E axis motor position
56-59	SL	A axis position error	168-171	SL	E axis position error
60-63	SL	A axis auxiliary position	172-175	SL	E axis auxiliary position
64-67	SL	A axis velocity	176-179	SL	E axis velocity
68-69	SL	A axis torque	180-181	SL	E axis torque
70-71	SW or UW [1]	A axis analog input	182-183	SW or UW [1]	E axis analog input
72-73	UW	B axis status – see bit field map below	184-185	UW	F axis status – see bit field map below
74	UB	B axis switches – see bit field map below	186	UB	F axis switches – see bit field map below
75	UB	B axis stop code	187	UB	F axis stop code
76-79	SL	B axis reference position	188-191	SL	F axis reference position
80-83	SL	B axis motor position	192-195	SL	F axis motor position
84-87	SL	B axis position error	196-199	SL	F axis position error
88-91	SL	B axis auxiliary position	200-203	SL	F axis auxiliary position
92-95	SL	B axis velocity	204-207	SL	F axis velocity
96-97	SL	B axis torque	208-209	SL	F axis torque
98-99	SW or UW [1]	B axis analog input	210-211	SW or UW [1]	F axis analog input
100-101	UW	C axis status – see bit field map below	212-213	UW	G axis status – see bit field map below
102	UB	C axis switches – see bit field map below	214	UB	G axis switches – see bit field map below
103	UB	C axis stop code	215	UB	G axis stop code
104-107	SL	C axis reference position	216-219	SL	G axis reference position
108-111	SL	C axis motor position	220-223	SL	G axis motor position
112-115	SL	C axis position error	224-227	SL	G axis position error
116-119	SL	C axis auxiliary position	228-231	SL	G axis auxiliary position
120-123	SL	C axis velocity	232-235	SL	G axis velocity
124-125	SL	C axis torque	236-237	SL	G axis torque
126-127	SW or UW [1]	C axis analog input	238-239	SW or UW [1]	G axis analog input
128-129	UW	D axis status – see bit field map below	240-241	UW	H axis status – see bit field map below
130	UB	D axis switches – see bit field map below	242	UB	H axis switches – see bit field map below
131	UB	D axis stop code	243	UB	H axis stop code
132-135	SL	D axis reference position	244-247	SL	H axis reference position
136-139	SL	D axis motor position	248-251	SL	H axis motor position
140-143	SL	D axis position error	252-255	SL	H axis position error
144-147	SL	D axis auxiliary position	256-259	SL	H axis auxiliary position
148-151	SL	D axis velocity	260-263	SL	H axis velocity
152-153	SL	D axis torque	264-265	SL	H axis torque
154-155	SW or UW [1]	D axis analog input	266-267	SW or UW [1]	H axis analog input

[1] Will be either a Signed Word or Unsigned Word depending upon AQ setting when used with the DB-28045. 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).

General Status (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

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	(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 decel.	Latch is armed	Off on Error armed	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

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:

https://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:

https://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:

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

Chapter 5 Command Basics

Introduction

The DMC-21x5 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-21x5 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-21x5, or an entire group of commands can be downloaded into the DMC-21x5 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-21x5 instruction set and syntax. A summary of commands as well as a complete listing of all DMC-21x5 instructions is included in the *Command Reference*.

https://galil.com/download/comref/com2105/index.html

Command Syntax - ASCII

DMC-21x5 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-21x5 command interpreter.

NOTE: If a Galil terminal program is being used, commands will not be processed until an <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 command

```
PR 4000 <return>            Position relative
```

Implicit Notation

PR is the two character instruction for position relative. 4000 is the argument which represents the required position value in counts. The <return> terminates the instruction. The space between PR and 4000 is optional.

For specifying data for the A,B,C and D axes, commas are used to separate the axes. If no data is specified for an axis, a comma is still needed as shown in the examples below. If no data is specified for an axis, the previous value is maintained.

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-21x5 provides an alternative method for specifying data. Here data is specified individually using a single axis specifier such as A, B, C or D. An equals sign is used to assign data to that axis. 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

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

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

2185 For controllers with 5 or more axes, the axes are referred to as A,B,C,D,E,F,G,H. The specifiers X,Y,Z,W and A,B,C,D may be used interchangeably.

`BG ABCDEFGH`	Begin all axes
`BG D`	Begin D only

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-21x5 returns a : for valid commands and a ? for invalid commands.

For example, if the command BG is sent in lower case, the DMC-21x5 will return a ?.

`bg`	invalid command, lower case
`?`	DMC-21x5 returns a ?

When the controller receives an invalid command the user can request the error code. The error code will specify the reason for the invalid command response. To request the error code type the command TC**1**. For example:

`?TC1`	Tell Code command
`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-21x5 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-21x5 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=_TPA
```

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.

https://galil.com/download/comref/com2105/index.html

Chapter 6 Programming Motion

Overview

The DMC-21x5 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-2115 are single axis controllers and use A axis motion only. Likewise, the DMC-2125 use A and B, the DMC-2135 use A, B, and C, and the DMC-2145 use A,B,C, and D. The DMC-2155 use A,B,C,D, and E. The DMC-2165 use A,B,C,D,E, and F. The DMC-2175 use A,B,C,D,E,F, and G. The DMC-2185 use the axes A,B,C,D,E,F,G, and H.

2185 For controllers with 5 or more axes, the specifiers, ABCDEFGH, are used.

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
2 to 8 axis coordinated motion where path is described by linear segments.	Linear Interpolation Mode	LM, LI, LE, VS,VR, VA, VD
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.	Vector Mode: Linear and Circular Interpolation Motion with Tangent Motion:	VM, VP, CR, VS,VA,VD, TN, VE
Electronic gearing where slave axes are scaled to master axis which can move in both directions.	Electronic Gearing	GA, GD, **_GP**, GR, GM (if gantry)
Master/slave where slave axes must follow a master such as conveyer speed.	Electronic Gearing and 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	Motion Smoothing	IT
Smooth motion while operating in vector or linear interpolation positioning	Motion Smoothing	IT
Smooth motion while operating with stepper motors	Stepper Motion 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-21x5 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-21x5 profiler.

Note: The actual motor motion may not be complete when the profile has been completed, however, the next motion command may be specified.

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

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

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

The DMC-21x5 also allows use of single axis specifiers such as `PRB=2000`

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
MO ABC;                      'disable A, B, and C axes
EN;                          'end Program
```

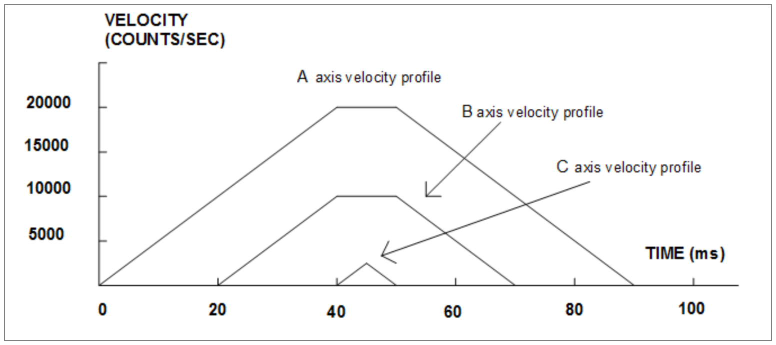


Figure 6.1: Velocity Profiles of ABC axes

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-21x5 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

Parameters can be set with individual axes specifiers such as `JGB=2000` (set jog speed for B axis to 2000).

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 A 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 A axis in this mode, the host would issue PT1 to the controller if both A and B 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 SP, AC, and DC 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.

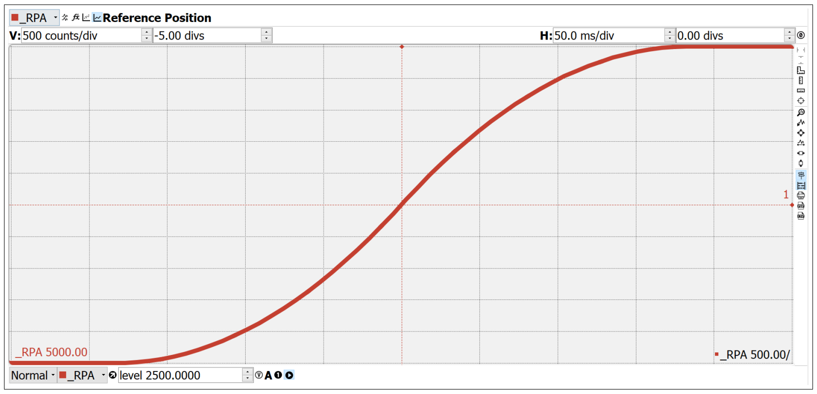


Figure 6.2: Position vs Time (msec) - Motion 1

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, shown in 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. In 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
```

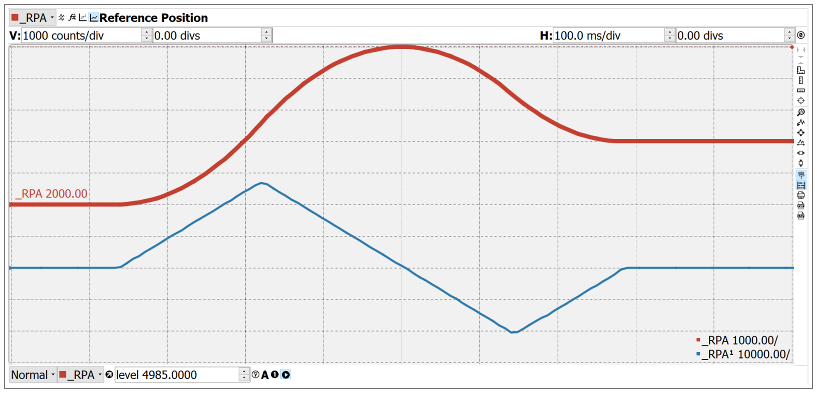


Figure 6.3: Position and Velocity vs Time(msec) for Motion 2

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.

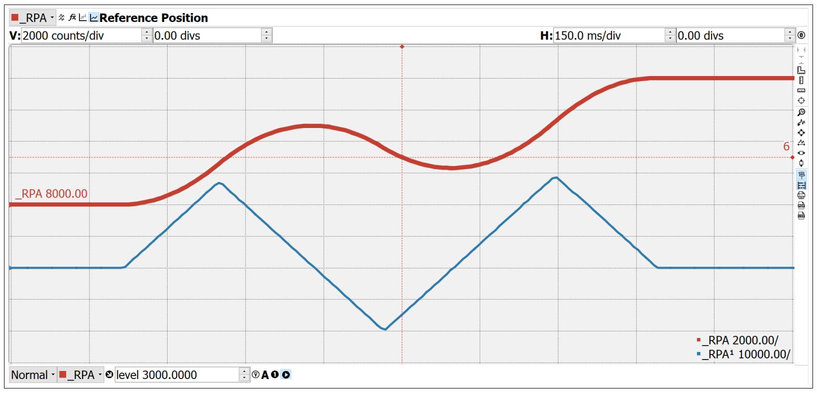


Figure 6.4: Position and Velocity vs Time (msec) for Motion 3

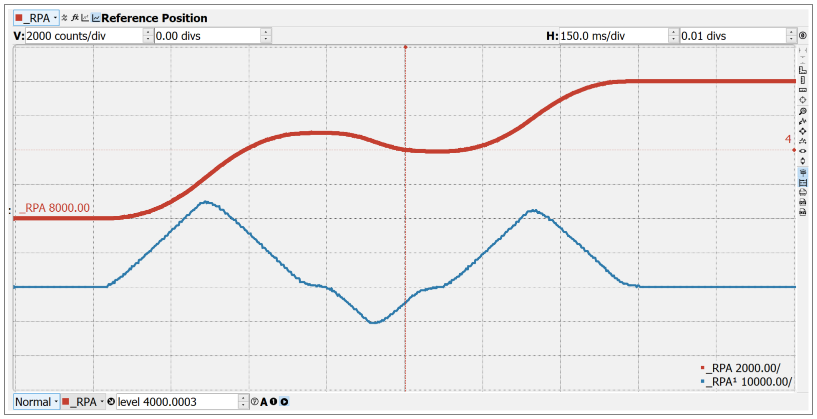


Figure 6.5: Position and Velocity vs Time (msec) for Motion 3 with IT 0.1

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 is allowed, however 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-21x5 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-21x5 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-21x5 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

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

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

The first command, < n, is equivalent to commanding VS**n** 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, > m, requires the vector speed to reach the value m at the end of the segment. Note that the function > 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 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.

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=5000. Suppose that when A=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 B=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 CD plane. Move to coordinates 40000,30000 counts at a vector speed of 100000 counts/sec and vector acceleration of 1000000 counts/sec^2.

```
#lmove;                    'label for program
SH CD;                     'enable C and D axes
LM CD;                     'specify axes for linear interpolation
LI,,40000,30000;           'specify CD 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 CD;                     'disable C and D 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.

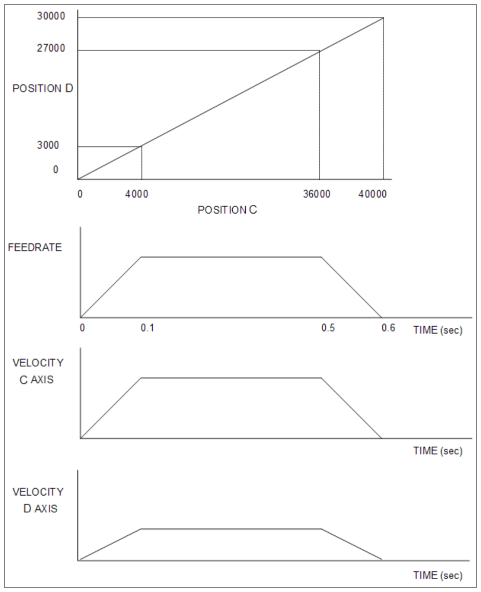


Figure 6.6: Linear Interpolation

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-21x5 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-21x5 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 command `VM mnp` where m and n are the coordinated pair of axes and p is the tangent axis

Specifying the Coordinate Plane

The DMC-21x5 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 to be zero for all movements in a sequence. **Note:** This local definition of zero does not affect the absolute coordinate system or subsequent coordinated motion sequences.

The 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-21x5 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. A zero means the buffer is full and no additional segments can be sent. As long as the buffer is not full, additional segments can be sent at PC bus speeds.

The 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 immediate command VS. 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 VSo 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-21x5 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-21x5 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-21x5 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=10,712.

The value of _CSS is 2.

_VPA, _VPB contain the coordinates of the point C.

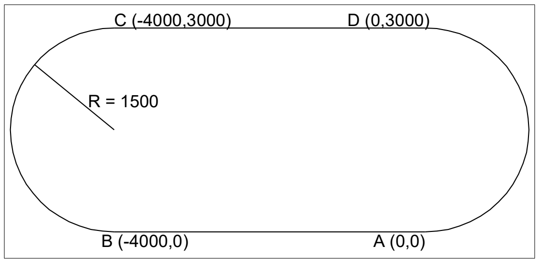


Figure 6.7: The Required Path

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
```

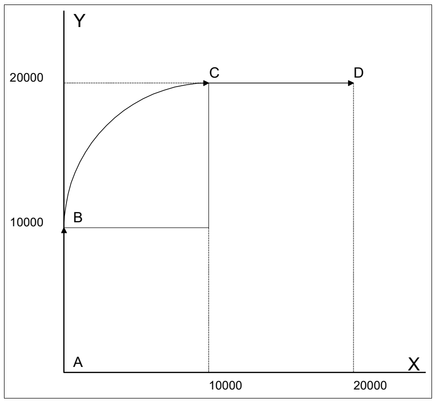


Figure 6.8: X-Y Motion Path

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	$\frac{R\lvert\Delta\theta\rvert 2\pi}{360}$ = 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

$$L_k=R_k\lvert\Delta\Theta_k\rvert 2\pi/360$$

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.

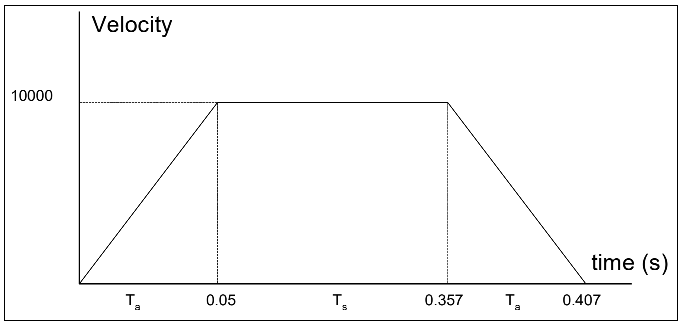


Figure 6.9: Vector Velocity Profile

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 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.

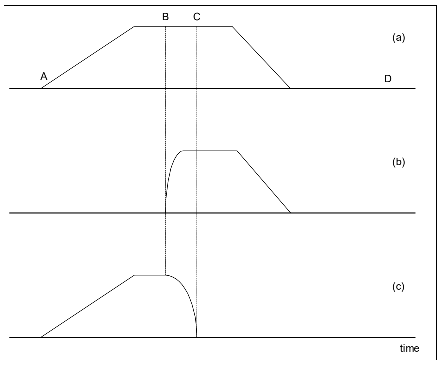


Figure 6.10: Vector and Axes Velocities

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. 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.

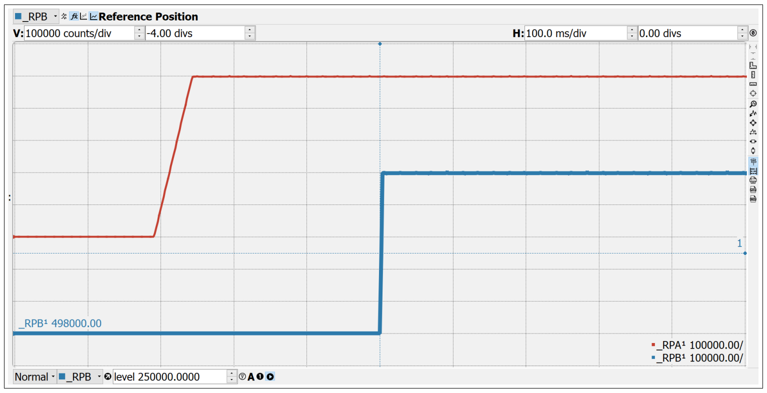


Figure 6.11: Velocity counts/sec vs. Time (msec) Instantaneous Gearing Engagement

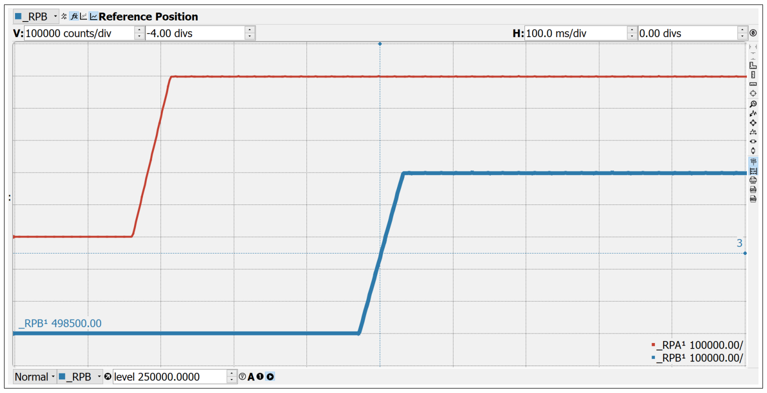


Figure 6.12: Velocity (counts/sec) vs. Time (msec) Ramped Gearing

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 is to 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-2135 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 will engage gearing gradually. Since the gearing is engaged over the interval of 6000 counts of the master, the slave 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. B is defined as the master. A, C, D are geared to 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-2135 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 follower are controlled by the DMC-21x5 controller, it may be desired to synchronize the follower with the commanded position of the master, rather than the actual position. This eliminates the coupling between the axes which may lead to oscillations.

For example, assume that a gantry is driven by 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 follower. 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 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 conveyor belts 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-2185 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 slave axis B, when the master is A. 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.

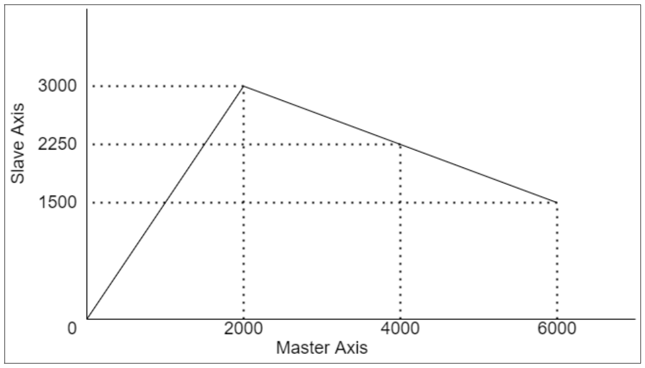


Figure 6.13: Electronic Cam Example

This disengages the slave 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:

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

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 slave 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 slave
AI -1;                     'wait for stop signal
EQ ,1000;                  'disengage slave
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, Z, and two slaves, A and Y.

