ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Parker / Compumotor 4000-M* at our website: *Click HERE*

Compumotor

Model 4000 Indexer User Guide

Compumotor Division
Parker Hannifin Corporation
p/n 88-007719-01 J January 10, 1994

Parker

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Important User Information

To ensure that the equipment described in this user guide, as well as all the equipment connected to and used with it, operates satisfactorily and safely, all applicable local and national codes that apply to installing and operating the equipment must be followed. Since codes can vary geographically and can change with time, it is the user's responsibility to identify and comply with the applicable standards and codes. WARNING: *Failure to comply with applicable codes and standards can result in damage to equipment and/or serious injury to personnel.*

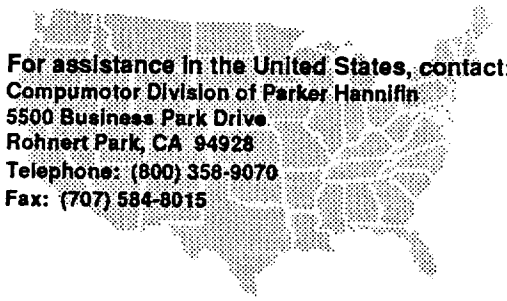
Personnel who are to install and operate the equipment should study this user guide and all referenced documentation prior to installation and/or operation of the equipment.

In no event will the provider of the equipment be liable for any incidental, consequential, or special damages of any kind or nature whatsoever, including but not limited to lost profits arising from or in any way connected with the use of this user guide or the equipment.

© Compumotor Division of Parker Hannifin Corporation, 1993
— All Rights Reserved —

The information in this user guide, including any apparatus, methods, techniques, and concepts described herein, are the proprietary property of Parker Compumotor or its licensors, and may not be copied, disclosed, or used for any purpose not expressly authorized by the owner thereof.

Since Parker Compumotor constantly strives to improve all of its products, we reserve the right to change this user guide and equipment mentioned therein at any time without notice.



For assistance in the United States, contact:
Compumotor Division of Parker Hannifin
5500 Business Park Drive
Rohnert Park, CA 94928
Telephone: (800) 358-9070
Fax: (707) 584-8015

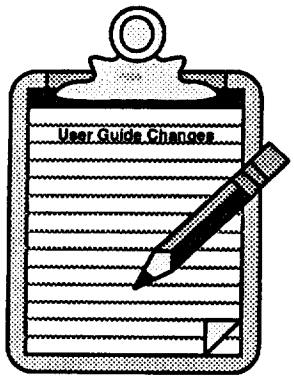


For assistance in Europe, contact:
Parker Digiplan
21 Balena Close
Poole, Dorset
England BH17 7DX
Telephone: 0202-690911
Fax: 0202-600820

Parker **Compumotor**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

User Guide Change Summary



The following is a summary of the primary changes to this user guide since the last version was released. This user guide, version 88-007719-01**J**, supersedes version 88-007719-01**H**.

The Model 4000 User Guides have been reorganized into three separate user guides:

Description	Part Number
Model 4000 User Guide	88-007719-01 K
Model 4000 X Command Language User Guide	88-011004-01 E
Model 4000 Options User Guide	88-013526-01 C
Model 4000 X Language Reference Guide	88-011004-01 E

Technical changes to the Model 4000 User Guide are described below:

- The following new statements were added to *Chapter 4, Software Reference*

  DEFINE ONRET

  ENABLE TRIG REV

  ENABLE REG SRV

  DEFINE TRIGDB

- The Index was updated

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TABLE OF CONTENTS

How To Use This User Guide v
Assumptions v
User Guide Contents v
Installation Process Overview vi
Developing Your Application vi
Installation Preparation vi
Related Publication vi

① **INTRODUCTION** **1**
Product Description 1
Model 4000 Features 1
Operator Panel Features 2
Optional Features 2
Sample Configuration 2
Operator Panel Keys and LEDs 3

② **GETTING STARTED** **7**
What You Should Have 7
Basic System Configuration 8
Pre-Wired Connections 8
Basic System Wiring 8
Model 4000 Functions 10
Model 4000 Tutorial 11
TEST Function Tutorial 12
Description of the Test Routines 13
Operations Tutorial 18

③ **INSTALLATION** **31**
Environmental 31
Electrical Noise 31
Airborne Contaminants 32
System Mounting 32
Operator Panel 34
Wiring Guidelines 35
Operator Panel Connections 36
Motor/Drive Connections 36
Encoder Connections 37
Auxiliary 2 40
Axis I/O 43
Fault Output Connections 45
Programmable I/O Connections 46
Input Power 49
PLC Operation 49
Thumbwheel Input Operation 50
System Verification 51
Model 4000 LEDs 51

④ **SOFTWARE REFERENCE** **53**
Model 4000 Command Language 53
Command Statement Description 53
Parent and Daughter Statements 54
Alphabetical List of Statements 56

⑤ **APPLICATION DESIGN** **161**
Front Panel Program Generation 161
Automatic Power-Up Execution 161
Creating a New Program 161
Running Your Program 163
Program Basics 163
Set-Up 163
Encoder 166
Inputs/Outputs 169

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Discrete I/O 169
System Data 169
Analog 170
Program Flow 173
Goto/ Gosub 173
Loop 173
Wait 173
Polling and Interrupts 174
Operator Interfacing 175
Error Handling 177
Command Summary 178
Complex Moves 178
Segmented Moves 182
Registration Moves 183
Command Summary 185
QQ Pointer Variables 185
Remote Operation 189
RS-232 Port Configuration 189
LCD Display 190
Multiple Program Manipulation 190
Downloading Programs 190
Editing Programs 191
Listing or Uploading Programs 191
Remote Execution 192
Application Developer Software 193
Communications 194
Common Problem Symptoms 194
Default RS-232 Parameters and POWER_UP Program 194
Error Messages and Prompts 194
Selecting the Program Name 195
Getting Back to Known Parameters (How to Get to Square One) 195
X Command Language 196

⑥ HARDWARE REFERENCE 197

Performance 197
Environmental 197
Inputs 197
Power 197
RS-232C 197
Home, CW, and CCW Limits 198
Triggers 198
Test 198
Estop 198
Encoder 198
Encoder Power 198
Programmable Inputs 198
Drive Fault 198
Pull-Ups 198
Joystick Release 199
Joystick Select 199
Joystick Trigger 199
Joystick Velocity Range 199
Analog Inputs 199
Outputs 199
+5VDC 199
RS-232C 199
Step, Direction, and Shutdown 199
Programmable Outputs 199
Axis-Programmable Outputs 200
CW & CCW Step 200
Fault 200
Board Monitor Alarm 200
Pin-Outs 200
Motor/Driver 200
Encoder 201
Programmable I/O 201
Auxiliary 1 202
Auxiliary 2 202

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Axis I/O 203
Fault Output 203
Operator Interface Connector (4-pin RJ-11 Jack) 203
Model 4000 Dimensions 204

⑦ MAINTENANCE & TROUBLESHOOTING 209

Maintenance 209
Spare Parts Table 209
Firmware Reference Table 210
Optional Features Table 210
Troubleshooting 210
Basic Troubleshooting Procedure 211
Diagnostic LEDs 211
Operator Panel 212
Reducing Electrical Noise 212
Model 4000 Power Supplies 213
Common Problems & Solutions 213
Closed-Loop Related Problems 214
Resetting the Model 4000 214
RS-232C Communications 215
Returning the System 215

APPENDIX 217

Functional Statement Grouping (Partial Listing) 217
Remote Commands and Program Statements 219
X Language Command Listing 221
Command Listing (X Series vs. 4000 Series) 223

INDEX 227

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

O V E R V I E W

How To Use This User Guide

This user guide is designed to help you install, develop, and maintain your system. Each chapter begins with a list of specific objectives that should be met after you have read the chapter. This section should help you find and use the information in this user guide.

Assumptions

This user guide assumes that the user has a fundamental understanding of computers, basic electrical concepts, basic motion control concepts, and basic serial communication (RS-232C) concepts.

User Guide Contents

This user guide contains the following information.

Chapter ① Introduction	This chapter provides a description of the product and a brief account of its features.
Chapter ② Getting Started	This chapter contains a detailed list of items you should have received with your Model 4000 shipment. It will help you to become familiar with the system and ensure that each component functions properly. You will learn how to navigate through menu-levels in this chapter.
Chapter ③ Installation	This chapter provides instructions for you to properly mount the system and make all electrical and non-electrical connections. Upon completion of this chapter, your system should be completely installed and ready to perform basic operations.
Chapter ④ Software Reference	This chapter explains Compumotor's Model 4000 programming language in detail. It describes statement syntax and system parameters that affect statement usage. An alphabetical listing of all statements, with a syntax and statement description for each statement is included. A separate reference guide is provided for the X Series language (***The Model 4000 Controller X Command Language Reference Guide***).
Chapter ⑤ Application Design	This chapter will help you customize the system to meet your application's needs. Important application considerations are discussed. Sample applications are provided.
Chapter ⑥ Hardware Reference	This chapter contains information on system specifications (dimensions and performance). This chapter may be used as a quick-reference tool for proper I/O connections.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter ⑦ Maintenance & Trouble-shooting

This chapter describes Compumotor's recommended system maintenance procedures. It also provides methods for isolating and resolving hardware and software problems.

Installation Process Overview

To ensure trouble-free operation, you should follow the installation procedures outlined in this user guide. Pay special attention to the environment in which the Model 4000 will operate, the layout and mounting, and the wiring and grounding practices used.

Developing Your Application

Before you develop and implement your application, there are several issues that you should consider.

① Clarify the requirements of your application. Clearly define what you expect the system to do.

② Assess your resources and limitations. This will help you find the most efficient and effective means of developing and implementing your application.

③ Follow the guidelines and instructions outlined in this user guide. Proper installation and implementation can only be ensured if all procedures are completed in the proper sequence.

Installation Preparation

Before you attempt to install this product, you should complete the following steps. Successful completion of these steps should prevent subsequent performance problems and allow you to resolve any potential system difficulties before they affect your system's operation.

① Become familiar with the user guide's contents so that you can find information that you need quickly.

② Develop a basic understanding of all system components, their functions, and interrelationships.

③ Complete the basic system configuration and wiring instructions (in a simulated environment, not a permanent installation) provided in *Chapter ② Getting Started*.

④ Perform as many basic moves and functions as you can with the preliminary configuration. Try to simulate (with no load attached) the task(s) that you expect to perform when you permanently install your application.

⑤ After you have familiarized yourself with and/or tested the system's functions and features, carefully read *Chapter ③ Installation*.

⑥ Begin the installation process. Do not deviate from the sequence or installation methods provided.

⑦ Before you begin to customize your system, check the system functions and features to ensure that you have completed the installation process correctly.

Related Publication

For more information on motion control concepts and Compumotor's complete product line and product capabilities, refer to the current *Compumotor Digiplan Positioning Control Systems and Drives.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

C H A P T E R ①

Introduction

The information in this chapter will enable you to:

- ❑ Understand the product's basic functions and features

Product Description

The Model 4000 Controller represents a significant advance in the technology of programmable motion controllers. Designed to control 1 to 4 axes of motion, this product is equipped with sophisticated multi-axis capability. The Model 4000 has a separate operator panel with an easy-to-read, backlit, alphanumeric LCD. These features, combined with a menu-driven control program, allow users to enter and store complex motion programs easily.

Nonvolatile, battery-backed RAM program storage and interactive programming make the Model 4000 ideal for small to medium-sized automation tasks. *In many instances, no other computer or programmable controller is needed.* For applications requiring host control, the Model 4000 may be programmed through industry-standard RS-232C or IEEE-488. The Model 4000 also provides open- or closed-loop control of all Compumotor motor/drives and standard 200-/400-step translators.

Model 4000 Features

- ❑ Menu-driven operation
- ❑ Simple BASIC-like motion control programming language
- ❑ Multiple programs may be entered
- ❑ 24 programmable inputs and 24 outputs compatible with OPTO-22 signal conditioning products
- ❑ Subroutine labels simplify programming
- ❑ Powerful program editor with insert, edit, copy, and delete functions
- ❑ Remote programming capability
- ❑ Application developer software for the IBM PC™ provided
- ❑ Teach mode uses analog joystick inputs or operator panel jog keys
- ❑ User-defined operator prompts for data entry
- ❑ Nonvolatile, battery backed RAM

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- ❑ Password restricted access
- ❑ Linear interpolation
- ❑ Continuous path motion
- ❑ Fault Output
- ❑ Flexible mounting arrangements (19" rack, back panel)
- ❑ Two RS-232C ports for programming operator messages or control of other devices (X drives, host computer, etc.)
- ❑ All I/O are optically isolated

Operator Panel Features

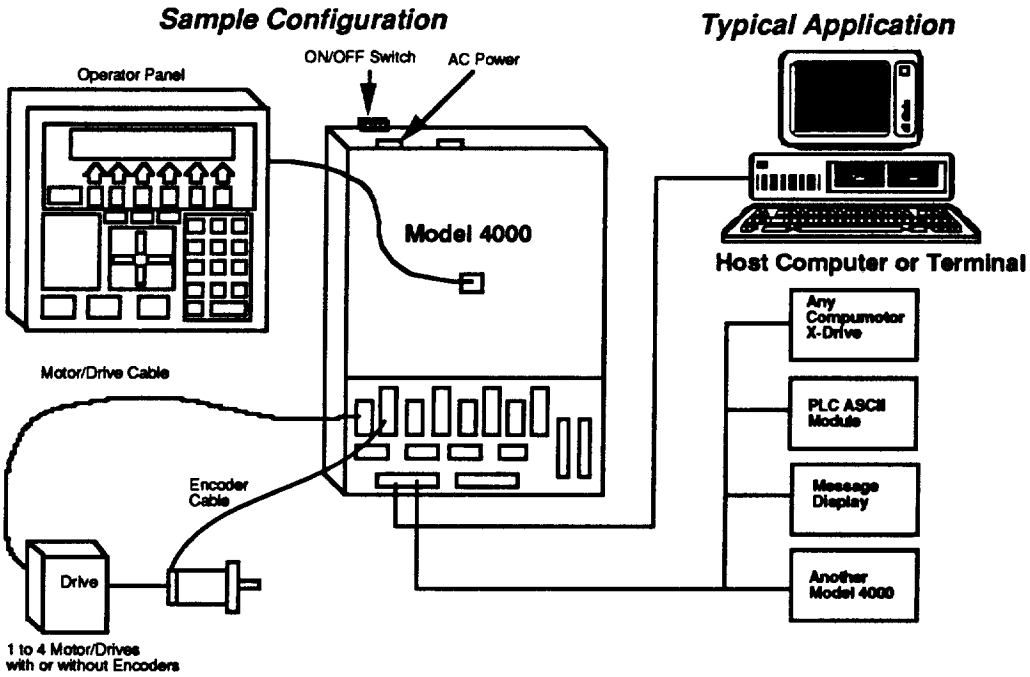
- ❑ Removable operator panel that can be mounted remotely
- ❑ Dedicated operator panel jog function for each axis
- ❑ Four-line, backlit LCD
- ❑ Six *soft* function keys labeled by the LCD change definition as required to simplify programming
- ❑ Sealed membrane keypad meets NEMA 12 specifications

Optional Features

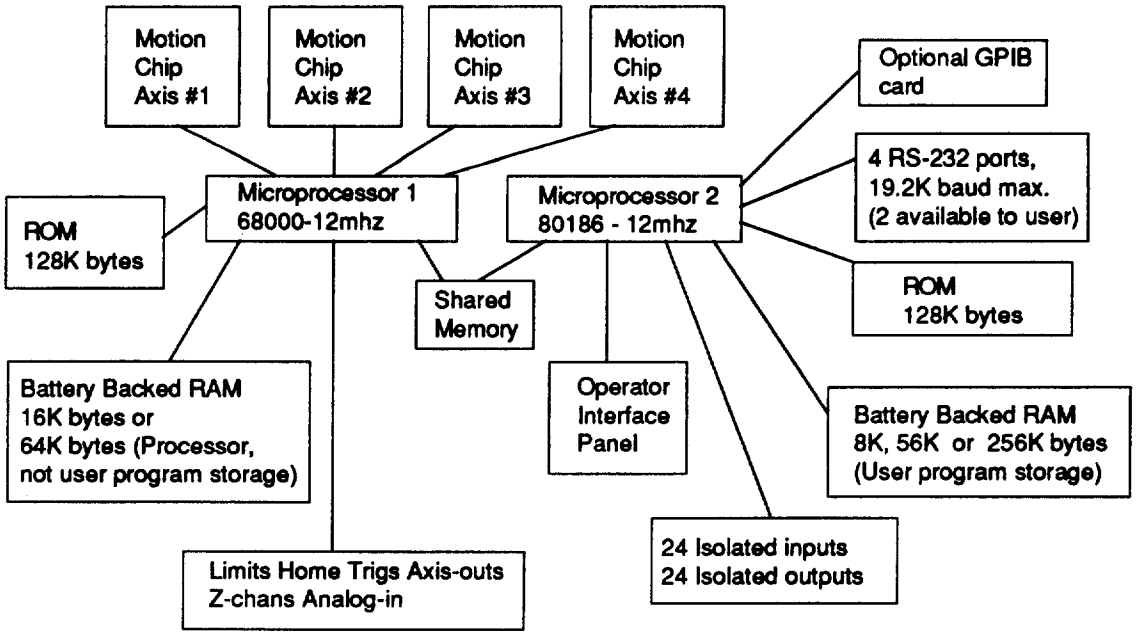
For complete information on these options refer to the ***Model 4000 Options User Guide***.

- ❑ Expanded memory (56K bytes)
- ❑ IEEE-488
- ❑ Contouring
- ❑ Following
- ❑ Multi-Tasking

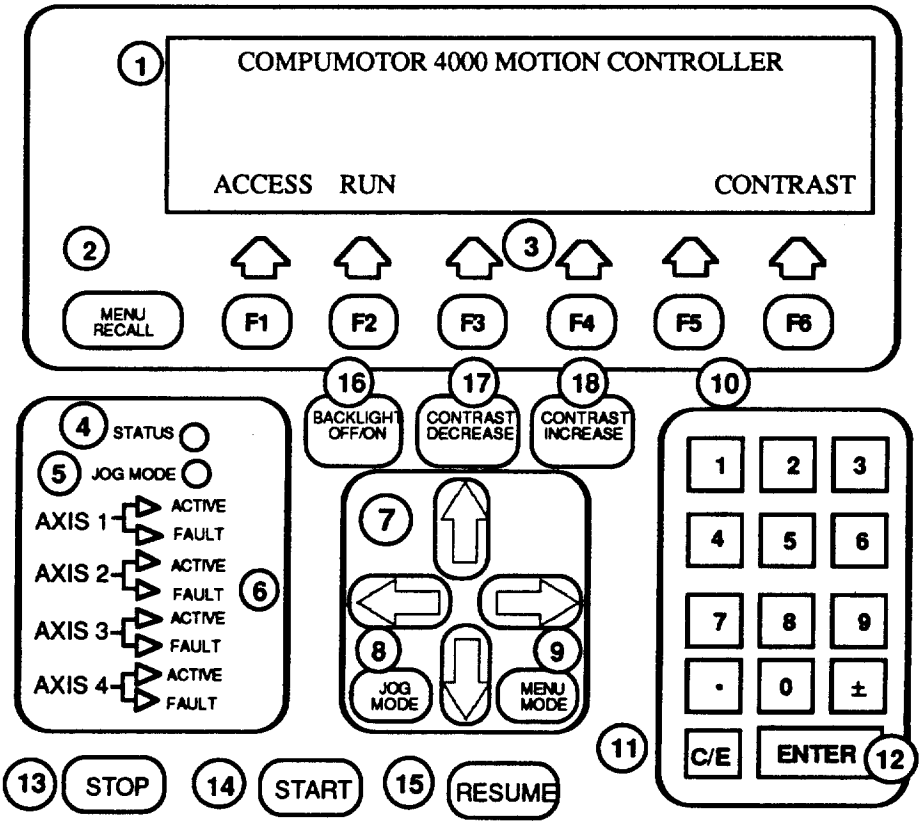
Sample Configuration



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Operator Panel Keys and LEDs



Display: The display is a 4-line, 40-character, backlit screen. (The backlight turns off automatically if you do not press an operator panel key after 3 minutes you can press any key to turn the backlight on). You can modify the screen's contrast (discussed later in this chapter). The display allows you to view, edit, or select the following types of information:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Menus, functions, intermediate functions, and options
- User-programmable information
- Programming status

② **MENU RECALL**: This key returns control to the previous menu. However, you cannot use it to return to the Start-Up menu from the Main menu, no matter how many times you press it. You must use the **GOBACK** function to return to the Start-Up screen (discussed later).

③ **Function Keys F1 - F6**: These are soft function keys. Soft function keys have different functions with each respective menu. You can select whatever function is shown on the display above the particular key. You can also program the function of these keys.

④ **Status LED**: When this LED is green, the Model 4000 is functioning properly. The LED will blink red whenever you press a key on the operator panel. The LED will be solid red if the microprocessor malfunctions. This indicates a non-recoverable error. You must power down (restart the system) to resume processing if a non-recoverable error occurs.

⑤ **Jog Mode LED**: When this LED is yellow, the Model 4000 is operating in Jog mode. This LED will not be on at any other time.

⑥ **Axes 1 - 4 Active/Fault LEDs**: When the controller sends steps to an axis, the axis' Active LED will be green. Each axis operates independently, the LEDs for each axis may turn on and off separately rather than in unison.

The Fault LED will be red when one of the following error conditions exist:

- Hardware limit is active
- A stall condition exists (only if the Stall Detection function is enabled)
- The ESTOP is activated
- An axis has reached a software limit
- The axis has been commanded to shutdown

The default **POWER-UP** program in the Model 4000 enables the hardware limits. If the unit is powered up with no limit switches connected properly, all four fault LED's will be red.

To disable the **FAULT LEDs** due to the limit inputs, use the following procedure:

① Press **ACCESS** (**F1**)
② Type **4000** and press **ENTER**
③ Press **IMMED** (**F3**)
④ Press **ENABLE** (**F5**)
⑤ Press **ETC** (**F6**)
⑥ Press **HLIMIT** (**F3**)
⑦ Press **NO NO NO NO** (**F3** - 4 times)
⑧ Press **ENTER**
⑨ Press **MENU RECALL** to return to the Main menu.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

⑦ **Arrow Keys**: These keys (⇐⇑⇓⇒) allow you to perform the following tasks:

- ❑ Scroll through statement and menu options.
- ❑ Scroll through programs that you created and stored in the Model 4000.
- ❑ If you are in Jog mode, you can use the arrow keys to jog the motors.
- ❑ If you are in Alpha mode, you can use the up and down arrow keys to scroll through the alphabet to display letters and other special characters that you may want to use.

⑧ **JOG MODE**: This key allows you to enter the Jog mode from the **ACCESS** or main Model 4000 menu. This mode allows you to jog (manually move) motors. You may not enter Jog mode using the operator panel keys if you are performing any of the following tasks :

- ❑ You must be at the main menu to enter Jog mode on the front panel
- ❑ Editing a program
- ❑ Running a program
- ❑ Printing to an external device

Refer to the **ENABLE JOG** statement description in *Chapter 4, Software Reference* for information about using jog in your program.

⑨ **MENU MODE**: This key exits control from the Jog mode and returns control to the previous menu. This key is also used for editing alpha characters in the Edit mode.

⑩ **Numerical Keypad**: This keypad allows you to select digits 0 - 9. The period (.) allows you to enter decimals. You can use the key to determine positive and negative values and set direction (CW or CCW).

⑪ **C/E**: This key allows you to clear or erase the character at the current cursor position.

⑫ **ENTER**: This key allows you to send input from the display to the Model 4000 for processing.

⑬ **STOP**: This key stops or interrupts program execution (including axis moves).

⑭ **START**: This key allows you to run or execute the default or last referenced program.

⑮ **RESUME**: This key allows you to reactivate processing of a program from the point at which it was stopped. You can use this key to resume a program that was interrupted by the **STOP** key. The program will begin from the point at which it was stopped. Moves that you interrupt with the **STOP** key will be completed when the program is resumed. If you stop a program and press the **MENU RECALL** key, you cannot resume the interrupted program (the **MENU RECALL** key returns control to the previous menu).

⑯ **BACKLIGHT OFF/ON**: This key allows you to turn the backlight on the front panel screen off or on.

⑰ **CONTRAST DECREASE**: This key allows you to decrease the brightness on the front panel screen.

⑱ **CONTRAST INCREASE**: This key allows you to increase the brightness on the front panel screen.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

C H A P T E R ②

Getting Started

The information in this chapter will enable you to:

- ❑ Verify that each component of your system has been delivered safely
- ❑ Become familiar with the system components and their interrelationships
- ❑ Ensure that each component functions properly by bench testing

What You Should Have

You should inspect your Model 4000 shipment upon receipt for obvious damage to its shipping container. Report any such damage to the shipping company as soon as possible. Parker Compumotor cannot be held responsible for damage incurred in shipment. The following items should be present and in good condition.

Part	Part Number
Operator Panel	CP4000
Model 4000 Main Unit	4000-NP

CP4000 Part	Part Number
Phone Cord, 8"	71-007683-01
Gasket	58-009135-01
Instruction Sheet	88-010716-01

4000-NP Part	Part Number
2 ea, 15D-25D, 10' Drive Cable	71-010432-10
Phone Cord (3 foot)	71-007683-03
User Guide	88-007719-01
AC Power Cord	44-000054-01
Model 4000 Application Developer Disk	95-010719-01
Model 4000 X Command Language User Guide	88-011004-01
Model 4000 Options User Guide (if IEEE488, Contouring, Multi-Tasking, or Following were purchased)	88-013526-01

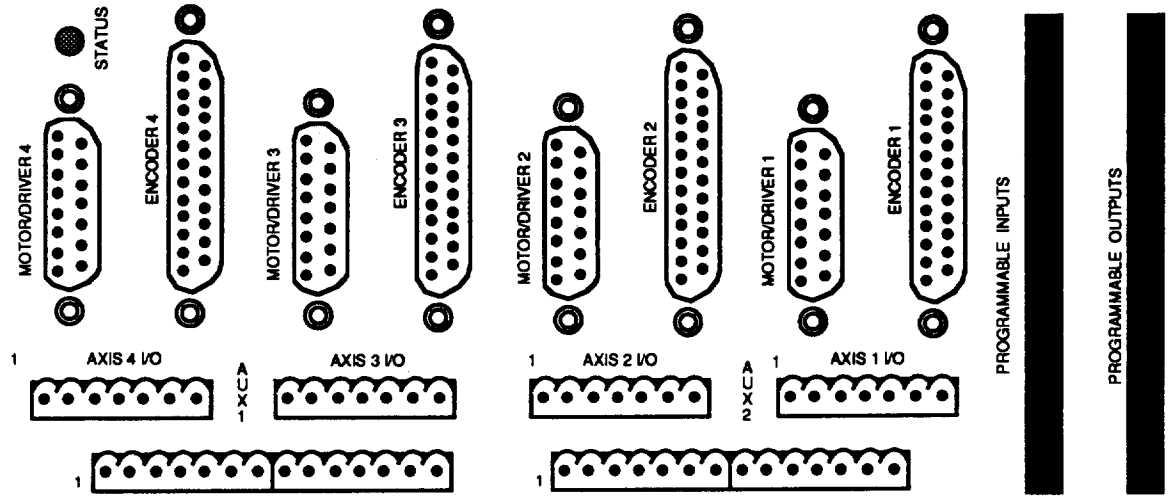
Option Part	Part Number
Model 4000 Rack Mount Kit	RMK4000
Instruction Sheet	88-010717-01

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If you need replacement parts, or would like to upgrade your system (expanded memory, IEEE-488, contouring, following, etc.), refer to Chapter 7, *Maintenance & Troubleshooting*. These parts (and their part numbers) are provided in this chapter along with information on ordering these options.

Basic System Configuration

You should have at least one motor/drive to proceed with the tests. The following figure shows the Model 4000's connectors as they appear on the unit.



Pre-Wired Connections

You should receive your Model 4000 with the following connections on AUX 1 and AUX 2 pre-wired (from the factory):

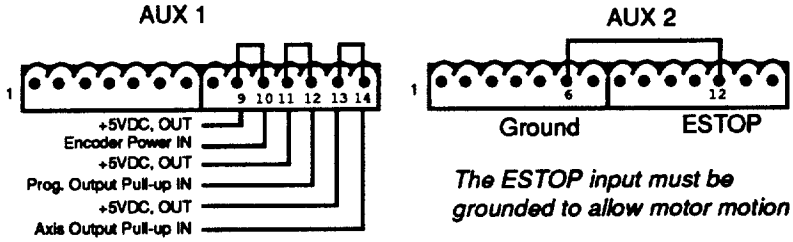
① ESTOP to Ground

② Axis Output Pull-up

③ Encoder Power-in

④ Programmable Output Pull-up

The following figure illustrates these pre-wired connections. Ensure that these connections are intact. ***You will not be able to perform the subsequent tutorials if these connections are not made.***

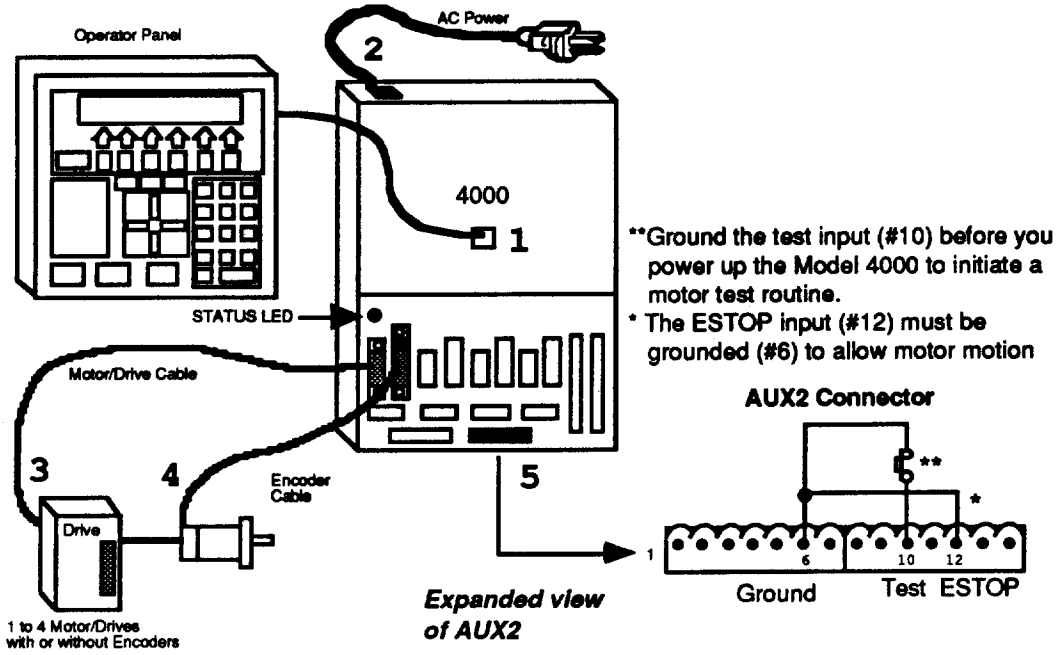


AUX1 Connector **AUX2 Connector**

Basic System Wiring and Test

Use the following instructions to wire the Model 4000 in a basic configuration (for test purposes only). The number of each step below is highlighted in the following diagram. *You will need a small flathead screwdriver to wire the motor/driver and the test input.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



① Connect the operator panel to the Model 4000 via the 4-wire *phone* cord.

② Connect the power cord to the top of the unit. It will later be attached to a 120VAC or 240VAC power source. ***Do not apply power at this time.***

③ Connect up to four motor/drives to the 15-pin D motor/driver connectors.

☛ Motion will occur if you complete this procedure correctly. Make sure all motor shafts are free to rotate.

④ Connect up to four encoders to the 25-pin D encoder connectors. ***Use of encoders is optional.***

⑤ To test the minimum operating functions of the unit, you must ground the test input. The test input is located on terminal #10 of the **AUX2** connector. Make sure that the ESTOP is grounded to allow motor motion.

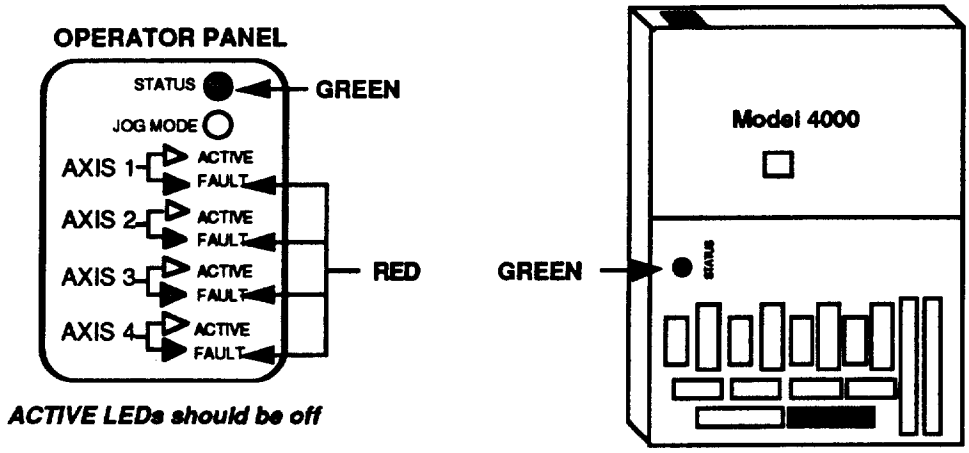
⑥ When you complete steps #1 - #5, plug the Model 4000's power cable into a 120VAC or 240VAC power source and turn the power switch on. All axes attached to the Model 4000 will move at 25,000 steps per second for a distance of 25,000 CW steps, and then 25,000 CCW steps. Following rotation, the board's monitor alarm will go on and the Status LED on the Model 4000 will turn red. *The display on the operator panel will remain blank during this test.*

⑦ Turn the unit off. Remove the test wire.

⑧ ***After you have removed the test wire, turn the Model 4000 on again.*** Check the following LEDs on the operator panel and the Model 4000 (refer to the following figure).

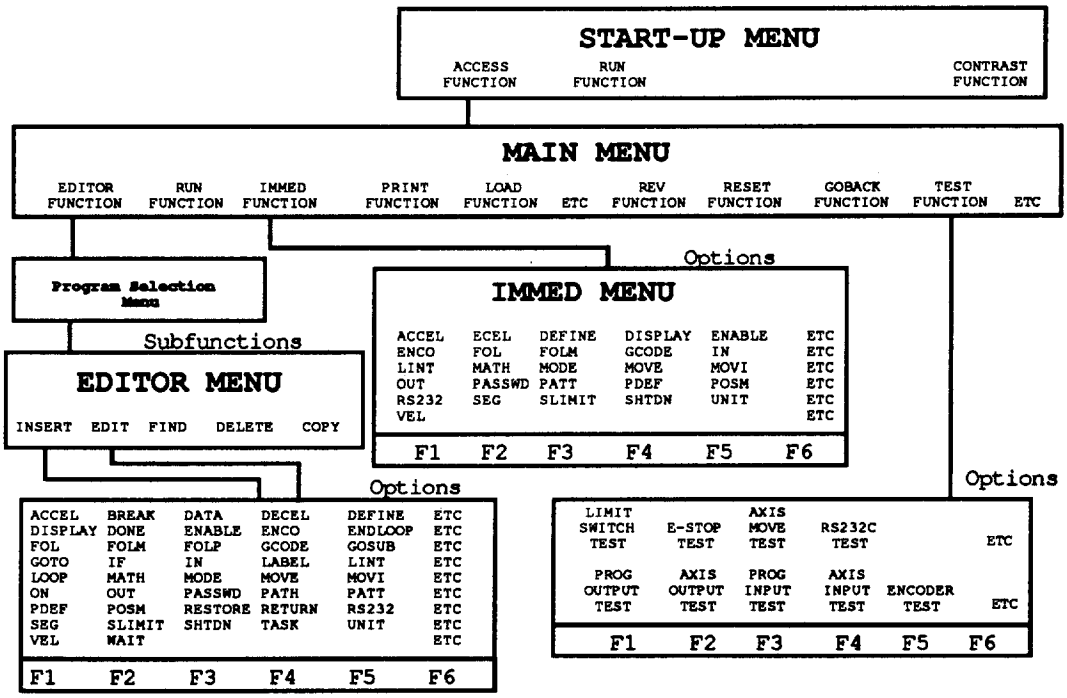
- ❑ The operator panel's Status LED should be on (green) and the Start-Up menu message should be displayed.
- ❑ The Axis Fault LEDs should all be on (red), indicating that the end-of-travel limit switches are not grounded.
- ❑ The Status LED on the Model 4000 should be on (green).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Model 4000 Functions

The Model 4000's operator panel displays menus that offer a variety of programming functions and options. Four components constitute the operator panel's programming capabilities: menus, functions, subfunctions, and options. The following figure is a hierarchical representation of these components.



Menus

Menus are the highest level of the Model 4000's operator panel hierarchy. From this level, you can select the task you want to perform (functions). Menus are similar to database directories. Within a directory resides a group of files. Within a menu resides a group of functions (e.g., **ACCESS**).

Functions

Functions are software instructions that you can send to the Model 4000 using the operator panel keys. They are not part of a program or statement sequence. Statements are discussed in Chapter 4, *Software Reference* (e.g., **ACCEL**).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Subfunctions

Some functions have subfunctions. They allow you to perform specific tasks within a function. For example, within the **EDITOR** function, you may **INSERT**, **EDIT**, **FIND**, **DELETE**, or **COPY**.

Option

Options are the lowest level of the Model 4000's software hierarchy. They allow you to edit, insert, select, and find data within function and subfunction levels. When you select a function or subfunction, options will be situated above the function keys. The same option may appear above different function keys from screen to screen, but its functionality will not change. Common options are defined below.

TAB — This option moves the cursor from word to word (left to right). You may use this option to position the cursor to enter values for various fields (i.e., line numbers or labels).

ALPHA — Since the Model 4000 renumbers lines whenever you insert statements, the **ALPHA** option allows you to create alphanumeric labels or prompts. You can scroll through the alphabet with the up (⇑)and down (⇓) arrow keys. Use the left (⇐)and right (⇒) arrow keys to position the cursor. You must use labels to differentiate programs, user prompts, subroutines, or statements (program line reference).

BEG — In many functions and subfunctions, you must specify a range of the program that will be affected (i.e., copy, insert, edit, and delete). This option allows you to quickly define the ***beginning*** of the program as the first parameter.

END — In many functions and subfunctions, you must specify a range of the program that will be affected (i.e., copy, insert, edit, and delete). This option allows you to quickly define the ***end*** of the program as the last parameter.

FND_LBL — This option allows you to scroll through and select existing program labels. The **FND_LBL** function works as follows:

If you do not use labels, the find label function will be inactive. To find a specific label, press **FND_LBL**. The first label will be displayed. Pressing **FND_LBL** will display the next label. **Labels are displayed in the order in which they were created (chronologically), not alphabetically.** When the desired label is displayed, press **ENTER** to select it.

ETC — Use the **ETC** key to show more options or go to the next level.

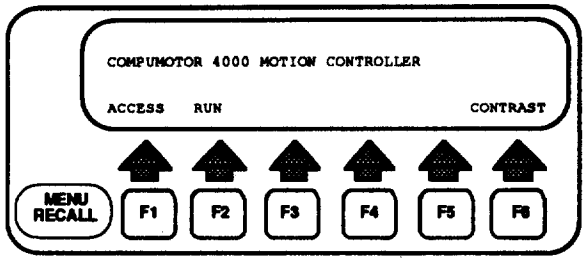
Model 4000 Tutorial

You must have an operator panel to complete the following tutorial. If you do not have an operator panel and will be using the Model 4000 with a host computer, refer to Chapter 3, Installation to wire your host computer. Chapter 5, Application Design explains Remote Operation (with a host computer).

Every Model 4000 function and subfunction is described in this section. The screen that appears when you select the function or subfunction from a menu is shown, as well as a description of the function. This section provides the procedures necessary to navigate through the Model 4000 control system.

Apply power to the system. The operator panel will display the Start-Up message shown below. The functions (**ACCESS**, **RUN**, and **CONTRAST**) that you can select above the function keys (**F1** - **F6**).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



From this screen, you can begin the TEST Function Tutorial. Follow the steps below.

TEST Function Tutorial

In this section, you will go through the **TEST** routine to ensure that the system is functioning properly. Built-in test routines are used to test the system's functionality.

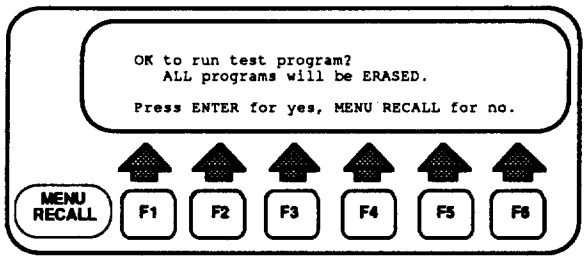
To enter the **TEST** routine from the Start-Up menu, perform the following steps:

① Press **ACCESS** (**F1**).

② Type **4000** (password) and press **ENTER** to get to the Main menu.

③ Press **ETC** to see more functions from the Main menu.

④ Press F5 to enter the **TEST** menu.

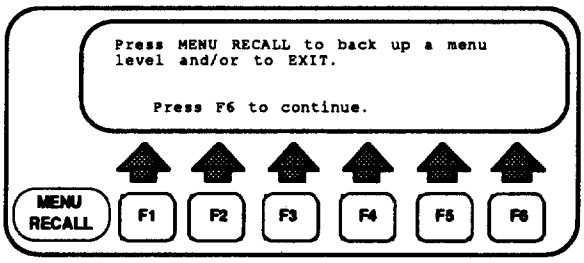


The **TEST** function copies the Model 4000 test program from ROM (read-only memory) into the Model 4000's memory. *All programs stored in the Model 4000's memory will be erased when you run the **TEST** function.*

The **TEST** function program helps you set up and test motor/drive connections, the limit inputs, the **ESTOP** input, the 24 programmable inputs, the axis inputs, the 24 programmable outputs, the 4-axis programmable outputs, the encoder inputs, and the RS-232C ports.

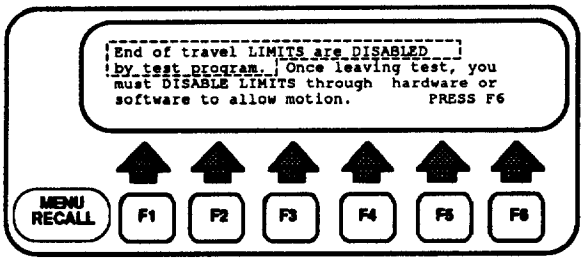
You should run the **TEST** function whenever you want to verify that Model 4000 functions are operating properly (axes, RS-232C, encoders, etc.). Use the following steps to run the **TEST** program:

① Press **ENTER** to run the **TEST** function. The second **TEST** program screen is shown below.



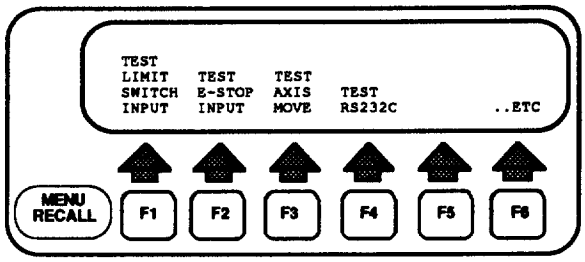
② Press **F6** to continue the **TEST** function. The third **TEST** program screen is shown below.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

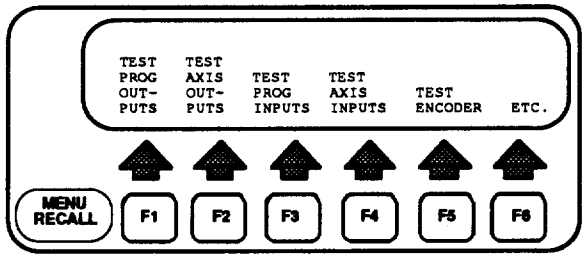


The first sentence will flash.

③ Press **F6** to continue the **TEST** function. The screen below shows the first **TEST** options screen. The tests that you may run are shown in the screen below.



④ Press **F6** to see more test options.



Press **F6** again to go back to the first Test Options screen.

Description of the Test Routines

This section will guide you through each of the Model 4000's tests options (shown in the two previous screens). *Remember, after you run any of the tests, press* **MENU RECALL** *to return to the previous screen (the first or second Test Options screens). If you press* **MENU RECALL** *again, you will return to the Main menu.*

Test Limit Switch Input

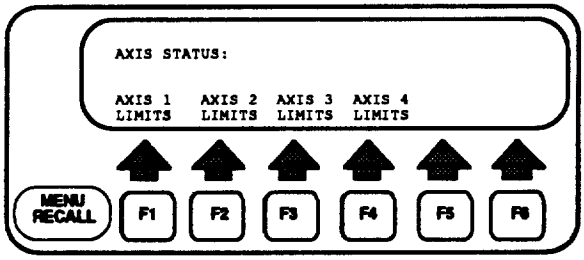
From the first Test Options screen, press **F1** *to test the limit switch input.* Whenever you run this test, the limits will be re-enabled (the fault LEDs will come on if limits are not grounded). This test continuously displays the state of the end-of-travel limits for all four axes. You may activate your limits to see if the Model 4000 recognizes them. When you enter the test limit function, both limits (CW and CCW) are active (*no motion*). If you ground a limit switch (axis input #2 or #3 connected to #6), the status will change.

The available options are:

- **LIMITS** (when CCW and CW are not connected)
- **CWLIMIT** (when CW is not connected and CCW is connected)
- **CCW LIMIT** (when CCW is not connected and CW is connected)
- **NOT MOVING** (when both limits are connected)

If none of the limits (CW & CCW) are connected, you should receive the following screen when you run this test.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

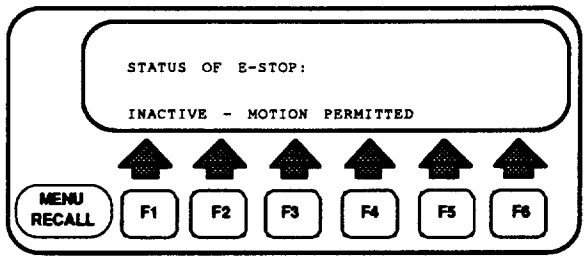


Related programming statement: **ENABLE HLIMIT**

Press **MENU RECALL** to return to the first Test Options screen. Press **F2** to begin the Test ESTOP Input.

Test ESTOP Input

This test reports the **ESTOP** input status (active/inactive). You can activate the **ESTOP** input and see if the Model 4000 recognizes it. The **ESTOP** input is jumpered at the factory (inactive, permitting motion) on AUX2, pins #6 & #12. If you disconnect/connect the **ESTOP**, the status will change. If the **ESTOP** input is active, motion is not permitted.



Related programming statement: **ON ESTOP**

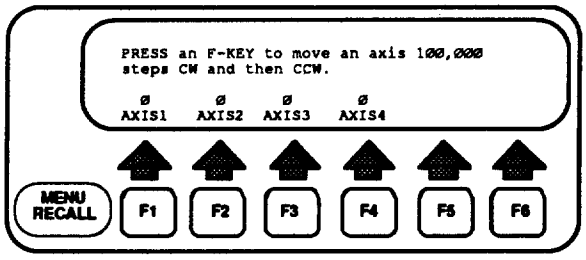
Press **MENU RECALL** to return to the first Test Options screen. Press **F3** to begin the Test Axis Move.

Test Axis Move

The Test Axis Move allows you to move a motor 100,000 steps CW and CCW. Press the function key below the axis number you wish to move. If the motor does not move, make sure the **ESTOP** input is grounded, check indexer/drive connections, drive power, etc. (refer to **TEST ESTOP INPUT**).

☛ This test will move your motors. The load should not be attached at this time.

This test, as shown in the screen below, will display the position of each axis continuously. Four axes positions will appear (even if you have less than four axes running) The Model 4000 sends 100,000 steps pulses CW and CCW to each motor drive connector.



Related programming statements: **UNITS, MOVE**

If the drives connected to the controller have a step pulse width requirement greater than 1 microsecond, this test will not work (i.e., Digiplan PK3 or SD Drives). Under program control, the pulses can be widened with the **ENCO MRES** statement.

Press **MENU RECALL** to return to the first Test Options screen. Press **F4** to begin the Test RS-232C.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Test RS-232C

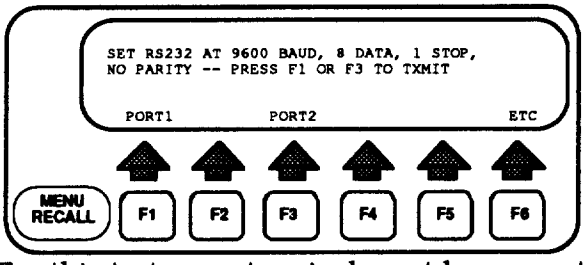
Do not attempt this test if you are using the Model 4000 in standalone mode. The Model 4000 will not respond if you run this test without a terminal. The 4000 Application Developer can be used in Terminal Emulator mode. Refer to Chapter 3, *Installation*, for proper RS-232C wiring. This test is normally skipped during the initial start-up procedures.

This test verifies that the Model 4000 and your terminal are communicating properly. You must configure your terminal's RS-232C parameters to the following settings:

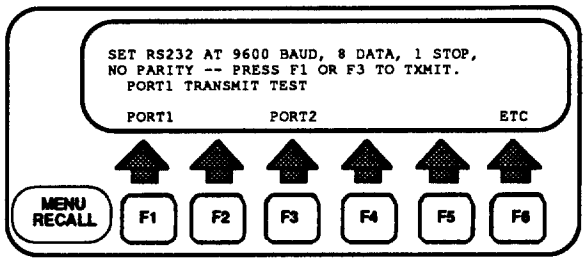
- Baud Rate: 9,600
- Data Bits: 8
- Stop Bit: 1
- Parity: No

If you cannot configure your terminal to these parameters, you will need to reconfigure the Model 4000's RS-232C parameters (**RS232** statement). ***The test will not function if the parameters do not match.*** You can then return to the **TEST** program and verify communication.

Press **F6** to start the **TEST RS232C** program. This screen will appear.



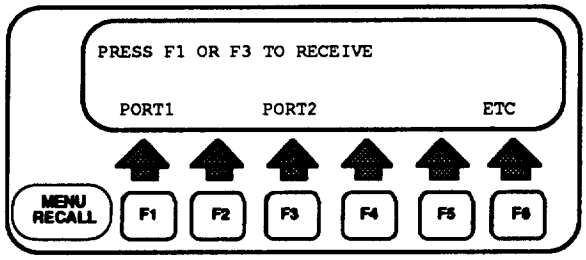
For this test, your terminal must be connected to Port 1 (the test applies to Port 2 as well). Press **F1** to select Port 1 and begin the transmit test. The following screen will be displayed.



The following message will appear on the computer terminal:

```
Port 1 Transmitting
```

Press **F6** (**ETC**) to test the receive function from the terminal. The following screen will be displayed