```
    #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.

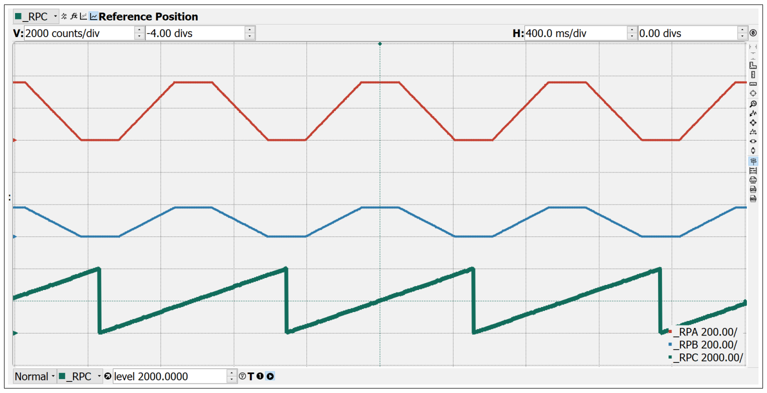


Figure 6.14: ECAM cycle with C axis as master

Contour Mode

The DMC-21x5 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 6.15. 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

The same trajectory may be represented by the increments

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

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

The programmed commands to specify the above example are:

```
#main
CM A;                      'specifies A axis for contour mode
CD 48=2;                   'first position increment and time interval, 2^2ms
CD 240=3;                  'second position increment and time interval, 2^3 ms
CD 48=4;                   'third position increment and time interval, 2^4 ms
CD 0=0;                    'end Contour buffer
#wait;                     'wait until path is done
WT 20
JP#wait,(_CM<>511);
EN;                        'end program
```

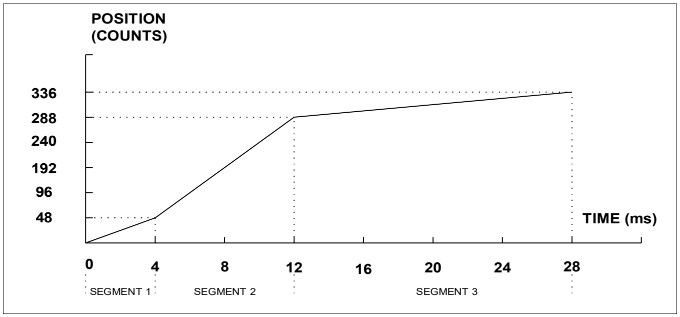


Figure 6.15: The Required Trajectory

Additional Commands

_CM gives the amount of space available in the contour buffer (511 maximum). Zero parameters for DT followed by zero parameters for CD will exit the contour mode.

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 ?.

Specifying a -1 for the DT or as the time interval in the CD command will pause the contour buffer.

Issuing the CM command will clear the contour buffer.

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
_CM	Amount of space left in contour buffer, 511 maximum

Table 6.12: List of commands related to Contour Mode

General Velocity Profiles

The Contour Mode is ideal for generating any arbitrary velocity profiles. The velocity profile can be specified as a mathematical function or as a collection of points. 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 6.16. 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:

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

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]

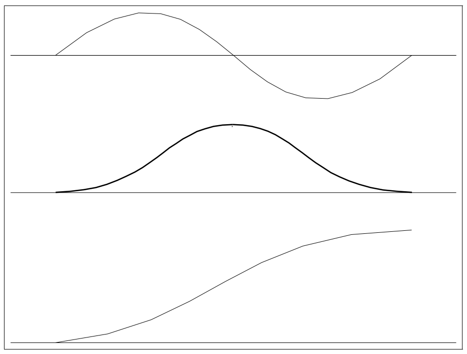


Figure 6.16: Velocity Profile with Sinusoidal Acceleration

The DMC-21x5 can compute trigonometric functions. However, the argument must be expressed in degrees. Using our example, the equation for X is written as:

X = 50T - 955 sin 3T

A complete program to generate the contour movement in this example is given below.

Contour Mode Example

```
#points;                    'label for program
DM pos[16];                 'dimension pos array
DM dif[15];                 'dimension dif array
c=0;                        'set c as index
t=0;                        't is time in ms
#a
v1=50*t
v2=3*t;                     'argument in degrees
v3=-955*@SIN[v2]+v1;        'compute position
v4=@INT[v3];                'integer value of v3
pos[c]=v4;                  'store in array pos
t=t+8
c=c+1
JP#a,(c<16)
#b;                         'program to find position differences
c=0
#c
d=c+1
dif[c]=pos[d]-pos[c];       'compute the difference and store
c=c+1
JP #c,(c<15)

#run;                       'program to run motor
CM A;                       'contour Mode
DT 3;                       '8 millisecond intervals
c=0
#e
CD dif[c];                  'contour distance is in dif
c=c+1
JP #e,(c<15)
CD 0=0;                     'end contour buffer
#wait;                      'wait until path is done
WT 20
JP #wait,(_CM<>511);
EN;                         'end the program
```

Teach (Record and Play-Back)

Several applications require teaching the machine a motion trajectory. Teaching can be accomplished using the DMC-21x5 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 6.13: 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;                     'end contour buffer
#wait;                      'wait until path is done
WT 20
JP #wait,(_CM<>511);
EN;                         'end program
```

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

Virtual Axis

The DMC-21x5 controller has a virtual axis designated as the N axis. This axis has no encoder and no DAC. However, it 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 axis 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 vector speed must be

VS=2π * 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:

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:

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.

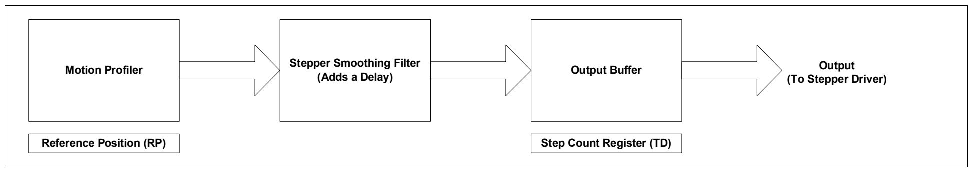


Figure 6.17: Flow Chart of how step pulses are generated

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

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

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 6.14: 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 6.15: 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 6.16: 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 step 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 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).

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 is set to 0 the axis stays in motion, if OE is set to 1 the axis is stopped.
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. If using a drive with a different step resolution, simply change the YA command.

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)
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-21x5 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 KP=0, KI=0 and to maximize the value of KD under the condition DV1. Once KD is found, increase KP gradually to a maximum value, and finally, increase KI, if necessary. 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-21x5 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 6.18 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
```

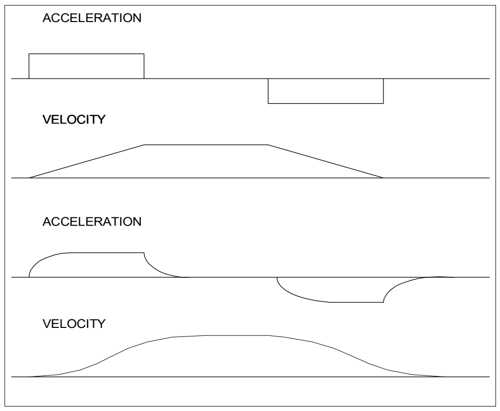


Figure 6.18: Acceleration and trapezoidal velocity without smoothing (above) and smoothed acceleration and trapezoidal velocity profiles (below)

Using the KS Command (Step Motor Smoothing):

When operating with step motors, motion smoothing can be accomplished with the command, KS. The KS command smoothes 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-21x5 defines the home position as the position at which the index was detected and sets the encoder reading at this point to zero.

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

			Direction of Motion		
Switch Type	CN Setting	Initial _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 6.17: 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
HV 1000;                  'slower speed for Stages 2 and 3
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 6.19 shows the velocity profile from the homing sequence of the example program above. For this profile, the switch is normally closed and CN,-1.

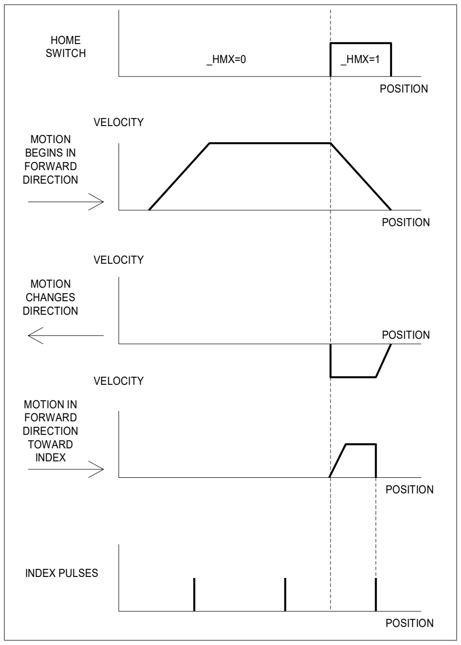


Figure 6.19: Homing Sequence for Normally Closed Switch and CN,-1

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 6.18: 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 6.19: 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 6.20, or on the index pulse for that axis. The position can be captured for either the main or auxiliary encoder within 25 microseconds of an high-to-low transition.

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 6.20: Inputs and corresponding axis latch

To insure 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-21x5 provides a powerful programming language that allows users to customize the controller for their particular application. Programs can be downloaded into the DMC-21x5 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-21x5 provides commands that allow the DMC-21x5 to make its own decisions. These commands include conditional jumps, event triggers and subroutines.

For greater programming flexibility, the DMC-21x5 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-21x5 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-21x5 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-21x5 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 a 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-21x5 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 Automatic Subroutines for Monitoring Conditions.

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)
#ININT	Input Interrupt subroutine (See II Command)
#LIMSWI	Limit Switch subroutine
#MCTIME	Timeout on Motion Complete trippoint
#POSERR	Position 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-21x5 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-21x5 can run up to 8 independent programs simultaneously. These programs are called threads and 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-21x5 provides commands and operands which are useful in debugging application programs. These commands include interrogation commands to monitor program execution, determine the state of the controller and the contents of the controllers program, array, and variable space. Operands also contain important status information which can help to debug a program.

Trace Commands

The trace command causes the controller to send each line in a program to the host computer immediately prior to execution. Tracing is enabled with the command, TR1. TR0 turns the trace function off.

NOTE: When the trace function is enabled, the line numbers as well as the command line will be displayed as each command line is executed.

Error Code Command

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

Stop Code Command

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

RAM Memory Interrogation Commands

For debugging the status of the program memory, array memory, or variable memory, the DMC-21x5 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-12115 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 15900 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-21x5

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 to BGA;AMA and re-download the program.

```
:XQ #a
```

Program Flow Commands

The DMC-21x5 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-21x5 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-21x5 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-21x5 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 Examples:

Event Trigger - Multiple Move Sequence

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

```
#twomove;                  '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-21x5 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-21x5 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-21x5 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-21x5 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-21x5 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-21x5 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-21x5 allows for IF conditional statements to be included within other IF conditional statements. This technique is known as 'nesting' and the DMC-21x5 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 ZS**1** command clears 1 level of the stack. This allows the program sequencer to continue to the next line. The ZS**0** 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-21x5 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-21x5 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)
#ININT	Input Interrupt subroutine (See II Command)
#LIMSWI	Limit Switch subroutine
#MCTIME	Timeout on Motion Complete trippoint
#POSERR	Position 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 - Limit Switch:

This program prints a message upon the occurrence of a limit switch. For the #LIMSWI routine to function, the DMC-21x5 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
```

This simple program will issue the message "A fell short" if the A axis does not reach the commanded position within 1 second of the end of the profiled move.

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

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-2115 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 serial port.

```
#main;                      'label for main of program
CI 2;                       'setup communication interrupt for serial port
MG {P1}"Type 0 to stop      'send message to serial port
motion";
MG {P1}"Type 1 to pause     'send message to serial port
motion";
MG {P1}"Type 2 to resume    'send message to serial 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,,                        'return sum
(^a+^b+^c+^d+^e+^f+^g+^h)

: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                     'first argument passed by reference
(&value,1,2,3,4,5,6,7);
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-21x5 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 smallest number representable (and thus the precision) 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-21x5 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-21x5 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-21x5 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-21x5 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.

Examples of valid and invalid variable names are:

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 (231) 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-21x5 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, variable=. For example, v1= , returns the value of the variable v1.

Example - Using Variables for Joystick

The example below reads the voltage of an X-Y joystick and assigns it to variables 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-21x5 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-21x5 command. The command reference indicates which commands have an associated operand.

Status commands such as Tell Position return actual values, whereas action commands such as KP or SP return the values in the DMC-21x5 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-21x5 provides a few additional operands which give access to internal variables that are not accessible by standard DMC-21x5 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, off by 2.4%, 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-21x5 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-21x5 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 indicates to begin 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                          'select arrays for capture
posA[],erA[],posB[],erB[];
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: “       'send message of recorded motor positions
,posA[i],posB[i];
MG “Position Error: “       'send message of recorded position errors
,erA[i],erB[i];
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-21x5 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-21x5 controller are stored with 4 bytes of integer and 2 bytes of fractional data.

Using Communication Interrupt

The DMC-21x5 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-21x5 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

A DMC-21x5 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 serial 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, reenable 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
```

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 serial port or {En} for the Ethernet port.

```
MG {P1} "Hello World";    'message sent through the serial 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 Serial 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-21x5 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 PF 10.0.

```
: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 VF 10.4 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-21x5 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-21x5 has an 8-bit uncommitted output port, the DMC-2155 through DMC-2185 has an additional 8 outputs. Each bit on the output port may be set and cleared with the commands 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-2115 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-21x5 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

Certain DMC-21x5 accessories provide the controller with eight analog inputs. See the Accessory 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
```

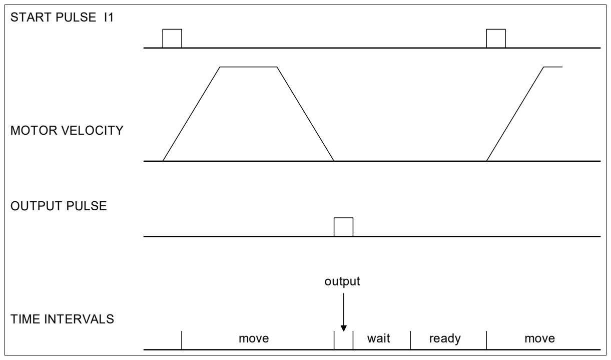


Figure 7.1: Motor Velocity and the Associated Input/Output signals

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.

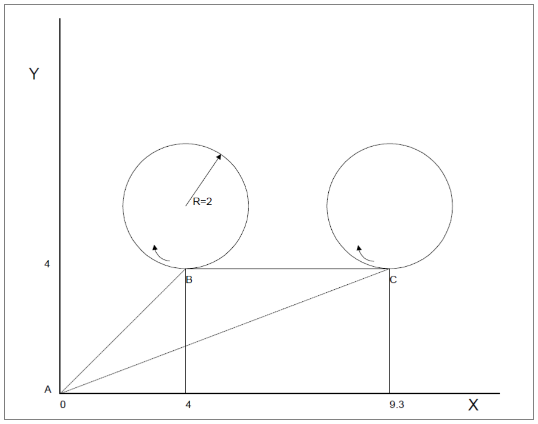


Figure 7.2: Motor Velocity and the Associated Input/Output signals

```
#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 count/sec

The program reads the input voltage periodically and calculates the necessary speed To get a speed of 200,000 ct/sec for 10 volts, select the speed as:

Speed = 20000 x @AN[]

The corresponding velocity for the motor is assigned to a variable.

```
#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-21x5 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-21x5 is an integral part of the machine, the engineer should design his overall system with protection against a possible component failure on the DMC-21x5. Galil shall not be liable or responsible for any incidental orf consequential damages.

Hardware Protection

The DMC-21x5 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 amp 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 Digital Inputs 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, the function of the Abort input can be changed with the ELO jumper. With the ELO jumper installed, the Abort input allows the user to shutdown the amplifier at a hardware level and the controller will automatically jump to the amplifier error routine, `#AMPERR`. For more detailed information on how specific Galil amplifiers behave when the ELO is triggered, see the amplifier section in the Accessory Components.

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-21x5 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-21x5. Along with position error and encoder failure detection, then DMC-21x5 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 OE1 or OE3
AEN Output Line	Switches to Motor Off state

The position error can be monitored during execution using the TE command.

Programmable Position Limits

The DMC-21x5 provides programmable forward and reverse position limits. These are set by the BL and FL commands. Once a position limit is specified, the DMC-21x5 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-21x5 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:

- The position error for the specified axis exceeds the limit set with the ER command
- A hardware limit is reached
- The abort command is given
- 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-21x5 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 for the. 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.

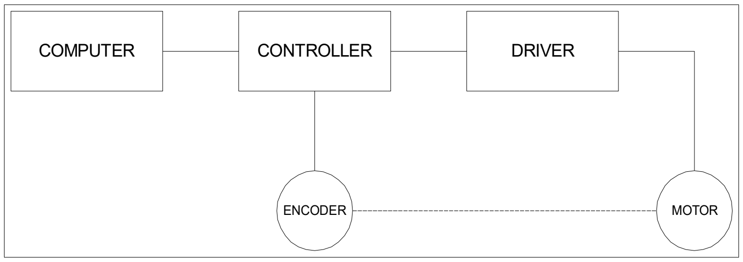


Figure 10.1: Elements of Servo Systems

The operation of such a system can be divided into three levels , the levels are:

1. Closing the Loop
2. Motion Profiling
3. Motion Programming

The first level, the closing of the loop, assures that the motor follows the commanded position. This is done by closing the position loop using a sensor. The operation at the basic level of closing the loop involves the subjects of modeling, analysis, and design. These subjects will be covered in the following discussions.

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

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

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.

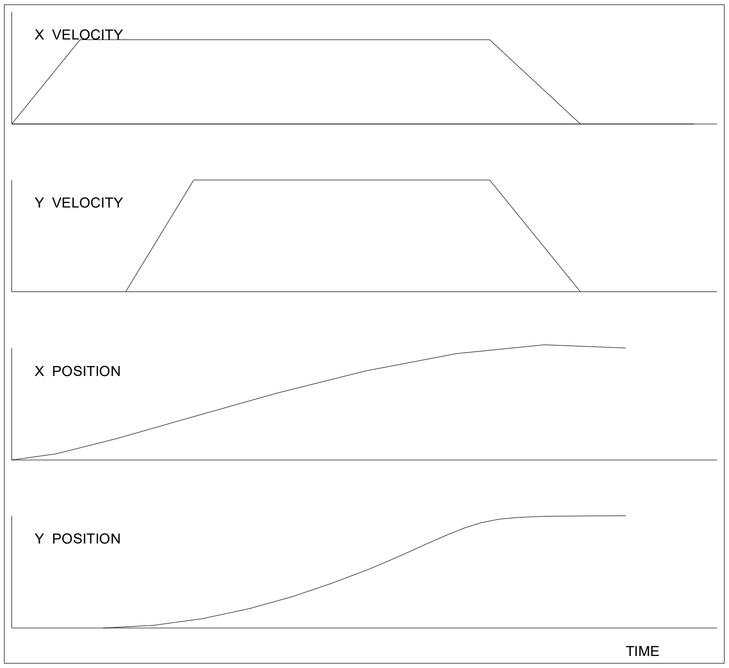


Figure 10.2: Velocity and Position Profiles

Operation of Closed-Loop Systems

To understand the operation of a servo system, it can be compared 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 the motor position needs to be commanded to move 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. The hot water faucet should be turned at the "right" rate. If it is turned 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 the faucet is turned too fast. The overreaction results in temperature oscillations. When the response of the system oscillates, the system is unstable. Clearly, unstable responses are bad when a constant level is desired.

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

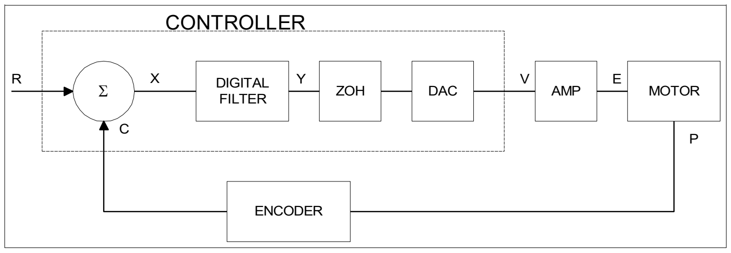


Figure 10.3: Functional Elements of a Motion Control System

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.

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

$$P/V = 1/[K_g s(sT_1+1)]$$

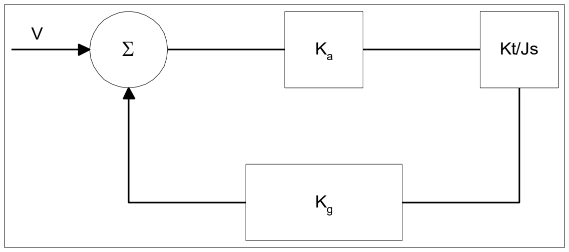


Figure 10.4: Elements of velocity loops

The resulting functions derived above are illustrated by the block diagram of Figure 10.5.

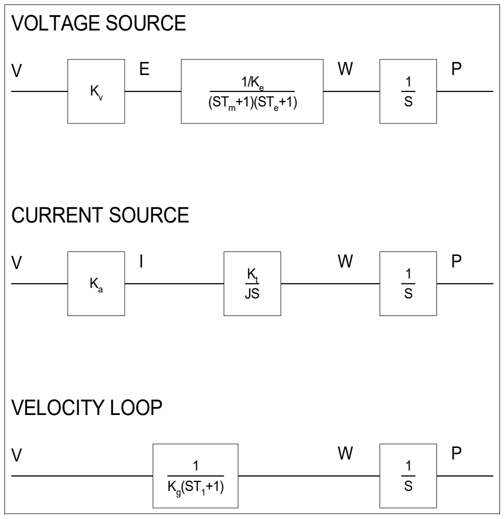


Figure 10.5: Mathematical model of the motor and amplifier in three operational modes

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 $$L(z) = \frac{1-B}{Z-B}$$

Notch $$N(z) = \frac{(Z-\bar{z})(Z-z)}{(Z-p)(Z-\bar{p})}$$

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).

$G(s) = (P + sD + I/s) \cdot a / (s + a)$

where,

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-21x5 are

KP = 16
KD = 144
KI = 2
PL = 0.75
T = 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

$G(s) = [16 + 0.144s + 2000/s] \cdot 250/ (s+250)$

The notch filter has two complex zeros, z and $\bar{z}$, and two complex poles, p and $\bar{p}$.

The effect of the notch filter is to cancel the resonance affect by placing the complex zeros on top of the resonance poles. The notch poles, P and 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, identify the resonance 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.

System Analysis

To analyze the system, 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-21x5 controller and the following parameters:

$K_t = 0.1$	Nm/A	Torque constant
$J = 2 * 10^{-4}$	kg.m^2	System moment of inertia
R = 2	Ω	Motor resistance
$K_a = 4$	Amp/Volt	Current amplifier gain
KP = 12.5		Digital filter gain
KD = 245		Digital filter zero
KI = 0		No integrator
N = 500	Counts/rev	Encoder line density
T = 1	ms	Sample period

The transfer function of the system elements are:

Motor

$M(s) = P/I = K_t/Js^2 = 500/s^2$ [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.

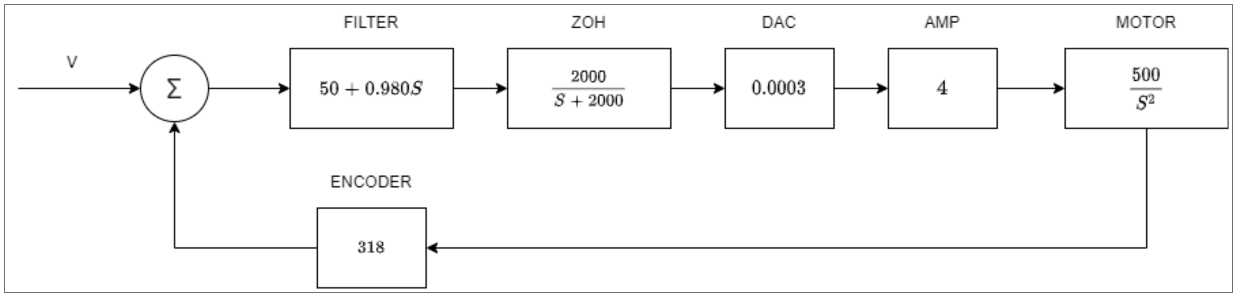


Figure 10.6: Mathematical model of the control system

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.

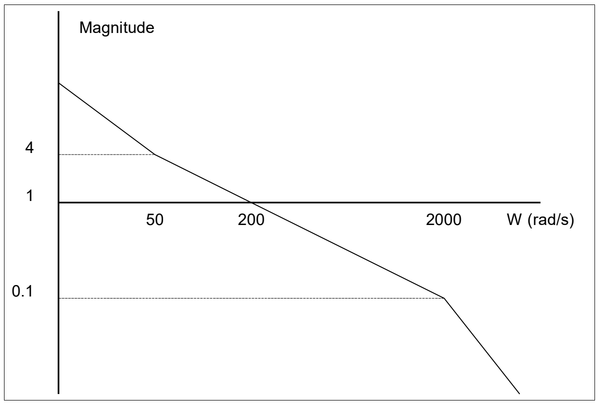


Figure 10.7: Bode plot of the open loop transfer function

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

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

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

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

$\alpha = 76° - 180° - 6° = -110°$

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.

System Design and Compensation

The closed-loop control system can be stabilized by a digital filter, which is preprogrammed in the DMC-21x5 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-21x5 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 $\omega_c = 500$.

$L(j500) = 3.17 \cdot 10^6/[(j500)^2\ (j500+2000)]$

This function has a magnitude of

$|L(j500)| = 0.00625$

and a phase

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

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

$|A(j500)| = 1$

$\mathrm{Arg}\ [A(j500)] = -135°$

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, a filter function G(s) needs to be selected 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 · 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-21x5 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-21x5

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

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 index 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

Digital Inputs: DI[16:1]*, Limit switches, home, Abort, reset	TTL, 0-5 Volts Pulled up internally to 5V through a 4.7kΩ resistor *[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]*	TTL, 0-5 Volts 20 mA sink/source *[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

Performance Specifications

Minimum Servo Loop Update Time	
DMC-2115 through DMC-2185	250 µsec
Position Accuracy	±1 quadrature count
Velocity Accuracy	
Long Term	Phase-locked, better than 0.005%
Short Term	System dependent
Position Range	±2,147,483,647 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	±2,147,483,647
Variable Resolution	1×10^{-4}
Number of Variables	510
Array Size	24,000 elements, 30 arrays
Program Size	4,000 lines x 80 characters

Ordering Options

Overview

The DMC-21x5 can be ordered in many different configurations and with different options. This section provides information regarding the different options available on the DMC-21x5 motion controller and its accessory components. For information on pricing and how to order the controller with these options, see the DMC-21x5 part number generator on our website.

https://galil.com/order/part-number-generator/dmc-21x5

DMC, "DMC-21x5(Y)" Options

The following options are the "Y" options that can be added to the DMC-21x5 part number. Multiple Y options can be ordered per controller board.

DIN – DIN Rail Mounting

The DIN option on the DMC-21x5 motion controller provides DIN rail mounts on the bottom of the controller. This will allow the controller to be mounted to any standard DIN rail.

Part number ordering example:	DMC-2135(DIN)

UP – Upward Facing 96 Pin DIN Connector

The UP option on the DMC-21x5 controller is on by default. The 96 pin DIN connector is on top of the board facing up.

Part number ordering example:	DMC-2165(UP,V,HP)

DOWN – Downward Facing 96 Pin DIN Connector

The DOWN option on the DMC-21x5 controller changes the location and orientation of the 96 pin DIN connector. By default, the connector is on top of the board facing up. With the DOWN option, the connector is moved to the bottom of the board facing down.

Part number ordering example:	DMC-2165(DOWN,V,HP)

V – Vertical RS232, Status Lights, and Ethernet

The V option on the DMC-21x5 is on by default. The communication ports and LED's are facing vertically.

Part number ordering example:	DMC-2165(UP,V,HP)

H – Horizontal RS232, Status Lights, and Ethernet

The H option on the DMC-21x5 changes the orientation of the of the RS232 and Ethernet ports as well as the power and connection LED's. By default, the communication ports and LED's are facing vertically. With the H option, the ports and LED's face horizontally.

Part number ordering example:	DMC-2165(UP,H,HP)

VP – Vertical Power Connector

The VP option on the DMC-21x5 is on by default. The power connector faces vertically from the board.

Part number ordering example:	DMC-2165(UP,H,VP)

HP – Horizontal Power Connector

The HP option on the DMC-21x5 changes the orientation of the 6 pin power connector. By default, the power connector faces vertically from the board. With the HP option, the power connector faces horizontally.

Part number ordering example: DMC-2165(UP,H,HP)

RA – Right Angle 96 Pin Connector

The RA option on the DMC-21x5 changes the orientation of the 96 pin DIN connector. By default, the connector is on facing vertically. With the RA option, the connector faces horizontally.

Part number ordering example: DMC-2125(RA)

MO – Motor Off Jumper Installed

The MO option on the DMC-21x5 changes the state of the axes when the controller powers on. By default, all axes are in a servo here (SH) state and in closed-loop control. With the MO option, all axes will be in a motor off (MO) state when the controller powers on.

Part number ordering example: DMC-2145(MO)-AMP-20545

TRES – Termination Resistors on Encoder Inputs

The TRES option on the DMC-21x5 adds an additional component to all of the main an auxiliary encoder inputs. By default, the controller has differential encoder inputs that can also be used with single ended encoders. With the TRES option, a 120 Ω resistor is placed between the positive and negative differential inputs on the A and B channels for each main and auxiliary encoder input. This gives the inputs more immunity to noise, however the inputs can no longer be used with single ended encoders.

Part number ordering example: DMC-2135(TRES)

D24 and DC48 – DC to DC Converter

The DC to DC option on the DMC-21x5 changes how the controller is powered. By default, the controller requires +5V, +12V, and -12V supply voltage. With the DC to DC option, only one supply voltage is required to power the controller. The DC24 option allows the controller to be powered with a single 24V supply. The DC48 option allows the controller to be powered with a single 48V supply.

Part number ordering example: DMC-2135-DC24

Accessory Specific Options

The following options can be added to accessory part numbers. Often, multiple options can be ordered per bank of 4 axis as long as they are separated by a comma.

ISCNTL – Isolated Controller Power

The ISCNTL option on the DMC-21x5 changes how the controller is powered when using an amplifier or stepper driver accessory. By default, power is passed from the accessory to the controller; only one power supply is needed to provide power to the unit. With the ISCNTL option, power is no longer passed through to the controller. Two power supplies are needed to power the unit, one for the accessory and one for the controller.

Part number ordering example: DMC-2145(ISCNTL)-AMP-20545

SSR – Solid State Relay on AMP-20341

The SSR option on the AMP-20341 changes how current is driven to the motors while axes are in a motor off (MO) state. The AMP-20341 is a linear amplifier that requires a bipolar power supply. It is possible that the plus and minus voltages rise at different rates during power up. This can be seen as an offset on the amplifier and could cause the motors to jump during power up. With the SSR option, a solid state relay is added and will eliminate any jump in the motors.

Part number ordering example: DMC-2145-AMP-20341(SSR)

LAEN – Low Amplifier Enable Configuration on ICM-20100 and ICM-20105

The LAEN option changes the external amplifier enable signal configuration of the controller when used with the ICM-20100 or ICM-20105. By default, the controller's amplifier enable signal with either ICM is 5V High-Amp-Enable (HAEN). With the LAEN option, the configuration will be changed to 5V Low-Amp-Enable(LAEN).

Part number ordering example: DMC-2145-ICM-20105(LAEN)

24V – 24V Amplifier Enable Configuration on ICM-20105

The 24V option changes the external amplifier enable signal configuration of the controller when used with the ICM-20105. By default, the controller's amplifier enable signal with either ICM is 5V High-Amp-Enable (HAEN). With the 24V option, the configuration will be changed to 24V HAEN.

Part number ordering example: DMC-2145-ICM-20105(24V)

BOX – Metal Enclosure on Unit with ICM-20105

The BOX option changes the form and fit of the controller when ordered with the ICM-20105. By default, the controller is exposed with the ICM-20105 installed on the first bank of axes. With the BOX option, a metal enclosure is installed on the controller and ICM.

Part number ordering example: DMC-2145-ICM-20105(BOX)

5V – 5V Logic for Extended I/O on DB-28045

The 5V option changes the logic level of the TTL extended I/O on the DB-28045. By default, the extended I/O is configured for 3.3V. With the 5V option, the extended I/O is configured for 5V.

Part number ordering example: DMC-2145-DB-28045(5V)

16BIT – 16Bit Resolution on Analog Inputs on DB-28045

The 16BIT option changes the resolution of the analog inputs provided by the DB-28045. By default, the analog inputs have a resolution of 12 bits. With the 16BIT option, the resolution is increased to 16 bits.

Part number ordering example: DMC-2135-DB-28045(16BIT)

SHUNT – SR-19900 Shunt Regulator on AMP-20440 and AMP-20545

The 16BIT option changes the resolution of the analog inputs provided by the DB-28045. By default, the analog inputs have a resolution of 12 bits. With the 16BIT option, the resolution is increased to 16 bits.

Part number ordering example: DMC-2135-DB-28045(16BIT)

Power Connector Part Numbers

Overview

The DMC-21x5 uses Molex connectors for connecting DC Power to the Amplifiers, Controller, and Motors. This section gives the specifications of these connectors. For specific Galil amplifier information , refer to the dedicated amplifier/driver section in Accessory Components.

The type of connectors on any given controller will be determined by how the controller is powered. Table A.1 indicates the type of Molex Connector used.

Power to Controller	On Board Connector	Terminal Pins
6 pin, 5V, +/-12V Power	MOLEX# 26-03-4061	MOLEX# 08-50-0189
4 pin, DC24 or DC48 Power	MOLEX# 26-03-4041	MOLEX# 08-50-0189

Table A.1: Molex connector part numbers for different power options

Pin-outs

J4 – Axes A-D 96 Pin DIN Connector

Pin#	Description	Pin#	Description	Pin#	Description
1	Digital Ground	33	Digital Ground	65	Digital Ground
2	Step D Axis	34	Direction D Axis	66	Motor Command D Axis
3	Step C Axis	35	Direction C Axis	67	Motor Command C Axis
4	Step B Axis	36	Direction B Axis	68	Motor Command B Axis
5	Step A Axis	37	Direction A Axis	69	Motor Command A Axis
6	Amplifier Enable D Axis	38	Digital Ground	70	Output Compare
7	Amplifier Enable A Axis	39	Amplifier Enable B Axis	71	Amplifier Enable C Axis
8	Home Switch D Axis	40	Reverse Limit Switch D Axis	72	Forward Limit Switch D Axis
9	Home Switch C Axis	41	Reverse Limit Switch C Axis	73	Forward Limit Switch C Axis
10	Home Switch B Axis	42	Reverse Limit Switch B Axis	74	Forward Limit Switch B Axis
11	Home Switch A Axis	43	Reverse Limit Switch A Axis	75	Forward Limit Switch A Axis
12	Digital Input 1 / Latch A Axis	44	Digital Input 2 / Latch B Axis	76	Digital Input 3 / Latch C Axis
13	Digital Input 4 / Latch D Axis	45	Digital Input 5	77	Digital Input 6
14	Digital Input 7	46	Digital Input 8	78	Abort
15	Digital Output 3	47	Digital Output 2	79	Digital Output 1
16	Digital Output 5	48	Digital Ground	80	Digital Output 4
17	Digital Output 8	49	Digital Output 7	81	Digital Output 6
18	A+ Main Encoder Input A Axis	50	A- Main Encoder Input A Axis	82	B+ Main Encoder Input A Axis
19	B- Main Encoder Input A Axis	51	I+ Index Pulse Input A Axis	83	I- Index Pulse Input A Axis
20	A+ Main Encoder Input B Axis	52	A- Main Encoder Input B Axis	84	B+ Main Encoder Input B Axis
21	B- Main Encoder Input B Axis	53	I+ Index Pulse Input B Axis	85	I- Index Pulse Input B Axis
22	A+ Main Encoder Input C Axis	54	A- Main Encoder Input C Axis	86	B+ Main Encoder Input C Axis
23	B- Main Encoder Input C Axis	55	I+ Index Pulse Input C Axis	87	I- Index Pulse Input C Axis
24	A+ Main Encoder Input D Axis	56	A- Main Encoder Input D Axis	88	B+ Main Encoder Input D Axis
25	B- Main Encoder Input D Axis	57	I+ Index Pulse Input D Axis	89	I- Index Pulse Input D Axis
26	Digital Ground	58	Digital Ground	90	Digital Ground
27	A+ Aux Encoder Input A Axis	59	A- Aux Encoder Input A Axis	91	B+ Aux Encoder Input A Axis
28	B- Aux Encoder Input A Axis	60	A+ Aux Encoder Input B Axis	92	A- Aux Encoder Input B Axis
29	B+ Aux Encoder Input B Axis	61	B- Aux Encoder Input B Axis	93	A+ Aux Encoder Input C Axis
30	B+ Aux Encoder Input C Axis	62	A+ Aux Encoder Input D Axis	94	Error Output
31	-12V	63	Reset Input	95	+12V
32	+5V	64	+5V	96	+5V

J5 – Axes E-H 96 Pin DIN Connector

Pin#	Description	Pin#	Description	Pin#	Description
1	Digital Ground	33	Digital Ground	65	Digital Ground
2	Step H Axis	34	Direction H Axis	66	Motor Command H Axis
3	Step G Axis	35	Direction G Axis	67	Motor Command G Axis
4	Step F Axis	36	Direction F Axis	68	Motor Command F Axis
5	Step E Axis	37	Direction E Axis	69	Motor Command E Axis
6	Amplifier Enable H Axis	38	Digital Ground	70	Output Compare
7	Amplifier Enable E Axis	39	Amplifier Enable F Axis	71	Amplifier Enable G Axis
8	Home Switch H Axis	40	Reverse Limit Switch H Axis	72	Forward Limit Switch H Axis
9	Home Switch G Axis	41	Reverse Limit Switch G Axis	73	Forward Limit Switch G Axis
10	Home Switch F Axis	42	Reverse Limit Switch F Axis	74	Forward Limit Switch F Axis
11	Home Switch E Axis	43	Reverse Limit Switch E Axis	75	Forward Limit Switch E Axis
12	Digital Input 9 / Latch E Axis	44	Digital Input 10 / Latch F Axis	76	Digital Input 11 / Latch G Axis
13	Digital Input 12 / Latch H Axis	45	Digital Input 13	77	Digital Input 1
14	Digital Input 15	46	Digital Input 16	78	Abort
15	Digital Output 11	47	Digital Output 10	79	Digital Output 9
16	Digital Output 13	48	Digital Ground	80	Digital Output 12
17	Digital Output 16	49	Digital Output 15	81	Digital Output 14
18	A+ Main Encoder Input E Axis	50	A- Main Encoder Input E Axis	82	B+ Main Encoder Input E Axis
19	B- Main Encoder Input E Axis	51	I+ Index Pulse Input E Axis	83	I- Index Pulse Input E Axis
20	A+ Main Encoder Input F Axis	52	A- Main Encoder Input F Axis	84	B+ Main Encoder Input F Axis
21	B- Main Encoder Input F Axis	53	I+ Index Pulse Input F Axis	85	I- Index Pulse Input F Axis
22	A+ Main Encoder Input G Axis	54	A- Main Encoder Input G Axis	86	B+ Main Encoder Input G Axis
23	B- Main Encoder Input G Axis	55	I+ Index Pulse Input G Axis	87	I- Index Pulse Input G Axis
24	A+ Main Encoder Input H Axis	56	A- Main Encoder Input H Axis	88	B+ Main Encoder Input H Axis
25	B- Main Encoder Input H Axis	57	I+ Index Pulse Input H Axis	89	I- Index Pulse Input H Axis
26	Digital Ground	58	Digital Ground	90	Digital Ground
27	A+ Aux Encoder Input E Axis	59	A- Aux Encoder Input E Axis	91	B+ Aux Encoder Input E Axis
28	B- Aux Encoder Input E Axis	60	A+ Aux Encoder Input F Axis	92	A- Aux Encoder Input F Axis
29	B+ Aux Encoder Input F Axis	61	B- Aux Encoder Input F Axis	93	A+ Aux Encoder Input G Axis
30	B+ Aux Encoder Input G Axis	62	A+ Aux Encoder Input H Axis	94	Error Output
31	-12V	63	Reset Input	95	+12V
32	+5V	64	+5V	96	+5V

JP6 – C-D Auxiliary Encoders IDC Pins

Pin #	Signal
1	+5V
2	Digital Ground
3	A+ Aux Encoder Input C Axis
4	A- Aux Encoder Input C Axis
5	B+ Aux Encoder Input C Axis
6	B- Aux Encoder Input C Axis
7	A+ Aux Encoder Input D Axis
8	A- Aux Encoder Input D Axis
9	B+ Aux Encoder Input D Axis
10	B- Aux Encoder Input D Axis

JP8 – G-H Auxiliary Encoders IDC Pins

Pin #	Signal
1	+5V
2	Digital Ground
3	A+ Aux Encoder Input G Axis
4	A- Aux Encoder Input G Axis
5	B+ Aux Encoder Input G Axis
6	B- Aux Encoder Input G Axis
7	A+ Aux Encoder Input H Axis
8	A- Aux Encoder Input H Axis
9	B+ Aux Encoder Input H Axis
10	B- Aux Encoder Input H Axis

RS-232 Serial Port (Male)

Standard connector and cable, 9Pin

Pin #	Signal
1	CTS
2	TXD
3	RXD
4	RTS
5	GND
6	CTS
7	RTS
8	CTS
9	NC

Ethernet (RJ45)

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.

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.
Forward Limit Switch	When active, inhibits motion in forward direction. Also causes execution of limit switch subroutine, **#LIMSWI**. The polarity of the limit switch may be set with the CN command.
Reverse Limit Switch	When active, inhibits motion in reverse direction. Also causes execution of limit switch subroutine, **#LIMSWI**. The polarity of the limit switch may be set with the CN command.
Home Switch	Input for Homing (HM) and Find Edge (FE) instructions. Upon BG following HM or FE, the motor accelerates to slew speed. A transition on this input will cause the motor to decelerate to a stop. The polarity of the Home Switch may be set with the CN command.
Input 1 - Input 8 Input 9 - Input 16	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	Uncommitted outputs. May be designated by the user to 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)

https://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: https://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.

Accessory Components

Overview

When ordered, the following components will reside on top of the DMC-21x5 motion controller. The amplifiers and stepper drivers provide power to the motors in the system while the interconnect modules and daughter board provide the connections for input and output signals.

A1 – AMP-20341

The AMP-20341 is a four axis, linear amplifier capable of driving small brushed motors with up to 1 Amp per axis. It requires a ± 12-30 V

A2 – AMP-20440

The AMP-20440 is a four axis amplifier that can operate brushed motors with up to 3 Amps per axis.

A3 – AMP-20545/20525

The AMP-20545 (four axis) and AMP-20525 (two axis) are capable of driving brushed and brushless motors with up to 10 Amps.

A4 – SDM-20242

The SDM-20242 is a user configurable microstepping driver. It is capable of driving up to 1.4 Amps/Phase to four bipolar two phase stepper motors.

A5 – SDM-20645

The SDM-20645 is a 1/64 microstepping driver. It is capable of driving up to 3 Amps/Phase to four bipolar two phase stepper motors.

A6 – ICM-20100

The ICM-20100 provides D-Sub connectors for easy access to signals for system elements such as amplifiers, encoders, digital inputs, and digital outputs.

A7 – ICM-20105

The ICM-20105 provides D-Sub connectors for easy access to signals for system elements such as amplifiers and encoders. The digital inputs and outputs are optoisolated.

A8 – DB-28045

The DB-28045 is an I/O daughter board that provides access to an additional 40 bits of configurable I/O as well as eight analog inputs.

A9 – SR-19900

The SR-19900 is a shunt regulator that has two default voltage thresholds of 33V or 66V. A user defined voltage threshold can be set by changing a simple resistor.

A1 – AMP-20341

Description

The AMP-20341 is a linear amplifier for operating small 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.

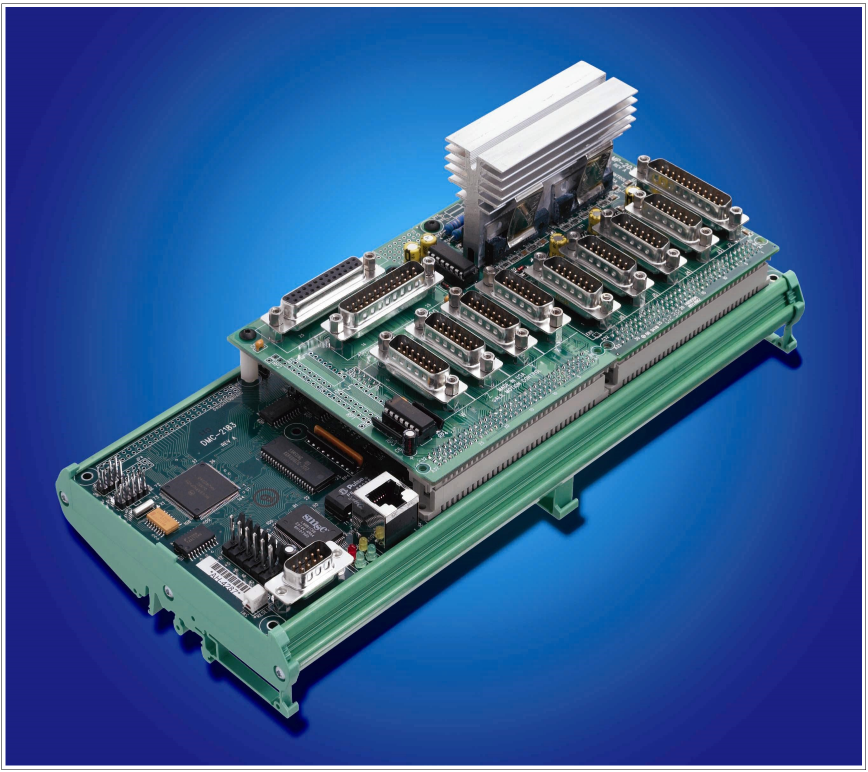


Figure A1.1: ICM-20100 (left) and AMP-20341 (right) mounted on DMC-2185-DIN

Electrical Specifications

	Servo (Brushless/Brushed)
Supply Voltage	± 12-30V
Peak Current	1 Amp
Amplifier Gain	0.1 Amps/Volt

Table A1.1: Amplifier Electrical Specifications

Mating Connectors

	On Board Connector	**Mating Connector**	**Terminal Pins**
POWER	Molex 26-48-1035	Molex 26-03-4030	Molex 08-50-0189
A,B,C,D: Motor Power Connectors	Molex 22-23-2021	Molex 22-01-3027	Molex 08-50-0114

Table A1.2: Connector and terminal pin Molex part numbers

Power Connector	
Pin Number	**Connection**
1	+VS (DC Power)
2	Ground
3	-VS (DC Power)
Motor Connector	
Pin Number	**Brushed**
1	Phase+
2	Phase-

Table A1.3: Power and Motor Connector Pin-outs

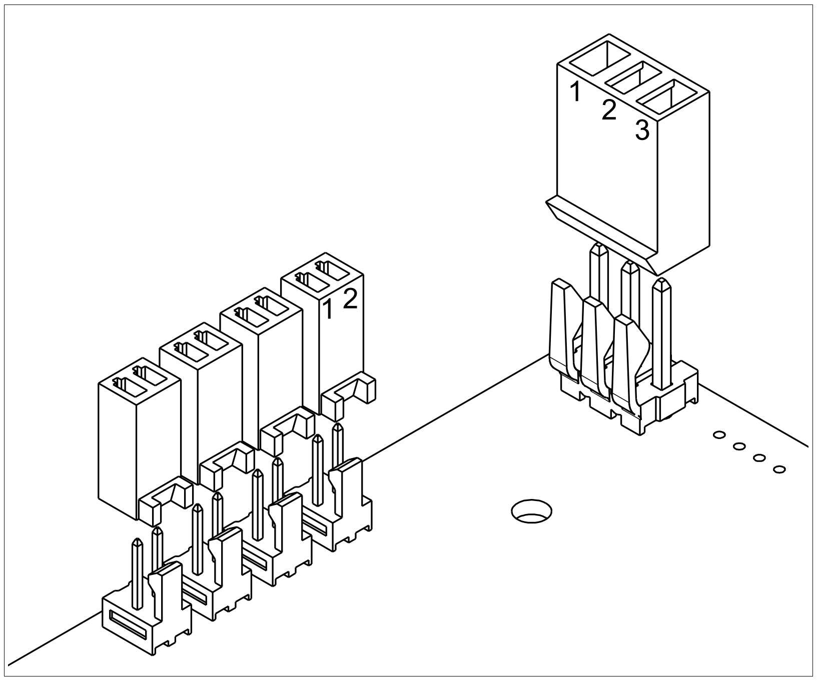


Figure A1.2: Power and Motor Connector Pin-outs

Pin-outs

J2 – I/O 25 pin D-Sub Connector (Female)

Pin #	Description	Pin #	Description
1	Digital Ground	14	+5V
2	Digital Input 1 / Latch A Axis	15	Digital Input 2 / Latch B Axis
3	Digital Input 3 / Latch C Axis	16	Digital Input 4 / Latch D Axis
4	Digital Input 5	17	Digital Input 6
5	Digital Input 7	18	Digital Input 8
6	Abort Input	19	Output Compare
7	Digital Output 1	20	Digital Output 2
8	Digital Output 3	21	Digital Output 4
9	Digital Output 5	22	Digital Output 6
10	Digital Output 7	23	Digital Output 8
11	Digital Ground	24	+5V
12	Reset Input	25	Error Output
13	No Connect		

Jm – Encoder 15 pin D-Sub Connector (Female)

Pin #	Description
1	Forward Limit Switch
2	Home Switch
3	+5V
4	A- Main Encoder Input
5	B- Main Encoder Input
6	I- Index Pulse Input
7	A- Aux Encoder Input
8	B- Aux Encoder Input
9	Reverse Limit Switch
10	Digital Ground
11	A+ Main Encoder Input
12	B+ Main Encoder Input
13	I+ Index Pulse Input
14	A+ Aux Encoder Input
15	B+ Aux Encoder Input

J8 – External Amplifier 10 pin Header

Pin #	Description
1	Amplifier Enable A Axis
2	Motor Command A Axis
3	Amplifier Enable B Axis
4	Motor Command B Axis
5	Amplifier Enable C Axis
6	Motor Command C Axis
7	Amplifier Enable D Axis
8	Motor Command D Axis
9	Digital Ground
10	Digital Ground

Servo Motor Operation

Brushed Motor Operation

For brushed motor operation, set MT and BR to 1 for the axis.

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 A1.3. 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 A1.3.