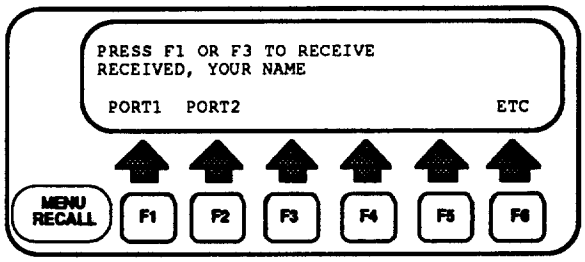


Press the appropriate function key (**F1** or **F3**) to select the desired port. The following message will appear nn your computer terminal:

```
Enter your name
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Type your name and press the Return key, the operator panel will respond with the following screen.



If you do not have the RS-232C connection configured properly, you will not be able to run these tests successfully. (Refer to *Chapter 7, Maintenance & Troubleshooting* for more information.)

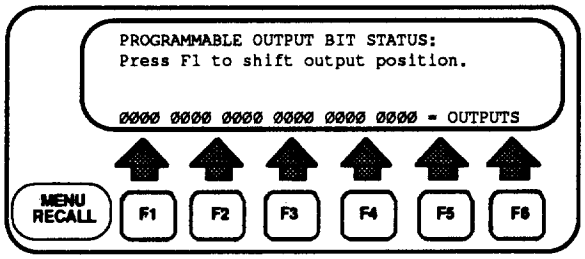
Related programming statements: **RS232, OUT PORT, IN PORT**

Press **MENU RECALL** to return to the first Test Options screen. Press **F6** to see the *second* Test Options screen. Press **F1** to begin the Programmable Outputs test.

Test Programmable Outputs

Each time you press **F1**, one of the output bits is set, starting with output #1 on pin #47 of the programmable output connector. If you press **F1** again, the current output will be cleared and the next output will be set. A set bit will be 0V and a reset bit will be 5V, using the default **DEFINE** setting of negative logic. To change the logic, refer to the **DEFINE** statement definition in Chapter 4, *Software Reference*. The status of the output is displayed as shown in the screen below.

If the programmable outputs fail to turn on, check the programmable output pull-up connection on the AUX1 connector. It should be connected to +5VDC or a voltage appropriate for your I/O. Refer to Chapter 3, *Installation*, for appropriate wiring instructions or diagrams.

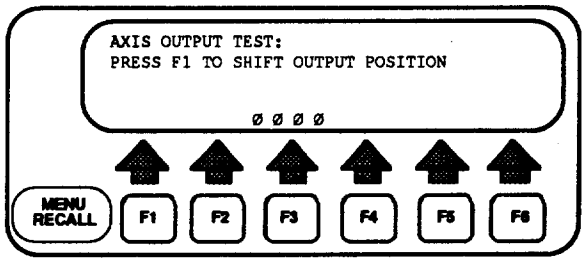


Related programming statements: **PATT, DEFINE OUT24, OUT OUT24, OUT BIT, DISPLAY**

Press **MENU RECALL** to return to the second Test Options screen. Press **F2** to begin the Axis Outputs test.

Test Axis Outputs

The Axis Outputs test sets the output on each of the Axis I/O connectors to 5V each time you press the **F1** key. If you press **F1** again, the next axis output will be set, the current axis will be cleared (0V), and the display will be updated. *Use a voltmeter across each axis output (pin #5) and ground (pin #6) to test the output.*



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

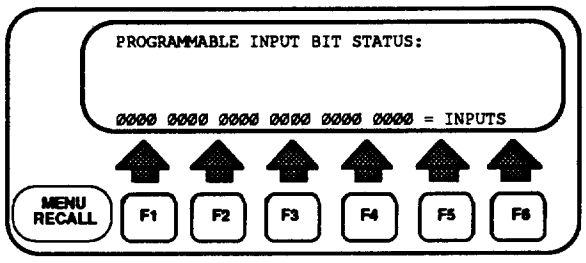
Related programming statement: **OUT POB**

Press **MENU RECALL** to return to the second Test Options screen. Press **F3** to begin the Programmable Inputs test.

Test Programmable Inputs

This test reads the 24 programmable inputs and continually displays their current state. This allows you to turn on and off each input while the display shows if the Model 4000 recognizes the change. Jumper an input to ground to verify that it is functioning properly.

Do not ground the +5VDC output (pin #49) on the connector. Refer to Chapter 6, *Hardware Reference* for the exact pin out.All even pins are logic ground.

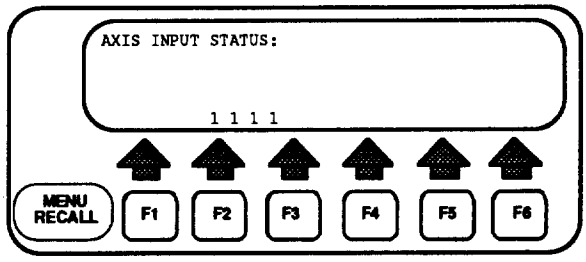


Related programming statements: **DEFINE IN24, IN IN24, IN BIT, DISPLAY, IF IN24, IF BIT, ON IN24, ON BIT**

Press **MENU RECALL** to return to the second Test Options screen. Press **F4** to begin the Axis Inputs test.

Test Axis Inputs

This test checks the state of the trigger inputs on each of the Axis I/O connectors and continually displays their current state. Jumper the trigger input (pin #4) to ground (pin #6) on an Axis I/O connector to verify operation.



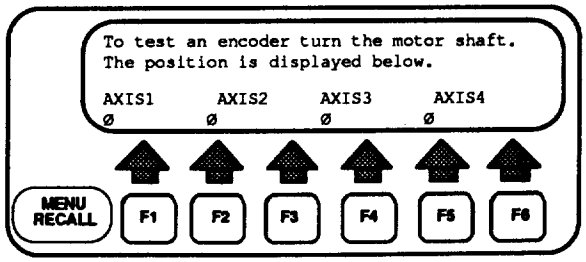
Related programming statements: **IN TRIG, IF TRIG, ON TRIG**

Test Encoder

Skip this test if you are not using an encoder or if using only Compumotor AR-C Absolute Encoders.

The encoder test shuts down the drive so that you can turn the motor shaft easily (the red Fault LEDs for each axis will be on). It also displays the encoder position repetitively so you can see the results of turning the motor (encoder) shaft. The drive will re-enable when you exit this test.

Turning the motor shaft CW facing the motor, should increase the encoder position. If the encoder position decreases, refer to *Chapter 3, Installation*, or *Chapter 7, Maintenance & Troubleshooting* for assistance.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Related programming statements: **ENCO, MOVE, MODE, POSM**

To return to the Start-Up menu , press the following keys on the operator panel:

① **MENU RECALL** (twice)

② **ETC**

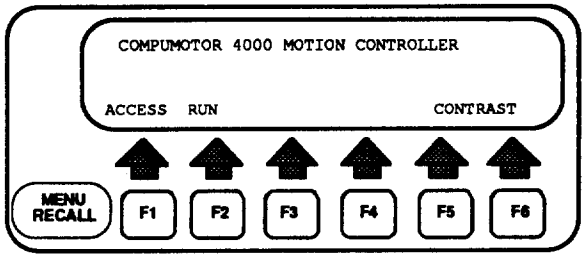
③ **GOBACK**

Operations Tutorial

After completing the Test Function Tutorial, you are ready to try the Model 4000's operational functions. These are the functions that allow you to create, modify, and run programs. The functions are presented in the order that they appear on Main menu. For more information on programming, refer to Chapter 4, *Software Reference.*

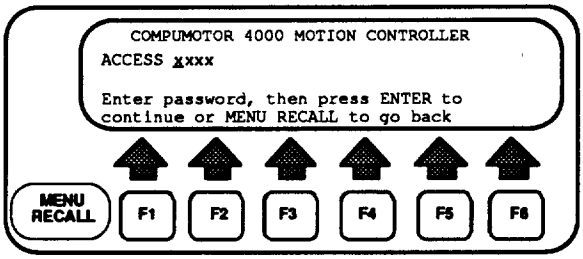
Start-Up Menu

You should have returned to the Start-Up menu.

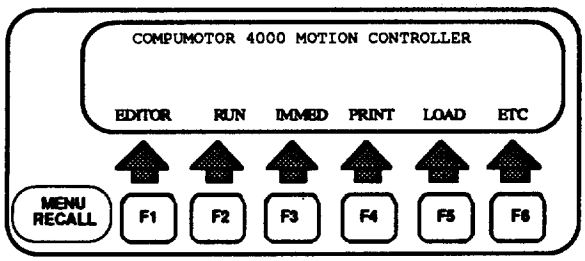


Note the function above **F1–ACCESS**. This is where we will begin the discussion of the menus, functions, subfunctions, and options. From the Start-Up menu, press **F1** to select **ACCESS**. The Model 4000 will display the screen shown in the screen below.

ACCESS Function F1: From Start-Up Menu

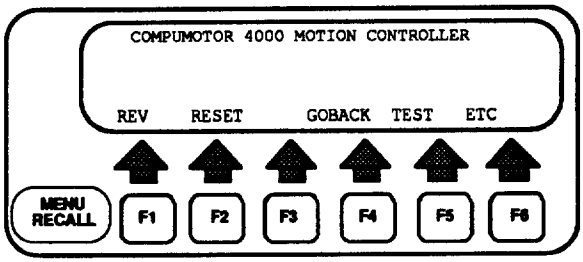


To continue, enter **4ØØØ** (*default password*) using the numerical keypad (the cursor is already positioned for you). You can define your own password (refer to *Chapter 4, Software Reference*). You can only use digits (Ø - 9) in the password. After you enter **4ØØØ**, press **ENTER.** The Main menu (shown below) will be displayed:



If you press **F6** (**ETC**), you will be able to select several additional functions from the Main menu. These menus provide paths to functions.

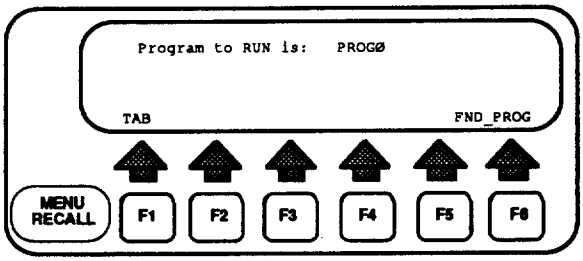
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



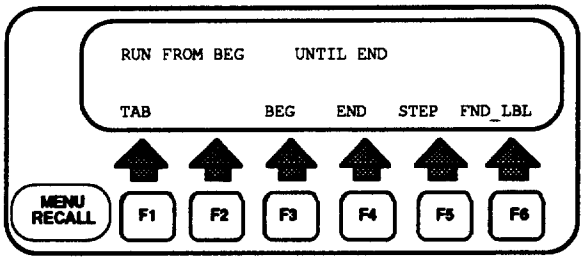
From the first Main menu screen, press **F2** to enter the **RUN** function. The program selection screen (shown below) will prompt you for the name of the program you want to **RUN**. Use the **FND_PROG** option to select a program.

RUN Function F2: From Start-Up Menu

The **RUN** function allows you to initiate program execution. The default statement (**RUN FROM BEG UNTIL END**) runs the entire program. You may, however, specify which portion of the program you want to run by entering program line numbers or new labels to replace the **BEG** and **END** labels.



Press **ENTER** to go to the next menu.



RUN also appears as a function within the Main menu. **RUN** performs identically in both instances, from the Start-Up or Main menu.

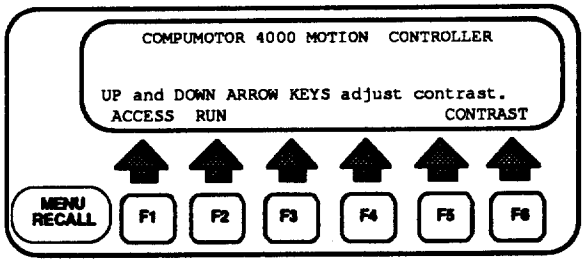
The **STEP** option allows you to execute one statement at a time. After each statement is executed, the Model 4000 waits for you to press the **ENTER** key before it executes the next statement. To step through the program, press **STEP**. Run the program by pressing **ENTER**. Each time you press **ENTER**, the Model 4000 will execute the next line. *At this point, you have not entered any programs. The Model 4000 will respond with* **PROGRAM FINISHED** *if you use the* **RUN** *function now.*

The **START** key, which was discussed earlier in this section, will also allow you to run an entire program, starting at the beginning.

After you receive the **PROGRAM FINISHED** message, press **MENU RECALL** to return to the Start-Up menu. Once you press **MENU RECALL**, you will not be able to use the **RESUME** function to continue the execution of an interrupted program.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CONTRAST Function F6: From Start-Up Menu



Press F6 to enter the CONTRAST menu. (You can also use the front panel keys to adjust the **CONTRAST**.) You can use the up (⇑)and down (⇓) arrow keys to modify the contrast of the Model 4000's operator panel display. Press the up key to increase the contrast. Press the down key to reduce the contrast (until the screen becomes blank—no characters can be seen). If you reduce contrast completely and accidentally exit from the **CONTRAST** function (e.g., press the wrong key and lose your place in the system), you can return to the **CONTRAST** function with the following steps:

① Cycle power to the system (turn the unit off and on).

② Press **F6**.

③ Press and hold the up arrow key to increase contrast and retrieve the screen.

At this point, we have covered all of the Start-up menu-level screens. Return to the Main menu and explore the functions and subfunctions that are available. Options—the lowest level—were discussed earlier in this section. Refer to *Common Options*.

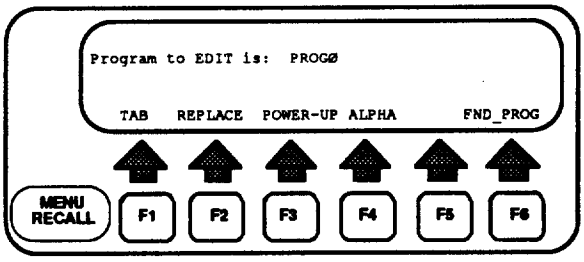
① Press **MENU RECALL** to return to the Start-Up menu.

② Press **F1** to enter the **ACCESS** function.

③ Enter **4000** and press **ENTER**.

EDITOR Function F1: From Main Menu (1st Screen)

Press **F1** to start the **EDITOR** function. Before entering the Editor, the intermediate screen below above will prompt you for the name of the program you want to edit. The default program name (the program shown) will be the most recent program that you selected for editing. Each of the options provided on the intermediate screen are described below (the **TAB** option allows you to move the cursor).



Replace

If you edit an existing program, the Model 4000 first copies that program so that you can save the *original version* when you exit the editor. If the program is larger than the available free memory space, an error message will prompt you to use the **REPLACE** option. The **REPLACE** option does not copy the original program before it allows you to edit it (all edits to that program are final). The system will remind you that your edits are final with the following message:

Use **REPLACE** option, **COPY** cannot be saved

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Power-Up

The **POWER_UP** program sets the default values of the **RS232** statement, as well as several other program parameters. If the **POWER_UP** program exists, the prompts and character echoes have the default value of **ON**. If the **POWER_UP** program does not exist, the prompts and character echoes have the default value of **OFF**. If the echo is turned off, you cannot download programs from the Model 4000 Application software. If you need to modify the default **POWER-UP** program, be sure to keep the **RS-232** statements set to avoid future communication problems.

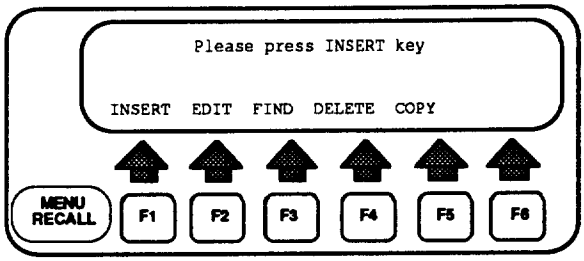
Alpha

To choose a different program, you can enter the name of another existing program with the **ALPHA** function and the numeric keypad. **PROGØ**, **START**, **HOME** are valid program names, but **BEG** and **END** are invalid because they are reserved labels. You cannot give a **PROG** statement the same name as a **LABEL** statement.

Find-Prog

The **FND_PROG** option allows you to scroll through the existing program names. Each time you press **F6**, the next program name in memory will be displayed at the cursor position.

Press **ENTER** to select the program to edit and you will enter the **EDITOR**. The subfunctions are displayed on the bottom line.



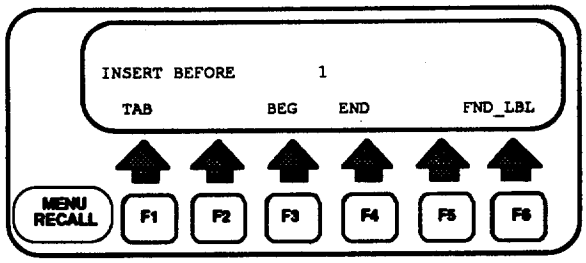
Line numbers only increment by 1. After an insertion or deletion, all line numbers are automatically renumbered. Thus, a program always starts with 1, and all subsequent line numbers differ by 1.

If you are deleting, inserting, or editing with the Model 4000 and power goes out, all changes to the program in the editor will be lost. If you are not deleting, inserting, or editing, no programs will be lost. However, when you turn the power back on, there may be two copies of the program you were editing when the power went out.

INSERT Subfunction F1: From Editor Menu

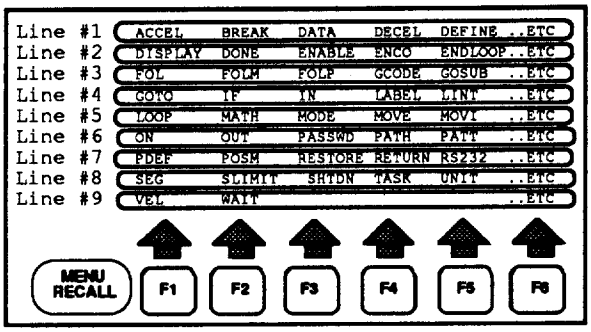
Press F1 from the EDITOR function to select the INSERT subfunction. **INSERT** allows you to insert lines before a line number or a label that you specify. You can determine where to insert data by entering the desired line number or label on line 1 of the display (following **BEFORE** as shown in the screen above). The very first line that you insert becomes line 1. If you are editing a new program, you will be prompted to **INSERT** before line 1. If a program exists, the data that was associated with the line or label you specified is now one line below the inserted data, and is displayed on the third display line. All subsequent lines will be renumbered. The second display line will remain blank except for the line number.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



If you press **BEG** or **END**, you will insert lines at the beginning or the end of your program respectively.

Press **ENTER** to begin inserting lines into a program. The fourth display line contains program statements that you can scroll through with the up/down (⇑ ⇓) arrow keys or with the **ETC** key. Press the appropriate function key to select the statement. There are nine different lines of statements. The statements available at this level are shown below. IEE-488 (**DEFINE GPIB**), Following (**FOL, FOLM**), G-Code (**GCODE**), and Contouring (**PATH**) statements will have no affect unless the appropriate option is purchased and installed in the Model 4000. For those users who have one or more of the above options, refer to the *Model 4000 Options User Guide* for further information.



Original line numbers that follow inserted text are increased by the number of lines inserted. If the line number or label that you specify as an insertion point (**BEFORE**) is not found, the following error message will be displayed.

LINE/LABEL NOT FOUND

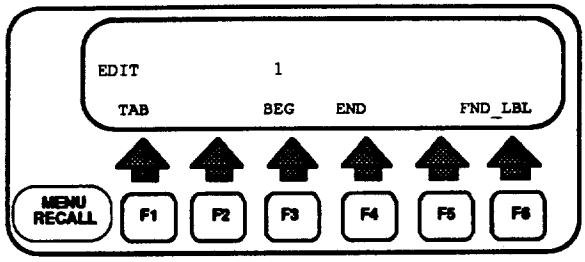
If you press **MENU RECALL** instead of the **ENTER** key, the current line that you are inserting will be erased and the screen shown below will be displayed. If you press **INSERT** again, the line that you were previously inserting is the line presented for insertion. This process allows you to clear a line if you make a mistake (You must recognize the mistake and clear the line before pressing **ENTER**.)

Press **MENU RECALL** to return to the Editor menu. Press **F2** to select the **EDIT** option.

EDIT Subfunction F2: From Editor Menu

EDIT allows you to edit lines starting with a line number or a label that you specify. You can determine where to edit data by entering the desired line number or label on display line 2 (as shown below). The **BEG** and **END** symbols allow you to enter **EDIT** mode at the beginning or the end of the program, respectively.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



After you enter the program line that you want to edit, press **ENTER** to begin editing. The group of statements that are available for the **INSERT** subfunction are also available for the **EDIT** subfunction.

After you press **ENTER**, the system will verify that you have entered a valid line number or label. The first line will be the line that you want to edit. The first, second, and third lines are the subsequent lines of the program. The bottom line contains statements that you can scroll through with the up and down arrow keys and select with the appropriate function key.

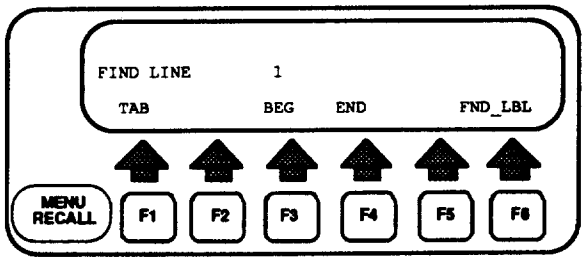
If a program is not stored in the Model 4000, you will not be able to use the **EDIT** subfunction (you must use the **INSERT** subfunction to create a new program). *You must have an existing program to edit.* If you attempt to edit an invalid program line, the Model 4000 will respond with: **INVALID DATA ENTERED**. You must press **ENTER** to continue.

MENU RECALL releases control from the **EDIT** subfunction and returns control to the **EDITOR** function's Main menu. If you press **MENU RECALL** instead of **ENTER** , the current line that you are editing will be erased and program control will return to the **EDITOR** function's Main menu. If you press **EDIT** again (*from the* **EDITOR** *function's Main menu*), the Model 4000 will display the line that you were previously editing as the first line for you to edit. This process allows you to clear a line if you make a mistake. (*You must recognize the mistake and clear the line before you press* **ENTER**.)

Press **MENU RECALL** to return to the Editor menu. Press **F3** to enter the **FIND** subfunction.

FIND Subfunction F3: From Editor Menu

You can locate a line number or a label name in a program with the **FIND** subfunction. Enter the desired line number or label on line 2 (following **FIND LINE** as shown above). The **BEG** and **END** options (**F3** and **F4**) allow you to quickly select the first or last line number or label in a program. If you press **ENTER** after selecting one of the function keys, the Model 4000 will search for the specified line or label. If it finds the requested information, it displays it on the second line of the display. The first display line contains the line (or label) that precedes the specified line and the third display line contains the line (or label) that follows the line. The fourth display line contains the intermediate functions: **INSERT EDIT FIND DELETE COPY**. You can scroll through programs with the up and down arrow keys.



If the system does not find the line number or label that you specified, the following error message will be displayed: **INVALID LINE/LABEL ENTERED**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Press **MENU RECALL** to return to the Editor menu. Press **F4** to select the **DELETE** option. The screen shown below will be displayed:

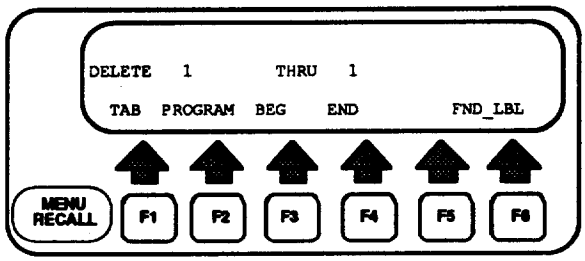
DELETE Subfunction F4: From Editor Menu

The **DELETE** subfunction erases one or more statements or program lines from a program. To delete a single statement, use the following procedure:

① Use the operator panel to enter the first line that you want to delete (following **DELETE**, as shown above).

② Use the operator panel to enter the last line to be delete deleted (as shown below).

③ Press **ENTER**.



The **DELETE** subfunction deletes any valid ***range*** of lines and/or labels. You may enter any valid combination of line numbers and labels to be deleted (e.g., you may enter a label as the first delete parameter and a line number as the second delete parameter, or vice versa).

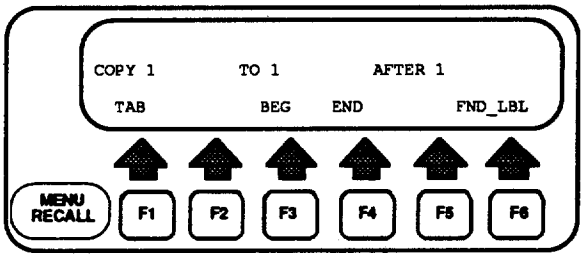
If the value of the first delete parameter is greater than the second parameter, the error message **INCORRECT LINE/LABEL NUMBER ENTERED** will be displayed. If either or both of the parameters that you enter do not exist, the error message **INCORRECT LINE/LABEL NUMBER ENTERED** will be displayed.

If you wish to delete an entire program, use the **DELETE PROGRAM** option (**F2**). *The* **DELETE BEG THRU END** *deletes all lines except the* program name. *You will have an empty program.*

Press **MENU RECALL** to return to the Editor menu. Press **F5** to select the **COPY** subfunction.

COPY Subfunction F5: From Editor Menu

COPY allows you to copy lines from a range of program lines (following **COPY** and **TO**, as shown below) to a specific location in a program (following **AFTER**). You can combine line numbers or label points for range, you may also specify a line number or label for point of insertion.



If you enter an invalid line copy parameter, the system will display the **INVALID LINE/LABEL NUMBER ENTERED** error message. The first copy parameter (following **COPY**) must be less than or equal to the second copy parameter (following **TO**).

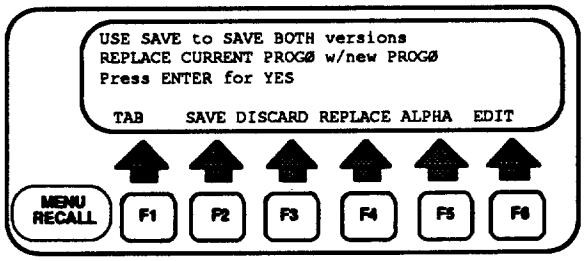
After processing the **COPY**, the first display line will contain the line (or label) that precedes the copied line(s). The second display line will contain the first copied line. If you copy one line, the third display line will display the line that follows the copied line. If you copy several lines, the third line will display the second line of the copied program. The fourth

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

display line contains the intermediate functions. You can scroll through programs with the up and down arrow keys.

Press **MENU RECALL** twice and you should see the exiting editor menu (as shown below) unless you are editing a new program, in which case the Main Menu will be displayed.

Exiting the Editor



Each of the Exiting menu options are described below.

To REPLACE the Original Program

The new program will replace the old copy if you press **ENTER**. You can change the new program name (at the far right of the previous screen) and the program will be stored with the new name. You can use the **ALPHA** option to create a new name.

To SAVE Both Versions

If you want to save both the original version and the recently edited version, use the **SAVE** option (**F2**). The default **SAVE** command line is shown in the previous screen example. When using the **SAVE** option, either or both of the program names must be changed, because two programs may not be saved with the same name. Use the **ALPHA** option and the numeric keypad to change one or both of the program names. The system will respond with the following message:

SAVE ORIGINAL PROGØ & NEW PROGØ

To DISCARD New Edits

To save the original version of the program and delete the edited version of the program, use the **DISCARD** option (shown in the screen above). It does not matter what program name is entered with the **DISCARD** command, only the edited copy will be deleted. The system will respond with the following message:

DISCARD edits just made to PROGØ

To return to editing the program, press the **EDIT** option (**F6**).

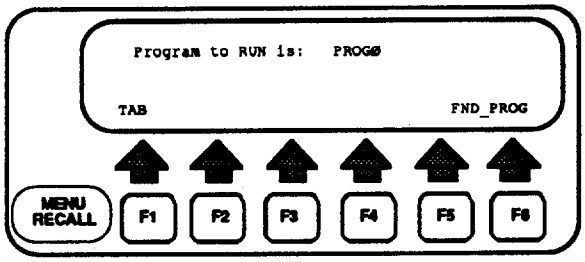
We have discussed all of the **EDITOR** function's subfunctions: **INSERT**, **EDIT**, **FIND**, **DELETE**, **COPY**, **REPLACE**, **SAVE**, and **DISCARD**. We will now return to the Main menu and discuss the remaining functions.

- **RUN**
- **IMMED**
- **PRINT**
- **LOAD**
- **RESET**
- **REV**
- **GOBACK**

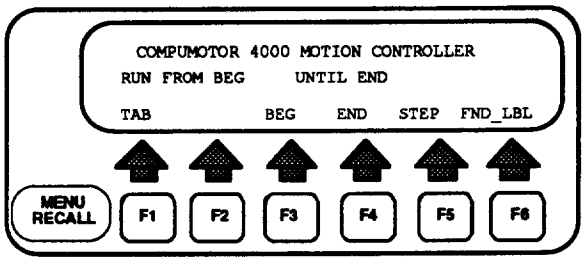
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RUN Function F2: From Main Menu (1st Screen)

The **RUN** function allows you to initiate program execution. First, you are asked which program you wish to run (refer to the example below).



Use the **FND_PROG** option to select the program you want to **RUN** (if not already displayed). Press **ENTER** to continue the **RUN** function or **MENU RECALL** to return to the Main menu.



The **RUN** function allows you to initiate program execution. The default statement (**RUN FROM BEG UNTIL END**) runs the entire program. You may, however, specify which portion of the program you want to run by entering program line numbers or labels to replace the **BEG** and **END** labels.

RUN also appears on the Start-Up menu. **RUN** performs identically from both menus (the Start-Up menu or the Main menu). Refer to the **RUN** menu item earlier in this section.

The **START** key, which was discussed earlier in this section, will also allow you to run an entire program.

① Press **MENU RECALL** to return to the Main menu.

② Press **STOP** to stop the program from running.

③ Press **MENU RECALL** to return to the Main menu.

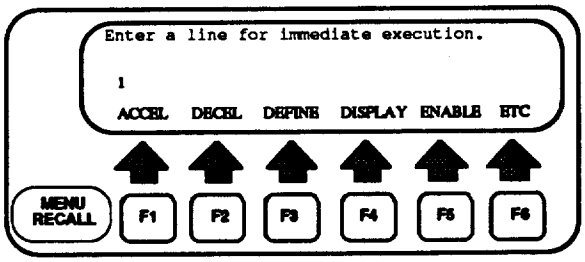
It is possible to enter single-step mode operation anywhere in a program by pressing the **STOP** key, and then immediately the **ENTER** key. The program will step through subsequent program lines each time the **ENTER** key is pressed. Normal program execution can begin again by pressing the **RESUME** key.

When communicating with the Model 4000 from a remote port (not the operator panel) use the **SELECT** function to select which program to run.

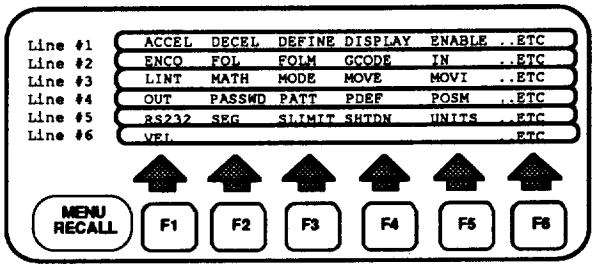
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IMMED Function F3: From Main Menu (1st Screen)

Press F3 to enter the IMMED function. The **IMMED** function allows you to execute one statement immediately. Enter the statement that you wish to process on the second line. The cursor is preceded by line number 1. *No branching can be used here.* This function is also available from remote (RS-232/IEEE-488) interfaces.



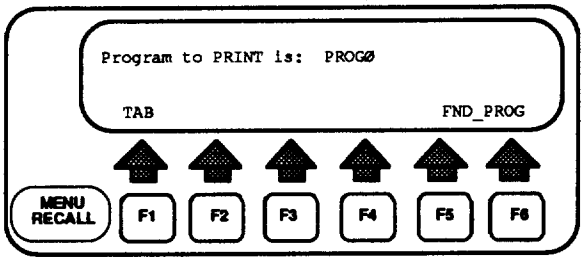
Use the up and down arrow keys to scroll through an alphabetical list of statements (select statements with function keys). There are six lines of statements to choose from (refer to the screen below). **DEFINE GPIB, FOL, FOLM,** and **GCODE** statements will have no affect unless the appropriate option is purchased and installed in the Model 4000.



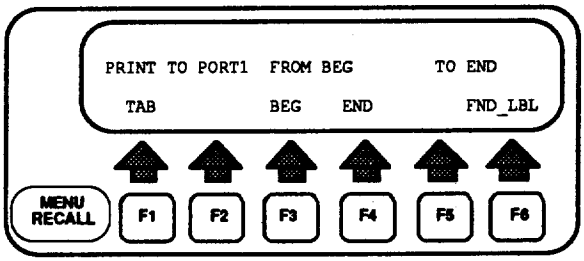
Press **MENU RECALL** to return to the Main menu.

PRINT Function F4: From Main Menu (1st Screen)

The **PRINT** function prints program statements to an RS-232C port. You must select which program you wish to **PRINT** with the screen shown below.



You can scroll through programs with the **FND_PROG** option. Press **ENTER** when the program you wish to **PRINT** appears or, press **MENU RECALL** to return to the **MAIN MENU**.



The **PRINT** function prints program statements to the port that you specify. Ports 1 and 2 are RS-232C ports. This function will only print to an external device (computer, terminal, or printer).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

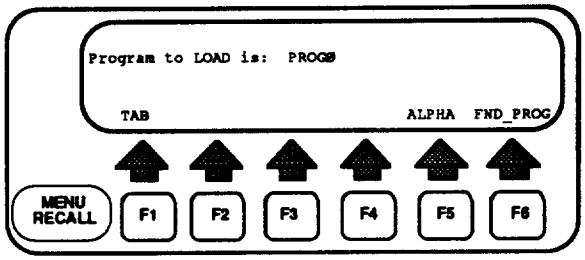
You can replace the **BEG** and **END** labels (print parameters following **FROM** and **TO**) with program line numbers or labels. The first print parameter (following **FROM**) must be less than or equal to the second print parameter (following **TO**).

Press **MENU RECALL** to return to the Main menu.

When communicating with the Model 4000 from a remote port (not the operator panel), use the **SELECT** function to select which program to **PRINT**.

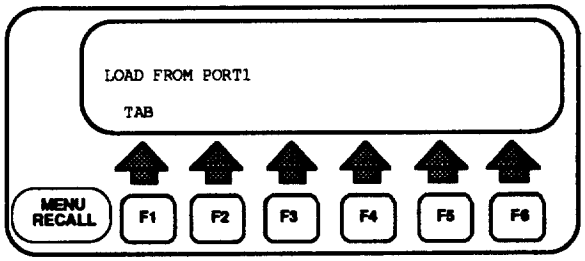
LOAD Function F5: From Main Menu (1st Screen)

The **LOAD** function allows you to enter programs into the Model 4000 via a remote RS-232C port (not the operator panel). First, you are asked to enter the name of the program to be loaded.



The next screen will ask you which port to load the program from. There are two remote ports available. The Model 4000 interprets all of the program statements that you entered on the remote interface. If you enter an invalid statement, an error message will be displayed on the screen and the invalid statement will be sent to the specified port. Select the appropriate port using the numeric pad. Press **ENTER** to allow loading the program. ***If you enter an existing name, that program will be erased before the new program is loaded Only one program may be loaded at a time. If another program is loaded into the Model 4000, you will receive the following message:***

`O.K. to delete current program.`



To exit from this function after you have entered data, enter an asterisk (*) and press the carriage return(↵) over the remote port.

If the program you are loading has comments after the statement, an apostrophe (') or a semi-colon (;) must precede the comment. If your comment takes up more than one line, each subsequent comment line must also have an apostrophe or semi-colon.

To abort the transmission of a load in process, press the **STOP** key on the operator panel.

When entering the **LOAD** function from a remote port (not the operator panel) the **SELECT** function should be used before the **LOAD** function to choose the name of the program. The **SELECT** function takes the place of the program selecting screen when using a remote port. If an existing name is selected, the Model 4000 sends out a tilde (~) when the existing program is deleted & it is ready to receive the new program.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

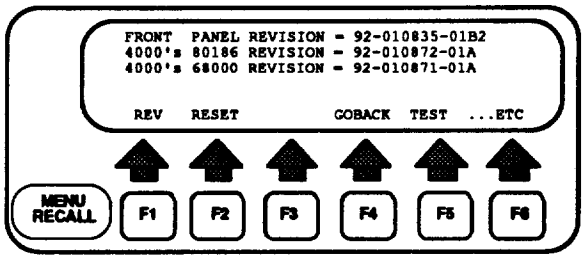
More information on remote printing and loading of programs can be found in Chapter 5, *Application Design*, the section titled Remote Operation.

Press the **MENU RECALL** key to return to the **MAIN** menu. Press **F6** to view the second Main menu screen.

It is not necessary to use the **F5 Load** key when transferring programs using the *Compumotor Application Support Software Disk*.

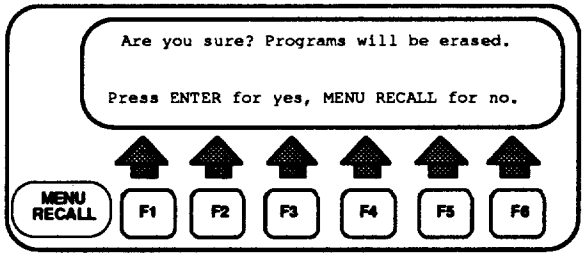
REV Function F1: From Main Menu (2nd Screen)

Press F1 to activate the REV function. The **REV** function displays on the operator panel the software revision of the Model 4000.



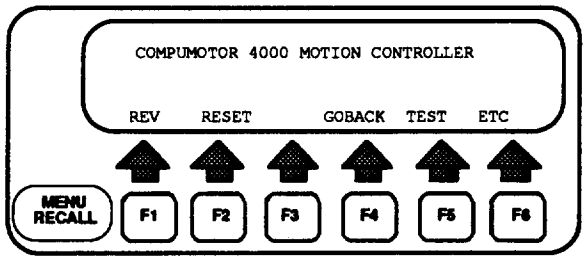
RESET Function F2: From Main Menu (2nd Screen)

Press F2 to enter the RESET function. The **RESET** function erases all memory, including all user programs, and returns program control to the Start-Up menu. The default **POWER-UP** program will be the only program present. All of the user-programmable settings are returned to their default values. Press **MENU RECALL** to return to the Main menu.



GOBACK Function F4: From Main Menu (2nd Screen)

Press F4 to enter the GOBACK function. The **GOBACK** function returns you to the Access menu.



This concludes the tutorial of the Model 4000's menus, functions, subfunctions, and options. You may now begin *Chapter 3, Installation*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

C H A P T E R ③

Installation

Chapter Objectives

The information in this chapter will enable you to:

- ❑ Mount all system components properly
- ❑ Connect all inputs and outputs properly
- ❑ Verify that the complete system is installed properly
- ❑ Perform basic system operations

You must complete the configuration and testing procedures outlined in Chapter 2, Getting Started, that pertain to your application before you perform the steps in this chapter. Be sure that you have performed these steps successfully before proceeding.

Environmental Considerations

The Model 4000 is designed to be mounted inside an enclosure rated for the environment in which it will be used. The Model 4000 is designed to work in temperatures up to 122°F (50°C) and it is fan cooled. In environments that expose the Model 4000 to temperatures above 122°F (50°C), additional cooling is required. An air-conditioned NEMA rated enclosure may be required. For units mounted in enclosures, the temperature of the air inside the enclosure is important, not the air outside. Humidity should be kept below 95%. In high-humidity environments, you may have to provide an air conditioner or de-humidifier.

Electrical Noise

You should minimize the potential for electrical noise before installing the Model 4000 rather than attempting to solve such problems after installation. You can prevent electrical noise by observing the following installation precautions:

- ❑ Do not put motor starters in the same cabinet with the Model 4000.
- ❑ Do not put coil wound relays in the same cabinet with the Model 4000.
- ❑ Do not put high-voltage wires and low-level signals in the same conduit.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- ❑ Do not put electric welders in the immediate area of the Model 4000.
- ❑ Do not put isolation transformers on the inputs of nearby noise producing devices such as chopping motor/drives.
- ❑ Do not put large AC motors (20 HP and up) in the immediate area of the Model 4000.
- ❑ Avoid exposed wiring (unshielded).
- ❑ Ensure that all components are properly grounded.

Airborne Contaminants

Contaminants that may come in contact with the Model 4000 should be carefully controlled. Contaminants such as cutting oil should be kept out of the Model 4000 by putting the Model 4000 in its own enclosure. Particulate contaminants, especially electrically conductive ones (such as metal shavings), can damage the Model 4000.

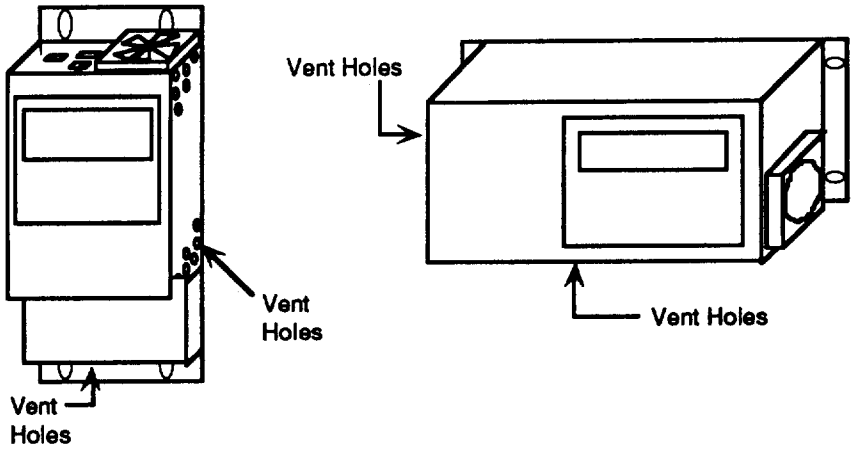
System Mounting

This section provides instructions for mounting the Model 4000 controller and the operator panel. These two components can be mounted in a variety of ways. Read through this section carefully to determine which mounting style is best for your application.

Proper mounting and panel layout are essential for trouble-free operation of the Model 4000. Allow sufficient space for unrestricted air flow over the internal heatsinks. The Model 4000 can be mounted horizontally or vertically. If you use a back-panel configuration, you should mount the Model 4000 vertically. If you use a 19" rack configuration, you should mount the Model 4000 horizontally. The Model 4000 must be given a clearance of at least 4" from the fan and 2" from each side with vent holes to provide adequate air flow. Clearance from the side without vent holes should be at least 1". If you mount the Model 4000 in an enclosure, observe the following guidelines:

① Do not mount large, heat-producing equipment directly beneath the Model 4000.

② Do not mount the Model 4000 directly above a motor/driver.

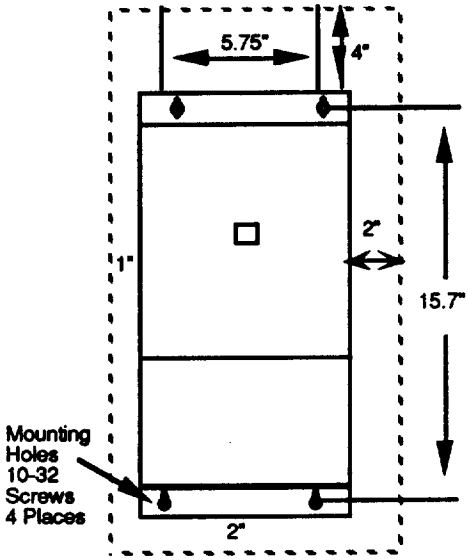


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Panel Mount

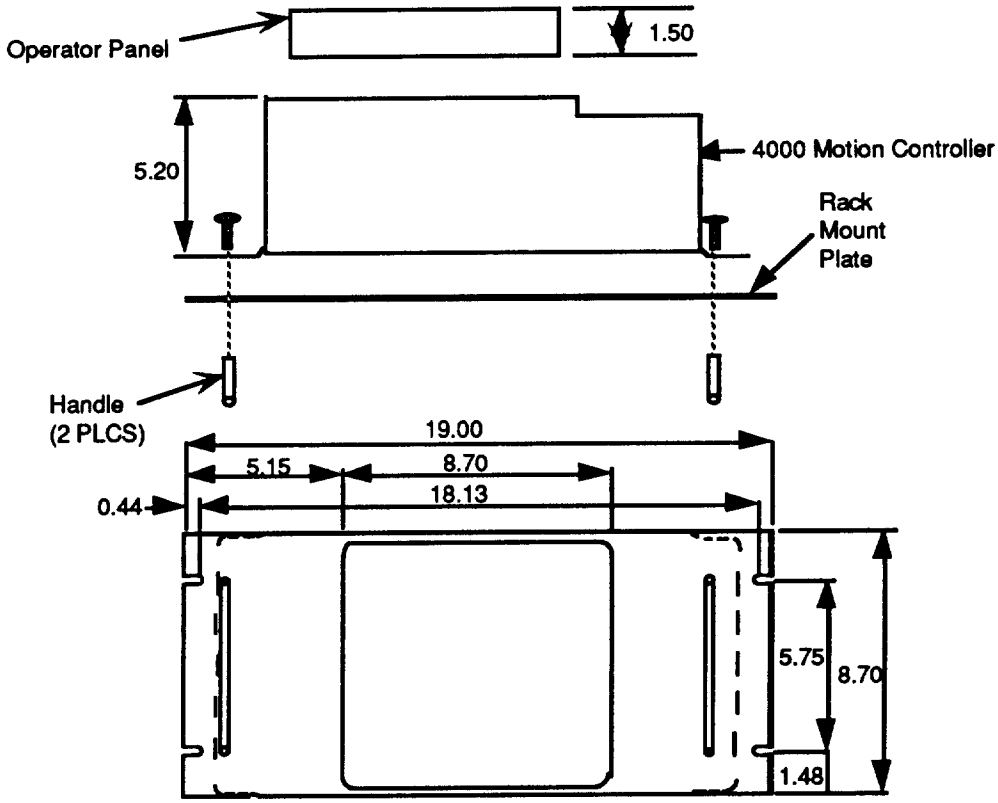
Mount the Model 4000 to the back panel (using 10-32 screws). Leave room on all sides for sufficient airflow.

☛ Never mount the Model 4000 so that vent holes are covered or adequate space for airflow is not provided.



19" Rack Mount

You can use the optional 19" rack mounting kit to install the Model 4000 in a 19" rack (refer to *Chapter 6, Maintenance & Troubleshooting* for part numbers). The kit is pre-drilled for mounting the Model 4000 and the operator panel. The kit includes the mounting plate and hardware for the controller and rack handles.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

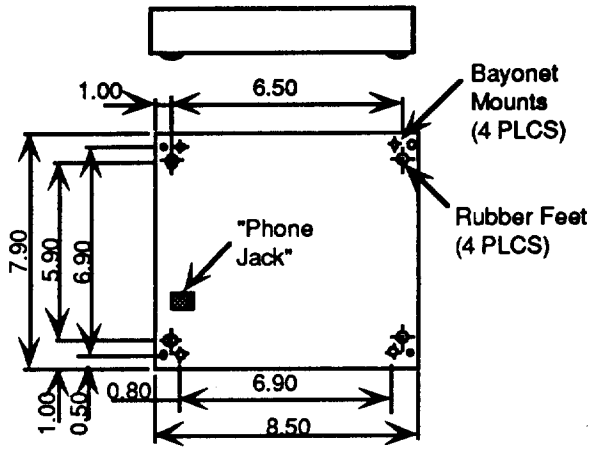
Operator Panel

You can attach the operator panel in a variety of ways.

- ❑ To the front of the Model 4000, where the controller is mounted vertically or horizontally
- ❑ To a door or panel front
- ❑ To the optional 19" rack kit
- ❑ It may be carried by hand
- ❑ Laid flat on a desk or bench
- ❑ To an angled stand for desk or bench top use

Operator Mount

The operator panel may be mounted in two orientations (as shown in the following diagram) on the front of the Model 4000 Controller without modifying the operator panel.



The rubber feet on the back side of the panel provide positive pressure to the mounting panel to prevent it from coming loose. To mount the operator panel to the controller, follow the steps below:

① Connect the 8" phone cord between the Model 4000 and operator panel

② Position the operator panel right side up (as shown in the figure above).

③ Push the 4 bayonet mounts into the four mating holes

④ Push back and then down to fit the operator panel in place.

Door Mount

To mount the operator panel on the door of an enclosure, or on an appropriate panel, follow the steps below:

① Drill five holes in your door as shown in the previous figure.

② Remove the bayonet mounts (screws and standoffs) and rubber feet from the back of the operator panel (Save the screws for mounting).

③ Put one screw through the door.

④ Put the gasket (provided) on the screw.

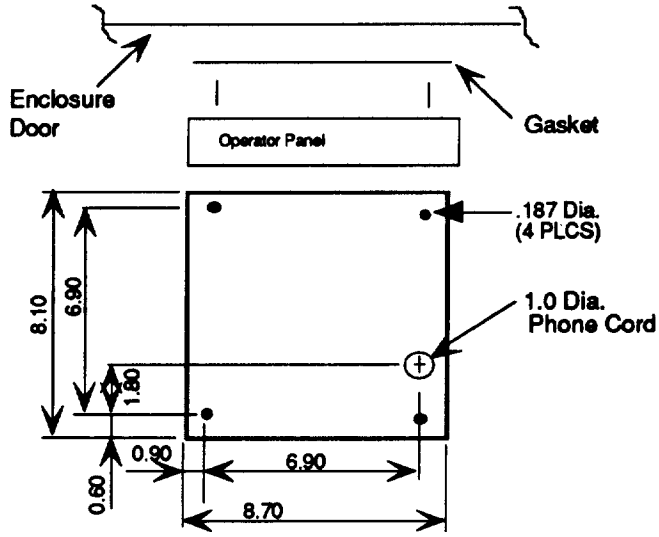
⑤ Put the operator panel onto the screw and loosely tighten.

⑥ Align the panel and gasket and insert the remaining three screws.

⑦ Connect a phone cord between the operator panel and the Model 4000 through the 1.0" diameter hole in the door.

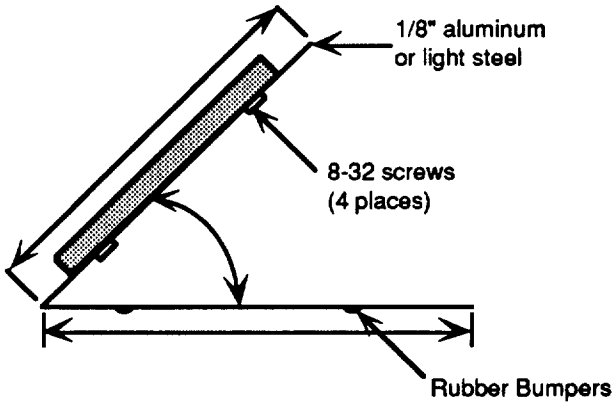
⑧ Do not overtighten the screws.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Desk Mount

A desk mount stand may be fabricated as shown in the following figure. Use the same hole pattern that was shown in the door mount example.



Wiring Guidelines

Proper grounding of electrical equipment is essential to ensure the safety of personnel. You can reduce the effects of electrical noise due to electromagnetic interference (EMI) by proper grounding. All Compumotor equipment should be properly grounded. A good source of information on grounding requirements is the National Electrical Code published by the National Fire Protection Association of Boston, MA.

In general, all components and enclosures must be connected to earth ground through a grounding electrode conductor to provide a low impedance path for ground fault or noise-induced currents. All earth ground connections must be continuous and permanent. Compumotor recommends a single-point grounding setup.

One commonly used method is to prepare components and mounting surfaces prior to installation so that good electrical contact is made between mounting surfaces of equipment and enclosure. Remove any paint from equipment surfaces where the ground contact will be bolted to a panel and use star washers to ensure solid, bare metal contact.

For temporary installation, or when you cannot implement the grounding method described above, you must connect the Ground terminal on the AC power connector to the earth ground.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Operator Panel Connections

☛ **Damage** to the Model 4000 will occur.

The operator panel may be connected to the front or the rear of the Model 4000, with modular phone cable supplied. *Do not connect a telephone company active phone line to the Model 4000. This is not a modem connector.* The connection between the operator panel and the Model 4000 is via four pin (RJ-11) modular jacks. The front and rear jacks are in parallel so either can be used. The cable may be up to 100 feet long, but line drivers should be used to ensure that the low level signals are not distorted due to noise.

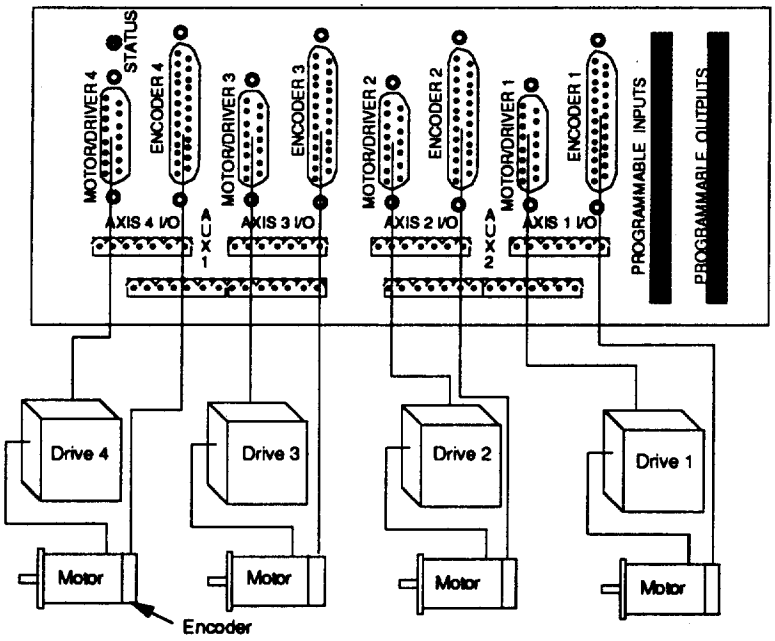
The cable between the Model 4000 and the operator panel contains low level logic signals and therefore should not be routed near high voltage signals such as 110VAC or 220VAC power, motor starter connections, or motor power.

Signal Descriptions

This section defines the electrical specifications of the RS-232C pins.

Power	+12V nominal
Receive (Rx)	RS-232C receive line for communication with front panel
Transmit (Tx)	RS-232C transmit line for communication with front panel
Ground	DC common

Motor/Drive Connections



Motor/Driver Connectors Pinouts

Cables are supplied with the Model 4000 to connect to Compumotor drives. ***Compumotor supplies two controller/drive cables.*** Cable part numbers can be found at the start of Chapter 2, *Getting Started*. ***If you require more than two cables, please contact your local Distributor or ATC.*** To connect to other drives or Compumotor drives with screw terminal connectors, use the information below:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Pin	In/Out	Description	Color
1	OUT	Step (+)	Red
2	OUT	Direction (+)	Green
3	---	Ground	—
4	---	Reserved	Gray
5	IN	Fault	Yellow
6	OUT	CW Step	—
7	OUT	CCW Step	—
8	---	Shield	(Bare Wire)
9	IN	Step Return (-)	Black
10	IN	Direction Return (-)	White
11	OUT	Shutdown (+)	Blue
12	IN	Shutdown Return (-)	Purple
13	---	Ground	Orange
14	---	Ground	Brown
15	---	Ground	—

Signal Description

This section defines the electrical specifications of the Model 4000's motor/driver connectors.

Step, Step Return	(Differential Output) Step (pulse) output to the drive. The step signal is active high.
Direction, Direction Return	(Differential output) A high signal on the Direction Output specifies motor rotation in the CW direction and a low signal specifies motor rotation in the CCW direction.
CW Step	(Open-collector output) For use with non-Compumotor drives. Presence of current indicates CW step (active low). This output can sink a maximum of 250 mA and be connected to up to 24V through a pull-up resistor.
CCW Step	(Open-collector output) For use with non-Compumotor drives. Presence of current indicates CCW step (active low). This output can sink a maximum of 250 mA and be connected to up to 24V through a pull-up resistor.
Shield	Connected to chassis ground (earth ground) within the Model 4000.
Shutdown, Shutdown Return	(Differential output) This signal is used to turn off current in the motor windings. A high signal on shutdown indicates the motor winding current should be off.
Ground	Isolated Logic Ground.
Fault	The fault input is an active high (current not flowing) signal that tells the Model 4000 the drive has faulted. You can use the **DEFINE FAULT** statement to change the active level to low (current flowing) if desired.

Controller to Drive Cable Length

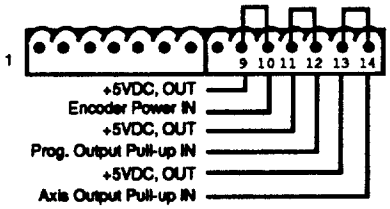
The maximum length of cable between the Model 4000 and motor driver is 50 feet. Cables longer than 10 feet are available from Compumotor. Contact Compumotor's Customer Service Department (800-722-2282) or your local Compumotor Distributor or ATC for assistance.

Encoder Connections

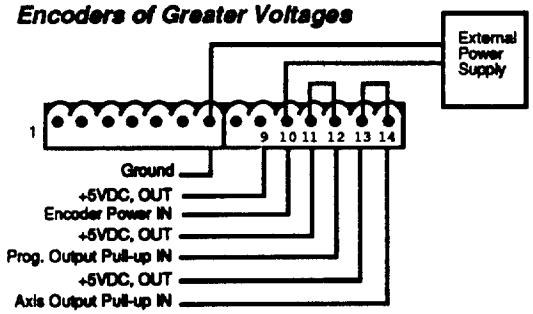
The Model 4000 supports incremental and absolute encoders (Compumotor Model AR-C). The following figure shows the wiring techniques that you must use to incorporate Compumotor and non-Compumotor encoders into your Model 4000 application.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

+5VDC and Compumotor Encoders



Encoders of Greater Voltages



The Model 4000 can accept square wave differential quadrature encoder signals of +5VDC. Each encoder channel return (i.e., A-, B-, Z-) must be connected when used differentially. Leave the returns (A-, B-, Z-) disconnected and isolated when connecting single ended encoder outputs.

Isolated +5VDC power is available on pin #9 of the AUX1 connector. If you are using a Compumotor or other 5VDC powered encoder, jumper from pin #9 to pin #10 (**ENCODER POWER IN**). This connects the internal isolated +5VDC power supply to pin #23 of the encoder connectors.

If your encoder requires power of up to 24VDC, you may connect an external power supply to pin #10 (**ENCODER POWER IN**) and pin #7 (ground), as long as the encoder outputs are still +5VDC TTL active level.

Encoder Pinouts

Each axis has a 25-pin D connector for an encoder connection. The pin-out description for the encoder connectors is provided in the table below.

Pin	In/Out	Description	Encoder Type
1	IN	Phase A	Incremental
2	IN	Phase A return	Incremental
3	IN	Phase B	Incremental
4	IN	Phase B return	Incremental
5	IN	Z Channel	Incremental
6	IN	Z Channel return	Incremental
7	—	NC	
8	OUT	Shield	Both
9	OUT	Data enable low	Absolute
10	OUT	Data request low	Absolute
11	OUT	Address zero	Absolute
12	OUT	Address one	Absolute
13	IN	Data valid low	Absolute
14	—	Ground	Both
15	IN	Data zero	Absolute
16	IN	Data one	Absolute
17	IN	Data two	Absolute
18	IN	Data three	Absolute
19	IN	Data four	Absolute
20	IN	Data five	Absolute
21	IN	Data six	Absolute
22	IN	Data seven	Absolute
23	OUT	V+ Encoder Power Out (AUX 2 connector pin 10)	Incremental
24	OUT	Transmit	Absolute
25	IN	Receive	Absolute

Signal Description

This section defines the electrical specifications of the Model 4000's encoder connectors.

Phase A+, Phase A- Return (Input) Phase A quadrature signal from incremental encoder (+5V TTL compatible).

Phase B+, Phase B- Return (Input) Phase B quadrature signal from incremental encoder (+5V TTL compatible).

Phase Z+, Phase Z- Return (Input) Z channel signal from incremental encoder (+5V TTL compatible).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Shield	Connection to chassis ground (earth ground) within the Model 4000.
Ground	Isolated logic ground
Encoder Power Out	This pin is connected internally to pin #10 of the AUX1 connector. Power can be supplied to the encoder through this pin by connecting the internal 5V (pin #9, AUX1), or an external supply to pin #10, AUX1. If connecting an external supply, be sure to connect the power supply ground to a logic ground on the Model 4000.

Controller to Encoder Cable Length

Compumotor encoders are supplied with a permanently attached cable 10 feet in length.

The maximum cable length between the encoder and the Model 4000 is 25 feet without using external line drivers.

You can order extender cables (up to 15 feet in length) that connect to the standard encoders 10 foot cable. Order part number is 71-003913-XX (XX is the length in feet).

You can also order encoders through Compumotor's Custom Products Group with the exact cable length you want, permanently attached. Contact Compumotor's Customer Service Department (800-722-2282) or your local Compumotor Distributor or ATC for assistance.

Auxiliary 1

The AUX1 connector is a 14-pin screw terminal. The pin-out description for AUX1 is provided in the following table.

Pin	In/Out	Description
1	IN	RS-232C Receive Port 1
2	OUT	RS-232C Transmit Port 1
3	—	Ground
4	—	Shield
5	IN	RS-232C Receive Port 2
6	OUT	RS-232C Transmit Port 2
7	—	Ground
8	OUT	Shield
9	OUT	+5VDC
10	IN	V+ Encoder Power IN
11	OUT	+5VDC
12	IN	Programmable Output Pull-Ups
13	OUT	+5VDC
14	IN	Axis Output Pull-ups

Signal Descriptions

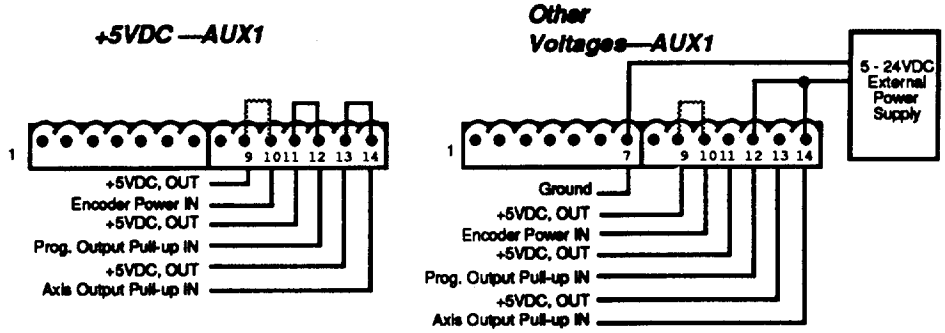
This section defines the electrical specifications of the Model 4000's auxiliary (AUX1 and AUX2) connectors.

Pull-Ups

The programmable output pull-up pin (located on pin #12 of AUX1) and the axis output pull-up pin (located on pin #14 of AUX1) are internally tied to resistor arrays from the outputs.

The pull-up inputs allow you to select the voltage used on the 24 programmable outputs and the 4-axis outputs. Up to 24VDC may be used. If +5VDC is appropriate for the devices connected to the outputs, the Model 4000's internal +5VDC may be used, wiring pin #11 to pin #12 on the AUX1 connector.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



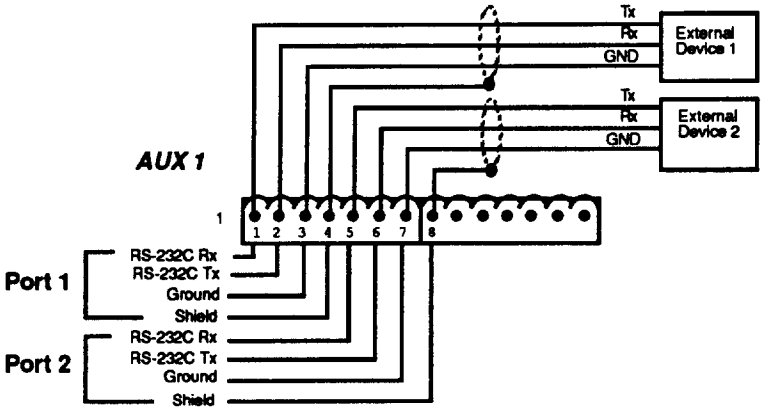
+5VDC Out

The Model 4000 provides three +5VDC outputs that are located on pins #9, #11, and #13 of AUX1. As much as 4A is available for you from the 5V supply. The 4A limit applies to the total load on all of the I/O connectors. For example, if the encoder power pin is jumpered to +5VDC and the encoders use a total of 2A, then 2A is left for other uses.

RS-232C Communication

Port 1 and Port 2 are three-wire implementations of RS-232C. You can use them to program the Model 4000, to communicate with other Model 4000s, to control Compumotor Indexers, to communicate with PLCs and computers, or to send messages to message displays. XON/XOFF software handshaking protocol is supported. No hardware handshaking is supported. Refer to the **RS232C** statement description in *Chapter 4, Software Reference*, for RS-232 port set-up information. The default settings are:

- ❑ 9,600 baud
- ❑ 8 data bits
- ❑ 1 stop bit
- ❑ no parity



Auxiliary 2

The AUX 2 connector is a 14-pin screw terminal. The pin-out description for AUX 2 is provided in the table below.

Pin	In/Out	Description
1	OUT	+5VDC
2	IN	Analog 1
3	IN	Analog 2
4	IN	Analog 3
5	IN	Analog 4
6	—	Ground
7	—	Shield
8	IN	Joystick Release
9	IN	Joystick Select
10	IN	Test
11	—	Reserved
12	IN	Estop
13	IN	Joystick Trigger
14	IN	Joystick Velocity Range

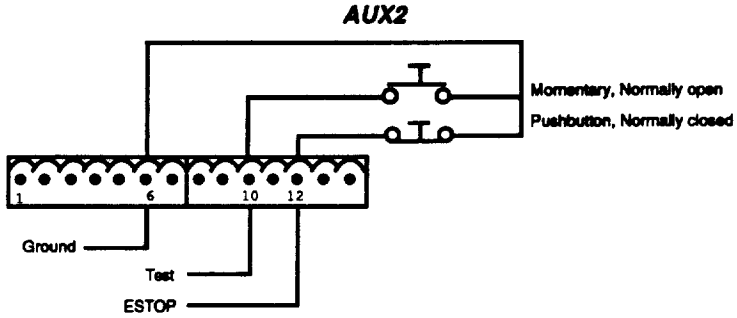
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Test Input

The test input is located on pin #10 of AUX2. This TTL-compatible input allows you to test the motor motion (it is active low). It verifies the basic functionality of the Model 4000 and any attached motor/drivers. If you ground the test input prior to power-up, the motors will rotate at 5,000 steps per second for 25,000 steps in the each direction (CW and CCW). , The Model 4000 will go into *Board Monitor Alarm Failure* and the status light will turn red. On a motor/driver with 25,000 steps per revolution, this failure causes the motor to turn one revolution in five seconds. All motors turn at once. *End-of-travel limit inputs are ignored during the test.*

The unit must be powered down and the test input ungrounded to resume normal operation.

The test input should normally be left ungrounded. The ESTOP must be grounded to run this test. If it is not grounded, the motors will not move.



ESTOP Input

The ESTOP input is located on pin #12 of AUX2. A loss of connection to ground on this input will shut off step pulses and open fault relay contacts.

The ESTOP input (a TTL-compatible input) stops the pulse output instantly while allowing the Model 4000's program to continue to run. This input must be grounded to allow motor motion. If it is not grounded, the motors will not move. Your programs may be written to take appropriate action upon receiving an ESTOP command (**ON ESTOP** statement), such as giving a message to the operator or stopping the program. Motor position may be lost when the ESTOP input is activated (encoder position will still be accurate).

☛ This input does not stop the Model 4000 program. An ON ESTOP subroutine must be written to handle such a condition.

If the ESTOP input becomes active, the fault output becomes active. Refer to the Fault Output description in *Chapter 6, Hardware Reference* for more information.

The ESTOP input does not shut power off to the Model 4000, however, the Model 4000 does provide a fault output relay that you can wire to a system power-off relay device.

Joystick and Analog Inputs

You can use the four analog inputs on the AUX2 connector for joystick control or feedrate override of the axes. Compumotor does not supply joysticks, but Daedal's Model JS7000 is compatible with the Model 4000.

The input range of the analog input is 0V to 2.5V. *A joystick with linear taper 5K potentiometer with 60° of travel is recommended* ***(the potentiometer has 300° of travel, but only 60° is usable with the joystick).*** The potentiometer should be adjusted so that its resistance is close to 0 ohms when the joystick is all the way to one side and about 1K ohms when the joystick is all the way to the other side. Also, connect a 1K resistor between the analog input and +5V.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Joystick Select Input

You can define two modes (A and B) that define which axes are controlled by which channels. The joystick select input allows you to select the current mode. **A joystick select input high is referred to in this chapter as *Mode A*. A joystick select input low is referred to in this chapter as *Mode B*.** This allows a user who has one joystick to control four axes. Using the select input, you could alternately control axes 1 and 2 (Mode A) and then axes 3 and 4 (Mode B). The axes which correspond to each channel are set with the **DEFINE JOYSL, JOYSH** statements. This arrangement is illustrated in the following figure. If the application requires two dedicated joysticks, the select input is not needed.

Joystick Release Input

The joystick release input allows the user to indicate to the Model 4000 that they have finished using the joystick and program execution may continue with the next statement. When a program enables joystick control of motion, program execution will stop and then resume when the user is finished with joystick mode.

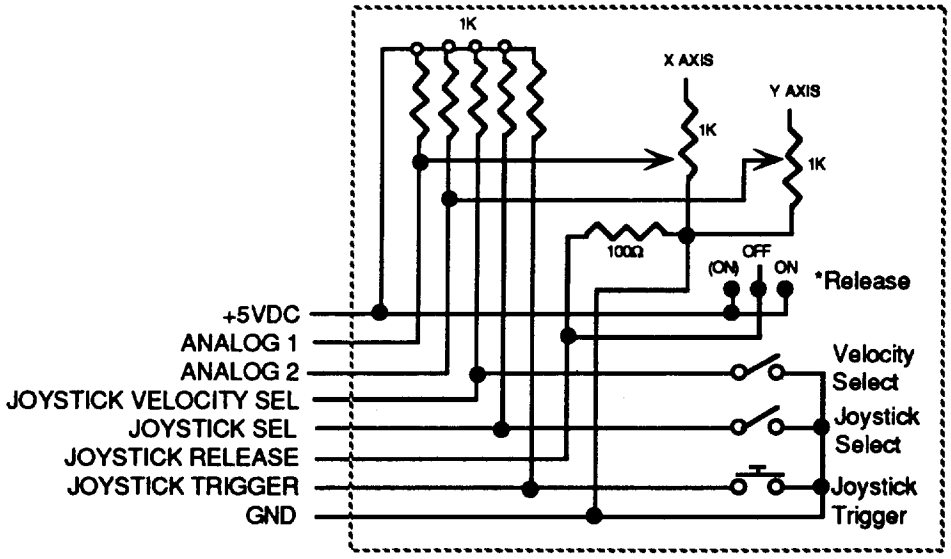
When joystick mode is enabled, 0V at this input will keep the axes in joystick mode, and 5V will release the joystick. This input has an internal pull-up resistor to +5V. Therefore, when a joystick is not connected, joystick enable statements will be finished immediately.

Joystick Velocity Input

This input may be used to select either the high (high on input) or low (low on input) velocity range as defined in software (**VEL JOYL, JOYH** statements). The high range could be used to quickly move to a location while the low range could be used for accurate positioning.

Joystick Trigger Input

The status of this input can be read by a program and may be used to control program flow. A program may be written that would allow the trigger input (active low) to enter the user into joystick mode. The following figure is an example of joystick wiring.



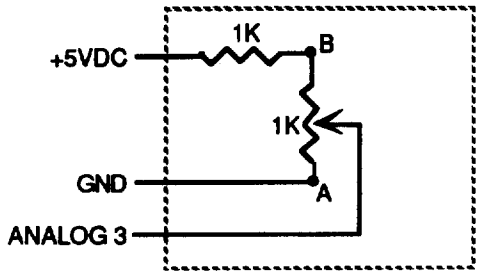
Joystick potentiometers are 5K with 60° of usable travel adjusted to span 0Ω to 1KΩ

* The release switch is a toggle switch with the following positions: Momentary ON - Center OFF - ON

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Analog; Inputs

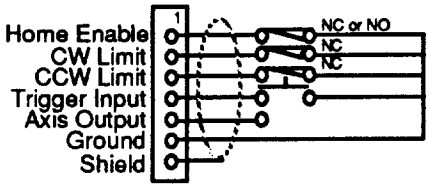
You can use the analog inputs for feedrate control of all four axes, or for joystick control of the axes. When used for feedrate control, an analog input may be used to control the overall velocity of the four axes from 0% to 100%. When used for joystick control of axes, an analog input can command an axis velocity from full CW to full CCW. The analog input voltage can also be read directly for general purpose use. The range of the analog input is 2.5 VDC, not to exceed 3.0 VDC.



0V at point A
2.5V at point B

Axis I/O

Axis Connectors



(Typical per axis)

CW and CCW limit must be grounded through a limit switch or disabled in software to allow motor motion.

Axis I/O Pinouts

The Model 4000 has four Axis I/O connectors, (7-Pin Screw Terminal) one per each axis.

Pin	In/Out	Description
1	IN	Home Enable
2	IN	CW limit
3	IN	CCW limit
4	IN	Trigger
5	OUT	Axis Output
6	-	Ground
7	-	Shield

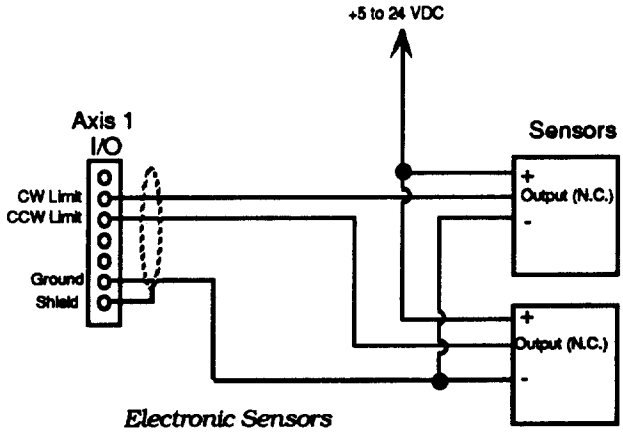
Signal Description

This section defines the electrical specifications of the Model 4000's Axis I/O connectors.

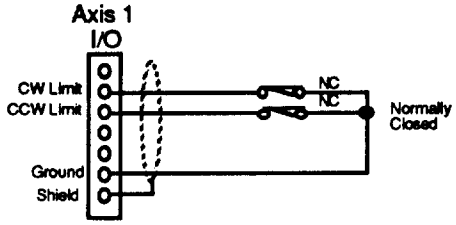
Home Enable	A high (open) or low (closed) indicates the motor is in the home position (may be used in conjunction with the encoder Z channel). This is a TTL-compatible input. The active level of the home enable input can be changed via software.
CW Limit	A high (open) indicates the CW limit has been encountered. This is a TTL-compatible input.
CCW Limit	A high (open) indicates the CCW limit has been encountered. This is a TTL-compatible input.
Trigger	User trigger input. This is a TTL-compatible input.
Axis output	User controllable output bit. This is an open-collector output pulled up to supply voltage on Auxiliary 2.
Ground	Isolated Logic Ground
Shield	Connected to chassis (earth ground)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

End-Of-Travel Limit Wiring



Electronic Sensors



Mechanical limit switches

End-of-travel limits must be grounded to allow motor motion for each axis. Limits may be bypassed in the control program if desired (see the **ENABLE** statement in the *Chapter 4, Software Reference* for more information).

You must use a normally-closed switch for the limit switch connection. The circuit should open if the load reaches the mechanical limits established for the application. Allow room for the motor and load to slow to a stop.

Sensors that require +5VDC may be powered from the AUX1 connector. Sensors that require more than +5VDC require an external power supply. The Model 4000's inputs will accept up to 24VDC.

Home Enable, Triggers, & Axis Outputs

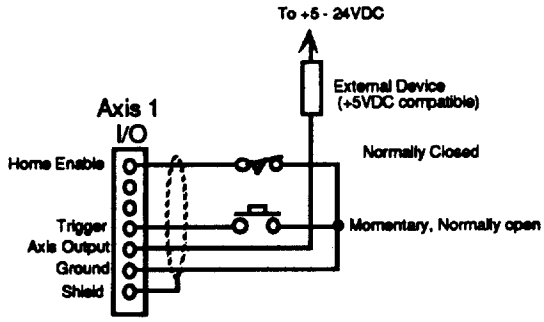
You may configure the home enable inputs under program control to be active high or low. When configured as active high, the home enable inputs should be grounded until activated (like end-of-travel limits). (Use a normally-closed switch.) When configured as active low, the home enable inputs should be open until activated. (Use a normally-open switch.) Place the home enable sensor or switch so that it activates when the load is in the home range. Do not use the output of a PLC to provide this signal. The time introduced by passing the switch input through a PLC may cause inaccurate positioning. The home enable input can be used as the home position sensor on its own or in conjunction with the Z channel marker pulse from the encoder. When used in conjunction with the Z channel, both the Z channel input and the home enable input must be on to be at home. This is useful when using a rotary encoder with a once per revolution Z channel signal in a multiple revolution application.

Trigger inputs can be connected to PLC outputs, discrete switches or electronic sensors. Triggers are monitored under program control.

The axis output is an open collector output with an internal pull-up resistor connected to the AUX1 connector, pin #14. You may use the isolated +5VDC available on the AUX1 connector, pin #13, or supply your own pull-up voltage, up to 24VDC.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Sensors that require +5VDC may be powered from the AUX2 connector, pin #1. Sensors that require more than +5VDC require an external power supply. The inputs will accept up to 24VDC.

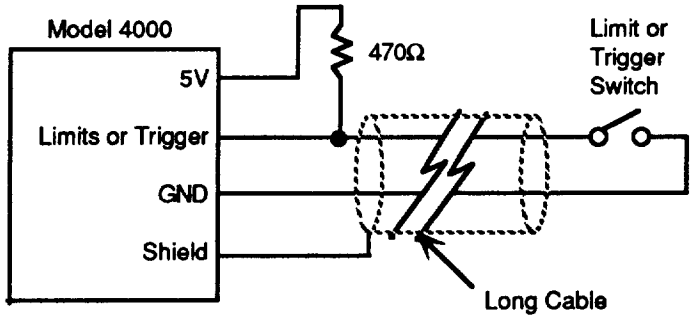


How to Extend Axis I/O Cables

To avoid interference from external noise, you must shield all I/O cables, regardless of the length. The maximum length of cables is determined by the environment in which the equipment will be used. For cables longer than 50' or in electrically noisy environments you should follow the guidelines below.

Recommended guidelines:

- 22 AWG wire is the minimum recommended wire size.
- Use twisted pair shielded cables and connect the shield to pin 7. Leave the other end of the shield disconnected.
- Do not route these signals in the same conduit or wiring trays as high voltage AC wiring.
- The axis inputs have internal pull-ups to +5VDC and are TTL compatible. They can also withstand up to 24V. In electrically noisy environments or when using long cable lengths, an external pull-up resistor with a value of 330Ω to 2.2KΩ between the input and +5V to 24V is recommended. The external resistor will lower the input impedance and will make the input less susceptible to electrical noise.



Fault Output Connections

The Fault outputs are contacts of a mechanical relay that is energized when the Model 4000 is working normally. Both normally-open and normally-closed contacts are provided. The relay becomes de-energized when:

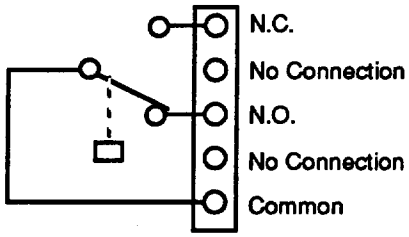
- The Model 4000 loses power.
- The Model 4000 detects an internal fault (turns on the red status LED).
- The ESTOP input is activated.

Any device connected to this output must supply at least 200mW to prevent contact oxidation.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

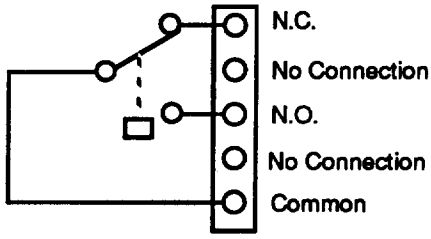
Fault Output Pin Out

Pin	IN/OUT	Description
1	OUT	Normally closed contact
2		No connection
3	OUT	Normally open contact
4		No connection
5	--	Common to pin 1 and 3



4000 Powered and NO Faults

(relay energized)



4000 Power OFF, Power ON and 4000 Faulted, or ESTOP asserted

(relay de-energized)

Programmable I/O Connections

The two programmable I/O connectors provide 24 inputs and 24 outputs, respectively. These inputs and outputs are optically isolated and the outputs may be pulled up using the programmable output pull-up on pin 12 of the AUX1 connector. If +5VDC logic is to be used, connect the pull-up input to pin 11, +5VDC. The inputs may be pulled up to 24VDC by using an external power supply.

The I/O are designed to be used with OPTO-22 I/O module racks such as the OPTO-22 model PB24 I/O racks or compatible equipment.

These options can be purchased from your local OPTO-22 distributor, ATC, or the OPTO-22 factory: 714-695-9299

Input Modules		Module Mounting Racks	
IAC5	90-140 VAC	PB4H	4 position
IAC5-A	180-280 VAC	PB8	8 position
IDC5	10-32 VDC	PB16	16 position
IDC5-B	4-16 VDC	PB24	24 position
Output Modules		**4000 to Mounting Rack Cable**	
OAC5	12-140 VAC	OD-2	2 foot
OAC5A	24-280 VAC	OD-4	4 foot
ODC5	2-60 VDC	OD-8	8 foot
ODC5A	2-200VDC	OD-10	10 foot

The Model 4000 is also compatible with OPTO-22's *Generation 4 I/O System*. Part numbers are the same as those listed above with the addition of *G4*, e.g., *G4PB16A*, *G4IAC5*, etc.

☛ Selecting and Installing OPTO22 Equipment

① Determine how many inputs and outputs you will need.

② Select one mounting rack for inputs and one for outputs that will accommodate the number of inputs and leave room for expansion

③ Select an input or output module of the appropriate voltage for each input or output (AC or DC).

④ Select one cable for each module mounting rack, which can be mounted up to 10 feet from the Model 4000. These 50 pin flat cables plug into the Model 4000 on one end and the mounting rack on the other. Compumotor does not provide these cables, but they can be purchased through OPTO22.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

⑤ There is a jumper pad on the OPTO-22 module mounting rack that connects +V to pin #49. This should be installed to operate the OPTO-22 board from the Model 4000. If the jumper pad is not installed on pin #49 +V you must use an external 5V supply.

⑥ Wire your inputs and outputs per the OPTO-22 instructions.

⑦ Connect the programmable output pull-ups (pin #12, AUX1) to +5VDC (pin #11, AUX1). If referencing the outputs at a voltage other than 5V, this reference voltage should be connected to pin #12 on the AUX1 connector.

☛Direct Connections

If you choose not to use OPTO22 I/O racks and modules for use with the 24 programmable I/O, you may want to purchase a Compumotor VM50. The VM50 breaks out the 50 pin header connection into easy to use screw terminals. The VM50 also comes with a 2 foot cable to connect directly to the Model 4000. Contact your local ATC to order the VM50.

Programmable I/O Pin Outs

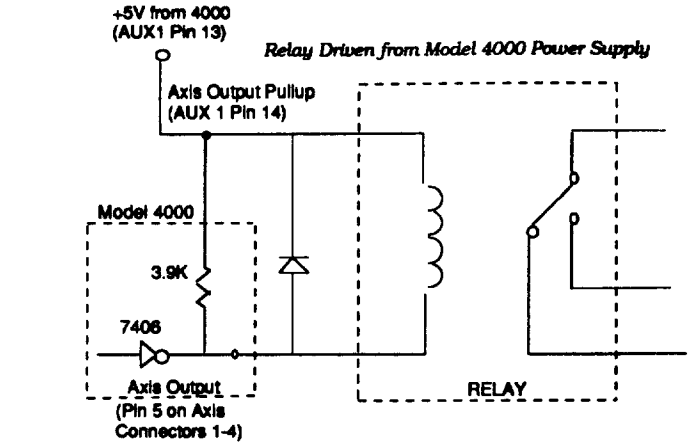
There are two 50-pin flat cable headers on the Model 4000. The following table contains the pin out descriptions for these programmable inputs and outputs.

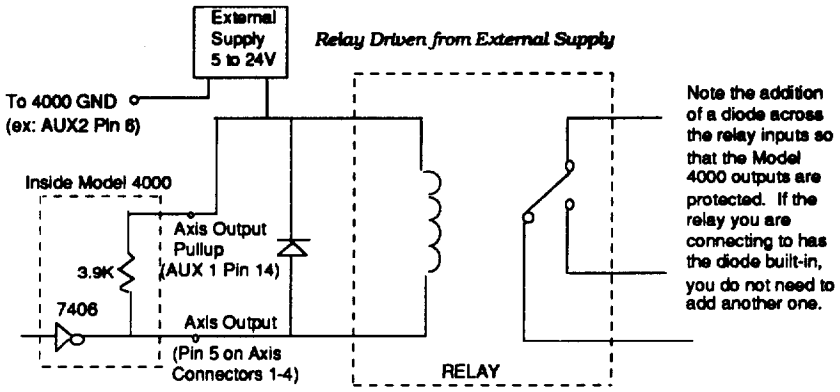
Pin No.	Output Connector	Input Connector
49	+5 VDC	+5 VDC
47	Output #1 (LSB)	Input #1 (LSB)
45	Output #2	Input #2
43	Output #3	Input #3
41	Output #4	Input #4
39	Output #5	Input #5
37	Output #6	Input #6
35	Output #7	Input #7
33	Output #8	Input #8
31	Output #9	Input #9
29	Output #10	Input #10
27	Output #11	Input #11
25	Output #12	Input #12
23	Output #13	Input #13
21	Output #14	Input #14
19	Output #15	Input #15
17	Output #16	Input #16
15	Output #17	Input #17
13	Output #18	Input #18
11	Output #19	Input #19
09	Output #20	Input #20
07	Output #21	Input #21
05	Output #22	Input #22
03	Output #23	Input #23
01	Output #24 (MSB)	Input #24 (MSB)

All even-numbered pins are connected to logic ground (DC ground).

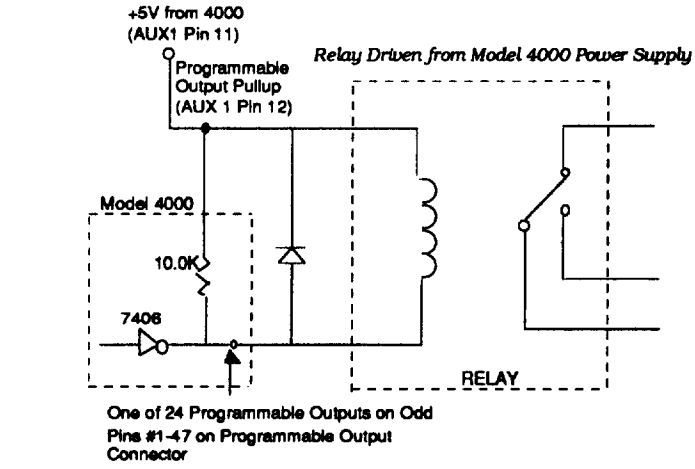
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

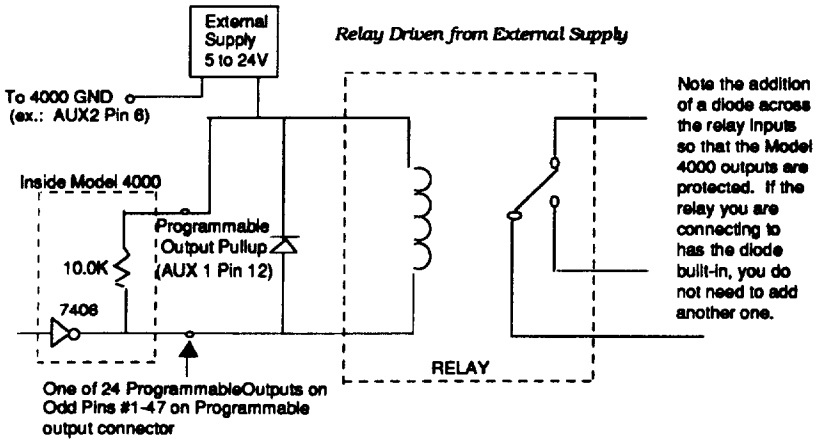
☛ Driving a relay with one of the 4 Axis outputs



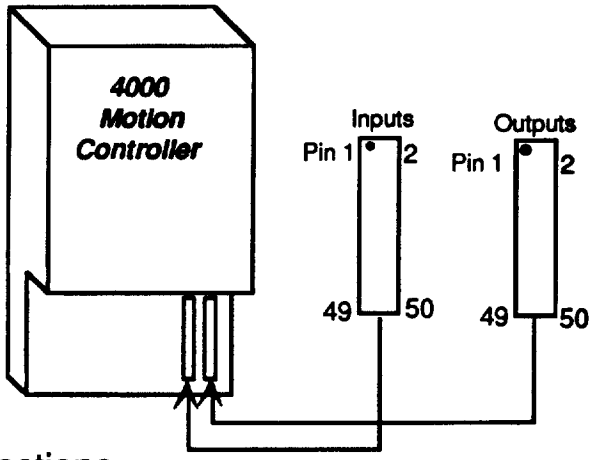


☛ Driving a relay with one of the 24 Programmable Outputs





Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Input Power Connections

Input power is provided by a molded three wire EIA receptacle. A 6-foot power cable is provided to plug into a grounded 120 VAC outlet. The power input accepts 95VAC - 240VAC (single phase) without any configuration change. If you cannot use the molded plug in your application, cut the three pronged plug off one end of the provided cable. Leave the other end intact to plug into the Model 4000 molded receptacle.

The earth ground connection must be made by plugging into a grounded AC receptacle or by physically connecting the green wire to earth ground.

Color	Signal
Black	AC Line
White	AC Neutral
Green	Earth GND*

PLC Operation

This section discusses PLC operation with the Model 4000 via discrete I/O.

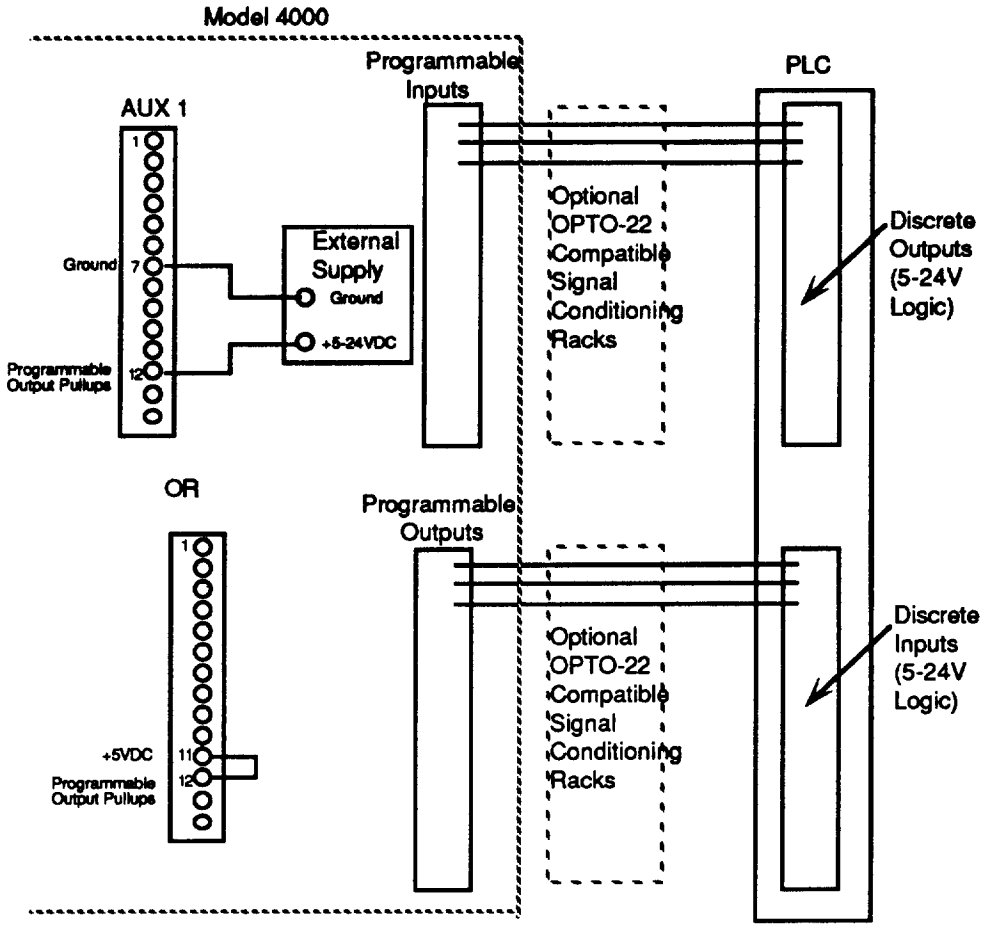
Discrete I/O

The Model 4000 may be connected to most PLCs with discrete inputs and outputs. The programmable I/O and axis I/O may be connected directly to the PLC's inputs and outputs. The PLC should be able to sink at least 1mA of current on its outputs. For +5VDC operation the programmable I/O may be pulled up to +5VDC using the programmable output pull-up on pin #12 of the AUX1 connector.

For voltages up to 24VDC an external power supply may be used to bias the outputs (see the section on programmable I/O for more information).

For voltages above 24VDC you must use external signal conditioning such as OPTO-22 compatible I/O signal conditioning racks. Contact your local Compumotor distributor for information on these products.

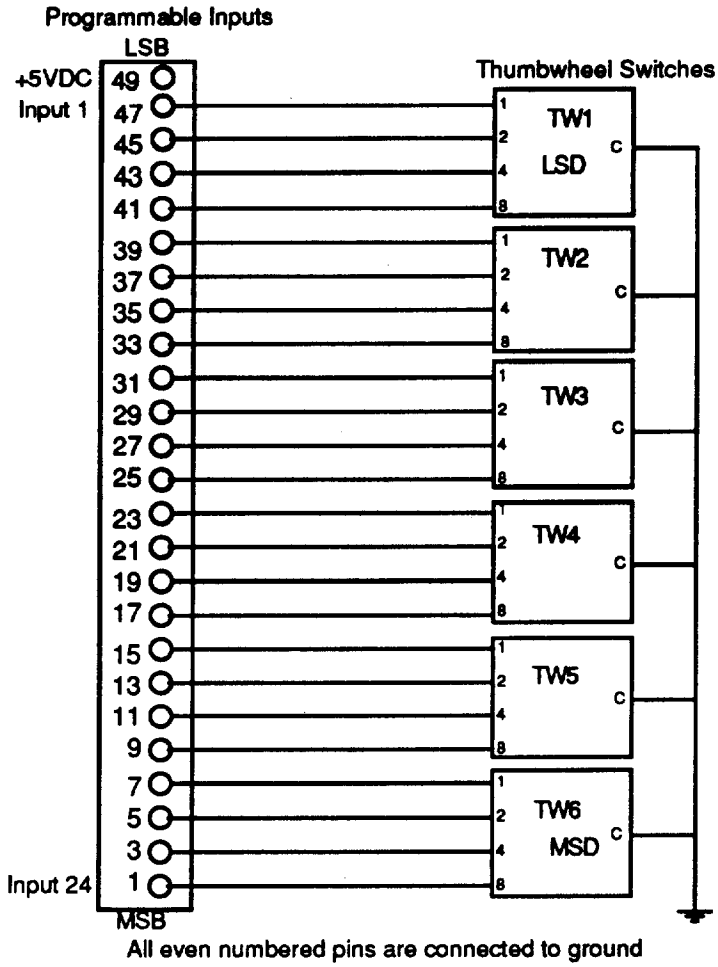
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Thumbwheel Input Operation

You may connect the Model 4000 to remote thumbwheel switches to allow operator selection of motion or machine control parameters. You can connect the Model 4000's programmable inputs to a bank of thumbwheel switches. The Model 4000's programming language allows direct input of BCD data (i.e., thumbwheels) via the 24 programmable inputs with the **`IN IN24`** statement.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



System Verification

This section discusses the following topics:

- LEDs and Faults
- Minimal Safety Verifications
- Functional Test Routines

Model 4000 LEDs

After you have completed the installation procedures described in *Chapter 2, Getting Started*, and the inputs and outputs have been wired to the correct devices, power may be applied to the Model 4000. When power is applied, the **STATUS LED** on the Model 4000 should be green. If the **LED** is off, the Model 4000 is not receiving power. If the **LED** is red, the microprocessors inside are not operating correctly, or the **TEST** input had been active.

The operator panel **STATUS LED** should be green after power is applied. A red **STATUS LED** indicates damage to the operator panel. For each axis, there is an **ACTIVE LED** and a **FAULT LED**. The **ACTIVE LED** should be off unless there is a motor moving. The **FAULT LED** should all be off unless one of the following conditions exist:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

① An end-of-travel limit switch is active.

② The **ESTOP** input is not grounded.

③ The axis has been commanded to disable the drive via the **SHTDN DRIVE** statement or the remote Shutdown (**ST**) command .

④ Software limit positions have been exceeded.

⑤ The motor has stalled (stall detect enabled).

Minimal Safety Verifications

To insure that the safety features of the Model 4000 operate properly, test the installation configuration of the following inputs:

① **ESTOP**: If there is a system **ESTOP** button or switch wired to the Model 4000, pressing (opening) the button should cause the following results.

- ❑ The **FAULT LEDs** on all axes will turn red.
- ❑ The **FAULT** relay on the Model 4000 will de-energize.

② **END-OF-TRAVEL LIMITS**: If limits are to be used in the system, it is not adequate to rely on a manual test of the limit switches. Using the jog function of the Model 4000, cautiously move the load to the end-of-travel limit. If the limit is properly installed, motion will stop and the axis Fault LED will turn red when the limit is encountered. This procedure verifies that the switch is aligned properly and that the CW limit is asserted when traveling CW and vice versa.

The remainder of the installation can be verified using the test routines described in *Chapter 2, Getting Started*, and reviewed below.

Functional Test Routines

The functional test routines (available from the main menu of the front panel) are language routines provided in the Model 4000 that you can use to test the Model 4000's basic components. The following tests are available:

- ❑ Move an axis.
  - ○ Move Axis 1
  - ○ Move Axis 2
  - ○ Move Axis 3
  - ○ Move Axis 4
- ❑ Turn on Outputs 1-24 one at a time.
- ❑ Read the 24 inputs and display results.
- ❑ Display state of the axis inputs
- ❑ Turn axis outputs on and off.
- ❑ Display any message received on either RS-232C port.
- ❑ Display the state of the limit switch inputs.
- ❑ Display the state of the Trigger and Estop inputs.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

C H A P T E R ④

Software Reference

Chapter Objectives

The information in this chapter will enable you to:

- ❑ Identify the four types of Model 4000 statements (set-up, programming, motion, and status)
- ❑ Understand the Model 4000's functions
- ❑ Understand the Model 4000's remote language statements
- ❑ Use this chapter as a reference for the function, range, default, and sample use of each statement and function

Model 4000 Command Language

This section provides a detailed description of all Model 4000 statements.

Command Statement Description

A parent and daughter approach to programming is used in the Model 4000. A parent may have many daughter statements. A daughter statement may be considered a substatement of the parent.

Program statements make up the Model 4000's motion control language. Valid statements begin with a line number, followed by a statement identifier, followed by various parameters that are appropriate for a particular statement. In the following alphabetical statement listing, statements are described using a box shown below.

①	**Name**	SAMPLE
②	**Descriptor**	Sample
③	**Type**	Motion
④	**Default**	
⑤	**Syntax**	
⑥	**Options**	TAB Q NULL JOG HOME ETC
⑦		F1 F2 F3 F4 F5 F6

① This is the statement identifier. This is the key word(s) that is used to identify the statement. It is an abbreviated form of the command name (mnemonic).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

② This is the statement descriptor. This is the long form of the statement identifier. This information describes the purpose of the statement.

③ This is the statement type. There are four types of Model 4000 statements

- *Motion*: Motion statements affect motor motion. For example, acceleration, velocity, distance, go home, stop, direction, mode, etc.
- *Status*: Status statements respond (report back) with information.
- *Set-Up*: Set-Up statements define setup conditions for the application.
- *Programming:* Programming commands affect program flow. For example, `OUT`, `IF`, `GOSUB`, `GOTO`, `ON`, `LOOP`, and `ENDLOOP`.

④ This field contains the default statement. This is the statement as it will appear on the Model 4000's operator panel (when you enter the statement from the Edit or Insert menus). *The default cursor location is shown with the appropriate character underlined.*

⑤ This field contains the statement syntax. Syntax is a description of the parameters required to construct a valid statement for the selected identifier. This information is most useful in writing statements over the remote interface.

⑥ This field contains the available options associated with the statement. Options are the additional parameters that you may use to construct the statement. The options appear on the bottom line of the operator panel in the Edit and Insert modes. If more than one line of options are available, **Etc** will be included above the **F6** key. Select **Etc** to display the additional options.

⑦ This box contains a representation of function keys **F1 - F6**. This illustrates the location of items in the operator panel display (with respect to the function keys).

Parent and Daughter Statements

Daughter or secondary statements are variations of a parent statement. This statement identifier uses the primary statement's keyword and an additional keyword that describes its specific function.

PARENT STATEMENT EXAMPLE

This is an example of a Parent or primary statement, They have a double underline.

DAUGHTER STATEMENT EXAMPLE

This is an example of a Daughter statement, they have a single thick underline.

Programming Notes

If you enter a numeric value that is larger than the Model 4000's front panel can display, but is small enough for the unit to store internally, ***small x's*** will be displayed in the field instead of the number you entered. ***This is not an error condition***

The largest `Q1 - Qnn` variable that you can enter (or a constant that has the same limitations as a `Q` variable) is **879,609,302.2207**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Functional Statement Grouping

Basic Point-to-Point Motion
- ENABLE HLIMIT
- MODE M_INC/M_ABS
- UNIT POS
- UNIT VEL
- UNIT ACCEL
- VEL
- ACCEL
- DECEL
- MOVE
- MOVI

Homing
- DEFINE HOME TYPE
- DEFINE HOME EDGE
- DEFINE HOME FDIR
- DEFINE HOME BACK
- ACCEL HOME
- VEL HOME
- MOVE HOMECW/HOMECCW
- MOVI HOMECW/HOMECCW

Encoder Mode
- ENCO MRES
- ENCO ERES
- MODE E_INC/E_ABS
- ENCO BKL
- ENABLE STALL
- ENABLE MOVKIL
- ON STALL
- ENCO DBND
- POSM GAIN
- POSM MAXVEL
- ENABLE POSM
- ENABLE NDBWAIT
- ENCO ABS
- ON EABSLIM

Segmented Move
- VEL
- ACCEL
- DECEL
- SEG BEG
- SEG MID
- SEG END
- MOVE SEGCW/SEGCCW
- MOVI SEGCW/SEGCCW

Jog Mode
- ACCEL JOG
- DECEL JOG
- VEL JOGL
- VEL JOGH
- DEFINE ARROW
- ENABLE JOG
- ENABLE TCH_JOG

Linear Interpolation
- UNIT PATH VEL
- UNIT PATH ACCEL
- VEL PATH
- ACCEL PATH
- DECEL PATH
- LINT MODE
- MOVI

Registration Move
- VEL
- ACCEL
- DECEL
- SEG REGn
- ENABLE REGn
- MOVE
- MOVI
- ENABLE TRIG REV
- ENABLE REG SRV
- DEFINE TRIGDB

RS-232 Interaction
- RS232
- DEFINE TERM CHAR
- ENABLE PORTn
- OUT PORTn
- IN PORTn

Front Panel
- DISPLAY
- ENABLE PORTn
- IF F-KEY
- IF ENTERK
- IF STARTK
- IF MENUREC
- IF ARROWR
- IF ARROWL
- IF ARROWU
- IF ARROWD
- IN LCD
- ON MENUREC
- ON STOPK
- ON STARTK
- ON F_KEY
- OUT LCD
- WAIT FOR F-KEY

Axis Status/Error Handling
- IN STATUS
- ON ESTOP
- ON ERROR
- IN ERROR
- ON FAULT
- ON STALL
- ON EABSLIM
- ON M_INC
- ON M_ABS
- ON E_INC
- ON E_ABS
- ON TIME
- SHTDN

Program Flow/Conditional
- ENABLE ON
- LOOP
- ENDLOOP
- GOSUB
- GOTO
- IF
- ON
- WAIT
- BREAK0
- DEFINE ON RET

24 Programmable I/O
- DEFINE IN24
- DEFINE OUT24
- IF IN24
- IF BIT
- IN IN24
- IN BIT
- ON IN24
- ON BIT
- OUT OUT24
- OUT BIT
- WAIT FOR IN24
- WAIT FOR BIT

Axis I/O
- IF TRIG
- IN TRIG
- ON TRIG
- OUT POB
- WAIT FOR TRIG

Velocity Scaling/Feedrate Override
- ACCEL VSCALE
- DEFINE VSCL_L
- DEFINE VSCL_H
- DEFINE END_DB
- VEL
- ENABLE VSCALE

Joystick Mode
- ACCEL JOY
- DECEL JOY
- DEFINE JOYSL
- DEFINE JOYSH
- DEFINE CENTER
- DEFINE CTR_DB
- DEFINE END_DB
- VEL JOYL
- VEL JOYH
- ENABLE JOY
- ENABLE TCH_JOY

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ACCEL

Name	**ACCEL**
Descriptor	Set Acceleration
Type	Motion
Default	ACCEL * * * *
Syntax	ACCEL 250000 200000 * *

Options						
	TAB	Q	NULL	JOG	HOME	ETC
	TAB	PATH	JOY	VSCALE		ETC
	F1	F2	F3	F4	F5	F6

Description

The **ACCEL** statement specifies the linear acceleration ramps for each axis. The entries may be a number, a variable, or the word NULL (*) , in which case the previously programmed value is used.

Any value you enter will be multiplied by the scale factor entered with the **UNIT ACCEL** statement. An exception to this rule is **ACCEL VSCALE**. It is always expressed as a %/sec, and is not scaled by **UNIT ACCEL**.

The default acceleration rate is 2,500,000 steps/sec^2 unless changed with this statement. This default value corresponds to 100 rps^2 for a standard 25,000-step/rev Compumotor motor.

The **ACCEL** value can range from 0.0001 to 879609302.2207. However, after the **ACCEL** value is multiplied by the **UNIT ACCEL** value, the result must not exceed 999 rps^2 (which corresponds to 24,975,000 steps/sec^2 for a 25,000 step/rev motor). If the value exceeds 999 rps^2, the Model 4000 will ignore it. For **JOY** and **JOG** statements, the minimum value for both acceleration and deceleration corresponds to 0.3 rps^2.

All **ACCEL** statement parameters are used for the corresponding decelerations, if those **DECEL** statements have not been entered or defined to be zero. If none of the **ACCEL** parameters (**JOG**, **HOME**, **JOY**, **VSCALE**, or **PATH**) are entered, the **ACCEL** values are used for the following **MOVE** statement options:

- **SLEWCW**
- **SLEWCCW**
- **CONSTANT**
- **Q VARIABLE**
- **STOP**

Example

The example shown below sets the acceleration for Axes 1 and 2 to 50,000 steps/sec^2. If the **UNIT ACCEL** scale factor is 1, this corresponds to an acceleration of 2 rps^2 with a standard 25,000-step/rev Compumotor. Axis #3 uses the value of variable **Q4** and axis #4 remains unchanged. If there was no previous setting, the default value of 25,000 is retained.

Statement	Description
`ACCEL 50000 50000 Q4 *`	`'Sets acceleration values`
`VEL 25000 20000 Q3 *`	`'Sets velocity values`
`MOVE 25000 25000 Q2 *`	`'Moves axes 1 & 2 25,000 steps (moves 'axis 3 Q2 steps)`
`ACCEL HOME 25000 25000 25000 *`	`'Sets go home acceleration for axes 1, '2 and 3`
`ACCEL JOG 25000 25000 25000 *`	`'Sets jog acceleration for axes 1,2, & '3`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ACCEL JOG

Name	**ACCEL JOG**
Descriptor	Set Jog Acceleration
Type	Motion
Default	ACCEL JOG * * * *
Syntax	ACCEL 250000 200000 300000 500000
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **ACCEL JOG** statement specifies the jog acceleration for each axis. The entries may be a number, a variable (**Q1** - **Qnn**), or the word **NULL** (*), in which case no change is made and the last defined value will remain in effect for that axis. The **ACCEL JOG** statement is scaled by the **UNIT ACCEL** parameter.

See Also: **DECEL JOG, ENABLE JOG, ENABLE TCH_JOG, VEL JOGL, VEL JOGH**

ACCEL HOME

Name	**ACCEL HOME**
Descriptor	Set Move Home Acceleration
Type	Motion
Default	ACCEL HOME * * * *
Syntax	ACCEL HOME Q1 25000 50000
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **ACCEL HOME** statement specifies the acceleration rate and deceleration rate that each axis will move during a move home. The entries may be a number, a variable (**Q1** - **Qnn**), or the word **NULL** (*), in which case no change is made, and the last defined value will remain in effect. The **ACCEL HOME** statement uses the **UNIT ACCEL** value as a multiplier.

See Also: **VEL HOME, DEFINE HOME**

ACCEL PATH

Name	**ACCEL PATH**
Descriptor	Set Linear Interpolation Path Acceleration
Type	Motion
Default	ACCEL PATH *
Syntax	ACCEL PATH 25000
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **ACCEL PATH** statement specifies the acceleration used for a linear interpolation move and the contouring option. The path acceleration is the vector sum of the accelerations of the axis participating in the interpolation. The **ACCEL** parameters for preset moves are not used for the axis in an interpolated move. The **ACCEL PATH** statement uses the **UNIT PATH ACCEL** value when scaling the path acceleration.

See Also: **LINT, MOVE, DECEL PATH, VEL PATH**

Example

In the program below the path acceleration is 250,000. Using the distances given in the statement example, the individual accelerations for axes 1, 3, and 4 calculated by the Model 4000 are:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

$$\text{Axis 1 ACCEL} = \left[\frac{25000}{\sqrt{((25000^2)+(200000^2)+(100000^2))}}\right] \times (250000) = 27777.78$$

$$\text{Axis 3 ACCEL} = \left[\frac{200000}{\sqrt{((25000^2)+(200000^2)+(100000^2))}}\right] \times (250000) = 222222.22$$

$$\text{Axis 4 ACCEL} = \left[\frac{100000}{\sqrt{((25000^2)+(200000^2)+(100000^2))}}\right] \times (250000) = 111111.11$$

Axes #1, #3, and #4 will start moving, reach their peak velocity, start decelerating, and stop at the same time. Axis #2 will not participate in the linear interpolated move, it does a preset move of 50,000 steps.

Statement	Description
LINT MODE YES NO YES YES	'Axis #2 is not participating in **LINT** 'move
ACCEL PATH 250000	
DECEL PATH 250000	
VEL PATH 250000	
MOVE 25000 50000 200000 100000	'These are distances used in the above 'calculations.

ACCEL JOY

Name	**ACCEL JOY**
Descriptor	Set Joy Acceleration
Type	Motion
Default	ACCEL JOY * * * *
Syntax	ACCEL JOY 250000 200000 300000 500000
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **ACCEL JOY** statement specifies the joy acceleration for each axis. The entries may be a number, a variable (**Q1 - QNN**), or the word **NULL** (*), in which case no change is made, and the last defined value will remain in **ACCEL** parameter. **ACCEL JOY** is scaled by the **UNIT ACCEL** parameter.

See Also: **ENABLE JOY, ENABLE TCH_JOY, VEL JOYH, VEL JOYL, DEFINE JOYSL, DEFINE JOYSH, DECEL JOY**

ACCEL VSCALE

Name	**ACCEL VSCALE**
Descriptor	Acceleration Velocity Scaling Parameters
Type	Set-up
Default	ACCEL VSCALE *
Syntax	ACCEL VSCALE 10000
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **ACCEL VSCALE** statement specifies the maximum velocity scale acceleration in terms of %/sec. This feature allows you to prevent rapid motor deceleration due to a sudden change in the commanded velocity scale percentage. Because the maximum value for velocity scaling is 100% and the update rate is 2 ms, the maximum value of velocity scale acceleration is 50,000%/sec (100/2ms = 50,000%).

See Also: **ENABLE VSCALE, DEFINE VSCL_L, DEFINE VSCL_H**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

BREAK

Name	**BREAK**
Descriptor	Clear GOSUB Calls and Loops
Type	Programming
Default	BREAK
Syntax	BREAK
Options	N/A

F1 F2 F3 F4 F5 F6

Description

The **BREAK** statement *breaks* the program control out of any loops or subroutines it is currently executing. This statement should be used when you are in a subroutine or loop and you need to get out without using the normal **RETURN** or **ENDLOOP** statements. This may happen if an **ON** statement becomes true and you use the **GOTO** option instead of the **GOSUB** option. If the **BREAK** statement is used, branching back to the original subroutine or loop after you have completed your task is not necessary. Do not follow a **BREAK** with a **RETURN**, all **GOSUB** calls have been cleared.

During the execution of the program below, activating input #1 will jump to the **STOP** label program, regardless of which line was executing. At the **STOP** label, the motors are stopped, moved home, and the **BREAK** statement resets the loop counter and clears any **GOSUB** calls in progress. A jump is then made back to the beginning of the program.

See Also: **GOSUB**, **LABEL**, **LOOP**, **ENDLOOP**

Example