```
#tk;                          'begin program
MO;                           'disable all axes
TLA=5;                        'set maximum continuous torque limit of 5 V
TKA=9.9982;                   'set maximum peak torque limit of 9.9982 V
SH A;                         'enable a axis
OFA=9.9982;                   'command torque signal to 9.9982 V
EN;                           'end program
```

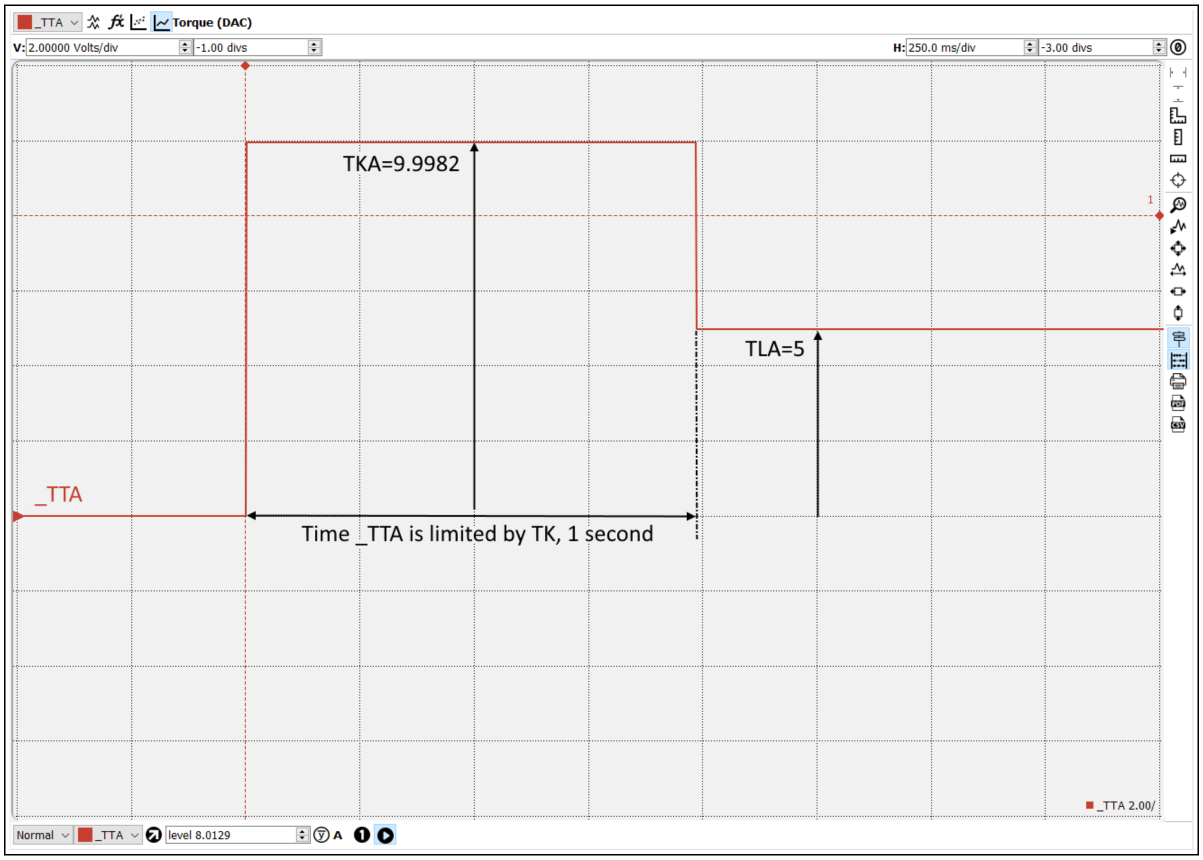


Figure A1.3: Peak Current Operation

A2 – AMP-20440

Description

The AMP-20440 is a transconductance switching amplifier capable of driving 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.

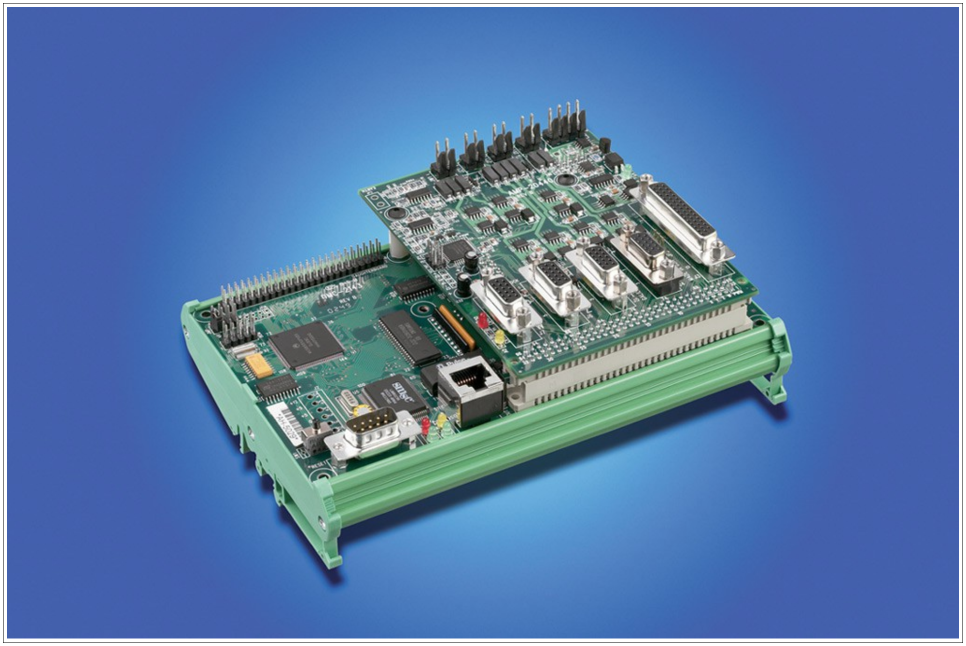


Figure A2.1: AMP-20440 mounted on DMC-2145-DIN

Electrical Specifications

	Servo (Brushless/Brushed)
Supply Voltage	18-60V
Continuous/Peak Current	3.3 Amps
Amplifier Gains	0.33 Amps/Volt
Switching Frequency	60 kHz
Minimum Load Inductance	0.5 mH

Table A2.1: Amplifier Electrical Specifications

Mating Connectors

	On Board Connector	Mating Connector	Terminal Pins
POWER	AMP 640445-4	AMP 770849-4	AMP 3-770476-1
A,B,C,D: Motor Power Connectors	Molex 26-48-1045	Molex 26-03-4020	Molex 08-50-0189

Table A2.2: Connector and terminal pin Molex part numbers

Power Connector	
Pin Number	**Connection**
1,3	DC Power Supply Ground
2,4	Ground
Motor Connector	
Pin Number	**Brushed**
1	Phase+
2	Phase-

Table A2.3: Power and Motor Connector Pin-outs

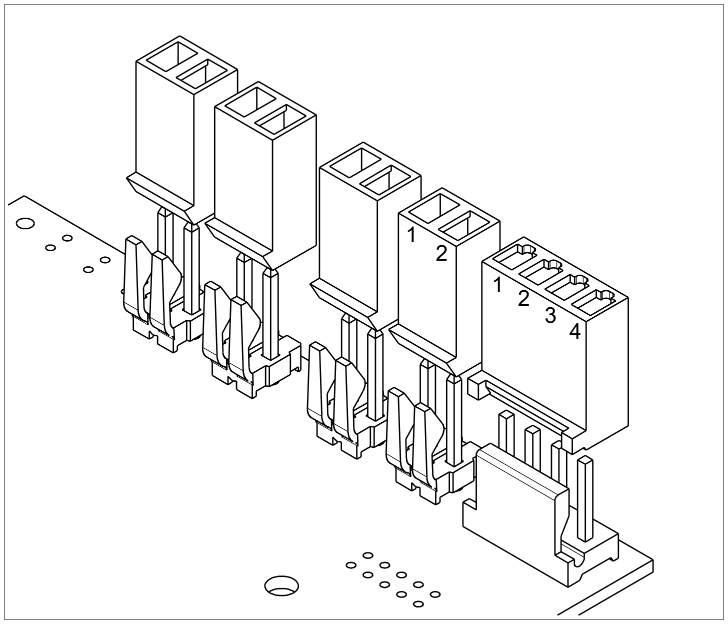


Figure A2.2: Power and Motor Connector Pin-outs

Pin-outs

J2 – I/O 44 pin HD D-Sub Connector (Female)

Pin #	Description	Pin #	Description
1	No Connect	23	Digital Input 4 / Latch D Axis
2	Digital Output 6	24	Digital Input 1 / Latch A Axis
3	Digital Output 8	25	No Connect
4	Digital Output 5	26	Motor Command A Axis
5	Digital Output 2	27	Motor Command B Axis
6	Abort Input	28	Motor Command C Axis
7	Digital Input 6	29	Motor Command D Axis
8	Digital Input 3 / Latch C Axis	30	Error Output
9	Amplifier Enable B Axis	31	No Connect
10	Output Compare	32	+5V
11	Direction A Axis	33	+5V
12	Direction B Axis	34	Digital Ground
13	Direction C Axis	35	Digital Ground
14	Direction D Axis	36	Digital Input 8
15	Step D Axis	37	Digital Input 5
16	Amplifier Enable D Axis	38	Digital Input 2 / Latch B Axis
17	Amplifier Enable C Axis	39	No Connect
18	Digital Output 7	40	Amplifier Enable A Axis
19	Digital Output 4	41	Step A Axis
20	Digital Output 1	42	Step B Axis
21	Digital Output 3	43	Step C Axis
22	Digital Input 7	44	Step D Axis

Jm1 – Encoder 15 pin HD D-Sub Connector (Female)

Pin #	Description
1	I+ Index Pulse Input
2	B+ Main Encoder Input
3	A+ Main Encoder Input
4	B+ Aux Encoder Input
5	Digital Ground
6	I- Index Pulse Input
7	B- Main Encoder Input
8	A+ Main Encoder Input
9	A- Aux Encoder Input
10	Forward Limit Switch
11	A+ Aux Encoder Input
12	B- Aux Encoder Input
13	Home Switch
14	Reverse Limit Switch
15	+5V

Servo Motor Operation

Brushed Motor Operation

For brushed motor operation, set MT and BR to 1 for the axis.

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 A2.3. 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 A2.3.