```
Statement                                        Description
LABEL START
PATT1 1XXX XXXX XXXX XXXX XXXX XXXX
ON IN24 = PATT1 GOTO STOP
LOOP Ø                                           'Beginning of loop; 0 causes
                                                 'infinite looping
IF F-KEY1 GOSUB GOBACK
IF F-KEY2 GOSUB FASTMV
IF F-KEY3 GOSUB SLOWMV
ENDLOOP                                          'End of loop
LABEL STOP
MOVE STOP STOP STOP STOP
MOVE HOMECW HOMECW HOMECW HOMECW
BREAK
OUT LCD2,Ø1 ^AT HOME^
GOTO START
```

DATA

Name	**DATA**
Descriptor	Store Data Values
Type	Programming
Default	DATA n
Syntax	DATA 12.78 67843 87 77 1.441
Options	N/A

F1 F2 F3 F4 F5 F6

Description

The **DATA** statement stores the values used by variable **Qn** in the **IN Qn = NEXT DATA** statement during program execution (refer to the **IN** statement later in this chapter). The **DATA** statement may contain as many numeric constants as will fit on a line up to a ten item limit. You can use any number of **DATA** statements in a program.

The Model 4000 treats the constants in all **DATA** statements as one continuous list of items or one continuous **DATA** statement. When an **IN Qn = NEXT DATA** statement is encountered during program execution, the Model 4000 searches through the entire program for **DATA** statements, then it assigns the next **DATA** element to the specified **Qn** variable. If an **IN Qn = NEXT DATA** statement is encountered again, the next data

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

element is assigned to the specified Qn variable. The **RESTORE** statement causes the Model 4000 to return to the first constant in the first **DATA** statement, but does not assign this value to the variable.

An automatic **RESTORE** is executed by the Model 4000 whenever a program is started (using **RUN** or the **START** keys). If a **GOSUB** or **GOTO** is used to branch to another program in the Model 4000, the **DATA** statements must appear in the calling or *initial* program.

Each **DATA** parameter must be within **±879,609,302.2207**. Enter each number using the keypad. Separate each number with a space by pressing **C/E** and enter the line of data by pressing **ENTER**.

See Also: **IN DATA**

Example

The program below shows the use of the **DATA** statement to move axis #1 50. 44, or 69 steps

The program resets the **DATA** statement to the first constant (50) using the **RESTORE** statement but does not assign this value to a **Q** variable (i.e., **Q1** maintains the last value assigned to it).

*If the **Qn = NEXT DATA** statement is encountered during execution, after all the **DATA** statement constants have been used up, the program will abort.*

Statement	Description
`DATA 50 44 69 99`	`'Defines 4 data points`
`MATH Q4 = 99`	`'Defines ending data point`
`LABEL MOVE_LP`	`'Defines label MOVE_LP`
`IN Q1 = NEXT DATA`	`'Sets Q1 to input data value`
`IF Q1 = Q4 GOTO ITS_DONE`	`'If ending data point, next data 'pointer and start over.`
`MOVE Q1 * * *`	`'Moves axis 1 Q1 distance`
`GOTO MOVE_LP`	`'Goes back to MOVE_LP label`
`LABEL ITS_DONE`	`'Defines ITS_DONE label`
`GOTO MOVE_LP`	`'Restart the move loop`
`RESTORE`	`'Restores data pointer to first data 'point`

DECEL

Name	**DECEL**					
Descriptor	Set Deceleration					
Type	Motion					
Default	DECEL * * * *					
Syntax	DECEL 25000 25000 * *					
Options	TAB	Q	NULL	SLIMIT	HLIMIT	ETC
	TAB	JOG	PATH	JOY		ETC
	F1	**F2**	**F3**	**F4**	**F5**	**F6**

Description

The **DECEL** statement specifies the deceleration value for each axis. The entries may be a number, a variable (**Q1 - QNN**), or the **NULL** (*) symbol, in which case the previous value will be used. *The value entered will be multiplied by the scale factor you entered using the **UNIT ACCEL** statement.*

Until a **DECEL** statement is executed, the corresponding **ACCEL** statement value will be used for the deceleration value. Once a **DECEL** value is executed, it will remain in effect until you enter a new value, zero, or the next power down cycle.

The Model 4000 decelerates each axis at a rate of 2,500,000 steps/sec² unless you change the rate with this statement. This default value corresponds to 100 rps² for a standard 25,000 step/rev motor.

When one of the **DECEL** options is not chosen (**SLIMIT**, **JOY**, **HLIMIT**, **JOG**, or **PATH**), the deceleration values you enter will affect the following **MOVE** statement options:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- ❑ SLEWCW
- ❑ SLEWCCW
- ❑ CONSTANT
- ❑ Q VARIABLE
- ❑ STOP

The DECEL value does not affect the MOVE HOMECW or MOVE HOMECCW. The ACCEL HOME value is used.

See Also: UNIT ACCEL, ACCEL, VEL, MOVE

Example

The example below sets the deceleration for axes #1 and #2 to 50,000 steps/sec^2. If the UNIT ACCEL scale factor is 1, this corresponds to an deceleration of 2 rps^2 with a standard 25,000-step/rev Compumotor motor. Axis #3 uses the value of variable Q4 and axis #4 remains undisturbed from the previous setting. If there was no previous setting, the default value of 25,000 is retained.

Statement	Description
`ACCEL 10000 50000 25000 *`	`'Sets acceleration values`
`DECEL 50000 50000 Q4 *`	`'Sets deceleration values`
`VEL 25000 20000 Q3 *`	`'Sets velocity values`
`MOVE 25000 25000 Q2 *`	`'Moves axes 1 & 2 25,000 steps.` `'Moves axis 3 Q2 steps.`
`DECEL HLIMIT 25000 25000 25000 *`	`'Sets the hardware limit` `'deceleration for axes 1, 2 and 3`
`ACCEL JOG 25000 25000 25000 *`	`'Sets jog deceleration for axes 1,2,` `'& 3.`

The exceptions to this general rule are the limit decelerations. In these cases, there is no corresponding acceleration. Specifying a zero value will instruct the Model 4000 to stop instantly upon encountering a limit.

DECEL SLIMIT

Name	DECEL SLIMIT
Descriptor	Deceleration When Software Travel Limit Reached
Type	Motion
Default	DECEL SLIMIT * * * *
Syntax	DECEL SLIMIT 25000 25000 25000 *
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The DECEL SLIMIT statement specifies the deceleration rate that each axis will use when a soft limit is reached if the move was moving in the direction of the active limits. The entries may be a number, a variable (Q1 - QNN), or the word NULL ([*] no change). The last defined value will remain in effect. *A zero value will instruct the Model 4000 to stop instantly upon encountering a limit.* The DECEL SLIMIT statement is scaled by the UNIT ACCEL parameter.

See Also: ENABLE M_SLIM, ENABLE E_SLIM, SLIMIT, ON SLIMIT

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DECEL HLIMIT

Name	**DECEL HLIMIT**
Descriptor	Deceleration When Hardware Travel Limit Reached
Type	Motion
Default	DECEL HLIMIT * * * *
Syntax	DECEL HLIMIT Q10 1000000 500 *
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **DECEL HLIMIT** statement specifies the deceleration rate that each axis will use when a hardware limit is reached. The entries may be a number, a variable (**Q1 - QNN**), or the word **NULL** (*). The last defined value will remain in effect. The **DECEL HLIMIT** statement is scaled by the **UNIT ACCEL** parameter.

See Also: **ENABLE HLIMIT, ON HLIMIT**

DECEL JOG

Name	**DECEL JOG**
Descriptor	Set Jog Deceleration
Type	Motion
Default	DECEL JOG * * * *
Syntax	DECEL JOG 72.1 Q87 9987*
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **DECEL JOG** statement specifies the maximum deceleration for each axis for **JOG** moves. The entries may be a number, a variable (**Q1 - QNN**), or the word **NULL** (*). The last defined value will remain in effect. The **DECEL JOG** statement is scaled by the **UNIT ACCEL** parameter.

See Also: **ENABLE JOG, ENABLE TCH_JOG, ACCEL JOG, VEL JOGL, VEL JOGH**

DECEL PATH

Name	**DECEL PATH**
Descriptor	Deceleration for Interpolation Moves
Type	Motion
Default	DECEL PATH *
Syntax	DECEL PATH Q10
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **DECEL PATH** statement specifies the **PATH** deceleration used in an interpolation move and the contouring option. For each linearly interpolated move, each axis' deceleration is computed based upon each axis distance component of that move. The **DECEL PATH** statement uses the **UNIT PATH ACCEL** value when scaling the path deceleration.

See Also: **LINT, MOVE, ACCEL PATH, VEL PATH**

Example

In the program below, the path velocity is 250,000. Using the distances given, the individual decelerations calculated by the Model 4000 are:

$$\text{Axis 1 DECEL} = \left[\frac{2500}{\sqrt{((250000^2)+(200000^2)+(100000^2))}}\right] \times (250000) = 27777.78$$

$$\text{Axis 3 DECEL} = \left[\frac{20000}{\sqrt{((250000^2))+(200000^2)+(100000^2)))}}\right] \times (250000) = 222222.22$$

$$\text{Axis 4 DECEL} = \left[\frac{10000}{\sqrt{((250000^2))+(200000^2))+(100000^2)))}}\right] \times (250000) = 111111.11$$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Axes 1, 3, and 4 will start moving, reach their peak velocity, start decelerating, and stop at the same time. Axis #2 will not participate in the linear interpolated move, it does a preset move of 50,000.

See Also: **LINT, MOVE, MOVI**

Statement	Description
`LINT MODE YES NO YES YES`	`'Axis #2 is not participating in 'LINT moves`
`ACCEL PATH 250000`	
`DECEL PATH 250000`	
`VEL PATH 250000`	
`MOVE 25000 50000 200000 100000`	`'These are distances used in the 'calculation above.`

DECEL JOY

Name	**DECEL JOY**
Descriptor	Set Joystick Deceleration
Type	Motion
Default	DECEL JOY * * * *
Syntax	DECEL JOY 72.1 Q87 9987 *
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **DECEL JOY** statement specifies the maximum deceleration for each axis for **JOY** moves. The entries may be a number, a variable (**Q1 - QNN**), or the word **NULL** (*). The last defined value will remain in effect. The **DECEL JOY** statement is scaled by the **UNIT ACCEL** parameter.

See Also: **ENABLE JOY, ENABLE TCH_JOY, ACCEL JOY, VEL JOYL, VEL JOYH, DEFINE JOYSL, DEFINE JOYSH**

DEFINE

Name	**DEFINE**					
Descriptor	Define Input and Output Parameters					
Type	Set-Up					
Default	DEFINE					
Syntax	DEFINE IN24 POSITIVE LOGIC					
Options	TAB	IN24	OUT24	JOYSL	JOYSH	ETC
	TAB	CENTER	CTR_DB	END_DB	VSCL_L	ETC
	TAB	VSCL_H	HOME	GPIB	FAULT	ETC
	TAB	TERM	VAR	ARROW		ETC
	F1	F2	F3	F4	F5	F6

Description

The **DEFINE** statement sets various system parameters. These parameters include input and output logic levels, analog input parameters, and velocity scaling parameters. You can define the 24 programmable inputs and outputs with positive or negative logic using the **DEFINE IN24** or **DEFINE OUT24** statements.

The joystick select input selects one of two configurations for the analog inputs in joystick mode. These two configurations are **JOYSL** (when the joystick select input is low) and **JOYSH** (when the joystick select input is high). The **DEFINE JOYSL** and **DEFINE JOYSH** statements define which analog input controls which axis (in joystick mode).

DEFINE CENTER The `DEFINE CENTER` statement defines the voltage level of an analog input that commands no motion for the joystick axis.

DEFINE CTR_DB The `DEFINE CTR_DB` statement defines the change in voltage (increase or decrease) from the center voltage required before motion is commanded for that joystick axis.

DEFINE END_DB The `DEFINE END_DB` statement determines the minimum and maximum voltages for the full range of analog input. This is used for both joystick and velocity scaling.

DEFINE VSCL_L The `DEFINE VSCL_L` statement sets the source for velocity scaling (feed rate over-ride) when the joystick select input is low (current flowing). The **DEFINE VSCL_H** statement sets the source for velocity scaling when the joystick select input is high (current not flowing).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DEFINE GPIB	The DEFINE GPIB statement allows optional IEEE-488 communications port parameters to be defined. (Refer to the *Model 4000 Options User Guide.)*
DEFINE FAULT	The DEFINE FAULT statement allows the fault input polarity to be selected (pin #5 on the motor drive connector).
DEFINE TERM	The DEFINE TERM statement allows the termination character for Ports 1, 2, and 3 to be changed from carriage returns to any other character.
DEFINE VAR	The DEFINE VAR statement determines how many numeric variables, how many string variables, and the length of each string variable you may use in a program.
DEFINE ARROW	The DEFINE ARROW statement allows you to change the function of the arrow keys.

DEFINE IN24

Name	**DEFINE IN24**
Descriptor	Sets 24 Inputs Logic Type
Type	Set-Up
Default	DEFINE IN24
Syntax	DEFINE IN24 POSITIVE LOGIC
Options	TAB POS NEG
	F1 F2 F3 F4 F5 F6

Description

The **DEFINE IN24** statement defines the correlation between the input voltage level and its corresponding logic level. If you choose *positive logic*, the Model 4000 assumes 5V on an input pin is a logic 1 and 0V on an input is a logic Ø.

Positive logic: Logic 1 $\geq$ 2.4VDC (Current not flowing)
Logic Ø $\leq$ 0.4VDC (Current flowing)

Negative logic: Logic 1 $\leq$ 0.4VDC (Current flowing)
Logic Ø $\geq$ 2.4VDC (Current not flowing)

After a **RESET** function is executed, the default **DEFINE** statement parameter is negative logic for both inputs and outputs. All outputs are 5V (current not flowing, [i.e., a BCD or HEX zero for all 24 outputs]).

See Also: **PATT, IN IN24, IN BIT, ON IN24, ON BIT, IF IN24, IF BIT**

Example

The following example defines the 24 inputs to the Model 4000 as the negative logic. Then, it waits for the inputs to match pattern 1 before it executes the move.

Statement	Description
`DEFINE IN24 NEGATIVE LOGIC`	`'Defines 24 inputs as negative 'logic`
`PATT1 1ØXX XXXX XXXX XXXX XXXX XXXX`	`'Defines pattern 1 as input 24 'going to logic 1 and input 23 'going to logic 0 the rest do 'not affect system operation`
`WAIT FOR IN24 = PATT1`	`'Waits for the 24 inputs to 'match pattern 1`
`MOVE 100000 * * *`	`'Moves axis #1 100000 steps`

DEFINE OUT24

Name	**DEFINE OUT24**
Descriptor	Sets 24 Output Logic Type
Type	Set-Up
Default	DEFINE OUT24
Syntax	DEFINE OUT24 NEGATIVE LOGIC
Options	TAB POS NEG
	F1 F2 F3 F4 F5 F6

Description

The **DEFINE OUT24** statement defines the correlation between the output voltage level and its corresponding logic level. If you choose *positive logic*, the Model 4000 assumes 5V on an output pin is a logic 1 and 0V on an output is a logic Ø.

If you change the logic type, the voltage at the output will also change. For example, if positive logic is selected and OUT1 has a value of 1, approximately 5V will be present at output pin #47 (no current flowing). If a **DEFINE OUT24 NEGATIVE LOGIC** statement is executed, the voltage at pin #47 will change to 0V (current flowing). *Changing the logic type of*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

the 24 outputs will not change the representation of the **OUT24** given by the **DISPLAY OUT24** statement, even though the voltages do change. The value of 1 will still be on the operator panel when the **DEFINE OUT24 NEGATIVE LOGIC** statement is executed.

Positive Logic	Logic 1 ≥ 2.4VDC (current not flowing)
	Logic Ø ≤ 0.4VDC (current flowing)
Negative Logic	Logic 1 ≤ 0.4VDC (current flowing)
	Logic Ø ≥ 2.4VDC (current not flowing)

After you run the **RESET** function, the default **DEFINE** statement parameter is negative logic for both inputs and outputs. All outputs are 5V (current not flowing, i.e., a BCD or HEX zero for all 24 outputs).

When power is cycled 5V (current not flowing) will be present at all 24 outputs. Thus, for positive logic, a hexadecimal value of 0FFFFFFH or all ones is the value of the outputs. For negative logic, all 24 outputs have a 0 value, which is 0 in BCD and hexadecimal.

See Also: **OUT OUT24, OUT BIT**

Example

```
Statement                                  Description
ACCEL 100000 50000 *    *                  'Sets acceleration for axes #1
                                           'and #2
VEL 50000 50000 *    *                     'Sets velocity for axes 1 and 2
DEFINE OUT24 POSITIVE LOGIC                'Sets 24 outputs as positive
                                           'logic
PATT2 111Ø XXXX XXXX XXXX XXXX XXXX        'Defines 24 output pattern as
                                           'output 24, 23, and 22 turning
                                           'on to logic 1 and output 21
                                           'going to logic 0
MOVE 20000 *    *    *                     'Moves axis 1 20000 steps
OUT OUT24 PATT2                            'Turn on 24 outputs as defined
                                           'by PATT2
```

DEFINE JOYSL

Name	**DEFINE JOYSL**
Descriptor	Define Joystick Source-Low
Type	Set-Up
Default	DEFINE JOYSL * * * *
Syntax	DEFINE JOYSL CH1 CH2 OFF OFF

Options						
	TAB	NULL	CH1	CH2	CH3	ETC
	TAB	CH4	OFF			ETC
	F1	F2	F3	F4	F5	F6

Description

The **DEFINE JOYSL** statement defines which analog input controls which axis, when the joystick select input is low (current flowing) in joystick mode.

The four analog inputs are referenced as CH1-CH4. If you enter the OFF option for an axis, that axis cannot be controlled by an analog input when the Joystick mode is selected. One analog input can control more than one axis, depending on how it is configured.

The default axes are Axes #1 - #4 are controlled by analog input channels 1-4.

See Also: **ENABLE JOY, ENABLE TCH_JOY, VEL JOYH, VEL JOYL, ACCEL JOY, DECEL JOY, DEFINE JOYSH, DEFINE CENTER**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example

Statement	Description
`ENABLE JOY`	`'Enter joystick mode`
`DEFINE JOYSL CH1 CH2 OFF OFF`	`'Analog input #1 controls axis 1 and 'analog input #2 controls axis 2 'when joystick input is low`
`DEFINE JOYSH OFF OFF CH3 CH4`	`'Analog #3 controls axis 3 and 'analog axis #4 controls axis 3 when 'the joystick select input is high`

DEFINE JOYSH

Name	**DEFINE JOYSH**
Descriptor	Define Joystick Source-High
Type	Set-Up
Default	DEFINE JOYSH * * * *
Syntax	DEFINE JOYSH OFF OFF CH3 CH4

Options						
	TAB	NULL	CH1	CH2	CH3	ETC
	TAB	CH4	OFF			ETC
	F1	F2	F3	F4	F5	F6

Description

The **DEFINE JOYSH** statement defines which analog input controls which axis, when the joystick select input is high (current not flowing), in joystick mode.

The four analog inputs are referenced as CH1-CH4. If you enter the OFF option for an axis, that axis cannot be controlled by an analog input when the Joystick mode is selected. One analog input can control more than one axis, depending on how it is configured. The default axes are Axes #1 - #4 are controlled by analog input channels 1-4.

See Also: **ENABLE JOY, ENABLE TCH_JOY, VEL JOYH, VEL JOYL, ACCEL JOY, DECEL JOY, DEFINE JOYSL, DEFINE CENTER**

Example

Statement	Description
`ENABLE JOY`	`'Enter joystick mode`
`DEFINE JOYSL CH1 CH2 OFF OFF`	`'Analog input #1 controls axis 1 and 'analog input #2 controls axis 2 when 'joystick input is low`
`DEFINE JOYSH OFF OFF CH3 CH4`	`'Analog #3 controls axis 3 and analog 'axis #4 controls axis 4 when joystick 'select input is high`

DEFINE CENTER

Name	**DEFINE CENTER**
Descriptor	Define Joystick Center
Type	Set-Up
Default	DEFINE CENTER *
Syntax	DEFINE CENTER ALL = AUTOREAD

Options						
	TAB	CH1	CH2	CH3	ALL	
	TAB	AUTOREAD	Q			
	F1	F2	F3	F4	F5	F6

Description

The **DEFINE CENTER** statement defines the voltage level of an analog input commanding no motion for the joystick axis. The voltage value can be a constant, a variable content, or the **AUTOREAD** option. If a constant or variable (**Q1 - QNN**) is used, the value must be between 0.15V and 2.4V. The maximum resolution is hundredths of a volt. If you choose the **AUTOREAD** option, the analog input voltage is read at the time the statement is executed. This voltage is set as the center value.

The **ALL** option means all four analog inputs center voltage levels will be set when the statement is executed. *The default center voltage is 1.25V.*

See Also: **ENABLE JOY, ENABLE TCH_JOY, VEL JOYH, VEL JOYL, ACCEL JOY, DECEL JOY, DEFINE JOYSL, DEFINE CENTER**

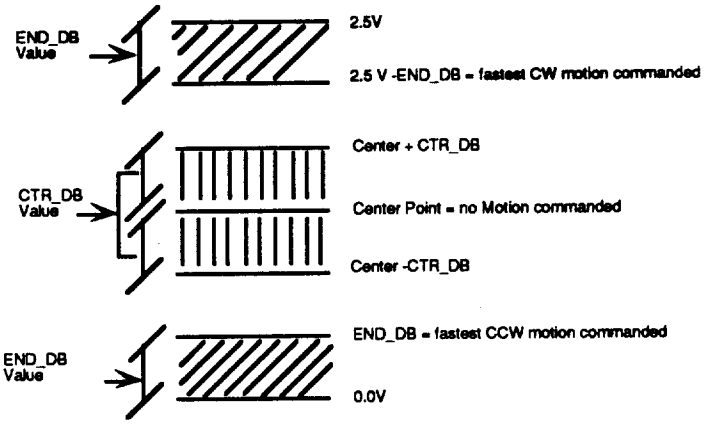
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DEFINE CTR_DB

Name	**DEFINE CTR_DB**
Descriptor	Define Joystick Center Deadband
Type	Set-Up
Default	DEFINE CTR_DB *
Syntax	DEFINE CTR_DB ALL = 0.2
Options	TAB CH1 CH2 CH3 CH4 ALL TAB Q F1 F2 F3 F4 F5 F6

Description

The **DEFINE CTR_DB** statement sets the range of voltage at the center voltage point which will command no motion in Joystick mode. This statement is useful if the joystick does not return to the same resting position. The value entered can be a constant or a variable. The value must be between 0 and 1.24V. The entire voltage range at the center point which commands no motion is equal to twice the value entered. *The default value is 0.1V.*



See Also: **ENABLE JOY, ENABLE TCH_JOY, VEL JOYH, VEL JOYL, ACCEL JOY, DECEL JOY, DEFINE JOYSL, DEFINE CENTER**

DEFINE END_DB

Name	**DEFINE END_DB**
Descriptor	Define Analog End Deadband
Type	Set-Up
Default	DEFINE END_DB *
Syntax	DEFINE END_DB CH3 = 0.1
Options	TAB CH1 CH2 CH3 CH4 ALL TAB Q F1 F2 F3 F4 F5 F6

Description

The **DEFINE END_DB** statement determines the minimum and maximum voltages for the full range of analog inputs. This is used for both joystick and velocity scaling. This statement is useful if your joystick does not reach either limit of the voltage range (0V or 2.5V). This statement reduces the range to fit the voltage range for that joystick. When an analog input is used for velocity scaling, this statement defines the range of voltage used to command the velocity scaling percentage. *The default value is 0.1V.*

See Also: **ENABLE JOY, ENABLE TCH_JOY, VEL JOYH, VEL JOYL, ACCEL JOY, DECEL JOY, DEFINE JOYSL, DEFINE CENTER, ENABLE VSCALE, DEFINE VSCL_L, DEFINE VSCL_H, ACCEL VSCALE**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DEFINE VSCL_L

Name	**DEFINE VSCL_L**
Descriptor	Define Velocity Scaling Source-Low
Type	Set-Up
Default	DEFINE VSCL_L SOURCE *
Syntax	DEFINE VSCL_L SOURCE = CH3

Options						
	TAB	CH1	CH2	CH3	CH4	ETC
	TAB	Q	OFF			ETC
	F1	F2	F3	F4	F5	F6

Description

The **DEFINE VSCALE_L** statement defines the source used for velocity scaling when the joystick select input is low (current flowing), and velocity scaling is enabled.

The four analog inputs are referred to as CH1-CH4. If one of these channels is given as the source and velocity scaling is enabled, all velocities will be scaled as a percentage of the full scale. The percentage is a function of the voltage present on that channel and the end deadband defined for that channel. The percentage can be expressed with the following formula:

Percentage = (V_in - end _db) / (2.5V - 2 * end_db).
where: V_in = input voltage, end_db = user-defined end deadband voltage

If the **OFF** option is selected, all velocities will be full scale (100%) when the joystick select input is low. *The default for this statement is OFF.*

If the **Qn** option is selected, that **Qn** variable is expected to contain the desired velocity scale percentage when velocity scaling is enabled. If velocity scaling is enabled and the **Qn** variable contains a number outside the range of 0 to 100, an execute error will result.

See Also: **DEFINE END_DB**, **ENABLE VSCALE**, **DEFINE VSCL_H**

DEFINE VSCL_H

Name	**DEFINE VSCL_H**
Descriptor	Define Velocity Scaling Source-High
Type	Set-Up
Default	DEFINE VSCL_H SOURCE = *
Syntax	DEFINE VSCL_H SOURCE = CH3

Options						
	TAB	CH1	CH2	CH3	CH4	ETC
	TAB	Q	OFF			ETC
	F1	F2	F3	F4	F5	F6

Description

The **DEFINE VSCALE_H** statement defines the source used for velocity scaling when the joystick select input is high (current not flowing), and velocity scaling is enabled.

The four analog inputs are referred to as CH1-CH4. If one of these channels is given as the source and velocity scaling is enabled, all velocities will be scaled as a percentage of the full scale. The percentage is a function of the voltage present on that channel and the end deadband defined for that channel. The percentage can be expressed with the following formula:

Percentage = (V_in - end _db) / (2.5V - 2 * end_db).
where: V_in = input voltage, end_db = user-defined end deadband voltage.

If the **OFF** option is selected, all velocities will be full scale (100%) when the joystick select input is low. *The default for this statement is OFF.*

If the **Qn** option is selected, that **Qn** variable is expected to contain the desired velocity scale percentage when velocity scaling is enabled. If velocity scaling is enabled and the **Qn** variable contains a number outside the range of 0 to 100, an execute error will result.

See Also: **DEFINE END_DB**, **ENABLE VSCALE**, **DEFINE VSCL_L**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DEFINE HOME

Name	DEFINE HOME
Descriptor	Define Home Parameters
Type	Set-Up
Default	DEFINE HOME *
Syntax	DEFINE HOME TYPE HE+ HE- HE+Z HE-Z
Options	TAB TYPE EDGE FDIR BACK
	F1 F2 F3 F4 F5 F6

Description

The **DEFINE HOME** statement defines the home input for each axis. Home may be described in terms of three parameters.

HOME TYPE Describes the combination of **HE** home enable active polarity and Z channel used.

HOME EDGE Specifies which edge of the **HE** and **Z** input is to be used as the active edge.

HOME FDIR Specifies the final approach direction used at the end of the homing sequence.

HOME BACK Determines whether the final home sequence move of backing up to the active edge is made.

Refer to *Chapter 5, Application Design* for a more detailed description of homing.

Example

Statement	Description
`DEFINE HOME TYPE HE+ HE-Z HE+ HE-Z`	`'Axes 1 and 3 use a high level for 'home enable. Axes 2 and 4 use the 'Z channel and low level home 'enable.`
`DEFINE HOME EDGE CW CW CCW CCW`	`'Axes 1 and 2 use the CW edge of 'home. Axes 3 and 4 use the CCW 'edge of home.`
`DEFINE HOME FDIR CCW CCW CW CW`	`'Axes 1 and 2 have CCW selected as 'the final approach direction, Axes '3 and 4 use CW as the final 'approach direction.`
`MOVE HOMECW HOMECCW * *`	`'Axis 1 uses CW as the initial 'search direction, axis 2 uses CCW 'as the initial search direction.`

DEFINE HOME TYPE

Name	DEFINE HOME TYPE
Descriptor	Define Home Active State
Type	Set-Up
Default	DEFINE HOME TYPE * * * *
Syntax	DEFINE HOME TYPE HE+ HE- HE+Z HE-Z
Options	TAB HE+ HE- HE+Z HE-Z NULL
	F1 F2 F3 F4 F5 F6

Description

This command sets the active state of the home input. It enables you to use either a normally closed or a normally open switch. **HE-** requires a closed circuit (current flowing, 0V) signal to activate the home limit input (normally open). **HE+** requires a open circuit (current not flowing, 5V) signal to activate the home limit input (normally closed).

If you choose **HE+Z** or **HE-Z** as the Z channel determines the home position. The switch determines the home region (in conjunction with a load activated switch connected to home limit). The Z channel determines the exact position in that region to be used as the home reference.

HE+ Use normally closed switch, and no Z channel from an incremental encoder
HE- Use normally open switch, and no Z channel from an incremental encoder
HE+Z Use normally closed switch, and the Z channel as the final home position
HE-Z Use normally open switch, and the Z channel as the final home position

See Also: **DEFINE HOME BACK, DEFINE HOME EDGE, DEFINE HOME FDIR, ACCEL HOME, VEL HOME, MOVE HOMECW/HOMECCW**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example

Statement	Description
`DEFINE HOME TYPE HE+ HE- HE+Z HE-Z`	`'Axes 1 and 3 use a high level 'for home enable. Axes 3 and 4 'use the Z channel and home 'enable.`
`DEFINE HOME EDGE CW CW CCW CCW`	`'Axes 1 and 2 use the CW edge of 'home. Axes 3 and 4 use the CCW 'edge of home`
`DEFINE HOME FDIR CCW CCW CW CW`	`'Axes 1 and 2 have CCW selected 'as the final approach 'direction, Axes 3 and 4 use CW 'as the final approach direction`
`MOVE HOMECW HOMECCW * *`	`'Axis 1 uses CW as the initial 'search direction, axis 2 uses 'CCW as the initial search 'direction`

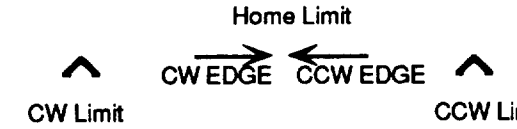
DEFINE HOME EDGE

Name	**DEFINE HOME EDGE**
Descriptor	Define Home Active Edge
Type	Set-Up
Default	DEFINE HOME EDGE * * * *
Syntax	DEFINE HOME EDGE CW CCW CW*
Options	TAB CW CCW NULL
	F1 F2 F3 F4 F5 F6

Description

- **EDGECCW:** Selects the CCW side of the home signal as the *edge* on which the final approach will stop (Factory Default)
- **EDGECW:** Selects the CW side of the home signal as the *edge* on which the final approach will stop

To achieve repeatable homing, the Model 4000 uses the **HOME EDGE** statement to select which side of the home switch the motor should stop at while homing. The side of the home switch that is closest to the CW limit is called the CW edge, and the side closest to the CCW limit is called the CCW edge.



See Also: **DEFINE HOME BACK, DEFINE HOME TYPE, DEFINE HOME FDIR, ACCEL HOME, VEL HOME, MOVE HOMECW/HOMECCW**

Example

Statement	Description
`DEFINE HOME TYPE HE+ HE- HE+Z HE-Z`	`'Axes 1 and 5 use a high level for 'home enable. Axes 3 and 4 use the Z 'channel and home enable`
`DEFINE HOME EDGE CW CW CCW CCW`	`'Axes 1 and 2 use the CW edge of 'home. Axes 3 and 4 use the CCW edge 'of home`
`DEFINE HOME FDIR CCW CCW CW CW`	`'Axes 1 and 2 have CCW selected as 'the final approach direction, Axes 3 'and 4 use CW as the final 'approach 'direction`
`MOVE HOMECW HOMECCW * *`	`'Axis 1 uses CW as the initial search 'direction, axis 2 uses CCW as the 'initial search direction`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DEFINE HOME FDIR

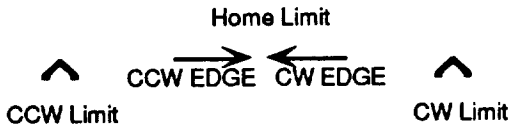
Name	DEFINE HOME FDIR
Descriptor	Define Home Final Direction
Type	Set-Up
Default	DEFINE HOME FDIR * * * *
Syntax	DEFINE HOME FDIR CW CCW CW *
Options	TAB CW CCW NULL
	F1 F2 F3 F4 F5 F6

Description

- ❑ **FDIRCW:** Sets final home approach direction as CW (Factory Default)
- ❑ **FDIRCCW:** Sets final home approach direction as CCW

When you execute the **MOVE HOME** statement, the motor will make the initial move at **VEL HOME**, until it sees the active home switch. Depending on the homing set-up, the final move to the home switch will be determined by **HOME FDIR** and **HOME EDGE** statements.

The **DEFINE HOME FDIR** determines the direction of the final creep move during homing. The final creep move can be used to eliminate mechanical backlash in the system.



This enables you to approach home from any direction yet stop at home repeatedly on the same edge of the switch.

See Also: **DEFINE HOME BACK, DEFINE HOME EDGE, DEFINE HOME TYPE, ACCEL HOME, VEL HOME, MOVE HOMECW/HOMECCW**

Example

Statement	Description
`DEFINE HOME TYPE HE+ HE- HE+Z HE-Z`	`'Axes 1 and 5 use a high level for 'Home Enable. Axes 3 and 4 use the Z 'channel and home enable.`
`DEFINE HOME EDGE CW CW CCW CCW`	`'Axes 1 and 2 use the CW edge of 'home. Axes 3 and 4 use the CCW edge 'of home`
`DEFINE HOME FDIR CCW CCW CW CW`	`'Axes 1 and 2 have CCW selected as 'the final approach direction, Axes 3 'and 4 use CW as the final approach 'direction`
`MOVE HOMECW HOMECCW * *`	`'Axis 1 uses CW as the initial search 'direction, axis 2 uses CCW as the 'initial search direction`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

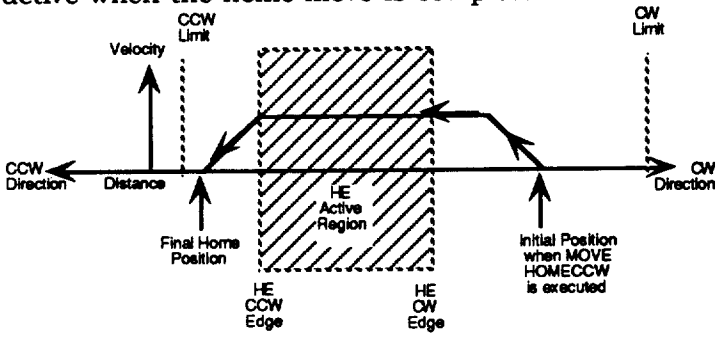
DEFINE HOME BACK

Name	DEFINE HOME BACK
Descriptor	Define Home Backup to Edge
Type	Set-Up
Default	DEFINE HOME BACK * * * *
Syntax	DEFINE HOME BACK YES YES NO *
Options	TAB YES NO NULL
	F1 F2 F3 F4 F5 F6

Description

The **DEFINE HOME BACK** statement determines whether the Model 4000 should back up to the final edge of home, the Home Enable input (**HOME TYPE HE+** or **HE-**) or the combination of the **HE** input and the Z channel (**HOME TYPE HE±Z**). If **DEFINE HOME BACK NO** is issued, the **MOVE HOME** function will not back up to home and the **DEFINE HOME FDIR** setting is ignored. When the active edge is encountered in the homing sequence, the Model 4000 decelerates to a stop and the homing sequence is complete. The final direction of the home sequence is dependent on the initial direction and whether a limit was encountered in the homing sequence. The default value for this statement is: **DEFINE HOME BACK YES YES YES YES**. *The default is to back up to the active final edge.*

The program and diagram below illustrate the use of the **DEFINE HOME** statements. The **DEFINE HOME TYPE HE+** statement is used. The **HE+** implies a normally closed switch is connected (when the load activates the switch the home enable input is at 5V, current not flowing). The active edge of the switch is set to the CCW edge. As you can see, since the axis does not back up and return to the active edge, the home input is no longer active when the home move is complete.

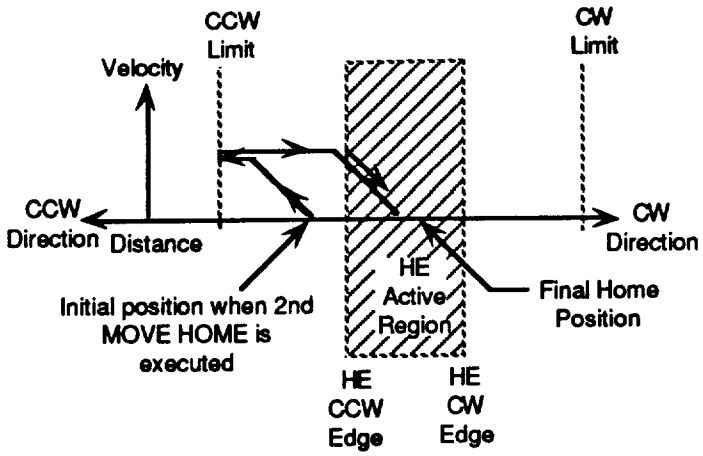


If another **MOVE HOMECCW** statement is executed, the load will reach CCW limit, the direction will be changed, the **MOVE HOME** will continue in the CW direction until the **CCW HE** input edge is found, and the load decelerates to a new final home position. The second **MOVE HOME** sequence is shown below.

```
DEFINE HOME EDGE CCW * * *
DEFINE HOME TYPE HE+ * * *
DEFINE HOME BACK NO * * *
MOVE HOMECCW * * * * *
```

The second sequence is poorly programmed. The resulting final position depends upon which side of the home switch the go home sequence is started. This means repeatable homing will not occur unless care is taken in programming. The load must always be on the same side of the home switch when the **MOVE HOMECCW** statement is executed if **DEFINE HOME BACK** is set to NO.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



See Also: **DEFINE HOME TYPE, DEFINE HOME EDGE, DEFINE HOME TYPE, ACCEL HOME, VEL HOME, MOVE HOMECW/HOMECCW**

DEFINE FAULT

Name	**DEFINE FAULT**
Descriptor	Define Fault Active Level
Type	Set-Up
Default	DEFINE FAULT * * * *
Syntax	DEFINE FAULT HIGH LOW HIGH*
Options	TAB HIGH LOW NULL F1 F2 F3 F4 F5 F6

Description

The **DEFINE FAULT** statement allows the active level of the drive fault input (pin #5 of the Motor Drive connector) to be selected. **HIGH** means that the fault input is active high (no current flowing, 5V). Active high means the Model 4000 detects a fault when the fault input is at 5V. **LOW** means the fault input is active low (current flowing, 0V). Active low means the Model 4000 detects a fault when the fault input is at 0V. *The default fault detection is active high.*

For some Compumotor drives the active level of the fault input must be set low, but for the majority of servo and stepper products the fault level should be high. Refer to the chart below for settings.

Compumotor Product	Drive Fault Level
L, LE, PS7, BLX	Active Low
Compumotor Plus, LN, OEM, S, Z, PK3, and Dynaserv	Active High

See Also: **ON FAULT**

Example

Statement	Description
DEFINE FAULT HIGH LOW HIGH *	'This sets up axes 1 and 3 for 'Compumotor servo drives and axis 2 for 'Digiplan BLX.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DEFINE TERM

Name	**DEFINE TERM**
Descriptor	Define New Termination Character
Type	Set-Up
Default	DEFINE TERM CHAR PORTn = n
Syntax	DEFINE TERM CHAR PORT1 = 8 = AUTOREAD
Options	TAB
	F1 F2 F3 F4 F5 F6

Description

The **DEFINE TERM** statement allows the termination character for Ports 1, 2, or 3 to be changed from the default value of 13 (CR) to any value below 125. The first n selects Ports 1, 2, or 3 and the n following the = sign should be changed to the new termination character value. This value must be a decimal number representing an ASCII character. For instance a decimal value of 32 is the ASCII equivalent to a space character. Refer to the *Appendix* for the ASCII to decimal table.

The termination character tells the Model 4000 to take the contents of its input buffer for the port specified and treat it as a command line. Changing this value allows the Model 4000 to read in data from devices that do not use a **CR** to terminate their data strings (i.e., Compumotor Plus).

See Also: **IN PORT**

Example

Statement	Description
`DEFINE TERM CHAR PORT2 = 13`	`'This defines Port2 termination 'character to be 'ASCII character CR 'whose decimal value is 13.`
`DEFINE TERM CHAR PORT1 = 8`	`'This defines Port1 termination 'character to be ASCII character 'Backspace.`
`IN Q1 = PORT1 ^ 1DPA ^`	`'This allows the Compumotor Plus' 'average current value to be read into 'the Model 4000. A increase in current 'value can indicate tool is excessively 'worn.`

DEFINE VAR

Name	**DEFINE VAR**
Descriptor	Set the Number of Numeric and Text Variables
Type	Set-up
Default	DEFINE VAR NUM = n STR = n n
Syntax	DEFINE VAR NUM = 1000 STR = 20 40
Options	TAB
	F1 F2 F3 F4 F5 F6

Description

The **DEFINE VAR** statement defines how many numeric variables, how many string (alpha-numeric) variables, and the length in bytes of each string variable. This statement configures how the Model 4000's battery backed-up RAM is to be divided between program storage space and variable storage space. A large number of variables will mean less space is available for program storage. There are three n parameters in the default statement. The default values are those in the example below.

- The first sets the number of numeric variables
- The second sets the number of string variables
- The third sets the length of each string variable

There are several factors which determine the maximum value for the first two n parameters. They are illustrated in the equation and chart below. The standard memory has 8,148 bytes of battery backed-up RAM.

The default values in program **POWER_UP** must be changed in order for variables with subscriupts exceeding 109 to be entered. Cycle power or run **POWER_UP** after changes are made.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
8,148 =[(program storage)]
+[(6)*(number of numeric variables +1)]
+[number of string variables)*(length of string variables+1)]
```

The large memory option allows 55,000 bytes to be used for variables and program storage instead of the 8,148 bytes in the small memory version.

	Program lines (approximate)	Maximum number of numeric variables	Maximum number of string variables	Length of each string
Large memory	85	9166	0	0
Small memory	15	1318	0	0

The first string variable number is one larger than the maximum number of numeric variables. The maximum length of each string variable is 40.

See Also: **MATH, IN LCD, IN PORT, IN IN24, IN TRIG, IN BIT**

Example

Statement	Description
`DEFINE VAR NUM = 99 STR = 10 20`	`'This creates 99 numeric 'variables and 10 string 'variables. Each string 'variable has 20 characters. 'The first string variable 'number is 100. The last 'string variable number is '109.`
`MATH Q99 = 987654321.1234`	`'Set last numeric variable.`
`MATH Q100 = ^PRESS F1 TO START MACHINE^`	`'Set first string variable.`

DEFINE ARROW

Name	**DEFINE ARROW**
Descriptor	Reverse Function of Arrow Keys in Jog Mode
Type	Set-Up
Default	DEFINE ARROW REVERSE U/D = Y LR = Y
Syntax	DEFINE ARROW REVERSE U/D = Y LR = Y
Options	TAB Y N F1 F2 F3 F4 F5 F6

Description

The **DEFINE ARROW** statement allows either set of the arrow keys to have their functions swapped when in jog mode. This means that if Y is entered for the U/D (up and down arrow keys) then the up arrow will turn the motor in the direction opposite direction than when the N option was entered. The default is the N option. This change is useful when the motor is mounted such that when the right arrow key is pressed the load moves to the left.

See Also: **ENABLE JOG, ENABLE TCH_JOG**

Example

Statement	Description
`DEFINE ARROW REVERSE U/D = Y L/R = Y`	`'Change direction of both the 'up and down and the left and 'right arrow keys`
`ENABLE JOG YES YES NO NO`	`'Only in jog mode are the 'arrow keys reversed, here for 'axis 1 and 2.`
`DEFINE ARROW REVERSE U/D = N L/R = N`	`'Do not change direction of 'the arrow keys.`
`ENABLE JOG NO NO YES YES`	`'Axis 3 and 4 in jog mode do 'not have their directions 'reversed.`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DEFINE ON RET

Name	DEFINE ON RET
Descriptor	Configure how an interrupt returns
Type	Set-Up
Default	DEFINE ON RET *
Syntax	DEFINE ON RET YES
Options	TAB YES NO NULL
	F1 F2 F3 F4 F5 F6

Description

Another statement added is **DEFINE ON RET**. This statement only affects **RETURN** statements that are returning from an **ON GOSUB** (the **RETURN** from an interrupt destination). This statement has no affect on **GOSUB** or **IF GOSUB** statements. This statement tells a **RETURN** statement whether to return to the statement interrupted or the statement following the one interrupted. The default for this statement is **NO**: return to the statement following the one interrupted. This statement remains in effect until it is disabled with the **NO** option or power is cycled. All **TASKS** and programs are affected by it. The statement can be executed within an interrupt service routine. This allows the return destination from that interrupt to be configured by the interrupt.

This statement can be used to return to re-execute a **WAIT** or **IN** statement but probably would not be used to return to a **MOVE** statement (unless the **MODE** was **M_ABS** or **E_ABS** so that no additional move would be commanded).

DEFINE TRIGDB

Name	DEFINE TRIGDB
Descriptor	Define trigger debounce time
Type	Set-Up
Default	ENABLE TRIGDB* * * *
Syntax	ENABLE TRIGDB 80 * Q1 *
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **DEFINE TRIGDB** statement defines the the total debounce time for the four trigger inputs. The debounce prevents noise or mechanical switch bounce from causing a false interrupt on the trigger input. A trigger is initially recognized on the rising edge of the input. That trigger will not be recognized again until it has gone low again and the debounce time, measured from the rising edge, has been exceeded. This debounce time affects registration and all of the **FOL** and **FOLM** statements which include triggers as a parameter. The initial value of 80 milliseconds will usually be long enough to debounce most mechanical and electronic switches, but this time may be lengthened if needed. In some applications, registration marks or master/slave synchronization marks may occur more frequently than 80 milliseconds. In these cases, the debounce time may be shortened, provided the signal *bounce* is short enough. The debounce times are only accurate to ±2 milliseconds of the specified value, and the actual values used will always be between 4 and 1000 milliseconds. The debounce times are specified for triggers 1, 2, 3, and 4 left to right on the statement line.

See Also: **FOL MOVEWT, FOL NEWCYC, FOL WAIT, FOL M_SYNC, FOL S_SYNC, FOLM DEF**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DISPLAY

Name	DISPLAY
Description	Repetitively Send ASCII Information to LCD
Type	Status
Default	DISPLAY ON LCDN * * * *
Syntax	DISPLAY ON LCD3 STAT STAT STAT STAT

Options						
	TAB	MABS	MINC	EABS	EINC	ETC
	TAB	OFF	VEL	LNUM	STATUS	ETC
	TAB	NULL	VSCL	SPOL		ETC
	TAB	OUT24	IN24	TRACE		ETC
	F1	F2	F3	F4	F5	F6

Description

The **DISPLAY** statement displays data during program execution on the LCD line that you specify. The display is updated every 232 ms as are the axis LEDs on the front panel. Once a **DISPLAY** statement is executed, either from the **IMMED** function or a program, it will remain enabled (while in the **IMMED** function or while executing a program) until the **DISPLAY OFF** option is executed. Cycling power turns off the **DISPLAY** statement, but stopping a program does not. The display will still be active when the next program is run.

The **DISPLAY** statement is temporarily disabled on lines 3 and 4 when in JOG mode, and on lines 2 and 4 when in **IMMED** mode. This allows text related to those modes to be displayed on those lines. It is re-enabled when executing a program.

All of the options, except **IN24**, **OUT24**, and **TRACE**, are axis specific. The four fields of the display statement imply axes 1-4. **OUT24**, **IN24**, and **TRACE** *use the entire line of the LCD, not the 10 characters used by all other options.*

OUT24 Displays (on an entire line) the current state of the 24 programmable output bits.

IN24 Displays (on an entire line) the current state of the 24 programmable input bits.

TRACE Displays (on an entire line) the program statement that is currently executing. This is useful for tracing through a program. However, if the program was not started with the RUN function and the STEP option, many statements will be run quickly and will not be displayed on the LCD. Only statements that take a long time (e.g., WAIT, IN LCD) will always be seen.

STATUS Displays the status of an axis: limits, moving, etc. Each axis field is 10 spaces, so a 40-space LCD line could be: **NOTMOVING CWLIMIT CCWLIMIT STALLED**. Or it could be any one of the following with priority given by the order stated. **WAITDBND STALLED LIMITS CWLIMIT CCWLIMIT MOVING NOTMOVING**

VEL Displays the current velocity of an axis. The display value will be scaled by the velocity scale factor (**UNIT VEL**).

MINC Displays the current motor incremental position of an axis. The entire time in the jog move is treated as one incremental move. The display value will be scaled by the distance scale factor (**UNIT POS**).

MABS Displays the current motor absolute position of an axis. The display value will be scaled by the distance scale factor (**UNIT POS**).

EABS Displays the current encoder absolute position of an axis. The display value will be scaled by the distance scale factor (**UNIT POS**).

EINC Displays the current encoder incremental position of an axis. The display value will be scaled by the distance scale factor (**UNIT POS**).

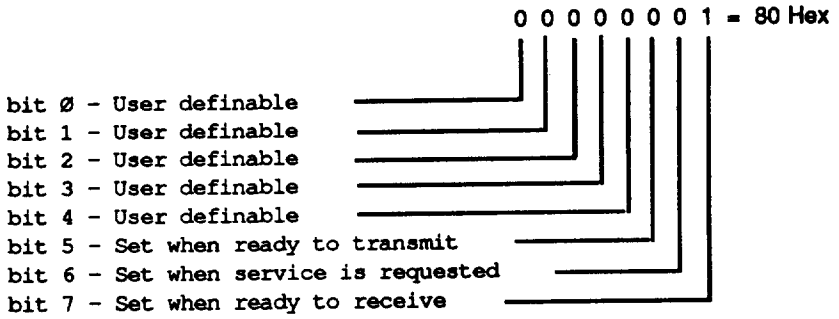
NULL Does not change the previous **DISPLAY** statements value for this position.

LNUM Displays on the axis that you specify, the line # of the program statement that is currently executing.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

OFF Stops updating the LCD on the line and axis area you specify. This should be done in the field before an **OUT LCD** or **IN LCD** statement can be executed. Otherwise, the output data or the input data will be over-written by the display function. This may result in an execution error with the IN LCD statement. Refer to the **OUT** statement for more information on displaying variables and alphanumeric characters.

SPOL Displays the 8 bits of the **GPIB** (optional) serial poll register. MSB (bit 7) is left most bit. The data byte below is a common value for the serial poll register.



VSCL Display in the axis area you specify, the velocity scaling factor in percent. *The **VSCL** is not axis specific. It scales the velocity of all four axes.*

You can use more than one line for displaying. For example, you can use line 1 to display 24 the inputs status, line 2 to display the 24 outputs status, and line 3 to display **MINC** (Motor Position) and **EINC** (Encoder Position) and the status of axes #3 and #4.

The **DISPLAY** statement decreases in performance of the **ON** and **SLIMIT** statements. The **DISPLAY** statement also increases the time between statement execution by up to 150 ms.

See Also: **OUT LCD, IN LCD**

Example

Statement	Description
`DISPLAY ON LCD1 IN24`	`'Displays status of the 24 'inputs on line 1`
`DISPLAY ON LCD2 OUT24`	`'Displays status of the 24 'outputs on line 2`
`DISPLAY ON LCD3 EINC MINC STATUS STATUS`	`'Displays the encoder 'incremental position of 'Axis 1, the motor 'incremental position of 'axis 2, and the status of 'axes 3 and 4 all on line 3`
`DISPLAY ON LCD4 OFF OFF * *`	`'Turn off all display 'functions for axes 1 and 2 'on line 4 leaving axes 3 'and 4 unchanged on the same 'line`

DONE

Name	**DONE**
Descriptor	Stop Program Execution
Type	Programming
Default	DONE
Syntax	DONE
Options	N/A

F1 F2 F3 F4 F5 F6

Description

This statement is placed anywhere in a program to finish or terminate a program. More than one **DONE** statement can be placed in the Model 4000's memory. When the **DONE** statement is encountered, the Model 4000 waits for all statements currently executing to finish, the Model 4000 clears the **LOOP** counter, **GOSUB** calls, continuous moves are stopped.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

disables **ON** statement checking, and disables the **DISPLAY** statement. The **DISPLAY** statement will be re-enabled if a program is started or if the **IMMED** function is entered. You must press the **RUN** function key (and then **ENTER**) or the **START** or **RESUME** keys to run a program.

The **ON STOPK** statement does not execute when a **DONE** statement is encountered, only when you press **STOPK**.

Example

Statement	Description
MOVE 1000 2000 * *	'Moves axes 1 and 2
DONE	'Ends program

ENABLE

Name	**ENABLE**
Descriptor	Enable or Disable System Functions
Type	Set-Up
Default	ENABLE
Syntax	ENABLE ON YES

Options

TAB	ON	NDBWAIT	STALL	POSM	ETC
TAB	MOVKIL	HLIMIT	M_SLIM	E_SLIM	ETC
TAB	JOG	JOY	TCH_JOG	TCH_JOY	ETC
TAB	VSCALE	REG	MOVOUT	PORT	ETC
TAB	DIR				ETC
F1	**F2**	**F3**	**F4**	**F5**	**F6**

Description

The **ENABLE** statement enables or disables many of the Model 4000's statements. The following options are available for the **ENABLE** statement.

ON	On Condition Functions
NDBWAIT	If not within deadband, wait function
STALL	Stall detection function
MOVKIL	Immediate kill move if stall detected
HLIMIT	Hardware end-of-travel limit function
E_SLIM	Software end-of-travel limit function for encoder positions
POSM	Position Maintenance function
M_SLIM	Software end-of-travel limit function for motor position
TCH_JOG	Begin Teach mode for following move statements with the Jog function
JOG	Enter Jog mode and wait for **MENU MODE** to be pressed
JOY	Enter joystick mode and then wait for joystick release input to become active
TCH_JOY	Begin Teach mode for following move statements using the joystick
VSCALE	Enter Velocity Scaling mode
REG	Allow trigger inputs to cause a registration move
MOVOUT	Causes an output to become active whenever there is motion on an axis.
PORT	Decreases program execution time by disabling unused ports
DIR REV	Reverses direction of all commanded motion

For the functions listed above, you may select one of the following options (except JOG and JOY, which do not have a NULL option):

YES	Enables this function
NO	Disables the function
NULL	No change from current setup

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENABLE ON

Name	**ENABLE ON**
Descriptor	Enable or Disable All ON Statements
Type	Set-Up
Default	ENABLE ON NO
Syntax	ENABLE ON NO
Options	TAB YES NO
	F1 F2 F3 F4 F5 F6

Description

The **ENABLE ON** statement enables or disables the **ON** condition statement. Once disabled, all **ON** conditions are no longer checked. If re-enabled, the Model 4000 will again check for **ON** conditions. *The default for this statement is YES (ON). The* ***ENABLE ON*** *statement is not axis specific.* Program execution is faster when **ON** conditions are disabled.

Individual **ON** statements can be disabled by executing that **ON** statement with the word **DISABLE** substituted for **GOTO LABELØ**. Refer to the **ON** statement.

See Also: **ON**

Example

Statement	Description
ENABLE ON YES	'Enable ON condition function

ENABLE NDBWAIT

Name	**ENABLE NDBWAIT**
Descriptor	Wait Until Within Deadband
Type	Set-Up
Default	ENABLE NDBWAIT * * * *
Syntax	ENABLE NDBWAIT YES NO * *
Options	TAB YES NO NULL
	F1 F2 F3 F4 F5 F6

Description

The **ENABLE NDBWAIT** statement causes program execution to wait until the axes specified are within the dead band range after each move before executing the next command. This is only active if the Position Maintenance (**POSM**) statement is enabled. This function is only meaningful when the motor and encoder resolutions have been defined with **ENC MRES** and **ENCO ERES**, the deadband range has been defined with **ENCO DBND**, and the commanded move is in mode **E_INC** or **E_ABS**. *The default for this statement is OFF.*

See Also: **ENCO DBND, ENABLE POSM, POSM GAIN, POSM MAXVEL**

Example

Statement	Description
ENABLE POSM YES YES * *	'Enable position maintenance on 'axes 1 and 2
ENABLE NDBWAIT YES NO * *	'Each move on axis 1 must be 'within deadband before 'subsequential command are 'executed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENABLE STALL

Name	**ENABLE STALL**
Descriptor	Enable or Disable Stall Detection
Type	Set-Up
Default	ENABLE STALL * * * *
Syntax	ENABLE STALL YES NO * *
Options	TAB YES NO NULL
	F1 F2 F3 F4 F5 F6

Description

The **ENABLE STALL** statement enables the Stall Detection statement. *The default for this statement is NO (OFF).* A motor stall can be detected when the Model 4000 uses encoder feedback. If stall detection is enabled, the Model 4000 will monitor the encoder input as it is executing the move.

The controller does not check for a stall at the beginning of a new move until after the number of steps specified by **ENCO BLK**. A stall will be detected if after four full motor steps (not micro steps) sent to the drive, at least half of the expected number of encoder steps are not received.

The **ENCO BKL** statement sets the number of encoder steps of backlash allowed. Also, the 25,000 step Compumotor stepper motor has 400 full motor steps. If **ENABLE MOVKIL** is enabled, the stalled move will be aborted.

You must execute **ENCO MRES**, and **ENCO ERES**, **MODE E_ABS** or **E_INC** before this function will execute. An axis does *not* need to be in encoder mode to have stall detect enabled.

See Also: **ENCO BKL, ENCO MRES, ENCO ERES, ON STALL**

Example

Statement	Description
`ENCO BKL 200 200 * *`	`'Specify backlash as 200 motor 'steps for axes 1 & 2`
`ENABLE STALL YES YES * *`	`'Enable stall detect function on 'axes 1 & 2`

ENABLE POSM

Name	**ENABLE POSM**
Descriptor	Enable or Disable Position Maintenance
Type	Set-Up
Default	ENABLE POSM * * * *
Syntax	ENABLE POSM YES NO NO *
Options	TAB YES NO NULL
	F1 F2 F3 F4 F5 F6

Description

The **ENABLE POSM** statement enables the Position Maintenance function. *The default for this statement is NO (OFF).*

If you enable position maintenance, the indexer will consider the motor to be In-Position only if the encoder is positioned within the deadband region. If the load is somehow moved outside the deadband region, the indexer will use the encoder signal as a reference and bring the motor back into position.

If you disable Position Maintenance, with **ENABLE POSM NO**, the controller will not try and position to within the deadband region. You must execute the following statements before you run the **ENABLE POSM** statement.

- **MODE E_INC or E_ABS**
- **ENCO MRES or ENCO ERES**
- **ENCO DBND (optional)**

The **MODE E_INC** or **E_ABS** statement must be executed for the **ENABLE POSM** statement to be in effect.

See Also: **ENCO DBND, ENCO MRES, ENCO ERES, MODE, ENABLE NDBWAIT**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example

Statement	Description
`ENABLE POSM YES YES * *`	'Enable position maintenance on 'axes 1 & 2

ENABLE MOVKIL

Name	**ENABLE MOVKIL**
Descriptor	Enables Kill-On-Stall Function
Type	Set-Up
Default	ENABLE MOVKIL * * * *
Syntax	ENABLE MOVKIL YES YES NO NO
Options	TAB YES NO NULL F1 F2 F3 F4 F5 F6

Description

The **ENABLE MOVKIL** statement shuts off the transmission of pulses when a Stall is detected. *The default for this statement is OFF (NO).* The following statements must be executed before the **ENABLE MOVKIL** statement is executed.

- **ENCO MRES** or **ENCO ERES**
- **ENCO EKL** (optional)
- **ENABLE STALL YES**

See Also: **ENCO MRES**, **ENCO ERES**, **ENABLE STALL**, **ON STALL**

Example

Statement	Description
`ENABLE MOVKIL YES YES NO *`	'Enable kill on Stall for axes 1 '& 2. Disable 'on axis 3 and 'leave axis 4 as last setting

ENABLE HLIMIT

Name	**ENABLE HLIMIT**
Descriptor	Enable or Disable Hardware Limit Input Checking
Type	Set-Up
Default	ENABLE HLIMIT * * * *
Syntax	ENABLE HLIMIT YES NO YES YES
Options	TAB YES NO NULL F1 F2 F3 F4 F5 F6

Description

The **ENABLE HLIMIT** statement enables the Hardware Limit Detection statement. *The default for this statement is YES (ON).*

If you do not have the physical limit switches installed and connected to the Model 4000, you must disable the Hardware Limit to allow the motors to move. You can do this with the **ENABLE HLIMIT NO** statement or by jumpering the input to ground.

ENABLE HLIMIT NO	Disables the Hardware Limit inputs
ENABLE HLIMIT YES	Enables the Hardware Limit inputs
ENABLE HLIMIT *	Leave the setting as it is

See Also: **DECEL HLIMIT**, **ON HLIMIT**

Example

Statement	Description
`ENABLE HLIMIT NO NO * *`	'Disable Hardware Limit Input 'checking on axes 1 & 2
`MOVE 20000 25000 * *`	'Move axes 1 & 2

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENABLE M_SLIM

Name	**ENABLE M_SLIM**
Descriptor	Enable/Disable Motor Position Limit Checking
Type	Set-Up
Default	ENABLE M_SLIM * * * *
Syntax	ENABLE M_SLIM YES NO NO YES
Options	TAB YES NO NULL F1 F2 F3 F4 F5 F6

Description

The **ENABLE M_SLIM** statement enables the motor step position limit detection statement. *The default for this statement is NO (OFF).* All position limits (except hardware limit input) are defined by the **SLIMIT** statement, and have a default value of zero. For this reason, **SLIMIT** statements should be executed before executing an **ENABLE M_SLIM YES** or **E_SLIM YES** statement. The **SLIMIT** statement enables **SLIMIT** checking when it executed.

See Also: **SLIMIT, DECEL SLIMIT, ON SLIMIT, ENABLE E_SLIM, ENABLE HLIMIT**

Example

See **SLIMIT** (Software Limit) command for instructions on setting the values of the software limit.

For safety purposes, Compumotor recommends that the Hardware limit switches be used. Software limit switches may be used in conjunction with hardware limit switches.

Statement	Description
ENABLE M_SLIM YES NO * *	'Enable Motor Step software 'limit on axis 1. Disable 'Software limit on axis 2, axes '3 & 4 unchanged

ENABLE E_SLIM

Name	**ENABLE E_SLIM**
Descriptor	Enable/Disable Encoder Position Limit Checking
Type	Set-Up
Default	ENABLE E_SLIM * * * *
Syntax	ENABLE E SLIM YES NO NO YES
Options	TAB YES NO NULL F1 F2 F3 F4 F5 F6

Description

The **ENABLE E_SLIM** statement enables the encoder step software limit detection statement. *The default for this statement is OFF (NO).* The default value for this statement is Ø. If you execute an **ENABLE M_SLIM** statement without previously defining the **SLIMIT** values, an execution error is likely to occur. The **SLIMIT** statement when executed will re-enable **SLIMIT** checking (if previously disabled by the **ENABLE M_SLIM NO**).

See Also: **SLIMIT, DECEL SLIMIT, ON SLIMIT, ENABLE M_SLIM, ENABLE HLIMIT**

Example

See **SLIMIT** (Software Limit) statement for instructions on setting the values of the software limit.

For safety purposes, Compumotor recommends that the Hardware limit switches be used. Software limit switches may be used in conjunction with hardware limit switches.

Statement	Description
ENABLE E_SLIM YES NO * *	'Enable Encoder Software limit 'on axis 1. Disable Software 'limit on axis 2, axes 3 and 4 'unchanged

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENABLE JOG

Name	ENABLE JOG
Descriptor	Enter/Exit Jog Mode
Type	Set-Up
Default	ENABLE JOG * * * *
Syntax	ENABLE JOG YES YES NO
Options	TAB YES NO NULL
	F1 F2 F3 F4 F5 F6

Description

The **ENABLE JOG** statement places the Model 4000 into Jog mode (just as if the **JOG MODE** key were pressed, when not running a program). The left and right arrow keys will move the axis whose number that appears above **F1**. The left arrow moves that axis CCW, the right arrow moves that axis CW. The up and down arrow keys will move the axis whose number appears above the **F4** key. The down arrow moves that axis CCW, the up arrow moves that axis CW.

You can change the axis that corresponds to a given set of arrow keys. Press **F1** to change the left/right arrow keys or press **F4** to change the up/down arrow keys. Once **F1** or **F4** have been pressed, press the new axis number on the numeric key pad and it will appear above the **F1** or **F4** key you have selected.

You can change the velocity that corresponds to a given set of arrow keys. Press **F2** or **F3** to change to the low or high velocities associated with the axis above **F1**. The asterisk is above **F2** or **F3** to indicate whether low or high velocity is selected.

To change the value of the low or high velocity associated with the axis above **F1** or **F4**:

① Press the function key that corresponds to the velocity type (LOW or HIGH) that you wish to change.

② Use the numeric keypad to enter the new velocity value.

③ Use **F3**, **F5**, or **F6** to move the cursor to the other velocity values and use the numeric key pad to change their value.

④ Press the **ENTER** key to enter the new velocities. To change an axis, you do not need to press **ENTER**, but to change a velocity you do. Upon changing an axis, you will notice the velocities for that axis displayed to the right of the axis number.

The **JOG LO** and **HI** velocities can also be set using the **VEL JOGL** and **VEL JOGH** statements.

When the **ENABLE JOG** statement is executed, any **MOVE** statements that were in progress are immediately killed (pulse shut-off) for the axes enabled in **JOG MODE**.

When in **JOG** mode, the beeper will sound only once when a key is pressed and held down, it will not beep repeatedly as in all other modes.

To exit **JOG** mode, press the **MENU MODE** key. An **ON** statement that becomes true while in **JOG** mode will terminate **JOG MODE** and execution will begin at the destination of the **ON** command.

ON F-KEY statements are temporarily disabled while in **JOG** mode. The **ACCEL JOG** statement determines the acceleration used in **JOG** mode, and the **DECEL JOG** statement determines the deceleration used in **JOG** mode. Cycling power will set the **DECEL** and **ACCEL JOG** values to their default values, it is recommended to place these statements in the program before **JOG MODE** is enabled.

The default **JOG LO** velocity is 25,000 steps/sec^2. The default for **JOG HI** velocity is 75,000 steps/sec^2.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

See Also: **VEL JOGL, JOG VELH, ACCEL JOG, DECEL JOG, ENABLE TCH_JOG, ON**

Example

Statement	Description
MOVE 10000 20000 * *	'Move axes 1 and 2
ENABLE JOG YES YES NO NO	'Enter JOG mode

ENABLE JOY

Name	**ENABLE JOY**
Descriptor	Enter/Exit Joystick Mode
Type	Set-Up
Default	ENABLE JOY * * * *
Syntax	ENABLE JOY YES YES NO YES
Options	TAB YES NO NULL
	F1 F2 F3 F4 F5 F6

Description

The **ENABLE JOY** statement enters joystick mode and suspends program execution until the joystick release input becomes active (5V, current not flowing). The **DEFINE JOYSL** and **DEFINE JOYSH** statements determine which analog inputs control which axis.

The joystick release input is on pin #8 of the auxiliary connector #2. The switch connected to this input should be a normally closed momentary switch. A one-second delay occurs when the joystick release input is activated. This allows time for the switch to become inactive so that the next time the joystick is activated it will not immediately exit. If this time is not long enough, a **WAIT** statement can be placed before the next statement using the joystick.

When joystick mode is enabled, the joystick voltage must be within deadband (or changed to be within deadband) before motion will occur. For instance, if the joystick is at full deflection when joystick is enabled, it must be moved to within deadband before motion will be allowed.

See Also: **DEFINE JOYSL, DEFINE JOYSH, VEL JOYL, VEL JOYH, DEFINE CTR_DB, DEFINE END_DB, ACCEL JOY**

ENABLE TCH_JOY

Name	**ENABLE TCH_JOY**
Descriptor	Enter/Exit Teach Mode Using Joystick
Type	Set-Up
Default	ENABLE TCH_JOY * * * *
Syntax	ENABLE TCH_JOY YES YES * YES
Options	TAB YES NO NULL
	F1 F2 F3 F4 F5 F6

Description

The **ENABLE TCH_JOY** statement causes the Model 4000 to learn distance parameters for all **MOVE** and **MOVI** statements that follow the **ENABLE TCH_JOY** statement. Only the axes in a **MOVE** statement that have teach joy enabled (**ENABLE TCH_JOY YES**) will have their distances learned. To disable Teach mode the **ENABLE TCH_JOY NO** statement should be executed. Teach mode must be disabled before the learned **MOVES** can be executed as normal **MOVE** statements.

Teach mode only learns a position when the following three conditions occur (in order):

① Teach mode is enabled by executing the **ENABLE TCH_JOY YES** function for the axis desired.

② A **MOVE** or **MOVI** statement is executed and that statement must have a constant or variable entry (**Q1 - Qnn**).

③ The joy release input is toggled or an **ON** statement becomes true, at which time the positions that the axes are to learn are stored in the **MOVE** or **MOVI**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

statement. Each position taught requires the user to toggle the Joy release switch once, regardless of the number of participating axes.

If **TCH_JOY** has been enabled for an axis, a **MOVE** statement is executed, and the **MOVE** statement's parameter for the axis is not a constant or a variable, then:

① The joystick(s) allows that axis to be moved

② The position of that axis is not stored when the joy release input is toggled.

This allows the joystick to simulate a **MOVE HOME** or **MOVE SLEW** statement. If **TCH_JOY** mode is not enabled on an axis, the **MOVE** parameter is executed as normal.

The type of position stored is determined by the **MODE** statement (i.e., Motor incremental or absolute, or Encoder incremental or absolute).

If an **ON** statement becomes true while in teach mode, then the current Teach mode is exited, the axis positions at the time the **ON** statement becomes true are stored in the **MOVE** statement, just as if the joy release inputs were toggled. Then the **ON** statement branch is taken.

The joystick release input is on pin #8 of auxiliary connector #2. The switch connected to this input should be a normally closed momentary switch. A one-second delay occurs when the joystick release input is activated. This allows time for the switch to become inactive so that the next time the joystick is activated it will not immediately exit. If this time is not long enough, a **WAIT** statement can be placed before the next statement using the joystick.

See Also: **DEFINE JOYSL**, **DEFINE JOYSH**, **VEL JOYL**, **VEL JOYH**, **DEFINE CTR_DB**, **DEFINE END_DB**, **ACCEL JOY**

ENABLE TCH_JOG

Name	**ENABLE TCH_JOG**
Descriptor	Enter/Exit Teach Mode Using Jog
Type	Set-Up
Default	ENABLE TCH_JOG * * * *
Syntax	ENABLE TCH_JOG YES YES * YES
Options	TAB YES NO NULL F1 F2 F3 F4 F5 F6

Description

The **ENABLE TCH_JOG** statement *teaches* the Model 4000 distance parameters for all **MOVE** or **MOVI** statements that follow the **ENABLE TCH_JOG** statement. Only the axis that has **ENABLE TCH_JOG YES** will have its distance taught. To disable distance learning the **ENABLE TCH_JOG NO** statement should be executed. **TCH_JOG** must be disabled before the learned moves can be executed as normal **MOVE** statements.

Teach mode only learns a position when the following three conditions occur in order.

① Teach mode is enabled with the **ENABLE TCH_JOG YES** function for the axis desired.

② A **MOVE** or **MOVI** statement is executed. The **MOVE** or **MOVI** statement must have a constant or a variable (**Q1 - QNN**) for each axis to be taught.

③ The menu mode key is pressed or an **ON** statement becomes true.

That sequence will cause the positions of the axis to be stored in the **MOVE** or **MOVI** statement (if a constant was used) or in the **Q** variable (if a variable was used).

If **TCH_JOG** mode is enabled, and the **MOVE** parameter for that axis is not a constant or a variable, then the **MOVE** parameter is not executed as normal, the keys allow that axis to be jogged, and the position of the axis

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

when **MENU MODE** is pressed is not stored. This allows the jog function to simulate a **MOVE SLEW** or **MOVE HOME**.

If **TCH_JOG** mode is not enabled on an axis, the **MOVE** parameter is executed as normal. The type of position stored is determined by the **MODE** statement (i.e. Motor incremental or absolute, or Encoder incremental or absolute).

If an **ON** statement becomes true while in teach mode, then the current **JOG** mode is terminated. The axes positions at the time the **ON** statement becomes true are stored in the **MOVE** statement, just as if **MENU MODE** were pressed. Then the **ON** statement branch is taken.

To make an entry into teach mode selectable, you may want to use a program section similar to that shown below.

Enable Tch_Jog: Example 1

```
Statement                          Description
MATH Q1 = 0                        'Q1 is a flag, 0=not in teach mode 1=in teach mode
ENABLE HLIMIT NO NO NO NO          'Disable limit inputs (unless you have them wired)
LABEL GET_KEY                      'Beginning of loop
OUT LCD3,01 ^TEACH DO^             'Create message on LCD display
OUT LCD4,01 ^MOVES MOVE^
ON F-KEY1 GOTO TEACH               'Allow user to select jog mode to teach
ON F-KEY2 GOTO DO_MOVES            'New positions for the move
LABEL WAITL                        'Wait here forever or until a key is pressed
GOTO WAITL
LABEL TEACH                        'Start here for teach mode
ENABLE TCH_JOG YES YES YES YES     'Enable teach mode using front panel arrow keys
MATH Q1 = 1                        'Set flag to be in teach mode
LABEL DO_MOVES
MOVE 0 0 0 0                       'These are the move statements
MOVE 0 0 0 0                       'that are modified when in teach
•                                  'mode
•
•
MOVE 0 0 0 0                       'If not in teach mode, go back
MOVE 0 0 0 0                       'and wait for another key
IF Q1 = 0 GOTO GET_KEY             'If in teach mode, take out of teach mode
ENABLE TCH_JOG NO NO NO NO
MATH Q1 = 0                        'Set flag to not in teach mode status
GOTO GET_KEY                       'Go back and wait for another key
```

Example 2

```
Statement                          Description
ENABLE TCH_JOG YES YES YES YES     'Activated teach mode
MOVE  0  Q1  Q2  *                 'When the MOVE statement is executed, the moves
                                   'are not made, instead JOG MODE is entered.  You
                                   'should jog the axes to the desired location and
                                   'then press the MENU MODE key.  Upon pressing the
                                   'MENU MODE key, the MOVE statement will have the
                                   'position for axis 1 stored as the first
                                   'parameter, and axes 2 and 3 positions are stored
                                   'as the contents of Q variables 1 and 2.
ENABLE TCH_JOG NO NO NO NO
```

Example 3

The following example is more advanced in that it uses variables and pointer variables to store the positions. Another difference is that this program uses the positions stored in variables in a **PATH LINE** statement (thus the motion when executed does not stop when each stored position is reached). This program also allows the user to change the velocity of the executable version during the teach mode, before each new position is stored. Care should be taken to ensure the endpoints do not create acute angles so that motion on one of the axes would have to decelerate or accelerate very quickly.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following 10 lines should be placed in the POWER_UP program.

Statement	Description
`DEFINE VAR NUM = 710 STR = 9 20`	`'Allows this program to use variable 'numbers 100 thru 710 as pointers, counters 'and storage. Variables 110 thru 710 'provide 200 endpoints of X- position data 'and 200 different velocities.`
`UNIT PATH POS 25000`	`'Define units to be inches.`
`UNIT POS 25000 25000 * *`	`'Define units to be inches.`
`UNIT VEL 25000 25000 * *`	`'Define units to be inches.`
`UNIT PATH VEL 25000`	`'Define units to be inches.`
`VEL JOGL 2 2 * *`	`'Define jog low velocity to be 2 inches per 'second.`
`VEL JOGH 3 3 * *`	`'Define jog high velocity to be 3 inches 'per second.`
`VEL PATH 2`	`'Define path velocity to be 2 inches per 'second.`
`MODE M_INC M_INC * *`	`'Set mode to motor incremental causes the 'positions stored to be incremental 'distances.`
`ENABLE HLIMIT NO NO * *`	`'Disable the limit inputs for demo 'purposes.`
`MATH Q1 = 0`	`'Variable Q1 is used as a flag to tell 'which mode we are in Q1 = 0 means we 'should exit teach mode, and return to the 'main menu. Q1 = 1 means we should enter 'teach mode. Q1 = 2 means we should enter 'a new velocity.`
`LABEL GET_KEY`	`'Main setup location, return here after 'teach or execute.`
`OUT LCD1,1 BLANK`	`'Send prompt messages to the LCD.`
`OUT LCD3,1 ^ TEACH DO ^`	
`OUT LCD4,1 ^MOTION MOTION ^`	
`ON F-KEY1 GOTO TEACH1`	`'Enable interrupt when F-KEY 1 is pressed.`
`ON F-KEY2 GOTO DO_MOVES`	
`LABEL WAITKEY`	`'Wait in this infinite loop for the user to 'press F1 or F2.`
`GOTO WAITKEY`	
`'`	
`'`	
`LABEL TEACH1`	`'The program branches here when the F1 key 'is pressed.`
`GOSUB CLEARVAR`	`'Before storing the new positions, first 'clear the old ones.`
`ON START GOSUB VELCHNG`	`'Allow the JOG mode to be aborted if the 'start key is pressed.`
`ENABLE TCH_JOG YES YES * *`	`'When the MOVE statement is encountered, go 'into JOG mode.`
`ON MENUREC GOSUB CLEARTCH`	`'Exit JOG mode and teach mode when the MENU 'RECALL key is pressed, return to the main 'menu.`
`GOTO GET_NEWV`	`'Pretend a START key was pressed to force 'the user to tell us the initial velocity 'of the executable move, this is NOT the 'JOG velocity.`
`LABEL DO_TEACH`	`'This is the start of the teach loop.`
`OUT LCD1,1 ^ PRESS MENU RECALL TO EXIT TEACH^`	`'Send a menu to the LCD.`
`OUT LCD2,1 BLANK`	
`OUT LCD2,1 ^TO CHANGE PATH VEL PRESS START KEY^`	
`MOVE QQ100 QQ101 * *`	`'When this statement executes, the 4000 'will go into jog mode so that the new 'position can be learned.`
`IF Q1 = 0 GOTO DONETCH`	`'Check if MENU RECALL was pressed, if so 'jump: exit teach mode.`
`IF Q1 = 2 GOTO GET_NEWV`	`'Check if START key was pressed, if so 'jump: get new velocity.`
`MATH Q100 = Q100 + 3`	`'Increment pointer variables to the next 'free locations.`
`MATH Q101 = Q101 + 3`	
`MATH Q102 = Q102 + 3`	
`MATH Q104 = Q104 + 1`	`'Increment counter of the number of points 'stored so far.`
`GOTO DO_TEACH`	`'Begin loop again.`
`LABEL DONETCH`	`'Here when MENU RECALL key pressed, exit 'teach mode.`
`ENABLE TCH_JOG NO NO * *`	`'Disable teach mode.`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
MATH Q1 = 0                                          'Clear flag that we are in teach mode.
OUT LCD2,1 ^TEACHING OF MOVES IS COMPLETE  ^         'Send message to LCD.
GOTO GET_KEY                                         'Return to main loop.
LABEL DO_MOVES                                       'Here to execute the taught moves.
OUT LCD1,1 ^ PRESS MENU RECALL TO ABORT MOVES^       'Display options on LCD.
ON MENUREC GOTO EXITMOV                              'Enable interrupt to abort moves.
OUT LCD2,1 BLANK
MATH Q100 = 110                                      'Reset pointers to beginning position
                                                     'endpoint.
MATH Q101 = 111
MATH Q102 = 112
IF Q104 = 0 GOTO GET_KEY                             'Check counter of endpoints to be sure we
                                                     'have some.
PATH DEF TEACH 1 2                                   'Start path definition.
PATH XY MINC                                         'Assume the endpoints are incremental
                                                     'distances.
LOOP Q104                                            'Compile and execute the number of
                                                     'endpoints stored.
IF QQ102 = 0 GOTO NO_NEWV                            'Check if a new velocity exists for the
                                                     'next segment.
VEL PATH QQ102                                       'If so, execute it.
LABEL NO_NEWV
PATH LINE  QQ100 QQ101                               'Use stored endpoints via the pointer
                                                     'variables. Pointer variable QQ100 will use
                                                     'Q110, Q113, Q116 ... and pointer variable
                                                     'QQ101 will use Q111, Q114 ... for its
                                                     'endpoints.
MATH Q100 = Q100 + 3                                 'Increment the pointer variables to the
                                                     'next endpoint.
MATH Q101 = Q101 + 3
MATH Q102 = Q102 + 3
ENDLOOP                                              'Return to the loop statement for each data
                                                     'point entered.
PATH END                                             'When all the data points have been used,
                                                     'end the path definition.
OUT LCD2,1 ^EXECUTION OF MOVES IS COMPLETE^
GOTO GET_KEY                                         'Return to the main menu.
LABEL CLEARTCH                                       'This interrupt routine sets a flag which
                                                     'the main teach loop checks for.  When the
                                                     'flag is clear, teach mode is exited.
MATH Q1 = 0
RETURN
LABEL VELCHNG                                        'This interrupt routine sets a flag which
                                                     'the main teach loop checks.  If flag is 2,
                                                     'a new velocity is requested.
MATH Q1 = 2
ON START GOSUB VELCHNG                               'Re-enable the interrupt.
RETURN
LABEL GET_NEWV                                       'This routine asks the user for a new
                                                     'velocity to be used when the taught
                                                     'positions are executed.
GOSUB CLEARLCD                                       'Before asking for the vel, clear the LCD.
ON ERROR GOTO ERROR_CK                               'Enable a special error handling routine in
                                                     'case an invalid velocity is entered.
IN QQ102 = LCD1,1 ^Enter the new path vel: ^         'Ask for the velocity and then store it
                                                     'with the endpoint data. QQ102 points to
                                                     'the velocity storage location.
OUT LCD1,1 ^TO CHANGE PATH VEL PRESS START KEY^      'Re-display the options on the LCD.
MATH Q1 = 1                                          'Re-set flag to in teach mode.
GOTO DO_TEACH                                        'Return to main teach loop.
LABEL CLEARLCD                                       'This subroutine clears the entire LCD.
OUT LCD1,1 BLANK TO END OF LINE
OUT LCD2,1 BLANK TO END OF LINE
OUT LCD3,1 BLANK TO END OF LINE
OUT LCD4,1 BLANK TO END OF LINE
RETURN
LABEL EXITMOV                                        'This interrupt routine stops the move
                                                     'being executed.
MOVE STOP STOP STOP STOP
GOTO GET_KEY                                         'Return to the main loop.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
LABEL ERROR_CK                                  'This interrupt routine checks if the
                                                'execution error was due to improper data
                                                'entered into the IN LCD statement, if so
                                                'the velocity is asked for again.  Else,
                                                'the program is terminated and the error
                                                'displayed on the LCD.
IN Q105 = ERROR                                 'Get the error value into variable Q105.
IF Q105 = 28 GOTO DO_TEACH                      'Check if the error is invalid LCD data.
GOSUB CLEARLCD                                  'Here if its not an LCD error.
OUT LCD1,1 ^EXECUTION ERROR = ^ Q105            'Display the error number.
BREAK                                           'Clear any GOSUBs buffered up.
DONE
LABEL CLEARVAR                                  'Here to initialize and clear variables and
                                                'pointer variables before actually entering
                                                'the teach mode loop.
IF Q104 = 0 GOTO VAR_DONE                       'If no points already stored, then there
                                                'are none to clear.
IF Q104 < 101 GOTO VAR_OK                       'If too many points to clear, just clear
                                                'the max. number.
MATH Q104 = 100
LABEL VAR_OK
MATH Q100 = 110                                 'Q110 is the start of the endpoint storage
                                                'space, it goes up to Q710
MATH Q104 = Q104 * 3                            'There are 3 data elements per endpoint (X-
                                                'Y and velocity)
LOOP Q104                                       'Clear all the elements of all the
                                                'endpoints.
MATH QQ100 = 0                                  'Clear the contents of the pointer.
MATH Q100 = Q100 + 1                            'Increment the pointer.
ENDLOOP                                         'Do it again.
LABEL VAR_DONE
MATH Q100 = 110                                 'Pre-set the pointer variables to their
                                                'starting values.
MATH Q101 = 111
MATH Q102 = 112
MATH Q1 = 1                                     'Set the flag to teach mode active.
MATH Q104 = 1                                   'Set the endpoint counter to endpoint.
RETURN
LCD1,1 ^ PRESS MENU RECALL TO ABORT MOVES ^     'Display options on LCD.
ON MENUREC GOTO EXITMOV                         'Enable interrupt to abort moves.
OUT LCD2,1 BLANK
```

ENABLE REGSRV

Name	**ENABLE REGSRV**
Descriptor	Enable registration compensation of servo following error
Type	Set-Up
Default	ENABLE REGSRV * * * *
Syntax	ENABLE REGSRV YES * * NO
Options	TAB YES NULL NO
	F1 F2 F3 F4 F5 F6

Description

The **ENABLE REGSRV** statement enables or disables servo following error compensation in registration moves. When a Model 4000 is controlling a servo drive axis, that axis is usually in Motor Step mode. This allows the drive to close its own position control loop. While in motion, however, there will often be drive following error, i.e., the actual load position may not match the commanded position. In registration applications, it is the actual load motion which triggers the registration input. When a Motor Step mode registration move input occurs, the Model 4000 normally issues the additional commanded steps to the drives, as specified in the registration move definition. If drive following error exists when the input occurs, this error becomes static overshoot after the move finishes and the drive corrects its following error.

This problem may be avoided by using the registration compensation feature of the Model 4000. To use this feature, the actual position of the load (or shaft) must be available to the Model 4000 through the encoder input of that axis, and the **ENCO MRES** and **ENCO ERES** values must be

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

correctly specified. Although the encoder position is not used to *control* the move profile, it is captured along with the motor step position upon receipt of the registration input. If servo following error compensation is enabled (**ENABLE REGSRV YES**), these values are used along with the motor and encoder resolutions to calculate the servo drive following error in terms of motor steps. The registration moves (Motor Step mode only) are adjusted by this error to result in correct positioning of the load after the drive has corrected its following error. The *reference* encoder position is captured when motion begins, so *it important that the load be stable and in position before the next move which results in registration actually starts.*

See Also: **SEG REG, ENABLE REG, MODE, ENCO MRES, ENCO ERES**

ENABLE TRIG REV

Name	**ENABLE TRIG REV**
Descriptor	Enable inversion of active trigger level
Type	Set-Up
Default	ENABLE TRIG REV * * * *
Syntax	ENABLE TRIG REV YES * * NO
Options	TAB YES NULL NO
	F1 F2 F3 F4 F5 F6

Description

The **ENABLE TRIG REV** statement enables or disables the inversion of the active state of the specified trigger input. The power-up default for all four trigger inputs is active high, i.e, a transition from a low to a high TTL level is required for activation of the functions associated with a trigger. These functions may include registration and several functions associated with Following. Refer to the **Model 4000 Options Guide** for a description of these Following functions. If the active level of a trigger is inverted with the **ENABLE TRIG REV YES** statement, a transition from a high to a low TTL level is required for activation of the functions associated with that trigger.

See Also: **SEG REG, ENABLE REG, DEFINE TRIGDB, FOL, FOLM**

ENABLE VSCALE

Name	**ENABLE VSCALE**
Descriptor	Enter/Exit Velocity Scaling Mode
Type	Set-Up
Default	ENABLE VSCALE NO
Syntax	ENABLE VSCALE NO
Options	TAB YES NO
	F1 F2 F3 F4 F5 F6

Description

The **ENABLE VSCALE** statement allows velocity scaling to used in some parts of the program and not in others. When velocity scaling is enabled, the state of the joystick select input and the definitions for **VSCL_H, VSCL_L** will determine the source of the velocity scaling percentage.

When velocity scaling is enabled, the velocity on all axes will be continuously scaled by the percentage defined by the **DEFINE VSCALE** statements. The **VSCALE** update rate is once every 2 ms.

Velocity scaling can be enabled and disabled on the fly. The Model 4000 will use **ACCEL VSCALE** to move to the appropriate velocity in either event.

When velocity scaling is not enabled, the velocity on all axes is just as specified by the **VEL** statements.

See Also: **DEFINE VSCL_L, DEFINE VSCL_H, ACCEL VSCALE**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example

Statement	Description
`DEFINE VSCL_L SOURCE = CH4`	`'Use analog channel four for the 'velocity scale input when 'joystick select is low.`
`DEFINE VSCL_H SOURCE = Q1`	`'Use Q1 as a velocity scale 'percentage specification when 'joystick select is high.`
`ACCEL VSCALE 1000`	`'Max vscale accel 1000% per sec.`
`ENABLE VSCALE YES`	`'Enable velocity scaling`
`MOVE 100000 * * *`	`'Move on axis 1 with scaled 'motion`
`ENABLE VSCALE NO`	`'Disable velocity scaling`

ENABLE REGn

Name	**ENABLE REGn**
Descriptor	Enable Registration Moves
Type	Set-Up
Default	ENABLE REGn * * * *
Syntax	ENABLE REG1 YES YES NO NO
Options	TAB YES NO NULL
	F1 F2 F3 F4 F5 F6

Description

The **ENABLE REG** statement allows the trigger inputs to cause a registration move to occur. The registration move interrupts whatever current move was in progress and performs the registration move. The registration move is defined by the **SEG REG** statement. You must define the **SEG REGn** statement before you define the **ENABLE REGn** statement.

The value of n can be from 1 to 4 and defines which trigger input is enabled to cause registration moves. The four **YES**, **NO**, or **NULL** parameters enable, disable or leave unchanged the specified axis. When not enabled, the trigger inputs will not cause a registration move even if the inputs are active.

The default value is **ENABLE REG NO NO NO NO** for all four trigger inputs, this means that registration moves are disabled. Refer to *Chapter 5, Application Design*, for an example registration move.

See Also: **SEG REG**

Example

Statement	Description
`SEG REG1 25000 50000 * *`	`'Define a registration move of '25000 steps on axis 1 and 50000 'steps on axis 2 both of which 'occur when trigger input #1 is 'active (if enabled)`
`ENABLE REG1 YES YES NO NO`	`'Enable trigger input 1 to cause 'registration moves on axes #1 'and #2.`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENABLE MOVOUT

Name	**ENABLE MOVOUT**
Descriptor	Enable Output for Moving/Not Moving
Type	Set-Up
Default	ENABLE MOVOUT AXISn BITn NO
Syntax	ENABLE MOVOUT AXIS2 BIT17 YES
Options	TAB YES NO
	F1 F2 F3 F4 F5 F6

Description

The **ENABLE MOVOUT** statement automatically activates the specified output number, whenever that axis number is in motion. The active level of the output is configured using the **DEFINE OUT24** statement. The output is deactivated as soon as motion stops. There is a time delay of 3 to 12 ms from the start of motion to the activation of the output. Only one output can be activated per axis, with a possible total of four (one per axis). The outputs are open collector and may be wired together, if one wire is to be active due to motion on any one of several axes (only for negative logic). The default is not output active.

If outputs set up as **MOVOUT**'s are commanded to change state with other output statements, the outputs will temporarily reflect the new commanded status and then change back to reflect the axis movement status.

Example

Statement	Description
`ENABLE MOVOUT AXIS2 BIT19 YES`	`'Bit 19 will become active 'whenever there is motion on 'axis 2`
`ENABLE MOVOUT AXIS3 BIT20 YES`	`'Bit 20 will become active 'whenever there is motion on 'axis 3`
`MOVE SLEWCW SLEWCCW SLEWCW STOP`	`'Just after motion starts on 'axis 2 and 3, outputs 19 and 20 'will become active`
`MOVE STOP STOP STOP STOP`	`'Just after axis 2 and 3 stop, 'outputs 19 and 20 will become 'inactive`

ENABLE PORTn

Name	**ENABLE PORTn**
Descriptor	Enable/Disable Communication Ports
Type	Set-UP
Default	ENABLE PORTn NO
Syntax	ENABLE PORT1 NO
Options	TAB YES NO
	F1 F2 F3 F4 F5 F6

Description

The **ENABLE PORT** statement can speed up program execution by decreasing the amount of time spent checking the communications buffers. If any of the five communications ports are not being used, they can be disabled with this statement. The functionality and default values are given below:

Port Description	n Value	Default State	Reduced execution time
RS-232 remote port 1	1	Enabled	40µsec/statement
RS-232 remote port 2	2	Enabled	40µsec/statement
*GPIB interface	3	*Enabled	20µsec/statement
Front Panel	4	Enabled	40µsec/statement
Unused RS232 port (encoder connector)	5	Disabled	0µsec/statement

* Optional card (if installed is automatically enabled)

Most statements typically execute in 2-3 ms, with **ON** conditions disabled, this time is reduced to less than a millisecond. The disabling of the ports

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

will reduce the execution time further by the amount specified in the above table.

See Also: **IN PORT, OUT PORT, IN LCD, OUT LCD, OUT SPOL, DISPLAY**

Example

Statement	Description
`001 ENABLE PORT2 NO`	`'Disable checking for incoming 'or outgoing characters on 'serial communications port 2.`
`002 ENABLE PORT1 YES`	`'Enable port 1 to receive or 'transmit characters`

ENABLE DIR REV

Name	**ENABLE DIR REV**
Descriptor	Enable/Disable Move Direction Reverse
Type	Set-Up
Default	ENABLE DIR REV * * * *
Syntax	ENABLE DIR REV YES NO YES *
Options	TAB YES NO NULL F1 F2 F3 F4 F5 F6

Description

The **ENABLE DIR** statement allows all motion on the specified axis to be reversed. This is useful when a motor is mounted in such a way that a positive move distance causes the load to move in the negative direction. All move types will have their commanded direction reversed when the **MOVE** statement executes. This statement has the same effect as reversing the wires on a stepper motor. The default is *not reversed*. The physical CW and CCW limit switches need to be reversed at the Model 4000 to function properly when direction is reversed.

See Also: **MOVE, ENABLE HLIMIT**

Example

Statement	Description
`ENABLE DIR REV YES NO * *`	`'Direction is reversed on axis '1, is not reversed on axis 2, 'and is unchanged on 3 and 4.`

ENCO

Name	**ENCO**
Descriptor	Set Encoder Parameters
Type	Set-Up
Default	ENCO BKL * * * *
Syntax	ENCO BKL 1000 1000 1000 1000
Options	TAB BKL MRES ERES DBND ABS F1 F2 F3 F4 F5 F6

Description

This command sets up various functions of the Model 4000 Controller to be used with the encoder feedback. The functions are:

BKL Defines the backlash of the system in motor steps. The backlash value is the number of motor steps that are sent out before stall checking is started. This is usually done to avoid a false stall detection which could result from the mechanical backlash between the linkage of the motor shaft to the encoder shaft.

MRES Defines the number of motor steps per 1 revolution of the motor.

ERES Defines the number of encoder steps per 1 revolution of the motor.

DBND Defines the deadband in encoder steps. The deadband determines the position tolerance allowed when using **ENABLE POSM** (position maintenance).

ABS Enables absolute encoders.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The order of the **ENCO** statements is important to ensure all encoder operations function properly. Below is the recommended order for encoder statements.

```
MODE
ENCO MRES
ENCO ERES
ENCO BKL
ENABLE STALL
ENABLE MOVKIL
ENCO DBND
POSM GAIN
POSM MAXVEL
ENABLE POSM
ENABLE NBWAIT
```

For information on commands required for absolute encoder operation, see *Chapter 5, Application Design, Absolute Encoders.*

Example

Statement	Description
`ENCO BKL 2000 2000 3000 *`	`'Defines the maximum backlash of 'axes 1 & 3`
`ENABLE STALL YES YES YES *`	`'Enable stall detect (If the 'backlash is exceeded during the 'move, a stall will be detected`
`ENABLE MOVKILL YES YES YES`	`'Kill Motion if a stall is 'detected`

ENCO BKL

Name	**ENCO BKL**
Descriptor	Set Backlash Distance
Type	Set-Up
Default	ENCO BKL * * * *
Syntax	ENCO BKL 1000 1000 1000 1000
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **ENCO BKL** statement sets the number of motor steps the Model 4000 will send before it starts checking for a stall. To enable stall checking the axis need not be in one of the encoder modes (either **E_INC** or **EABS**), but the **ENABLE STALL YES** must be executed.

This statement is most useful when the encoder is mounted on the load. When a new move is started, the Model 4000 will not check for a stall until after the specified number of steps have been sent out. *The default backlash is zero.* The **ENCO BKL** statement is scaled by the **UNIT POS** parameter.

To disable Stall Detection, execute **ENABLE STALL NO.**

See Also: **ENABLE STALL, ENABLE MOVKIL**

Statement	Description
`MODE E_INC E_INC * * *`	`'Axes in encoder mode`
`ENCO BKL 2000 5000`	`'Set the backlash value to 2000 'motor steps for axis 1 and 5000 'motor steps for axis 2`
`ENABLE STALL YES YES`	`'Enable stall detection`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENCO MRES

Name	ENCO MRES
Descriptor	Set Motor Resolution
Type	Set-Up
Default	ENCO MRES * * * *
Syntax	ENCO MRES 25000 25000 25000 25000
Options	TAB NULL
	F1 F2 F3 F4 F5 F6

Description

This statement defines the number of steps the Model 4000 should send out to command the drive to turn the motor 1 revolution. This number of steps must be one of the integer values in the table below. It is possible to use a Q variable for this value, but the value will be ignored at execution time if it is not one of the entries below. Align move parameters (i.e., **VEL**, **ACCEL**, **DECEL**) after issuing the **ENCO MRES** statement. If **ENCO MRES** is issued after move parameters, velocities, etc., the Model 4000 will be set back to the default values for those motion parameters. If **ENCO MRES** needs to be changed, re-issue the motion statements.

*The **ENCO MRES** command defines several parameters with one statement.*

- The maximum possible velocity that a move can achieve is limited by the **MRES** statement, regardless of any **VEL** statements issued.
- The minimum pulse width is the minimum high or low pulse width of the step signal. The table below has two ranges. The high range or low range are selected by the **VEL RANGE** statement.

MRES — Defines the number of motor steps per 1 revolution of the motor.

	For Vel Range High				For Vel Range Low			
Motor Resolution	RPS	Max frequency of step signal, KHz	VEL increment Hz	Min. pulse width µsec	RPS	Max frequency of step signal, KHz	VEL increment Hz	Min. pulse width µsec
200	250	50	1.526	10	25	5	0.1526	100
400	125	50	1.526	10	12.5	5	0.1526	100
500	100	50	1.526	10	10	5	0.1526	100
800	62.5	50	1.526	10	6.25	5	0.1526	100
1000	100	100	3.052	5	10	10	0.3052	50
2000	50	100	3.052	5	5	10	0.3052	50
4000	125	5000	15.259	1	12.5	50	1.5259	10
4096	122	499.7	15.259	1	12.2	49.9	1.5259	10
5000	100	500	15.259	1	10	50	1.5259	10
10000	50	500	15.259	1	5	50	1.5259	10
12500	40	500	15.259	1	4	50	1.5259	10
12800	39	499	15.259	1	3.9	49.9	1.5259	10
16384	30	491.52	15.259	1	3	49.15	1.5259	10
20000	25	500	15.259	1	2.5	50	1.5259	10
21600	23	496.8	15.259	1	2.3	49.68	1.5259	10
25000	20	500	15.259	1	2.0	50	1.5259	10
25400	19.5	495.79	15.259	1	19	49.57	1.5259	10
25600	19.5	499.2	15.259	1	19	49.82	1.5259	10
36000	13.8	496.8	15.259	1	13	49.68	1.5259	10
50000	10	500	15.259	1	1	50	1.5259	10
50800	9.8	497.84	15.259	1	0.9	49.78	1.5259	10
25000H	40.0	1000	30.518	0.5	4.0	100	3.0518	5.0
50000H	20.0	1000	30.518	0.5	2.0	100	3.0518	5.0
63488	15.7	1000	30.518	0.5	1.5	100	3.0518	5.0
76800	13.0	1000	30.518	0.5	1.3	100	3.0518	5.0
81920	12.2	1000	30.518	0.5	1.2	100	3.0518	5.0
100000	10.0	1000	30.518	0.5	1.0	100	3.0518	5.0
101600	9.8	1000	30.518	0.5	0.9	100	3.0518	5.0
102400	9.7	1000	30.518	0.5	0.9	100	3.0518	5.0
126976	7.8	1000	30.518	0.5	0.7	100	3.0518	5.0
128000	7.8	1000	30.518	0.5	0.7	100	3.0518	5.0
153600	6.4	1000	30.518	0.5	0.6	100	3.0518	5.0
163840	6.0	1000	30.518	0.5	0.6	100	3.0518	5.0
204800	4.8	1000	30.518	0.5	0.4	100	3.0518	5.0
253952	3.9	1000	30.518	0.5	0.3	100	3.0518	5.0
256000	3.8	1000	30.518	0.5	0.3	100	3.0518	5.0
278528	3.5	1000	30.518	0.5	0.3	100	3.0518	5.0
307200	3.2	1000	30.518	0.5	0.3	100	3.0518	5.0
327680	3.4	1000	30.518	0.5	0.3	100	3.0518	5.0

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

409600	2.4	1000	30.518	0.5	0.2	100	3.0518	5.0
425984	2.3	1000	30.518	0.5	0.2	100	3.0518	5.0
507904	1.9	1000	30.518	0.5	0.1	100	3.0518	5.0
512000	1.9	1000	30.518	0.5	0.1	100	3.0518	5.0
614400	1.6	1000	30.518	0.5	0.1	100	3.0518	5.0
655360	1.4	1000	30.518	0.5	0.1	100	3.0518	5.0
819200	1.2	1000	30.518	0.5	0.1	100	3.0518	5.0
1024000	0.9	1000	30.518	0.5	0.09	100	3.0518	5.0

See Also: **ENCO ERES, MODE ENABLE STALL, ENABLE MOVKIL, ENABLE POSM, ENABLE NDBWAIT**

Example

If you are using a standard S Series Motor Drive with a Compumotor supplied rotary encoder attached to the back of the motor, the S motor has 25000 steps/rev and the encoder has 400 steps/rev.

```
Statement                                   Description
ENCO MRES 278528 * * *                      'Axis 1 has 278528 steps
UNIT POS 278528 * * *
UNIT VEL 278528 * * *
UNIT ACCEL 278528 * * *
VEL 1 * * *                                 'Velocity of 1rps. Max possible
                                            'is 4.4 rps
ACCEL 1 * * *                               'Accel of 1 rps^2
DECEL 1 * * *                               'Deceleration of 1 rps^2
MOVE 1 * * *                                'Turn 1 rev on the Dynaserv

'Or if you want the motor resolution to be 425984, use:
ENCO MRES 425984 * * *                      'Axis 1 has 278528 steps
UNIT POS 425984 * * *
UNIT VEL 425984 * * *
UNIT ACCEL 425984 * * *
VEL 1 * * *                                 'Velocity of 1rps. Max possible
                                            'is 2.9 rps
ACCEL 1 * * *                               'Accel of 1 rps^2
DECEL 1 * * *                               'Deceleration of 1 rps^2
MOVE 1 * * *                                'Turn 1 rev on the Dynaserv
```

ENCO ERES

Name	**ENCO ERES**
Descriptor	Set Encoder Resolution
Type	Set-Up
Default	ENCO ERES * * * *
Syntax	ENCO ERES 4000 4000 4000 4000
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **ERES** (Encoder Resolution) is the number of Encoder pulses (after quadrature) expected by the Model 4000 after sending out the number of motor steps required to move 1 revolution. The motor step to encoder step ratio is used to detect a stall (**ENABLE STALL**) and also used to position the motor in encoder step moves.

The **ENCO ERES** statement must be executed before the Model 4000 can read an absolute encoder. The **ENCO ERES** statement should be executed before any move statements. Refer to *Chapter 5, Application Design* for more information on absolute and incremental encoders.

In some cases, the encoder may be mounted on the load instead of the motor. In such cases, it is possible that the number of encoder steps per motor revolution will not be an integer.

MRES Defines the number of motor steps per 1 revolution of the motor.
ERES Defines the number of encoder steps per 1 revolution of the motor.

See Also: **ENCO MRES, MODE ENABLE STALL, ENABLE MOVKIL, ENABLE POSM, ENABLE NDBWAIT, ENCO ABS**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example

A 1,000-line encoder will provide 4000 encoder steps per revolution after the quadrature detection circuitry in the Model 4000. An **ENCO ERES 4000** should be entered if axis #1 has a 1000 line encoder mounted on the motor.

Statement	Description
`ENCO MRES 25000 * * *`	`'Motor resolution is 25000 steps`
`ENCO ERES 4000 * * *`	`'Encoder resolution is 4000 steps for 1 'revolution of the motor`

ENCO DBND

Name	**ENCO DBND**
Descriptor	Set Deadband Position Maintenance
Type	Set-Up
Default	ENCO DBND * * * *
Syntax	ENCO DBND * * * *
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **ENCO DBND** statement sets the number of encoder steps of error allowed in either direction before a position maintenance correction move is initiated.

See Also: **ENABLE POSM, ENABLE NDBWAIT, POSM GAIN, POSM MAXVEL**

Example

Statement	Description
`ENCO DBND 10 5 * *`	`'Axes 1 and 2 are allowed 10 and '5 encoder steps position error 'respectively.`
`ENABLE POSM YES YES * *`	`'Enable position maintenance`
`ENABLE NDBWAIT YES YES * *`	`'Require position to be in 'deadband before proceeding`

It also specified the allowable tolerance that can be used with **ENABLE NDBWAIT**. **ENCO DBND** is scaled by the **UNIT POS** statement.

ENCO ABS

Name	**ENCO ABS**
Descriptor	Enable Absolute Encoders
Type	Set-Up
Default	ENCO ABS * * * *
Syntax	ENCO ABS YES NO YES *
Options	TAB YES NO NULL
	F1 F2 F3 F4 F5 F6

Description

The **ENCO ABS** statement causes the Model 4000 to check each encoder port specified for the existence of an absolute encoder. If the encoder is not connected or is not operating properly, an execution error is generated (#63) and the program is stopped (unless an **ON ERROR** statement has been executed). Axes with the NO or the NULL option are not checked for the presence of an absolute encoder.

Those axes which are specified to check for an absolute encoder and that do detect the absolute encoder will use the absolute encoder for functions such as: Stall Detection (Enable Stall), **MODE** Encoder Incremental (**E_INC**) or Encoder Absolute (**E_ABS**), and Position Maintenance (**POSM**). This also causes the Model 4000 to start checking for noise on the absolute encoder cable which would cause the Model 4000 to read an incorrect position. If noise is detected, execution error # 62 is generated.

The default values for this statement are: **ENCO ABS NO NO NO NO**

See Also: **ENCO MRES, ENCO ERES, ENABLE STALL, ENABLE MOVKIL, ENABLE POSM, ENABLE NDBWAIT, MODE**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENDLOOP

Name	ENDLOOP
Descriptor	End Loop
Type	Programming
Default	ENDLOOP
Syntax	ENDLOOP
Options	N/A

F1 F2 F3 F4 F5 F6

Description

The **ENDLOOP** statement defines the end of a group of statements to be executed repetitively. The **LOOP** and **ENDLOOP** statements are always used in pairs. The **BREAK** statement can be used to clear loops. If an **ENDLOOP** statement is encountered after a **BREAK** statement an execution error will occur.

See Also: **LOOP**, **BREAK**

Example

Statement	Description
LOOP 5	'Loop 5 times
MOVE 1000 2000	'Move axes 1 and 2
ENDLOOP	'End loop

GOSUB

Name	GOSUB
Descriptor	Gosub Label or Program
Type	Programming
Default	GOSUB LABELØ
Syntax	GOSUB LABELØ
Options	TAB ALPHA LABEL BEG FND LBL

F1 F2 F3 F4 F5 F6

Description

This statement causes the Model 4000 to jump to a subroutine. When the **GOSUB** statement is executed, the Model 4000 knows where it is in the program and then jumps to the label specified in the **GOSUB** statement. You cannot specify a line number. When the Model 4000 encounters a **RETURN** statement, it comes back to the first statement after the **GOSUB** statement. You can nest **GOSUB** statements (i.e., ***you can GOSUB to a subroutine, which in turn does a GOSUB to another subroutine, etc., to a depth of 12 levels***). Once you define a subroutine using the **LABEL** command, you can call it with the **GOSUB** statement as many times as needed. The **GOSUB** and **RETURN** statements are not matched. Once a **GOSUB** is executed, the first **RETURN** encountered is used to return to the most recent **GOSUB**.

The **GOSUB** label destination can be another program. If the destination is a program, it doesn't need to have a **RETURN** statement to execute without error. For example, if program X calls program Y, a **RETURN** statement is not needed to return execution to program X. Upon finishing program Y the Model 4000 automatically executes a **RETURN** statement, this causes program X to resume execution, and allows program Y to be programmed without a **RETURN** statement. Program Y can be run on its own as well as being called up by program X.

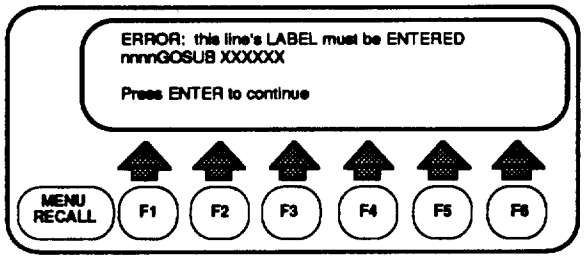
The **BREAK** statement clears all **GOSUB**'s. If a **RETURN** is executed after a **BREAK** has been executed (before another **GOSUB**), a program execution error will occur.

You can use up to 8 characters to define your subroutine. The following options are available for programming the **GOSUB** statement via the front panel:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

GOSUB MOVEFAST	You can use the **ALPHA** (Alphabet) function to define a meaningful subroutine. If you choose the **ALPHA** option, you can use the **UP** and **DOWN** arrow to scroll through the alphabet and use the left and right arrow keys to position the cursor. When you are finished editing all the characters, press the **ENTER** key. The first must be a letter, the next seven can be letters or numbers.
GOSUB BEG	The statement will jump to the beginning of the program
FND_LBL	This command will place previously defined **LABEL** statement labels on the display. When you press the **FND_LBL** key the up and down arrows will scroll through previously defined labels.

If the Model 4000 does not find the label destination during program execution, the program will abort. The editor will check for missing labels upon exiting and the following message will be displayed upon terminating that mode.



Where **xxxxxx** is the missing **LABEL** statement label.

See Also: **BREAK, ON, LABEL, RETURN**

Example

When the Model 4000 executes the program below, it goes from line to line number for **0010**, wait 2.5 seconds, and then return to line number to perform a 1,000-step move on axis #1.

Statement	Description
`GOSUB MOVE_2`	`'Call subroutine MOV_2`
`MOVE 1000 0 0 0`	`'Move axis #1 1000 steps`
`DONE`	`'End of program`
.	.
.	.
.	.
`LABEL MOVE_2`	`'Subroutine MOVE_2`
`WAIT FOR 2.5 SECONDS`	`'Delay of 2.5 seconds`
`RETURN`	`'Return to program line`
	`'following the GOSUB call`

GOTO

Name	GOTO
Descriptor	Goto Label or Program
Type	Programming
Default	GOTO LABELØ
Syntax	GOTO LABELØ
Options	TAB ALPHA LABEL BEG FND_LBL
	F1 F2 F3 F4 F5 F6

Description

The **GOTO** statement causes the Model 4000 to transfer program execution to the statement specified in the **GOTO** statement and begin executing instructions starting with the line that follows the specified label. You cannot **RETURN** to the line following a **GOTO** using a **RETURN** statement like you can with a **GOSUB**.

The **GOTO** label destination can be another program name. At the end of this program, another **GOTO** statement can be used to continue program execution. You cannot **GOTO** a label within another program directly.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If a **GOTO** or **GOTO** interrupt occurs in a subroutine, a **DONE** or **RETURN** at the end of the subroutine may send the program execution back to the original calling program.

The following options are available for programming the **GOTO** statement via the Model 4000 front panel:

GOTO MOVEFAST	You can use the **ALPHA** (Alphabet) function to define a meaningful subroutine. If you choose the **ALPHA** option, you can use the **UP** and **DOWN** arrow to scroll through the alphabet. When you are finished editing all the characters, press the **ENTER** key.
GOTO BEG	The statement will jump to the beginning of the program
FND_LBL	This command will place the existing labels on your display. To use this function, you must have previously defined, destinations using the **LABEL** statement. The up and down arrow keys will scroll through the previously define labels.

The editor will check for missing labels upon exiting and the following message will be displayed if a **LABEL** is not found:

ERROR: this line's LABEL must be ENTERED

See Also: **BREAK, ON, LABEL, RETURN**

Example

The example shown would cause the program, upon hitting line **0001**, to jump directly to label 1 on line and begin executing at that line. Program lines - would not be executed.

Statement	Description
`GOTO LABEL1`	`'Transfer program execution to 'subroutine`
`MOVE 1ØØ 1ØØØ * Ø`	`'Move on Axes 1, 2, and 4`
`MOVE Ø 2ØØ Ø 2ØØ`	`'Move on all four axes`
`DONE`	`'End of program`
`LABEL LABEL1`	`'Subroutine LABEL1`
`MOVE 1ØØØ Ø 1ØØØ Ø`	`'Move on all four axes`

IF

Name	**IF**
Descriptor	If Then Branch
Type	Programming
Default	IF GOTO LABELØ
Syntax	IF Q1 > Q2 GOTO LABELØ

Options

TAB	Q	IN24	BIT	TRIG	ETC
TAB	JOYTRIG	SLIMIT	HLIMIT	ESTOP	ETC
TAB	F-KEY	ENTERK	STARTK	MENUREC	ETC
TAB	ARROWR	ARROWL	ARROWU	ARROWD	ETC
TAB	STALL				ETC
F1	**F2**	**F3**	**F4**	**F5**	**F6**

Description

The condition is checked at the time the statement executes, if it is true, then the branch is taken. If it is not true, the statement following the subroutine label is executed. The options available for the **IF** statement are listed below:

IF Q	Check the value of the variable Q and branch to a routine if the condition is true.
IF IN24	Check the input pattern of the 24 inputs and branch to a routine if the condition is true.
IF BIT	Check 1 of the 24 inputs and branch to a routine if the condition is true.
IF TRIG	Check the value of one of the four trigger inputs and branch to a routine if the condition is true.
IF JOYTRIG	Check the Joystick Trigger Input. If active, branch to a routine.
IF SLIMIT	If Software Limit is active, branch to a routine.
IF HLIMIT	If Hardware Limit is active, branch to routine.
IF ESTOP	If Emergency Stop input is active branch to a routine.
IF F-KEY	Check the F-Key pressed, and branch to a routine if the condition is true.
IF ENTERK	If ENTER is pressed, branch to a routine.
IF STARTK	If START is pressed, branch to a routine.
IF MENUREC	If MENU RECALL is pressed, branch to a routine.
IF ARROWR	If Right Arrow key is pressed, branch to a routine.
IF ARROWL	If Left Arrow key is pressed, branch to a routine.
IF ARROWU	If Up Arrow key is pressed, branch to a routine.
IF ARROWD	If Down Arrow key is pressed, branch to a routine.
IF STALL	If a Stall Condition is detected, branch to a routine.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Since **IF** statements are all checked once at the time of execution, a common programming practice is to place the **IF** conditions to be checked in a loop of some sort in the program. This way, when one of the **IF** conditions becomes true, the loop is exited and a program branch is taken. See the example under the **IF F-KEY** statement for more information.

See Also: **BREAK, ON, LABEL, RETURN, GOSUB, GOTO**

Example

```
Statement                          Description
LABEL CHOOSE                       'Subroutine CHOOSE
IF F-KEY1 GOTO START               'If F1 key is pressed, goto start routine
IF F-KEY2 GOTO STOP                'If F2 key is pressed, goto stop routine
GOTO CHOOSE                        'Continue checking is function keys pressed
LABEL START                        'Define Start routine
MOVE SLEWCW SLEWCW * *             'Move at constant speed
GOTO CHOOSE                        'Return to menu
LABEL STOP                         'Define Stop routine
MOVE STOP STOP * *                 'Stop both axes
GOTO CHOOSE                        'Return to menu
```

IF Q

Name	**IF Q**
Descriptor	Compare Q Variable
Type	Programming
Default	IF Qn = Qn GOTO LABELØ
Syntax	IF Q1 = Q2 GOTO LABELØ

Options						
	TAB	Q	=	<	>	ETC
	TAB	OR	AND		FND_LBL	ETC
	TAB	BEG	AND	ALPHA	GOSUB	ETC
	F1	**F2**	**F3**	**F4**	**F5**	**F6**

Description

The **IF** Qn statement compares the first Qn to either another Qn variable or a constant number. If the expression is true the program branches to the **LABEL** specified. The operators are =, <, >, <>, <=, >=, **OR**, and **AND**. Only the numeric variables are allowed (i.e., **Q1 - Qnn**). String variables (**Q100 - Q109**) cannot be used.

See Also: **DEFINE VAR, MATH**

Example

This example checks the value of **Q56**, and branches to different routines, depending on the value of **Q56**.

```
Statement                          Description
IF Q56 = Q35 GOTO GRIND            'If equal, take the branch. If Q56 is equal
                                   'to 10,000 take the branch
IF Q56 = 10000 GOTO FINISHED       'If Q56 is less than 10,000, take the
                                   'branch
IF Q56 < 10000 GOTO INTER          'If Q56 is greater than or equal to Ø, take
                                   'the branch
IF Q56 >= 0 GOTO STILL
```

For **OR** and **AND**, *true* means non-zero, and *false* means zero. In the example below, the first two branches will not be taken, the second two will be taken. Also, only positive operands are allowed with the **OR** and **AND** operators. The signal is ignored and the fractional portion truncated.

```
Statement                          Description
MATH Q1 = 0
MATH Q2 = 0
MATH Q3 = 1
MATH Q4 = 1
IF Q1 AND Q3 GOTO LABELØ           'Branch not taken
IF Q1 OR Q2 GOTO LABELØ            'Branch not taken
IF Q1 OR Q3 GOTO LABEL1            'Branch taken
IF Q3 AND Q4 GOTO LABEL1           'Branch taken
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IF IN24

Name	IF IN24
Descriptor	Compare 24 Input to Pattern
Type	Programming
Default	IF IN24 = PATTn GOTO LABELØ
Syntax	IF IN24 = PATT1 GOTO LABELØ
Options	TAB BEG END ALPHA GOSUB FND LBL
	F1 F2 F3 F4 F5 F6

Description

The **IF IN24** statement compares the 24 programmable inputs to a previously defined **PATT** variable (see **PATT** statement), if they match, the program branches to the routine specified.

The **DEFINE 24** statement determines which logic type (positive logic or negative logic) the **IF IN24** statement uses to determine if the 24 programmable inputs match the pattern variable.

If the = is changed to an ! symbol and the 24 inputs do not match the **PATT** specified, the branch is taken. Use the **ALPHA** option to get the ! symbol if entering this command via the front panel.

See Also: **ON IN24, WAIT FOR IN24, DEFINE IN24**

Example

In this example, the Model 4000 checks the inputs and tells the user whether the inputs match the pattern that was defined in the program.

Statement	Description
`PATT1 1XXX XXXX XXXX XXXX XXXX XXXX`	`'Define Pattern 24 as Input 1 = '1; others don't care`
`IF IN24 = PATT1 GOTO LABEL1`	`'If IN24 matches pattern 1 goto 'label1`
`OUT LCD2,01 ^DOES NOT MATCH^`	`'Write a message to LCD`
`DONE`	
`LABEL LABEL1`	`'Define label as 001`
`OUT LCD2,01 ^MATCHED^`	`'Define label 001 write a 'message to LCD`

IF BIT

Name	IF BIT
Descriptor	Compare One of the 24 Inputs
Type	Programming
Default	IF BITn = 0 GOTO LABELØ
Syntax	IF BIT1 = 1 GOTO LABELØ
Options	TAB BEG END ALPHA GOSUB FND LBL
	F1 F2 F3 F4 F5 F6

Description

The **IF BIT** statement compares one of the 24 programmable inputs to zero or 1, as specified, and if it matches then a branch is taken. The check is made only once. The value n can be 1 to 24.

See Also: **ON BIT, WAIT FOR BIT, DEFINE IN24**

Example

This program checks programmable input 2 to determine which of two different moves to make.

Statement	Description
`IF BIT2 = 1 GOTO MOVE1`	`'If input 2 is 1, goto MOVE1`
`GOTO MOVE2`	`'Input 2 is 0, goto MOVE2`
`DONE`	
`LABEL MOVE1`	`'Define move 1`
`MOVE 50000 20000 * *`	`'Execute move 1`
`DONE`	
`LABEL MOVE2`	`'Define move 2`
`MOVE 2500 2500`	`'Execute move 2`
`DONE`	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IF TRIG

Name	**IF TRIG**
Descriptor	Compare 1 of 4 Trigger Inputs
Type	Programming
Default	IF TRIGn = 0 GOTO LABELØ
Syntax	IF TRIG1 = 1 GOTO LABELØ
Options	TAB BEG END ALPHA GOSUB FND_LBL F1 F2 F3 F4 F5 F6

Description

The **IF TRIG** statement compares one of the four axis trigger inputs to Ø or 1 (as specified). If it matches, it branches to the routine specified. The value of n can be 1 to 4. This command always uses positive logic.

See Also: **ON TRIG, WAIT FOR TRIG**

IF JOYTRIG

Name	**IF JOYTRIG**
Descriptor	Compare Joystick Input
Type	Programming
Default	IF JOYTRIG GOTO LABELØ
Syntax	IF JOYTRIG GOTO LABELØ
Options	TAB BEG END ALPHA GOSUB FND_LBL F1 F2 F3 F4 F5 F6

Description

The **IF JOYTRIG** statement checks once to see if the **JOY TRIG** (joystick trigger input) input is active. If the input is active 0V (current flowing), the program jumps to the routine specified. The Joystick Trigger is not axis specific, there is only one joystick trigger input.

The **JOYTRIG** input can be used as a general purpose programmable input if required in your program. It does not automatically enable or release joystick mode if it goes active.

See Also: **ON JOYTRIG, WAIT FOR JOYTRIG**

Example

```
Statement                                  Description
IF JOYTRIG GOTO MESSAGE                    'See if joystick is asked for
    •
    •
    •
LABEL MESSAGE
ENABLE JOY YES YES NO NO                   'Actually turn on joystick
OUT LCD2,01 ^JOYSTICK ACTIVE^              'Display message on LCD that
                                           'joystick is active
```

IF SLIMIT

Name	**IF SLIMIT**
Descriptor	Branch If Position Passed
Type	Programming
Default	IF SLIMIT ANY AXIS GOTO LABELØ
Syntax	IF SLIMIT AXISn GOTO LABELØ IF SLIMIT AXIS1 GOTO LABELØ IF SLIMIT ANY AXIS GOTO LABELØ
Options	TAB END ANY ALPHA GOSUB FND_LBL F1 F2 F3 F4 F5 F6

Description

The **IF SLIMIT** statement checks once to see if a software limit has been reached on any one of the four axes. If the software limit has been passed, the program will jump to the routine specified.

See Also: **ON SLIMIT, SLIMIT, ENABLE M_SLIM, ENABLE E_SLIM**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example

Statement	Description
`IF SLIMIT ANY AXIS GOTO MESSAGE`	'If software limit has been 'reached jump to message routine
. . .	
`LABEL MESSAGE`	'Define message routine
`MOVE KILL KILL KILL KILL`	'Kill all axes-shut off pulses
`OUT LCD2,01 ^SOFTWARE LIMIT^`	'Display message on LCD

IF HLIMIT

Name	IF HLIMIT
Descriptor	Branch If Limit Input Active
Type	Programming
Default	IF HLIMIT ANY AXIS GOTO LABELØ
Syntax	IF HLIMIT AXISn GOTO LABELØ IF HLIMIT AXIS1 GOTO LABELØ IF HLIMIT ANY AXIS GOTO LABELØ
Options	TAB END ANY ALPHA GOSUB FND_LBL F1 F2 F3 F4 F5 F6

Description

The **IF HLIMIT** statement checks once to see if the end of travel limit inputs are active (5V, current not flowing). If the **ANY AXIS** default option is used, the CW and CCW limit inputs for all four axes are checked. If the **AXISn** option is chosen, only the CW and CCW limit inputs for the axis specified are checked. If the limit input (s) is active, the program jumps to the routine specified.

See Also: **ON HLIMIT, ENABLE HLIMIT**

Example

Statement	Description
`IF HLIMIT AXIS1 GOTO MESSAGE`	'If axis 1 end-of-travel limit 'is active jump to message

IF ESTOP

Name	IF ESTOP
Descriptor	Branch if Estop Input Active
Type	Programming
Default	IF ESTOP GOTO LABELØ
Syntax	IF ESTOP GOTO LABEL
Options	TAB BEG END ALPHA GOSUB FND_LBL F1 F2 F3 F4 F5 F6

Description

The **IF ESTOP** statement checks once to see if the **ESTOP** (Emergency Stop) input is active (5V, current not flowing). If the input is active, the program branches to the routine specified.

If the **ESTOP** input is active, pulses will not be sent even if the Model 4000 is in a move. Pulses are cut-off with a hardware switch in the Model 4000. To terminate moves in progress, a **MOVI KILL KILL KILL KILL** or a **MOVE KILL KILL KILL KILL** statement should be placed after the **IF ESTOP**.

The **ON ESTOP** statement will become active before the **IF ESTOP**. The **IF ESTOP** should only be used after the **ON ESTOP** statement has become true.

Each time a program is started, the Model 4000 automatically executes an **ON ESTOP GOTO END** statement, which will terminate the program should an **ESTOP** occur. You must execute an **ON ESTOP** statement of your own if you wish to have a different function for the **ESTOP** input.

The **ESTOP** input does not shut power off to the Model 4000, however, a fault output is provided that can be wired to a system power-off relay

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

device if desired. Refer to *Chapter 6, Hardware Reference* for a pin out of the **ESTOP** input and the fault output.

See Also: **ON ESTOP**

Example

Statement	Description
`IF ESTOP GOTO ESTOP`	`'If ESTOP is active jump to 'ESTOP routine`
`MOVE 10000 20000 * *`	`'If ESTOP is not active make the 'move`
`DONE`	`'End of program`
`LABEL ESTOP`	`'Define ESTOP routine`
`OUT LCD2,01 ^EMERGENCY STOP^`	`'Display ^EMERGENCY STOP^ on the 'LCD`

IF F-KEY

Name	**IF F-KEY**
Descriptor	Branch if Function Key Pressed
Type	Programming
Default	IF F-KEYn GOTO LABELØ
Syntax	IF F-KEY1 GOTO LABELØ
Options	TAB BEG END ALPHA GOSUB FND LBL F1 F2 F3 F4 F5 F6

Description

The **IF F-KEY** statement checks to see if the specified **F-KEY** is currently being pressed. If it is, the program jumps to the routine specified.

See Also: **ON F-KEY, WAIT FOR F-KEY**

Example

The example below displays the message **PRESS ONE OF THE F-KEYS** on line and on line displays the choices: **CUT DRILL JOG HOME**. Upon pressing one of the keys, the subroutine that corresponds to the description is executed and then the Model 4000 looks for another **F-KEY** to be pressed.

Statement	Description
`OUT LCD3,01 ^PRESS ONE OF THE F-KEYS^`	`'Display message on LCD`
`OUT LCD4,01 ^CUT DRILL JOG^`	`'Display Function Key menu on 'LCD`
`OUT LCD4,23 ^HOME^`	`'Display Function Key menu on 'LCD`
`LABEL GETFKEY`	`'Find routine for each 'function`
`IF F-KEY1 GOSUB CUT`	`'If F1 is pressed, goto CUT 'routine`
`IF F-KEY2 GOSUB DRILL`	`'If F2 is pressed, goto DRILL 'routine`
`IF F-KEY3 GOSUB JOG`	`'If F3 is pressed, goto JOG 'routine`
`IF F-KEY4 GOSUB HOME`	`'If F4 is pressed, goto HOME 'routine`
`GOTO GETFKEY`	`'Loop get F-KEY routine`
`LABEL CUT`	`'Define CUT routine`
`RETURN`	`'Return`
`LABEL DRILL`	`'Define DRILL routine`
`RETURN`	`'Return`
`LABEL JOG`	`'Define JOG routine`
`RETURN`	`'Return`
`LABEL HOME`	`'Define LABEL routine`
`RETURN`	`'Return`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IF ENTERK

Name	IF ENTERK
Descriptor	Branch if Enter Key Pressed
Type	Programming
Default	IF ENTERK GOTO LABELØ
Syntax	IF ENTERK GOTO LABELØ
Options	TAB BEG END ALPHA GOSUB FND LBL
	F1 F2 F3 F4 F5 F6

Description

The **IF ENTERK** statement checks once to see if **ENTER** on the front panel is pressed. If **ENTER** is pressed, the program jumps to the routine specified.

Example

```
Statement                              Description
LABEL GETENT
IF ENTERK GOSUB ENTER                  'If ENTER key pressed, GOSUB enter
GOTO GETENT
LABEL ENTER
RETURN                                 'ENTER subroutine
```

IF STARTK

Name	IF STARTK
Descriptor	Branch if Start Key Pressed
Type	Programming
Default	IF STARTK GOTO LABELØ
Syntax	IF STARTK GOTO LABELØ
Options	TAB BEG END ALPHA GOSUB FND LBL
	F1 F2 F3 F4 F5 F6

Description

The **IF STARTK** statement checks once to see if **START** on the front panel is pressed. If **START** is pressed, the program jumps to the routine specified.

See Also: **ON STARTK**

Example

```
Statement                              Description
LABEL GETSTART
IF STARTK GOSUB START                  'If START key pressed, GOSUB start
GOTO GETSTART
LABEL START
RETURN                                 'START subroutine
```

IF MENUREC

Name	IF MENUREC
Descriptor	Branch if Menu Recall Key is Pressed
Type	Programming
Default	IF MENUREC GOTO LABELØ
Syntax	IF MENUREC GOTO LABELØ
Options	TAB BEG END ALPHA GOSUB FND LBL
	F1 F2 F3 F4 F5 F6

Description

The **IF MENUREC** statement checks once to see if **MENU RECALL** is pressed. If **MENU RECALL** is pressed, the branch is taken.