```
#tk;                       'begin program
MO;                        'disable all axes
TLA=5;                     'set maximum continuous torque limit of 5 V
TKA=9.9982;                'set maximum peak torque limit of 9.9982 V
SH A;                      'enable a axis
OFA=9.9982;                'command torque signal to 9.9982 V
EN;                        'end program
```

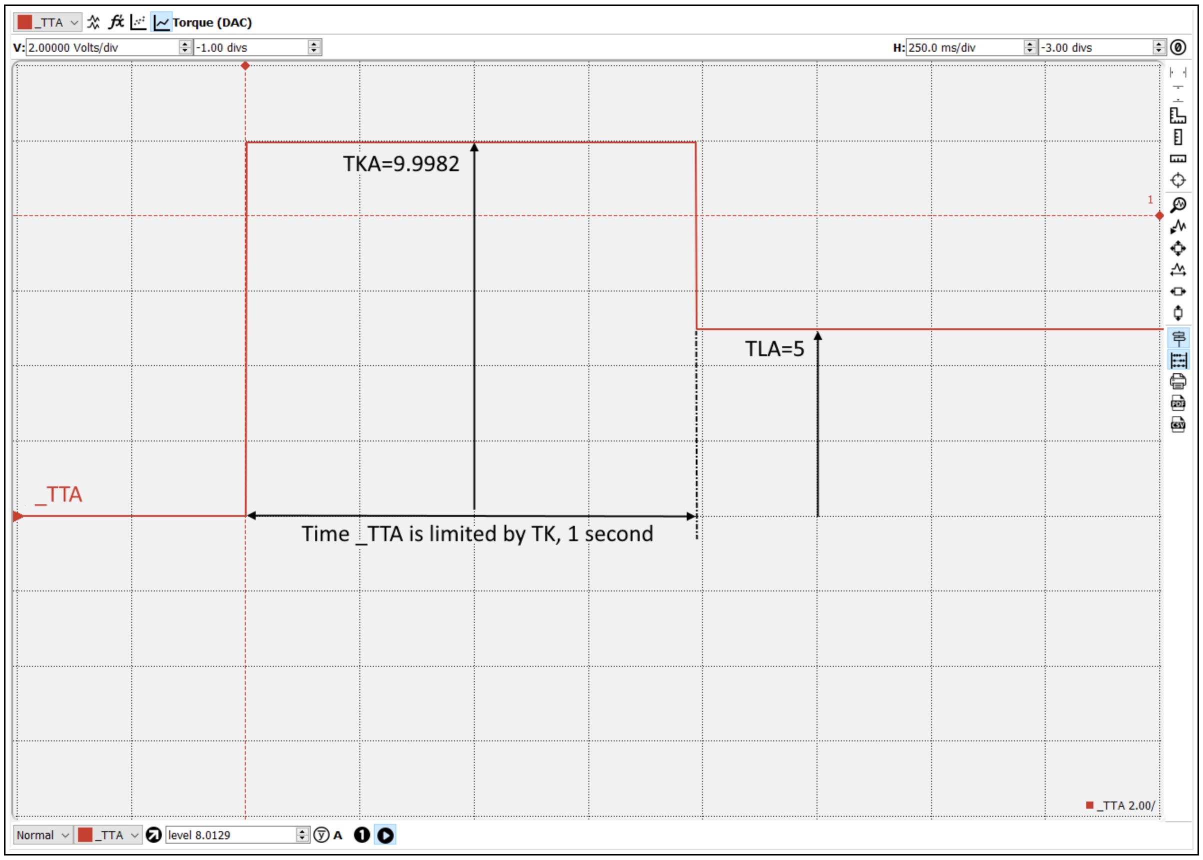


Figure A2.3: Peak Current Operation

Error Monitoring and Protection

The amplifier is protected against Over-Current and Over-Voltage conditions. These conditions are reported as amplifier errors via the TA command.

The user has the option to include the automatic subroutine **#AMPERR** in their program to handle amplifier errors. As long as the **#AMPERR** label is included in the program that is on the controller, the program will jump to the label when an amplifier error is detected and begin executing the user defined routine.

TA **n** is used to monitor amplifier errors. The command will return an eight bit bitmask representing specific error conditions. It will also report whether an amplifier error is latched for the bank of axes A - D or the bank of axes E - H. See the TA command in the Command Reference for detailed information on bit status during error conditions.

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. Since the AMP-20440 is a trans-conductance amplifier, the amplifier should not encounter this error during normal operation.

The behavior of the controller while in an Over-Current condition is dependent on the state of JP5. Table A2.4 below details how the controller will handle the condition:

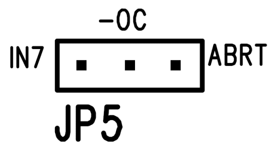
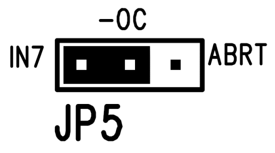
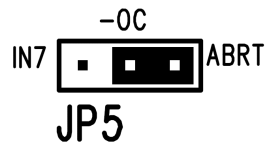
 *No Jumper (Default)*	The controller takes no action when an Over-Current condition is present.
 *Jumper between OC and IN7*	Over-Current error is transmitted to the controller's Digital Input 7. With the CN**,,,,1** and the AE command, TA will report the error and **#AMPERR** will automatically run.
 *Jumper between OC and ABRT*	Over-Current error is transmitted to the controller's Abort Input. Behavior can be modified with the OE command.

Table A2.4: Controller behavior during Over-Current

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.

Abort/ELO Input

If the ELO jumper is installed, the power stage of the amplifier will be disabled when the Abort input is asserted. The Abort input is a latching input with the ELO jumper installed, see the Clearing Latched Amplifier Errors section for details.

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, issue the MO command followed by SH. If an axis is moving, motion must be halted before attempting to clear amplifier errors.

A3 – AMP-20545/20525

Description

The AMP-205x5 is used with the DMC-21x5 while the AMP-205x0 is used with the DMC-21x3. The AMP-20545 is a trapezoidally commutated, 16 bit PWM amplifier. It can also be ordered in a two axis configuration specified as AMP-20525. The AMP-20545 can be configured for the motors below with just a few commands.

- 3-phase Brushless Motors
- 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.

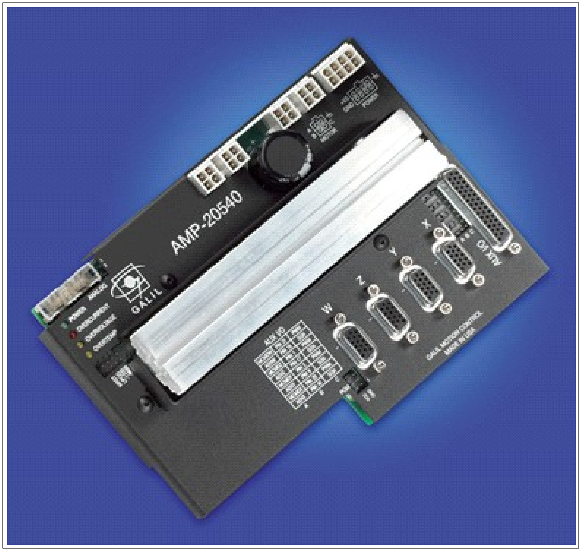


Figure A3.1: AMP-20545

Electrical Specifications

	Servo (Brushless/Brushed)
Supply Voltage	18-60 V
Continuous Current	7 Amps
Peak Current	10 Amps
Switching Frequency	60 kHz
Amplifier Gains	0.4, 0.7, 1.0 Amps/Volt
Minimum Load Inductance	0.5 mH

Table A3.1: Amplifier Electrical Specifications

Mating Connectors

	On Board Connector	Mating Connector	Terminal Pins
POWER	AMP 1-794065-0	AMP 770579-1	AMP 170361-1
A,B,C,D: Motor Power Connectors	AMP 1-770174-0	AMP 172167-1	AMP 170361-1

Table A3.2: Connector and terminal pin Molex part numbers

Power Connector		
Pin Number	**Connection**	
1	Earth Ground	
2,3,4	+VS (DC Power)	
5,6,7,8	DC Power Supply Ground	
Motor Connector		
Pin Number	**3-phase brushless**	**Brushed**
1	No Connect	No Connect
2	Phase A	Phase A+
3	Phase C	No Connect
4	Phase B	Phase A-

Table A3.3: Power and Motor Connector Pin-outs

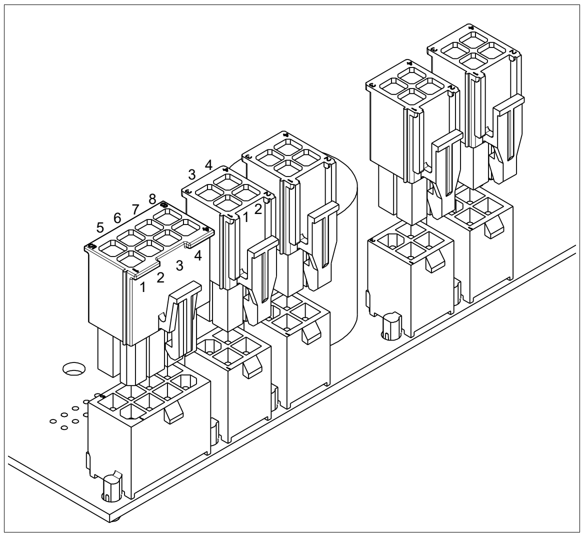


Figure A3.2: Power and Motor Connector Pin-outs

Pin-outs

J3 – Aux I/O 44 pin HD D-Sub Connector (Female)

Pin #	Description	Pin #	Description
1	Motor Command / Step C Axis	23	Digital Input 4 / Latch D Axis
2	Digital Output 6	24	Digital Input 1 / Latch A Axis
3	Digital Output 8	25	Motor Command/Step A Axis
4	Digital Output 5	26	Home Switch A Axis
5	Digital Output 2	27	Home Switch B Axis
6	Abort Input	28	Home Switch C Axis
7	Digital Input 6	29	Home Switch D Axis
8	Digital Input 3 / Latch C Axis	30	Error Output
9	Amplifier Enable / Direction B Axis	31	Motor Command / Step D Axis
10	Output Compare	32	+5V
11	Reverse Limit Switch A Axis	33	+5V
12	Reverse Limit Switch B Axis	34	Digital Ground
13	Reverse Limit Switch C Axis	35	Digital Ground
14	Reverse Limit Switch D Axis	36	Digital Input 8
15	Forward Limit Switch D Axis	37	Digital Input 5
16	Amplifier Enable / Direction D Axis	38	Digital Input 2 / Latch B Axis
17	Amplifier Enable / Direction C Axis	39	Motor Command / Step B Axis
18	Digital Output 7	40	Amplifier Enable / Direction A Axis
19	Digital Output 4	41	Forward Limit Switch A Axis
20	Digital Output 1	42	Forward Limit Switch B Axis
21	Digital Output 3	43	Forward Limit Switch C Axis
22	Digital Input 7	44	Reset Input

m – Encoder 15 pin HD D-Sub Connector (Female)

Pin #	Description
1	I+ Index Pulse Input
2	B+ Main Encoder Input
3	A+ Main Encoder Input
4	B+ Aux Encoder Input
5	Digital Ground
6	I- Index Pulse Input
7	B- Main Encoder Input
8	A+ Main Encoder Input
9	A- Aux Encoder Input
10	Hall A
11	A+ Aux Encoder Input
12	B- Aux Encoder Input
13	Hall B
14	Hall C
15	+5V

J11 – Analog Input 16 pin IDC Header

Pin #	Description	Pin #	Description
1	Digital Ground	2	Digital Ground
3	Analog Input 1	4	Analog Input 2
5	Analog Input 3	6	Analog Input 4
7	Analog Input 5	8	Analog Input 6
9	Analog Input 7	10	Analog Input 8
11	Digital Ground	12	Digital Ground
13	-12V	14	+12V
15	+5V	16	Digital Ground

Servo Motor Operation

3-Phase Brushless Motor Operation

For 3-phase brushless motor operation, MT must be set to 1 for the axis.

The AMP-20545 is a trapezoidally commutating amplifier. Hall sensors are required to be connected to the proper inputs in sequence with the correct motor phases for motion to be produced. It is important to verify the hall sensors and motor phases are in the right combination as the first step to setting up the amplifier.

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-20545 has 3 amplifier gain settings. The gain is set with the AG command as shown in Table A3.4 for AGm=**n**, where m is a specific axis. The axis must be in a motor off (MO) state prior to execution of the AG command.

AG setting	Gain Value
n = 0	0.4 A/V
n = 1	0.7 A/V
n = 2	1.0 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-20545. This command sets the gain for the current loop on the amplifier. Table A3.5 indicates the recommended AUm=**n** settings for 24 and 48 VDC power supplies based on motor inductance. The axis must be in a motor off (MO) state prior to execution of the AU command.

Vsupply VDC	Inductance L (mH)	n	Description
24	$0.5 < L \leq 5$	0	Inverter mode, Normal current loop gain
24	$0.2 \leq L < 0.5$	0.5	Chopper mode, Normal current loop gain
24	$5 < L$	1	Inverter mode, Higher current loop gain
24	$5 < L$	1.5	Chopper mode, Higher current loop gain
48	$0.5 < L \leq 10$	0	Inverter mode, Normal current loop gain
48	$0.2 \leq L < 0.5$	0.5	Chopper mode, Normal current loop gain
48	$10 < L$	1	Inverter mode, Higher current loop gain
48	$10 < L$	1.5	Chopper mode, Higher current loop gain

Table A3.5: 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 A3.3. 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 A3.3.

```
#tk;                        'begin program
MO;                         'disable all axes
TLA=5;                      'set maximum continuous torque limit of 5 V
TKA=9.9982;                 'set maximum peak torque limit of 9.9982 V
SH A;                       'enable a axis
OFA=9.9982;                 'command torque signal to 9.9982 V
EN;                         'end program
```

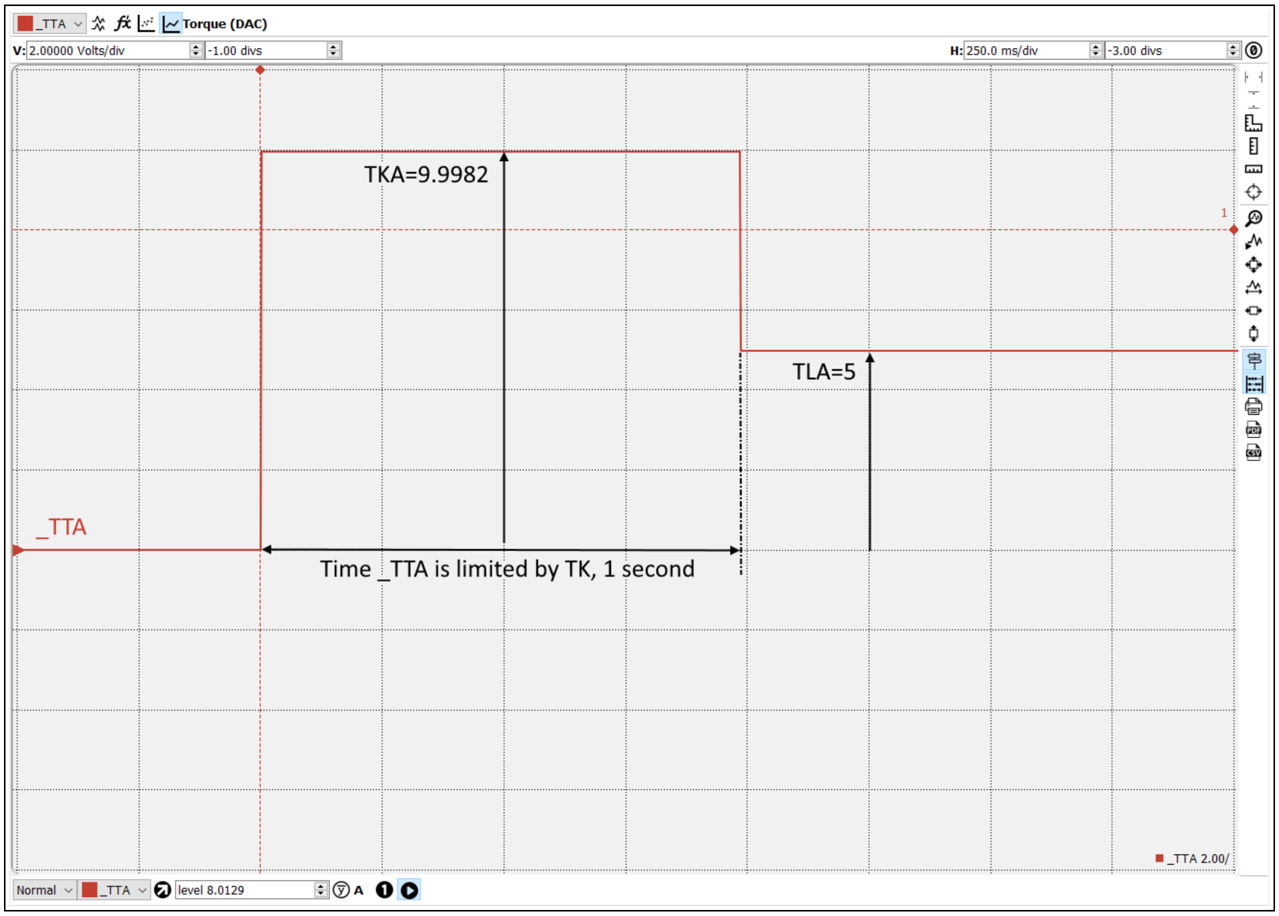


Figure A3.3: Peak Current Operation

Error Monitoring and Protection

The amplifier is protected against Over-Current, Over-Voltage, Over-Temperature, and Under-Voltage conditions. These conditions are reported as amplifier errors via the TA command. See the Clearing Latched Amplifier Errors section for details.

The user has the option to include the automatic subroutine **#AMPERR** in their program to handle amplifier errors. As long as the **#AMPERR** label is included in the program that is on the controller, the program will jump to the label when an amplifier error is detected and begin executing the user defined routine.

TA n is used to monitor amplifier errors. The command will return an eight bit bitmask representing specific error conditions. It will also report whether an amplifier error is latched for the bank of axes A - D or the bank of axes E - H. See the TA command in the Command Reference for detailed information on bit status during error conditions.

NOTE: Amplifier errors are not available when use external amplifiers.

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. Since the AMP-20545 is a trans-conductance amplifier, the amplifier should not encounter this error during normal operation. 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 110°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.

NOTE: 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 AZ2 is issued to the controller. See the Clearing Latched Amplifier Errors section for details.

Abort/ELO Input

If the Abort Input is asserted while the ELO jumper is installed, the power stage of the amplifier will be disabled and the ELO amplifier error will be reported. This is a latching error, see the Clearing Latched Amplifier Errors section for details.

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.

External Amplifier Interface

The AMP-20545 breaks out the step/direction or amplifier enable/motor command signals to control an external servo or stepper amplifier. For example, a machine might have two axes that use the AMP-20545 and two stepper axes that use external drivers. The step and direction signals are accessed through the high density 44-pin D-sub connector. The same connector pins are used to bring out the amplifier enable and motor command line. Which signals are brought out is set via jumpers, see Figure A3.4. If no jumpers are installed (factory default), the corresponding pins on the 44-pin connector will be no-connects. In this example, the C axis will output the motor command on pin 31 and amplifier enable on pin 16. The D axis will output step and direction to the appropriate pins.

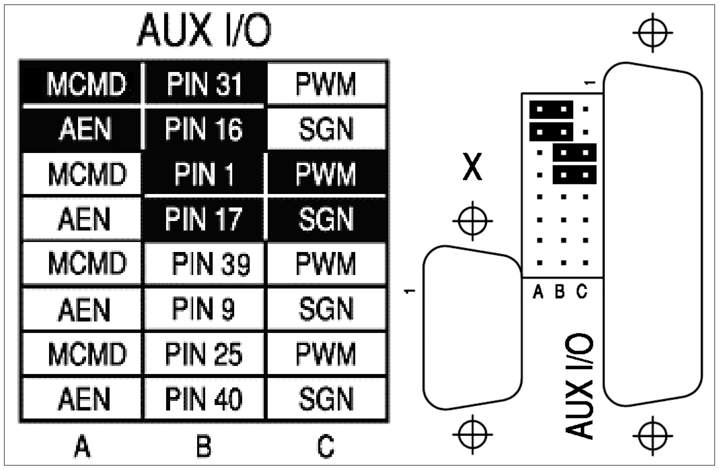


Figure A3.4: D axis is configured for external servo amplifier while the C axis is configured fro external stepper

Analog Inputs

The AMP-20545 has eight analog inputs configured for the range between -10V and 10V. The inputs are decoded by a 12-bit ADC giving a voltage resolution of approximately 0.005V. The value reported by the analog inputs can be queried with the `@AN[n]` operand where n is the input number.

Electrical Specifications

Input Impedance (12 and 16 bit)

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

A4 – SDM-20242

Description

The SDM-20242 is a stepper driver module capable of driving up to four bipolar 2-phase 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

	Stepper
Supply Voltage	12-30 V
Step Resolution, set by JP1	Full, Half, Quarter, 1/16
Amplifier Gains, set by JPn1	0.5, 0.75, 1.0, 1.4 Amps/Phase

Table A4.1: Amplifier Electrical Specifications

Mating Connectors

	On Board Connector	**Mating Connector**	**Terminal Pins**
POWER	AMP 640445-4	AMP 770849-4	AMP 3-770476-1
A,B,C,D: 4-pin Motor Power Connectors	Molex 22-23-2041	Molex 22-01-3047	Molex 08-50-0114

Table A4.2: Connector and terminal pin Molex part numbers

Power Connector	
Pin Number	**Connection**
1,3	+VS (DC Power)
2,4	DC Power Supply Ground
Motor Connector	
Pin Number	**Stepper**
1	Phase A+
2	Phase A-
3	Phase B+
4	Phase B-

Table A4.3: Power and Motor Connector Pin-outs

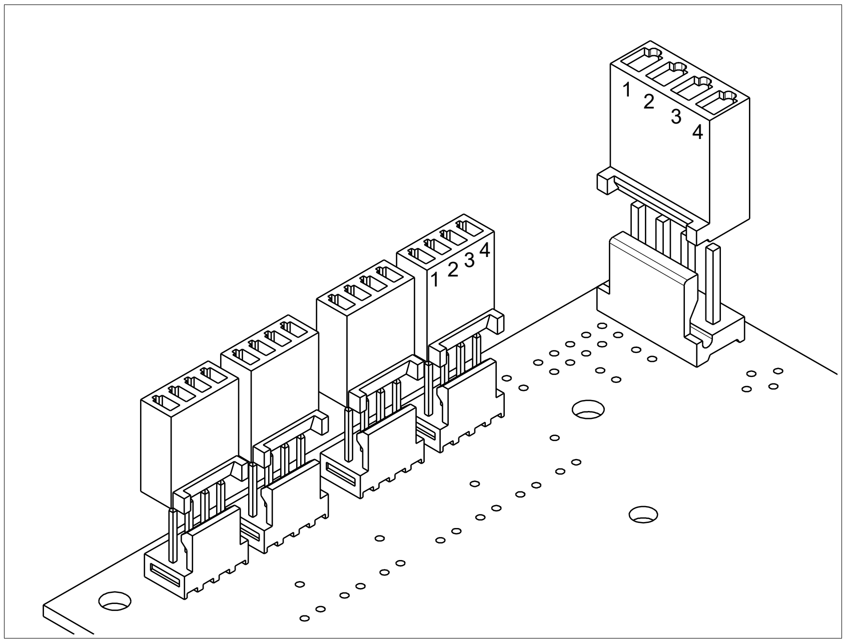


Figure A4.1: Power and Motor Connector Pin-outs

Pin-outs

Jm2 – Encoder 9 pin D-Sub Connector (Male)

Pin #	Description
1	Forward Limit Switch
2	Home Switch
3	+5V
4	A- Main Encoder Input
5	B- Main Encoder Input
6	Reverse Limit Switch
7	Digital Ground
8	A+ Main Encoder Input
9	B+ Main Encoder Input

J11 – I/O 25 pin D-Sub Connector (Female)

Pin #	Description	Pin #	Description
1	Digital Ground	14	+5V
2	Digital Input 1 / Latch A Axis	15	Digital Input 2 / Latch B Axis
3	Digital Input 3 / Latch C Axis	16	Digital Input 4 / Latch D Axis
4	Digital Input 5	17	Digital Input 6
5	Digital Input 7	18	Digital Input 8
6	Abort Input	19	Output Compare
7	Digital Output 1	20	Digital Output 2
8	Digital Output 3	21	Digital Output 4
9	Digital Output 5	22	Digital Output 6
10	Digital Output 7	23	Digital Output 8
11	Digital Ground	24	+5V
12	Reset Input	25	Error Output
13	No Connect		

JP8 – Servo Motor Signals for External Drives

Pin #	Description
1	Amplifier Enable A Axis
2	Motor Command A Axis
3	Amplifier Enable B Axis
4	Motor Command B Axis
5	Amplifier Enable C Axis
6	Motor Command C Axis
7	Amplifier Enable D Axis
8	Motor Command D Axis
9	Digital Ground
10	Digital Ground

Stepper Motor Operation

To configure an axis for a bipolar stepper motor, set MT to -2 for that axis.

Setting Amplifier Gain and Step Resolution

The amplifier gain and the step resolution of the SDM-20242 are set via jumpers.

Amplifier Gain Setting:

The SDM-20242 has four amplifier gain (current) settings. The amplifier gain for each axis is set based on its corresponding jumper: JPX1, JPY1, JPZ1, JPW1.

JPm1	Gain Value
0.5 A	0.5 A/Phase
0.75 A	0.75 A/Phase
1.0 A	1.0 A/Phase
1.4 A	1.4 A/Phase

Table A4.4: Amplifier Gain Settings

Step Resolution Setting:

The step resolution of all axes of the SDM-20242 is set via JP1.

Step Resolution	JP1 Setting
Full Step	No Jumper
Half Step	M1 only
Quarter Step	M2 only
1/16 Step	M1 and M2

Table A4.5: Step Resolution Settings

Low Current Mode

By default, 100% of the current 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 lower the current to 0% while holding position.

LC Command	JP1 Setting	Description
n = 0	LC Jumper On or Off	100% holding current
n = 1	LC Jumper On	25% holding current
n = 1	LC Jumper Off	0% holding current

Table A4.6: Low Current Mode Configurations

Error Monitoring and Protection

The amplifier is protected against Over-Current and Under-Voltage conditions. These conditions are reported as amplifier errors via the TA command.

The user has the option to include the automatic subroutine **#AMPERR** in their program to handle amplifier errors. As long as the **#AMPERR** label is included in the program that is on the controller, the program will jump to the label when an amplifier error is detected and begin executing the user defined routine.

TA **n** is used to monitor amplifier errors. The command will return an eight bit bitmask representing specific error conditions. It will also report whether an amplifier error is latched for the bank of axes A - D or the bank of axes E - H. See the TA command in the Command Reference for detailed information on bit status during error conditions.

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 should not encounter this error during normal operation.

The behavior of the controller while in an Over-Current condition is dependent on the state of JP5. Table A4.7 below details how the controller will handle the condition:

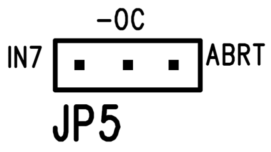
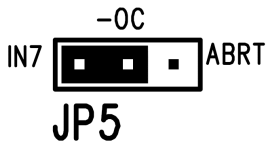
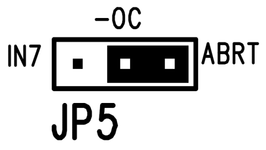
 *No Jumper (Default)*	The controller takes no action when an Over-Current condition is present.
 *Jumper between OC and IN7*	Over-Current error is transmitted to the controller's Digital Input 7. With the CN**,,,,1** and the AE command, TA will report the error and **#AMPERR** will automatically run.
 *Jumper between OC and ABRT*	Over-Current error is transmitted to the controller's Abort input. Behavior can be modified with the OE command.

Table A4.7: Controller behavior during Over-Current

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.

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.

Abort/ELO Input

If the ELO jumper is installed, the power stage of the amplifier will be disabled when the Abort input is asserted. The Abort input is a latching input with the ELO jumper installed, see the Clearing Latched Amplifier Errors section for details.

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, issue the `MO` command followed by `SH`. If an axis is moving, motion must be halted before attempting to clear amplifier errors.

A5 – SDM-20645

Description

The SDM-20645 is used with the DMC-21x5 while the SDM-20640 is used with the DMC-21x3. The SDM-20465 is a microstepping driver module capable of driving up to four bipolar 2-phase 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.

Figure A5.1: SDM-20645

Electrical Specifications

	Stepper
Supply Voltage	12-60 V
Step Resolution	1/64
Amplifier Gains	0.5, 1.0, 2.0, 3.0 Amps/Phase

Table A5.1: Amplifier Electrical Specifications

Mating Connectors

	On Board Connector	Mating Connector	Terminal Pins
POWER	AMP 1-794065-0	AMP 770579-1	AMP 170361-1
A,B,C,D: 4-pin Motor Power Connectors	AMP 1-770174-0	AMP 172167-1	AMP 170361-1

Table A5.2: Connector and terminal pin Molex part numbers

Power Connector	
Pin Number	**Connection**
1	Earth Ground
2, 3, 4	+VS (DC Power)
5, 6, 7, 8	DC Power Supply Ground
Motor Connector	
Pin Number	**Stepper**
1	Phase B+
2	Phase A+
3	Phase B-
4	Phase A-

Table A5.3: Power and Motor Connector Pin-outs

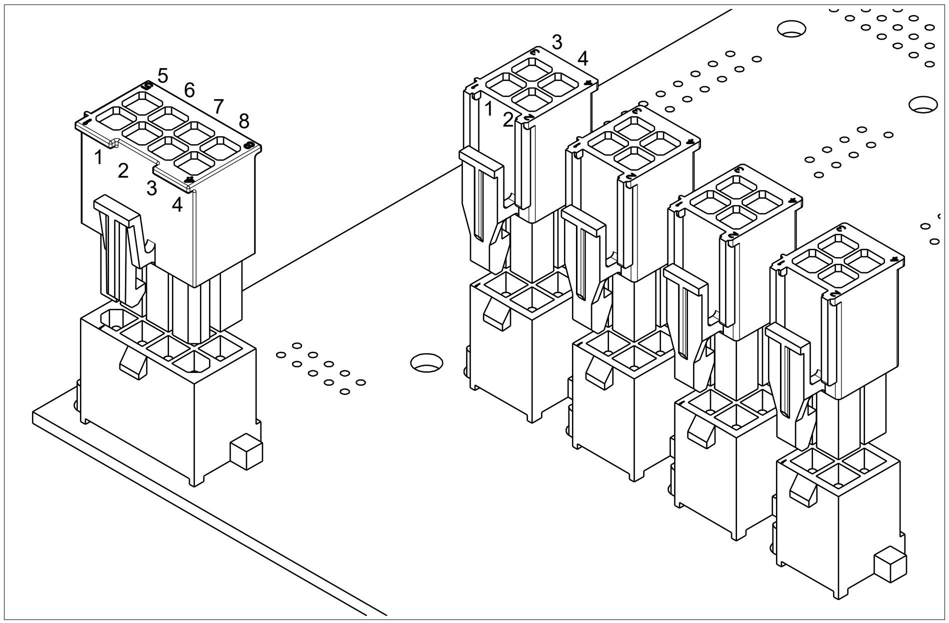


Figure A5.2: Power and Motor Connector Pin-outs

Pin-outs

Jn2 – Encoder 9 pin D-Sub Connector (Male)

Pin #	Description
1	Forward Limit Switch
2	Home Switch
3	+5V
4	A- Main Encoder Input
5	B- Main Encoder Input
6	Reverse Limit Switch
7	Digital Ground
8	A+ Main Encoder Input
9	B+ Main Encoder Input

J3 – I/O 25 pin D-Sub Connector (Female)

Pin #	Description	Pin #	Description
1	Digital Ground	14	+5V
2	Digital Input 1 / Latch A Axis	15	Digital Input 2 / Latch B Axis
3	Digital Input 3 / Latch C Axis	16	Digital Input 4 / Latch D Axis
4	Digital Input 5	17	Digital Input 6
5	Digital Input 7	18	Digital Input 8
6	Abort Input	19	Output Compare
7	Digital Output 1	20	Digital Output 2
8	Digital Output 3	21	Digital Output 4
9	Digital Output 5	22	Digital Output 6
10	Digital Output 7	23	Digital Output 8
11	Digital Ground	24	+5V
12	Reset Input	25	Error Output
13	No Connect		

J11 – Analog Input 16 pin IDC Header

Pin #	Description	Pin #	Description
1	Digital Ground	2	Digital Ground
3	Analog Input 1	4	Analog Input 2
5	Analog Input 3	6	Analog Input 4
7	Analog Input 5	8	Analog Input 6
9	Analog Input 7	10	Analog Input 8
11	Digital Ground	12	Digital Ground
13	-12V	14	+12V
15	+5V	16	Digital Ground

JP8 – Servo Motor Signals for External Drives

Pin #	Description
1	Amplifier Enable A Axis
2	Motor Command A Axis
3	Amplifier Enable B Axis
4	Motor Command B Axis
5	Amplifier Enable C Axis
6	Motor Command C Axis
7	Amplifier Enable D Axis
8	Motor Command D Axis
9	Digital Ground
10	Digital Ground

Stepper Motor Operation

To configure an axis for a bipolar stepper motor, set MT to -2 for that axis.

Setting Amplifier Gain and Step Resolution

The amplifier gain and the step resolution of the SDM-20645 are set via jumpers.

Amplifier Gain Setting:

The SDM-20645 has four amplifier gain (current) settings. The axis must be in a motor off (MO) state prior to execution of the AG command. The gain is set with the AG command as shown in for AGm=**n**:

AG Setting	Gain Value
n = 0	0.5 A/Phase
n = 1	1.0 A/Phase
n = 2	2.0 A/Phase
n = 3	3.0 A/Phase

Table A5.4: Amplifier Gain Settings

Low Current Mode

By default, 100% of the current 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.

LC Setting	Description
n = 0	100% holding current
n = 1	25% holding current

Table A5.5: Low Current Mode Configurations

Error Monitoring and Protection

The amplifier is protected against Over-Current and Under-Voltage conditions. These conditions are reported as amplifier errors via the TA command.

The user has the option to include the automatic subroutine **#AMPERR** in their program to handle amplifier errors. As long as the **#AMPERR** label is included in the program that is on the controller, the program will jump to the label when an amplifier error is detected and begin executing the user defined routine.

TA **n** is used to monitor amplifier errors. The command will return an eight bit bitmask representing specific error conditions. It will also report whether an amplifier error is latched for the bank of axes A - D or the bank of axes E - H. See the TA command in the Command Reference for detailed information on bit status during error conditions.

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 should not encounter this error during normal operation. 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.

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.

Abort/ELO Input

If the Abort Input is asserted while the ELO jumper is installed, the power stage of the amplifier will be disabled and the ELO amplifier error will be reported. This is a latching error, see the Clearing Latched Amplifier Errors section for details.

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.

Analog Inputs

The SDM-20645 has eight analog inputs configured for the range between -10V and 10V. The inputs are decoded by a 12-bit ADC giving a voltage resolution of approximately 0.005V. The value reported by the analog inputs can be queried with the `@AN[n]` operand where n is the input number.

Electrical Specifications

Input Impedance (12 and 16 bit)

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

A6 – ICM-20100

Description

The ICM-20100 interconnect module provides D-Sub connections between the DMC-21x5 controller and other system elements such as amplifiers, encoders, and external switches for up to 4 axes.

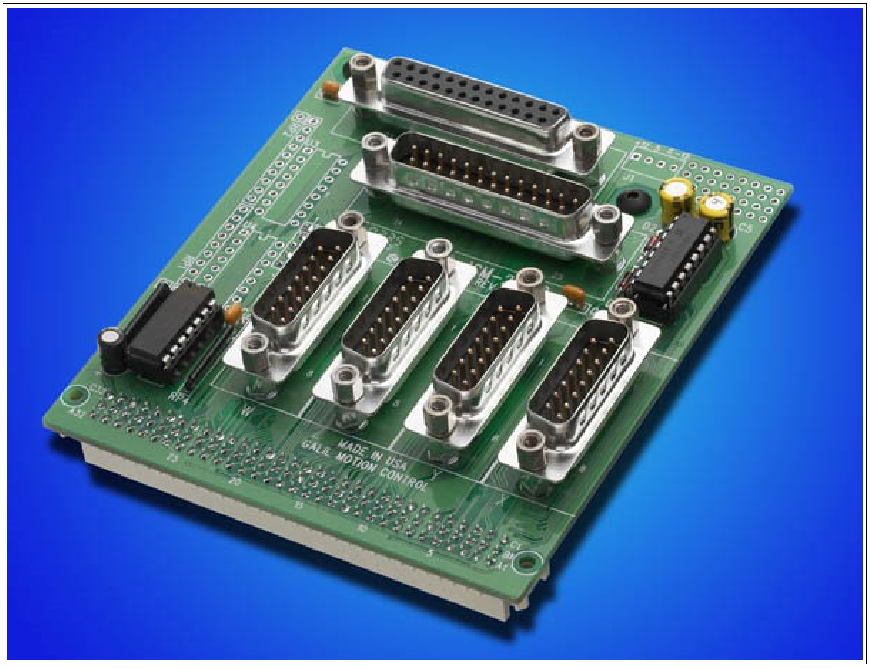


Figure A6.1: ICM-20100

Pin-outs

Jn – Encoder 15 pin D-Sub Connector (Male)

Pin #	Description
1	Forward Limit Switch
2	Home Switch
3	+5V
4	A- Main Encoder Input
5	B- Main Encoder Input
6	I- Index Pulse Input
7	Amplifier Enable
8	Direction
9	Reverse Limit Switch
10	Digital Ground
11	A+ Main Encoder Input
12	B+ Main Encoder Input
13	I+ Index Pulse Input
14	Motor Command
15	Step

J10 – Aux Encoders 25 pin D-Sub Connector (Female)

Pin #	Description	Pin #	Description
1	Reset Input	14	Error Output
2	B- Aux Encoder Input D Axis	15	B+ Aux Encoder Input D Axis
3	A- Aux Encoder Input D Axis	16	A+ Aux Encoder Input D Axis
4	B- Aux Encoder Input C Axis	17	B+ Aux Encoder Input C Axis
5	A- Aux Encoder Input C Axis	18	A+ Aux Encoder Input C Axis
6	B- Aux Encoder Input B Axis	19	B+ Aux Encoder Input B Axis
7	A- Aux Encoder Input B Axis	20	A+ Aux Encoder Input B Axis
8	B- Aux Encoder Input A Axis	21	B+ Aux Encoder Input A Axis
9	A- Aux Encoder Input A Axis	22	A+ Aux Encoder Input A Axis
10	+5V	23	Digital Ground
11	+5V	24	Digital Ground
12	+12V	25	-12V
13	No Connect		

J11 – I/O 25 pin D-Sub Connector (Male)

Pin #	Description	Pin #	Description
1	Digital Ground	14	+5V
2	Digital Input 1 / Latch A Axis	15	Digital Input 2 / Latch B Axis
3	Digital Input 3 / Latch C Axis	16	Digital Input 4 / Latch C Axis
4	Digital Input 5	17	Digital Input 6
5	Digital Input 7	18	Digital Input 8
6	Abort Input	19	Output Compare
7	Digital Output 1	20	Digital Output 2
8	Digital Output 3	21	Digital Output 4
9	Digital Output 5	22	Digital Output 6
10	Digital Output 7	23	Digital Output 8
11	Digital Ground	24	+5V
12	No Connect	25	No Connect
13	No Connect		

A7 – ICM-20105

Description

The ICM-20105 interconnect module provides D-Sub connections between the DMC-21x5 controller and other system elements such as amplifiers, encoders, and external switches for up to 4 axes.

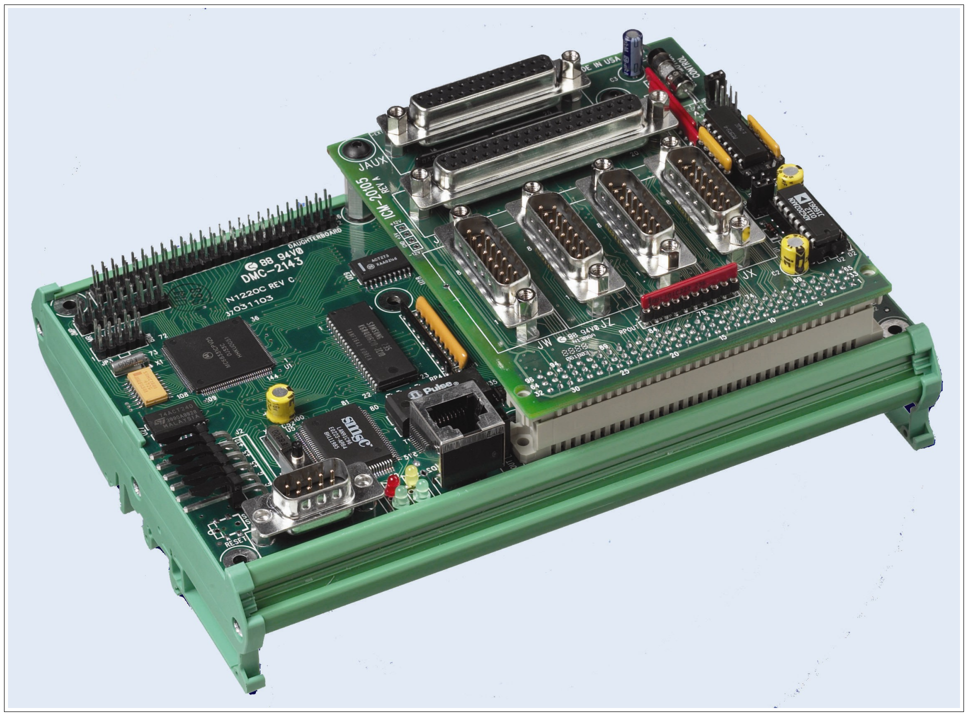


Figure A7.1: ICM-20105

Pin-outs

Jn – Encoder 15 pin D-Sub Connector (Male)

Pin #	Description
1	Amplifier Enable Common 1
2	Amplifier Enable
3	+5V
4	A- Main Encoder Input
5	B- Main Encoder Input
6	I- Index Pulse Input
7	No Connect
8	Direction
9	Amplifier Enable Common 2
10	Digital Ground
11	A+ Main Encoder Input
12	B+ Main Encoder Input
13	I+ Index Pulse Input
14	Motor Command
15	Step

JAUX – Aux Encoders 25 pin D-Sub Connector (Female)

Pin #	Description	Pin #	Description
1	No Connect	14	No Connect
2	B- Aux Encoder Input D Axis	15	B+ Aux Encoder Input D Axis
3	A- Aux Encoder Input D Axis	16	A+ Aux Encoder Input D Axis
4	B- Aux Encoder Input C Axis	17	B+ Aux Encoder Input C Axis
5	A- Aux Encoder Input C Axis	18	A+ Aux Encoder Input C Axis
6	B- Aux Encoder Input B Axis	19	B+ Aux Encoder Input B Axis
7	A- Aux Encoder Input B Axis	20	A+ Aux Encoder Input B Axis
8	B- Aux Encoder Input A Axis	21	B+ Aux Encoder Input A Axis
9	A- Aux Encoder Input A Axis	22	A+ Aux Encoder Input A Axis
10	+5V	23	Digital Ground
11	+5V	24	Digital Ground
12	+12V	25	-12V
13	No Connect		

JIO – I/O 37 pin D-Sub Connector (Male)

Pin #	Description	Pin #	Description
1	Input Common	20	Digital Input 1
2	Digital Input 2	21	Digital Input 3
3	Digital Input 4	22	Digital Input 5
4	Digital Input 6	23	Digital Input 7
5	Digital Input 8	24	Abort Input
6	Output Power	25	Digital Output 1
7	Digital Output 2	26	Digital Output 3
8	Digital Output 4	27	Digital Output 5
9	Digital Output 6	28	Digital Output 7
10	Digital Output 8	29	Output Return
11	Limit Switch Common	30	Forward Limit Switch A Axis
12	Reverse Limit Switch A Axis	31	Home Switch A Axis
13	Forward Limit Switch B Axis	32	Reverse Limit Switch B Axis
14	Home Switch B Axis	33	Forward Limit Switch C Axis
15	Reverse Limit Switch C Axis	34	Home Switch C Axis
16	Forward Limit Switch D Axis	35	Reverse Limit Switch D Axis
17	Home Switch D Axis	36	+5V
18	+5V	37	Digital Ground
19	Digital Ground		

Optoisolated Inputs

The ICM-20105 allows for optoisolation on all of the digital inputs of the DMC-21x5. Input Common (INCOM) located on the I/O 37 pin D-Sub connector of the ICM-20105 provides power to digital inputs (DI[8:1]), the Abort input (ABRT), and Reset input (RST). Limit Switch Common (LSCOM0) located on the I/O 37 pin D-Sub connector of the ICM-20105 provides power to the Forward Limit Switch inputs (FLSm), Reverse Limit Switch inputs (RLSm), and Home Switch Inputs (HOMm). The optoisolated inputs are powered in banks. Table A7.2 shows all the input banks power commons and their corresponding inputs for 1-4 axis controllers and Table A7.3 shows the input banks for 5-8 axis controllers.

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 136 in the Appendices for electrical specifications. Using the controller reference power completely bypasses optoisolation and is not recommended for most applications.

Electrical Specifications

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

Table A7.1: Specifications for voltage and current limit

Common Signal	Common Signal Location	Powers Inputs Labeled
INCOM (Bank 0)	J10 on ICM-20105, pin 1	DI[8:1], ABRT, RST
LSCOM (Bank 0)	J10 on ICM-20105, pin 11	FLSA, RLSA, HOMA FLSB, RLSB, HOMB FLSC, RLSC, HOMC FLSD, RLSD, HOMD

Table A7.2: 1-4 axis controller Input Common (INCOM) and Limit Switch Common (LSCOM) and corresponding inputs powered

Common Signal	Common Signal Location	Powers Inputs
INCOM (Bank 0)	J10 on first ICM-20105, pin 1	DI[8:1], ABRT, RST
LSCOM (Bank 0)	J10 on first ICM-20105, pin 11	FLSA, RLSA, HOMA FLSB, RLSB, HOMB FLSC, RLSC, HOMC FLSD, RLSD, HOMD
INCOM (Bank 1)	J10 on second ICM-20105, pin 1	DI[16:9]
LSCOM (Bank 1)	J10 on second ICM-20105, pin 11	FLSE, RLSE, HOME FLSF, RLSF, HOMF FLSG, RLSG, HOMG FLSH, RLSH, HOMH

Table A7.3: 5-8 axis controller Input Common (INCOM) and Limit Switch Common (LSCOM) banks and corresponding inputs powered

Wiring the Optoisolated Digital Inputs

Banks of inputs can be used as either active high or low. Connecting $+V_s$ to INCOM/LSCOM will configure the inputs for active low as current will flow through the diode when the inputs are pulled to the isolated ground. Connecting the isolated ground to INCOM/LSCOM will configure the inputs for active high as current will flow through the diode when the inputs are pulled up to $+V_s$.

Figure A7.2, and Figure A7.3 are the optoisolated wiring diagrams for powering INCOM and LSCOM and its respective inputs.

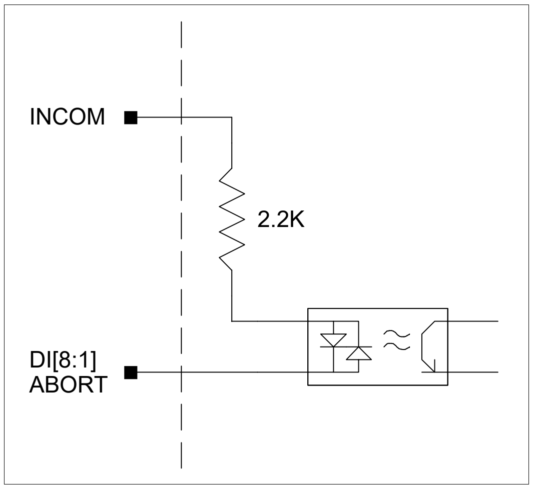


Figure A7.2: Digital Inputs 1-8 (DI[8:1]) and Abort Input

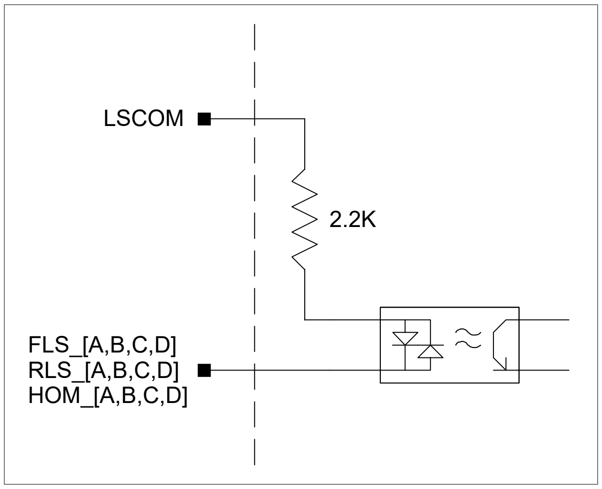


Figure A7.3: Limit Switch Inputs for Axes A-D

Optoisolated Outputs

The ICM-20105 allows for optoisolation for the digital outputs (DO). The ICM outputs are 500mA sourcing outputs.

The amount of uncommitted, optoisolated outputs the DMC-21x5 has depends on the number of axis. 1-4 axis controllers have a single bank of 8 outputs, Bank 0 (DO[8:1]). 5-8 axis controllers have an additional bank of 8 outputs, Bank 1 (DO[16:9]).

Description

The outputs are capable of sourcing up to 500mA per output and up to 3A per bank of outputs. 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 Power Max Voltage	24 V_{DC}
Output Power Min Voltage	12 V_{DC}
Maximum current per input	0.5 A (not to exceed 3A per Bank)

Table A7.4: Specifications for voltage and current limit

Wiring the Optoisolated Digital Outputs

The output power supply will be connected to Output power (Output PWR) and the power supply return will be connected to Output GND (Output RET). Note that the load is wired between DO and Output GND. The wiring diagram is shown in Figure A7.4.

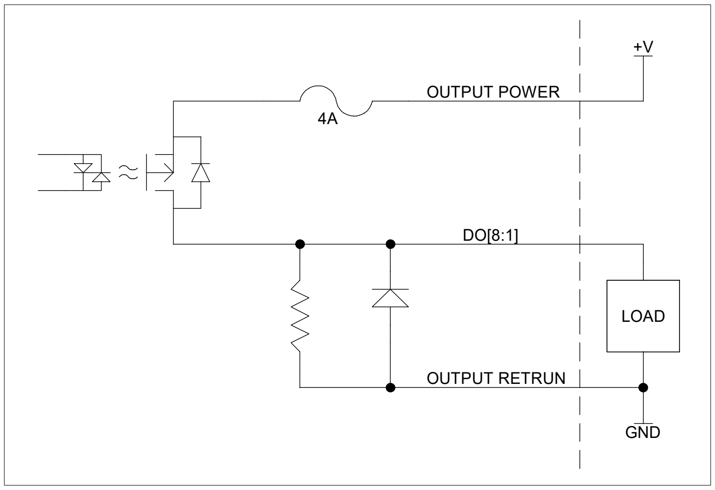


Figure A7.4: 500mA sourcing wiring diagrams for Bank 0, DO[8:1]

Amplifier Enable Circuit

This section describes how to configure the ICM-20105 for different Amplifier Enable configurations.

The ICM-20105 gives the user a broad range of options with regards to the voltage levels present on the enable signal. 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. Table A7.5 and Table A7.6 found below illustrate the settings for jumpers, resistor packs, and the socketed optocoupler IC. Refer to Figure A7.5 and Figure A7.6 for precise physical locations of all components. Note that the resistor pack located at RPAE1 may be reversed to change the active state of the amplifier enable output. However, the polarity of RPAE2 must not be changed; a different resistor value may be needed to limit the current to 6 mA. The default value for RPAE2 is 820 Ω, which works at 5V. When using 24 V, RPAE2 should be replaced with a 4.7 kΩ resistor pack.

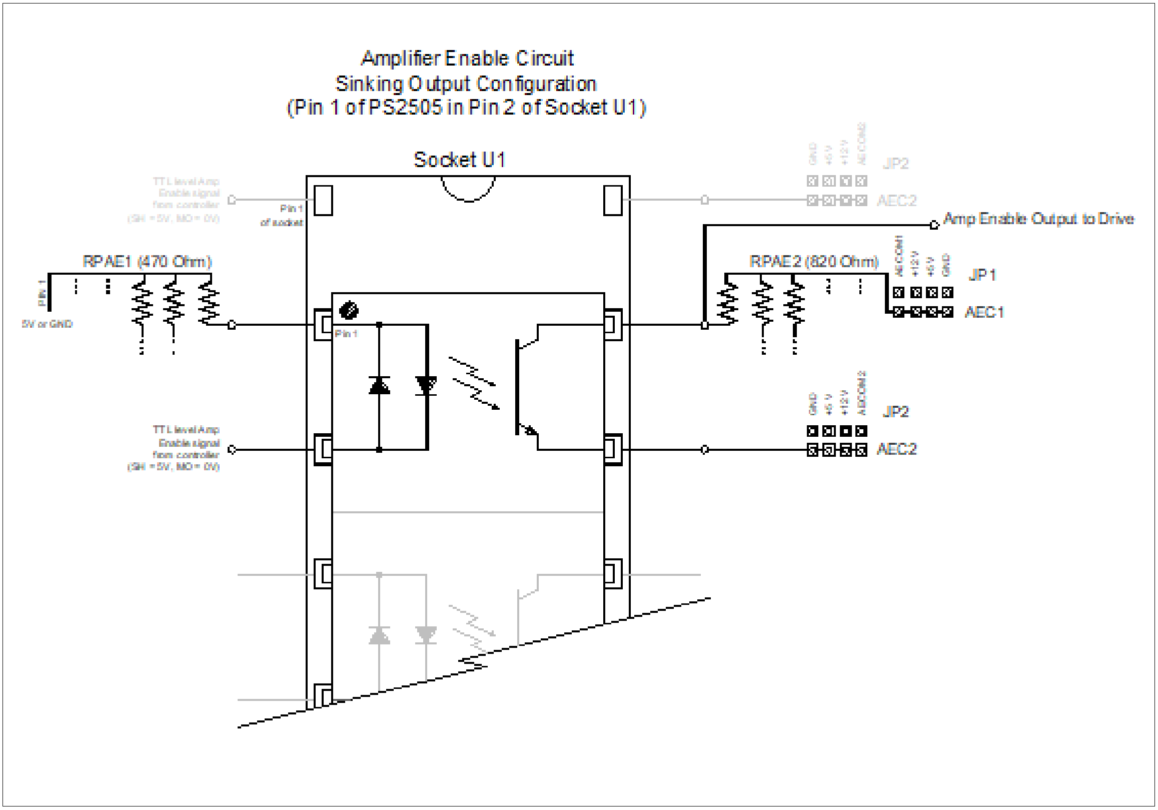

Amplifier Enable Circuit
Sinking Output Configuration
(Pin 1 of PS2505 in Pin 2 of Socket U1)
Socket U1
TTL level Amp Enable signal from controller (SH = 5V, MO = 0V)
Pin 1 of socket
RPAE1 (470 Ohm)
Pin 1
5V or GND
Pin 1
GND +5V +12V AECOM2
JP2
AEC2
Amp Enable Output to Drive
RPAE2 (820 Ohm)
AECOM1 +12V +5V GND
JP1
AEC1
TTL level Amp Enable signal from controller (SH = 5V, MO = 0V)
GND +5V +12V AECOM2
JP2
AEC2

Figure A7.5: Amplifier Enable Circuit Sinking Output Configuration

Sinking Configuration (pin1 of chip in pin2 of socket U1)			
Logic State	**JP1**	**JP2**	**RPAE1** (square pin next to RPAE1 label is 5V)
5V, HAEN (Default configuration)	5V – AEC1	GND – AEC2	Dot on R-pack next to RPAE1 label
5V, LAEN	5V – AEC1	GND – AEC2	Dot on R-pack opposite RPAE1 label
12V, HAEN	+12V – AEC1	GND – AEC2	Dot on R-pack next to RPAE1 label
12V, LAEN	+12V – AEC1	GND – AEC2	Dot on R-pack opposite RPAE1 label
Isolated 24V, HAEN	AECOM1 – AEC1	AECOM2 – AEC2	Dot on R-pack next to RPAE1 label
Isolated 24V, LAEN	AECOM1 – AEC1	AECOM2 – AEC2	Dot on R-pack opposite RPAE1 label
For 24V isolated enable, tie +24V of external power supply to AECOM1 at any axis D-sub connector, tie common return to AECOM2. Replace RPAE2 with a 4.7 kΩ resistor pack. AECOM1 and AECOM2 are located on any axis D-sub connector. All pins labeled AECOM1 are connected. All pins labeled AECOM2 are connected.			

Table A7.5: Sinking Configuration

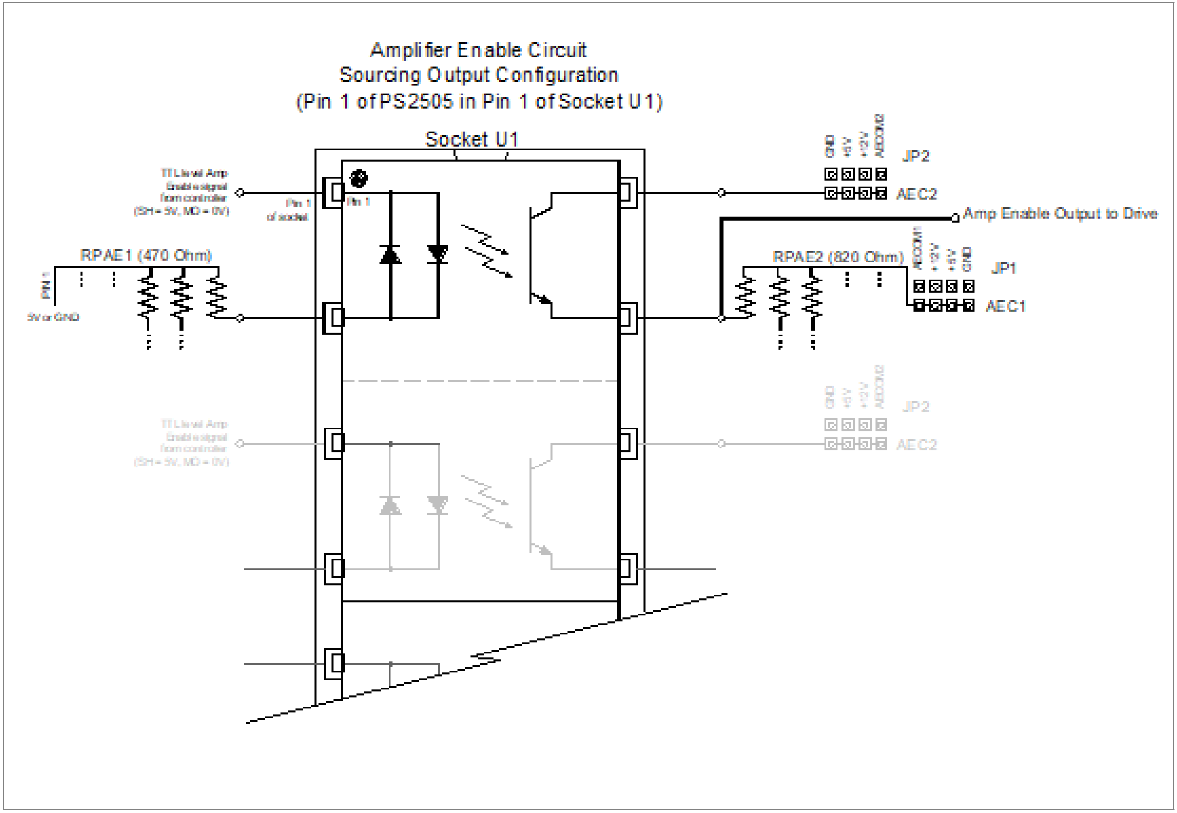

Amplifier Enable Circuit
Sourcing Output Configuration
(Pin 1 of PS2505 in Pin 1 of Socket U1)
Socket U1
TTL level Amp Enable signal from controller (SH = 5V, MO = 0V)
Pin 1 of socket
Pin 1
RPAE1 (470 Ohm)
Pin 1
5V or GND
GND
+5V
+12V
AECOM2
JP2
AEC2
Amp Enable Output to Drive
RPAE2 (820 Ohm)
AECOM1
+12V
+5V
GND
JP1
AEC1

Figure A7.6: Amplifier Enable Circuit Sourcing Output Configuration

Sourcing Configuration (pin1 of chip in pin1 of socket U1)			
Logic State	**JP1**	**JP2**	**RPAE1** (square pin next to RPAE1 label is 5V)
5V, HAEN	GND – AEC1	5V – AEC2	Dot on R-pack opposite RPAE1 label
5V, LAEN	GND – AEC1	5V – AEC2	Dot on R-pack next to RPAE1 label
12V, HAEN	GND – AEC1	+12V – AEC2	Dot on R-pack opposite RPAE1 label
12V, LAEN	GND – AEC1	+12V – AEC2	Dot on R-pack next to RPAE1 label
Isolated 24V, HAEN	AECOM1 – AEC1	AECOM2 – AEC2	Dot on R-pack opposite RPAE1 label
Isolated 24V, LAEN	AECOM1 – AEC1	AECOM2 – AEC2	Dot on R-pack next to RPAE1 label
For 24V isolated enable, tie +24V of external power supply to AECOM2 at any axis D-sub connector, tie common return to AECOM1. Replace RPAE2 with a 4.7 kΩ resistor pack. AECOM1 and AECOM2 are located on any axis D-sub connector. All pins labeled AECOM1 are connected. All pins labeled AECOM2 are connected.			

Table A7.6: Sourcing Configuration

A8 – DB-28045

Description

The DB-28045 is used with the DMC-21x5 while the DB-28040 is used with the DMC-21x3. The DB-28045 daughter board provides access to an additional 40 bits of configurable I/O as well as 8 analog inputs.

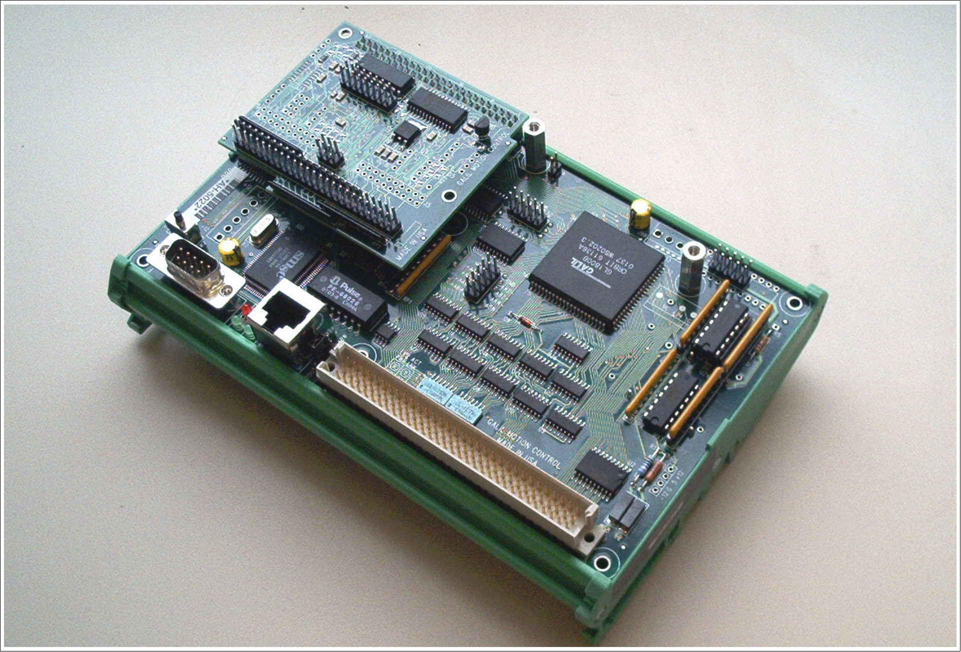


Figure A8.1: DB-28045 mounted on DMC-21x3

Pin-outs

J1 – Digital I/O 50 pin IDC Header

Pin #	Description	Pin #	Description
1	Bit 40	2	Bit 41
3	Bit 39	4	Bit 42
5	Bit 38	6	Bit 43
7	Bit 37	8	Bit 44
9	Bit 36	10	Bit 45
11	Bit 35	12	Bit 46
13	Bit 34	14	Bit 47
15	Bit 33	16	Bit 48
17	Bit 32	18	Bit 49
19	Bit 31	20	Bit 50
21	Bit 30	22	Bit 51
23	Bit 29	24	Bit 52
25	Bit 28	26	Bit 53
27	Bit 27	28	Bit 54
29	Bit 26	30	Bit 55
31	Bit 25	32	Bit 56
33	Bit 24	34	Digital Ground
35	Bit 23	36	Digital Ground
37	Bit 22	38	Digital Ground
39	Bit 21	40	Digital Ground
41	Bit 20	42	Digital Ground
43	Bit 19	44	Digital Ground
45	Bit 18	46	Digital Ground
47	Bit 17	48	Digital Ground
49	+5V	50	Digital Ground

J3 – Analog Inputs 16 pin Header

Pin #	Description	Pin #	Description
1	Ground	2	Ground
3	Analog Input 1	4	Analog Input 2
5	Analog Input 3	6	Analog Input 4
7	Analog Input 5	8	Analog Input 6
9	Analog Input 7	10	Analog Input 8
11	Ground	12	Ground
13	-12V	14	+12V
15	+5V	16	Ground

Extended I/O

The DB-28045 offers 40 extended TTL I/O points which can be configured as inputs or outputs in banks of 8. Banks of I/O are configured with the CO command. The I/O points are accessed through the 50 pin IDC header. The state of the bits that are setup as inputs can be queried with the @IN[n] operand where n is the bit number. The state of the bits that are configured as outputs can be queried with the @OUT[n] operand where n is the bit number.

Electrical Specifications (3.3V – Standard)

Inputs

Max Input Voltage	3.4 VDC
Minimum Voltage	0 VDC

Inputs are internally pulled up to 3.3V through a 4.7kΩ resistor

Outputs

Sink/Source	4mA per output

Electrical Specifications (5V Option)

Inputs

Max Input Voltage	5.25 VDC
Minimum Voltage	0 VDC

Inputs are internally pulled up to 5V through a 4.7kΩ resistor

Outputs

Sink/Source	20mA

Analog Inputs

The DB-28045 has eight analog inputs configured for the range between -10V and 10V. The inputs are decoded by a 12-bit ADC giving a voltage resolution of approximately 0.005V. A 16-bit ADC is available as an option. The value reported by the analog inputs can be queried with the @AN[n] operand where n is the input number.

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

A9 – SR-19900

Description

For applications requiring a shunt regulator, Galil offers a small mountable model that can be configured for varying voltage levels. Two fixed voltage threshold settings are available with jumpers, which can be set at either 33 or 66 volts. Additionally, a user defined voltage threshold can be set by changing a simple resistor. This shunt regulator operates with hysteresis, where the regulator switches on at the set voltage threshold and switches off at 2 volts below.

The shunt regulator should be placed in parallel with the power supply as in the figure below, and it should be mounted to a metal surface using thermal grease to aid in heat transfer. Connections are made to the unit at VS (voltage supply) and PG (power ground) using either the 4-pin connector or the 8-pin connector.

For a summary of shunt regulator operation, as well as details to help determine if one is required , please refer to application note #5448 at: https://www.galil.com/support/appnotes/miscellaneous/note5448.pdf.

Layout

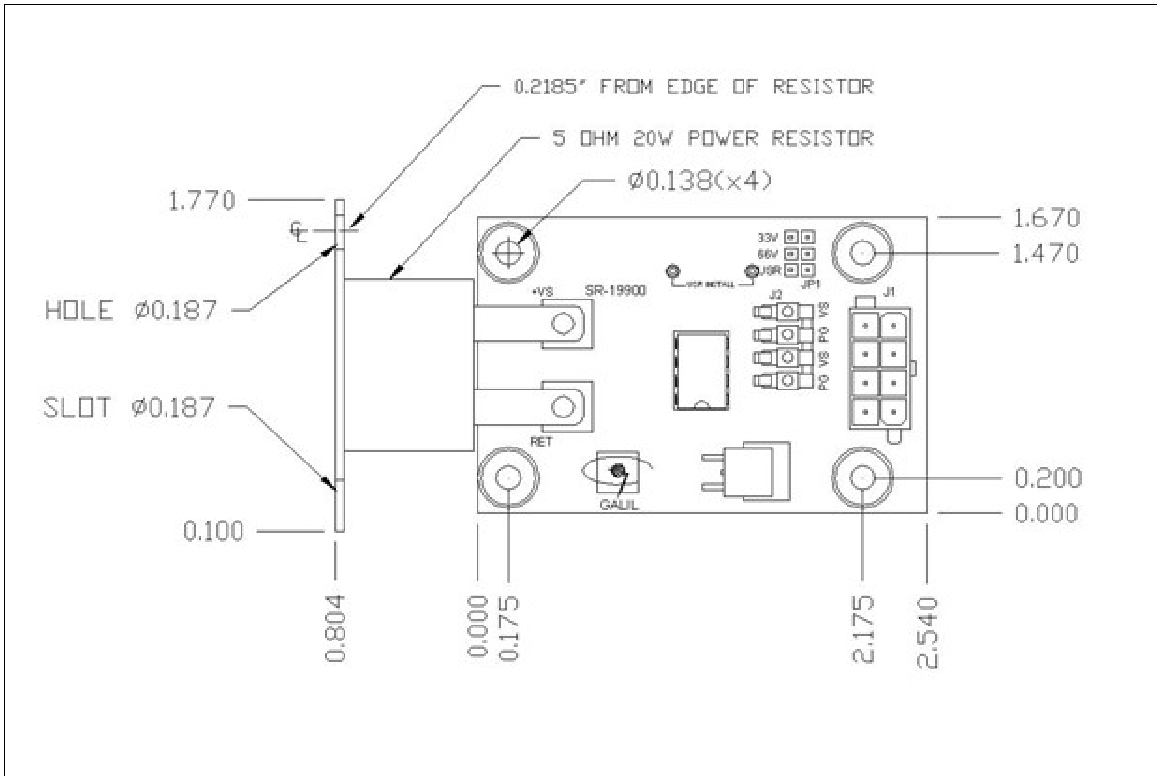


Figure A9.1: SR-19900 Layout

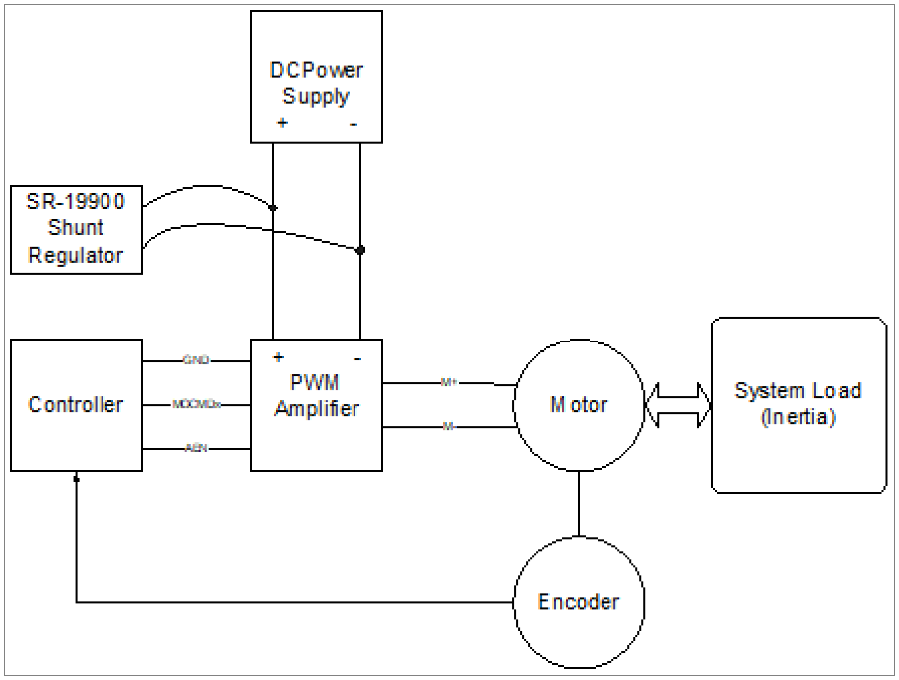


Figure A9.2 Shunt Regulator Placement in a Typical Servo System

Electrical Specifications

JP1	Voltage Threshold Setting (Vs)
33V	33 Volts
66V	66 Volts
USR	User Selectable [1]

*1 - R8 = 1930 * Vs – 42.2k*

Table A9.1: Amplifier Electrical Specifications

Mating Connectors

	On Board Connector	Terminal Pins
J2 4-pin Molex Connector	Molex #26-03-4041	Molex #08-50-0189
J1 8-pin Molex Connector	8-pin Mini Universal MATE-N-LOK AMP# 770579-1	AMP# 170361-1

Table A9.2: Connector and terminal pin Molex part numbers

J2 4-pin Molex Connector	
Pin Number	**Connection**
1,3	DC Power Supply Ground
2,4	+VS (DC Power)
J1 8-pin Molex Connector	
Pin Number	**Connection**
1	Earth Ground
2,3,4	+VS (DC Power)
5,6,7,8	DC Power Supply Ground

Table A9.3: Power and Motor Connector Pin-outs