See Also: **ON MENUREC**

Example

```
Statement                              Description
LABEL GETMENU
IF MENUREC GOSUB MENU                  'If MENUREC key pressed, GOSUB menu
GOTO GETMENU
LABEL MENU
RETURN                                 'MENU subroutine
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IF ARROWR

Name	**IF ARROWR**
Descriptor	Branch if Arrow Right Key is Pressed
Type	Programming
Default	IF ARROWR GOTO LABEL0
Syntax	IF ARROWR GOTO LABEL0
Options	TAB BEG END ALPHA GOSUB FND_LBL F1 F2 F3 F4 F5 F6

Description

The **IF ARROWR** statement checks once to see if the **RIGHT ARROW** on the front panel is pressed. If the **ARROW RIGHT** is pressed, the program jumps to the routine specified.

IF ARROWL

Name	**IF ARROWL**
Descriptor	Branch if Arrow Left Key is Pressed
Type	Programming
Default	IF ARROWL GOTO LABEL0
Syntax	IF ARROWL GOTO LABEL0
Options	TAB BEG END ALPHA GOSUB FND_LBL F1 F2 F3 F4 F5 F6

Description

The **IF ARROWL** statement checks once to see if the **LEFT ARROW** on the front panel is pressed. If the **ARROW LEFT** is pressed, the program jumps to the routine specified.

Example

Statement	Description
LABEL GETARROWL	
IF ARROWL GOSUB ARROWL	'If left arrow pressed, GOSUB 'ARROWL
GOTO GETARROWL	
LABEL ARROWL	
OUT LCD2,01^Left arrow key pressed^	'ARROWL subroutine
RETURN	

IF ARROWU

Name	**IF ARROWU**
Descriptor	Branch if Arrow Up Key is Pressed
Type	Programming
Default	IF ARROWU GOTO LABEL0
Syntax	IF ARROWU GOTO LABEL0
Options	TAB BEG END ALPHA GOSUB FND_LBL F1 F2 F3 F4 F5 F6

Description

The **IF ARROWU** statement checks once to see if the **UP ARROW** on the front panel is pressed. If the **ARROW UP** is pressed, the program jumps to the routine specified.

IF ARROWD

Name	**IF ARROWD**
Descriptor	Branch if Arrow Down Key is Pressed
Type	Programming
Default	IF ARROWD GOTO LABEL0
Syntax	IF ARROWD GOTO LABEL0
Options	TAB BEG END ALPHA GOSUB FND_LBL F1 F2 F3 F4 F5 F6

Description

The **IF ARROWD** statement checks once to see if the **DOWN ARROW** on the front panel is pressed. If the **ARROW DOWN** is pressed, the program jumps to the routine specified.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IF STALL

Name	IF STALL
Descriptor	Branch if Stall Condition Exists
Type	Programming
Default	IF STALL ANY AXIS GOTO LABELØ
Syntax	IF STALL AXISn GOTO LABELØ IF STALL AXIS1 GOTO LABELØ IF STALL ANY AXIS GOTO LABELØ
Options	TAB END ANY ALPHA GOSUB FND_LBL F1 F2 F3 F4 F5 F6

Description

The **IF STALL** statement checks once if a stall condition exists for the axis specified. The stall detect function must be enabled using the **ENABLE STALL** statement to use **IF STALL** statement. The stall condition can be detected only if you are using an encoder to receive position feedback.

See the **ENCO BKL** (Encoder Backlash) statement for more details on how to set up the stall function. See the **ON STALL** statement for more options in stall detection.

The following statements must be executed before the **ON STALL** statement before the **ON STALL** statement will work

- **MODE E_INC** or **E_ABS**
- **ENCO BKL** (optional)
- **ENABLE STALL YES**

Example

Statement	Description
`ENCO BKL 250 250 * *`	`'Set the maximum backlash in 'motor`
`ENABLE STALL YES YES * *`	`'Enable stall detection on axes '1 and 2 steps to be 250 steps 'for axes 1 and 2`
`MOVI 20000 20000 * *`	`'Start the move`
`LABEL CK-STALL`	`'Label to loop and check for 'stall`
`IF STALL ANY AXIS GOTO STALL`	`'If stall is detected, jump to 'routine STALL`
`GOTO CK_STALL`	`'GOTO label to continue checking`
•	
•	
•	
`LABEL STALL`	`'Define Stall routine`
`MOVE KILL KILL KILL KILL`	`'Kill all four axes`
`OUT LCD2,01 ^STALL DETECTED^`	`'Display messages on LCD`

IN

Name	IN
Descriptor	Input to a Q Variable
Type	Programming
Default	IN
Syntax	IN Qn = LCD1,01 ^Q1=^
Options	TAB LCD PORT IN24 POS ETC TAB VEL STATUS DATA BIT ETC TAB TRIG ANALOG ERROR ETC F1 F2 F3 F4 F5 F6

Description

The **IN** statement allows you to assign a value to a variable during program execution from one of several different sources. The input options available to the Model 4000 are:

IN LCD	User can input numeric data using the operator panel.
IN PORT	User can input data from the external communications port.
IN IN24	User can input data through the 24 inputs. The data can be read as BCD input or HEX inputs.
IN POS	This statement uses the current motor position of the specified axis as the value of the Q variable.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IN VEL	This statement uses the most current velocity on the specified axis as the value of the Q variable.
IN STATUS	This statement assigns the current status of an axis (1 - 7) as the value of the Q variable.
IN DATA	This statement looks for a DATA statement and assigns the next data value as the value of the Q variable.
IN BIT	Assigns either a Ø or 1 depending on the state of one of the 24 inputs, and depending on the DEFINE IN24 statement logic parameter.
IN TRIG	Assigns either a 0 or 1 depending on the state of one of the four axis trigger inputs.
IN ANALOG	This statement reads the analog input channel specified, and stores the values as a variable. The voltage at the analog input can be from 0V to 2.5V, and the value assigned to the variable will be from 0V to 2.5V.
IN ERROR	This statement assigns a Ø if no execution error has occurred or a number from 1-64 is assigned depending on which error occurred.
Q1-Q99	These default numeric variables must be real numbers (integers are included), not character strings.
Q100-Q109	These default numeric variables are character strings and must be less than 40 characters. These values can be defined from the external communication port using the VARI command or by using the IN Q100 = PORT1 statement.
Q1-Q99	The numeric Q variables may be used as a pointer variable by using the notation QQn.

QQ1 = Q[value of Q1 variable]

For example, if **Q3 = 10**, then executing **MATH QQ3 = 5** does the following:

QQ3 = Q[valQ3] = Q10. So **Q10** gets set to 5.

QQ Pointer variables are discussed in *Chapter 5, Application Design.*

Example

Statement	Description
MATH Q3 = 10	'Set Q3 to 10
MATH QQ3 = 5	'Read value of Q3, use that number 'as the Q number to assign 5 to, 'Q10 is set to 5
IN Q100 = PORT1 ^ENTER YOUR NAME^	'Wait for the user to type their 'name and a return
MATH Q1 = 100	'Set Q1 to 100
OUT LCD1,01 QQ1	'Display on the LCD the Q100 'string, whatever name was 'entered

See the following Daughter statement descriptions for specific definitions and examples.

IN LCD

Name	**IN LCD**
Descriptor	Prompt the LCD for Input
Type	Programming
Default	IN Qn = LCDn,01 ^your message^
Syntax	IN Q1 = LCD1,01 ^Q1=^ IN Qn = LCD1,01 Qnnn IN QQn = LCD1,01 Qnnn
Options	TAB ALPHA ^ Q F1 F2 F3 F4 F5 F6

Description

The **IN LCD** prompts the operator for numeric data. A string is displayed on the LCD and then the Model 4000 waits for the you to enter a number and press the **ENTER** key. Do not to have a **DISPLAY** statement running during the time the Model 4000 waits for the your keypad entry, if both statements are using the same space on the LCD, then numbers cannot be entered. The prompt string displayed on the LCD when the statement is executed can be modified using the alpha option and the keypad, or if left as is, a **^Qn^** = prompt string will be created. A **Q1-Q109** variable can also be used as the source for the prompt, as well as **QQ1-QQ99** pointer variables that point to 1-109. The string will be located at LCD line n and at column Ø1-40.

If an **IN LCD** statement is executed and the user enters no number via the key pad, but just presses **ENTER**, a program error will occur. This can be

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

worked around by using an **OUT LCD** statement to place the current value of the variable on the LCD in the same place the **IN LCD** expects to read it. The **OUT LCD** is immediately followed by the **IN LCD** so if the user enters no new value, the variable is set equal to its current value.

The **IN LCD** statement will pause program execution until some data is entered.

See Also: **OUT LCD, DISPLAY**

Example

Statement	Description
`IN Q1 = LCD2,01 ^PART NUMBER^`	`'Prompts the user to input the 'part number via the front panel`
`OUT LCD2,01 ^MAKING PART^ Q1`	`'Write message to the operator`
`GOSUB MKPART`	`'Jump to MKPART (make part) 'subroutine`

IN PORT

Name	**IN PORT**
Descriptor	Prompt a Remote Port for Input
Type	Programming
Default	IN Qn = PORTn ^your message^
Syntax	IN Qn = PORT1 Qnnn IN QQn = PORT2 Qnnn
Options	TAB ALPHA ^ Q N NI **F1 F2 F3 F4 F5 F6**

Description

The **IN PORT** statement suspends program execution until data is received from one of two RS-232C ports or the optional GPIB port. Ports 1 and 2 are RS-232 and port 3 is the optional GPIB port.

Once entered, the data is stored in the variable specified. A prompt string can be sent to the port before the data is entered and this string can be contained in the statement between the ^ ^ symbols or it can be a variable. The prompt string output to the port when the statement is executed can be modified using the alpha option and the keypad. If `^your message^` is left unchanged, the following output string will be generated: `^Qn=^`.

The maximum length of a string between the ^ ^ signs is 18 when entered from the front panel (maximum of 40 when entered over RS-232C). The default length of string variables is 20 characters (up to 20 will be displayed on the front panel while editing). A remote terminal can be used to enter strings over 20 characters If using port 3, the optional GPIB interface, refer to *Chapter 5, Application Design* for configuration parameters. You must end the string with the **CR** (ØD hex) byte for the data to be accepted, and program execution to continue.

The **N** option looks for the first number returned after sending the prompt string. This option will ignore any letters or non-numeric characters until the first number is encountered. After the first number is encountered, any non-numeric character can be used to complete the **IN** statement. In the example below, the backspace character terminates the entry.

The **NI** option is identical to the **N** option, except that the device connected to the Model 4000 is known to echo the prompt string. After sending the prompt string this Model 4000 will ignore the same number of characters received as were sent in the prompt string. The characters that follow the ignored characters are used to complete the **IN** statement. This is useful when talking to Compumotor X language products that have the character echo turned on.

See Also: **DEFINE VAR, OUT PORT, DEFINE TERM CHAR, DEFINE GPIB**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Examples

Statement	Description
`DEFINE TERM CHAR PORT 2 = 8`	`'Set backspace = to terminating`
	`'character = backspace character`
`IN Q1 = PORT2 ^1DPA  ^ N`	`'The Model 4000 reads the data.`
	`'When a carriage return is`
	`'reached or 85 characters are`
	`'entered, the data is parsed.`
	`'The first number parsed is used`
	`'for Q1.(In this example Q1 = 1)`

Statement	Description
`DEFINE TERM CHAR PORT 1 = 8`	`'Set backspace = to terminating`
	`'character = backspace character`
`IN Q1 = PORT1 ^1DPA  ^ NI`	`'The Model 4000 does the same as`
	`'for the N option, except that`
	`'the number of characters found`
	`'between the ^ ^ symbols are`
	`'stripped from the front of the`
	`'response string.  This`
	`'eliminates the echo of these`
	`'characters '(In this example Q1`
	`'= the DPA response, i.e.,`
	`'00025000)`

Statement	Description
`IN Q106 = PORT1 ^ENTER YOUR NAME^`	`'Prompts the user for their name`
	`'via serial port 1`
`OUT LCD2,01 ^NAME IS^ Q106`	`'Print your name on the LCD`

IN IN24

Name	**IN IN24**
Descriptor	Get Status of 24 Inputs
Type	Programming
Default	IN Qn = IN24 BCDm.n IN Qn = IN24 HEXn IN PATTN = IN24 IN24
Syntax	IN Q1 = IN24 BCD 4.2
Options	TAB PATT Q HEX BCD F1 F2 F3 F4 F5 F6

Description

The **IN24** statement assigns to the first variable a value derived from the 24 programmable inputs. The 24 inputs can be treated as BCD or hexadecimal.

The BCD option of the **IN24** statement causes **BCD m.n** to appear. This variable refers to the number of nibbles to use from the 24 programmable inputs. For the BCD total option, m + n = number of nibbles looked at, m = number of nibbles to the left of the decimal, and n = number of nibbles right of the decimal. Most significant nibbles that you do not use are treated as *zero's. The HEX option does NOT allow specification of a radix point.* For the HEX option, n = number of nibbles used.

See Also: **DEFINE IN24, PATT, IN BIT, ON IN24, ON BIT, IF IN24, IF BIT**

Example

To understand how the assignment is made, lets assume that we are using **DEFINE IN24 POSITIVE LOGIC** and the 24 programmable inputs are wired as:

MSB (pin #1) (pin #47) **LSB**

⇑ ⇑

```
ØØØØ ØØØØ ØØ11 1ØØ1 Ø111 Ø1ØØ
8421 8421 8421 8421 8421 8421
```

The Most Significant Bit (MSB), bit 24, is on pin #1 and the least significant bit (LSB), bit 1 is on pin #47.

Assume **IN Q1 = IN24 BCD 3.1** is executed. The value of **Q1** after this statement is 397.4. It reads the least significant digit as 1/10 place digit.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If the command was **1INQ1 = IN24 BCD2.2**, the value of **Q1** would have been 39.74

If an incorrect pattern of inputs is active when the **IN IN24** BCD statement is executed (i.e., 1111 on inputs 1-4), then a program execution error will occur. A value of 9 on any 4 inputs is the maximum value for that nibble.

Statement	Description
`IN Q5 = IN24 BCD 3.1`	`'Read 24 inputs in binary code 'decimal mode`
`OUT LCD2,01 ^BCD VALUE IS^ Q5`	
`MOVE Q5 * * *`	`'Move axis 1 Q5 steps`

IN POS

Name	**IN POS**
Descriptor	Get Current Position of an Axis
Type	Programming
Default	IN Qn = POSITION OF AXISn MINC
Syntax	IN Q1 = POSITION OF AXIS4 MINC
Options	TAB MABS MINC EABS EINC Q F1 F2 F3 F4 F5 F6

Description

The **IN POS** statement assigns to the **Q** variable the current position at the axis specified. The options **MABS, MINC, EABS**, or **EINC** determine the type of position data assigned. **MABS** refers to the number of motor steps from the last **PDEF** (in **MODE M_ABS**) or power cycle. **MINC** refers to the number of motor steps from the beginning of a move in progress. If not moving the incremental motor position is the last move distance. **EABS** refers to the number of encoder steps from the last **PDEF** (in **MODE E_ABS**) or power cycle. **EINC** refers to the number of encoder steps from the beginning of a move in progress, or the last move distance.

See Also: **MODE, PDEF, ON M_INC, ON M_ABS, ON E_INC, ON E_ABS**

Example

Statement	Description
`PDEF 0 0 0 0`	`'Define the current position as zero`
`MOVE 5000 * * *`	
`IN Q3 = POSITION OF AXIS1 M_ABS`	`'Q3 becomes the 5,000`
`MATH Q3 = Q3 + 10000`	`'Add 10,000 to variable Q3`
`MOVE Q3 0 0 0`	`'Move axis 1 15,000 steps`

IN VEL

Name	**IN VEL**
Descriptor	Get Current Velocity of an Axis
Type	Programming
Default	IN Qn = VELOCITY OF AXISn
Syntax	IN Q1 = VELOCITY OF AXIS2
Options	TAB Q F1 F2 F3 F4 F5 F6

Description

The **IN VEL** statement assigns to the **Q** variable the current velocity value of the axis specified. The velocity read is the instantaneous velocity in the users **VEL** units.

Example

Statement	Description
`VEL 20000 10000 * *`	`'Set velocity`
`LOOP 5`	`'Loop 5 times`
`IN VEL Q2 = VELOCITY OF AXIS 2`	`'Input velocity of axis 2 as Q2`
`VEL = Q2 Q2 * *`	`'Set new velocity for axis 2 to Q2`
`MOVE 2000 5000 0 0`	`'Move axes 1 and 2`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IN STATUS

Name	IN STATUS
Descriptor	Get Status of an Axis
Type	Programming
Default	IN Qn = STATUS OF AXISn
Syntax	IN Q2 = STATUS OF AXIS1
Options	TAB Q
	F1 F2 F3 F4 F5 F6

Description

The **INPUT STATUS** statement assigns to the **Q** variable the current status of the axis specified. The value is assigned according to the table below.

Not Moving	0
Moving	1
CCW Limit	2
CW Limit	4
Stalled	8
Wait DBND	16

If more than one status is active at one time, the sum is assigned. If you enable or disable hardware limits with the **ENABLE HLIMIT** statement, you may need to add a minor program delay if you check the status of the axes right after updating the limits. This will ensure a correct status representation.

Use the **IF Q1** or an = (1, 2, 4 ...) statement to determine if an individual event has occurred.

Example

```
Statement                          Description
MOVI 200000 * * *                  'Starts moving axis 1
LOOP 0                             'Begins continuous loop
IN Q2 = STATUS OF AXIS1            'Sets variable Q2 as input status of
                                   'the Model 4000
IF Q2 = 0 GOTO NOT_MOVE            'If Q2 is 0, jump to NOT_MOVE routine
IF Q2 = 1 GOTO IS_MOVE             'If Q2 is 1, jump to IS_MOVE routine
IF Q2 = 2 GOTO CCWLIMIT            'If Q2 is 2, jump to CCWLIMIT routine
IF Q2 = 8 GOTO STALLED             'If Q2 is 8, jump to STALLED routine
ENDLOOP
```

IN DATA

Name	IN DATA
Descriptor	Get the Next Data Element
Type	Programming
Default	IN Qn = NEXT DATA
Syntax	IN Q1 = NEXT DATA
Options	TAB Q
	F1 F2 F3 F4 F5 F6

Description

The **IN DATA** statement assigns to the **Q** variable the next **DATA** statement value. *The **DATA** statement does not need to be placed before the use of the **IN DATA** statement.*

The **RESTORE** statement resets the data statement pointer so that the next **IN DATA** statement will assign the very first **DATA** statement's value to a **Q** variable.

*If the **Qn** variable in the **INPUT Qn = NEXT DATA** statement is encountered during execution after all the **DATA** statement constants have been used up, the program will abort, and the error message **OUT OF DATA FOR THE IN Qn = NEXT DATA** will be displayed.*

See Also: **DATA, RESTORE**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example

The program below shows the use of the **DATA** statement to move axis #1 50 steps, 44 steps , and then 69 steps. The program resets the **DATA** statement to the first constant 50.

Statement	Description
`DATA 50 44 69 99`	`'Defines 4 data points`
`MATH Q4 = 99`	`'Defines ending data point`
`LABEL MOVE_LP`	`'Defines label move`
`IN Q1 = NEXT DATA`	`'Sets input data value to Q1`
`IF Q1 = Q4 GOTO ITS_DONE`	`'If ending data point, end program`
`MOVE Q1 * * *`	`'Moves axis 1 Q1 distance`
`GOTO MOVE_LP`	`'Goes back to move LP label`
`RESTORE`	`'Restores data pointer to first data point`
`LABEL ITS_DONE`	`'Defines ITS-DONE label`
`DONE`	`'Ends the program`

IN BIT

Name	**IN BIT**
Descriptor	Get Status of 1 of 24 Inputs
Type	Programming
Default	IN Qn = BITn
Syntax	IN Q3 = BIT1
Options	TAB Q
	F1 F2 F3 F4 F5 F6

Description

The **IN BIT** statement assigns to the **Q** variable either a 1 or a Ø depending on the state of the specified programmable input bit and on the **DEFINE IN24** statements value of positive or negative logic.

MSB (pin #1) (pin #47) **LSB**

⇑ ⇑

ØØØØ ØØØØ ØØ11 1ØØ1 Ø111 Ø1ØØ

Most Significant Bit (MSB), bit #24, is on pin 1 and the least significant bit (LSB), bit 1 is on pin #47. The value of n can be 1 to 24.

See Also: **DEFINE IN24, IN IN24, ON BIT, ON IN24, IF BIT, IF IN24, PATT**

Example

Statement	Description
`IN Q1 = BIT3`	`'The Q1 variable becomes the 'value of the status of input 'bit 3`
`OUT LCD2,Ø1 ^THE BIT 3 STATUS^ Q1`	`'Displays status of input bit 3`

IN TRIG

Name	**IN TRIG**
Descriptor	Get Status of 1 of 4 Trigger Inputs
Type	Programming
Default	IN Qn = TRIGn
Syntax	IN Q3 = TRIG4
Options	TAB Q
	F1 F2 F3 F4 F5 F6

Description

The **IN TRIG** statement assigns to the **Q** variable either a Ø or 1 depending on the state of the specified axis trigger input. There are four axis inputs, one on each axis connector. The **DEFINE** statement does not affect the logic level of the trigger inputs. Positive logic is always used.

See Also: **IF TRIG, ON TRIG**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example

```
Statement                          Description
IN Q1 = TRIG3                      'The Q1 variable is assigned
                                   'either a 1 or a 0
OUT LCD2,01 ^TRIG3 IS ^ Q1         'Display the status of TRIG3
                                   'input
```

IN ANALOG

Name	IN ANALOG
Descriptor	Read Analog Inputs
Type	Programming
Default	IN Qn = ANALOG CHANn
Syntax	IN Q1 = ANALOG CHAN2
Options	TAB Q
	F1 F2 F3 F4 F5 F6

Description

The **IN ANALOG** statement reads the analog input channel specified, and stores that value in the variable specified (**Q1 - Qnn** or pointer variable **QQ1 - QQ99**). The input voltage range is 0V to 2.5V. The analog inputs are on pins 2, 3, 4, and 5 of auxiliary connector #2. The resolution of analog voltage is 0.01V. Therefore, the voltage you read will be between 0.01V and 2.5V.

See Also: **ENABLE JOY, ENABLE VSCALE, ENABLE TCH_JOY**

IN ERROR

Name	IN ERROR
Descriptor	Store Error Code in a Variable
Type	Programming
Default	IN Qn = ERROR CODE
Syntax	IN Q1 = ERROR CODE
Options	TAB Q
	F1 F2 F3 F4 F5 F6

Description

The **IN ERROR** statement will take the most recent execution error value and store it into the variable specified. The error value can be zero through 63, where zero means no error has occurred since the or the last error was read in. The variable specified can range from Q1 through Qnn, or what ever maximum numeric variable is configured via the **DEFINE VAR** statement.

The **IN ERROR** statement is useful when executed after the destination of an **ON ERROR** statement. If no **ON ERROR** statement is executed, the program will stop when an error occurs and the error message is displayed on the LCD. If an **ON ERROR** statement is executed, the program will not abort but instead will branch to the label destination specified in the **ON ERROR** statement. If an **IN ERROR** statement is executed, the type of error can be determined by the program and corrective action taken. For instance, if an **IN LCD** statement is executed and the Model 4000 receives incorrect data the Model 4000 would stop the program and display error message #28 (see below). If an **ON ERROR** statement had been executed, then at the **ON ERROR** destination error #28 could be checked for via the **IN ERROR** and **IF Q** statements. The program could then branch back to that **IN LCD** statement and ask for the data again (perhaps with another message explaining that data should be entered).

See Also: **ON ERROR**

The list below are the possible errors the Model 4000 can detect.

```
Error message  1 is:   ERROR: No error message available.
Error message  2 is:   ERROR: No IEEE488 card to write to.
Error message  3 is:   ERROR: VEL SCALE Q var too large.
Error message  4 is:   ERROR: JOG and JOY enabled on same axis
Error message  5 is:   ERROR: Too many nested GOSUBs, max = 12
Error message  6 is:   ERROR: RETURN before a GOSUB call made.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
Error message  7 is:   ERROR: IF THEN indirect Q index > 99.
Error message  8 is:   ERROR: IF THEN AND with a negative num.
Error message  9 is:   ERROR: IF THEN OR  with a negative num.
Error message 10 is:   ERROR: IN indirect Q index > 99.
Error message 11 is:   ERROR: IN from non- existent port.
Error message 12 is:   ERROR: IN DATA:  used up or got bad data.
Error message 13 is:   ERROR: Too many nested LOOPs: max = 8
Error message 14 is:   ERROR: ENDLOOP before a LOOP statement.
Error message 15 is:   ERROR: MATH indirect Q index > 99.
Error message 16 is:   ERROR: MATH Subtraction underflow.
Error message 17 is:   ERROR: MATH Addition overflow.
Error message 18 is:   ERROR: MATH Multiplication overflow.
Error message 19 is:   ERROR: MATH Division overflow.
Error message 20 is:   ERROR: Indirect Q index > 99.
Error message 21 is:   ERROR: MOVE distance too large.
Error message 22 is:   ERROR: Data too large.
Error message 23 is:   ERROR: SEG distance data negative.
Error message 24 is:   ERROR: SEG distance data too large.
Error message 25 is:   ERROR: ACCEL VSCALE data > 50000.
Error message 26 is:   ERROR: OUT LCD Indirect Q index > 99.
Error message 27 is:   ERROR: MATH radian value > + or - 100.
Error message 28 is:   ERROR: data for IN statement is invalid
Error message 29 is:   ERROR: ENCO ERES value > 32767.
Error message 30 is:   ERROR: OUT SPOLL variable's data > 31.
Error message 31 is:   ERROR: IN PORTn already in IMMED mode.
Error message 32 is:   ERROR: MATH SQRT input was negative.
Error message 33 is:   ERROR: MATH  AND input was negative.
Error message 34 is:   ERROR: MATH   OR input was negative.
Error message 35 is:   ERROR: MATH  XOR input was negative.
Error message 36 is:   ERROR: MATH  NOT input was negative.
Error message 37 is:   ERROR: OUT OUT24 data too large.
Error message 38 is:   ERROR: OUT POB data too large: > 15.
Error message 39 is:   ERROR: Velocity for PATH is zero.
Error message 40 is:   ERROR: Radius for PATH is zero.
Error message 41 is:   ERROR: PATH radius is too small.
Error message 42 is:   ERROR: PATH endpt & center pt radius >tol
Error message 43 is:   ERROR: PATH END executed before DEF.
Error message 44 is:   ERROR: PATH name is not compiled.
Error message 45 is:   ERROR: No more storage space for PATHs.
Error message 46 is:   ERROR: A PATH axis is not ready.
Error message 47 is:   ERROR: Too many PATH DEFs: 100 is max.
Error message 48 is:   ERROR: PATH EXECUTE already given.
Error message 49 is:   ERROR: PATH DEF axis number invalid.
Error message 50 is:   ERROR: No PATH motion segments defined.
Error message 51 is:   ERROR: PATH P_RATIO > +/- 1,000.
Error message 52 is:   ERROR: Big Ram not installed in U15,16
Error message 53 is:   ERROR: PATH RAD tolerance is negative.
Error message 54 is:   ERROR: PATH UNCOMP label not found.
Error message 55 is:   ERROR: PATH must be within DEF & END.
Error message 56 is:   ERROR: PATH can not be within DEF&END.
Error message 57 is:   ERROR: NO SPACE FOR DEFINE VAR request.
Error message 58 is:   ERROR: ON TIME value > 429496.729 msec.
Error message 59 is:   ERROR: Zero vel (M17 Vn) for G00 move.
Error message 60 is:   ERROR: GCODE mode incorrect for data.
Error message 61 is:   ERROR: Invalid GCODE time data.
Error message 62 is:   ERROR: Absolute encoder read error.
Error message 63 is:   ERROR: Absolute encoder not connected.
Error message 64 is:   ERROR: Zero vel Sn*Fn for G01,G02,G03.
Error message 65 is:   ERROR: IMMED can NOT execute this line.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

LABEL

Name	LABEL
Descriptor	Define a Label
Type	Programming
Default	LABEL LABELØ
Syntax	LABEL LABEL1
Options	TAB ALPHA
	F1 F2 F3 F4 F5 F6

Description

The **LABEL** statement defines a destination within the program. Other statements, such as **GOTO**, use the label destination for program branching.

If entering labels via the front panel, the **ALPHA** option over **F3** places characters (letters) and punctuation symbols at the cursor position, starting with the letter **A**, the up and down arrow keys then scroll through the alphabet. In this way, you can build any alphanumeric label name on the line. The Model 4000 provides a default label name **LABELØ**, which can be modified according to the following rules:

- Labels can be a maximum of 8 characters and a minimum of 1 character.
- The first character must be a letter.
- The remaining seven characters can be any combination of letters, numbers, or punctuation marks.

A **LABEL** statement can not be given the same name as a program name.

See Also: **IF**, **ON**, **GOTO**, **GOSUB**

Example

```
Statement                              Description
MOVE 100000 20000 0 0                  'Moves axes 1 & 2
GOSUB MOVEY                            'GOSUB MOVEY
•
•
•
LABEL MOVEY                            'Defines label MOVEY
•
•
•
RETURN                                 'Returns program control to line 3
```

LINT

Name	LINT
Descriptor	Enable/Disable Linear Interpolation
Type	Programming
Default	LINT MODE * * * *
Syntax	LINE MODE YES NO * NO
Options	TAB YES NO NULL
	F1 F2 F3 F4 F5 F6

Description

The **LINT** statement enables or disables Linear Interpolation mode and defines which axes will participate in linearly interpolated moves. Once this mode is enabled, the Model 4000 treats **MOVE** statements in a different way than standard. For the axes enabled by the **LINT** statement, the **MOVE** statement uses the distance specified (constants or **Q** variables) as the end point of a linear interpolated move. Only axes which are enabled for **LINT MODE** and have a constant or a **Q** variable as a distance will participate in the linearly interpolated move. Other moves (e.g., **SLEWCW**, **HOME**, etc.) will be performed normally, ***even if that axis has been enabled for LINT MODE.***

Axes participating in **LINT** mode have their individual velocities and accelerations calculated by the Model 4000 so that the load is moved *in a straight line* at the path velocity and acceleration entered (assuming the motors are mounted perpendicular).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

In other words, the qualifications for an axis to participate in a **LINT** move are:

① The axis has been selected with the **LINT MODE YES** statement.

② The **MOVE** statement has a constant or **Q** variable for that axis.

To calculate the individual axis' velocity and acceleration from the **VEL PATH** and **ACCEL PATH** values entered, the **MOVE** statement values are used. These **MOVE** statement values are the endpoint of the **LINT** move. For example, a **VEL PATH 50000** and an **ACCEL PATH 250000** is entered. Also, assume 3 of the 4 possible axes are participating in this move (**LINT YES NO YES YES**). When the **MOVE 25000 50000 200000 100000** is executed, the following calculation occurs.

$$VEL_{AXIS\,1} = \left[\frac{25000}{\sqrt{((25000^2)+(200000^2)+(100000^2))}}\right] \times 50000$$

$$ACCEL_{AXIS\,1} = \left[\frac{25000}{\sqrt{((25000^2)+(200000^2)+(100000^2))}}\right] \times 250000$$

The other axes **VEL** and **ACCEL** values are calculated similarly. When the **MOVE** statement is executed, all axes start at the same time at their accelerations and all reach maximum velocity at the same time. All axes will remain at constant velocity for the same amount of time and then they will start decelerating at the same time, at their calculated rates. If these motors are all mounted orthogonally, the load will travel a straight path the entire move. This could be the diagonal path through a cube from bottom to top, if the move distances for all axes are the same.

The axes participating in a **LINT** move may be in any of the 4 distance modes: motor incremental, motor absolute, encoder incremental, or encoder absolute. Since encoder moves base their acceleration upon the number of encoder steps received, it is possible that a non-linear **LINT** move will be performed if the encoder feedback is non-linear (i.e., the load has great inertia) from some of the axes. Thus an optimum **LINT** move would be achieved if there were not load effects on encoder feedback, and a non-optimum **LINT** move would be achieved if one axis were in encoder mode with a heavy load and another axis in motor step mode.

⇨ **Note** The **MOVI** statement should not be used for **LINT** mode motion.

See Also: **ACCEL PATH, DECEL PATH, VEL PATH, UNIT PATH VEL, UNIT PATH ACCEL**

Example

In the program example below, axes #1, #3, and #4 will perform a linearly interpolated move. Their movement will start at the same time, reach max velocity at the same time, start decelerating at the same time, and stop at the same time. Axis #2 will perform a move during the linearly interpolated move, however, it's motion will not be synchronized with the other axes.

Statement	Description
LINT MODE YES NO YES YES	
ACCEL PATH 50000	
DECEL PATH 500000	
VEL PATH 250000	
MOVE 25000 50000 200000 100000	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

LOOP

Name	**LOOP**
Description	Repeat a Series of Statements
Type	Programming
Default	LOOP n
Syntax	LOOP 12
Options	TAB Q
	F1 F2 F3 F4 F5 F6

Description

The Model 4000 repeats the instructions that occur between the **LOOP** and **ENDLOOP** statements a specified number of times. You must always use the **LOOP** and **ENDLOOP** statements together. The Model 4000 allows you to nest loops. That is, you can have a loop within a loop, nested up to eight levels. The variable **n** represents the number of times to repeat the statements within the loop.

The number **n** cannot exceed 65,535. If **n** equals zero, the Model 4000 will repeat the group of statements forever. You may use a variable (**Q1-Q99**), but its value must be below 65,536.

See Also: **BREAK**, **ENDLOOP**

Example

In the example below, a 25,000-step/rev motor connected to axis #1 will move one revolution and then wait for one second. This sequence will be repeated 10 times; then the Model 4000 will set output bit 16, leaving the state of all other output bits unchanged.

Statement	Description
PATT1 XXXX XXXX XXXX XXX1 XXXX XXXX	
LOOP 10	'Loop 10 times
MOVE 25000 0 0 0	'Move axis 1
WAIT FOR 1 SECOND	'Wait 1 second after move
ENDLOOP	'Ends loop
OUT OUT24 PATT1	
DONE	

The following is an example of how ***not*** to use the **LOOP** statement: *Never* branch out of a **LOOP** if you do not return to the **LOOP** or if you do not clear the **LOOP** and **GOSUB** with the **BREAK** statement. A better programming alternative is to use **LABEL** and **GOTO LABEL** statements to create an endless loop.

Statement	Description
ON F-KEY1 GOTO F-KEY1	'Incorrectly
LOOP Ø	'use the LOOP statement
ENDLOOP	'to wait for the ON statement
LABEL F-KEY1	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MATH

Name	**MATH**
Descriptor	Mathematic Calculations
Type	Programming
Default	MATH Qn = Ø + Ø
Syntax	MATH Q12 = Q12 + 5 or MATH Q50 = SIN 90 DEG

Options						
	TAB	Q	+	-	*	ETC
	TAB	=	/	SQRT	AND	ETC
	TAB	OR	XOR	NOT	SIN	ETC
	TAB	COS	TAN	ARCSIN	ARCCOS	ETC
	TAB	ALPHA				ETC
	F1	**F2**	**F3**	**F4**	**F5**	**F6**

Description

The Model 4000 can process numbers and variables. The **MATH** statement allows you to specify parameters with numeric variables instead of constants and perform arithmetic operations on these variables. You can use this statement to dynamically alter the distance traversed, the number of times a loop is executed, the length of a time wait, etc. You can define numeric variables dynamically during program execution with the **IN** statement or with the remote interfaces **VARI** statement. The variables can then be scaled, offset, converted, etc. with the **MATH** statement. The **MATH** statement allows you to perform the operations shown in the table below.

Operator	Function	Operator	Function
+	Addition	XOR	Logical exclusive OR
-	Subtraction	NOT	Logical NOT
/	Division	SIN	Sine
*	Multiplication	COS	Cosine
SQRT	Square root	TAN	Tangent
AND	Logical AND	ARCSIN	Inverse sine
OR	Logical OR	ARCCOS	Inverse cosine
No Operator	Set a variable with numeric or alpha value		

When using the trigonometric functions **SINE** and **COSINE**, the result is accurate to ±0.0001 degrees or radians (it may not be as accurate when using the **TANGENT** function). The **TANGENT** function divides the results from the **SINE** and **COSINE** functions, so the error in the **TANGENT** function varies with the angle. The error is greater for angles approaching 90° because the **COSINE** function has greater error. Angles between 89.9980° and 90.0000° will result in a **COSINE** value of zero, therefore the tangent value is invalid, and will result in an execution error. When using angles given in radians, the value 1.5708 result in a **COSINE** value of zero and a tangent value that is invalid. The maximum angle for input is ±879,609,302.22078° or ±100.00 radians. The units for trigonometric functions can be in radians or degrees. The **DEG** or **RAD** specifier is the last item on the program line.

When using the trigonometric functions **ARCSIN** and **ARCCOS**, the result is accurate to ±0.degrees or ±0.0001 radians (the valid range for **ARCSIN** and **ARCCOS** operands is ±1.0000).

The logical functions **AND**, **OR**, **XOR**, and **NOT** truncate the fractional portion and ignore the sign of the operands. The output of the logical function will not exceed 16,777,215 or ØFFFFFFH in hexadecimal, regardless of what the input operands were. This will always be positive. This allows the result to be output to the 24 programmable outputs without sending a value that is too large.

By default up to 99 variables, labeled **Q1** through **Q99**, are available. This number can be changed with the **DEFINE VAR** statement. Each variable has the range of ±829,609,302.2207. Variables can be used by a variety of statements (**MOVE**, **VEL**, **ACCEL**, **OUT**, **IN**, **LOOP**, **WAIT**, **IF**, **MOVI**, etc.), but

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

they should be defined with a **MATH**, a remote interface **VARI**, or an **IN** statement prior to their use in one of those statements.

String variables are those that can store letters, numbers, and punctuation symbols as text data. These can be used as prompt messages on the LCD or the remote ports. The **DEFINE VAR** statement configures the number and size of string variables. The ALPHA option of the **MATH** statement allows letters and numbers to be stored, use the ^ symbol to mark the start and finish of the string. Refer to the **IN** statement and the remote interface **VARI** function.

You can read variables into the Model 4000 *on-the-fly* while the program is executing by entering the variables remotely via the front panel or remote port. For example, VARI Q1 = 947.1982<cr> will set variable Q1 while a program is executing.

MATH Example

In the example below, Qn is the variable **Q1** through **Q99**, operand #1 and operand #2 are numbers or other variables, and operator is one of those listed in the above table. Press the **ETC** function key to display all the operators that can be used with this statement. *The operator and operand #2 are not required for all forms of this statement.*

There must be a space on either side of the operator for the Model 4000 to interpret the statement correctly.

```
Statement                              Description
IN Q1 = LCD2,01 ^PARTS^                'Q1 is entered from the panel
MATH Q6 = -Q1                          'Q6 is set to -Q1
MATH Q4 = Q1 * 500                     'Q4 is set to Q1*500
MATH Q5 = Q4 / Q6
MATH Q6 = Q6 + 1000
MATH Q10 = Q4 - Q1
MATH Q100 = ^NUMBER OF PARTS= ^        'Q100 is set to: NUMBER OF PARTS =
LABEL TRNCATQ1                         'Routine to truncate, or remove the
                                       'fractional part of real-number Q1
MATH Q2 = 16777215
MATH Q3 = Q1 AND Q2
MATH Q1 = Q3
LABEL BIN_COMP                         'Routine demonstrating binary
                                       'comparisons
MATH Q21 = 15                          'Q21 is set to 15, (1111 in binary)
MATH Q22 = Q21 AND 7                   'Q22 is set to the result of 01111 and
                                       '0111 (0111 in binary) Q22 is set to 7
MATH Q29 = 27 OR 16                    'Q29 is set to 23, 16 is 010000 and 7
                                       'is 0111 the result is 010111 or 23.
LOOP Q29                               'Use Q29 as a loop counter
MOVE Q1 Q6 0 0                         'Use Q1 and Q6 as the endpoints of a
                                       'move
WAIT FOR Q11 SECONDS                   'Use Q11 as a time counter
ENDLOOP
```

*Numeric variables must be defined with the **MATH** or **IN** functions before they are used by the rest of the program.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MODE

Name	MODE
Descriptor	Mode of Positioning
Type	Set-Up
Default	MODE * * * *
Syntax	MODE M_INC M_ABS E_INC E_ABS
Options	TAB NULL M_INC M_ABS E_INC E_ABS
	F1 F2 F3 F4 F5 F6

Description

The **MODE** statement allows you to select either absolute or incremental positioning for each axis at any point in the program. The choice of absolute or incremental positioning affects the meaning of the distance specified for the **MOVE** and **MOVI** statements. For axes where incremental positioning is specified, the distance on the **MOVE** or **MOVI** parameter line is the distance and direction to move away from the current position. For axes in which absolute positioning is specified, the distance on the **MOVE** and **MOVI** parameter line is the position relative to the zero reference position established on power up or by the **PDEF** function.

The **MODE** statement also determines whether distance units (before multiplication by each axis' **UNITS** multiplier) are to be specified as motor steps or encoder steps. *If you select encoder steps as the distance units with the **MODE** statement, remember that the Model 4000's quadrature detector multiplies the encoder's standard line rate by 4, so a 1,000-line encoder produces 4,000 steps/rev when you use it with a Model 4000.*

Example

The example shows a setup that configures Axes 1, 2, and 3 as open loop, with motor steps as distance units (before multiplication by the **UNITS** multiplier) and absolute positioning relative to each axis' home position. Axis #4, equipped with an encoder, interprets all distance entries to be encoder counts after quadrature detect and operates with an incremental coordinate system.

Statement	Description
`MODE M_ABS M_ABS M_ABS E_INC`	`'Axis 1, 2, 3 motor absolute mode.` `'Axis 4 encoder incremental.`

The table below defines the descriptors on the second line of the example.

Label	Description
*	No change
M_INC	Motor steps, incremental positioning
M_ABS	Motor steps, absolute positioning
E_INC	Encoder steps (post-quadrature), incremental positioning
E_ABS	Encoder steps (after quadrature), absolute positioning

MOVE

Name	MOVE
Descriptor	Start a Move
Type	Motion
Default	MOVE * * * *
Syntax	MOVE 0 STOP Q23 *
Options	TAB STOP SLEWCW KILL SLEWCCW ETC
	TAB HOMECCW NULL HOMECW Q ETC
	TAB RESUME SEGCW SEGCCW ETC
	F1 F2 F3 F4 F5 F6

Description

The **MOVE** statement allows you to specify for preset moves the distance to travel or the end point of the move, for continuous moves (**SLEW**) the initial direction of that move and for **HOME** moves the initial direction of that move. Also any of the previous move types may be **STOPPED** or

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

KILLED once started, or RESUME once STOPPED. The four fields of the MOVE statement imply parameters for axis #1 - axis #4. The program will not execute the next line until all the MOVE parameters specified have completed, with the exception of SLEW moves.

Constant or Variable

Preset moves are those that have a constant or a variable (Q1 - Q99) as the parameter. If the MODE M_INC or E_INC statement has been issued, then the move parameter is the incremental distance to travel from the current position. If the MODE M_ABS or E_ABS statement has been issued, then the move parameter is the absolute position to which to travel. The absolute position is zero on power-up, and can be set to any value using the PDEF statement. Program execution is suspended until the entire preset move is completed, then the next statement is executed. The preset maximum move distance (after the MOVE parameter is scaled using the UNIT POS entry) is +99,999,999 steps.

Move Velocity

Refer to the chart under ENCO MRES for valid velocity ranges determined by the VEL RANGE statement. The velocity increment for low speed ranges should be noted when the desired velocity will change from very high to very low in an application, or when the velocity accuracy's critical.

Move STOP

The STOP parameter causes a move on an axis to decelerate using the DECEL value if specified or the ACCEL value if not specified. This can be executed to stop any move in progress. Also, if an ON statement becomes true, any move parameter can be stopped using a MOVE STOP statement. Program execution is suspended until the motors are completely stopped.

A preset move functions differently if the ENABLE TCH_JOY or TCH_JOG statement has been executed. See the descriptions of those statements for a complete description of their functions.

Move SLEW

The SLEWCW or SLEWCCW parameters initiate a continuous move in the direction specified. Continuous moves are only stopped by the MOVE STOP, MOVE or KILL statements or the front panel stop key (if not disabled with an ON STOPK statement) or program completion. The front panel stop key and program termination both issue a MOVE stop statement on all four axes. A continuous move accelerates at the ACCEL parameters until the VEL parameter is reached, then it remains at the VEL velocity until a new VEL is executed and another MOVE SLEWCW or SLEWCCW, or the move is stopped as described above. Program execution is suspended until the ACCEL portion of the SLEW move is complete, then the next statement is executed. Once the MOVE SLEW reaches constant velocity, the next statement is executed. However, this next statement should not be a MOVE RESUME, MOVE HOME, PRESET MOVE, or SHTDN because they are ignored while the MOVE SLEW is in progress.

To change the velocity of a MOVE SLEW, enter a VEL statement with the next velocity to change to, then enter another MOVE SLEW statement to initiate the change to that new velocity. The acceleration of a MOVE SLEW value is changed similarly with the ACCEL statement.

Move KILL

The KILL parameter shuts off pulses for the axis specified. The KILL parameter can be used in the same situations as the STOP parameter. Note that a motor is likely to stall if this statement is issued while the motor is traveling with a heavy load and/or very fast.

Move HOMECCW

The HOMECCW option initiates a go home sequence in the counterclockwise direction. The home definition is done with the DEFINE HOME statements. Program execution is suspended until its HOME move is complete.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Move HOMECW — The **HOMECW** option initiates a go home sequence in the clockwise direction. The home definition is done with the **DEFINE HOME** statement. Program execution is suspended until its **HOME** move is complete.

Move RESUME — The **RESUME** option completes a preset move or restarts a **SLEW** move. The **RESUME** option will only resume a move that was stopped by the **MOVE STOP** statement, the front panel **STOP** key, or the remote terminal **STOP** command. Also, **MOVE RESUME** is executed when the front panel **RESUME** key is pressed or if **RESUME** is entered over the remote interface. *A **MOVE HOME** statement that is stopped and then **RESUME**d will start the **MOVE HOME** sequence from the beginning.* Program execution is suspended until all **RESUME**d moves have completed, with the exception of **SLEW** moves, which only complete the acceleration phase before the next statement is executed.

Move SEGCW — The **SEGCW** option initiates a complex CW move. The complex move is defined using the **SEG** statement. Program execution is suspended until the entire complex move is complete.

Move SEGCCW — The **SEGCCW** option initiates a complex CCW move. The complex move is defined using the **SEG** statement. Program execution is suspended until the entire move is complete. Refer to *Chapter 5, Application Design* for more information on Home parameters.

See Also: **MODE, MOVI, DEFINE HOME, SEG, ACCEL, VEL**

Example

Statement	Description
`ACCEL * 20000 30000 40000`	`'Sets acceleration.`
`VEL 20000 20000 30000 50000`	`'Sets velocity.`
`ACCEL HOME 10000 * * *`	`'Set home accel for axis 1`
`VEL HOME 20000 * * *`	`'Set home velocity for axis 1`
`MOVE HOMECW 20000 50000 0`	`'Moves axis 1 to home velocity, 'moves axis 2 20,000 steps, and 'moves axis 3 50,000 steps.`

MOVI

Name	**MOVI**
Descriptor	Start a Move Don't Wait for its Finish
Type	Motion
Default	MOVI * * * *
Syntax	MOVI Q12 KILL STOP SLEWCCW

Options

TAB	STOP	SLEWCW	KILL	SLEWCCW	ETC
TAB	HOMECCW	NULL	HOMECW	Q	ETC
TAB	RESUME	SEGCW	SEGCCW		ETC
F1	**F2**	**F3**	**F4**	**F5**	**F6**

Description

This statement is identical in all respects to the **MOVE** statement except that the next program statement is executed immediately after the statement is executed. The **MOVI** statement allows the Model 4000 to begin a move and then proceed to the next instruction without waiting for preset moves to be completed or slewing moves to reach their programmed velocity.

The **MOVI** statement includes the **KILL** option, which cuts off step signal pulses (infinite deceleration). The **MOVI** statement includes the **HOMECCW** and **HOMECW** option, which set the initial go home direction and begins the go home move.

MOVI's can be buffered by executing that statement within a loop, or issuing several in rapid succession without using the **WAIT FOR MOVES** statement. If you wish to have a controlled deceleration ramp and dump all remaining **MOVI**'s from the buffer, you can issue a **MOVE STOP** and then immediately **MOVE KILL** statement within the program.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

☛A word of caution

A word of caution to users programming with **MOVI**'s: If several **MOVI**'s are buffered up and a **MOVE** or **MOVI STOP** command is issued, the only **MOVI** that will stop is the one currently executing. ***Other MOVI's in the buffer will execute.*** To clear the buffer of remaining **MOVI**'s, a **MOVE** or **MOVI KILL** statement should be executed.

WARNING

Use of the MOVI statement in any tight loop structure may cause a delayed response to the STOP key. Use of the MOVE statement does not have this problem. Below are two examples of the use of MOVI which should be avoided:

Example 1:
Incorrect usage

```
LABEL MLOOP1
MOVI SLEWCW 25ØØØØ * *
GOTO MLOOP 1
```

Example 2:
Incorrect usage

```
LOOP 2ØØØ
MOVI SLEWCW 25ØØØØ * 5ØØØØ
ENDLOOP
```

☛Note: **MOVI** should not be used for linearly interpolated motion (**LINT MODE**)

If **MOVI**'s are in the buffer and one of the following commands are issued, the Model 4000's program execution will halt at the command until all **MOVI**'s are completed, then the statement will execute and the program will continue with the next command: **MOVE** (any parameter except **STOP** or **KILL**), **PDEF**, **MOVE**, **WAIT FOR MOVES**.

*In Absolute mode, a **MOVI** Ø moves the motor to position Ø. In incremental mode, **MOVI** Ø will not move the motor. **MOVI** * should be used in Absolute mode if no move is desired.*

See Also: **MOVE, DEFINE HOME, SEG, ACCEL, VEL, MODE**

Example

Correct Usage:

Statement	Description
`MOVI 200000 * * *`	`'Starts moving axis 1.`
`LOOP Ø`	`'Begins continuous loop.`
`IN Q2 = STATUS OF AXIS 1`	`'Sets variable Q2 as input status of 'the indexer.`
`IF Q2 = 1 GOTO MOVING`	`'If Q2 is 1, jump to MOVING routine.`
`IF Q2 = 2 GOTO CW_LIM`	`'If Q2 is 2, jump to CW_LIM routine.`
`IF Q2 = 8 GOTO STALLED`	`'If Q2 is 8, jump to STALLED routine.`
`IF Q2 = 0 GOTO STOPPED`	`'If Q2 is 2, jump to STOPPED routine.`
`ENDLOOP`	

ON

Name	**ON**
Descriptor	Interrupt on Condition True
Type	Programming
Default	ON GOTO LABELØ ON GOSUB LABELØ
Syntax	ON F-KEY1 GOTO LABEL

Options

TAB	IN24	BIT	TRIG	JOYTRIG	ETC
TAB	SLIMIT	HLIMIT	ESTOP	ERROR	ETC
TAB	MENUREC	STOPK	STARTK	F-KEY	ETC
TAB	STALL	FAULT	Q	EABSLIM	ETC
TAB	M_INC	M_ABS	E_INC	E_ABS	ETC
TAB	TIME				ETC
F1	**F2**	**F3**	**F4**	**F5**	**F6**

Description

The **ON** statement initiates continuous checking for a specified condition. The Model 4000 continues to check for the **ON** condition while it executes subsequent statements. If the **ON** statement is true, and a interrupt is generated, program execution continues at the destination specified in

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

the ON statement. WAIT, IN, JOY, and JOG modes are aborted before an ON statement branch is taken. *No other statements are interrupted by an ON statement, including MOVE statements. If you want to abort a move due to an ON statement becoming true you must issue a MOVE STOP or MOVE KILL at the ON statement subroutine label.*

The Model 4000 does not continue to test for an ON statement's true condition once it becomes true. If continued checking is desired, another ON statement should be placed after the label's specified destination. If the GOTO option is used, and the condition is still true, the program will loop around the ON destination indefinitely. *A GOSUB can be substituted for the GOTO.* If the GOSUB option is used and the ON statement is re-enabled within the destination subroutine, if the condition remains true, the program will loop within that subroutine without executing the RETURN statement and an execution error will eventually occur. If a GOTO or GOSUB Interrupt occurs in a subroutine, a DONE or RETURN at the end of the subroutine may send the program execution back to the original calling program. The BREAK statement will clear all previous GOSUB and LOOP statements.

When an ON condition becomes true, any type of WAIT statement will be terminated and execution will resume with the ON statement destination. If you use an ON GOSUB statement, the statement that follows the interrupted statement is executed when the RETURN is issued.

The GOSUB or GOTO label destination can also be another program name. If the GOSUB destination is a program, it does not need a RETURN statement to execute without error. For example, if program X calls program Y, a RETURN statement is not needed to return execution to program X, when program Y is finished the Model 4000 executes a RETURN statement and program X will resumes execution. Program Y can be run on its own, or you can call it up with program X.

If no ON statements are executed, you can speed up program execution by disabling ON conditions using the ENABLE ON NO statement. The Model 4000 has an ON ERROR GOTO END statement in its default power-up program. It can be disabled with the ENABLE ON NO statement.

ON statements can be individually disabled by re-executing the statement with the DISABLE option instead of the GOTO and LABEL name. To temporarily disable checking for an ON BIT statement, and re-enable checking later in the program, refer to the program section below.

Example

```
Statement                          Description
ON BIT4 = GOTO BIT4_LOW            'Enables checking for bit 4 = 0.
•
•
•
ON BIT4 = Ø DISABLE                'Disables checking for bit 4 = 0.
•
•
•
ON BIT4 = Ø GOTO BIT4_LOW          'Re-enables checking for bit 4 = 0.
```

The editor checks for missing labels upon exiting. An error message will be displayed if any labels are missing or if they appear twice in the program.

All ON conditions are checked after every statement executes and during execution of some statements. Since all statements take varying amounts of time to execute, the frequency at which ON statements are checked

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

varies. Typically, most statements (ON conditions) take 2 to 3 ms to execute. The order in which the ON statements are checked is given below. The first one found active is serviced.

ON IN24 = PATT1
•
ON IN24 = PATT20
ON STALL AXIS 1
ON STALL AXIS 2
ON STALL AXIS 3
ON STALL AXIS 4
ON STALL AXIS 4
ON SOFTLIMIT AXIS 1
ON SOFTLIMIT AXIS 2
ON SOFTLIMIT AXIS 3
ON SOFTLIMIT AXIS 4
ON BIT 1
•
ON BIT 24
ON HARDLIMIT AXIS 1
ON HARDLIMIT AXIS 2
ON HARDLIMIT AXIS 3
ON HARDLIMIT AXIS 4
ON ESTOP
ON JOYTRIGGER
ON ERROR CONDITION
ON FAULT AXIS 1
ON FAULT AXIS 2
ON FAULT AXIS 3
ON FAULT AXIS 4
ON Q1
•
ON Q10
ON TRIG1
ON TRIG2
ON TRIG3
ON TRIG4
ON MOTOR INCREMENTAL POSITION AXIS 1
ON MOTOR INCREMENTAL POSITION AXIS 2
ON MOTOR INCREMENTAL POSITION AXIS 3
ON MOTOR INCREMENTAL POSITION AXIS 4
ON MOTOR ABSOLUTE POSITION AXIS 1
ON MOTOR ABSOLUTE POSITION AXIS 2
ON MOTOR ABSOLUTE POSITION AXIS 3
ON MOTOR ABSOLUTE POSITION AXIS 4
ON ENCODER INCREMENTAL POSITON AXIS 1
ON ENCODER INCREMENTAL POSITON AXIS 2
ON ENCODER INCREMENTAL POSITON AXIS 3
ON ENCODER INCREMENTAL POSITON AXIS 4
ON ENCODER ABSOLUTE POSITION AXIS 1
ON ENCODER ABSOLUTE POSITION AXIS 2
ON ENCODER ABSOLUTE POSITION AXIS 3
ON ENCODER ABSOLUTE POSITION AXIS 4

See Also: **BREAK, GOSUB, LABEL, RETURN, IF, WAIT, GOTO**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ON IN24

Name	ON IN24
Descriptor	Interrupt on Matching 24 Inputs
Type	Programming
Default	ON IN24 = PATTn GOTO LABELØ
Syntax	ON IN24 = PATT20 GOTO LABELØ
Options	TAB DISABLE ALPHA GOSUB FND_LBL ETC TAB BEG END GOTO ETC F1 F2 F3 F4 F5 F6

Description

The **ON IN24** statement continually checks the 24 inputs against a previously defined **PATT** variable. If they match, the branch is taken. The value of **n** can be 1-20.

You can have up to 20 different **ON IN24 = PATTn** statements that are being checked for *at the same time*. The **n** number in **PATTn** specifies a unique **ON** Statement that is checked approximately every 3 milliseconds.

If the **=** is changed to an **!** the branch is taken if the 24 inputs do not match the **PATT** variable specified. Use the **ALPHA** option to get the **!** symbol when in the editor via the front panel.

The logic type (positive or negative) used in the check is determined by the **DEFINE IN24** statement.

See Also: **IF IN24, WAIT FOR IN24, IN IN24, DEFINE IN24**

ON BIT

Name	ON BIT
Descriptor	Interrupt on Matching 1 of 24 Inputs
Type	Programming
Default	ON BITn = 0 GOTO LABELØ
Syntax	ON BIT1 = 0 GOTO LABEL
Options	TAB DISABLE ALPHA GOSUB FND_LBL ETC TAB BEG END GOTO ETC F1 F2 F3 F4 F5 F6

Description

The **ON BIT** statement continually checks one of the 24 input bits for the specified state, if true the branch is taken.

There can be up to 24 different **ON BITn = X** statements that are being checked for *at the same time*. The n number in **BITn** specifies the bit number.

The logic type (positive or negative) used in the check is determined by the **DEFINE IN24** statement.

See Also: **DEFINE IN24, IF BIT, WAIT FOR BIT, IN BIT**

Example

Statement	Description
`ON BIT1 = 0 GOTO RUN`	`'When bit 1 becomes low, GOTO Run routine.`
`MOVE 1000 1000 0 0`	`'Move axes 1 & 2.`
•	
•	
•	
`LABEL RUN`	`'Defines Run routine.`
`MOVE 2000 1000 0 0`	`'Move axes 1 & 2.`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ON TRIG

Name	**ON TRIG**
Descriptor	Interrupt on Trigger Input
Type	Programming
Default	ON TRIGn = 0 GOTO LABELØ
Syntax	ON TRIG1 = 0 GOTO LABELØ
Options	TAB DISABLE ALPHA GOSUB FND_LBL ETC TAB BEG END GOTO ETC F1 F2 F3 F4 F5 F6

Description

The **ON TRIG** statement continually checks one of the four axis trigger inputs for the specified state, if true the branch is taken. The value of **n** can be 1 to 4.

The logic type is always positive logic. The **DEFINE** statement does not affect the logic type used by the **ON TRIG** statement.

Each of the four trigger inputs can be checked at the same time. The **n** in **TRIGn** specifies a unique **ON** statement to be checked (approximately every 3 ms).

See Also: **WAIT FOR TRIG, IF TRIG, IN TRIG**

ON JOYTRIG

Name	**ON JOYTRIG**
Descriptor	Interrupt on Joytrig Input Active
Type	Programming
Default	ON JOYTRIG GOTO LABELØ
Syntax	ON JOYTRIG GOTO LABELØ
Options	TAB DISABLE ALPHA GOSUB FND_LBL ETC TAB BEG END GOTO ETC F1 F2 F3 F4 F5 F6

Description

The **ON JOYTRIG** statement continually checks the joystick trigger input. If the input is active (0V, current flowing) the branch is taken.

ON SLIMIT

Name	**ON SLIMIT**
Descriptor	Interrupt on Software Limit
Type	Programming
Default	ON SLIMIT ANY AXIS GOTO LABELØ
Syntax	ON SLIMIT AXIS3 GOTO LABELØ
Options	TAB DISABLE ALPHA GOSUB FND_LBL ETC TAB BEG END GOTO ETC F1 F2 F3 F4 F5 F6

Description

The **ON SLIMIT** statement continually checks the current **MODE**, **POSITION** and **SLIMIT** parameters to determine if the load has moved past the software limit. You may erase **ANY** from the syntax and enter a specific axis (**AXIS**n) to become:

ON SLIMIT AXISn GOTO LABELØ

The value of **n** may be any number between 1 and 4.

See Also: **ENABLE M_SLIM, ENABLE E_SLIM, SLIMIT, DECEL SLIMIT**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ON HLIMIT

Name	**ON HLIMIT**
Descriptor	Interrupt on Limit Input Active
Type	Programming
Default	ON HLIMIT ANY AXIS GOTO LABELØ
Syntax	ON HLIMIT AXIS2 GOTO LABELØ
Options	TAB DISABLE ALPHA GOSUB FND_LBL ETC TAB BEG END GOTO ETC F1 F2 F3 F4 F5 F6

Description

The **ON HLIMIT** statement checks the limit inputs. If any of the inputs are active the branch is taken. You may erase **ANY** from the syntax and enter a specific axis (**AXIS**n) entered, to become:

ON HLIMIT AXISn GOTO LABELØ

The value of **n** may be any number between 1 and 4.

See Also: **ENABLE HLIMIT, DECEL HLIMIT**

ON ESTOP

Name	**ON ESTOP**
Descriptor	Interrupt on Estop Input Active
Type	Programming
Default	ON ESTOP GOTO LABELØ
Syntax	ON ESTOP GOSUB LABELØ
Options	TAB DISABLE ALPHA GOSUB FND_LBL ETC TAB BEG END GOTO ETC F1 F2 F3 F4 F5 F6

Description

The **ON ESTOP** statement continually checks the **ESTOP** input. If the input is active the branch is taken.

The statements following the **LABEL** statement specified with **ON ESTOP** should include a **MOVE STOP** or a **MOVE KILL** if move termination is desired. Otherwise, the pulses will be stopped, but the Model 4000 will continue to execute the **MOVE** statement.

The **ESTOP** input does not shut power off to the Model 4000. However, a fault output is provided that you can wire to a system power off relay device if desired.

Each time a program is started, the Model 4000 automatically executes an **ON ESTOP GOTO END** statement, which will terminate the program should an **ESTOP** occur. You must execute an **ON ESTOP** statement of your own if you wish to have a different function for the **ESTOP** input.

Example

Statement	Description
ON ESTOP GOTO TROUBLE	'Specified destination.
MOVE 25000 * * *	'Start axis 1.
•	
•	
•	
LABEL TROUBLE	'Destination code.
MOVE STOP STOP STOP STOP	'Stop motor.
OUT LCD3,01 ^ESTOP ASSERTED^	'Send warning.
WAIT FOR F-KEY1	'Continue with program when operator 'presses F-Key1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ON ERROR

Name	ON ERROR
Descriptor	Interrupt on Error Condition
Type	Programming
Default	ON ERROR GOTO LABELØ
Syntax	ON ERROR GOTO LABELØ
Options	TAB DISABLE ALPHA GOSUB FND_LBL ETC TAB BEG END GOTO ETC F1 F2 F3 F4 F5 F6

Description

The **ON ERROR** statement checks if any of several types of error's have occurred after each statement executes. The errors checked for include math overflow, division by zero, invalid data entered to an **IN** statement, a **RETURN** without a prior **GOSUB** call, a loop **Q** variable value that is too large or when using a **QQn** variable where the contents of **Qn** are not within 1 - 99 at the time of execution. The error code can be read with the **IN ERROR** statement, and then the program can prompt the operator to take corrective action (i.e., try entering the data again). For a list of all error conditions refer to the **IN ERROR** command.

If an **ON ERROR** statement is not executed, then the program will terminate when an error is detected, and the error message: **PROGRAM ABORTED. ERROR FOUND AFTER LINE** will be displayed on LCD line 2. On line 3 will be the statement after which the error was detected.

See Also: **IN ERROR**

ON MENUREC

Name	ON MENUREC
Descriptor	Interrupt on Menu Recall Pressed
Type	Programming
Default	ON MENUREC GOTO LABELØ
Syntax	ON MENUREC GOTO LABELØ
Options	TAB DISABLE ALPHA GOSUB FND_LBL ETC TAB BEG END GOTO ETC F1 F2 F3 F4 F5 F6

Description

The **ON MENUREC** statement continually checks to see if **MENU RECALL** has been pressed. If it is pressed, the branch is taken.

ON STOPK

Name	ON STOPK
Descriptor	Interrupt on Stop Key Pressed
Type	Programming
Default	ON STOPK GOTO LABELØ
Syntax	ON STOPK GOSUB LABELØ
Options	TAB DISABLE ALPHA GOSUB FND_LBL ETC TAB BEG END GOTO ETC F1 F2 F3 F4 F5 F6

Description

The **ON STOPK** statement continually checks to see if **STOP** has been pressed. If **STOP** is pressed, the branch is taken. If the **ON STOPK** statement is executed, the default function of the **STOP KEY** will not occur. This default function includes stopping the program and any statement in progress, including moves. If it is desired to stop a move when the **STOP** key is pressed, a **MOVE STOP** should be issued after the subroutine label.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ON STARTK

Name	**ON STARTK**
Descriptor	Interrupt on Start Key Pressed
Type	Programming
Default	ON STARTK GOTO LABELØ
Syntax	ON STARTK GOTO LABELØ
Options	TAB DISABLE ALPHA GOSUB FND_LBL ETC TAB BEG END GOTO ETC F1 F2 F3 F4 F5 F6

Description

The **ON STARTK** statement continually checks to see if **START** has been pressed. If **START** is pressed, the branch is taken.

ON F-KEY

Name	**ON F-KEY**
Descriptor	Interrupt on 1 of 6 F-Keys Pressed
Type	Programming
Default	ON K-KEYn GOTO LABELØ
Syntax	ON F-KEY1 GOSUB LABEL
Options	TAB DISABLE ALPHA GOSUB FND_LBL ETC TAB BEG END GOTO ETC F1 F2 F3 F4 F5 F6

Description

The **ON F-KEY** statement continually checks to see if the specified **F-KEY** is pressed. If the **F-KEY** is pressed, the branch is taken.

See Also: **IF F-KEY, WAIT FOR F-KEY**

ON STALL

Name	**ON STALL**
Descriptor	Interrupt on Stall Condition
Type	Programming
Default	ON STALL ANY AXIS GOTO LABELØ
Syntax	ON STALL AXIS1 GOTO LABELØ
Options	TAB DISABLE ALPHA GOSUB FND_LBL ETC TAB BEG END GOTO ETC F1 F2 F3 F4 F5 F6

Description

The **ON STALL** statement continually checks for a stall condition. If a stall occurs, the branch is taken.

You may erase **ANY** from the syntax and enter a specific axis **(AXISn)** to become: **ON STALL AXISn GOTO LABELØ**. The variable **n** may be specified as any number between 1 - 4.

The following statements must be executed before the **ON STALL** statement before the **ON STALL** statement will work:

- **MODE E_INC** or **E_ABS**
- **ENCO BKL** (optional)
- **ENABLE STALL YES**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ON FAULT

Name	ON FAULT
Descriptor	Interrupt on Drive Fault Interrupt Active
Type	Programming
Default	ON FAULT ANY AXIS GOTO LABELØ
Syntax	ON FAULT AXIS2 GOTO LABELØ

Options						
	TAB	DISABLE	ALPHA	GOSUB	FND_LBL	ETC
	TAB	BEG	END	GOTO		ETC
	F1	F2	F3	F4	F5	F6

Description

The **ON FAULT** statement continuously checks for the motor drive fault input and branches if any are active. A Fault Condition on any axis will not affect program execution unless programmed to do so using the **ON FAULT** statement.

☛Note: The Axis Fault Led's do not turn on in the event of an externally generated fault, even though the **ON FAULT** condition is detected. The axis fault LEDs will turn on if a program uses the **SHTDN** command to remove power from a drive in response to a fault.

A drive fault input (pin #5 of the motor drive connector) is active when 5V is present (no current flowing). *When unconnected the input is internally pulled to 5V, thus a fault will be sensed if unconnected.* The **DEFINE FAULT** statement may be used to set the fault active level.

Please refer to your motor drive documentation for a list of possible fault conditions.

Compumotor Product	Drive Fault Level
L, LE, PS7, BLX	Active Low
Compumotor Plus, LN, OEM, S, Z, PK3, and Dynaserv	Active High

See Also: **DEFINE FAULT**

ON Qn

Name	ON Qn
Descriptor	Interrupt on Qn Variable
Type	Programming
Default	ON Qn = Qn GOTO LABELØ
Syntax	ON Qn = 9993.04 GOTO LABELØ

Options						
	TAB	Q	=	<	>	ETC
	TAB	OR	AND	DISABLE	FND_LBL	ETC
	TAB	BEG	END	ALPHA	GOSUB	ETC
	F1	F2	F3	F4	F5	F6

Description

The **ON Qn** statement continually compares the first variable (Q1–Q10) to either another variable (Q1-Q99) or a constant. If the expression is true, the program branches to the label specified. The operators available are **=**, **<**, **>**, **<=**, **>=**, **<>**, **OR**, and **AND**.

Only Q1–Q10 can be specified for the first variable number. The first variable number denotes a unique **ON** condition to be searched for. Thus, as many as 10 (Q1-Q10) **ON Qn** statement will be checked *at the same time.*

When using **OR** and **AND**, *true* means that a non-zero result was obtained from the logical operation. The **OR** and **AND** logical operators truncate the fractional portion of the operands and ignore their signs (±) .

Example

Statement	Description
MATH Q1 = Ø	'Defines Q1 as 0.
MATH Q2 = Ø	
ON Q1 AND Q2 GOSUB ANDTRUE	'Not taken.
ON Q1 OR Q2 GOSUB ORTRUE	'Not taken.
MATH Q1 = 1	'Branch taken to ORTRUE after this statement 'executes.
MATH Q2 = 1	'Branch taken to ANDTRUE after this 'statement executes.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ON EABSLIM

Name	ON EABSLIM
Descriptor	Check for Absolute Encoder Roll-Over
Type	Programming
Default	ON EABSLIM ANY AXIS GOTO LABELØ
Syntax	ON EABSLIM ANY AXISn GOTO LABELØ

Options						
	TAB	DISABLE	ALPHA	GOSUB	FND_LBL	ETC
	TAB	BEG	END	GOTO		ETC
	F1	F2	F3	F4	F5	F6

Description

The Model 4000 continuously checks for an absolute encoder roll-over. If a roll-over occurs (or a roll-under) refer to *Chapter 5, Application Design.*

See Also: **ENCO ABS, ENCO MRES, ENCO ERES**

Example

Statement	Description
`ON EABSLIM AXIS1 GOSUB ABSLIM`	`'Enable roll-over detection.`
`MOVE 5ØØØØØØ * * *`	`'Move to an invalid position to illustrate roll-over detection.`
`OUT LCD3,1 ^Returned from ABSLIM^`	`'Program will reach this statement after gosub to 'ABSLIM.`
`DONE`	
`LABEL ABSLIM`	
`MOVE STOP * * *`	`'Stop the move that has roll-over.`
`MOVE -2ØØØØ * * *`	`'Return to a pre roll-over position.`
`OUT LCD2,1`	`'Roll-over error detected and corrected`
`ON EABSLIM AXIS1 GOSUB ABSLIM`	`'Re-enable roll-over checking.`
`RETURN`	

ON M_INC

Name	ON M_INC
Descriptor	Interrupt on Motor Step Incremental Position Met
Type	Programming
Default	ON M_INCn GOTO LABELØ
Syntax	ON M_INC4 > Q5 GOTO LABELØ

Options						
	TAB	>	<			
	TAB	DISABLE	ALPHA	GOSUB	FND_LBL	ETC
	TAB	BEG	END	GOTO		ETC
	F1	F2	F3	F4	F5	F6

Description

The **ON M_INC** statement continually checks the incremental position of the specified axis number (n). If the expression becomes true, the branch is taken. **M_INC** tells the Model 4000 to compare the number of motor steps sent out (since the beginning of the current move) to the number or **Q** variable specified.

When using a distance scale factor (**UNIT POS**), the motor distance in steps is divided by the distance scale factor, before the comparison is made.

See Also: **UNIT POS, IN POS**

Example

Statement	Description
`UNIT POS 25ØØ * * *`	
`MODE M_INC * * * *`	
`ON M_INC1 > 5 GOTO LABELØ`	`'This condition will branch to 'LABELØ. When the motor has 'traveled 125,000 steps.`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ON M_ABS

Name	ON M_ABS
Descriptor	Interrupt on Motor Step Absolute Position Met
Type	Programming
Default	ON M_ABSn GOTO LABELØ
Syntax	ON M_ABS1 > Q5 GOTO LABELØ

Options						
	TAB	>	<			
	TAB	DISABLE	ALPHA	GOSUB	FND_LBL	ETC
	TAB	BEG	END	GOTO		ETC
	F1	F2	F3	F4	F5	F6

Description

The **ON M_ABS** statement continually checks the absolute position of the specified axis number (n). If the expression becomes true, then the branch is taken. **M_ABS** tells the Model 4000 to compare the number of motor steps sent out (since the last **PDEF** or last power-up) to the number or **Q** variable specified.

When using a distance scale factor (**UNIT POS**), the motor distance in steps is divided by the distance scale factor, before the comparison is made.

See Also: **UNIT POS, IN POS**

ON E_INC

Name	ON E_INC
Descriptor	Interrupt on Encoder Step Incremental Position Met
Type	Programming
Default	ON E_INCn GOTO LABELØ
Syntax	ON E_INC2 > Q5 GOTO LABELØ

Options						
	TAB	>	<			
	TAB	DISABLE	ALPHA	GOSUB	FND_LBL	ETC
	TAB	BEG	END	GOTO		ETC
	F1	F2	F3	F4	F5	F6

Description

The **ON E_INC** statement continually checks the incremental position of the specified axis number (n). If the expression becomes true, the branch is taken. **E_INC** tells the Model 4000 to compare the number of encoder steps received (since the beginning of the current move) to the number or **Q** variable specified.

When using a distance scale factor (**UNIT POS**), the encoder distance in counts is first divided by the distance scale factor, before the comparison is made.

See Also: **UNIT POS, IN POS**

Example

Statement	Description
`UNIT POS 4ØØØ * * *`	
`ENCO ERES 4000 * * * *`	
`ENCO MRES 25ØØØ * * *`	
`ON E_INC1 > 5 GOTO LABELØ`	`'This condition will branch to 'LABELØ. When the encoder has gone '20,000 counts, the motor has gone '125,000 steps.`
`MOVE 10 * * *`	`'Start axis moving 40,000 encoder 'counts`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ON E_ABS

Name	ON E_ABS
Descriptor	Interrupt on Encoder Step Absolute Position Met
Type	Programming
Default	ON E_ABS GOTO LABELØ
Syntax	ON E_ABS > Q5 GOTO LABELØ

Options						
	TAB	>	<			
	TAB	DISABLE	ALPHA	GOSUB	FND_LBL	ETC
	TAB	BEG	END	GOTO		ETC
	F1	F2	F3	F4	F5	F6

Description

The **ON E_ABS** statement continually checks the absolute position of the specified axis number (n). If the expression becomes true, the branch is taken. **E_ABS** tells the Model 4000 to compare the number of encoder steps received (since the last **PDEF** or last power-up) to the number or **Q** variable specified.

When using a distance scale factor (**UNIT POS**), the encoder distance in counts is first divided by the distance scale factor, before the comparison is made.

See Also: **UNIT POS, IN POS**

ON TIME

Name	ON TIME
Descriptor	Interrupt After the Specified Time has Elapsed
Type	Programming
Default	ON TIME = n GOTO LABELØ
Syntax	ON TIME = 5.123 GOSUB TIMEOUT

Options						
	TAB	DISABLE	ALPHA	GOSUB	FND_LBL	ETC
	TAB	BEG	END	GOTO		ETC
	F1	F2	F3	F4	F5	F6

Description

The **ON TIME** statement will interrupt the program after the specified number of seconds has elapsed (from the time the **ON TIME** statement executes). The resolution of the parameter n is in ms and the units of n are seconds (i.e., n = 4.321 is 4 seconds 321 ms).

See Also: **WAIT FOR SECONDS**

OUT

Name	OUT
Descriptor	Send Out Data
Type	Status
Default	OUT
Syntax	OUT LCDn,01 ^DISTANCE AXIS1 =^ Q1

Options						
	TAB	LCD	PORT	OUT24	BIT	ETC
	TAB	POB	SPOL			ETC
	F1	F2	F3	F4	F5	F6

Description

The **OUT** statement sends variables to the LCD, RS-232C ports, or to the 24 programmable outputs, one bit of the 24 programmable outputs, or to the axes programmable outputs.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

OUT LCD

Name	**OUT LCD**
Descriptor	Send Out ASCII String and/or Variable to LCD
Type	Status
Default	OUT LCDn,01 ^your message^ Qn
Syntax	OUT LCD4,21 ^Q10 = ^ Q10 OUT LCDn,01 ^your message^ Qn OUT LCDn,01 Qn Qn

Options	TAB	ALPHA	^	Q	BLANK	
	F1	F2	F3	F4	F5	F6

Description

The **OUT LCD** statement sends an ASCII string to the front panel LCD. The value of n can be 1 through 4, it is the LCD line number. The second number is the column number the string will start at. Between the ^ symbols, you can create a prompt message using the alpha function. Two messages can be sent, the first can be the prompt message or a variable (Q1 - Q109), the second can only be a variable (Q1 - Q109). *ASCII character string variables are Q100 - Q109 by default.* If the Qn is removed from the end of the statement, only the first message is sent. The maximum length of a string between the ^ symbols is 40 (up to 22 will be displayed on the front panel while editing). A remote terminal can be used to enter strings over 22 characters.

When a variable is displayed on the LCD, fourteen characters are sent regardless of how many digits are in the number. The numeric values are left justified within those fourteen characters and the remaining characters are spaces. This allows several variables to be displayed at the same location on the LCD.

The number of digits to the right of the decimal point can be no more than four and can be selected using the **MATH** statement. Refer to Example 2 below.

See Also: **MATH, DEFINE VAR, DISPLAY, IN LCD**

Example #1

Statement	Description
`MATH Q5 = 35`	`'Defines Q5 as 35.`
`OUT LCD3,01 ^VARIABLE IS ^ Q5`	`'Prints VARIABLE IS 35 on line 3, 'column 1.`

Example #2

Statement	Description
`MATH Q1 = 99.1234`	`'This is the variable to be 'displayed.`
`MATH Q1 = Q1 + .0000`	`'This ensures Q1 is displayed with 'four digits to the right of the 'decimal. If .00 or .000 were 'added, two or three digits to the 'right would be displayed.`
`OUT LCD3,01 Q1`	`'Q1 is displayed with four digits to 'the right of the decimal.`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

OUT PORT

Name	OUT PORT
Descriptor	Send Out ASCII String and/or Variable to Remote Port
Type	Status
Default	OUT PORTn ^Q10=^ Q10 OUT PORTn ^your message^ OUT PORTn Q5 Q103
Syntax	OUT PORT1 ^Q11=^ Q11
Options	TAB ALPHA ^ Q ; $ F1 F2 F3 F4 F5 F6

Description

The **OUT PORT** statement sends an ASCII string to one of three possible ports. Ports 1 and 2 are RS-232C channels, port 3 is the optional GPIB interface.

The ASCII string can be 1 or 2 messages. The first message can be a prompt message entered using the ^ symbol and the ALPHA function, or it can be a variable (**Q1 - Q109**). The second message must be a variable (**Q1 - Q109**). If two messages are sent neither a space or a carriage return will be sent *after the first message.* After the last message sent, a carriage return and a line feed are automatically sent.

If the prompt message display is on, a **>** and a space are also sent. All four of those characters (**CR**, **LF**, **>**, (**space**)) are not sent if the ; option is placed after the first message and before the second message (if a second message is desired) i.e., **OUT PORT1 ^MESSAGE1 ^ ; Q1.**

If you are using the Compumotor *4000 Application Developer Software version 2.1* the ; parameter may only be entered in a program line from the Model 4000 Front Panel. When transferred from a remote computer during a load, all text after the ; is ignored.

Port1 or Port2 ; => no CR, LF, >, ,

Port 3 ; => no CR, LF, >, ,

$ => no CR, LF, >, , or EOI

For **GPIB** operation, the $ sign may be used. The $ option will send the first and/or second strings, without any following characters or the **EOI** signal (no **CR**, **LF**, **>**, , or **EOI** are sent or activated). For information on the default parameters of the **GPIB** interface, refer to the **Model 4000 Options User Guide** i.e., **OUT PORT3 ^MESSAGE GPIB ^ $ Q1.**

If the **Qn** is removed from the end of a **PORT** statement, only the first message is sent. The maximum number of characters between the ^ ^ symbols is 40 (up to 24 will be displayed on the front panel while editing). A remote terminal can be used to enter strings over 24 characters.

The number of digits to the right of the decimal point can be no more than four and can be selected using the **MATH** statement. Refer to Example 2 below.

See Also: **RS232, DEFINE GPIB, IN PORT, MATH**

Example

Statement	Description
MATH Q5 = 100	'Defines Q5 as 100.
OUT PORT1 ^THE PART NUM IS ^ Q5	'Prints **THE PART NUM IS 100** through serial port 1.
OUT PORT1 ^WHATS THE PART #? ^	'Prints **WHATS THE PART #?.**
OUT PORT1 Q1 ; Q3	'Sends both variables to the port, but because of the ; no carriage return, line feed, > (prompt) and space are sent.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example #2

Statement	Description
`MATH Q1 = 99.1234`	`'This is the variable to be 'displayed.`
`MATH Q1 = Q1 + .0000`	`'This ensures Q1 is displayed with 'four digits to the right of the 'decimal. If .00 or .000 were 'added, two or three digits to the 'right would be displayed.`
`OUT PORT3,01 Q1`	`'Q1 is displayed with four digits to 'the right of the decimal.`

OUT OUT24

Name	OUT OUT24
Descriptor	Set 24 Programmable Outputs
Type	Status
Default	OUT OUT24 OUT OUT24 Qn HEX n OUT OUT24 Qn BCD n n OUT OUT24 PATTn
Syntax	OUT OUT4 PATT5
Options	TAB PATT HEX BCD F1 F2 F3 F4 F5 F6

Description

The **OUT** statement controls the state (Ø or 1) of the optically isolated output bits on the 24-bit programmable output connector. The **OUT** statement is useful for triggering external events at predetermined times during program execution.

Data can be sent to the outputs from **PATT** variables or from **Q** variables. The **Q** variables data will be displayed on the 24 outputs in either the BCD format or the hexidecimal format.

You can use **Q** variables to control the bit pattern of the output bits. The format of the **OUT** statement using variables is **OUT Qn** where **Qn** is the bit pattern of the output bits. These variables must be defined as decimal numbers by the **MATH** statement before referencing them in the **OUT** statement. These decimal numbers are converted to BCD or Hexidecimal by the Model 4000 to produce the desired bit pattern and the mask. The maximum decimal number allowed for an **OUT** using the HEX option is 16,777,215 or FFFFFFH in hexidecimal. For an **OUT OUT24** using the BCD option, the maximum number is 999,999.

The default is negative logic and BCD and HEX zero on the outputs (i.e., all outputs are 5V).

The output bits can be sent to digital equipment that expects this output to be in BCD form. This equipment can be either negative logic, which sets the bits to 1 by being connected to ground, or positive logic, which sets the bits to 1 by being connected to VDC. The **DEFINE OUT24** statement allows the user to define which type of logic the data should be formatted.

The **BCD** option causes **BCD n.n** to appear. The n values refer to the number of nibbles (4 bits) to use as data to send to the 24 outputs. There are 6 nibbles available. The data is right justified. For example BCD2.2 uses the 4 least significant nibbles. Most significant nibbles that you do not use are treated as *don't cares*. The HEX option does not allow specification of a radix point.

Example

To understand how the bits are referred: if **Q1 = 397.4** and if **OUT OUT24 Q1 BCD 3.1** are executed, the following data is sent to the 24 programmable outputs.

XXXX XXXX 1001 0111 0100

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The left-most **X** above indicates that pin #1 (bit 24, the most significant bit) is unchanged. The right-most **0** of **0100** indicates that pin #47 (bit 1, the least significant bit) is set to 5V (current not flowing) assuming negative logic is programmed using the **DEFINE OUT24** statement.

See Also: **DEFINE OUT24, IN IN24, OUT BIT**

Statement	Description
MATH Q2 = 324	'Defines Q2 as 324.
OUT OUT24 Q1 HEX 3	'Sets 3 least significant digits (12 'LSBs) of the 24 outputs.

OUT BIT

Name	**OUT BIT**
Descriptor	Set 1 of 24 Programmable Outputs
Type	Status
Default	OUT BITn = 1
Syntax	OUT BIT1 = 1
Options	TAB
	F1 F2 F3 F4 F5 F6

Description

The **OUT BIT** statement sets or resets one of the 24 programmable outputs. The **DEFINE OUT24** statement determines the logic level of the outputs.

The value of **n** may be 1-24. The value of the output may be 0 or 1. Bit #1 corresponds to pin #47 of the 50 pin output connector; this is the farthest right character when the **DISPLAY OUT24** statement is executing and is the least significant bit. Bit #24 corresponds to pin #1 of the 50 pin output connector; this is the farthest left character when the **DISPLAY OUT24** statement is executing, and is the most significant bit.

If positive logic is chosen, an **OUT BIT1=1** causes 5V (current not flowing) to be present on pin #47. If negative logic is chosen, an **OUT BIT1=1** causes 0V (current flowing) to be present.

See Also: **DEFINE OUT24, IN BIT, OUT OUT24**

OUT POB

Name	**OUT POB**
Descriptor	Set 1 of 4 Axis Outputs
Type	Status
Default	OUT POB XXXX
Syntax	OUT POB 11X0
Options	TAB X Q
	F1 F2 F3 F4 F5 F6

Description

The **OUT POB** statement sets or resets one of the 4 axis programmable outputs. An X means, don't change the output. A 1 sets the output (5V, no current flowing), and a 0 resets the output (0V, current flowing). The first X corresponds to the output bit on axis 1 connector.

If a variable is entered, the contents of the variable is used to create a binary number of 4 bits of magnitude (less than 16, as in the example below). The fractional portion and the sign of the variable are ignored (digits to the right of the decimal). Also, the number must not exceed 15 or an execution error will be generated.

Positive logic is always used in the **OUT POB** statement. The **DEFINE** statement does not affect the logic type used in the **OUT POB** statement.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PASSWD

Name	PASSWD
Descriptor	Define Password
Type	Set-Up
Default	PASSWD = nnnn
Syntax	PASSWD = 1234
Options	TAB
	F1 F2 F3 F4 F5 F6

Description

The **PASSWD** statement allows you to change the password to gain access to all of the Model 4000's functions. The default password is **4ØØØ**. The **PASSWD** can be any integer from 0 to 9999. Changing the password within a program changes the menu default back to the **ACCESS** menu. This means a user will need to enter the password after program execution, even if it had been previously entered.

PATT

Name	PATT
Descriptor	Define I/O Pattern
Type	Programming
Default	PATTn XXXX XXXX XXXX XXXX XXXX XXXX
Syntax	PATT20 1XX0 111X 1100 0000 1010 0010
Options	TAB X
	F1 F2 F3 F4 F5 F6

Description

The pattern statement allows you to define a 24-bit pattern for use with the **OUT**, **ON IN24**, **WAIT**, **IN** and **IF** statements. An **X** is a *don't care sign*. When outputting a pattern a *don't care means don't change the previous value. When inputting or comparing with a pattern, a don't care means don't use.* The value of **n** can be 1 through 20.

The first X of the **PATT** statement corresponds to bit #24 (pin #1, the most significant bit). The last X corresponds to bit #1, (pin #47, the least significant bit).

See Also: **DEFINE IN24**, **DEFINE OUT24**, **OUT**, **IN ON**, **IF**, **WAIT**

Example

Statement	Description
PATT1 1010 XXXX XXXX XXXX XXXX XXXX	'Sets pattern as bits 22 and 24 high 'and bits 21 and 23 low
WAIT FOR IN24 = PATT1	'Waits for trigger inputs 1 - 24 to 'match pattern 1 before going on to 'the next command

PDEF

Name	PDEF
Descriptor	Position Definition
Type	Programming
Default	PDEF * * * *
Syntax	PDEF 25000 50000 0 *
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

Upon executing a **PDEF** statement, the values are stored as the present absolute position for each axis.

The Model 4000 keeps track of four types of positions:

- ❑ Absolute motor
- ❑ Incremental motor
- ❑ Absolute encoder
- ❑ Incremental encoder

The **PDEF** statement effects only absolute positions, and only the absolute position corresponding to the current **MODE** statement. If you are in **MODE M_INC** or **M_ABS**, then the **PDEF** statement changes only the absolute

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

motor position. If you are in **MODE E_INC** or **E_ABS**, then only the absolute encoder positions are changed.

On power up the default is **PDEF Ø Ø Ø Ø** and **MODE M_INC**. The **PDEF** values and the **MODE** values are not changed when a program is started.

PDEF assigns an absolute position value to the present position. You may give each axis a different value with a unique number or variable. Any variable or number will be multiplied by the position scale factor specified with the **UNITS POS** statement. Any axis may have the NULL (*) entry, which indicates that it is left alone (not redefined). Axes that do not have a **NULL (*)** entry must be stopped (not moving) when this statement is performed. If you use incremental positioning, the **PDEF** statement has no immediate effect. However, if you use the **MODE** statement, the position reference is determined by the **PDEF** statement that the system most recently encountered.

The four sets of parameters define the value for the absolute position for each of the four axes. To enter a numeric value for position, enter the number on the keypad and press the **ENTER** key. To choose a variable, press the function key labeled Q, the variable number (entered on the keypad), and then press **ENTER**.

Example

Statement	Description
MATH Q9 = 25000	'Set current position to 25000
PDEF Q9 * * *	

POSM

Name	**POSM**
Descriptor	Set Position Maintenance Factors
Type	Set-Up
Default	POSM
Syntax	POSM MAXVEL 25000 25000 25000 25000
Options	TAB GAIN MAXVEL
	F1 F2 F3 F4 F5 F6

Description

The **POSM** statement defines the position maintenance move parameters. To enter Position Maintenance mode. The following statements must be executed in the order specified:

- ❑ **ENCO ERES** or **ENCO MRES**
- ❑ **MODE E_INC** or **E_ABS**
- ❑ **POSM GAIN** (Value)
- ❑ **POSM MAXVEL** (Value)
- ❑ **ENABLE POSM YES** (Optional)
- ❑ **ENABLE NDBWAIT** (Optional)
- ❑ **ENABLE STALL** (Optional)
- ❑ **ENCO DBND** (Optional)

POSM GAIN

Name	**POSM GAIN**
Descriptor	Set Velocity Gain for Position Maintenance
Type	Set-Up
Default	POSM GAIN * * * *
Syntax	POSM GAIN 250 250 250 250
Options	TAB NULL
	F1 F2 F3 F4 F5 F6

Description

The **POSM GAIN** statement sets a velocity gain factor (motor steps per second for each encoder step error) that is used to determine the position maintenance velocity. The factory default is 250 motor steps per second for each encoder step error.

The **POSM GAIN** value is scaled by the **UNIT VEL** parameter. Q variables can not be used as arguments for the **POSM GAIN** statement.

See Also: **ENCO, ENABLE POSM**

Example

Statement	Description
POSM GAIN 20 * * *	'Set position maintenance gain 20 steps per sec. 'leave the other axis at the previous setting.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

POSM MAXVEL

Name	POSM MAXVEL
Descriptor	Set Maximum Velocity for Position Maintenance
Type	Set-Up
Default	POSM MAXVEL * * * *
Syntax	POSM MAXVEL 25000 25000 25000 25000
Options	TAB NULL
	F1 F2 F3 F4 F5 F6

Description

The **POSM MAXVEL** statement sets the maximum velocity a position maintenance correction move achieves. The default position maintenance maximum velocity is 25,000 steps/second if the motor resolution is 25,000 steps/rev.

The **POSM MAXVEL** value is scaled by the **UNIT VEL** parameter. Q variables can not be used as arguments for the **POSM MAXVEL** statement.

See Also: **ENCO, ENABLE POSM**

Example

Statement	Description

`POSM MAXVEL 200000 * * *`	`'Sets the maximum correction`
	`'velocity to 200,000 steps per`
	`'second. Leaves the other axes at`
	`'their previous settings`

RESTORE

Name	RESTORE
Descriptor	Reset Data Statement Pointer
Type	Programming
Default	RESTORE
Syntax	RESTORE
Options	N/A
	F1 F2 F3 F4 F5 F6

Description

The **RESTORE** statement works with the **DATA** statement. The **DATA** statement is a list of numeric data. After using one of the **DATA** entries in the list (with the **IN DATA** statement), the Model 4000 uses the next entry. By executing a **RESTORE** statement, the Model 4000 knows to use the first entry in the list.

See Also: **IN DATA**

Example

Statement	Description
`RESTORE`	`'Resets the data pointer to the` `'first data point`

RETURN

Name	RETURN
Descriptor	Return from Subroutine
Type	Programming
Default	RETURN
Syntax	RETURN
Options	N/A
	F1 F2 F3 F4 F5 F6

Description

This statement is used in conjunction with the **GOSUB** statement. It causes the Model 4000 to return to the program statement following the **GOSUB** statement that caused the program to jump to the subroutine. For more information, refer to the **GOSUB** statement description.

If **ON CONDITION** interrupts use the **GOSUB** option, when the **RETURN** is executed, program flow continues at the statement following the statement where the interrupt occurred.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The **BREAK** statement clears all active **LOOP**'s and **GOSUB**'s. If a **RETURN** is encountered after a **BREAK**, an execution error will occur.

See Also: **GOSUB, ON, IF, BREAK**

RS232

Name	**RS232**
Descriptor	Set RS-232C Parameters
Type	Set-Up
Default	RS232 PORTn * ** ** *
Syntax	RS232 PORTNUM BAUD DATABITS PARITY STOPBIT ECHO ERRDISP
	RS232 PORT1 = 9600 8 OFF 1 * *
Options	TAB ON OFF NULL EVEN ETC
	TAB ODD ALL ERR ETC
	F1 F2 F3 F4 F5 F6

Description

The following parameters are associated with the RS-232C interface.

PORTNUM The **RS-232C** statement changes the operating parameters of ports 1 and 2.

BAUD Baud rates available are **NULL(*)**, 19,200; 9,600; 4,800; 2,400; 2,000; 1,200; 600; 300; and 150. *The default baud rate is 9,600.*

DATA BITS Data bits per character can be **NULL(*)**, 5, 6, 7, or 8. The default for this parameter is 8.

PARITY Parity can be **NULL(*)**, **OFF**, **EVEN**, or **ODD**. The default for this parameter is OFF.

STOP BIT **STOP** bit length can be **NULL (*)**, **1**, or **2**. The default for this parameter is 1. Note: the stop bit length when 5 data bits are selected is 1.5 or 2. The value is entered as 1, but the actual stop bit length is 1.5. If entered as 2 stop bits, the length is 2.

ECHO The character echo feature can be **NULL(*)**, **ON** or **OFF**. The default for this parameter is ON. *The PC-based Model 4000 application developer software will not work if echo is* **OFF**.

ERR DISP Error message display can be: **NULL(*)**, **OFF**, **ERR**, or **ALL**. The default for this option is **ALL**. The **OFF** option turns off the display of prompts, help messages and error messages to the port specified. The **ERR** option turns off the display of prompts and help messages and turns on the display of error messages. The **ALL** option turns on the display of prompts, the display of help messages and the display of error messages. The following prompts are used:

- **>**
- **LOAD>**
- **INSERT>**
- **IMMED>**

Error messages include **INVALID DATA IN LINE**.

Example

Statement

```
RS232 PORT1 = 9600 8 OFF 1 ON ALL
```

Description

```
'Sets the baud rate to 9600, 8
'data bits, parity is off, there
'is 1 stop bit, echo is on, and
'the error message function is on.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SEG

Name	SEG
Descriptor	Define Complex Move Segment
Type	Motion
Default	SEG *
Syntax	SEG BEG 25000 Q2 75000 *
Options	TAB BEG MID END REG
	TAB NULL Q
	F1 F2 F3 F4 F5 F6

Description

The **SEG** statement defines a distance segment of a complex motion profile. A complex motion profile is one that changes goal velocities after specified distances are completed.

A **VEL** statement executed before the **SEG** statement determines the goal velocity. Whether or not the goal velocity is achieved depends on the magnitude of the velocity change, the distance in which the velocity change is to take place, and the **ACCEL** or **DECEL** value which determine the rate of velocity change.

When using the **BEG**, **MID**, or **END** options, the next segment is started as soon as the distance of the current segment is traveled. When using the **REG** option, the segment is started as soon as the specified **REG** input is active.

SEG statements do not start motion. The **BEG**, **MID**, and **END** options only define the type of motion to be initiated when a **MOVE SEGCW** or **SEGCCW** statement is executed. The **SEG BEG**, **MID**, **END**, and **VEL** or **ACCEL** statements are all stored as they are executed. The motion starts when the **MOVE SEGCW** or **MOVE SEGCCW** statements are executed.

All segments are defined by the distance specified on the **SEG** statement and the current velocity, acceleration, deceleration, and mode values. The distances of all **SEG** statements must be positive. After scaling by the **UNIT POS** statement the value must be less than 99,999,999 or an execution error is generated. All distances are incremental, even if the current mode is absolute. The mode in effect will determine whether the distances are encoder steps or motor steps.

The first segment of a complex move profile is defined with the **SEG BEG** statement. Zero to three **SEG MID** statements may follow. Each **SEG** statement should be preceded by the motion parameter statements (**ACCEL**, **VEL**, **DECEL**, and **MODE**) required to completely define the segment. A segmented move definition must be completed with a **SEG END** statement.

A **SEG REG** statement defines a distance to travel after a registration input becomes active. Motion is started when the **MOVE SEGCW** statement is executed (assuming the **SEG BEG** and **SEG END** statements have been executed).

See Also: **ACCEL**, **VEL**, **UNIT**, **MODE**, **MOVE**

SEG BEG

Name	SEG BEG
Descriptor	Define First Complex Move Segment
Type	Motion
Default	SEG BEG * * * *
Syntax	SEG BEG 25000 75000 Q10 50000
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **SEG BEG** statement defines the first complex move segment. The parameters are distances the four axes should travel during the first segment.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If a parameter is a variable (Q1 - Q99), the distance taken from the variable is the value of that variable at the time the **SEG BEG** statement is executed, not the value of the variable at the time the **MOVE SEGCW** or **MOVE SEGCW** statement is executed.

The **SEG BEG** statement may be followed by zero to three **SEG MID** statements, but must be followed by a **SEG END** statement.

If a parameter is entered as a variable it's value is evaluated when the **SEG BEG** statement is executed, not at the time the **MOVE SEG CW** or **MOVE SEG CCW** statement is executed. **SEG** distances are computed internally in the definition stage, not in the motion stage.

See Also: **SEG MID, SEG END, MOVE SEGCW/SEGCCW**

SEG MID

Name	**SEG MID**
Descriptor	Define 2nd, 3rd, or 4th Complex Move Segment
Type	Motion
Default	SEG MID * * * *
Syntax	SEG MID 25000 Q10 Q11 Q12
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **SEG MID** statement defines second, third or fourth segments of a complex move. The parameters are the distances the four axes should travel during this segment.

If a parameter is a variable (Q1-Q99), then the distance taken from the variable is the value of that variable at the time the **SEG MID** statement is executed, not the value of the variable at the time the **MOVE SEGCW** or **MOVE SEGCW** statement is executed.

There can be as many as three **SEG MID** statements. After you have entered the number of **SEG MID** statements you desire, you must enter a **SEG END** statement.

See Also: **SEG BEG, SEG END, MOVE SEGCW, SEGCCW**

SEG END

Name	**SEG END**
Descriptor	Define Last Complex Move Segment
Type	Motion
Default	SEG END * * * *
Syntax	SEG END 25000 25000 25000 Q99
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **SEG END** statement defines the last segment of a complex move. The parameters are the distances the four axes should travel during the last segment.

A **SEG END** statement completes the complex move definition. The motion the complex moves define is initiated (in the specified direction) by the **MOVE SEGCW** or **MOVE SEGCCW** statements.

See Also: **SEG BEG, SEG MID, MOVE SEGCW, SEGCCW**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SEG REG

Name	**SEG REG**
Descriptor	Define Registration Distance
Type	Motion
Default	SEG REGn * * * *
Syntax	SEG REG1 Q99 Q98 25000 50000
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **SEG REG** statement defines the distance to travel when a registration input becomes active. The parameter n specifies the registration input that will cause motion on the axes which distances have been entered. The value of n can be 1 -4.

If a distance parameter is a variable (**Q1 - Q99**), the distance taken from the variable is the value of that variable at the time the **SEG REG** statement is executed, not the value of the variable when the registration input becomes active.

Up to four different registration moves can occur on a single axis while a move is in progress. Subsequent registration moves can interrupt a registration move already in progress. A registration input will generate another registration move if all the following conditions are met.

① A **SEG REG** statement has been executed for that registration input.
② The registration input is enabled.
③ 100 ms have elapsed from the time the registration input was last asserted (this is to debounce the input)
④ The registration input has become inactive, and then active.

See Also: **ENABLE REG**

SLIMIT

Name	**SLIMIT**
Descriptor	Set Software Position Limit Parameters
Type	Programming
Default	SLIMIT
Syntax	SLIMIT M CW 90000 90000 90000 90000
Options	TAB M_CW M_CCW E_CW E_CCW TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **SLIMIT** statement defines a software position limit. The position limit entered must be a value reflecting the absolute distance from zero as defined by the last **PDEF** statement or power cycle. This feature is most useful when operating in an absolute mode (**E_ABS** or **M_ABS**). When operating in an incremental mode (**E_INC** or **M_INC**), repeated moves may command the motor beyond the position allowed with **SLIMIT**.

If the position limit is exceeded, then moves are terminated using the **DECEL SLIMIT** values if entered, or the **DECEL** values if not.

There are two types of **SLIMIT** statements, motor step checking and encoder step checking. Each allows CW and CCW limits to be set. The values entered may be positive or negative for CW or CCW. Positive values are those positions in the CW direction from zero. Negative values are those positions in the CCW direction from zero.

The checking for **SLIMIT** may be turned off and on by using the **ENABLE M_SLIM** and **E_SLIM** statements.

Once enabled via the **SLIMIT** statement or the **ENABLE** statement, the **SLIMITs** will always be checked for (during program execution and in **IMMED** mode), unless they are disabled by the **ENABLE** statement. This is

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

true even if the program is stopped. They are cleared by a **RESET**, **POWER CYCLING**, or an **ENABLE** statement.

If a software limit has been hit, then the limit may be de-activated by moving in the opposite direction or by disabling **SLIMIT**s via the **ENABLE** statement.

The amount of time between actually reaching a **SLIMIT** position and when the Model 4000 recognizes this is up to 12 ms. This will result in a position error that is dependent upon the velocity at which the motor was turning and the deceleration rate (**DECEL SLIMIT**). *The default value is Ø.*

See Also: **ENABLE M_SLIM, E_SLIM, DECEL SLIMIT**

Example

Statement	Description
`SLIMIT M_CCW 20000 20000 20000 20000`	`'Sets all axes CW software 'limit to position 20000`
`SLIMIT M_CW 20000 20000 20000 20000`	`'Sets all axes CCW software 'limit to position 20000`

SHTDN

Name	**SHTDN**
Descriptor	Shutdown Motor/Drive
Type	Programming
Default	SHTDN DRIVE * * * *
Syntax	SHTDN DRIVE YES YES NO NO
Options	TAB YES NO NULL
	F1 F2 F3 F4 F5 F6

Description

The **SHTDN** statement turns on the **SHTDN** output in the motor/drive connector, which removes torque from the motor. You can use this statement to remove holding torque. This feature allows you to manually position a load.

The **SHTDN** parameter must be either **NULL**, **YES**, or **NO**. If you select **NULL**, the torque remains the same. If you select **YES**, regular torque is removed from a motor. If you select **NO**, regular torque is reapplied to a motor.

For the **SHTDN** statement to have any effect, standard cables should be used with Compumotor stepper drives. Alternately, the shutdown pins on the motor/driver 15 pin connector could be connected to any other input on a drive that would disable the amplifier.

When torque is removed from Compumotor's A Series Drive, the fault input will become active and the fault LED on the operator panel will be on.

Example

Statement	Description
`MOVE 10000 20000 * *`	`'Moves axes 1 & 2.`
`SHTDN DRIVE YES YES * NO`	`'Shuts down axis 1, 2 and leaves 'axis 3 as it was, turns axis 4 on.`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

UNIT

Name	**UNIT**
Descriptor	Set Scale Factor
Type	Set-Up
Default	UNIT
Syntax	UNIT VEL 25000 25000 25000 25000
Options	TAB POS VEL ACCEL PATH
	F1 F2 F3 F4 F5 F6

Description

The **UNIT** statement is used to scale the position, velocity, acceleration, and deceleration parameters to allow you to program those statements in the units that you find convenient. *Real numbers or Q values are allowed for the unit statement.*

UNIT POS

Name	**UNIT POS**
Descriptor	Set Position Scale Factor
Type	Set-Up
Default	UNIT POS * * * *
Syntax	UNIT POS 25000 25000 25000 25000
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **UNIT POS** statement defines the distance scale factor used in the **MOVE** and **MOVI** statement. Variables (**Q1 - Q99** or pointer variables **QQ1 - QQ99**) could also be used as the scale factor. The default value is 1. After scaling the **MOVE** parameter with the **UNIT POS** parameter, the result must not be larger than 99,999,999 or an execution error will result. The default value is 1. Fractional values are allowed, but may result in truncation error. Other statements scaled by **UNIT POS** include: **ENCO BKL**, **DISPLAY POS**, **IN POS**, and **SEG**.

Example

Statement	Description
`UNIT POS 2.513 10 * *`	`'Sets distance scale factor to 2.513 for '1 and 10 for 2.`
`MOVE 2513 25 0 0`	`'Moves axis 1-6315 steps and axis 2-250 'steps.`

UNIT VEL

Name	**UNIT VEL**
Descriptor	Set Velocity Scale Factor
Type	Set-Up
Default	UNIT VEL * * * *
Syntax	UNIT VEL 25000 25000 Q1 *
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **UNIT VEL** statement defines the velocity scale factor used to scale all **VEL** statements. The default value is 1 step/sec. Fractional values are allowed. Position maintenance velocities are also scaled by **UNIT VEL**.

Example

Statement	Description
`UNIT VEL 25000 25000 * *`	`'Sets velocity scale factor of axis '1 to 25,000 and axis 2 to 25,000.`
`VEL 5 3 * *`	`'Sets maximum velocity of axis 1 to '125,000 steps per sec and axis 2 to '75,000 steps per sec.`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

UNIT ACCEL

Name	UNIT ACCEL
Descriptor	Set Acceleration Scale Factor
Type	Set-Up
Default	UNIT ACCEL * * * *
Syntax	UNIT ACCEL 25000 25000 25000 25000
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **UNIT ACCEL** statement defines the acceleration scale factor used in all the **ACCEL** and **DECEL** statements. The default value is 1 step/sec^2. Fractional values are allowed.

Example

Statement	Description
`UNIT ACCEL 3000 1000 * *`	`'Sets acceleration scale factor to '3,000 and axis 2 to 1,000.`
`ACCEL 30 20 * *`	`'Sets acceleration of axis 1 to '90,000 steps per sec2 and axis 2 '20,000 steps per sec2.`

UNIT PATH VEL

Name	UNIT PATH VEL
Descriptor	Set Scale Factor for Path Velocity
Type	Set-Up
Default	UNIT PATH VEL *
Syntax	UNIT PATH VEL
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **UNIT PATH VEL** statement scales the **VEL PATH** statement parameters used in **LINT MODE** and in the Contouring option. Fractional values are allowed, but truncation errors may occur. Variables (**Q1 - Q99** or pointer variables **QQ1 - QQ99**) could also be used as the scale factor. The default value is 1. After scaling the **VEL PATH** parameter with the **UNIT PATH VEL** parameter, the result must not be larger than 99,999,999 or an execution error will result.

See Also: **LINT, VEL PATH**

UNIT PATH ACCEL

Name	UNIT PATH ACCEL
Descriptor	Set Scale Factor for Path Acceleration
Type	Set-Up
Default	UNIT PATH ACCEL *
Syntax	UNIT PATH ACCEL
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **UNIT PATH ACCEL** statement scales the **ACCEL PATH** statement parameters used in **LINT MODE** and in the Contouring option. Fractional values are allowed, but truncation errors may occur. Variables (**Q1 - Q99** or pointer variables **QQ1 - QQ99**) could also be used as the scale factor. The default value is 1. After scaling the **ACCEL PATH** parameter with the **UNIT PATH ACCEL** parameter, the result must not be larger than 99,999,999 or an execution error will result.

See Also: **LINT, ACCEL PATH, DECEL PATH**

Example

This following example shows how to use variables to change the overall dimensions of a contoured figure.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Statement	Description
`MATH Q80 = 1`	'Set a scale factor to scale the position scale 'factor.
`MATH Q81 = 125000`	'Set the position scale factor.
`MATH Q82 = Q81 * Q80`	'Set the variable used in the UNIT POS statement.
`UNIT PATH POS Q82`	
`LABEL TRYAGAIN`	'Display F-KEY functions on LCD line 4.
`OUT LCD4, 0 ^ UP DOWN START^`	
`ON F-KEY1 GOTO SCALE_UP`	'Increase Q80 scale factor so that all PATH 'position parameters are scaled up.
`ON F-KEY2 GOTO SCALE_DN`	'Decrease Q80 scale factor so that all PATH 'position parameters are scaled down.
`ON F-KEY3 GOTO START`	
`GOTO TRYAGAIN`	
`LABEL START`	'This section does the motion.
`ON F-KEY1 DISABLE`	'Disable the F-KEY ON statements so that the
`ON F-KEY2 DISABLE`	'motion is not interrupted.
`ON F-KEY3 DISABLE`	
`PATH COMPILE CIRCLE`	'Re-compile to account for any changes made to 'scale factors.
`PATH DEF CIRCLE 1 2`	'Start a path definition.
`PATH OCCW 0 0 0 9`	'Define a circle of radius 9.
`PATH END`	
`GOTO TRYAGAIN`	'Go back and allow dimensions to be adjusted.
`LABEL SCALE_UP`	
`MATH Q80 = Q80 + .01`	'Increase dimensions by 1%.
`OUT LCD1,0 ^Overall scaling factor is: ^ Q80`	'Display scaling factor on LCD.Go back and allow 'dimensions to be adjusted.
`GOTO TRYAGAIN`	
`LABEL SCALE_DN`	
`MATH Q80 = Q80 - .01`	'Decrease dimensions by 1%.
`OUT LCD1,0 ^Overall scaling factor is:  ^ Q80`	'Display scaling factor on LCD.
`GOTO TRYAGAIN`	'Go back and allow dimensions to be adjusted.

VEL

Name	**VEL**					
Description	Set Peak Velocity					
Type	Motion					
Default	VEL * * * * *					
Syntax	VEL 100000 100000 100000 100000					
Options	TAB	Q	NULL	SS	PATH	ETC
	TAB	HOME	RANGE	JOGL	JOGH	ETC
	TAB	JOYL	JOYH			ETC
	F1	**F2**	**F3**	**F4**	**F5**	**F6**

Description

The **VEL** statement specifies the maximum velocity a move may achieve. The minimum starting and stopping velocities that are used, the range of frequencies the motor drive will receive, the maximum velocity a jog move may achieve, and the maximum velocity a Go Home move may achieve.

The minimum resolution of any **VEL** statement is determined by the **VEL RANGE** and the **ENCO MRES** statements, regardless of the resolution entered on a **VEL** statement. *The **UNIT VEL** statement parameters scale the **VEL** statement parameters.*

The default peak velocity for each axis is 5,000 steps/sec unless you change that value with this statement. This default value corresponds to 0.2 rev/sec for a standard 25,000-step/rev Compumotor.

When one of the **VEL** options are not chosen, (**PATH**, **HOME**, **JOYL**, **JOYH**, **RANGE**, **JOGL**, or **JOGH**), the **VEL** statement values entered are used for the following **MOVE** statement options.

- ❑ **SLEWCW**
- ❑ **SLEWCCW**
- ❑ **CONSTANT**
- ❑ **Q VARIABLE**

See Also: **UNIT VEL**, **VEL RANGE**, **ENCO MRES**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example

The example below sets the maximum velocity for axis #1 to 100,000 and 50,000 for axis #2. This number is multiplied by the velocity scale factor set with the **UNIT VEL** statement. Axis #3 will remain unchanged and Axis #4 will use the value of variable Q7 as its velocity.

Statement	Description
`VEL 100000 50000 * Q7`	

VEL SS

Name	**VEL SS**
Descriptor	Set Start and Stop Velocity
Type	Motion
Default	VEL SS * * * *
Syntax	VEL SS 5000 5000 5000 5000
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **VEL SS** (start/stop) statement sets the velocity in which all future moves will instantaneously start from before beginning acceleration, and stop at when decelerating.

The **VEL SS** velocity only applies to preset moves and **SLEW** (continuous) moves. **HOME**, **LINT**, and **JOG** moves do not use the start/stop velocity. The **VEL SS** statement is scaled by the **UNIT VEL** parameter. If too large a value is entered for **VEL SS**, it may cause the axis to stall.

Example

In this example, both motors will immediately jump to 200 steps/sec, then ramp to 50,000 steps/sec.

Statement	Description
`VEL SS 200 200 * *`	`'Sets the (start/ stop) velocity to '200 steps/sec for axes 1 & 2.`
`ACCEL 100000 100000 * *`	`'Sets the acceleration rate to '100,000 steps/sec`^{2} `for axes 1 & 2.`
`VEL 50000 50000 * *`	`'Sets the maximum velocity rate to '50,000 steps per sec.`
`MOVE 100000 100000 * *`	`'Moves axes 1 & 2.`

VEL PATH

Name	**VEL PATH**
Descriptor	Set Path Velocity
Type	Motion
Default	VEL PATH *
Syntax	VEL PATH 25000
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **VEL PATH** statement specifies the velocity used for a linear interpolated move. The path velocity is the vector sum of the velocity of the axis participating in the interpolation. The **VEL PATH** parameter also determines the velocity for paths executed with the Contouring option.

Example

In the program below the path velocity is 250,000. Using the distances and **VEL PATH** given, the Model 4000 calculates the peak velocity for each axis. The peak velocity is reached, if the move is long enough for each axis. The Model 4000 makes the calculations shown below when the **MOVE** statement is executed. The **VEL PATH** statement is scaled by the **UNIT PATH VEL** parameter.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

$$\text{Axis 1 VEL} = \left[\frac{25000}{\sqrt{((250002)+(200000^2)+(100000^2))}}\right] \text{X } (250000) = 27777.78$$

$$\text{Axis 3 VEL} = \left[\frac{200000}{\sqrt{((250002)+(200000^2)+(100000^2))}}\right] \text{X } (250000) = 222222.22$$

$$\text{Axis 4 VEL} = \left[\frac{100000}{\sqrt{((250002)+(200000^2)+(100000^2))}}\right] \text{X } (250000) = 111111.11$$

Axes 1, 3, and 4 will start moving at the same time, reach their peak velocity at the same time, start decelerating at the same time, and stop at the same time. Axis 2 will not participate in the linear interpolated move, it does a preset move of 50,000.

See Also: **LINT, MOVE, MOVI, UNIT PATH VEL**

Statement	Description
LINT MODE YES NO YES YES	'Axis #2 is not participating in 'LINT move.
ACCEL PATH 250000	
DECEL PATH 250000	
VEL PATH 250000	
MOVE 25000 50000 200000 100000	'These are distances used in the 'calculation above.

VEL HOME

Name	**VEL HOME**
Descriptor	Set Move Home Peak Velocity
Type	Set-Up
Default	VEL HOME * * * *
Syntax	VEL HOME 200000 Q5 9.8
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **VEL HOME** statement sets the maximum velocity a **MOVE HOME** move may achieve. The **VEL HOME** statement is scaled by the **UNIT VEL** parameter.

Example

Statement	Description
VEL HOME 10000 20000 * *	'Set the go home velocity rate on Axes 1&2
MOVE HOMECW HOMECCW * *	'Moves axis 1 home in the CW direction and 'moves axis 2 home in the CCW direction.

VEL RANGE

Name	**VEL RANGE**
Descriptor	Set Velocity Range
Type	Set-Up
Default	VEL RANGE * * * *
Syntax	VEL RANGE LOW HIGH LOW *
Options	TAB HIGH LOW NULL
	F1 F2 F3 F4 F5 F6

Description

The **VEL RANGE** statement selects either a high or low frequency range for the step output signal. A complete table of the frequency ranges available is given in the **ENCO MRES** statement description. The default for this statement is **VEL RANGE HIGH**. If using low resolution drives, or slow speed applications, **VEL RANGE LOW** can give better low frequency velocity control.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

After the **VEL RANGE** statement is executed, motion parameters should be redefined. The following statements should be executed after **VEL RANGE**: **ENCO MRES**, **VEL**, **ACCEL**, and **DECEL**.

See Also: **ENCO MRES**

Example

Statement	Description
VEL RANGE LOW LOW * *	'Set axes 1 and 2 to low peak 'velocity range.

VEL JOGL

Name	**VEL JOGL**
Descriptor	Set Jog Mode Low Velocity
Type	Motion
Default	VEL JOGL * * * *
Syntax	VEL JOGL 25000 * 1000 Q3
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **VEL JOGL** statement sets the **JOG** mode's low velocities for each axis. This statement should be executed before an **ENABLE JOG** statement so that the velocities specified are active in **JOG** mode. *The default JOG low velocity is 25,000 steps/sec.* The **VEL JOGL** statement is scaled by the **UNIT VEL** parameter.

See Also: **ENABLE JOG**, **ENABLE TCH_JOG**, **ACCEL JOG**, **DECEL JOG**

VEL JOGH

Name	**VEL JOGH**
Descriptor	Set Jog Mode High Velocity
Type	Motion
Default	VEL JOGH * * * *
Syntax	VEL JOGH 25000 Q2 * 75000
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **VEL JOGH** statement sets the **JOG** mode's high velocities. This statement should be executed before an **ENABLE JOG** statement so that the velocities specified are active in **JOG** mode. *The default JOG high velocity is 75,000 steps/sec.* The **VEL JOGH** statement is scaled by the **UNIT VEL** parameter.

See Also: **ENABLE JOG**, **ENABLE TCH_JOG**, **ACCEL JOG**, **DECEL JOG**

VEL JOYL

Name	**VEL JOYL**
Descriptor	Set Joystick Mode Low Velocity
Type	Motion
Default	VEL JOYL * * * *
Syntax	VEL JOYL 25000 * 5000 Q3
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **VEL JOYL** statement sets the **JOY** mode's low velocities for each axis. This statement should be executed before an **ENABLE JOY** statement so that the velocities specified are active in **JOY** mode. *The default JOY low velocity is 5,000 steps/sec.* The **VEL JOYL** statement is scaled by the **UNIT VEL** parameter.

See Also: **DEFINE JOYSL**, **JOYSH**, **CENTER**, **CTR_DB**, **END_DB**, **ENABLE JOY**, **ENABLE TCH_JOY**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

VEL JOYH

Name	**VEL JOYH**
Descriptor	Set Joystick Mode High Velocity
Type	Motion
Default	VEL JOYH
Syntax	VEL JOYH 25000 Q2 * 12500
Options	TAB Q NULL
	F1 F2 F3 F4 F5 F6

Description

The **VEL JOYH** statement sets the **JOY** mode's high velocities. This statement should be executed before an **ENABLE JOY** statement so that the velocities specified are active in **JOY** mode. *The default **JOY** high velocity is 12,500 steps/sec.* The **VEL JOYH** statement is scaled by the **UNIT VEL** parameter.

See Also: **DEFINE JOYSL, JOYSH, CENTER, CTR_DB, END_DB, ENABLE JOY, ENABLE TCH_JOY**

WAIT

Name	**WAIT**
Descriptor	Wait for Condition
Type	Programming
Default	WAIT FOR
Syntax	WAIT FOR 12.5 SECONDS
Options	TAB SECOND MOVES IN24 BIT ETC
	TAB F-KEY JOYTRIG TRIG ETC
	F1 F2 F3 F4 F5 F6

Description

The **WAIT** statement suspends program execution until the specified condition is met. If an **ON** statement becomes true while the Model 4000 is suspended due to a **WAIT** statement, the **WAIT** statement is terminated immediately and the **ON** statement branch is taken and program execution continues.

WAIT FOR SECONDS

Name	**WAIT FOR SECONDS**
Descriptor	Wait for Time
Type	Programming
Default	WAIT FOR SECONDS
Syntax	WAIT FOR 12.346 SECONDS
Options	TAB Q
	F1 F2 F3 F4 F5 F6

Description

The **WAIT FOR SECONDS** statement suspends program execution until the specified number of seconds have elapsed. The maximum resolution of the time entered is milliseconds. The maximum value is 429,496.729.

See Also: **ON TIME**

Example

Statement	Description
MOVE 10000 10000 0 0	'Moves axes 1 & 2 10,000 steps.
WAIT FOR 0.1 SECONDS	'Waits 100 ms after the move is 'complete.
MOVE -10000 -10000 0 0	'Moves axes 1 & 2 -10,000 steps.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

WAIT FOR MOVES

Name	**WAIT FOR MOVES**
Descriptor	Wait for Move Completion
Type	Programming
Default	WAIT FOR MOVES NO NO NO NO
Syntax	WAIT FOR MOVES YES NO NO YES
Options	TAB YES NO
	F1 F2 F3 F4 F5 F6

Description

The **WAIT FOR MOVES** statement instructs the Model 4000 to wait for specific axes to have completed their preset moves or to have completed acceleration or deceleration of continuous moves started with the **MOVI** statement. Each axis may be specified to wait or not wait individually. The next program statement will not be executed until all axes specified to wait have finished. This statement is intended to be used with the **MOVI** statement to allow flexibility in sequences of moves on multiple axes.

Note The * character cannot be used to substitute for **YES** and **NO** specifications.

See Also: **MOVE, MOVI**

Example

In this example, axes 1 and 2 both begin preset moves of different lengths. The Model 4000 waits for axis #1 to finish its first move and will start axis #1 in its second move before axis #2 finishes its first move. The Model 4000 then waits for both axes to finish before starting the final pair of moves.

Statement	Description
`MOVI 25000 50000 0 0`	
`WAIT FOR MOVES YES NO NO NO`	
`MOVI 25000 * 0 0`	
`WAIT FOR MOVES YES YES NO NO`	
`MOVE -36000 22525 0 0`	

WAIT FOR IN24

Name	**WAIT FOR IN24**
Descriptor	Wait for 24 Programmable Inputs
Type	Programming
Default	WAIT FOR IN24 = PATTn
Syntax	WAIT FOR IN24 = PATT20
Options	TAB
	F1 F2 F3 F4 F5 F6

Description

The **WAIT FOR IN24** statement suspends program execution until the 24 inputs match the **PATT** variable specified. Then the program will continue to execute. The **DEFINE IN24** statement affects the polarity of the inputs.

If the = is changed to an ! symbol the program will continue when the 24 inputs do not match the **PATT** variable specified.

See Also: **DEFINE IN24, PATT, ON IN24, IF IN24, IN IN24**

Example

Statement	Description
`PATT1 1010 XXXX XXXX XXXX XXXX XXXX`	`'Sets pattern as bits 22 and 24 'high and bits 21 and 23 low 'assuming positive logic.`
`WAIT FOR IN24 = PATT1`	`'Waits for trigger inputs 1 - 24 'to match pattern 1 before going 'to next command.`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

WAIT FOR BIT

Name	WAIT FOR BIT
Descriptor	Wait for 1 of 24 Programmable Inputs
Type	Programming
Default	WAIT FOR BITn = 0
Syntax	WAIT FOR BIT21 = 1
Options	TAB
	F1 F2 F3 F4 F5 F6

Description

The **WAIT FOR BIT** statement suspends program execution until the input bit specified matches the level specified. Then the program will continue to execute. The **DEFINE IN24** statement affects the value of the inputs.

See Also: **DEFINE IN24, ON BIT, IF BIT, IN BIT**

Example

Statement	Description
`WAIT FOR BIT1 = 0`	`'Before making the move, wait until 'input 1 is low (ØVDC, current 'flowing).`
`MOVE 10000 0 0 0`	`'Moves axis 1.`

WAIT FOR F-KEY

Name	WAIT FOR F-KEY
Descriptor	Wait for an F-Key
Type	Programming
Default	WAIT FOR F-KEYn TO BE PRESSED
Syntax	WAIT FOR F-KEY5 TO BE PRESSED
Options	TAB
	F1 F2 F3 F4 F5 F6

Description

The **WAIT FOR F-KEY** statement suspends the program until the specified **F-KEY** is pressed. Once the **F-KEY** is pressed, the program will continue.

See Also: **IF F-KEY, ON F_KEY**

Example

Statement	Description
`OUT LCD2,01 ^PRESS F1 TO START^`	`'Display message on LCD line 2, 'first column.`
`WAIT FOR F-KEY1 TO BE PRESSED`	`'Waits until F1 is pressed.`
`MOVE 10000 10000 * *`	`'Moves axes 1 & 2.`

WAIT FOR JOYTRIG

Name	WAIT FOR JOYTRIG
Descriptor	Wait for the Joystick Input
Type	Programming
Default	WAIT FOR JOYTRIG
Syntax	WAIT FOR JOYTRIG
Options	TAB
	F1 F2 F3 F4 F5 F6

Description

The **WAIT FOR JOYTRIG** statement suspends program execution until the **JOYTRIG** input becomes active. Once the **JOYTRIG** input becomes active, the program continues.

See Also: **ON JOYTRIG, IF JOYTRIG**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example

Statement	Description

`WAIT FOR JOYTRIG`	`'Waits for JOYTRIG input to go`
	`'active before going to next`
	`'command.`

WAIT FOR TRIG

Name	**WAIT FOR TRIG**
Descriptor	Wait for 1 of 4 Trigger Inputs
Type	Programming
Default	WAIT FOR TRIGn = 0
Syntax	WAIT FOR TRIG4 = 1
Options	TAB
	F1 F2 F3 F4 F5 F6

Description

The **WAIT FOR TRIG** statement suspends program execution until the specified axis **TRIG** input becomes active. Once active the program continues.

The logic type is always positive logic. The **DEFINE** statement does not affect the logic type used by the **WAIT FOR TRIG** statement. Use of the **ENABLE TRIG REV** has no effect on this command.

See Also: **ON JOY, IF TRIG, IN TRIG**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

C H A P T E R ⑤

Application Design

Chapter Objectives

The information in this chapter will enable you to:

- Recognize and understand important considerations that must be addressed before you implement your application
- Customize the system to meet your requirements

Front Panel Program Generation

This section discusses the following aspects of program creation and operation:

- Automatic Power-Up Execution
- Creating a New Program
- Running Your Program

Automatic Power-Up Execution

The **POWER_UP** program name is searched for when power is applied to the Model 4000. If a program exists with this name, that program is started immediately.

Creating a New Program

To create a new program you must perform the following procedures:

- Move to the **MAIN MENU** (Press **ACCESS** and enter **4000**)
- Select the **EDITOR** (F1)
- Select a program name from the name list or use the **ALPHA** key to enter a new program name (eg.: P1 or PROG1) (the ↑↓ arrows scroll through the alphabet). (Press **ENTER**)
- Press the **INSERT** function key (F1)
- Press **ENTER** to begin a program at line 1

The list of Model 4000 statements will be above the soft function keys. Use the **UP** and **DOWN** arrow keys or **ETC** to display other available statements.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

New programs can also be created by downloading from a remote computer. The *Remote Operation* section of this chapter describes the downloading procedure in detail.

Example

Move a motor back and forth 25,000 steps.

☛This program disables over-travel limits. Exercise caution if your load is attached.

After following the steps described under *Creating a New Program*,

- ❑ Select the **ACCEL** statement (Acceleration)
- ❑ Enter 25000 (steps/sec^2)
- ❑ Press the **ENTER** key

```
1  ACCEL 25000 * * *
2
```

- ❑ Scroll to the **VEL** statement
- ❑ Select **VEL** (Velocity)
- ❑ Enter 25000 (steps/sec)
- ❑ Press the **ENTER** key

```
2  VEL 25000 * * *
3
```

- ❑ Scroll to the **ENABLE** Statement
- ❑ Select **ENABLE**
- ❑ Scroll through the options (**ETC**) to **HLIMIT** (Hardware Limits)
- ❑ Enter NO to disable hardware limits *
- ❑ Press the **ENTER** key

```
3  ENABLE HLIMIT NO * * *
4
```

- ❑ Scroll to the **MOVE** Statement
- ❑ Select **MOVE**
- ❑ Enter 25000 (steps)
- ❑ Press the **ENTER** key

```
4  MOVE 25000 * * *
5
```

- ❑ Select **MOVE** again
- ❑ Enter -25000 (steps)
- ❑ Press the **ENTER** key

```
5  MOVE -25000 * * *
6
```

- ❑ Scroll to the **DONE** statement
- ❑ Select **DONE**
- ❑ Press the **ENTER** key

```
6  DONE
7
```

- ❑ Press **MENU RECALL** twice

You have just entered your first 4000 program!

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Running Your Program

- Press **START** to run your program

The motor will move 25,000 steps CW and then 25,000 steps CCW. Press **MENU RECALL** to clear the **PROGRAMS FINISHED** message.

Pressing **RUN** and **ENTER** twice will do the same thing as **START**.

Single Stepping

You may want to execute your program one step at a time (during debugging). To execute your program step-by-step:

- Select **RUN**
- Select the program name to run, press **ENTER**
- Select **STEP** (via soft function key) before hitting **ENTER**
- Press **ENTER** to execute and display each line of your program.

Press **STOP** to stop program execution.

Editing an Existing Program

To **EDIT** an existing program, select the **EDITOR** (F1 from the **MAIN MENU**). This will bring up the intermediate menu that allows you to select which program to **EDIT** . Use the **FND_PROG** option over F6 to scroll through existing programs. Press **ENTER** to edit the program selected.

Use the arrow keys to scroll through program statements. Use the **INSERT** subfunction to insert statements. Use **EDIT** to modify existing statements. Refer to *Chapter 2, Getting Started* for a complete description (navigating through the Model 4000's functions).

Though Model 4000 programs are stored with line numbers, each insert and delete results in renumbering the entire program. Use the **LABEL** statement to declare subroutines or branching locations.

Auto-Run on Power-Up

Saved programs may be run automatically on power-up using the **POWER_UP** program name when entering the editor. Simply enter the statements you want executed at power-up into the **POWER_UP** program.

Program Basics

This section discusses the following aspects of basic program design:

- Password
- RS232
- Mode
- Units
- Limits
- Jog/Joystick
- Encoder

Set-Up

The Model 4000 has a number of set-up parameters that allow you to configure the Model 4000 to your application. The following is a list of modes and parameters that should help when programming your application. Many of the following set-up parameters have been included in the default **POWER_UP** program.

Password

The password allows you to restrict access to all of the Model 4000's functions. The default password is **4ØØØ**. The password can be any integer from 0 to 9999 (refer to the **PASSWD** statement in *Chapter 4, Software Reference*).

RS232C

The Model 4000 is sent from the factory with 9600 baud, 8 data bits, 1 stop bit and no parity. If your communication device is set differently then refer to the **RS232** statement in *Chapter 4, Software Reference*, for proper configuration. For more information, refer to the *Remote Operation* section of this chapter.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Mode

The Model 4000 can be programmed to position in either incremental or absolute mode. You should select the mode that will be most convenient for your application. For example, a repetitive cut-to-length application would be appropriate for incremental positioning. X-Y positioning, on the other hand, might be better served in the absolute mode. Both absolute and incremental modes can be specified in either motor or encoder steps. Refer to the **MODE** statement in *Chapter 4, Software Reference*, for more information.

```
MODE M_ABS M_INC * E_INC
```

- Axis #1 Motor absolute
- Axis #2 Motor incremental
- Axis #3 No change
- Axis #4 Encoder incremental

Limits

The Model 4000 can respond to both hardware and software end-of-travel limits. The Model 4000 is sent from the factory with the hardware limits enabled. **It is your responsibility to disable these limits either through software or hardware before motion can occur. End-of-travel limits help prevent your machine from crashing.** Once a hardware or software limit is reached, the Model 4000 will decelerate that axis at a rate specified by the user. Typically, software and hardware limits are positioned in such a way that when the software limit is reached the motor will start to decelerate towards the hardware limit. This will allow for a much smoother stop at the hardware limit. Software limits can be referenced using either motor or encoder positioning. Refer to **ENABLE HLIMIT**, **ENABLE E_SLIM**, **ENABLE M_SLIM**, **DECEL HLIMIT**, and **DECEL SLIMIT** in *Chapter 4, Software Reference*, for more information

The example below uses the **UNITS** and **ENCO** statement and defines software limits in revolutions. When you define software limits they become active. They may be temporarily disabled by the **ENABLE E_SLIM** command, but will be re-enabled if a limit definition or another **ENABLE** is encountered. The software limits are referenced from a position of absolute zero. Both ends of the travel may be positive (axis #2) or both may be negative. Care must be taken when performing incremental moves because the software limits are always defined in absolute terms. They must be large enough to accommodate the moves or a new zero point must be defined before each move.

In this example, the hardware limits have been enabled on axes #3 and #4 and disabled on axes #1 and #2. Deceleration rates have been specified for both software and hardware limits. If a limit is encountered, the motors will decelerate to a stop.

```
ENABLE HLIMIT NO NO YES YES
DECEL HLIMIT * * 10 10
SLIMIT M_CW 5 10 * *
SLIMIT M_CCW 0 2 * *
SLIMIT E_CW * * 0 10
SLIMIT E_CCW * * -10 -10
DECEL SLIMIT 10 10 10 10
ENABLE E_SLIM NO NO YES YES     'It is not necessary to include these
ENABLE M_SLIM YES YES NO NO     'enables in an actual program because
                                'the SLIMIT command automatically
                                'enables them
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Units

The **UNIT** statement allows the user to scale acceleration, deceleration, velocity, and position to values that are appropriate for the application. For example, a user has a 25,000 step/rev motor attached to a 5 pitch leadscrew that he wants his operator to position in inches. (25,000 steps/rev x 5 revs/inch = 125,000 steps/inch) A scale factor of 125,000 could then be assigned to the move distance. If the operator entered a **MOVE** value of 1.000 the Model 4000 would send out 125,000 pulses which would correspond to one inch. (Refer to **UNIT ACCEL**, **UNIT PATH**, **UNIT VEL**, and **UNIT POS** in *Chapter 4, Software Reference* for more information.) The **UNIT ACCEL** also scales the **DECEL** statement.

The step pulse width is a function of the motor resolution (ENCO MRES) and velocity range, and cannot be changed independently. This is important if you are using Digiplan or other full or half steppers. These drives normally require a wider pulse width than the default.

The example below defines the units that are used for all four axes. The units for position are 125000, 25000, 5000 and 200 for axes #1 through #4, respectively. The units for axis #1 enable the user to program a 25000 step/rev motor with a 5 pitch lead screw in units of inches. Axis #2 uses a 25000 step/rev motor and allows the user to program in revs. Axes 3 and 4 have motors with 5000 and 200 steps/rev, respectively and allow the user to program in revs. The velocity and acceleration units allow the user to program all axes in rps and rps^2. The **ENCO MRES** has been included to define parameters associated with the step pulse output. Refer to the description in the command listing for more information.

```
ENCO MRES 25000 25000 5000 200
UNIT POS 125000 25000 5000 200
UNIT VEL 25000  25000 5000 200
UNIT ACCEL 25000 25000 5000 200
```

Jog/Joystick

The **JOG** mode can be accessed through the front panel or by enabling it in a program (**ENABLE JOG**). The **JOG** mode allows the operator to manually position a load. Although all axes may be enabled for **JOG** mode at any time, only one axis can be moving at a time. A high and low jog velocity may be specified for each axis. These values are displayed on the front panel when entering the **JOG** mode. The **JOG** velocities may be defined with the **VEL JOGL** and **VEL JOGH** statements, or modified while in **JOG** mode. The acceleration and deceleration of the **JOG** move can be configured using **ACCEL JOG**, and **DECEL JOG** respectively. (For further information, refer to *Chapter 4, Software Reference.*)

Similarly **JOYSTICK** mode may be activated in a program using the **ENABLE JOY** statement. In **JOYSTICK** mode, the analog voltage inputs on Auxiliary Connector #2 are used to manually position the load. In **JOYSTICK** mode up to all 4 axes can be enabled and moving at the same time. **JOYSTICK** mode is described in greater detail later in this chapter.

The example below uses the units from above and defines the jog accelerations and decelerations in units of rps^2. The **ENABLE JOG** statement puts the Model 4000 into **JOG** mode and allows you to manually jog the motors to a specific position. The **MENU MODE** key disables **JOG** mode and continues running the program after the **ENABLE JOG** statement.

```
VEL JOGL 1 1 .5 .1
VEL JOGH 5 5 2.5 .5
ACCEL JOG 100 100 10 1
DECEL JOG 10 10 10 1
ENABLE JOG YES YES YES YES
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

JOG or JOYSTICK mode can also be used to teach absolute axis positions. The positions you wish to store can either be stored as actual values, or as values of numeric Q variables. See the ENABLE TCH_JOG and ENABLE TCH_JOY statement descriptions in *Chapter 4, Software Design* for more details.

Encoder

When using an encoder in an application, there are several set-up configurations to be considered. You must specify the motor and encoder resolutions used for the Model 4000. The ENCO MRES and ENCO ERES commands define the motor and encoder resolutions respectively (4000 counts/rev is a typical encoder resolution value). The power-up default values for motor resolution and encoder resolution are 25,000 and 4,000 respectively. To achieve accurate encoder step positioning, the motor resolution should be at least 2-3 times greater than the encoder resolution. This allows the motor to successfully find the commanded encoder step position. When calculating the distance or velocity of a move, the operating mode, either MODE E_INC/E_ABS or M_INC/M_ABS, must be factored in. The UNIT commands should be changed to match the MODE.

There are several facets to encoder mode operation, some or all of which may be appropriate for particular applications. General features to encoder mode operation in the Model 4000 include Stall Detection, Backlash Compensation, Position Maintenance, and Wait for Deadband.

Stall detection with backlash compensation can be enabled whether you are programming in motor or encoder step mode (MODE). However, position maintenance and wait for deadband can only be used in encoder step mode. The different aspects of encoder operation are described below.

Stall Detection

The ENABLE STALL statement allows the Model 4000 to detect stall conditions. If used with ENABLE MOVKIL the move will be aborted upon detecting a stall. If used with ON STALL, the user may branch to any other section of program when a stall is detected. By definition, if the Model 4000 commands the motor to move 4 full motor steps (not microsteps) and does not receive back at least half of the expected encoder pulses, it signals a stall condition. For example, if ENCO MRES is set to 25,000, and ENCO ERES is set to 4,000 the Model 4000 would need to send out 500 pulses (125 microsteps/full step *4 full steps) and get back less than 80 encoder pulses to indicate a stall has occurred. The axis fault light will turn red on the front panel when a stall occurs. Commanding a successful move will clear the stall conditions. Refer to the ENCO, ENABLE, ON, and POSM statements in *Chapter 4, Software Reference*, for more information.

Backlash Compensation

In some systems, there may be several elements of mechanical linkage between the motor shaft and the load, which would allow the motor shaft to turn before the load starts moving. This is referred to as *mechanical backlash*. The ENCO BKL command allows you to specify the number of motor steps the Model 4000 will send before it starts receiving encoder pulses. This allows you to compensate for backlash without causing the Model 4000 to react as if a stall had occurred. The ENCO BKL value is added on to the standard number of motor steps sent before a stall is detected to widen the stall detect window.

Position Maintenance

Another encoder set-up parameter is the ENCO DBND statement. The deadband statement sets the number of error counts allowed before a position maintenance correction move is initiated. This is useful for situations in which it is not desirable to have the load position continuously corrected, but correction is required outside the deadband. Positioning under a microscope is an example.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Basically, when position maintenance is enabled in the Model 4000, it uses the values previously defined for **ENCO MRES** and **ENCO ERES** to command the motor to move the distance it thinks is necessary to get back the correct number of encoder steps (the distance commanded in encoder mode). After the initial move is complete, if the axis is not within the defined deadband of encoder steps, a correction move is initiated until the axis is within the deadband. Without Wait for Deadband enabled, program execution will move on after the initial move is commanded.

The **POSM GAIN** and **POSM VEL** statements define the gain factor and the maximum velocity a position maintenance correction move achieves. The gain factor is the velocity increment used per encoder step of error. The **POSM MAXVEL** statement sets the maximum velocity a position maintenance move can achieve. If either of these is too large, position instability may result. If either is too small, position correction may be too slow. The values are determined by experiment.

Position maintenance is not recommended when using the Model 4000 with Digital Servo Drives, which have their own position maintenance capabilities.

Wait for Deadband

The **ENABLE NDBWAIT** statement causes program execution to wait until the axes specified are within the deadband range before executing the next line. You must define a deadband (**ENCO DBND**) and enable position maintenance (**ENABLE POSM**) for **ENABLE NDBWAIT** to work. This feature is useful when the axis just moved must be in the correction position to allow safe motion or safe program execution in subsequent statements.

Command Summary

The order of Encoder mode commands is very important to ensure proper operation. A recommended order for the commands follows:

ENCO ABS
ENCO MRES
ENCO ERES
MODE E_INC or **E_ABS**
ENCO BKL
ENABLE STALL
ENABLE MOVKILL
ENCO DBND
POSM GAIN
POSM MAXVEL
ENABLE POSM
ENABLE NDBWAIT

If you will not be using a particular function, it is not necessary to define the associated set-up commands (i.e., **ENCO DBND**, **POSM GAIN**, and **POSM MAXVEL** for position maintenance).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example

The example below illustrates the features discussed in the previous paragraphs. The first statement defines the motor resolutions. The next statement defines the number of encoder steps per encoder rev. Standard 1000-line encoders are used on all axes that produce 4000 quadrature steps/rev. The encoder backlash is defined in steps that the controller will produce before it expects to see an encoder step back. The stall detect is turned on and **MOVKILL** is also enabled. This means if a stall is detected, the motor's movement is killed. The next group of statements defines the deadband width, and enables the position maintenance function. The Model 4000 will attempt to move to the correct encoder position until it is within the specified deadband. The **ENABLE NDBWAIT** command instructs the Model 4000 to wait until the move is within the deadband before proceeding to the next statement.

```
ENCO MRES 25000 25000 5000 200
ENCO ERES 4000 4000 4000 4000
ENCO BKL 100 100 100 100
ENABLE STALL YES YES YES YES
ENABLE MOVKIL YES YES YES YES
ENCO DBND 0 0 10 10
POSM GAIN 0.01 0.01 0.01 0.01
POSM MAXVEL 1    1     1     1
ENABLE POSM YES YES YES YES
ENABLE NDBWAIT YES YES YES YES
```

Absolute Encoders

The main advantage of using absolute encoders is that homing after a motor stall, slip, or after power-up is not required, because the absolute position is always known.

The Model 4000 is compatible with the Compumotor AR-C Absolute Encoder. The AR-C is a 512 turn rotary absolute encoder.

The Model 4000 has several software statements which relate to absolute encoders. The **ENCO ABS** statement is used to detect the presence of an absolute encoder connected to any axis. If the **ENCO ABS** statement specifies that an axis has an absolute encoder attached, and the Model 4000 cannot read the absolute encoder on that axis, an execute error will occur. In this case, you should verify connections and power to the encoders. The **ENCO ERES** statement must be executed prior to the **ENCO ABS** statement.

After the **ENCO ABS** statement is executed, all specified axes with absolute encoders attached will use the absolute encoder in place of the incremental encoder inputs. All subsequent statements and functions which use the encoder position of an axis will then use the absolute encoder position for that axis. These statements are listed below.

Statement	Description
ENCO ERES	Must be executed before the Model 4000 starts using the absolute encoder. Also specifies the number of encoder steps per motor revolution.
ON EABSLIM	When an absolute encoder roll-over is detected the branch is taken.
DISPLAY EABS	Repetitively displays the absolute encoder position on the LCD.
ON EABS	Program branches after the absolute position is achieved.
IN POS EABS	Stores absolute encoder position into a variable.
MODE E_ABS	Defines MOVE statement parameters to be based on absolute encoder positions.
PDEF (after **MODE E_ABS** is executed)	Sets an offset from the actual absolute encoder position. This new offset position is used in all other statements (i.e., MOVE statement parameters, IN POS, ON POS, DISPLAY).
ENABLE STALL	Enables stall detection using the absolute encoder.
ENABLE POSM	Enables position maintenance using the absolute encoder.
ENABLE NDBWAIT	Suspends program execution until load is within deadband.
ENCO BKL	Sets backlash for stall detection prevention in absolute encoder steps.
ENCO DBND	Sets deadband for position maintenance in absolute encoder steps.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The Model 4000 supports stall detection and position maintenance on absolute encoder axes. It also detects and corrects for isolated bad reads of the absolute encoder which may be caused by electrical noise. If two sequential invalid reads occur, the Model 4000 will stop the move and abort the program with an execute error.

When using the AR-C encoder. The Model 4000 internally will keep track of how many times the encoder rolls-over. The user will not be aware that the encoder has rolled-over unless the **ON ENABSLIM** statement is executed. By default, the AR-C encoder is set for 16,384 steps/rev and has a range of ±4,194,303 steps. If you program the Model 4000 to move beyond that range, the Model 4000 will keep track of roll-overs within a range of ±2,147,483,648. However, if the Model 4000 is powered down the number of roll-overs of the AR-C encoder is lost and the Model 4000 will think the load is very far away from where it actually is. Make sure the load is within the same range of ±4,194,303 steps on power-up of the Model 4000 as it is when the Model 4000 is powered down.

The AR-C's resolution may be changed from 16,384 to 8,192, 4096, or 2,048 by switch settings in the AR-C Decoder Box. Changing the resolution will also change the roll-over point. The roll-over point is equal to ±(256) * (resolution selected). The Model 4000 becomes aware of the switch settings in the AR-C Decoder Box when the **ENCO ABS** statement is executed.

Inputs/Outputs

An important part of the Model 4000 is its extensive input and output ability and the statements that use this I/O. Topics covered here include Discrete I/O, Analog Inputs, and System Data.

Discrete I/O

The inputs referred to as **IN24** and **TRIGs** in *Chapter 4, Software Reference* are provided to allow the Model 4000 to detect and respond to the state of switches, thumbwheels, outputs of other equipment such as PLC's, and electronic sensors. The Model 4000 can either poll these inputs (see the **IF** and **WAIT** statements) or be interrupted (see the **ON** statement) by a change in input status.

Based on the state of the inputs, the Model 4000 can make program flow decisions and assign values to variables for subsequent **MATH** statements or tests. The outputs referred to as **POBs** and **OUT24** allow the Model 4000 to interact with PLCs or indicators attached as part of a system.

The **TRIGs** and **POBs** are meant to be associated with individual axis operation, and are accessible via the X command language as well as various Model 4000 statements. They have easy screw terminal connections well suited to small input needs. The **IN24** and **OUT24** offer greater flexibility, and can be used either as data inputs for assignment to variables or as discrete inputs. When used as data input, the **IN24** statement can read the inputs as BCD digits or as hex data, and when used as discrete inputs the **IN24** inputs can be compared to a bit pattern (see **PATT**) or read as individual bits.

☛ Helpful Hint: The **DEFINE** statement sets the logic level parameters.

When used as data output, the **OUT OUT24** statement can output data as BCD digits or as hex data. When used as discrete outputs, the **OUT24** can output a bit pattern (see **PATT**) or individual bits. You can define the logic of his 24 inputs and 24 outputs as positive or negative.

Command Summary

Refer to the following commands in *Chapter 4, Software Reference* for detailed information on how the discrete I/O can be used within a program:

```
DEFINE IN24          IN BIT           OUT OUT24
DEFINE OUT24         PATT             OUT BIT
IF IN24              IN TRIG          OUT POB
IF BIT               ON IN24          WAIT FOR IN24
IF TRIG              ON BIT           WAIT FOR BIT
IN IN24              ON TRIG          WAIT FOR TRIG
```

System Data

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

It is often required to communicate with a machine operator, a host computer, or a PLC ASCII module during the course of a program. The front panel of the Model 4000 can be very useful in receiving or sending data from or to the machine operator. The **IN LCD** statement allows variable entry for things such as speeds, move distances, cycle counts, etc.. The **OUT LCD** and **DISPLAY** statements can be used to prompt the operator and provide information on current operation. The front panel can also be used as a sophisticated operator interface with many menu levels for all aspects of the operation. These capabilities are described in the *Operator Interfacing* section of this chapter.

The **IN PORT** and **OUT PORT** statements can similarly be used to communicate system data between the Model 4000 and some remote device. More details on other remote commands available in addition to **IN PORT** and **OUT PORT** statements executed from within a program can be found in the *Remote Operation* section of this chapter.

Analog Inputs

The Model 4000 has four 8-bit analog inputs referred to as channels one through four (CH1 - CH4). These inputs may be used as joystick control for any axis, an analog input for velocity scaling, or as a variable.

The analog inputs are configured as four discrete single-ended inputs, with an input range of zero to 2.5V. When used as a joystick input or as a velocity scaling input, the input will be connected to a potentiometer. Because of travel limitations in the potentiometers and voltage drops along the cables, it may not be possible to achieve the full zero to 2.5V range at the input. The **DEFINE END_DB** statement allows the user to define the top and bottom end of the voltage range which will be used for joystick or velocity scaling input.

Velocity Scaling

Velocity scaling, activated with the **ENABLE VSCALE** command, allows an external voltage to over-ride the speed of the axes. Once enabled, all 4 axes' velocity will be scaled by the analog voltage. If full voltage (2.5V) is present at the channel used as the vscale source (see **DEFINE VSCL_L** and **DEFINE VSCL_H**), then the axes will travel at the last commanded **VEL**, if one half full voltage is present (1.25V), the axes will move at 1/2 * vel. See *Chapter 4, Software Reference* under the velocity scaling commands for more information. For velocity scaling, full-scale velocity is simply the instantaneous velocity commanded. For velocity scaling, the percentage can be expressed as below:

Percentage = (V_in - end _db) / (2.5V - 2 * end_db).
where : V_in is the input voltage end_db is the user-defined end deadband voltage.

Joystick

When the analog inputs are for joystick, the center voltage and center deadband also become important. The **DEFINE CENTER** statement allows you to specify what voltage represents the center joystick position. The **AUTOREAD** option of that statement will cause the Model 4000 to read the current voltage and use that as the center definition. The **DEFINE CTR_DB** statement specifies the voltage *no motion* deadband about the center voltage. This allows minor voltage drift or variation in joystick center position without causing motion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

When the joystick is enabled, the voltage must first be within center deadband before motion will start. (If the joystick is enabled with the joystick at full deflection, the joystick must be moved to center deadband.) The joystick must also enter the deadband region after the select input switch in order to allow motion.

The **DEFINE END_DB**, **CENTER**, and **CTR_DB** statements allow complete specification of the use of the analog inputs in determining the percentage of full-scale velocity for joystick and velocity scaling. For the joystick, full-scale velocity is defined with the **VEL JOYL** and **VEL JOYH** statements.

For CW motion:

Percentage = (V_in - V_ctrh) / (V_endh - V_ctrh)
where : V_ctrh = center voltage + center deadband V_endh = 2.5V - end deadband

For CCW motion:

Percentage = (V_ctrl - V_in) / (V_ctrl - V_endl)
where: V_endl = end deadband V_ctrl = center voltage - center deadband

Both velocity scaling and joystick features use the joystick select input to allow the user switch sources with an external switch. The **DEFINE JOYSL** and **DEFINE JOYSH** statements define which channels are to be used with which joystick axes when the joystick select input is low or high respectively. The **DEFINE VSCL_L** and **DEFINE VSCL_H** statements define which channels or **Q** variables are to be used for velocity scaling when the joystick select input is low or high respectively. The following example illustrates the use of these features.

A typical application might require that there be four axes of motion control, all of which require joystick control. In addition, the velocity scaling is to be controlled with a potentiometer located near the joystick when the operator is controlling the process, but controlled with **Q1** while under normal operation. The operator can take control of either joystick or velocity scaling at the touch of a button.

The voltage level of any one of the four analog inputs can be stored into a variable and then displayed (or used in other ways) using the **IN ANALOG** statement. The **IN ANALOG** statement will store a value from 0 to 2.5V in increments of 0.01V. The **OUT LCD** statement allows that variable value to be displayed on the LCD.

Example

In the example shown here, there are four separate programs. The program name does not appear as a statement within a program, but is created either with the front panel **EDITOR**, or with the remote port **SELECT** command. It is assumed with the **GOSUB** statements in the first program that the next three programs are named **JOYSTICK**, **VSCALE**, and **EXVSCALE**, in that order.

```
UNIT VEL 25000 25000 25000 25000          'Set units of velocity for rps.
VEL JOYL 5 5 5 5                          'Set joy vel to 5 when joy vel input is low.
VEL JOYH 10 10 10 10                      'Set joy vel to 10 when joy vel input is high.
MATH Q1 = 50                              'Q1 is used to scale whichever velocity the most
                                          'recent VEL statement commanded. 50 implies 50%
                                          'of the VEL statements value is used.

DEFINE VSCL_L SOURCE = Q1                 'When in main program, velocity scaling
DEFINE 'VSCL_H SOURCE = Q1                'is controlled by the variable Q1.
ENABLE VSCALE  YES                        'Enable velocity scaling.
LABEL L1
OUT LCD2,01 ^Use PROGVS to control^
OUT LCD2,21 ^1 VSCALE w/ program^
OUT LCD3,01 ^Use MAN_VS to control^
OUT LCD3,21 ^1 VSCALE manually^
OUT LCD4,21 ^PROGVS MAN_VS^
ON JOYTRIG GOSUB JOYSTICK                 'Whenever joytrigger button is pressed, gosub to
                                          'the joystick code.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
ON F-KEY6 GOSUB VSCALE                    'Enter manual vscale mode.
ON F-KEY5 GOSUB EXVSCALE                  'Exit manual vscale mode.
GOTO  L1
'This program defines the analog inputs for joystick mode and then enters Joystick mode.
'The following statements must be entered in a program named JOYSTICK.
DEFINE JOYSL CH3 CH4 OFF OFF              'Control axis 1 and 2 with analog  channels 3
                                          'and 4 when joystick select switch is low
                                          '(current flowing, 0V).
DEFINE JOYSH OFF OFF CH3 CH4              'Control axes 3 and 4 with analog  channels 3
                                          'and 4 when joystick select switch is high
                                          '(current not flowing, 0V).
DEFINE CENTER CH3 = AUTOREAD              'Analog inputs 3 and 4 are used for the
DEFINE 'CENTER CH4 = AUTOREAD             'joystick, here we read current input voltage
                                          'and assign it as the no motion commanded
                                          'voltage.
DEFINE CTR_DB CH3 = .2 VOLTS              'Set deadband within which no motion is
                                          'commanded.
DEFINE CTR_DB CH4 = .2 VOLTS              'Set deadband within which no motion is
                                          'commanded.
ENABLE JOY YES YES YES YES                'Program waits here (while joystick is active)
                                          'for the joystick release input to become active
                                          '(current not flowing, 5V).
WAIT FOR .1 SECONDS                       'Debounce joytrigger button, in case joy release
                                          'on
DONE
'This program sets the velocity scaling source to be analog input 1.
'The following statements must be entered in a program named VSCALE.

                                          'Here to enter velocity scaling using a
                                          'potentiometer connected to analog input 1.
DEFINE VSCL_L SOURCE = CH1                'Set the source of vel scaling to be channel 1,
DEFINE VSCL_H SOURCE = CH1                'regardless of the joy select input value.
DONE                                      'Return to caller of VSCALE.
'This program sets the velocity scaling source to be variable Q1.
'The following statements must be entered in a program named EXVSCALE.
DEFINE VSCL_L SOURCE = Q1                 'Return control of velocity scaling to Q1.
DEFINE VSCL_H SOURCE = Q1                 'Return control of velocity scaling to Q1.
DONE                                      'Return to caller of EXVSCALE.
```

Notice that program **MAIN** calls other programs (**JOYSTICK**, **VSCALE**, **EXVSCALE**) and that the called programs do not have **RETURN** statements.

A **RETURN** statement in the called program is optional when calling a program.

Joystick Teach Mode

The analog joystick inputs can also be used to teach axis positions for later execution. Absolute positions can be stored as actual values or set equal to numeric Q variables. For more information refer to the **ENABLE TCH_JOY** statement and the programming examples at the **ENABLE TCH_JOG** statement.

Command Summary

The following is a summary of the statements used in **JOYSTICK** mode and velocity scaling in the order recommended for execution:

Joystick Statements	Vscale Statements
DEFINE JOYSL	DEFINE VSCL_L
DEFINE JOYSH	DEFINE VSCL_H
DEFINE CENTER	DEFINE END_DB (optional)
DEFINE CTR_DB (optional)	ACCEL VSCALE
DEFINE END_DB (optional)	ENABLE VSCALE
VEL JOYL	
VEL JOYH	
ACCEL JOY	
DECEL JOY	
ON JOYTRIG	
ENABLE JOY or ENABLE TCH_JOY	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Program Flow

A discussion of program flow is appropriate before an example is given of I/O use, because most of the decisions and control based on I/O will be implemented via program flow control. Program flow refers to the order in which statements will be executed, and whether they will be executed at all. In general, statements are executed in the order in which they are viewed from the **EDITOR** or as **PRINT**ed. The exceptions are with the **GOTO** statement, the **GOSUB** and **RETURN** pair, and the **LOOP** and **ENDLOOP** pair.

This section discusses the following aspects of program flow:

- GOTO/GOSUB
- Loop
- Wait
- Polling and Interrupts

Goto/ Gosub

GOTO is a branch without a return to a group of statements starting with a **LABEL** or another program name. **GOSUB** is a compact way to execute a group of statements starting with a **LABEL** and ending with a **RETURN**, then proceeding with the statement following the **GOSUB**. You can also **GOSUB** another program, and in this case the **RETURN** statement is optional. If the Model 4000 reaches the end of a program it will return to the statement following the **GOSUB** automatically. Both **GOTO**'s and **GOSUB**'s require **LABEL**s or program names as destinations and both can be used either unconditionally or as part of **IF** or **ON** statements.

Loop

The **LOOP** and **ENDLOOP** pair is a convenient way to execute a group of statements a pre-determined number of times without having to repeat those statements. The Model 4000 keeps track of the fact that it is executing a **LOOP** group or a **GOSUB** subroutine, and expects to find a corresponding **ENDLOOP** or **RETURN** statement. If the program branches out of one of these loops or subroutines via a **GOTO**, it is important that the statements following the destination label include **BREAK**. This relieves the Model 4000 of the expectation of finding an **ENDLOOP** or **RETURN**. Failure to do so may cause unexpected results or error messages.

Wait

The **WAIT** statement suspends program flow until the specified condition is met. A variety of conditions can be Waited on, including input patterns, time, move complete, and others. **WAIT** shares many conditions in common with **IF** and **ON**.

The following example demonstrates the use of most of the I/O and program flow discussed earlier. The example is coded for the situation in which a Model 4000 reads a digital voltmeter and takes action based on the value. The value is always displayed on line 2 of the operator panel and sent to a PLC via the outputs. If the value is to large, a warning is sent to Port1, output POB1 is set to turn on an alarm, and the Model 4000 waits for **TRIG1** to indicate the system is OK before proceeding. The program starts by announcing it is ready and asking the you for the maximum allowed value.

```
OUT LCD2,1 ^Ready to read^                  'Operator to enter max value via
                                            'the front panel
IN Q1 = LCD3,1 ^Maximum value?^
LABEL MAIN                                  'Start of main loop
OUT LCD2,1 BLANK TO END OF LINE             'Clear previous text
OUT LCD2,1 ^Reading values^
LOOP 0                                      'Loop forever
GOSUB VREAD                                 'Read into Q2
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
IF Q2 > Q1 GOTO DANGER                    'Check for max value
ENDLOOP                                   'Do next check

LABEL DANGER                              'Max value exceeded subroutine
BREAK                                     'Break from loop count
OUT PORT1 = ^Excess value = ^ Q2          'Send warning
OUT POB 1XXX                              'Set POB1
OUT LCD2,1 BLANK TO END OF LINE           'Clear previous text
OUT LCD2,1 ^Waiting for OK^               'Inform operator
WAIT FOR TRIG1 = 1                        'Wait for OK signal
GOTO MAIN                                 'Go back to loop

LABEL VREAD                               'Read and display subroutine
IN Q2 = IN24 BCD 4.2                      'Read voltage to .01V
OUT OUT24 Q2 BCD 4.2                      'Send to PLC
OUT LCD3,1 BLANK TO END OF LINE           'Clear previous text
OUT LCD3,1 ^Voltage is now^ Q2            'Display new value
RETURN                                    'Return to loop
```

Polling and Interrupts

There are two general methods to allow the Model 4000 to respond to a change or a condition. The simplest is through polling, which is implemented with the **IF** statement. Polling is useful when program flow is in the form that takes it through **LOOPs** or through the same section of program statements via **GOTO** statements. Polling requires the execution of an **IF** statement, so it is appropriate for conditions which only need to be monitored while executing that section of program statements. The **IF** statement will check a condition **once** and branch if the condition is met. If the condition is not met, the branch will not occur and the next program statement will execute.

Interrupts are implemented via the **ON** statement, and are used to become aware of conditions that are of concern without having to repeatedly execute **IF** statements. The interrupt can occur at any point in program execution, and is serviced in the section of program statements that is the destination of the **ON GOTO** or **ON GOSUB**. This program section is called the interrupt service routine. The **ON** statement merely sets up the Model 4000 to continuously monitor the condition, and is generally executed along with setup statements for initialization. Inside the appropriate interrupt service routine, the **ON** condition **must** be re-enabled if the branch is to be taken again. This is illustrated below:

```
ON BIT3 = 1 GOSUB EXAMPLE
•
•
•
LABEL EXAMPLE
•
•
•
ON BIT3 = 1 GOSUB EXAMPLE
RETURN
```

☛ Important note about **ON** condition interrupts

An important point to note about **ON** condition interrupts is that they do affect overall program execution speed. The more **ON** conditions requiring service, the longer each individual program statement will take to execute. When **ON** conditions are not needed, it is advised to turn them off.

Each type of **ON** statement can be disabled individually by re-executing that **ON** statement with the word **DISABLE** substituted for **GOTO LABELØ**. Each statement is individually re-enabled by re-executing the original **ON** statement. All **ON** statements can be disabled using the **ENABLE ON NO**, and re-enabled with the **ENABLE ON YES**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

All **ON** statements are polled at one time within the Model 4000 every 2ms to 8ms. They are checked after every statement executes and within statements that take longer than 6ms to execute. The first **ON** statement found to be true during that polling is the one serviced. The **ON** statement descriptions in *Chapter 4, Software Design* provide the order in which the **ON** statements are checked. Some types of **ON** statements can be set to interrupt on more than one condition. For instance, twenty different input patterns can be checked at the same time using twenty **ON PATT** statements.

Example

The following example illustrates the difference in the use of polling and interrupts. It builds on the previous example and adds motion control. The **DANGER** and **VREAD** routines are not repeated here. In this case, moving the motors in the positive direction tends to cause the voltage read by the digital voltmeter to increase. **TRIG 2** is to be used as an interrupt to request the motors to move in the negative direction. The program is always checking to see if **TRIG 2** is asserted, even when waiting for **TRIG 1** as the OK signal. In the **TRIG 2** service routine, **TRIG 2** is polled until it is no longer asserted. Control returns to the main loop.

```
ON TRIG2 = 1  GOTO BACKOFF                'Initialize interrupt
OUT LCD2,1 ^Ready to read^
IN Q1 = LCD3,1 ^Maximum value?^
LABEL MAIN                                'Start of main loop
OUT LCD2,1 BLANK TO END OF LINE           'Clear previous text
OUT LCD2,1 ^Reading values^
LOOP 0                                    'Loop forever
MOVE 10 20 * *                            'Move in positive dir.
GOSUB VREAD                               'Read into Q2
IF Q2 > Q1 GOTO DANGER                    'Check for max value
NDLOOP                                    'Do next check

LABEL BACKOFF
BREAK                                     'Break any loops
MOVE -10 -20 * *                          'Move in negative dir.
GOSUB VREAD                               'Read into Q2
IF TRIG2 = 1 GOTO BACKOFF                 'Check for trig 2
ON TRIG2 = 1 GOTO BACKOFF                 'Initialize interrupt
GOTO MAIN
```

Operator Interfacing

The Model 4000 has many ways to provide a convenient interface to the operator or a computer. The previous sections have already illustrated methods of prompting the operator via the **IN Q2 = LCD** statement, and sending messages to the **LCD** and to **PORT1**. By using the **ON F-KEY** or **F-KEY** statement and the **OUT LCD** statement it is possible to create menus over the function keys. These menus can be customized to change according to the current situation under program control.

Example

The following example illustrates how to make a simple menu system.

```
LABEL MENU
    GOSUB MENU1                           'Label the function keys and
                                          'turn on function key
                                          'interrupts
    LABEL LOOP_IT                         'The program will pause here
                                          'waiting for an F-KEY
    GOTO LOOP_IT                          'It is necessary to have 2 "GOTO
                                          'LOOP_IT" statements
DONE
SUBROUTINES FOLLOW
'
LABEL MENU1
    GOSUB CLRSCRN
    OUT LCD4,1 ^SETUP RUN^                'Label Function keys
    ON F-KEY1 GOSUB SETUP
    ON F-KEY2 GOSUB HOME
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    ON F-KEY3 GOSUB RUN
RETURN

LABEL SETUP
    GOSUB CLRSCRN
    OUT LCD4,1 ^START PAUSE RETURN^          'Label function keys
    ON F-KEY1 GOSUB START
    ON F-KEY2 GOSUB PAUSE
    ON F-KEY3 GOSUB RETURN
RETURN

LABEL HOME
    'INSERT HOMING ROUTINE HERE
    ON F-KEY2 GOSUB HOME                     'Re-enable function key
RETURN

LABEL RUN
    'INSERT RUN SETUP ROUTINE HERE
    ON F-KEY3 GOSUB RUN                      'Re-enable function key
RETURN

LABEL START
    'INSERT MACHINE'S START ROUTINE HERE
    ON F-KEY1 GOSUB START                    'Re-enable function key
RETURN

LABEL PAUSE
    'INSERT MACHINE'S PAUSE ROUTINE HERE
    ON F-KEY2 GOSUB PAUSE                    'Re-enable function key
RETURN

LABEL RETURN
    GOSUB MENU1                              'Restore's MENU1
RETURN

'THIS SUBROUTINE CLEARS THE ENTIRE DISPLAY
LABEL CLRSCRN
    OUT LCD1,1 BLANK TO END OF LINE
    OUT LCD2,1 BLANK TO END OF LINE
    OUT LCD3,1 BLANK TO END OF LINE
    OUT LCD4,1 BLANK TO END OF LINE
RETURN
*
```

☛ Important note about **ON** condition interrupts

In the example above **ON** statements were used to set up the menu structure. Although this works fine, a word of caution is called for. Unless the **ON** conditions are disabled, a user could mistakenly push an **F-KEY** during an operation and the most recently defined branch *will still take place.*

A more fail-safe way to program a menu structure is to loop around **IF** statements. For example, the set-up subroutine above could also be written as follows:

```
LABEL SETUP
GOSUB CLRSCRN                        'Clear the LCD screen
GOSUB CLRSCRN
OUT LCD4,1 ^START PAUSE RETURN^      'Put up prompt message
LABEL SETUP1
IF F-KEY1 GOTO START                 'Intermediate loop
IF F-KEY2 GOTO PAUSE                 'Here we loop continually until one of
IF F-KEY3 GOTO RETURN                'the function keys is pressed
GOTO SETUP1
```

With the above example there are no **ON** statements to disable for safety concerns. The only time a function key will cause a program branch is when intended.

Example

The following example illustrates the use of custom menus and teaching with **JOG MODE**. The lines before the **LABEL TEACH** write the menus above **F1** and **F2**, and enable those keys to teach moves and normal moves respectively. Depending on which key is pressed, the moves are either taught using **JOG** or performed using the previously taught values.

```
MATH Q1 = 0                          'Q1 is a flag, 0=not in teach mode
                                     '1=in teach mode

ENABLE HLIMIT NO NO NO NO            'Disable limit inputs (unless you have
                                     'them wired)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
LABEL GET_KEY                          'Beginning of loop
OUT LCD3,01 ^TEACH DO^                 'Create message on LCD display
OUT LCD4,01 ^MOVES MOVE^
ON F_KEY1 GOTO TEACH                   'Allow user to select jog mode to teach
ON F_KEY2 GOTO DO_MOVES                'New positions for the move
LABEL WAITL                            'Wait here forever or until a key is
                                       'pressed
GOTO WAITL
LABEL TEACH                            'Start here for teach mode
ENABLE TCH_JOG YES YES YES YES         'Enable teach mode using front panel
                                       'arrow keys
MATH Q1 = 1                            'Set flag to be in teach mode
LABEL DO_MOVES
MOVE Ø Ø Ø Ø                           'These are the move statements
MOVE Ø Ø Ø Ø                           'that are modified when in teach mode
•
•
•
MOVE Ø Ø Ø Ø                           'If not in teach mode, go back
MOVE Ø Ø Ø Ø                           'and wait for another key
IF Q1 = 0 GOTO GET_KEY                 'If in teach mode, take out of teach
ENABLE TCH_JOG NO NO NO NO             'mode
MATH Q1 = Ø                            'Set flag to not in teach mode status
GOTO GET_KEY                           'Go back and wait for another key
```

Error Handling

In all applications there are many possibilities for system error from which the Model 4000 can recover. Types of errors include: invalid operator entry, stall condition, limit hit, **ESTOP** activated, drive fault and many others. The interrupt and variable handling capabilities of the Model 4000 allow graceful recovery from almost any error.

In the example below, assume the operator is prompted many times during the program for various pieces of data. If the operator inadvertently presses the enter key without entering a number via the front panel keypad, normally the program would crash due to invalid data. The **ON ERROR** and **IN ERROR** statements allow recovery from the error and the program will no longer crash.

```
ON ERROR GOTO ERROR                          'Enable error handling routine
.
.
.
LABEL IN-VEL                                 'Data entry subroutine
MATH Q1 = 1                                  'Set flag variable for entry
                                             'tracking
IN Q10 = LCD1,1 ^ENTER DESIRED VELOCITY^     'Enter velocity
VEL Q10 * * *
LABEL IN_ACCEL
MATH Q1 = 2
IN Q11 = LCD2,1 ^ENTER DESIRED ACCEL^        'Enter acceleration
ACCEL Q11 * * *
LABEL IN_DIST
MATH Q1 = 3
IN Q12 = LCD3,1 ^ENTER DESIRED DISTANCE^     'Enter distance
MOVE Q12 * * *                               'Move using above parameters
GOTO IN_VEL                                  'Repeat sequence
.
.
.
LABEL ERROR                                  'Error subroutine
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
IN Q99 = ERROR CODE                          'Read error code into variable
IF Q99 = 28 GOTO BAD_DATA                    'Check for bad data entered
                                             '(see IN ERROR statement)
OUT LCD1,1 ^SYSTEM ERROR^                    'Recovery from general error
OUT LCD2,1 ^PRESS F-KEY1 TO RESTART^
WAIT FOR F-KEY1
GOTO BEG                                     'Begin again
LABEL BAD_DATA                               'Bad data entry
OUT LCD4,1 ^BAD OPERATOR ENTRY, TRY AGAIN^   'Prompt operator to enter data
                                             'again
WAIT FOR 2 SECONDS
OUT LCD4,1 BLANK
IF Q1 = 1 GOTO IN_VEL                        'Only force data re-entry from
                                             'where the error took place
IF Q1 = 2 GOTO IN_ACCEL
IF Q1 = 3 GOT IN_DIST
```

Command Summary

Operator Interface Commands	Error Handling Commands
IF (Front Panel Keys)	IN STATUS
ON (Front Panel Keys)	ON ESTOP
IN LCD	ON ERROR
IN PORT	ON FAULT
WAIT FOR F-KEY	ON STALL
LABEL	ON M_INC
GOTO	ON M_ABS
GOSUB	ON E_INC
OUT LCD	ON E_ABS
DISPLAY	ON TIME
	SHTDN

Complex Moves

In addition to the standard continuous and preset moves, the Model 4000 offers three complex move types. These move profiles allow you to perform specialized moves involving predefined velocities and distances without coming to a stop.

- ❑ Homing sequence (**DEFINE HOME**)
- ❑ Segmented move
- ❑ Registration move

Homing Sequence

The **HOMECW** or **HOMECCW** parameters initiate a sequence of moves that position the axis using the Home Enable (**HE**) and/or the Z channel inputs. The result of any move home sequence is to approach the previously defined edge of an input (**HE** or **Z**). The home inputs to be used, the edge of those inputs, and the final approach direction may all be defined by the user. When **DEFINE HOME TYPE** is selected to be **HE+** or **HE-** the **MOVE HOME** sequence only uses the **HE** input, the **Z** channel input is ignored. When the **DEFINE HOME TYPE** is selected to be **HE+Z** or **HE-Z** the **MOVE HOME** sequence uses both the **HE** input and the Z channel input. The input polarity of the **HE** input is also defined with the **DEFINE HOME TYPE** statement. When **HE+** or **HE+Z** is used, the **HE** input is active high. The **HE-** or **HE-Z** is used the **HE** input is active low.

The figures below show the following move sequences.

The **HOMECW** move sequence (3 examples)

The **HOMECCW** move sequence (1 example)

(The Z channel is not included in these first four examples)

The **HOMECW** move sequence is shown when the Z channel is included (1 example)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Assumptions for all examples below:

DEFINE HOME EDGECW (CW is active edge)

DEFINE HOME FDIR CW (Final approach direction is CW)

It is important to note that if the **HE** edge direction and the final approach directions are the same, the **GO HOME** sequence will end just outside the home enable active region. If the **HE** and final approach directions are opposite, the **GO HOME** sequence will end inside the home enable active region.

The Final Home Velocity or *creep* speed is always 2,500 steps/sec, regardless of the motor or encoder resolution.

Example 1

```
DEFINE HOME EDGE CCW
DEFINE HOME FDIR CW
DEFINE HOME BACK YES
MOVE HOMECW
```

① The **HOMECW** move is started in the CW direction at the **ACCEL HOME** acceleration. Default **ACCEL HOME** is 2,500,000 steps/sec^2

② The **VEL HOME** velocity is reached, the move continues at that velocity until the **HE** input becomes active.

③ The CCW edge of the **HE** input is detected, this means the **HE** input is active. At this time the move is decelerated if the **ACCEL HOME** value is used. It does not matter if the **HE** input becomes inactive during this deceleration.

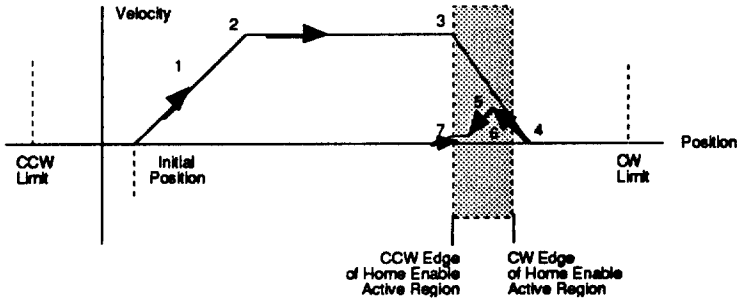
④ After stopping, the direction is reversed and a move is started that will place the load just CW of the CCW edge of the **HE** input.

⑤ The peak velocity of this move is:

V = .707 [**VEL HOME** value]

⑥ When this move decelerates to 2,500 steps/sec, the deceleration is ended and the move continues at that velocity until the CCW edge of the **HE** input is reached.

⑦ The direction is reversed and another move at 2,500 steps/sec is started in the CW direction. This is so backlash in any mechanical linkages is removed. As soon as the **HE** input becomes active this move is killed (pulses shut-off). The load is at home.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example 2

```
DEFINE HOME EDGE CCW
DEFINE HOME FDIR CW
DEFINE HOME BACK YES
MOVE HOMECW
'CW LIMIT HIT
```

① The **HOMECW** move is started in the CW direction at the **ACCEL HOME** acceleration. Default **ACCEL HOME** is 2,500,000 steps/sec^2

② The **VEL HOME** velocity is reached, the move continues at that velocity until the **HE** input becomes active.

③ A hardware input or software programmed limit is encountered while searching for the **HE** to become active. In this diagram the CW limit became active (5V, open circuit).

④ The move is decelerated at the **DECEL HLIMIT** if specified or the default value of 2,500,000 steps/sec^2 if not specified.

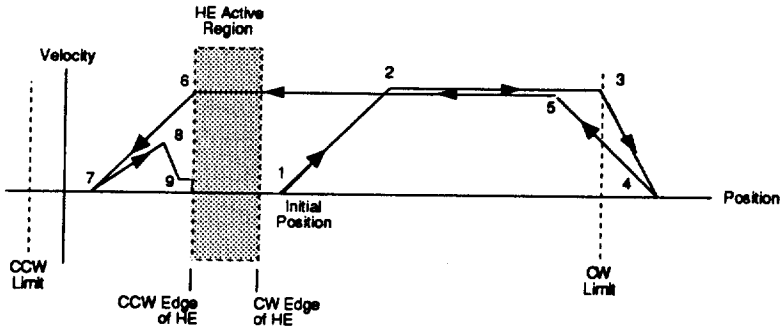
⑤ The direction is changed and the move is accelerated in the CCW direction using the **ACCEL HOME** value. If the opposite limit is reached before the **HE** active region, then the **MOVE HOME** sequence is terminated after the limit deceleration.

⑥ The CCW edge of the **HE** input is detected, this means the **HE** input has gone from inactive to active and back to inactive. At this time the move is decelerated using the **ACCEL HOME** value.

⑦ After stopping, the direction is reversed and a move is started which will place the load just CCW of the CCW edge of the **HE** input.

⑧ The move is decelerated to 2,500 steps/sec and begins to creep CW at this speed.

⑨ Once the **HE** input becomes active pulses are shutoff and the load is at home.



Example 3

```
DEFINE HOME EDGE CCW
DEFINE HOME FDIR CW
DEFINE HOME BACK YES
MOVE HOMECW
'HE INPUT ALREADY ACTIVE
```

① Although the CW direction was specified as the initial home direction (**HOMECW**), the CW direction is ignored and the move is started in the CCW direction. This is because the CCW edge must be CCW of the current position if the **HE** input is active when the **MOVE HOMECW** statement is executed.

② The move is started in the CCW direction, because the **HE** input is initially active.

③ **VEL HOME** velocity is reached

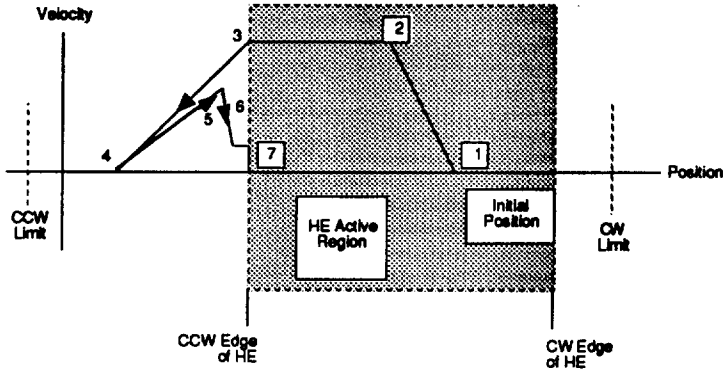
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

④ The CCW edge of the **HE** input is detected, this means the **HE** input has gone from active to inactive. At this time the move is decelerated using the **ACCEL HOME** value.

⑤ After stopping, the direction is reversed and a move is started which will place the load just CCW of the CCW edge of the **HE** input.

⑥ The move is decelerated to 2,500 steps/sec and begins to creep CW at this speed.

⑦ Once the **HE** input becomes active pulses are shutoff and the load is at home.



Note: This scenario is very common if the home type is defined incorrectly, i.e., **DEFINE HOME TYPE HE+** when the home switch being used is actually a normally open switch.

Example 4

```
DEFINE HOME EDGE CCW
DEFINE HOME FDIR CW
DEFINE HOME BACK YES
MOVE HOMECCW
```

① The move is started in the CCW direction as specified.

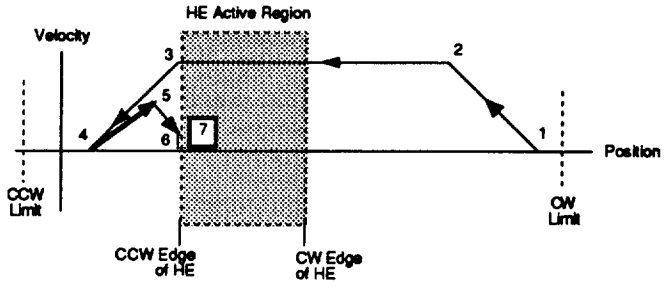
② **VEL HOME** velocity is reached

③ The CCW edge of the **HE** input is detected, this means the **HE** input has gone from inactive to active and back to inactive. At this time the move is decelerated using the **ACCEL HOME** value.

④ After stopping, the direction is reversed and a move is started which will place the load just CCW of the CCW edge of the **HE** input.

⑤ The move is decelerated to 2,500 steps/second begins to creep CW at this speed.

⑥ Once the **HE** input becomes active, pulses are shutoff and the load is at home.



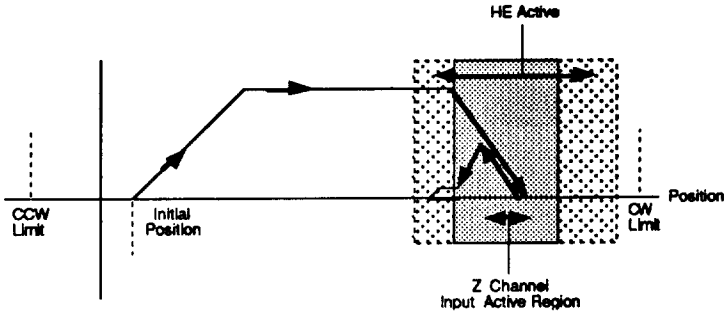
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example 5

```
DEFINE HOME EDGE CCW
DEFINE HOME FDIR CW
DEFINE HOME BACK YES
MOVE HOMECW
'ENCODER Z CHANNEL USED
```

This profile is exactly the same as Example 1, with the exception of the **Z** channel **CCW** edge is used instead of the **HE CCW** edge. *The HE input must be active while the Z channel is active.* When **DEFINE HOME TYPE** is specified as **HE+Z** or **HE-Z**, the **HE** edges do not determine the deceleration point or the final home position, the edges of the Z pulse make these determinations. To avoid multiple home locations, only one Z channel pulse should be seen within the home enable.

For **HOMECCW** with Z channel included, the profile is exactly the same as in Example 3, except the **Z** channel **CCW** edge is used instead of the **HE** input CCW edge, as described above.



Segmented Moves

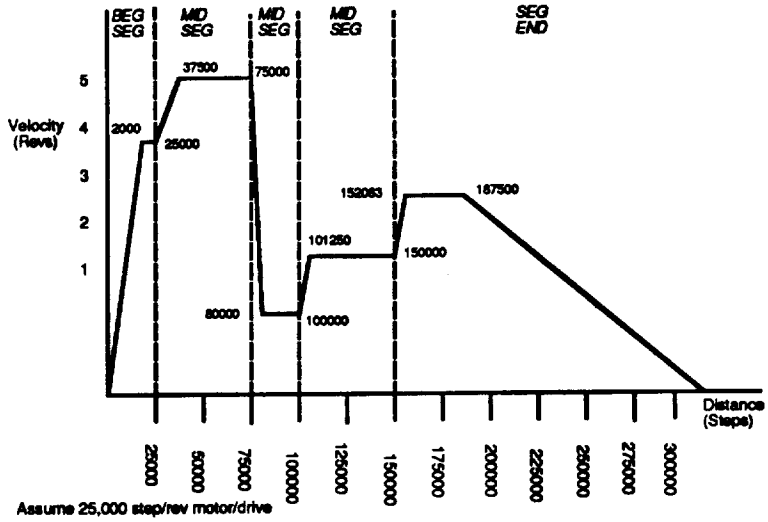
In some applications it may be necessary to describe a move profile which travels at different velocities to reach certain distances along the move, yet finishes the entire move without stopping and with a predefined final distance. A common example is that of a move which requires a rapid approach to a position close to the end of the move, followed by a slower approach to final position. The 4000 segmented move construction allows the user to define a move with up to five separate segments, each with its own velocity, acceleration, deceleration, and end of segment distance. The sum of the distances in the individual segments will be the total distance traveled by the segmented move. The distances specified for each segment definition must be *positive incremental distances*, although the entire segmented move may be executed in the positive or negative direction.

The beginning segment definition may be defined as a motor step segment or as an encoder step segment, depending on the **MODE** in affect when the segment is defined. Middle and ending segment definitions must have the same **MODE** as the beginning segment. If the **MODE** is changed after the beginning segment is defined, middle and ending segment definitions will be ignored. Beginning segment definitions start the segmented move construction process, and delete any previously defined segmented move for that axis. Segmented move definition is completed with a **SEG END** statement definition. A segmented move definition must be completed before the move can be executed. Up to three middle segments are allowed, but none are required. If three middle segments have been defined, addition middle segment definitions will be ignored. Beginning and ending segment definitions are required.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example

The following complex motion profile is to be generated with segmented moves.




```
UNIT ACCEL 25000 * * *
UNIT VEL 25000 * * *
UNIT POS 1 * * *
MODE  M_INC * * *          'Specify the mode for the segmented move.

ACCEL 10 * * *             'Accelerate at 10 rps² to 4 rps and rotate at
VEL 4 * * *                'velocity until 25000 steps from the beginning
                           'of the move have gone by.
SEG BEG 25000 * * *

ACCEL 1 * * *              'Accelerate at 1 rps² to 5 rps and rotate at
VEL 5 * * *                'this velocity until 50000 steps from the
SEG MID 50000 * * *        'beginning of the move have gone by.

DECEL 40 * * *             'Decelerate at 40 rps² to 1 rps and rotate at
VEL 1 * * *                'this velocity until 25000 steps from the
SEG MID 25000 * * *        'beginning of the move have gone by.

ACCEL 40 * * *             'Accelerate at 40 rps² to 2 rps and rotate at
VEL 2 * * *                'this velocity until 50000 steps from the
SEG MID 50000 * * *        'beginning of the move have gone by.

ACCEL 6 * * *              'Accelerate at 6 rps² to 3 rps and rotate at
DECEL 1 * * *              'this velocity until 150000 steps from the
VEL 3 * * *                'beginning of the move, minus the steps required
SEG END 150000 * * *       'to decelerate to a stop, have gone by.

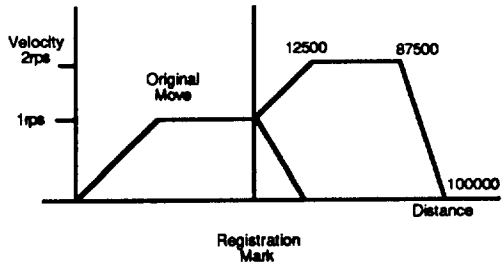
MOVE SEGCW * * *           'Make a segmented move in the CW direction.
DONE
*
```

Registration Moves

Some applications require a move whose overall length is indeterminate at the time that the move is started, but whose distance and velocity must be predefined with respect to a registration input. The Model 4000 allows the use of triggers 1 through 4 as registration interrupts, which will cause predefined moves to occur. A registration move may be defined for each trigger input separately, and for each axis separately. Thus with four triggers, each axis may have four different defined registration moves. Each trigger must be enabled separately on each axis before that trigger input will cause the defined move to take place on that axis. This allows the user to have moves defined on all four axes for a given trigger input, yet selectively enable the axes for that one input. A registration move must be defined for a given axis and trigger before the enable command for that axis and trigger will be recognized.

As with segmented moves, the registration move may be defined to be a motor step move or an encoder step move. The segment definition for registration moves will use the velocity, acceleration, deceleration, and mode currently in effect. For example, upon receiving a registration input, the following motion is desired.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com




```
UNIT POS 1 * * *
UNIT VEL 25000 * * *
UNIT ACCEL 25000 * * *
MODE M_INC * * *              'Specify the mode for the registration move.
ACCEL 1 * * *                 'Accelerate at 1 rps2.
DECEL 4 * * *                 'Decelerate at 4 rps2.
VEL 2 * * *                   'Velocity of 2 rps.
SEG REG1 100000 * * *         'Define registration move for trigger 1.
ENABLE REG YES NO NO NO       'Enable registration if trigger 1 becomes
                              'active for axis 1 only.

ACCEL 1 * * *
VEL 1 * * *
MOVE 100000 * * *             'This is the move to be interrupted by
                              'registration.

DONE
*
```

A registration move may interrupt any preset, continuous, segmented, or registration move, even if the move mode of the interrupted move does not match the move mode of the interrupting registration move. The fact that registration moves can interrupt other registration moves means that the user may construct segmented moves whose segment distances are determined by the placement of position sensors hooked to registration inputs. This would only require that the distance specified in the registration definition be long enough to reach the next position sensor. ***Registration inputs will be ignored when the motor is not performing a move, or when going home, or when in joystick mode, jog mode, or decelerating due to stop, kill, soft limit, or hard limit.***

☛You must define the **SEG REGn** statement before you define the **ENABLE REGn** statement.

Triggers default to active-high when used for registration inputs, **ENABLE TRIG REV** allows the active-level to be changed to high or low. If a trigger input is enabled as a registration input on any axis, that trigger input will be debounced before another input on that trigger will be recognized. The default Trigger Debounce time is 100 ms. Although triggers enabled as registration inputs may still be used with **IF**, **ON**, and **WAIT** statements, it will be the debounced state that is recognized by the Model 4000, so rapid transitions will not be noticed by these statements. Triggers not enabled as a registration input will not be debounced. The registration input generates an interrupt to the Model 4000. This interrupt will be serviced by the Model 4000 within 100 µs of its occurrence. Therefore, the maximum position error possible will equal velocity when registration input occurred x 100 µs. The **DEFINE TRIGDB** command is used to define the trigger debounce time.

Once a particular input is activated, it must go back to the *off* state before any other registration moves for that input will occur. In software revisions A4/A5 for 12 MHz units and D for 10 MHz units, the input debounce can be changed with the **DEFINE TRIGDB** command and the input active level can be changed with the **ENABLE TRIG REV** command.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

General Segment and Registration Definitions

Each segment definition for segmented moves and registration moves uses the velocity, acceleration, deceleration, and mode currently in effect at the time of the segment definition. To properly define a segment, the motion parameter statement must precede the segment definition. It is possible to define segments with velocities which cannot be reached in the distance specified using the acceleration or deceleration specified. If such a segment is not the final segment, the motor will simply accelerate or decelerate until the segment distance is reached, and the next segment will start at whatever velocity was reached. If such a segment is the final segment or a registration segment, and zero velocity cannot be reached within the final distance using the given deceleration, the Model 4000 will step down to a velocity that allows it to decelerate to zero velocity at final distance. In applications where the commanded motor position is not equal to the actual position (i.e., encoder feedback position lags behind the commanded position) it may be appropriate to set the **ENABLE REGSRV** for that axis. Refer to *Chapter 4, Software Reference* for an explanation of the **REGSRV** command.

The following example illustrates the construction of segmented moves and trigger #1 registration moves on axes #1, #2, and #3. The segmented move for axis #1 has 4 segments. The segmented move for axis #2 has only 3 segments. The segmented move for axis #3 has only two segments. All trigger #1 registration moves share the same motion parameters as the final segments of the segmented moves.

```
UNIT POS 25000 25000 25000 *
UNIT ACCEL 25000 25000 25000 *
UNIT VEL 25000 25000 25000 *
ACCEL 50 30 40 *
DECEL 25 15 20 *
MODE M_INC M_INC M_INC *

VEL 2 2 2 *
SEG BEG 10 20 14 *
VEL 5 6 * *
SEG MID 20 30 * *
VEL 3 * * *
SEG MID 10 * * *
VEL 1 1 1 *
SEG END 5 5 5 *
MOVE SEGCW SEGCW SEGCW *
SEG REG1 10 10 10 *
VEL 5 5 5 *

ENABLE REG1 YES YES YES NO
MOVE SEGCW SEGCW SEGCW *
DONE
```

Command Summary

Homing Commands	Segmented Move Commands	Registration Commands
DEFINE HOME TYPE	VEL	VEL
DEFINE HOME EDGE	ACCEL	ACCEL
DEFINE HOME FDIR	DECEL	DECE
DEFINE HOME BACK	SEG BEG	SEG REGn
ACCEL HOME	SEG MID	ENABLE REGn
VEL HOME	SEG END MOVE SEGCW/SEGCCW	MOVE
MOVE HOMECW/HOMECCW		DEFINE TRIGDB
		ENABLE TRIGREV
		ENABLE REGSRV

QQ Pointer Variables

QQ1 - QQ99 are pointers to locations specified by the corresponding Q variable (e.g., If Q1 = 5, then the Model 4000 statement, **MATH QQ1 = 52**, would result in the variable Q5 being set to equal 52. i.e., QQ pointers allow the programmer to specify a destination variable by using a pointer value stored in a separate variable.).

QQ variables are useful in eliminating redundant code. For example:

An operation requires a operator to enter 20 different cut lengths. One approach to program this task would be to enter the following.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
IN Q1 = LCD1,1 ^Enter variable 1^
IN Q2 = LCD1,1 ^Enter variable 2^
IN Q3 = LCD1,1 ^Enter variable 3^
•
•
•
IN Q20 = LCD1,1 ^Enter variable 20^
```

A more efficient way to code this would be to enter:

```
MATH Q21 = 1
LABEL LABEL21
OUT LCD1,1 ^Enter Variable^
IN QQ21 = LCD1,16 Q21
MATH Q21 = Q21+1
IF Q21 < 20 GOTO LABEL21
```

The second approach cut the number of code lines from 20 to 6.

It may be helpful to think of the QQ pointers like a one dimensional array in BASIC where the syntax would include parenthesis:

```
Q(Q15)       ; the array location is pointed to by variable Q15
```

Another useful feature of the QQ variable is the ability to create buffers for data entry.

The following statements implement a buffer with 59 locations. An operator will be allowed to enter data at any time, until the table is filled, at which point he must wait until the program has removed some data.

```
'This program will take input from an operator at any time
'and use it up at 10 second intervals.  The program
'uses a circular buffer, with a head and tail pointer, to
'implement this.  The buffer has 59 locations (Q1-Q60).
'There must always be 1 empty location in the buffer, that's why
'there are 59 locations instead of 60.

MATH Q70 = 1                          'Head pointer
MATH Q71 = 1                          'Tail pointer
MATH Q99 = -999999                    'Value used to designate empty buffer

OUT LCD1,1 BLANK TO END OF LINE
OUT LCD2,2 BLANK TO END OF LINE
OUT LCD3,1 BLANK TO END OF LINE
OUT LCD4,1 BLANK TO END OF LINE
OUT LCD3,34 ^Enter^
OUT LCD4,34 ^Data^

ON F-KEY6 GOSUB MOREDATA

'This is the main program
LABEL MAIN
OUT LCD1,1 BLANK TO END OF LINE
MATH Q98 = 0
  LABEL TEN_SEC                       'Begin ten second loop (not real time)
  WAIT FOR 1 SECONDS
  MATH Q98 = Q98 + 1
  OUT LCD1,1 ^Time=^ Q98
  IF Q98 < 10 GOTO TEN_SEC            'End of ten second loop
GOSUB REM_DATA
IF Q99 = -999999 GOTO MORE_MES        'If Q99=-999999, data buffer is empty
OUT LCD1,1 ^Data value=^ Q99          'Display data value on the LCD
WAIT FOR 1 SECONDS
GOTO MAIN
  LABEL MORE_MES
  OUT LCD1,1^ENTER DATA PLEASE^       'Message to display when data table empty
  WAIT FOR 1 SECONDS
  GOTO MAIN

'This subroutine places 1 data value in the buffer.

LABEL MOREDATA
MATH Q72 = Q79 - Q71                  'Compare head pointer to tail pointer
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
IF Q72 = -1 GOTO FULL                'This implies data buffer is full
IF Q72 = 59 GOTO FULL                'This implies data buffer is full
IN QQ70 = LCD2,1 ^Enter Data^        'Prompt operator for data value
IF Q70 < 60 GOTO SKIP                'If not at top of data buffer Goto SKIP
MATH Q70 = 0                         'If at top of data buffer, set Q70 equal
                                     '   to 1 less than bottom of data buffer
  LABEL SKIP
MATH Q70 = Q70 + 1                   'Increment head pointer by 1 (point to next
                                     'empty location)
GOTO DONE_MD
  LABEL FULL
OUT LCD2,1 ^Data Table Full^         'Display data full message on LCD
WAIT FOR 1 SECONDS
  LABEL DONE_MD
OUT LCD2,1 BLANK TO END OF LINE
ON F-KEY6 GOSUB MOREDATA             'Re-enable interrupt condition
RETURN

'This subroutine moves 1 data value from the buffer.
LABEL REM_DATA
MATH Q73 = Q70 - Q71                 'Compare head pointer to tail pointer
IF Q73 = 0 GOTO EMPTY                'If they are equal, data buffer is empty
MATH Q99 = QQ71                      'Read data from top of data buffer into Q99
IF Q71 < 60 GOTO SKIP1               'If not at top of data buffer, goto SKIP1
MATH Q71 = 0                         'If at top of buffer, set Q71 = bottom -1

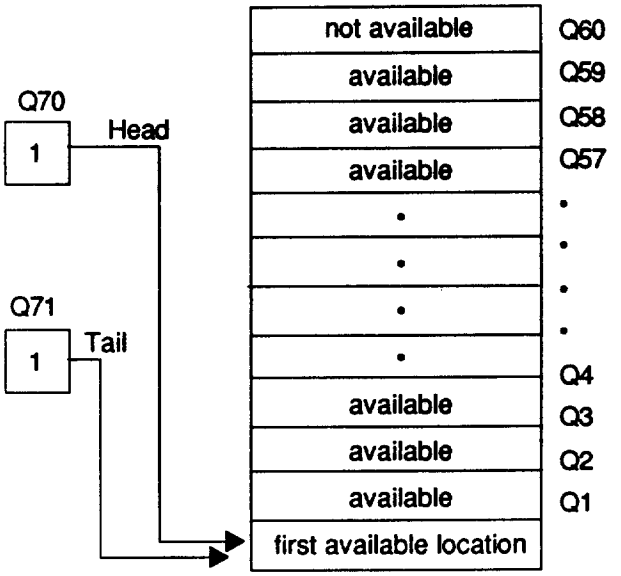
  LABEL SKIP1
MATH Q71 = Q71 +1                    'Increment tail pointer by 1 (point to
                                     '   next data point)
GOTO DONE_RD
  LABEL EMPTY
MATH Q99 = -999999                   'Data table empty, so set Q99=-999999
  LABEL DONE_RD
RETURN

*
```

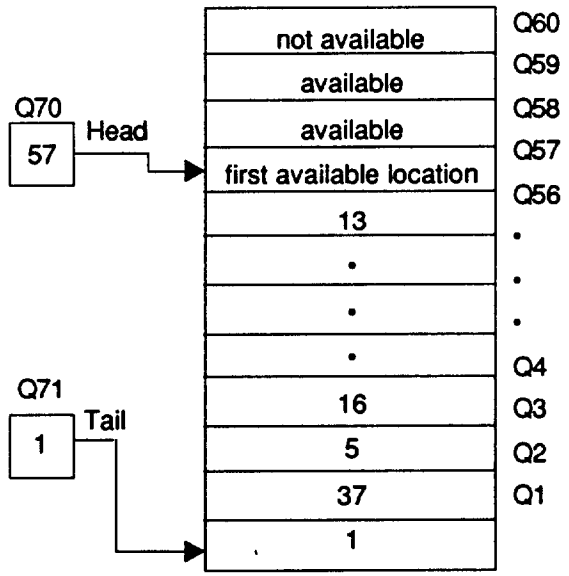

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Circular Buffer Pictorially

Empty data buffer (59 locations available)

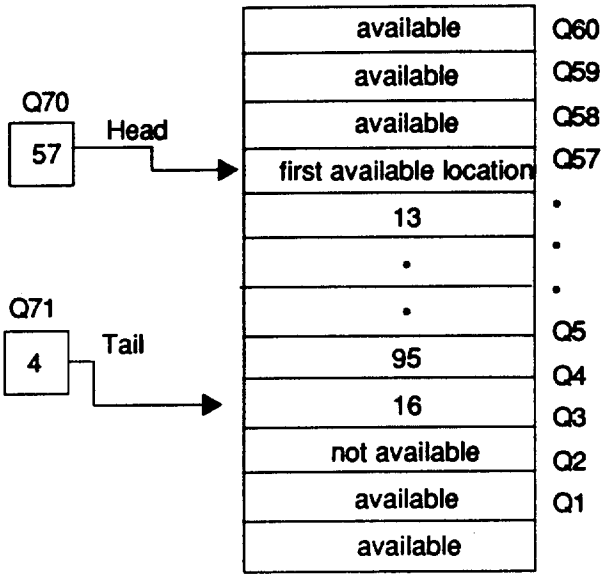


As data is added (3 locations available)

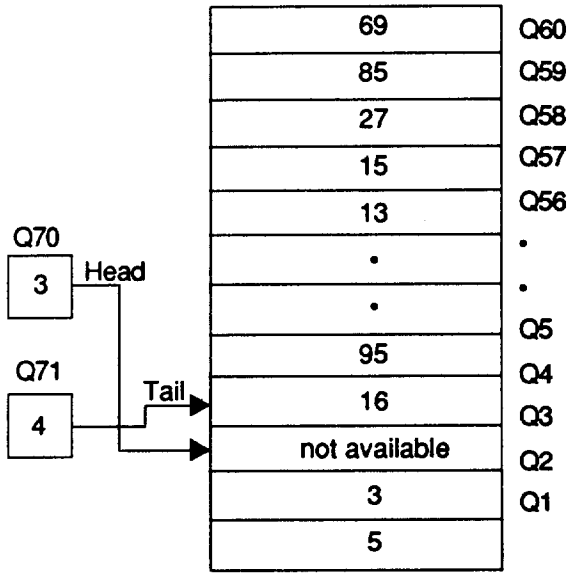


Circular Buffer Pictorially

As data is removed (6 locations available)



Full data table (0 locations available, since there must always be 1 empty location)



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Remote Operation

The Model 4000 has two general-purpose serial ports labeled PORT1 and PORT2. Both ports operate identically, and either or both may be used for remote operation. Remote operation is provided to allow complete control of the Model 4000 when the operator panel is either not present or when control from a computer or terminal is desired.

RS-232 Port Configuration

The **RS232** statement allows you to configure both serial ports independently, including the selection of echo on or off and the level of prompting. In the default **POWER-UP** program, the RS232 port parameters are set to 9600 baud, 8 data bits, 1 stop bit, no parity, echo on, error messages and prompts on. If you modify the default **POWER-UP** for your application, be sure to keep the RS232 statements set so the Model 4000 can communicate with your computer or terminal. To download programs from the Model 4000 application disk, ***the echo must be on.***

Since the Model 4000 has 2 RS232 ports, this allows the Model 4000 to be connected to another RS-232C device, such as an SX Drive or CPX Drive. If both the Model 4000 and the other device(s) are echoing characters, an endless echo loop is created which prevents the Model 4000 from operating.

If the remote device is echoing its incoming commands back, the Model 4000 will try to interpret these commands coming in the port. Since they are not legal 4000 commands (see menu below), the Model 4000 will display parsing errors. There are no hardware handshaking signals on the Model 4000's RS232 interfaces, instead, the **XON** and **X-OFF** protocol is used.

By typing **MENU** over the port the following message is displayed. Only the first 4 characters of the command need to be typed to execute that command.

To enter the X command language, first type a "/" and then a return. To exit the X command language, first type ; and then press the Return key.

Remote command list	Description
BACKLITE <#>	Choose 1 for ON, 0 for OFF.
CONTRAST <#>	Change LCD contrast, range 1-255, try 60
DIR	Display directory of programs.
DELETE <# or lbl> TO <# or lbl>	Delete lines from program
IMMEDIATE	Enter immediate execution mode Exit = ;
INSERT BEFORE <# or lbl>	Insert lines into program
LCD <#>	Display line from LCD to calling port
LOAD	Load a program, deletes program first
MEM	Remote memory size display
MENU	Display this menu
PRINT	Print program
RESUME	Resume program execution if stopped
RUN FROM <# or lbl> TO <# or lbl>	Start program execution
SELECT (Program Name)	Select a program
START	Start 4000 program
STOP	Stop motors and program
VARI Q<#> = <#>	Set variable
VARI Q<#> = ?	Display contents of variable Q100-Q109 are ASCII string variables

Again, all the commands listed in the above menu may be invoked with just the first four characters. For the sake of clarity, the following discussion will refer to the complete command name. Remote operation takes a few basic forms.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

LCD Display

Two commands, **BACKLITE** and **CONTRAST** allow remote control of the operator panel display readability. The **BACKLITE** command can be used to override the automatic timeout which protects the life of the backlite. The **CONTRAST** command could be used to allow contrast adjustment at any time, even when the **CONTRAST** software key is not present on the operator panel. A higher number gives more contrast and a lower number gives less, and the range is 1-255. The best value is found by experimentation. This would be useful in the case the contrast changes due to temperature variations while a program is running.

Statement	Description
> BACKLITE 1	Turn on the backlite
> CONTRAST 85	Choose a different contrast value

Multiple Program Manipulation

Two remote commands are devoted to the support of multiple programs. The **DIR** command will list the names of all existing programs in the Model 4000. The **SELECT** command will create the named program if it does not already exist, and will select the program if it does. The **SELECT** command specifies which program is affected by the program related commands (they are the **LOAD**, **RUN**, **START**, **PRINT**, **DELETE**, and **INSERT** commands).

Example

Statement	Description
>DIR	'Display directory of current programs
PROGØ	
PART1	
MAKEPART	
>SELECT PART1	'PART1 is now the active program name

Four remote commands are devoted to uploading, downloading, and modifying programs. The four programming functions, **LOAD**, **INSERT**, **DELETE**, and **PRINT**, can not be performed if the Model 4000 is in the **EDITOR**, in **IMMEDIATE** mode, or running a program. An attempt to do so will result in the message **INVALID DATA ENTERED**.

Downloading Programs

The **LOAD** command allows a new program to be downloaded from either remote port into the Model 4000. Before new statements are loaded, which ever program was most recently selected is first deleted. Thus a new program name should be chosen with the **SELECT** command if you no not wish to overwrite a previous program. The **LOAD** command may specify the port from which to load, or, if no specification is given, it will load from the port on which it received the **LOAD** command. The Model 4000 responds to the **LOAD** command with a special prompt, as shown in the example below if the prompts are turned on with the **RS232** statement. To exit from the **LOAD** function, type an asterisk (*) at the **LOAD** > prompt. If any of the downloaded statements contain syntax errors, and the RS-232C error function is turned on, the Model 4000 will respond with:

```
Invalid data in line:
```

and display the invalid line.

The front panel will display the line:

```
ERROR in parsing remote line.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

In this case, the flawed statement will not be stored, and the program should be corrected before it is run.

Statement	Description
>LOAD	'Load program statements in the selected program 'name
LOAD> VEL 25 25 44 44	
LOAD> LABEL JUNK	'LOAD> prompts return from the 4000 in this mode.
LOAD> MOVE 25 25 * *	'Exit the LOAD mode
LOAD> *	
>	'Standard prompt

Once the **LOAD** mode is entered, the selected program will be deleted before the new statements are loaded, if the program name exists. The Model 4000 will send a ~ (tilde) when the deletion is complete. It is very important to have the correct program selected before loading so programs are not inadvertently overwritten.

Editing Programs

The remote **INSERT** and **DELETE** have exactly the same syntax and functions as the corresponding **EDITOR** subfunctions described in chapter two. The **INSERT** command will result in a new prompt (**INSERT >**) to be used for accepting program statements. The Model 4000 continues to respond with this prompt until an asterisk (*) is given. The syntax is displayed below:

Example

Statement	Description
>INSERT BEFORE LABELØ	'Begin inserting program commands 'before a certain label
INSERT > IF BIT 7=0 GOTO LABEL2	'Program prompts reflect **INSERT** mode
INSERT > IF BIT 7=1 GOTO LABEL3	
INSERT >*	'* to exit **INSERT** mode
>	
>DELETE LABELØ THRU LABEL1	'Delete program lines starting at '**LABELØ** through **LABEL1**

If you wish to delete an entire program from memory, type the following from the remote interface.

```
> SELECT PROG10
> DELETE PROGRAM
```

The program titled **PROG10** will be removed from memory. It is also possible to erase a program by selecting it and issuing the load command without any program data.

Listing or Uploading Programs

The **PRINT** command has the same syntax and function as the operator panel **PRINT** function described in chapter two, although the port destination and line range are optional. If only the word **PRINT** is given, the entire program will be printed to the port on which the **PRINT** command was received.

Statement	Description
>SELECT PROG1	'Select program **PROG1**
>PRINT	'Print currently selected program to RS232 port
VEL 25 25 44 44	
MOVE 12 23 34 45	
*****	'*Signifies printing complete
>	'Standard prompt

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Remote Execution

It is also possible to **START**, **STOP**, **RUN**, and **RESUME** from the remote ports. **START**, **RUN**, and **STOP** functions operate just as the corresponding keys on the operator panel do, as described in chapter two. **RESUME** also performs the same function as the operator panel **RESUME** key, but will not resume a stopped program if any of the programming commands **LOAD**, **INSERT**, **DELETE**, or **PRINT** have been entered after **STOP**.

START begins the currently selected program at the beginning of the program. **RUN** allows program execution from certain line numbers or labels. On the standard Model 4000, it is not possible to issue a **RUN** or **START** command during program execution.

Misc.

IMMEDIATE allows program statements to be executed immediately. Once **IMMEDIATE** mode is entered, no editing functions (i.e., **LOAD**, **PRINT**, etc.) can be used. Similarly, the **IMMEDIATE** function cannot be used while in the editor or while programs are running.

The **IMMED>** prompt is returned from the Model 4000 when this mode is enabled. Most commands can be executed in **IMMEDIATE** mode, however, since no program is running, no commands dealing with program labels are applicable (i.e., **ON**, **IF**, **GOSUB**, **GOTO**, etc.). When in **IMMEDIATE** mode, only LCD lines 1 and 3 are available for display purposes. Lines 2 and 4 have special functions consistent with **IMMED** mode via the front panel. **IMMEDIATE** mode can be exited by entering a ; at the **IMMED>** prompt.

See the example below for information on using the remote **IMMED** mode:

```
Statement                      Description
>IMMED                        'Enter Immediate mode
IMMED> ACCEL 25000 * * *      'Set Axis 1 acceleration
IMMED> VEL 25000 * * *        'Set Axis 1 velocity
IMMED> MOVE 10000 * * *       'Move axis 1 100000 counts
IMMED> OUT BIT12 = 1          'Turn on output bit #12
IMMED>;                       'Exit Immediate mode
>                             'Standard prompt
```

Most statements execute very quickly in the Model 4000 and can be sent down quickly in **IMMEDIATE** mode. However, if a statement takes several seconds to execute and complete, the next statement in the buffer will not execute until the previous statement is done. This is true of the **MOVE**, **WAIT**, and some **IN** statements.

The two commands **LCD** and **VARIABLE** allow remote interaction with the Model 4000 at any time, even during program execution. The only times these commands are not allowed are while in **IMMED** mode, or when in the process of using editor commands (**INSERT**, **LOAD**, etc.).

The **LCD** command will export the current contents of the **LCD** line number specified. This can be useful as a remote view of the operator panel, or as a way to check what is being displayed during program execution. The LCD remote command can be used even if there is no front panel connected.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To simulate a continuous position report over RS-232, turn on a position **DISPLAY** on the front panel and continually send the **LCD** remote command.

The **MEM** statement reports back the size of the user memory. It reports the total memory size, not the available memory. Either 65,535 or 8,191 bytes will be reported depending upon options installed. (It is possible to get a report of free memory available after a load. Immediately after the load is complete and no errors have been reported issue an **LCD 3** command to see the free memory display from the LCD screen.)

The **VARIABLE** command allows the remote assignment and report of the contents of any **Q** variable, including string variables typically, **Q100 - Q109**. The following example illustrates the use of these commands.

Statement	Description
`>LCD 1`	'Display text currently on LCD line 1
`  COMPUMOTOR Model 4000 MOTION CONTROLLER`	
`>VARIABLE Q10 = 45.5`	'Set Q10 to 45.5
`>VARIABLE Q10 = ?`	'Poll current value of Q10
`  45.5`	
`>VARIABLE Q100 = COFFEE TIME`	'Set string variable Q100
`>VARIABLE Q100 = ?`	'Poll current contents of Q100
`COFFEE TIME`	
`>`	'Standard prompt for next command

In later Model 4000 software revisions the **VARIABLE** command has been shortened considerably:

```
VARI Q1=1 → Q1 1
VARI Q1=? → Q1
```

This new simplified syntax is available in software revision C6 and later for the 10MHz controllers, and revision A and later for the 12MHz controllers.

The final remote command is the **RESET** command. **RESET** deletes all user programs and restores the default **POWER-UP** program.

Application Developer Software

An IBM-compatible application software diskette is provided with each Model 4000. You can use this software for the following tasks.

- Setting up IBM/IBM-compatible as a dumb terminal for communicating over RS-232
- Creating and Editing 4000 programs
- Saving and recalling programs to disk
- Uploading and downloading programs to Model 4000
- A list of all statements, Functions, and Commands
- Programming Examples

The **HELP** key (**F1**) should be used generously, until you become proficient with the use of the application developer. Use help to:

- Display command definitions & syntax
- File storage (imports and exports)

This software is a user-friendly, menu-driven program that guides you through all the options. To start the program, type **4ØØØ** from your IBM DOS Operating System. Use the arrow keys to move to the appropriate menus.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Common Problem Symptoms

- ❒ No program present in Model 4000 with user's intended name
- ❒ User's program is present in Model 4000, but under **POWER_UP** program
- ❒ When program in software editor, download attempted but unsuccessful (i.e. no "% completed" message on screen)
- ❒ No echo from Model 4000 when in Terminal Emulator
- ❒ **POWER_UP** program (or user's program) is empty inside Model 4000
- ❒ **LOAD>** prompt not present after downloading
- ❒ No error messages during download, but program lines missing
- ❒ Downloading a program takes a **very** long time

Default RS-232 Parameters and POWER_UP Program

All the symptoms above, and many other oddities during communication with the Model 4000 can be explained by knowing more about the default **POWER_UP** program present in the unit at the time of purchase.

Besides setting default scale factors and numbers of variables, etc., a very important task of the default **POWER_UP** program is to set the communication parameters. If you look at the program in the Model 4000's editor or upload it to the PC, search for the following two statements:

```
RS232 PORT1 = 9600 8 OFF 1 ON ALL
RS232 PORT2 = 9600 8 OFF 1 ON ALL
```

These statements set the communication parameters as follows: 9600 baud, 8 data bits, no parity, 1 stop bit, echo on, and error messages/prompts on.

It is very important to have the default RS-232 parameters present and executed in the Model 4000 before attempting to download. If the parameters reside in the unit's **POWER_UP** program, the user is ensured of successful download. If the **POWER_UP** program needs to be modified for a user's application, make sure those statements exist in the modified version.

If the echo is turned off, the Model 4000 editor will not be able to successfully and quickly download programs. If a download is attempted when the echo is off, it will appear that the 4000 just *hangs up* when the F3 and **ENTER** keys is pressed to start the process.

Error Messages and Prompts

With the error messages and prompts turned on, you will see the prompt **LOAD >** next to each line of the program successfully loaded. If there is a syntactical problem with a statement, extra carriage returns and an error message will appear at the offending statement. To see all errors, simply page down in the editor to view the entire program. When all errors are noted, just recall the original program into the editor, fix the errors and download again. If the error messages are turned off, the Model 4000 will not be able to let the user know that there was a syntax problem with a particular statement they tried to download, and the statement will not be present in the user's 4000 program.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

It is important not to attempt to download a program with the **LOAD >** prompts in the editor screen. Remote commands such as **LOAD**, **SELECT**, **START**, etc. should also be absent from the editor's contents. The editor should only contain the actual program statements, when the F3 key is pressed to start the download, the program name is given, etc. to get the program set up correctly in the Model 4000. If any remote commands or prompts are present when a download is attempted, error messages will result.

Selecting the Program Name (4000 Application Support Software Version 1.0 and 1.1)

Using the Model 4000 Communications software correctly when attempting to download programs will ensure that the **POWER_UP** program (and other program names) only contain what the user intends. When the user presses F3 to download while the program is called up into the editor, they will see a pull-up menu with the following commands:

```
SELECT   PROG0
LOAD
```

PROG0 is just a default name the user may want to use. If they want to call the program **TEST**, they should use the arrow keys to simply change the program name, but **leave the command SELECT in place**:

```
SELECT   TEST
LOAD
```

After typing the new name, press **<enter>**, the cursor will go down to the next line, and press **<enter>** again to begin loading. The menu to upload a program is similar.

If the user deletes the **SELECT** command, the Model 4000 can not recognize the new intended name, and the program downloaded will reside under the last active program name (usually the **POWER_UP** program). The program is there, it just has the wrong name and the old program of that name is deleted.

Getting Back to Known Parameters (How to Get to Square One)

No matter how it happened, when the **POWER_UP** program has been deleted or corrupted, the user simply needs to reset the Model 4000 to retrieve the default **POWER_UP** program. Resetting the unit does delete all other user programs, however, so it should be used somewhat cautiously. Information on how to reset the unit can be found in Chapter 7, *Maintenance and Troubleshooting*. After the unit has been reset, the downloading and uploading procedures will operate as expected if the RS-232 wiring is correct and the tips above are followed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

☛If running both the X language and a Model 4000 program at the same time, entering any characters or spaces on the computer keyboard will step through the program you are running and may cause motion.

In addition to the operations described above, it is possible to interact with the individual axes of the Model 4000 via the X command language. The commands described above are useful for working with Model 4000 statement programs, which coordinate the motions and actions of multiple axes. The X command language is useful for interactive operation of the individual axes in an independent manner. *It is possible, but not advisable,* to operate the X command language while a Model 4000 statement program is running. The X command language is intended to allow use of the four axes as though they were four individual controllers tied through a daisy chained serial link. If no axis specifier is given to an X command, that command will go to axis one. All other axes require an axis specifier. There are no global commands. Access to the X command language is gained by typing a / and carriage return at the Model 4000 prompt on either port. Either port may be used, which makes it possible to have both ports in X command language mode at the same time. In this case, all responses to X command requests will be sent to Port 2, regardless of which port originated the request. After the slash is entered, the Model 4000 stops sending prompts, and merely echoes the received characters. X commands must be all capital letters and are delimited with a space or carriage return. To return to the standard Model 4000 prompt, type a ; and a carriage return. For a complete description of the X command language, refer to the **X Series Command Language Reference Guide**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

C H A P T E R ⑥

Hardware Reference

Chapter Objectives

The information in this chapter will enable you to:

- ❑ Use this chapter as a quick-reference tool for most system specifications (performance, I/O, pin-outs, dimensions, etc.)

Performance

Stepping Accuracy : ±0 steps from preset total
Velocity Accuracy : ±0.02% of maximum rate
Velocity Repeatability : ±0.02% of set rate
Contouring Deviation from an arc, due to straight line approximation to a curve.

$$\text{Error in steps} = \left[\frac{[V_p(.001\text{sec})^2]}{2r}\right]$$

V_p is in steps/sec

r is radius in steps

Environmental

Operating Temperature: 36°F to 122°F (5°C to 50°C)
Storage Temperature: -22°F to 185°F (-30°C to 85°C)
Humidity: 0% to 95% non-condensing

Inputs

The following section contains specifications for the Model 4000's inputs.

Power

95 - 240VAC, 50/60Hz, 1.5A @120VAC, 0.8A @220VAC

RS-232C

A three-wire implementation (TX, RX, GND) is used. Compumotor recommends that you use a shielded cable (a shield terminal is provided).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Home, CW, and CCW Limits

These inputs are TTL-compatible[1] and have internal 22kΩ pull-ups to 5V. The voltage range is ØV - 24V.

Triggers

These inputs are TTL-compatible and have internal 22kΩ pull-ups to 5V. The voltage range is ØV - 24V.

Test

This input is TTL-compatible and has an internal 22kΩ pull-up to 5V. The voltage range is ØV - 24V. When grounded during power-up, this input sets the 4000 into a self-test mode. ***This mode will cause the axes to move.***

Estop

This input is TTL-compatible and has internal 22kΩ pull-up to 5V. The voltage range is ØV - 24V. You must set this input low (0V) to enable motion. This input is intended to be used as an emergency stop input. The motion cut-off is a function of hardware and does not require microprocessor intervention. If you use this input, the 4000 may lose track of the motor's position (encoder position will still be accurate).

Encoder

The differential comparator accepts two-phase quadrature incremental encoders with differential or single-ended outputs (+5V TTL-compatible). The maximum frequency is 100 kHz. For higher frequencies, consult the factory.

Encoder Power

This pin on the 14-pin AUX1 connector goes directly to the 25-pin D encoder connector. This pin may be connected to the adjacent 5V pin on the AUX1 connector, or to a user-supplied power supply.

Programmable Inputs

The Model 4000 has 24 programmable inputs. These inputs are optically isolated. They are TTL-compatible and have internal 22kΩ pull-ups to 5V. The voltage range is ØV - 24V. The 50-pin plug for these inputs is compatible with OPTO-22 standard signal conditioning equipment. You can control these inputs with the Model 4000's programming language.

Drive Fault

These inputs have TTL-compatible voltage levels and internal 3.3KΩ pull-ups to 5V. The voltage range is 0V to 5V.

Pull-Ups

These inputs on the AUX1 connector allow the user to control the *high* output voltage level on the axis outputs and 24 programmable outputs. Connecting these inputs to the adjacent +5V pin makes the outputs TTL compatible. Connection to other voltages (max 24V) allows for compatibility with other signal levels.

---

[1] TTL compatible voltage levels: Low ≤ 0.4V, High ≥ 2.4V

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Joystick Release

This input it TTL compatible and has an internal 22K pull-up to 5V. The voltage range is 0V - 24V. A high on this input indicates that program execution may continue (after a **JOYSTICK ENABLE** statement).

Joystick Select

This input it TTL compatible and has an internal 22K pull-up to 5V. The voltage range is 0V - 24V. This input selects between two modes of joystick control as defined in software (**DEFINE JOYSL, JOYSH** commands).

Joystick Trigger

This input is CMOS compatible and has an internal 22K pull-up to 5V. The voltage range is 0V - 24V. This input may be used to control program flow such as using an IF statement that checks the state of the input.

Joystick Velocity Range

This input is CMOS compatible and has an internal 22K pull-up to 5V. The voltage range is 0V - 24V. This input may be used to select either the high or low velocity range as defined in the software (**VEL JOYL, JOYH** commands).

Analog Inputs

The analog inputs have a voltage range of 0V (min. scale) to 2.5V (full scale). The input voltage must not exceed 3V.

Outputs

The following section contains specifications for the Model 4000's outputs.

+5VDC

The 4000 provides +5VDC at up to 4 amps for the user. The 4 amp limit applies to the total load on all of the I/O connectors. For example, if the encoder power pin is jumpered to +5VDC and the encoders use a total of 2 amps then 2 amps is left for other uses.

RS-232C

A three-wire implementation (TX, RX, GND) is used. Compumotor recommends that you use a shielded wire (a shield terminal is provided).

Step, Direction, and Shutdown

These outputs come from differential line drivers although they may be used single ended. A signal high is 3.0VDC minimum at +30mA. A signal low is 1.0VDC minimum at -30mA. The **+output** for each differential driver is active high. The **-output** for each driver is active low. The step pulse width is 0.5 ms to 1,000ms (depending on velocity range).

Programmable Outputs

The Model 4000 has 24 programmable outputs. These outputs are optically isolated. They have TTL-compatible voltage levels and internal pull-up to 5V (supplied) or user supplied voltage up to 24V. You must use an external diode when driving inductive loads (connect the anode to ground and the cathode to the output). The maximum voltage is 24V[1]. The 50-pin plug for these outputs is compatible with OPTO-22 standard signal conditioning equipment. You can control these outputs with the Model 4000's programming language. Outputs are capable of sinking current up to 30 mA.

[1]This is the maximum voltage that may be applied to the output when it is in the OFF state (not sinking current).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Axis-Programmable Outputs

The Model 4000 has 4 axis programmable outputs. These outputs are optically isolated. They have TTL-compatible voltage levels and internal pull-ups to 5V (supplied) or user supplied voltage up to 24V. You must use an external diode when driving inductive loads (connect the anode to ground and the cathode to the output).

CW & CCW Step

Open collector outputs can sink up to 250mA. Maximum voltage is 24V.

Fault

This is a dry contact relay. The maximum current is 5A at 115V AC, 2.5 A at 220V AC (24VDC), 250mA minimum at 120V, 125mA minimum at 220V. Normally open output is closed when the 4000 is powered and no fault condition exists. Normally closed output is closed when the 4000 is not powered or when a fault condition exists. A fault condition exists if the ESTOP input is active or the Board Monitor Alarm (BMA) is active.

Board Monitor Alarm

The Board Monitor Alarm detects un-recoverable faults in hardware and software. When the BMA detects a fault, the status light on the Model 4000 and on the Front Panel will turn red and the Fault output will indicate a fault. The BMA can be reset only by cycling power to the Model 4000.

Pin-Outs

The section contains pin-outs for the following Model 4000 components:

Encoder Connectors	25-pin D per axis
Motor/Driver Connectors	15-pin D per axis
Axis I/O	7-pin screw terminal per axis
Auxiliary 1	14-pin screw terminal
Auxiliary 2	14-pin screw terminal
Programmable I/O Connectors	50-pin flat cable headers
Fault Output Connector	5-pin screw terminal

Motor/Driver Connectors

The following table contains the pin outs for the Model 4000's four 15-pin D Motor/Driver connectors.

Pin	Input/Output	Description	Color
1	OUT	Step (+)	Red
2	OUT	Direction (+)	Green
3	—	Ground	—
4	—	Reserved	Grey
5	IN	Fault	Yellow
6	OUT	CW Step	—
7	OUT	CCW Step	—
8	—	Shield	(Bare Wire)
9	IN	Step Return (-)	Black
10	IN	Direction Return (-)	White
11	OUT	Shutdown (+)	Blue
12	IN	Shutdown Return (-)	Purple
13	—	Ground	Orange
14	—	Ground	Brown
15	—	Ground	—

Compumotor supplies two indexer/drive cables. If you need more than two cables, contact your local distributor or ATC (P/N 71-010432-10).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Encoder Connectors

The following table contains the pin outs for the Model 4000's four 25-pin D Encoder connectors. Pins #1 - #7 address the incremental encoder interface while pins #8 - #25 address the absolute encoder interface.

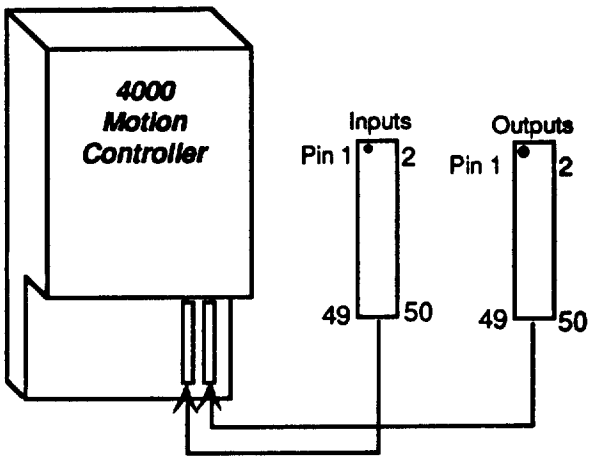
Pin	In/Out	Description
1	IN	A Channel
2	IN	A Channel Return
3	IN	B Channel
4	IN	B Channel Return
5	IN	Z Channel
6	IN	Z Channel Return
7	—	NC
8	OUT	Shield
9	OUT	Data enable low
10	OUT	Data request low
11	OUT	Address zero
12	OUT	Address one
13	IN	Data valid one
14	—	Ground
15	IN	Data zero
16	IN	Data one
17	IN	Data two
18	IN	Data three
19	IN	Data four
20	IN	Data five
21	IN	Data six
22	IN	Data seven
23	OUT	V+ Encoder Power Out (from AUX 1 connector pin #10)
24	OUT	Transmit
25	IN	Receive

Programmable I/O Connectors

The following table contains the pin outs for the Model 4000's two 50-pin I/O connectors. ***All even-numbered pins are connected to logic ground.***

Pin	Output Connector	Input Connector
49	+5VDC	+5VDC
47	Output #1 (LSB)	Input #1 (LSB)
45	Output #2	Input #2
43	Output #3	Input #3
41	Output #4	Input #4
39	Output #5	Input #5
37	Output #6	Input #6
35	Output #7	Input #7
33	Output #8	Input #8
31	Output #9	Input #9
29	Output #10	Input #10
27	Output #11	Input #11
25	Output #12	Input #12
23	Output #13	Input #13
21	Output #14	Input #14
19	Output #15	Input #15
17	Output #16	Input #16
15	Output #17	Input #17
13	Output #18	Input #18
11	Output #19	Input #19
09	Output #20	Input #20
07	Output #21	Input #21
05	Output #22	Input #22
03	Output #23	Input #23
01	Output #24 (MSB)	Input #24 (MSB)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Auxiliary 1

This table contains the pin outs for the Model 4000's Auxiliary 1 (AUX1) 14-pin screw terminal.

Pin	In/Out	Description
1	IN	RS-232C Receive Port 1
2	OUT	RS-232C Transmit Port 1
3	—	Ground
4	—	Shield
5	IN	RS-232C Receive Port 2
6	OUT	RS-232C Transmit Port 2
7	—	Ground
8	OUT	Shield
9	OUT	+5VDC
10	IN	V+ Encoder Power In
11	OUT	+5VDC
12	IN	Programmable Output Pull-ups
13	OUT	+5VDC
14	IN	Axis Output Pull-ups

Auxiliary 2

The following table contains the pin outs for the Model 4000's Auxiliary 2 (AUX2) 14-pin screw terminal.

Pin	In/Out	Description
1	OUT	+5VDC
2	IN	Analog 1
3	IN	Analog 2
4	IN	Analog 3
5	IN	Analog 4
6	—	Ground
7	—	Shield
8	IN	Joystick release
9	IN	Joystick select
10	IN	Test
11	—	Reserved
12	IN	Estop
13	IN	Joystick trigger
14	IN	Joystick velocity range

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Axis I/O

This table contains the pin-outs for the Model 4000's four Axis I/O 7-pin screw terminals.

Pin	In/Out	Description
1	IN	Home Enable
2	IN	CW Limit
3	IN	CCW Limit
4	IN	Trigger
5	OUT	Axis Output
6	—	Ground
7	—	Shield

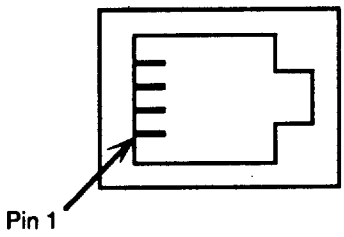
Fault Output Connector

The following table contains the pin outs for the Model 4000's Fault Output 5-pin screw terminal.

Pin	In/Out	Description
1	N.C.	Normally Closed Contact
2	—	Not Connected
3	N.O.	Normally Open Contact *
4	—	Not Connected
5	Common	Common

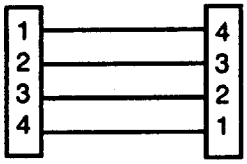
The normally open contact will close when the Model 4000 is powered with no faults.

Operator Interface Connector (4-pin RJ-11 Jack)



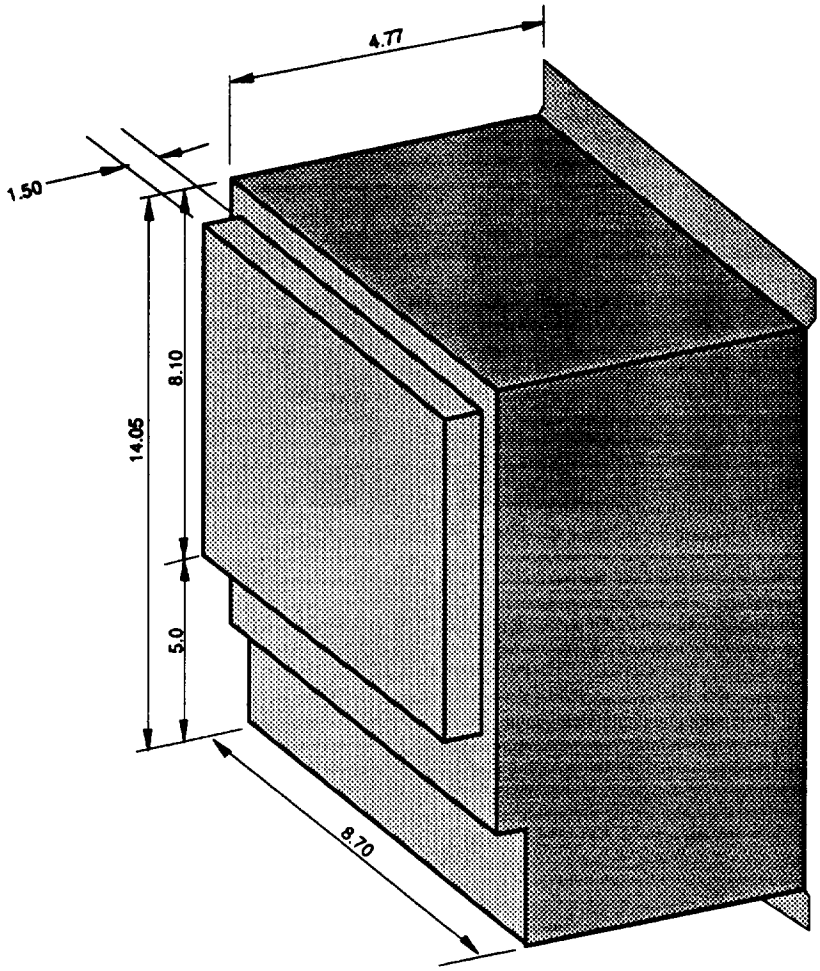
Modular Jacks on Main Unit			Modular Jack on Operator Panel		
Pin	In/Out	Description	Pin	In/Out	Description
1	OUT	+12	1	--	Ground
2	IN	Receive	2	IN	Receive
3	OUT	Transmit	3	OUT	Transmit
4	--	Ground	4	IN	+12

Standard phone cables have the correct pin connections to connect the Model 4000 to the operator panel. If making your own cable use the following diagram.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

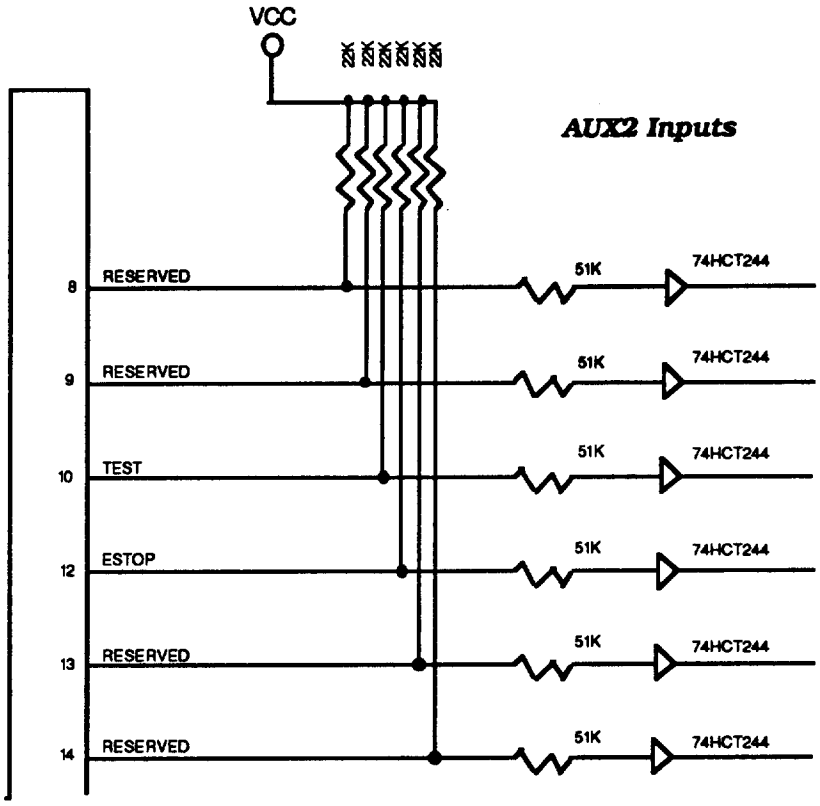
Model 4000 Dimensions



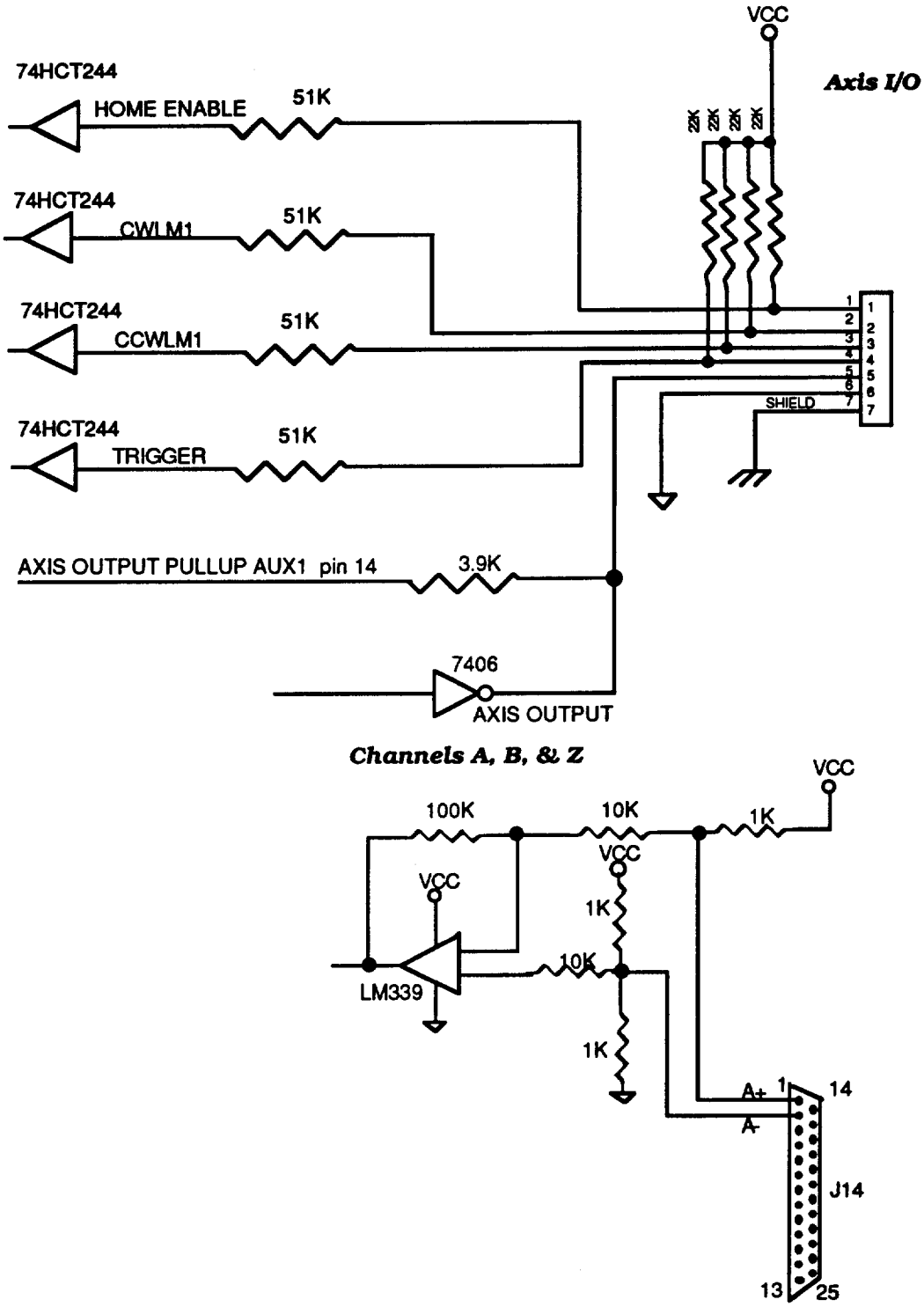
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Model 4000 Schematics

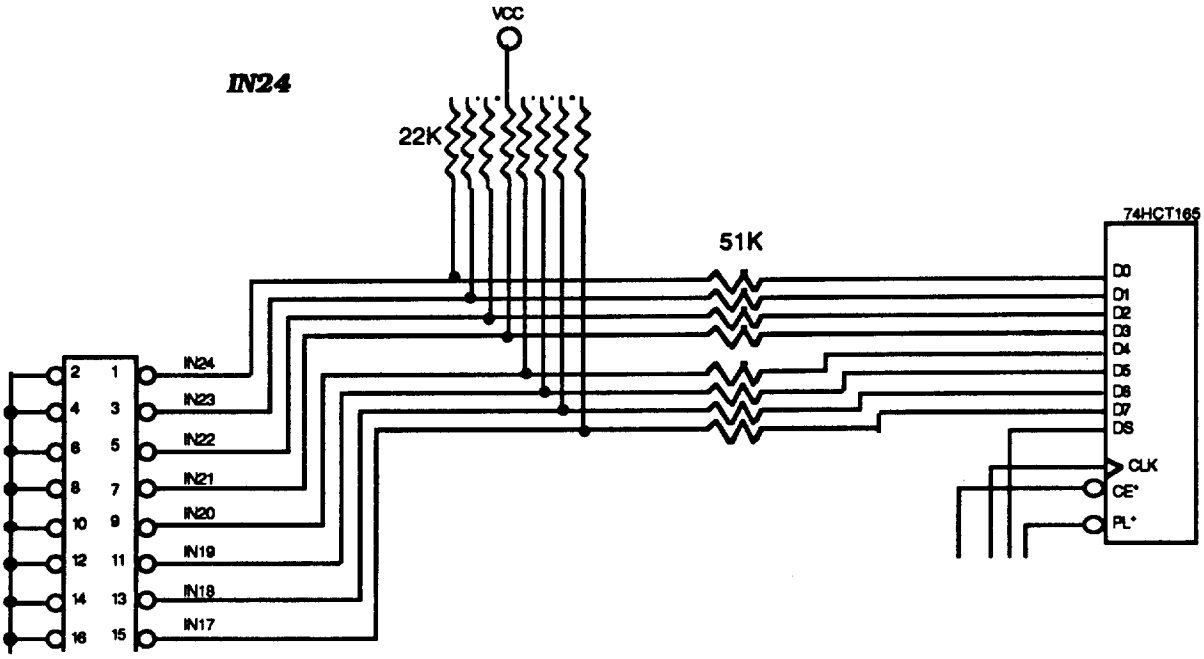
☛The following diagrams show components internal to the Model 4000



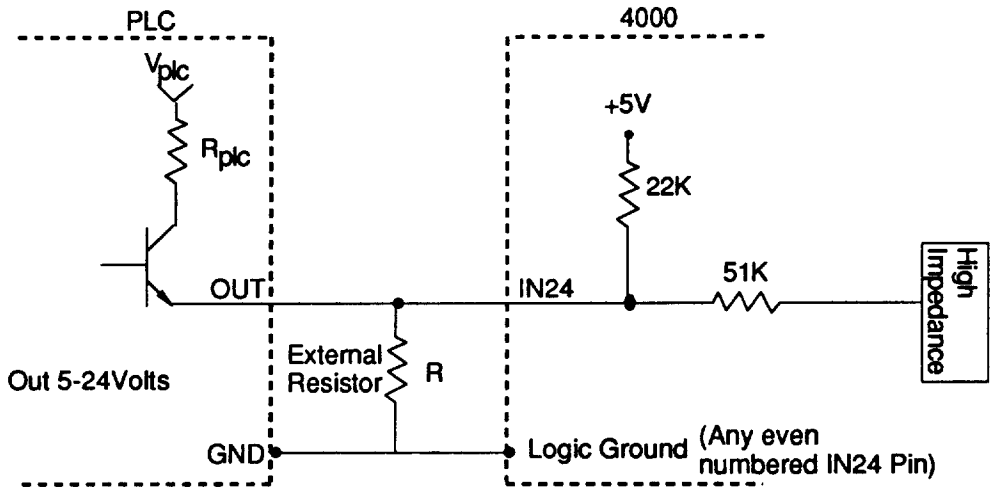
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



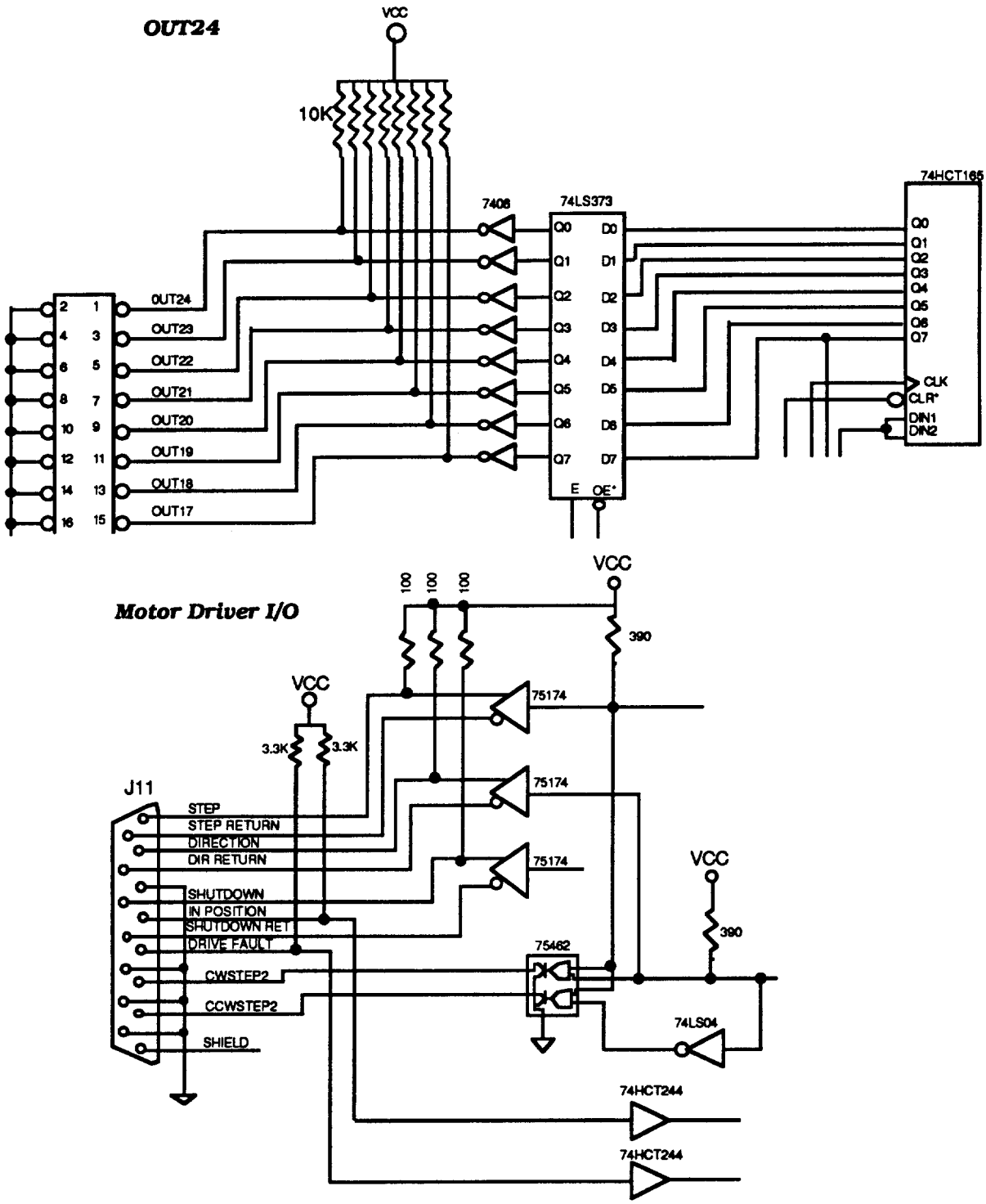
IN24 to PLC Sourcing Outputs



Typical value for R = 450Ω (assuming R_{plc} = 0)

Note: The value of R may vary depending on the value of Rplc and Vplc

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

C H A P T E R ⑦

Maintenance & Troubleshooting

Chapter Objectives

The information in this chapter will enable you to:

- ❑ Maintain the system's components to ensure smooth, efficient operation
- ❑ Isolate and resolve system hardware problems
- ❑ Isolate and resolve system software problems

Maintenance

The only periodic maintenance requirement of the Model 4000 is keeping the fan filter clean. This is required to allow the fan to provide adequate cooling of the internal circuitry.

☛ Do not operate the Model 4000 without the fan filter installed. Contaminants may be drawn inside and damage the unit.

To clean the filter:

① Gently pull off the plastic cover.

② Take the filter and blow out the dust with pressurized air. ***Do not use water.***

③ Replace the filter and plastic cover.

④ Check the filter often and re-clean if necessary.

Spare Parts Table

This table contains parts that you may order from Compumotor. These parts replace parts that were included in the original ship kit or are special accessories and options.

Replacement Part	Description
71-009086-01	8" Phone Cord
71-007683-03	3' Phone Cord
95-010719-01	Applications Software Diskette
44-000054-01	AC Power Cord
71-010432-10	15D - 25D Cables
58-009135-01	Gasket
43-010262-01	Phoenix Connector
58-010395-01	Fan Kit
92-010835-01	PROM (Front Panel)
47-007709-01	Panasonic BR2032 3V Battery (older 10 MHz Units)
47-011743-01	Panasonic BR2450 (Newer 12 MHz units, Qty 2)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Firmware Reference Table

10 MHz Main Boards

If your unit has a serial number less than 91-1114XXXX you must use this table.

IC #	Non-Contouring	Contouring	Memory Expansion
U2	92-010871-01	92-010871-01	
U4	92-010871-11	92-010871-11	
U14		32-011057-01	
U15		32-011057-11	
U58			32-011057-01
U59			32-011057-11
U62	92-010872-01	92-011619-01	
U63	92-010872-11	92-011619-11	

12 MHz Main Boards

If your unit has a serial number greater than or equal to 91-1114XXXX you must use the following table. These units were manufactured after November 14, 1991.

IC #	Non-Contouring	Contouring	Memory Expansion	Following
U2	92-012117-01	92-012117-01		92-013550-01
U4	92-012117-11	92-012117-11		92-013550-11
U14		32-012051-01		32-012051-01
U15		32-012051-11		32-012051-11
U58			32-012051-01	32-012051-01
U59			32-012051-11	32-012051-11
U62	92-012118-01	92-012119-01		92-013083-01
U63	92-012118-11	92-012119-11		92-013083-11

Optional Features Table

Part Number	Upgrade Kit	Option Available
-M	MEM4000*	Expanded Memory
-G	4884000	IEEE-488
-C	CIR4000*	Contouring Option
-CFM	FOL4000-02	Multi-Axis Following
AR-C		Absolute Encoder

*For units with serial numbers after 911114ØXXXX, order upgrade MEM4000-02 and/or CIR4000-02. The Following option can only be used in units with serial numbers 911114XXXX and higher.

Troubleshooting

When your system does not function properly (or as you expect it to operate), the first thing that you must do is identify and isolate the problem. When you have accomplished this, you can effectively begin to resolve the problem.

The first step is to isolate each system component and ensure that each component functions properly when it is run independently. You may have to dismantle your system and put it back together piece by piece to detect the problem. If you have additional units available, you may want to exchange them with existing components in your system to help identify the source of the problem. It may be helpful to swap drive units in the event that some axes move and some don't.

Determine if the problem is mechanical, electrical, or software-related. Can you repeat or re-create the problem? Do not attempt to make quick rationalizations about problems. Random events may appear to be related, but they are not necessarily contributing factors to your problem. You must carefully investigate and decipher the events that occur before the subsequent system problem.

You may be experiencing more than one problem. You must isolate and solve one problem at a time. Log (document) all testing and problem isolation procedures. You may need to review and consult these notes later. This will also prevent you from duplicating your testing efforts.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If your programs are backed up on your computer or floppy disk it may be useful to perform a software reset (reset the system). Download the software and record any compilation errors.

If you are having difficulty isolating a problem be sure to document all occurrences of the problem along with as much specific information, such as time of occurrence, 4000 status, and anything else that was happening when the problem occurred.

Once you have isolated a problem, take the necessary steps to resolve it. Refer to the problem solutions contained in this chapter. If your system's problem persist, contact Parker Compumotor's Applications Department at (800) 358-9070.

Basic Troubleshooting Procedure

The following questions should be answered when first attempting to troubleshoot the Model 4000.

① Is the Model 4000 Status LED on and **green**? If it is off, the Model 4000 is not receiving the proper AC input voltage. Check the power cord to make sure it is installed correctly. Measure your input voltage to make sure its between 95VAC - 240VAC. If all voltages are correct and the light is off, the internal power supply fuse could be blown or the power cord could be bad. Try another EIA molded power cord. Contact Compumotor Applications Engineering Department (800-358-9070) if trouble persists.

☛ **Verify that the test input is not grounded.**

If Status LED is on and **red**, the internal microprocessors have faulted. Document what was happening when it faulted (it may be helpful when you attempt to isolate the cause). Cycle power to the Model 4000. If the LED is still RED, there may be an internal failure in the unit. If the LED is green, the fault may have been caused by electrical noise. Check to make sure all PROMS in the unit are seated properly.

② Is the Operator Panel Status LED on and **green**? If it is off the Operator Panel is not receiving power from the Model 4000. Check to be sure the Model 4000 has power (see step 1, above). Check the phone cord connections to be certain they are plugged in and properly seated. If they are and the LED stays off, try another phone cord. If still off, contact Compumotor Applications Engineering Department

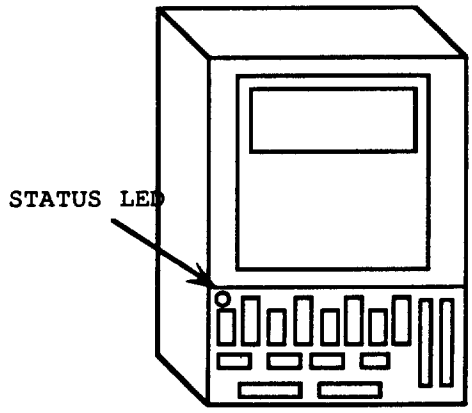
If the Operator Panel Status LED is on and **red** (and the 4000 Status LED is green), the microprocessor in the Model 4000 operator panel is not working properly. Cycle power to the Model 4000. If the LED went from **red** to **green** after you cycled power, you may have an electrical noise problem.

Diagnostic LEDs

The Model 4000 has several LED indicators that can assist your troubleshooting efforts.

Status LED

☛ The status LED on the Model 4000 Main Unit indicates that the internal microprocessors are functioning properly. This bi-color LED will be *green, red, or off.*



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Green	The LED will be green if power is applied and the Model 4000 is functioning properly.
Red	The LED will be red if the watchdog timer circuitry has not been reset, indicating a microprocessor fault. It is very common for the status light to be red if a user has installed new PROMS incorrectly (i.e., backwards, wrong slots, bent pin, etc.). Unless the PROMS were installed backward, correct the problem and repower the Model 4000.
Off	The LED will be off if power is not applied to the Model 4000.

Operator Panel

Status LED

The operator panel status LED indicates that the operator panel is functioning properly. This bi-color LED will be:

Green	The LED will be green if power is applied and the microprocessor inside the operator panel is operating properly
Red	The LED will be red if the operator panel microprocessor is not executing properly or if the Model 4000 microprocessor has failed. This LED also blinks RED and back to green every time a key is pressed, giving you an indication that the operator panel is recognizing the keystrokes. If the front panel status light is red, but the main unit status light is green, there is very often a problem with the phone cord or the phone jack connectors. Check to make sure none of the pins in the cable or socket are touching each other.
Off	The LED will be off if no power is applied to the operator panel and/or the Model 4000.

JOG Mode LED

This LED will be on (orange) when the Model 4000 is in the **JOG MODE** and will be off if in the normal operating mode. Programs will not run when in **JOG MODE** from the front panel. However, it is possible to enable **JOG** mode under program control. (**MENU MODE** key is used to return from **JOG MODE**.)

Axis LEDs

Each axis (four) has an Active LED and a Fault LED.

Operator Panel Active LED
The Active LED is ON (green) when the motor is being commanded to move. The Active LED is OFF when the axis is not commanded to move.

Operator Panel Fault LED

The Fault LED will be off unless one of the following conditions exist:

① An end-of-travel limit switch input is active (default)

② The **ESTOP** input is not grounded

③ The axis has been disabled via the **SHTDN DRIVE** statement or the remote Shutdown (**ST**) command.

④ Software limit positions have been exceeded.

⑤ A motor stall has been detected (stall detect enabled with the **ENABLE STALL** statement)

Reducing Electrical Noise

For detailed information on reducing electrical noise in your system, refer to the current Compumotor Catalog.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Model 4000 Power Supplies

The Model 4000 has two internal power supplies, a 12VDC and a 5VDC supply. The 5VDC suppy is the reference for all digital logic circuits as well as the pullup voltage supply for the encoders and outputs. The 12VDC supply provides power to the front panel and RS-232C. In the event that communications or front panel operations are faulty, but the main status light is green, there may be a problem with the internal 12VDC supply.

Common Problems & Solutions

The following table presents some guidelines to help you isolate problems with your motion control system. Some common symptoms are listed along with a list of possible causes and remedies.

- ❑ Look for the symptom that most closely resembles what you are experiencing.
- ❑ Look through the list of possible causes so that you better understand what may be preventing proper operation.
- ❑ Start from the top of the list of remedies and use the suggested procedures to try and isolate the problem.
- ❑ Refer to other sections of the manual for more information on wiring, software commands, etc..

A **TEST** program is available to test the functionality of many system components (*Chapter 2, Getting Started*). If you use the **TEST** program make sure you save a copy of your program on an external disk. The **TEST** program will erase your Model 4000 program.

Symptom	Cause	Remedy
Motor(s) do not move *Axis fault LED ON*	Hardware over-travel limit not grounded and enabled through software	Use the DISPLAY function to determine what has caused the fault
	ESTOP input not grounded (all 4 fault LEDs ON)	Check wiring of over-travel and ESTOP inputs
	Software Over-travel limit exceeded	Use ENABLE SLIMIT NO as an immediate command and attempt to jog motors
	The motor has stalled	
	The SHTDN DRIVE statement or **ST** command has been issued.	
Motor(s) do not move *Axis fault LED OFF, Active LED goes ON*	Motor/drive cable not connected properly	Check Model 4000 to Motor/drive wiring
	Motor/drive not functioning properly	Refer to motor/drive manual to verify operation
Motor(s) do not move *Axis fault LED off, Active LED stays OFF*	Model 4000 in wrong mode; i.e. must be in JOG MODE to jog, MENU MODE to run program	Check JOG MODE LED, try jogging the motor.
	Program not functioning as expected.	Use Display Trace function to help debug program
Operator Panel not working *Panel status LED green*	Contrast set at lowest or highest level	Change Contrast (F6)
Operator Panel not working *Panel status LED red or off*	Microprocessor or memory IC's loose	See step two of Basic Troubleshooting Procedure
	Phone Cable damaged	
	4000 Status LED red	
Programmable Outputs not working *Using OPTO-22 Equipment*	Jumper on OPTO board not installed	Make sure jumper from pin 49 to V+ on OPTO-22 board is installed.
	Cable from 4000 to OPTO-22 board not connected properly	Issue Immed DISPLAY IN24 or OUT24 command to check status
	Damaged OPTO-22 equipment	Check connections, swap cables
		Swap working output module with the non-working one.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

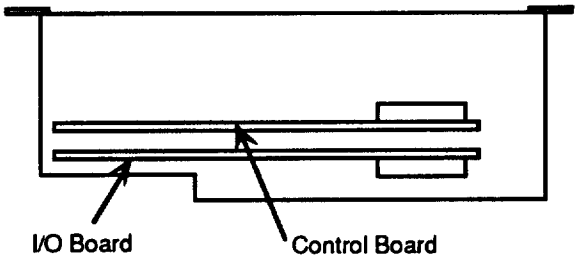
Outputs not working	Faulty Connection	Check connections
Direct connection to programmable or axis output	Program Logic incorrect	Program Logic changed with `DEFINE OUT24`
	Pullups not connected	Measure input voltage (+5VDC high = logic "0", 0.1VDC low = logic "1")
Cannot Access Model 4000	The default ACCESS code is 4000	Enter 8609853 as the ACCESS code
	ACCESS code has been changed	Use PASSWD function to change code
	You forgot the ACCESS code you had entered	Perform a hardware refer on the unit.
Cannot save Model 4000 program	Memory should be expanded to 56K	See MEM function in Remote Operation section of Chapter 5, Application Design
No RS-232C Communication	Faulty connection	Refer to RS232 Troubleshooting Tips
	Wrong communication Port	Try shifting ports
Garbled RS-232C Communications	Faulty connection	Refer to RS232 Troubleshooting Tips
	Wrong communications parameter	Try shifting ports

Closed-Loop Related Problems

Symptom	Cause	Remedy
Motor doesn't position accurately	Noise causing missing or extra encoder counts	Check coupling
	Loose coupling on encoder	Check cables
		Use DISPLAY E_ABS and make open loop moves to check encoder count & direction
Motor stops or stalls or creeps at low speed	Stall detect enabled	Check encoder connections
	Encoder improperly connected	Check cables
	Channel A & B swapped	Use DISPLAY E_ABS and make open loop moves to check encoder count & direction
	Mechanical backlash greater than programmed amount	Check calculations for encoder resolution
	Encoder Resolution specified improperly	

Resetting the Model 4000

After installing new software PROMS in the Model 4000 or in the event of a system error, it is common to see an error message on the front panel. If a *BATTERY HAS FAILED* or *RAM* error is present, it is necessary to reset the Model 4000. It is possible to reset the unit via the **`RESET`** function key off the main menu, or via hardware jumper #5 on the controller digital board if there is no front panel, or it is not functioning.



☛For safety, it is advisable to only apply power to the Model 4000 when it is contained in the outside sheetmetal

For 10 Mhz units (serial #'s less than 91-1114XXXX), JU5 is located around the middle of the controller board. Without power to the Model 4000, remove the jumper, apply power (you will hear the fault relay click), remove power, re-insert the jumper, and reassemble the unit.

For 12mHz units (serial #'s greater then 91-1114XXXX) JU5 is located near PROM sockets U2 and U4. Without power applied, move the jumper to the other available position (▪▪• to •▪▪), apply power for a few seconds, remove power, re-position the jumper, and reassemble the unit.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Resetting the Model 4000 via the front panel or the hardware jumper will erase all user programs.

RS-232C Communications

If you are having problems communicating with the Model 4000, use the following procedure to troubleshoot the communications interface.

① Make certain the transmit of the host is wired to the receive of the 4000 and receive of the host (terminal or computer) is wired to the transmit of the 4000.

Try switching the receive and transmit wires on either the host or 4000 if you fail to get any communication.

Verify that you have wired RS232 on the Model 4000 to the port selected.

② Some serial ports require handshaking. If so you may establish three-wire communication by connecting RTS to CTS (usually pins 4 and 5) and DSR to DTR (usually pins 6 to 20) at the host end.

③ Configure the host and peripheral to the same baud rate, number of data bits, number of stop bits, and parity.

④ If you receive double characters (e.g., typing **A** and receiving **AA**), your computer is set for half duplex. It should be set to full duplex.

⑤ Use DC common or signal ground as your reference, *not* earth ground.

⑥ Cable lengths should not exceed 50 ft. unless you are using some form of line driver, optical coupler, or shield. As with any control signal, be sure to shield the cable to earth ground at one end only. (The absolute encoder cable cannot exceed 20 feet.)

⑦ To test your terminal or terminal emulation software for proper three-wire communication, unhook your peripheral device and transmit a character. You should not receive an echoed character. If you do , you are in half duplex mode. Change the setup to full duplex. Connect the host's transmit and receive lines and send another character. You should receive the echoed character. If not, consult the manufacturer of the host's serial interface for proper pin outs.

⑧ Check to make sure the **RS232** statements in the Model 4000 are set. The **RS232** statements are set in the default **POWER_UP** program and if these statements get modified or deleted, you will most likely have communication problems. If you have a front panel, enter the RS232 statement parameters as follows: **RS232 PORT1**(or 2) = **9600 8 OFF 1 ON ALL**, as described in *Chapter 4, Software Reference*. If you have no front panel and can't communicate at all, it may be necessary to perform the hardware reset (see previous section) on the Model 4000 to bring back the default **POWER_UP** program and default RS232 settings. If you are using Model 4000 Applications Software version 1.0 through 2.0 refer to ***More RS-232 Hints*** in the *Communication Troubleshooting* section of this user guide.

Returning the System

If you must return your Model 4000 system to affect repairs or upgrades, use the following steps:

① Get the serial number and the model number of the defective unit, and a purchase order number to cover repair costs in the event the unit is determined by the manufacturers to be out of warranty.

② Before you return the unit, have someone from your organization with a technical understanding of the Model 4000 system and its application include answers to the following questions.

- ❑ What is the extent of the failure/reason for return?
- ❑ How long did it operate?
- ❑ Did any other items fail at the same time?
- ❑ What was happening when the unit failed (e.g., installing the unit, cycling power, starting other equipment, etc.)?
- ❑ How was the product configured (in detail)?

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- ❑ What, if any, cables were modified and how?
- ❑ With what equipment is the unit interfaced?
- ❑ What was the application?
- ❑ What was the system environment (temperature, enclosure spacing, unit orientation, contaminants, etc.)?
- ❑ What upgrades, if any, are required (hardware, software, user guide)?

③ In the USA, call Parker Compumotor for a Return Material Authorization (RMA) number. Returned products cannot be accepted without an RMA number. The phone number for Parker Compumotor Application Engineering Department is (800) 358-9070.

Ship the unit to: Parker Hannifin Corporation, Compumotor Division
5500 Business Park Drive, Suite D
Rohnert Park, CA 94928
Attn: RMA # xxxxxxx

④ In the UK, call Parker Digiplan for a GRA (Goods Returned Authorization) number. Returned products cannot be accepted without a GRA number. The phone number for Parker Digiplan Repair Department is 0202-690911. The phone number for Parker Digiplan Service/Applications Engineering Department is 0202-699000.

Ship the unit to: Parker Digiplan, LTD
21 Balena Close
Poole, Dorset
England, BH177DX
Attn: GRA # xxxxxxx

④ Elsewhere: Contact the distributor who supplied the equipment.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ACCESS
CONTRAST
EDITOR
- COPY
- DELETE
- EDIT
- INSERT
- FIND

GOBACK
IMMED
LOAD
PRINT
RESET
REV
RUN
TEST

Statements

ACCEL (Acceleration)
ACCEL JOG (Jog Acceleration)
ACCEL HOME (Accelerate Home)
ACCEL PATH (Accelerate, Linear Interpolated)
ACCELJOY (Joystick Acceleration)
ACCEL VSCALE ((Velocity Scaling Acceleration)

BREAK (Break)

DATA (Stores Data)
DECEL ((Deceleration)
DECEL SLIMIT (Deceleration Software Position)
DECEL HLIMIT (Deceleration Hardware Input)
DECEL JOG (Jog Deceleration)
DECEL PATH (Deceleration Linear Interpolation)
DECEL JOY (Joystick Deceleration)
DEFINE Define Input & Output)
DEFINE IN24 (Set Input Logic Type)
DEFINE OUT24 (Set Output Logic Type)
DEFINE JOYSL (Define Input & Output Parameters)
DEFINE JOYSH (Define Input & Output Parameters)
DEFINE CENTER (Define Input & Output Parameters)
DEFINE CTR_DB (Define Input & Output Paramters)
DEFINE END_DB (Define Input & Output Paramters)
DEFINE VSCL_L (Define Velocity Scaling Parameters)
DEFINE VSCL_H (Define Velocity Scaling Parameters)
DEFINE HOME (Define Home Parameters)
DEFINE HOME TYPE (Define Home Parameters)
DEFINE HOME EDGE (Define Home Parameters)
DEFINE HOME FDIR (Define Home Parameters)
DEFINE HOME BACK (Home Backup to Edge Setting)
DEFINE FAULT (Define a Fault)
DEFINE ARROW (Reverse Arrow Key Function)
DEFINE VAR (Configure Variables)
DEFINE TERM (Define New Termination Character)
DEFINE ON RET (Configure How An Interrupt Returns)
DEFINE TRIGDB(Define Trigger Debounce Time)
DISPLAY (Display ASCII Information)
DONE (Stop Program Execution)

ENABLE (Enable/Disable System Function)
ENABLE ON (Enable/Disable All Statements)
ENABLE NDBWAIT (Wait Until Within Deadband)
ENABLE STALL (Enable/Disable Stall Detection)
ENABLE POSM (Enable/Disable Position Maintenance)
ENABLE MOVKIL (Enables Kill-on-Stall)
ENABLE HLIMIT (Enable/Disable Limit Input)
ENABLE M_SLIM (Enable/Disable Position Limit)
ENABLE E_SLIM (Enable/Disable Position Limit)
ENABLE JOY (Enable Joystick)
ENABLE JOG (JOG Mode)
ENABLE TCH_JOG (Enable Teach Jog)
ENABLE TCH_JOY (Enable Teach Joystick)
ENABLE MOVOUT (Enable Output When in Motion)
ENABLE PORT (Enable/Disable Communications port)
ENABLE DIR REV (Enable/Disable Move Direction Reversal)
ENABLE VSCALE (Enable Velocity Scaling)
ENABLE REGn (Enable Registration Moves)
ENABLE TRIG REV (Enable inversion of active trigger level)
ENABLE REG SRV (Enable Registration Compensation Of Servo Following Error)
ENCO (Set Encoder Parameters)
ENCO BKL Set Backlash Distance)
ENCO MRES (Set Motor Resolution)
ENCO ERES (Set Encoder Resolution)
ENCO DBND Set Deadband Position Maintenance)
ENCO ABS (Enable Absolute Encoders)
ENDLOOP (Endloop)

GOSUB (Gosub)
GOTO (Go to)

IF (If Then Branch)
IF Q (Compare Q Variable)
IF IN24 (Compare 24 Input to Pattern)
IF BIT (Compare one of the 24 inputs)
IF TRIG (Compare 1 of 4 Trigger inputs)
IF JOYTRIG (Compare Joystick Input)
IF SLIMIT (Branch if Position Passed)
IF HLIMIT (Branch if Limit Input Active)
IF ESTOP (Branch if ESTOP Input Active)
IF F-KEY (Branch if Function Key Input Pressed)
IF ENTERK (Branch if ENTERK Key Pressed)
IF STARTK (Branch if STARTK Key Pressed)
IF MENUREC (Branch if MENU RECALL Key Pressed)
IF ARROWR (Branch if Arrow Right Key Pressed)
IF ARROWL (Branch if Arrow Left Key Pressed)
IF ARROWU (Branch if Arrow Up Key Pressed)
IF ARROWD (Branch if Arrow Down Key Pressed)
IF STALL (Branch if Stall Condition Exists)
IN (Input to a Q Variable)
IN LCD (Prompt the LCD for Input)
IN PORT (Prompt the a Remote Port for Input)
IN IN24 (Get Status of 24 Inputs)
IN POS (Get Current Position of an Axis)
IN VEL (Get Current Velocity of an Axis)
IN STATUS (Get Current Status of an Axis)
IN DATA (Get the Next Data Element)
IN BIT (Get Status of 1 of 24 inputs)
IN TRIG (Get Status of Trigger Input)
IN ANALOG (Read Analog Inputs)
IN ERROR (Get the Execution Error Number)

LABEL (Define a Label)
LINT Enable/Disable Linear Interpolation)
LOOP (Repeat a series of statements)

MATH (Mathematic Calculations)
MODE (Mode)
MOVE (Start a Move)
MOVI (Start a Move Don't wait for its finish)

ON (Interrupt on Condition Met)
ON IN24 (Interrupt on Matching 24 Inputs)
ON BIT (Interrupt on on matching 1 of 24 inputs)
ON JOYTRIG (Interrupt on Joytrig Input Active)
ON TRIG (Interrupt on Position Limit Reached)
ON SLIMIT (On Soft Limit Condition)
ON HLIMIT (Interrupt on Limit Input Active)
ON ESTOP (Interrupt on ESTOP Input Active)
ON ERROR (Interrupt on Error Condition)
ON MENUREC (Interrupt on Menu Recall Pressed)
ON STOPK (Interrupt on Stop Key Pressed)
ON STARTK (Interrupt on Start Key Pressed)
ON F-KEY (Interrupt on 1 of 6 F-Key Pressed)
ON STALL (Interrupt on Stall Condition)
ON FAULT (Interrupt on Drive Fault)
ON EABSLIM (Check For Absolute Encoder Roll-over)
ON Qn (Interrupt on Qn Variable)
ON M_INC (Interrupt on Motor Incremental Position)
ON M_ABS (Interrupt on Motor Step Absolute Position)
ON E_INC (Interrupt on Encoder Incremental Position)
ON E_ABS (Interrupt on Encoder Absolute Position)
ON TIME (Interrupt After Time Elapsed)
OUT (Send out data)
OUT LCD (Send out ASCII string to LCD)
OUT PORT (Send out ASCII string to RS232)
OUT OUT24 (Set 24 Programmable Outputs)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

OUT BIT (Set 1 of 24 Programmable Outputs)
OUT POB (Set 1 of 24 Axis Outputs)

PASSWD (Define Password)
PATT (Define I/O Pattern)
PDEF (Position Definition)
POSM (Set Position Maintenance Factors)
POSM GAIN (Set Velocity Gain for Position Maint.)
POSM MAXVEL (Set Maximum Velocity for Position Maint.)

RESTORE (Reset Data Statement Pointers)
RETURN (Return from Subroutine)
RS232 (Set RS-232C Parameters)

SEG (Define Complex Move Segment)
SEG BEG (Define First Complex Move Segment)
SEG MID (Define 2nd, 3rd, or 4th Complex Move Segment)
SEG END (Define Last Complex Move Segment)
SEG REGn (Define Registration Distance)
SLIMIT (Set Software Position Limit Parameters)
SHTDN (Shutdown Motor/Drive)

UNIT (Set Scale Factors)
UNIT POS (Set Position Scale Factor)
UNIT VEL (Set Velocity Scale Factor)
UNIT ACCEL (Set Acceleration Scale Factor)
UNIT PATH POS (Set Scale Factor for Path Velocity)
UNIT PATH VEL (Set Scale Factor for Path Velocity)
UNIT PATH ACCEL (Set Scale Factor for Path Acceleration)

VEL (Set Peak Velocity)
VEL S/S (Set Start Stop Velocity)
VEL PATH (Set Linear Interpolation Path)
VEL HOME (Set Move Home Peak Velocity)
VEL RANGE (Set Velocity Range)
VEL JOGL (Set Velocity Jog Low)
VEL JOGH (Set Velocity Jog High)
VEL JOYL (Set Velocity Joystick Low)
VEL JOHL (Set Velocity Joystick High)

WAIT (Wait for Condition)
WAIT FOR SECONDS (Wait for Time)
WAIT FOR MOVES (Wait for Move Completion)
WAIT FOR IN24 (Wait for 24 Programmable Inputs)
WAIT FOR BIT (Wait for 1 of 24 Programmable Inputs)
WAIT FOR F-KEY (Wait for an F-Key)
WAIT FOR JOYTRIG (Wait for the Joystick Input)
WAIT FOR TRIG (Wait for 1 of 4 Trigger Inputs)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

X Language Command Listing

A (Acceleration)
ABS (Enable Absolute Encoders)
AD (Deceleration)
AP (Path Acceleration)
ADP (Path Deceleration)
AVS (Velocity Scale Acceleration)
B (Buffer Status)
BSEG (Define Start Segment)

C (Continue)
CG (Correction Gain)
CR (Carriage Return)

D (Distance)
DB (Dead Band)
DCR (Define C Axis Resolution)
DP (Define Path)
DPA (Define Path Absolute)
DPI (Define Path Incremental)
DPL (Define Path Local Coordinates)
DPO (Define Path Segment Outputs)
DPR (Define P Axis Ratio)
DPW (Define Path Work Coordinates)
DTR (Disable Trigger Interrupt)
DW (Dead Band Window)

ENDP (End Path Definition)
ESEG (Define Endinbg Segment)
ETR (Enable Trigger Interrupt)
ER (Encoder Resolution)

FR (Encoder Functions Report)
FSA (Set to Incremental/Absolute Mode)
FSB (Set Indexer to Motor/Encoder Mode)
FSC (Enable/Disable Position Maintenance)
FSD (Stop on Stall)

G (Go)
GA (Go Home Acceleration)
GH (Go Home)
GL (Go Linear)
SP (Execute Path)
GSEG (Start Segment Move)

H± (Set Direction)
^H (Delete)

ID (X Center Data)

J (Enable/Disable Joystick)
JA (Joystick Acceleration)
JAD (Joystick Deceleration)
JCDB (Joystick Center Deadband)
JD (Y Center Data)
JEDB (Joystick End Deadband)
JDH (Define Joysel High Channel)
JDL (Define Joysel Low Channel)
JVH (Joystick High Velocity)
JVL (Joystick Low Velocity)
JZ (Joystick Zero)

K (Kill)

L (Loop)
LA (Limit Acceleration)
LINE (Define Line Segment)

MC (Mode Continuous)
MSEG (Start Middle Segment)
MN (Mode Normal)
MR (Select Motor Resolution)
MV (Maximum Correction Velocity)

N (End of Loop)

O (Output)
OCW (Center Specified CW Arc Segment)
OCCW (Center Specified CCW Arc Segment)
OR (Report Status of Functional Setup)
OS (Function Setup Report)
OSA (Invert Encoder Direction)
OSB (Backup to Home Switch)
OSC (Define Active State of Home Switch)
OSD (Enable Encoder Z Channel Input)
OSE (Enable/Disable Stall Detect)
OSG (Final Homing Direction)
OSH (Reference Edge of Home Switch)

P (Incremental Position Report)
PD (Position Definition)
PDEL (Delete Path)
PR (Absolute Position Report)
PS (Pause)
PX (Report Encoder Absolute Position)
PZ (Position Zero)

Q (Quit Buffer)

R (Request Controller Status)
RA (Limit Switch Status Report)
RB (Report Loop, Pause, Shutdown, Trigger Status Request)
RC (Report Closed Loop and Go Home Status)
RCW (Radius Specified CW Arc Segment
RCCW (Radius Specified CCW Arc Segment)
RD (Radius Data)
RE (Resume)
REG (Define Registration Move)
RI (Simulate Trigger Interrupt)
RV (Revision Level)

S (Stop)
SR (Special Status Request)
SSF (Select Velocity Range)
SSG (Clear/Save Command Buffer on Limit)
SSH (Clear/Save Command Buffer on Stop)
ST (Shutdown)

T (Time)
TR (Wait for Trigger)
TS (Trigger Input Status)

U (Pause and Wait for Continue)
UR (Scale Factor Status)
US (Set Distance Scale Factor)

V (Velocity)
VP (Path Velocity)
VS (Set Start/Stop Velocity)
VSE (Velocity Scale Enable)
VSH (Velocity Scale Source High)
VSL (Velocity Scale Source Low)

W3 (Hexadecimal Position Report)

XD (X Axis Data)
XLD (Local X Coordinate Data)
XWD (Work X Coordinate Data)

Y (Stop Loop)
YD (Y Axis Data)
YLD (Local Y Coordinate Data)
YWD (Work Y Coordinate Data)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Command Listing (X Series vs. 4000 Series)

X Series	Description	Code Command List
"	Quote	OUT PORTn
#	Step Sequence	*Available Thru Front Panel*
;	Comment Field	'
A	Acceleration	ACCEL
AB	Report Analog Voltage, Binary	IN ANALOG
AD	Deceleration	DECEL
AV	Report Analog Voltage	IN ANALOG
B	Buffer Status	*Available with IEEE-488 option*
BCPE	Buffer Configure Position Error	ENCO BKL
BCPG	Buffer Configure Proport. Gain	POSM GAIN
BCPM	Buffer Configure Prop. Max.	POSM MAXVEL
BL	Backlash	
BS	Buffer Status Report	
C	Continue	*RESUME–Remote Computer, Front Panel*
CG	Correction Gain	POSM GAIN
CM	Set Correction Mode For Pos. Maint.	
CPE	Configure Position Error	ENCO BKL
CPG	Configure Proportional Gain	POSM GAIN
CPM	Configure Proportional Max.	POSM MAXVEL
CR	Carriage Return	OUT PORTn ^^
D	Distance	*See MOVE, MOVI*
DCLR	Clear RP240 Display	OUT LCD BLANK
DCNT	Enable/Disable RP240 Pause/Continue	
DFS	Display Flags For Servo Errors	*Indexer errors assigned to Q VAR with IN*
DFX	Display Flags For Indexer Status	*Axis Status assigned to Q VAR with IN*
DIN	Disable Inputs	ENABLE
DLED	Turn On/Off RP240 Leds	N/A
DOUT	Disable Outputs	ENABLE
DP	Distance Point	
DPA	Display Actual Position	DISPLAY MABS, EABS
DPC	Position RP240 Cursor	N/A
DPE	Display Position Error	
DR	Display Parameters	
DRD	Read Distance Via Parallel I/O	IN IN24
DSTP	Enable/Disable RP240 Stop	ON STOPK
DTXT	Display Text On RP240 Lcd	OUT LCD
DVA	Display Actual Velocity	DISPLAY VEL
DVO	Display Variable Data On RP240 Lcd	OUT LCD
DVS	Display Velocity Setpoint	
DW	Deadband Window	ENCO DBND
E	Establish Comm. Interface	ENABLE PORTn
ELSE	Else Portion Of If Command	
ER	Encoder Resolution	ENCO ERES
F	Disable Communications	ENABLE PORTn
FAC	Set Following Sync. Rate	*See FOL MDIST*
FBS	Following Base	*See FOL SYNC_OFF*
FC	Following Learning Count	*See FOL SYNC_OFF*
FEN	Set Following Sync Count	*See FOL MDIST*
FIN	Following Increment	*See FOL SHIFT*
FOL	Following Percent	FOL RATIO
FOR	Following Ratio	FOL UNIT MASTER, UNIT POS
FP	Follwing Encoder Point	FOL WAIT, FOL MOVEWT
FPA	Following Encoder Abs Point	FOL WAIT, FOL MOVEWT
FRD	Read Following Via Parallel I/O	
FS	Encoder Functions Report	
FSA	Set Incremental/Abs Mode	MODE E_INC/E_ABS
FSB	Motor/Encoder Mode	MODE M_INC/E_INC MODE E_ABS/M_ABS
FSC	Position Maintenance	ENABLE POSM
FSD	Stop On Stall	ON STALL
FSE	Enable Output#6 On Stall	ON STALL

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

FSI	Enable/Disable Following	FOL ENABLE
FSK	Set Following Learn Mode	
FSL	Enable/Disable Self Correction	*See FOL SYNC_OFF*
FSM	Set Absolute Encoder	ENCO ABS
FSN	Set Pulse Following	
FSP	Set Tracking Mode	
FUN	Enable/Read RP240 Function Keys	ON F_KEY, WAIT FOR F_KEY, IF F-KEY
G	Go	MOVE, MOVI
Gnnn	Synchronized Go	LINT MODE, MOVE, MOVI
GA	Go Home Acceleration	ACCEL HOME
GD	Go Defined	
GDEF	Configure Move Definition	
GH	Go Home	MOVE HOMECW/HOMECCW
	Movi Homecw/Homeccw	
GHA	Go Home Acceleration	ACCEL HOME
GHAD	Go Home Deceleration	ACCEL HOME
GHF	Go Home Final Velocity	*Fixed at 2,500 steps/sec*
GHV	Go Home Velocity	VEL HOME
GOSUB	Branch To Subroutine	GOSUB
GOTO	Go To A Subroutine	GOTO
^H	Backspace	^H
H	Set Direction	
HALT	Halt Program Execution	*STOP-Remote comuter-RS-232, IEEE*
I	Load Move Data	
ID	Immediate Distance	
IF	If Command	IF
IN	Set Input Functions	ENABLE
INL	Set Input Active Level	DEFINE IN24, ENABLE TRIG REV
INR	Enable/Disable Reg Input	ENABLE REGn
IO	Immediate Output	OUT
IS	Input Status	IN BIT, DISPLAY IN24
IV	Immediate Velocity	
J	Enable/Disable Joystick	ENABLE JOY
JA	Jog Acceleration	ACCEL JOG
JAD	Jog Decelaration	DECEL JOG
JB	Joystick Backlash	DEFINE
JD	Set Joystick Deadband	DEFINE CTR_DB, DEFINE END DB
JV	Joystick Compensation Velocity	
JVH	Jog Velocity High	VEL JOGH
JVL	Jog Velocity Low	VEL JOGL
JZ	Joystick Zero	DEFINE CENTER
K	Kill	MOVE/I KILL
L	Loop	LOOP
LA	Limit Acceleration	
LAD	Limit Deceleration	DECEL HLIMIT
LD	Limit Disable	ENABLE HLIMIT
LF	Line Feed	
LRD	Read Loop Count Via Parallel I/O	IN IN24
MA	Mode Alternate	
MC	Mode Continuous	MOVE SLEW, MOVI SLEW
MN	Mode Mormal	*All but MOVE/MOVI SLEW*
MPA	Mode Position Absolute	MODE E_ABS/M_ABS
MPI	Mode Position Incremental	MODE E_INC/M_INC
MPP	Mode Position Profile	*See MOVI*
MR	Motor Resolution	ENCO MRES
MSL	Clock Source For Timed Data Stream	
MSS	Start Master Clock For T/D Stream	
MV	Set Max Correction Velocity	POSM MAXVEL
N	End Loop	ENDLOOP
NG	End Position Profile	WAIT FOR MOVES
NIF	End If Command	
NUM	Enable RP240 Numeric Keypad	
NWHILE	End While Command	
O	Output Activation	OUT BIT, OUT OUT24, OUT POB
OFF	Drive Shutdown	SHTDN DRIVE YES
ON	Drive Activate	SHTDN DRIVE NO

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

OR	Report Function Setups	
OS	Report Function Setups	
OSA	Set Encoder Direction	
OSB	Back Up To Home	DEFINE HOME BACK
OSC	Define Active State Of Home Switch	DEFINE HOME TYPE
OSD	Define Active State Of Z Chnl Input	DEFINE HOME TYPE
OSE	Enable Stall Detect	ENABLE STALL
OSF	Set Max Joystick Velocity	VEL JOYH, VEL JOYL
OSH	Set Final Go Home Direction	DEFINE HOME FDIR
OUT	Set Output Functions	ENABLE
OUTL	Output Active Level	DEFINE OUT24
OUTP	Output On Position	*See ON POSN AXISn*
P	Report Incremental Position, ASCII	
PB	Report Incremental Position, Binary	
PF	Follower Position Report	IN Qn AXISn MAS_P
PFZ	Set Follower Counter To Zero	*See FOL NEWCYC*
PR	Report Absolute Position	DISPLAY ON LCD M_ABS, IN Qn=POS, OUT PORT/LCD
PS	Pause	WAIT
PX	Report Encoder Abs. Position, Ascii	DISPLAY ON LCD E_ABS, IN Qn=POS, OUT PORT/LCD
PXB	Report Encoder Abs. Postion, Binary	
PZ	Position Zero	PDEF
Q0	Exit Velocity Profiling Mode	
Q1	Enter Velocity Profiling Mode	
QS	Report Status Of Qs Commands	
QSA	Interrupt On Trigger #1 High	ON TRIG
R	Request Indexer Status	
RA	Limit Switch Status Request	IN STATUS
RB	Report Loop, Pause, Shutdown, Trig	
RC	Report Closed Loop & Go Home Status	
REG	Configure Registration Move	SEG REGn
REPEAT	Repeat Command	LOOP, GOTO
RG	Go Home Status Report	
RIFS	Return Indexer To Factory Settings	*RESET-Remote computer, Front Panel*
RM	Rate Multiplier Velocity Stream Mode	
RS	Report Status Of Sequence Execution	
RSE	Report Servo Errors	IN ERROR
RSIN	Set Variables Interactively	*IN PORT, VAR- Remote Computer*
RV	Report Software Revision	*REV-Front Panel*
S	Stop	*Front Panel Stop Key*
SCR	Set Standby Current Reduction	
SD	Streaming Data	
SL	Software Limit	SLIMIT
SLD	Software Limit Disable	ENABLE M_SLIMIT/E_SLIMIT
SN	Scan Delay Time	
SP	Set Absolute Position	PDEF
SR	Report Configuration Status	
SS	Report Configuration Status	
SSA	Rs232 Echo Control	RS232, DEFINE GPIB
SSD	Mode Alternate Stop Mode	
SSF	Normal/Low Velocity Range	VEL RANGE
SSG	Clear/Save Command Buffer On Limit	
SSH	Clear/Save Command Buffer On Stop	
SSI	Enable/Disable Interactive Mode	RS232
SSJ	Enable Disable Cont. Scan Mode	IF IN24, ON IN24
SSL	Enable Resume Execution	
SSN	Set Message Mode	RS232, DEFINE GPIB
SSO	Set Sequence Select	
SSQ	Set Drive Fault Polarity	DEFINE FAULT
ST	Shutdown Drive	SHTDN DRIVE
STOP	Stop	*STOP KEY-Remote Computer,Front Panel*
STR	Strobe Output Delay	
T	Time Delay	WAIT, ON TIME
TD	Time Delay For Data Stream Mode	
TDR	Set Mininum Time Between Reg Moves	DEFINE TRIG DB
TEST	Test Motion	*Front Panel Test Prog, Test Input Pin*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TF	Set Following Time	**See** *FOL SMOOTH*
TM	Move Time Report	
TR	Wait For Trigger	**WAIT FOR TRIG, WAIT FOR BIT**
TRD	Read Timer From Parallel I/O	**IN IN24**
TS	Report Trigger Status	**IN TRIG, IN IN24**
TW	Set Thumbwheel Input Mode	
TX	Transmit Variable And String	**OUT PORT**
U	Pause And Wait For Continue	
UNTIL	Until Part Of Repeat	
UR	Report Scale Factor Status	
US	Set Position Scale Factor	**UNIT POS**
V	Velocity	**VEL**
VAR	Variable	**MATH**
VARD	Read Variables Via Parallel I/O	**IN IN24**
VRD	Read Velocity Via Parallel I/O	**IN IN24**
VS	Start/Stop Velocity	**VEL SS**
W1	Signed Binary Position Report	**INQ=POSN, MATH, PATT**
W3	Hexadecimal Position Report	*/W3 -Remote Computer*
WHEN	Set When Condition	**ON**
XBS	Sequence Memory Available	*Displayed When Exiting Front Panel Editor*
XC	Sequence Checksum Report	
XD	Sequence Definition	*LOAD–Remote Computer*
XDIR	Sequence Directory	*DIR–Remote Computer*
XE	Sequence Erase	*Select Prog, Delete Prog-Remote Computer*
XEALL	Erase All Sequences	*RESET-Remote Computer, Front Panel*
XFK	Set Fault Or Kill Sequence	**ON**
XG	Goto Sequence	**GOTO**
XQ	Sequence Interrupt Mode	
XR	Run A Sequence	*GOSUB Run Or Start-Remote*
XRD	Read A Sequence Via Parallel I/O	**IN IN24**
XRP	Sequence Run With Pause	
XS	Sequence Execution Status	
XST	Sequence Step Mode	***RUN STEP–Remote Computer, Front Panel***
XT	Sequence Termination	**DONE**
XTR	Set Trace Mode	**DISPLAY ON LCD TRACE**
XU	Upload Sequence	*PRINT-Remote Computer*
XWHEN	Set When Sequence	**ON**
Y	Stop Loop	**BREAK**
Z	Reset	***RESET-Front Panel***

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

I N D E X

#

!157
+5VDC 199
19" Rack Mount 33

A

ABS 94
Absolute 123
  Coordinates 164
Absolute Encoder 97
Absolute Encoders 98
Absolute Position 168
AC Power 49
Acceleration Velocity Scaling 58
Airflow 33
AL-C 98
Analog 116
Analog Inputs 65, 66, 67, 68, 199
Analog Voltage Limit 67
Application Design 161
Application Developer Software 193
Application Software Diskette 193
AR-C 98
Arithmetic Operations 121
Arrow Key Direction 75
Arrow Keys 5, 75
Auto-Run On Power-Up 163
Automatic Power-Up Execution 161
Aux1 202
Aux2 202
Aux2 Inputs 205
Auxiliary 1 39
Axis
  Fault Led 212
  Fault Light 166
  I/O 43, 203, 205
  I/O Cables 45
  I/O Pinouts 43
  Inputs 17
  LEDS 212
  Programmable Outputs 200

B

Backlash Compensation 166
Backlash Distance 95
Backlight 3
BAS RS-232 Setting 194
Basic System Configuration 8
Basic System Wiring 8
Basic Troubleshooting Procedure 211
Battery-Backed Ram 1
Baud 145
BCD 112, 140, 169
Bench Test 8
Bit 103, 115
BKL 94
Board Monitor Alarm (Bma) 200
Branch 173, 176
  If Arrow Down Key Is Pressed 108
  If Arrow Left Key Is Pressed 108
  If Arrow Right Key Is Pressed 108
  If Arrow Up Key Is Pressed 108
  If Enter Key Pressed 107
  If Estop Input Active 105
  If Function Key 106
  If Limit Input Active 105
  If Menu Recall Key Is Pressed 107
  If Position Passed 104
  If Stall Condition Exists 109
  If Start Key Pressed 107
Branching 99
Break 99, 175

C

Cable
  Length 36, 37, 215
  Encoders 39
Center 66
Center Deadband 170
Ch1-Ch4 65, 66, 68
Circular Buffer 188
Closed-Loop Related Problems 214
Command Statement Description 53
Common Problems & Solutions 213
Communication Ports 93
Compare
  Trigger Inputs 104
  Input To Pattern 103
  Joystick Input 104
  One Of The 24 Inputs 103
  Q Variable 102
Comparison 102
Complex Motion Profile 146, 182
Complex Moves 178
Conditional Interrupts 174
Connections
  Aux 1 8
  Aux 2 8
  Encoder 37
  Fault Output 45
  Input Power 49
  Motor/Drive 12
  Programmable I/O 46
Connector
  Aux1 39
  Aux2 39, 40
  Auxiliary 1 202
  Auxiliary 2 202
  Encoder 9
  Fault Output 203
  Motor/Driver 9, 200
  Operator Interface 203
  Programmable I/O 201
Continuous Moves 123, 125
Contour Velocity 153
Contouring Deviation 197
Contouring Option 210
Contrast 20
Copy Subfunction 24
Creating A New Program 161
Creep Speed 179
CTS 215
Current Sinking 199
CW & CCW Step 200

D

Daughter Statement 54
DC Common 215
Deadband 67, 80
Debounce 148
Deceleration 60
Default Configuration 195
Define
  Position 142
  Registration Distance 148
  A Label 118
  Arrow 64
  Center 63
  Complex Move Segment 146, 147
  CTR_DB 63
  END_DB 63
  Fault 64
  GPIB 64
  I/O Pattern 142
  OUT24 93
  Password 142
  Term 64
  VAR 64
  VSCL_L 63
Degrees 121
Delay 156, 173
Delete Program 191
Delete Subfunction 24
Diagnostic Leds 211
Digiplan 165
Dimensions 204
Dir 190
Direction Reverse 94
Disable
  All On Statements 80
  Encoder Position Limit 83
  Hardware Limit 82
  Motor Position Limit 83
  Position Maintenance 81
  Stall Detection 81
  System Functions 79
Discard 25
Discrete I/O 49, 169
Display 110
Distance Scale Factor 150
Download 195
Downloading 194
Downloading Programs 190
Drive Fault 198

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DSR 215
DTR 215

E

EABS 77
EABSLIM 135
Echo 189
Edit Subfunction 23
Editing An Existing Program 163
Editing Programs 191
Editor 161
Editor Screen 195
EINC 77
Electrical Noise 31, 35
Emergency Stop 105, 131, 198
Enable Stall 95
Enable/Disable Linear Interpolation 118
Enclosure 31
Encoder 17, 166, 198
  Absolute 168
  AR-C 37, 168
  Backlash 109
  Connectors 201
  Mode Commands 167
  Parameters 94
  Power 38, 198
  Resolution 97
  Rolls-Over 169
  Step Mode 166
  Steps 123
End Loop 99
End-Of-Move Correction 166
End-Of-Travel Limits 52
Environmental 31, 197
Erase Program 191
ERES 94, 97
Error
  Codes 116
  Handling 177
  In Parsing Remote Line 190
  Messages 194
  Parsing Remote Line 194
Errors 213
Estop 8, 9, 14, 52, 198, 200
Estop Input 41
Execution Speed 174
E_ABS 143

F

Fan Filter 209
Fault 134, 200
  Active Level 73, 200
  Led 51, 212
  Output Relay 41
Features 1
Feedrate 91
Final Approach Directions 179
Final Home Velocity 179
Find Subfunction 23
Firmware Reference Table 210
FND_LBL 100, 101
Following Option 210
Frequency Range 154
Front Panel 170
  Axes 4
  Axis Fault Leds 9
  Backlight Off/On 5
  C/E 5
  Contrast Decrease 5
  Contrast Increase 5
  Display 3
  Enter 5
  Fault Leds 4
  Function Keys 4
  Functions 10
  Jog Mode 5
  Jog Mode Led 4
  Leds 9
  Menu Mode 5
  Menu Recall 4
  Menus 10
  Numerical Keypad 5
  Options 11
  Resume 5
  Start 5
  Status Led 4, 9
  Stop 5
  Subfunctions 11
  Program Generation 161
Full Or Half Steppers 165
Full Step 96
Function
  Access 18
  Alpha 21
  Arrow 108
  Contrast 20, 189, 190
  Editor 20
  Find-Prog 21
  Goback 29
  Immed 27
  Load 28
  Power-Up 21
  Print 27
  Reset 29
  Rev 29
  Run 19, 26
Function Keys 175
Functional Statement Grouping 55
Functional Test Routines 52
Functions 10

G

Gain 143
Get
  Current Position Of An Axis 113
  Current Velocity Of An Axis 113
  Next Data Element 114
  Status Of Inputs 115
  Status Of 24 Inputs 112
  Status Of 1 Of 24 Inputs 115
  Status Of 1 Of 4 Trigger Inputs 115
  Status Of An Axis 114
GOSUB Beg 100
GOTO Beg 101
GPIB 111, 139
Grounding 35

H

Half Step 96
Handshaking 189, 215
Hardware Travel Limit 62
Hexadecimal 121
Hexidecimal 112, 140, 169
Hlimit 131
Holding Torque 149
Home
  Active Edge 70
  Active State 69
  Back 69
  Backup To Edge 72
  CW, And CCW Limits 198
  Edge 69
  Enable 179
  Enable Active Region 179
  FDIR 69
  Final Direction 71
  Limit Input 69
  Parameters 69
  Type 69
  Velocity 154
HomeCCW178
HomeCW 178
Homing Sequence 178
Host Computer 11
Humidity 197

I

IBM-Compatible 193
Immed 27
IN Error 116
IN24 64, 77, 112, 115, 129
Incremental 123
  Coordinates 164
Input 197
  Analog 41, 42, 43, 170
  Axis 12
  Axis Outputs 44
  Bit 129, 158
  Encoder 12
  Estop 41
  Home Enable 44
  Joystick 41
  Joystick Release 42
  Joystick Trigger 42
  Joystick Velocity 42
  Registration 183
  Test 141
  To a Q Variable 109
  Trigger 44
  TTL-Compatible 198
Inputs/Outputs 169
Insert 192
Insert Subfunction 21
Installation Vi
Instantaneously Start 153
Interrupt 174, 183
  After The Specified Time Has Elapsed 137
  Inputs 129
  On 1 Of 6 F-Keys Pressed 133
  On Absolute Encoder 135
  On Condition True 126
  On Drive Fault Interrupt Active 134
  On Encoder Step Absolute Position Met 137
  On Encoder Step Incremental Position 136
  On Error Condition 132
  On Estop Input 131
  On Inputs 129
  On Joytrig Input 130
  On Limit Input Active 131
  On Matching 1 Of 24 Inputs 129
  On Matching 24 Inputs 129
  On Menu Recall Pressed 132
  On Motor Step Absolute Position Met 136
  On Motor Step Incremental Position Met 135
  On Qn Variable 134
  On Software Limit 130
  On Stall Condition 133
  On Start Key Pressed 133
  On Stop Key Pressed 132
  On Trigger Input 130
  Registration 183
  Triggers 183
Invalid Data In Line 190

J

Jog 86

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Acceleration 57
Mode 75
Mode LED 212
Joystick 65, 66, 85, 165, 170
Center 66
Center Deadband 67
Control 170
Deadband Voltage 67
High Velocity 155
Input 158
Low Velocity 155
Mode 85
Release 199
Release Input 42
Select 42, 199
Select Input 171
Source—High 66
Source—Low 65
Statements 172
Teach Mode 172
Trigger 199
Trigger Input 42, 104
Velocity Input 42
Velocity Range 199
Voltage Limit 67
Wiring 42
Joytrig 130

K

Kill-On-Stall Function 82

L

Label 99, 102, 173
LCD 110, 138
LCD Display 190
LCD For Input 110
LEDS 51
Limits 164
End-Of-Travel 13, 44, 164
End-Of-Travel 13, 164
Hardware 164
Software 164
Switch Input 13
Switches 82
Linear Interpolation 118
Linear Interpolation Path Acceleration 57
Listing 191
LNUM 77
Load 28
Logical Functions 121
Loop 99, 173

M

MABS 77
Maintenance 209
Math 110
Mathematic Calculations 121
Maximum Velocity 144
Memory 3
Menu
Structure 176
Menus 10, 175
MINC 77
Minimal Safety Verifications 52
Mode 164
Modular Jacks 36
Monitor Alarm 9
Motion Output 93
Motor Position 113
Motor Resolution 96
Motor Steps 123
Motor/Drive
Connectors 200
Motor/Drive Connections 36
Mounting 32
Desk Mount 35
Door 34
Operator Panel 34, 36
Rack 33
Move 118
Completion 157
Direction Reverse 94
Home Acceleration 57
Homeccw 124
Homecw 125
Kill 124
Profile 182
Resume 125
Segccw 125
Segcw 125
Segmented 183
Slew 124
Stop 124, 125
Velocity 124
Moves
Beginning Segment 182
Ending Segment 182
Middle Segments 182
Moving/Not Moving Output 93
Movkil 95
Movout's 93
MRES 94, 97
Multiple Program Manipulation 190
M_ABS 142

N

National Electrical Code 35
Nesting 99
Next Data 59
Null 77
Numeric And Text Variables 74

O

Off 78
Open Collector 200
Operating Temperature 197
Operations Tutorial 18
Operator Interfacing 175
Operator Panel 10, 11, 34, 212
Backlight 3
Connections 36
Status LED 211
Optional Features 2
Optional Features Table 210
Opto-22 1, 46, 49, 198, 199
OUT24 64, 77
Outputs 1, 199
Axis 16
Axis Programmable 12, 200
Differential Line Drivers 199
Programmable 12, 199
Thumbwheels 169
Over-Ride 170

P

Packing List 7
Panel Layout 32
Parent And Daughter Statements 54
Parent Statement 54
Parity 145
Password 12, 18, 142, 163
Path
Acceleration 151
Velocity Scale Factor 151
Patt 112, 142, 157
Pattern Statement 142
Pause 156, 173
Performance 197
Phone Cable 9, 36, 203
Pin-Outs 200
Pinouts
Axis Output 43
CCW Limit 43
CW Limit 43
Encoder 38
Fault Output 46
Ground 43
Home Enable 43
Motor Driver 36
Shield 43
Trigger 43
PLC Operation 49
Pointer 185
Polling 174
Polling And Interrupts 173, 174
Port 111, 139
Port 1 15
Port 2 15
Portnum 145
Ports 94
Position 113
Correction 166
Deadband 98
Definition 142
Limit Parameters 148
Maintenance 81, 98, 143, 166
Positioning Mode 123
Positive Logic 64
Positive Logic 64
Potentiometer 171
Power 197
Power Supplies 213
Power_up 161
Preset Moves 123, 125
Print 27
Program Basics 163
Program Creation 161
Program Flow 173
Program Name 195
Programmable
I/O Connectors 201
I/O Pin Outs 47
Input 104
Inputs 1, 17, 198
Opto Isolation 46
Output 16, 140, 141
Outputs 199
Programmable Inputs 103
Prompt 110
Prompt A Remote Port For Input 111
Prompts 194
Pull-Ups 39, 198
Pulse Width 96

R

Radians 121

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Read Analog Inputs 116
Reducing Electrical Noise 212
Registration
  Input 148
  Moves 92, 148, 183
  Segment 185
Relay Wiring 48
Remote
  Command 189
  Device 189
  Execution 192
  Operation 11, 189
  Port 190
Repairs 215
Repeat 120
Repeat A Series Of Statements 120
Replace 20, 25
Replacement Part 209
Reset 29, 195, 214
Reset Data Statement Pointer 144
Reset Jumper 214
Resetting The Model 4000 214
Resolution 96
Restart 195
Restore 114
Return 99
Return From Subroutine 144
Returning The System 215
Rev 29
RJ-11 Jack 203
ROM 12
RS-232 15, 40, 139, 163, 193, 197, 199
  Communications 215
  Line Termination Character 74
  Parameters 145, 194
  Port Configuration 189
  Ports 12
RS-232 Communication
  Baud 40
  Parity 40
  Stop Bit 40
RTS 215
Run 26
Running Your Program 163

S

Scale Factor 165
Schematic
  Channels A, B, & Z 206
Schematics 205
Seconds 137
Seg Reg 92
Seg Regn 184
Segmented Moves 182
Send ASCII Information To LCD 77
Send Out Data 137
Send String And/Or Variable To LCD 138
Send String Or Variable To Remote Port 139
Serial Ports 189
Set
  Acceleration Scale Factor 151
  Axis Output 141
  Jog Mode High Velocity 155
  Jog Mode Low Velocity 155
  Joystick Mode High Velocity 156
  Joystick Mode Low Velocity 155
  Maximum Velocity 144
  Move Home Peak Velocity 154
  Path Velocity 153
  Peak Velocity 152
  Position Scale Factor 150
  Programmable Outputs 140
  Scale Factor 150
  Scale Factor For Path Acceleration 151
  Scale Factor For Path Velocity 151
  Start And Stop Velocity 153
  Velocity Range 154
Set-Up 163
Shipment 7
Shutdown Motor/Drive 149
Signal Ground 215
Significant Digits 138
Single Stepping 163
Software Limit 104, 130
Software Travel Limit 61
Spol 78
Stall 95, 133
Stall Detection 81, 166
Start 26
Start A Move 123, 125
Statement
  ACCEL 56
  ACCEL HOME 57
  ACCEL JOG 57, 84
  ACCEL JOY 58
  ACCEL PATH 57
  ACCEL VSCALE 58
  BREAK 59
  DATA 59, 144
  DBND 94
  DECEL 60
  DECEL HLIMIT 62, 164
  DECEL JOG 62, 84
  DECEL JOY 63
  DECEL PATH 62
  DECEL SLIMIT 61, 164
  DEFINE 63
  DEFINE ARROW 75
  DEFINE CENTER 66
  DEFINE CTR_DB 67
  DEFINE END_DB 67, 170, 171
  DEFINE FAULT 73
  DEFINE HOME 69
  DEFINE HOME BACK 72
  DEFINE HOME EDGE 70
  DEFINE HOME FDIR 71
  DEFINE HOME TYPE 69, 178
  DEFINE IN24 64
  DEFINE JOYSH 66, 85, 171
  DEFINE JOYSL 65, 85, 171
  DEFINE ON RET 76
  DEFINE OUT24 64, 140, 141
  DEFINE TERM 74
  DEFINE TRIGDB 76
  DEFINE VAR 74
  DEFINE VSCL_H 68, 171
  DEFINE VSCL_L 68, 171
  DELETE 189, 191
  DIR 189
  DISPLAY 77, 193
  DONE 78
  ENABLE 79, 94
  ENABLE E_SLIM 83, 164
  ENABLE HLIMIT 82, 164
  ENABLE JOG 84, 165
  ENABLE JOY 85, 165
  ENABLE MOVKIL 81, 82, 166
  ENABLE MOVOUT 93
  ENABLE M_SLIM 83, 164
  ENABLE NDBWAIT 80, 167
  ENABLE ON 80
  ENABLE ON YES 174
  ENABLE PORTN 93
  ENABLE POSM 81, 167
  ENABLE REGN 92
  ENABLE REGSRV 90
  ENABLE STALL 81, 166
  ENABLE TCH_JOG 86, 172
  ENABLE TCH_JOY 85, 172
  ENABLE TRIG REV 91
  ENABLE VSCALE 91, 170
  ENCO 94
  ENCO ABS 98, 168
  ENCO BKL 81, 95, 109, 166
  ENCO DBND 80, 98, 166, 167
  ENCO ERES 97, 166
  ENCO MRES 96, 166
  ENDLOOP 99
  ENTER 107
  ESTOP 105, 177
  F-KEY 158
  FND_LBL 100, 101
  GOBACK 29
  GOSUB 99, 118, 144, 173
  GOTO 100, 118
  HLIMIT 105
  HOMECCW 124
  HOMECW 125
  IF 101
  IF ARROWD 108
  IF ARROWL 108
  IF ARROWR 108
  IF ARROWU 108
  IF BIT 103
  IF ENTERK 107
  IF ESTOP 105
  IF F-KEY 106
  IF HLIMIT 105
  IF IN24 103
  IF JOYTRIG 104
  IF MENUREC 107
  IF Q 102
  IF SLIMIT 104
  IF STALL 109
  IF STARTK 107
  IF TRIG 104
  IMMEDIATE 189, 192
  IN 109, 192
  IN ANALOG 116, 171
  IN BIT 115
  IN DATA 60, 114, 144
  IN ERROR 116, 177
  IN IN24 112
  IN LCD 110, 170
  IN PORT 111, 170
  IN POS 113
  IN Q2 = LCD 175
  IN STATUS 114
  IN TRIG 115
  IN VEL 113
  IN24 169
  INSERT 191
  JOG 165
  JOG MODE 84
  JOYH 199
  JOYSTICK 165
  JOYSTICK ENABLE 199
  KILL 124
  LABEL 118
  LCD 189, 192
  LINT (MODE) 118
  LOAD 189, 194, 195
  LOOP 120, 173
  MATH 121
  MEM 189, 193
  MENU 189
  MENU MODE 84, 165
  MENU RECALL 132
  MENUREC 107
  MODE 123, 182
  MOVE 123, 192
  MOVE HOME 178
  MOVE KILL 131
  MOVE STOP 131
  MOVI 125
  ON 126
  ON BIT 129
  ON EABSLIM 135, 169

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ON ERROR 132, 177
ON ESTOP 131
ON E_ABS 137
ON E_INC 136
ON F-KEY 133
ON FAULT 134
ON GOSUB 174
ON GOTO 174
ON HLIMIT 131
ON IN24 129
ON JOYTRIG 130
ON MENUREC 132
ON M_ABS 136
ON M_INC 135
ON PATT 175
ON QN 134
ON SLIMIT 130
ON STALL 133, 166
ON STARTK 133
ON STOPK 132
ON TIME 137
ON TRIG 130
OUT 137
OUT BIT 141
OUT LCD 138, 170, 175
OUT OUT24 140
OUT POB 141
OUT PORT 139, 170
PASSWD 142
PATT 129, 140, 142
PDEF 113, 123, 142
POSM 143
POSM GAIN 143, 167
POSM MAXVEL 144, 167
POSM VEL 167
POWER-UP 189, 193
POWER_UP 194
PRINT 189, 191
RESET 193
RESTORE 144
RESUME 125, 189, 192
RETURN 144
RS232 145, 190
RUN 192
RUN FROM 189
SEG 146
SEG BEG 146
SEG END 147
SEG MID 147
SEG REG 148
SEGCCW 125
SEGCW 125
SELECT 189, 190
SHTDN 149
SLEWCCW 124
SLEWCW 124
SLIMIT 104, 130, 148
STALL 109
START 107, 163, 189, 192
STOP 124, 189, 192
TRIGS 169
UNIT 150, 151
UNIT ACCEL 56, 61, 62, 63, 151, 165
UNIT PATH 165
UNIT PATH ACCEL 57, 62, 151
UNIT POS 150, 165
UNIT VEL 150, 165
VARI 189
VARIABLE 192, 193
VEL 152, 77
VEL HOME 154
VEL JOGH 84, 155
VEL JOGL 84, 155
VEL JOYH 156, 171
VEL JOYL 155, 171, 199
VEL PATH 153
VEL RANGE 154
VEL SS 153
WAIT 156, 192
WAIT FOR BIT 158
WAIT FOR F-KEY 158
WAIT FOR IN24 157
WAIT FOR JOYTRIG 158
WAIT FOR MOVES 157
WAIT FOR SECONDS 156
WAIT FOR TRIG 159
X-OFF 189
XON 189
Statement Run 163
Status 77
Status Led 51, 211, 212
Step Output Signal 154
Step Pulse 96
Stepping Accuracy 197
Stop Key 132
Storage Temperature 197
Store Data Values 59
Store Error Code In A Variable 116
String 111
String Variables 138
Subfunction
  Copy 24
  Delete 24
  Edit 22
  Find 23
  Insert 21
  Save 25
Subfunctions 11, 21
Subroutine 99, 144
Subroutines 173
Switches
  End-Of-Travel Limit 9
System
  Configuration 8
  Error 177
  Verification 51

T

Teach Joy 85
Teach Mode 85, 86
Terminal Emulation 215
Terminal Emulator 194
Terminals 47
Termination Character 74
Test 8, 198
Test Input 41
Test Procedure
  Axis Outputs 16
  Encoder 17
  Estop Input 14
  Function 12
  Limits 13
  Programmable Inputs 17
  Programmable Outputs 16
  RS-232c 15
  Test Axis Move 14
  Test Routine 12
  Trigger Inputs 17
Thumbwheel Input Operation 50
Thumbwheel Wiring 50
Time 137
Time Delay 156
Trace 77
Transfer 190
Travel Limits 83, 148
Trig 104, 115, 130
Trigger 158, 184, 198
  Debounce Time 184
  Inputs 92, 104, 130
Trigonometric Functions 121
Troubleshooting 210
  Common Problem Symptoms 194
  Communications 194
  RS-232 215
Tutorial 11

U

Unit Scaling Factor 150
Units 165
Upgrade 8
  Contouring 8
  Following 8
  IEEE-488 8
Upgrades 215
Uploading Programs 191

V

VARI 121
Variable 68, 185
Variable Memory 74
Velocity
  Accuracy 197
  Gain Factor 143
  Gain For Position Maintenance 143
  High 156
  Jog 155, 212
  Maximum 152
  Repeatability 197
  Scale Factor 150
  Scaling 68, 170, 171
  Scaling Mode 91
  Scaling Source—High 68
  Scaling Source—Low 68
Ventilation 32
VM50 47
Voltage
  Center 66
VScale Statements 172
VSCL 78

W

Wait 173
  For 1 Of 24 Programmable Inputs 158
  For An F-Key 158
  For Condition 156
  For Deadband 167
  For Inputs 158
  For Joystick Input 158
  For Move Completion 157
  For Programmable Inputs 157
  For Time 156
  For Trigger Inputs 159
  Wait On Deadband 80
Wiring
  EMI 35
  ESTOP 41
  Fault Outputs 45
Wiring Guidelines 35

X

X Command Language 189, 196
Xon/Xoff 40

Z

Z Channel Inputs 178

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP