Compumotor PC-23

Indexer Adapter Box

$915.00

In Stock

Qty Available: 1

Used and in Excellent Condition

Open Web Page

https://www.artisantg.com/62390-1

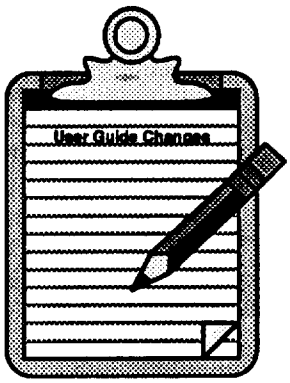
Compumotor

PC23 Indexer
User Guide

Compumotor Division
Parker Hannifin Corporation
p/n 88-007015-03E

Parker

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 88-007015-03**E**, supersedes version 88-007015-03**D**.

The entire user guide has been changed according to the new Compumotor user guide styles and illustration standards. Also, the chapters have been renumbered and reorganized.

The following commands have been added to Chapter 5, *Software Reference*.

DB — Dead Band

DW— Set Dead Band Window

DPA —Display Position Actual

MR — Motor Resolutions (changes to allow for Dynaserv)

O,D — Output on the Fly

SA — Stop All Commands

Table of Contents

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 is intended to help you find and use the information in this user guide.

Assumptions

This user guide assumes that you have the skills or fundamental understanding of the following information:

- IBM (or IBM-compatible) computer experience
- Basic electronics concepts (voltage, switches, current, etc.)
- Working knowledge of at least one computer language (Basic, C, Pascal, assembly, etc.)
- Basic motion control concepts (torque, inertia, velocity, acceleration, distance, etc.)

With this basic level of understanding, you will be able to effectively use this user guide to install, develop, and maintain your system.

Contents of This User Guide

This user guide contains the following information:

Chapter 1: Introduction This chapter provides a description of the product and a brief account of its specific features.

Chapter 2: Getting Started This chapter contains a detailed list of items you should have received with your PC23 Indexer shipment. It will help you to become familiar with the system and ensure that each component functions properly. You will learn how to configure the system properly in this chapter.

Chapter 3: Installation This chapter provides instructions for you to properly mount the system and make all electrical connections. Upon completion of this chapter, your system should be completely installed and ready to perform basic operations.

Chapter 4: 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 5: Software Reference This chapter explains Compumotor's X-Series programming language in detail. It provides an alphabetical listing of all commands, with a syntax and command description for each command. It also describes system parameters that affect command usage.

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 switch and jumper settings and I/O connections.

Chapter 7: Troubleshooting This chapter describes Compumotor's recommended system maintenance and troubleshooting procedures. It also provides methods for isolating and resolving hardware and software problems.

Installation Process Overview

To ensure trouble-free operation, you should pay special attention to the following:

- The environment in which the PC23 will operate
- The system layout and mounting

- ❑ The wiring and grounding practices used

These recommendations are intended to help you easily and safely integrate the PC23 system into your manufacturing facility. Industrial environments often contain conditions that may adversely affect solid state equipment. Electrical noise or atmospheric contamination may also affect the PC23.

Installation Recommendations

Before you attempt to install this product, you should complete the following steps:

① Review this entire manual. Become familiar with the manual's contents so that you can quickly find the information you need.

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

③ Complete the basic system configuration and wiring instructions provided in *Chapter 2, Getting Started. This is a bench test intended to be performed in a bench-top environment.*

④ Perform as many basic moves and functions as you can with the preliminary configuration. You can perform this task only if you have reviewed the entire manual. You should try to simulate the task(s) that you expect to perform when you permanently install your system. **However, do not attach a load at this time.** This will give you a realistic preview of what to expect from the complete configuration.

⑤ After you have tested all of the system's functions and used or become familiar with all of the system's features, carefully read *Chapter 3, Installation*.

⑥ After you have read Chapter 3 and clearly understand what must be done to properly install the system, you should begin the installation process. **Proceed in a linear manner**; do not deviate from the sequence or installation methods provided.

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

The successful completion of these steps will prevent subsequent performance problems and allow you to isolate and resolve any potential system difficulties before they affect your system's operation.

Developing Your Application

Before you attempt to develop and implement your application, you should consider the following:

- ❑ Recognize and clarify the requirements of your application. Clearly define what you expect the system to do.
- ❑ Assess your resources and limitations. This will help you find the most efficient and effective means of developing and implementing your application.
- ❑ Follow the guidelines and instructions outlined in this user guide. **Do not skip any steps or procedures.** Proper installation and implementation can be ensured only if all procedures are completed in the proper sequence.

Conventions

To help you understand and use this user guide effectively, the conventions used throughout this user guide are explained in this section.

Warnings & Cautions

Warning and caution notes alert you to possible dangers that may occur if you do not follow instructions correctly. Situations that may cause bodily

injury are presented as warnings. Situations that may cause system damage are presented as cautions. Refer to the examples shown below.

WARNING

Do not touch the motor immediately after it has been in use for an extended period of time. The unit will be hot.

Related Publications

The following publications may be helpful resources:

- *Parker Compumotor Programmable Motion Control Catalog*
- Seyer, Martin. RS-232C Made Easy: Connecting Computers, Printers, Terminals and Modems. Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1984
- Operations user guide for the IBM or IBM-compatible computerthat you will use with the PC23 Indexer
- Schram, Peter (editor). *The National Electric Code Handbook (Third Edition).* Quincy, MA: National Fire Protection Association

C H A P T E R ①

Introduction

Chapter Objective

The information in this chapter will enable you to understand the PC23's basic functions and features.

Product Description

The Compumotor PC23 is a microprocessor-based indexer that is designed to be inserted into an IBM Personal Computer (PC, XT, or AT) or compatible. *The PC23 is not compatible with PS2 Models 50, 60, or 80.* Manufactured as a single board, the PC23 is designed to be inserted into a single open slot in the computer expansion bus or I/O channel. A separate adaptor box is included to provide an interface to the external motor drivers, limit switches, analog joystick, trigger inputs, programmable outputs, and incremental encoders.

The PC23 will control up to three axes of any size Compumotor motor/drive system, most 200/400 step translator and stepper motor combinations, and other drive systems that accept pulsed control signals.

The PC23 uses a 16-bit processor with custom circuits to simplify generation of motion profiles and shorten processing time.

Product Features

The PC23 provides the following features:

- Three axes of motion control
- Two-axis analog joystick input
- Designed for the IBM PC, AT, XT and most IBM PC compatibles, PS2 (Model 30 only)
- Compatible with all Compumotor drives
- Compatible with Compumotor GCI software
- May be programmed to interrupt the host processor
- 1,000-character buffer per axis permits downloading of multiple move sequences per axis
- Supplied software support disks provide user utility programs, set-up and test routines, and help information
- Supports motor/drive resolutions up to 50,000 steps/rev
- Closed-loop control with an incremental encoder

- User-defined trigger inputs (6) and outputs (6)
- All inputs and outputs are optically isolated
- CW and CCW end-of- travel limits and Home limit provided for all three axes
- User control of pulse source provides for complex move profiles
- Time distance streaming for arbitrary contours

Theory of Operation

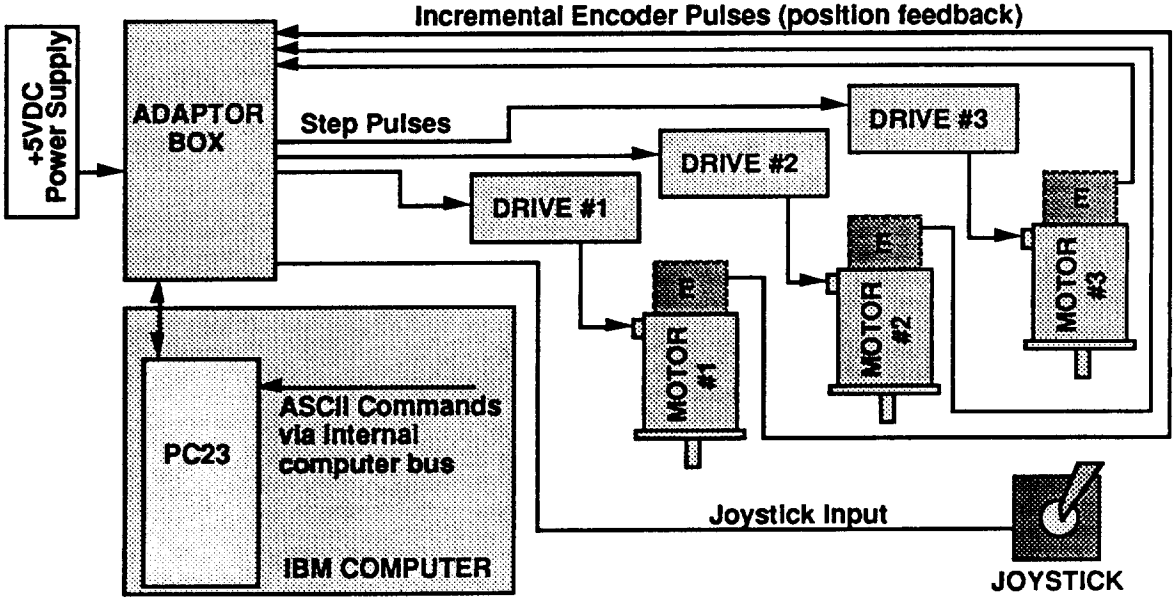
The PC23 receives acceleration, velocity, and position information (in an ASCII format) from your computer. In turn, the PC23 uses that information to generate motion profile command signals for the three axes. Move commands are then sent to the external drive in the form of *step* pulses at a rate of up to 500 kHz. Additional logic provides other necessary functions, such as monitoring limit switches, setting the direction of motor motion, turning outputs on and off, and waiting for trigger inputs. The following figure shows a block diagram illustrating the system's functional processes.

With simple indexer commands, you can move any or all of the three axes as follows:

- Rotate to a precise position and stop
- Rotate at a constant velocity
- Alternate back and forth between two angular positions
- Use a sequential combination of the above moves

You can position two axes with a joystick by wiring a simple 5kΩ potentiometer joystick to the PC23. You can combine joystick control with programming for easy hands-on manipulation.

You may use the PC23 in an open-loop system with no external position feedback devices. To enhance the accuracy of position control you can use an incremental encoder (closed-loop operation). The encoder, mounted either on the load or on the motor, provides a feedback signal to the indexer to indicate actual motor or load position. The encoder interface is intended for applications where desired positional accuracy exceeds the accuracy of the mechanical system components (such as a leadscrew on an X-Y table).



C H A P T E R ②

Getting Started

Chapter Objectives

The information in this chapter will enable you to do the following:

- ❑ Verify that your system arrived without damage
- ❑ Become familiar with system components and their interrelationships
- ❑ Establish the basic system configuration
- ❑ Ensure that each component functions properly

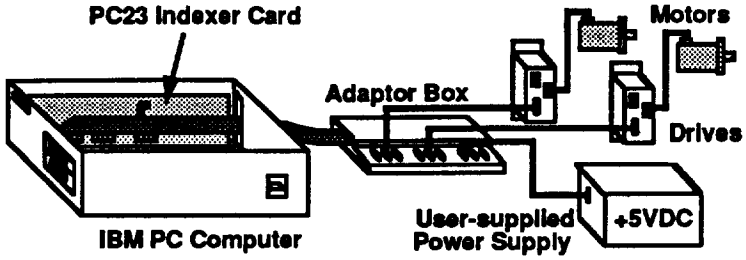
What You Should Have

Upon receipt, you should inspect your PC23 Indexer shipment for obvious damage to its shipping container. Report any damage to the shipping company as soon as possible. Parker Compumotor cannot be held responsible for damage incurred in shipment. The items listed in the following table should be present and in good condition.

Description	Part Number
PC23 PCA card	71-006085-02
Adaptor box w/cable assembly	71-006457-02
PC23 User Guide	88-007015-03
Software support disk #1	95 -011353-01
Software support disk #2	95 -011354-01
25-pin drive cable assemblies (3)	71-007566-10
PC23 card guide	58-004144-01
PCA harness clamp	53-006886-01
Adaptor box mounting brackets (2)	53-006007-01
#4 machine screws (2)	51-006021-01
#6 machine screws (4)	51-006037-01
25-Pin connector plug (3)	43-001989-01
25-Pin connector shell (3)	43-001990-01

Bench Test

This section provides bench test procedures for the PC23 Indexer system. The following figure illustrates the temporary bench test configuration. ***Use the same computer that you will use in the final system configuration discussed in Chapter 3, Installation.*** For test purposes, connection of the encoder and the auxiliary switches is not required.



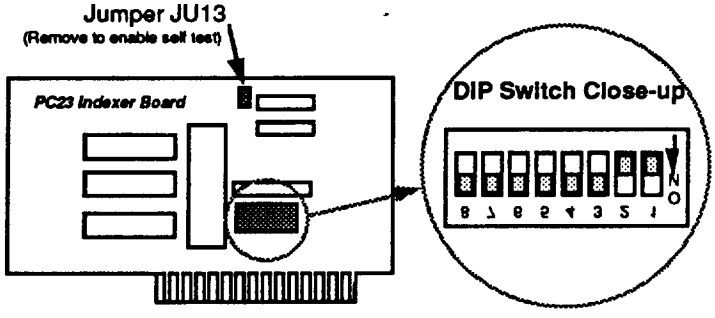
To complete the Bench Test, you will need the following tools and additional items (not provided with the PC23 ship kit):

- ❑ IBM computer
- ❑ Small standard screw driver
- ❑ Needle-nose pliers
- ❑ 5VDC external power supply
- ❑ Wire (to connect external 5VDC power to the adaptor box)

① Set the Address

For the computer to control the PC23, it must know where to write instructions and read responses. This requires that the PC23 have an address.

The address is set using the 8-position DIP switch located on the PC23 indexer card. This package consists of eight switches representing a binary number. Only two of the four address locations are significant: one for control and one for data. Input and output operations use the same address. The address may be set to any number that the PC will recognize as valid.



The DIP switches are *negative true* in that any switch in the position marked ON has a binary value of zero. Switches that are OFF have a non-zero binary value. The sum of the binary values of DIP switches 1 through 8 is the board's *base address*. The binary values assigned to each of the eight DIP switches are listed in the table below.

Switch #	Address	Binary Value (OFF) Decimal	HEX	Default Setting
1	9	512	200	OFF
2	8	256	100	OFF
3	7	128	80	ON
4	6	64	40	ON
5	5	32	20	ON
6	4	16	10	ON
7	3	8	8	ON
8	2	4	4	ON

The PC23 is shipped from the factory with switches 1 and 2 in the OFF position, all others ON . The board address then is 256 + 512 = 768 (or

100 + 200 = 300 hex). As such, the PC23 is configured to occupy I/O address locations 300 hex through 303 hex. The control and status registers are at the odd address location (301 hex), and data registers are at the even address location (300 hex). The default address is suitable for testing and for general use if no other peripheral devices in the computer are using the same address. This address does not conflict with typical devices that reside on the I/O bus, such as graphics adaptors, disk drives, and serial cards. The PC23 occupies four address locations on the I/O bus.

② Enable Self Test

Remove jumper JU13, located above the edge connector and near the top of the PC23 PCA card. Removing this jumper enables you to initiate the self test function. The self test will be used later in the system functional test. You must re-install the jumper when testing is complete.

③ Install the Indexer & Connect the Adaptor Box

Use the following steps to install the PC23 PCA card into your computer:

① **Turn off the power to the computer.**

② Remove the computer's cover to access the internal slots where peripheral PCA cards are added.

③ Remove the sheet metal bracket that covers the external access slot. Save the screws. On IBM and IBM-compatible computers, this will be at the rear access panel where all external connections are made. This bracket will be replaced by the bracket on the end of the PC23 PCA card.

④ Mount the supplied card guide in the computer to support the opposite end of the PCA card. This guide snaps into the holes in the computer's chassis.

CAUTION

While handling the PCA card, be sure to observe proper grounding techniques to prevent electro-static discharge (ESD).

⑤ Run the cable harness from the adaptor box through the slot in the Computer's access panel and plug its four flat cables into the PCA card connectors. Each cable connector is marked with a J designation (1 through 4) that matches a connector on the PC23 PCA card.

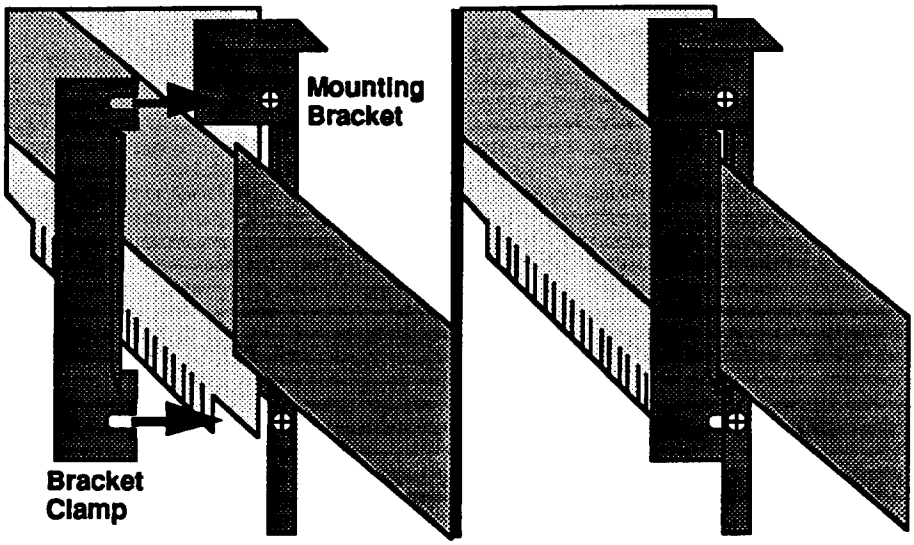
CAUTION

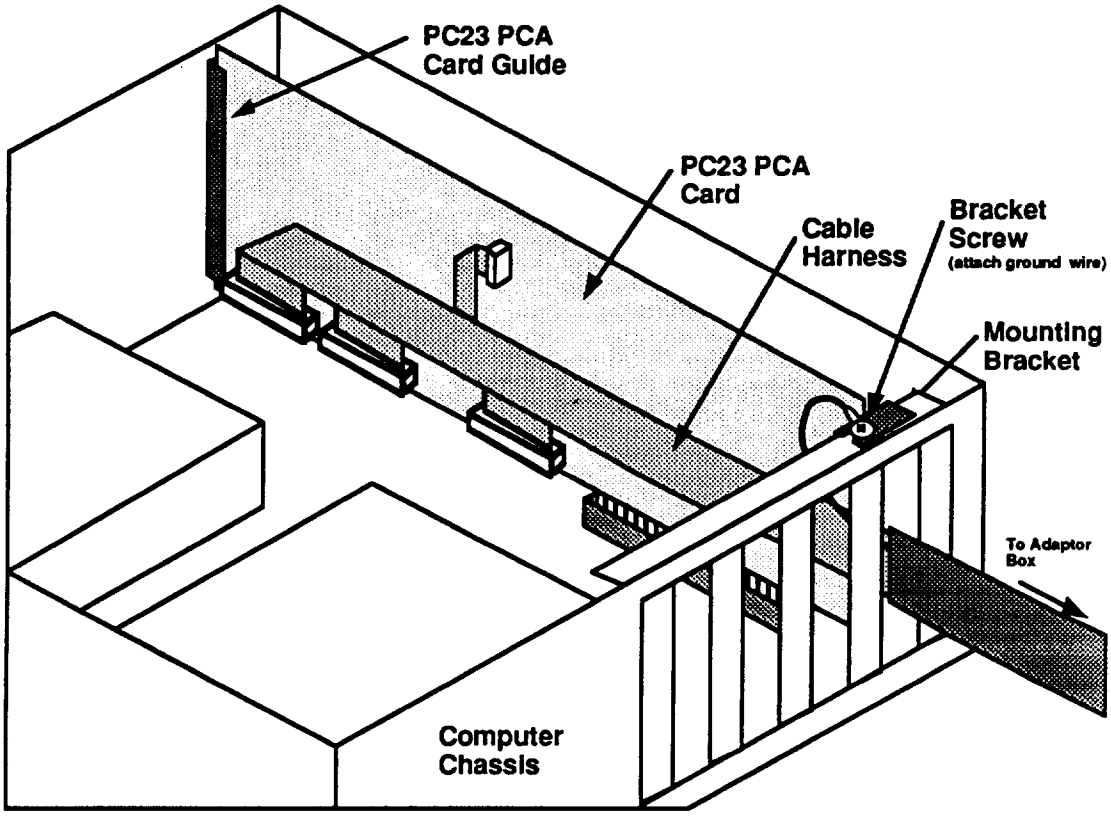
Be sure to align the adaptor connectors properly over the pins on the PCA card connectors. Any misalignment can damage the cables and the PCA card connectors.

⑥ Use the two (#4) machine screws supplied with the PC23 to attach the cable harness clamp to the mounting bracket. Install the bracket clamp so that the two screws fit in its adjustment slots . Squeeze the cable harness tightly against the mounting bracket and tighten the screws.

⑦ Insert the bottom corner of the PC23 board into the PCA card guide slot. Ease the card and the bracket-end simultaneously down into the computer until the PCA card edge connector reaches the mating connector at the bottom. Adjust the PCA card until the edge connectors align and press it down into the mating connector. **Be careful not to disconnect the adaptor box cables.**

⑧ Using the screw that secured the original access slot cover bracket, fasten the PC23 mounting bracket and the adaptor box ground wire to the computer chassis.





④ Connect the Drive

The PC23 is shipped with its step and direction outputs configured as TTL-compatible differential. If your system accepts only single-ended indexer outputs, refer to *Chapter 6, Hardware Reference*, for procedures to change the outputs from differential to single-ended.

The PC23 ship kit includes three pre-wired cables with 25-pin connectors. Although the PC23's differential outputs are compatible with all Compumotor drives, the attached connectors may not be compatible with the drive's connector (e.g., PK130). If you must remove the cable's drive-end 25-pin connector to access the wires, refer to the following table for wire color codes and functions. If custom wiring is called for, use 24 AWG

wire. The maximum recommended cable length is 50 feet. Shield your custom cables with the shield connected to earth ground (pin 5).

Pin #	Wire color	Function
1	Red	STEP +
2	Green	DIRECTION +
5	-	SHIELD
14	Black	STEP -
15	White	DIRECTION -
16	Blue	SHUTDOWN +
17	White/Red	SHUTDOWN -

Use the following procedure to connect the drive to the PC23 adaptor box:

① Set up your drive(s) and connect the motor(s) according to the set-up instructions in your drive user guide.

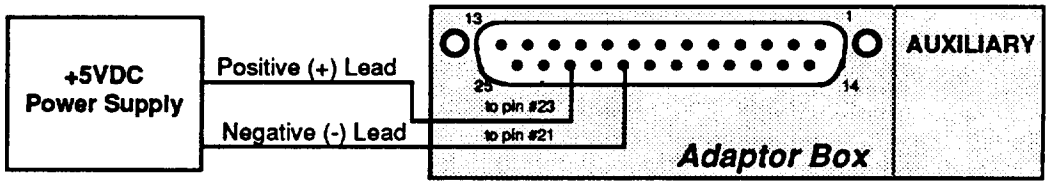
① **Turn off the power to the drive.**

① Connect one end of the drive cable to the axis 1 **MOTOR DRIVER** connector on the PC23 adaptor box. Connect the other end of the cable to the drive. If you are using two or three drives, connect them in the same manner to the axis 2 and axis 3 **MOTOR DRIVER** connectors.

⑤ Connect the +5VDC Power Supply

The output circuits for all adaptor box connectors require an external +5VDC power supply. For the purpose of this bench test, the power supply must have a current capacity of 0.75A. Supply regulation should not exceed 10%.

Connect the power supply to any one of the three adaptor box **AUXILIARY** connectors. Connect the power supply's positive terminal to pin 23, and the negative terminal to pin 21.



⑥ Test the System

The successful completion of the steps in this test will verify that your system connections are correct and that the system operates properly. ***The PC23's default resolution is 25,000 steps/rev (1.0 µs step pulse width). If you are using a full or half-step drive (e.g., PK3), the self test may not work.***

1. Verify that the drive and motor connections are secure.
2. Verify that you have removed jumper JU13 on the indexer card.
3. If the drive you are using is equipped with limits (e.g., Compumotor Plus), disable them for this test.
4. Apply power to the drive(s) in accordance with the drive user guide. The motor(s) must be free to rotate in either direction.
5. Apply power to the computer to initiate the self test.

   If the PC23 detects a problem on power-up, it will turn on the fault LED which is visible through the mounting bracket above the cable harness. If the LED illuminates on power-up, switch off power and refer to *Chapter 7, Troubleshooting.*

   Immediately after applying power, the motor(s) will rotate first in the clockwise (CW) direction, and then in the counter-clockwise (CCW) direction. The distance and rate of rotation depends on the motor's resolution. Standard Compumotor 25,000 step/rev motors will rotate 1 revolution at a rate of 0.2 revs/sec (25,000 steps at 5,000 steps/sec).

After the test move is complete, the fault LED should illuminate. Also, after the test move, the PC23 will not function until you cycle power to the computer.

6. **Turn off power to the computer** and replace jumper JU13 on the PC23 indexer card. This disables the self-test routine.
7. Turn on power to the computer and insert PC23 support diskette #1 into the computer disk drive.
8. At the **A>** prompt, type: **PC23TERM**.
9. Enter the device address (**768**), and answer **Y** (yes) to the reset question.
10. You should now be in the terminal emulation program and able to enter motion commands. Enter the command sequence below. **If you make a mistake while entering the commands, do not backspace. Instead, re-type the command sequence.**

Command	Description
`1LD3`	Disables limits for axis #1
`1A10`	Sets acceleration to 10 rps^2
`1V1`	Sets velocity to 1 rps
`1D25ØØØ`	Sets distance to 25,000 steps (1 rev @ 25,000 steps/rev)
`1G`	Go (execute the move)

The motor should rotate one clockwise revolution and stop.

11. Press the <F10> key to exit the program.
12. After successfully completing the system test, remove power from all components (including the computer) and disconnect the external +5VDC power supply and motor driver from the adaptor box.
13. Proceed to *Chapter 3, Installation.* In Chapter 3, you will configure you system for permanent operation

C H A P T E R ③

Installation

Chapter Objectives

The information in this chapter will enable you to do the following:

- ❑ Mount all system components properly
- ❑ Connect all electrical system inputs and outputs properly
- ❑ Verify (test) system connections and operability

Before you begin this chapter, you must complete all the steps and instructions discussed in Chapter 2.

Installation Precautions

To help ensure personal safety and long life of system components, pay special attention to the following installation precautions:

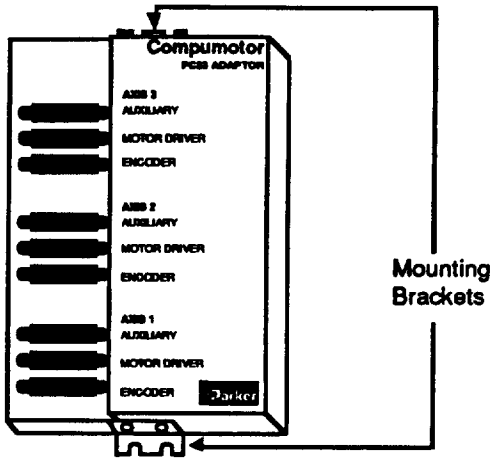
- ❑ Always remove power to the computer before doing the following:
  - Installing or removing the PC23 indexer board
  - Connecting or disconnecting the cable harness
- ❑ Always remove power to the computer and adaptor box before doing the following:
  - Connecting or disconnecting system components
  - Changing the indexer or adaptor box jumpers
- ❑ Make sure the indexer is operated in room temperatures between 32 and 122°F (0 to 50°C) and at a relative humidity between 0 and 90% (non-condensing).

Installation Procedures

This section provides step-by-step procedures for adaptor box connections and mounting, and system testing.

① Mount the Adaptor Box

The adaptor box may be installed wherever it is convenient, within reach of the supplied cable. Attach the two small brackets (supplied in the ship kit) to both ends of the adaptor box for mounting it to a surface (refer to the following figure).



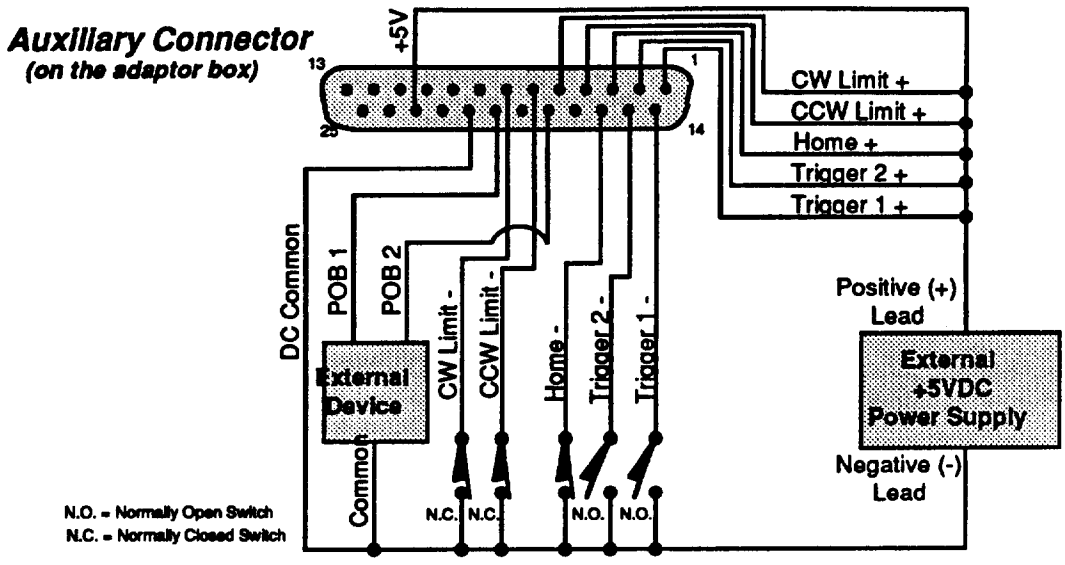
② Connect the Motor Driver

Connect the drive to the adaptor box's **MOTOR DRIVER** connector as you did in Chapter 2.

③ Make Auxiliary Connections

There are three **AUXILIARY** connectors located on the adaptor box, one for each axis. Refer to the table below for auxiliary connector pin assignments. The following figure is a typical I/O wiring diagram for one axis.

Pin #	Axis #1	Axis #2	Axis #3
1	Trigger #1 Input +	Trigger #3 Input +	Trigger #5 Input +
2	Trigger #2 Input +	Trigger #4 Input +	Trigger #6 Input +
3	Home Enable Input +	Home Enable Input +	Home Enable Input +
4	CCW Limit Input +	CCW Limit Input +	CCW Limit Input +
5	CW Limit Input +	CW Limit Input +	CW Limit Input +
6	CCW Limit Return -	CCW Limit Return -	CCW Limit Return -
7	CW Limit Return -	CW Limit Return -	CW Limit Return -
8	*NC*	*NC*	*NC*
9 - 13	*NC*	*NC*	*Joystick Connections*
14	Trigger #1 Input Return -	Trigger #3 Input Return -	Trigger #5 Input Return -
15	Trigger #2 Input Return -	Trigger #4 Input Return -	Trigger #6 Input Return -
16	Home Enable Return -	Home Enable Return -	Home Enable Return -
17	*NC*	*NC*	*NC*
18	Programmable Output #2	Programmable Output #4	Programmable Output #6
19	DC Common	DC Common	DC Common
20	Programmable Output #1	Programmable Output #3	Programmable Output #5
21	DC Common	DC Common	DC Common
22	*NC*	*NC*	*NC*
23	Ext +5VDC Input (user-supplied)	Ext +5VDC Input (user-supplied)	Ext +5VDC Input (user-supplied)
24 & 25	*NC*	*NC*	*NC*



Programmable Output Connections

Each auxiliary connector has two programmable output bits (POBs) (the previous table). Using the o command, you can instruct the PC23 to set these outputs high or low in the course of executing commands. They are used to signal external equipment, or turn process-related devices on and off. These outputs can source 30mA, enough current to drive an optical isolation circuit or solid state relay. Refer to *Chapter 6, Hardware Reference*, for the output circuit configuration diagram.

Electrical noise is usually caused by voltage spikes, not current. Therefore, circuits designed to use these outputs in a current mode (rather than voltage signal mode) are less likely to experience electrical interference.

Input Connections

Auxiliary inputs include six *trigger* inputs, a *home enable* input, and end-of-travel *CW and CCW limit* inputs (refer to the previous table). **When an encoder is not used, the CW, CCW, and home inputs provide the only load position feedback to the PC23.**

Each input has two terminals (+ and -) and is isolated from the other input pairs. Refer to *Chapter 6, Hardware Reference*, for the input circuit configuration diagram.

All inputs are current-driven optical isolators, not likely to suffer false triggering due to electrical interference. Inputs may be directly driven by voltage sources of 3.5V to 10V. *Higher voltages will require the addition of a current-limiting resistor wired in series. The value of this resistor may range from 50Ω to 100Ω per volt above 10V. For example, a 24V source calls for a 700Ω to 1,400Ω resistor wired in series with the input.*

☛Home Enable Input

Use the Home Enable input to establish a *home* position and return to that position when desired. The most common way to use this input is to mount a normally-closed switch at a home reference position. Connect the switch's positive lead to auxiliary connector pin 16, connect the negative lead to the external power supply ground, and connect pin 3 to the positive side of the external power supply. The home input polarity may be reversed with the OSC command.

The home input is used for homing the motor in open-loop (FSBØ) and closed loop (FSB1) operations. In the closed-loop mode, the encoder's Z channel pulse is used in conjunction with the home switch to determine the home position. Both the open loop and closed loop homing functions are described later in this chapter in the System Test section.

CAUTION

Compumotor cannot guarantee proper homing performance with the home and end-of-travel limit inputs tied together.

☛CW & CCW Limit Inputs

The CCW and CW end-of-travel limit inputs serve as safety stops that prevent the load from crashing into mechanical stops and damaging equipment or injuring personnel.

As illustrated in the previous figure, connect pins 4 (CCW) and 5 (CW) to the positive side of the external power supply. Connect the positive side of the normally-closed switches to pins 6 (CCW) and 7 (CW), and connect the negative side of the switches to the external power supply ground.

Mount the switches such that the load forces them to open before it reaches the physical end-of-travel (**leave enough room for the load to stop**). When the load opens the limit switch, the motor comes to a halt. The motor will not be able to move in that same direction until the limit is cleared (switch closes). The actual stopping distance depends on motor speed and the Limit Acceleration (LA) setting.

CAUTION

Make sure the LA setting (deceleration rate) is increased when moving heavy loads or operating at high speeds.

Use the LD command to disable or enable the limit inputs. Use the RA or IS commands to check the status of the limit switches.

☛Trigger Inputs

There are two trigger inputs on each axis (total of six). Connect pin 1 (trigger A) and pin 2 (trigger B) to the positive side of the external power supply. Connect the trigger switch's positive lead to pin 14 (trigger A return) and pin 2 (trigger B return) and connect the switch's negative lead to the external power supply ground.

The trigger switches can be normally-open (not grounded) or normally-closed, depending on the TR command setting. The indexer can be programmed to wait until one or more inputs switch to a desired state (closed or open, 1 or 0).

④ Connect the Encoder (optional)

The 25-pin **ENCODER** connectors mate with the cables of the optional Compumotor encoders available on microstepping motors. These encoders use the same +5VDC external power supply as the rest of the adaptor circuitry.

If you use an encoder other than one supplied by Compumotor, pay special attention to the following requirements:

- It must be an incremental encoder with two-phase quadrature output. An index or *Z channel* output is optional. Outputs may be differential or single-ended.
- It must be a 5V encoder to use the Adaptor power supply. Otherwise, it must be separately powered, with TTL-compatible or open-collector outputs.

- ❑ **The decoded quadrature resolution must be less than the motor resolution by a factor of four to take advantage of the PC23's position tracking capability.**

Refer to the following table for **ENCODER** connector pin assignments. If custom wiring is called for, use 24 AWG wire. The maximum recommended cable length is 25 feet. Encoder cables should be shielded with the shield connected to earth ground (pin 8). Refer to *Chapter 6, Hardware Reference*, for a diagram of the circuit common to all encoder inputs.

Pin #	Function
1	Quadrature Channel A+
2	Quadrature Channel A- (optional)
3	Quadrature Channel B+
4	Quadrature Channel B- (optional)
5	Channel Z+ (index)
6	Channel Z- (optional)
7	*NC*
8	Shield
9 - 13	*NC*
14 - 20	Ground
21 & 22	*NC*
23 - 25	+5VDC (250mA max), user supplied

⑤ Connect the External Power Supply

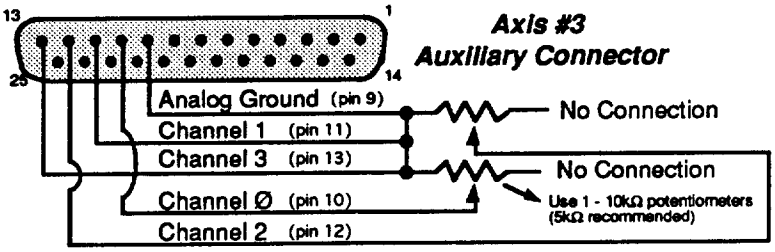
The output circuits for all adaptor box connectors require an external +5VDC power supply. This power supply must have the current capacity to drive the adaptor box circuitry, including any encoders plugged into the **ENCODER** connectors. Supply regulation should not exceed 10%. See the following table for power supply guidelines

Use	Current Requirement
MOTOR DRIVERs only	0.75A
Plus AUXILIARY connections	1.0A
Plus ENCODERs	1.5A to 2.0A

Connect the power supply to any one of the three adaptor box **AUXILIARY** connectors. Connect the power supply's positive terminal to pin 23, and the negative terminal to pin 21 (see example in Chapter 2, *Getting Started*).

⑥ Connect the Joystick (optional)

The PC23 uses an 8-bit analog-to-digital (A/D) converter to accept analog voltage input (generally from a potentiometer joystick) to control motor velocity. Connect the joystick using pins 9 through 13 on the axis #3 **AUXILIARY** connector (see the following figure). *Although you can use any 1kΩ to 10kΩ pot as a joystick reference, Compumotor recommends that you use a 5kΩ potentiometer to ensure good deflection linearity.*



System Test

This section is provided to test for proper system connections and operability. The recommended tests in this section can be performed on all three axes. To conserve space, the command examples are tailored to axis 1. To test the other axes, you must prefix each command with the axis number.

In the bench test configuration, the end-of-travel limit function is disabled in software (factory default), allowing testing without wiring limit switches. If you are now testing the system with a motor connected to a limited-travel mechanism, you must enable the limit function with the **LDØ** command and connect limit switches to protect the mechanism. The motors can be stalled indefinitely without damage to the drive.

Utility Commands

The utility commands below may be useful throughout this section while testing your PC23.

Command	Description
K	Halts the motor immediately. The immediate deceleration may cause a loss of position. **This is a *panic* stop.** This command takes effect immediately after you issue it.
S	Decelerates the motor to a stop using the last defined acceleration value. The system executes this command immediately after you issue it.
LD3	Disables the limit switch functions. It allows motor motion with no limits connected.
^H	Instructs the PC23 to backspace the cursor and delete the last character you entered
PS	Instructs the indexer to pause for a period of time that you specify. Buffered commands that follow the **PS** command are not executed until the indexer receives a Continue (**C**) command to clear the pause and resume execution.

① Enter the Terminal Emulation Mode

This system test uses the same terminal emulation mode used for the bench test you performed in Chapter 2. Complete the following steps to enter the terminal emulation mode:

① Insert PC23 support diskette #1 into the disk drive.

② At the **A>** prompt enter **PC23TERM**.

③ Enter device address **768** and answer **Y** (yes) to the reset question.

The PC23 should now be in terminal emulation mode, ready to accept X language commands. Press **<F10>** at the end of this system test if you wish to exit the program.

② Test CW & CCW Limit Input Connections

Before you verify that the limit switches are working properly, check the following connections:

- ❑ Ensure that the CW and CCW limit switches are wired properly (normally closed switches that open when the load moves to the limit position).
- ❑ Make sure that the load is not attached to the motor.
- ❑ Make sure that you can manually open and close the limit switches.

Use the following procedure to test the limit input switches on axis #1 (repeat for each axis):

① Open both CW and CCW switches.

② Type `1IS`. If all other inputs are opened, the response should be `1:001110`; this means that both CW and CCW limits (represented by the 3rd and 4th digits) are opened.

③ Close the CW limit switch.

④ Type `1IS`. Assuming all other inputs are open, the response to this command should be `1:000110`.

⑤ Close the CCW limit switch.

⑥ Type `1IS`. The response should be `1:000010`.

⑦ To test the CW limit switch, enter the following string of commands:

Command	Description
`LDØ`	Enables CW and CCW limits
`MC`	Sets indexer to continuous mode
`LA10`	When limit is encountered motor decelerates at 10 rps^2
`A5`	Sets acceleration to 5 rps^2
`V2`	Sets velocity to 2 rps
`H+`	Sets motor direction (+ = CW)
`G`	Executes the move (G0)

The motor should move at a constant velocity.

⑧ Open the CW limit switch. The motor should come to a stop at 10 rps^2.

⑨ To test the CCW limit switch, enter the following string of commands:

Command	Description
`MC`	Sets indexer to Continuous mode
`A1`	Sets acceleration to 1 rps^2
`V1`	Sets velocity to 1 rps
`H-`	Sets move to the CCW direction
`G`	Executes the move (Go)

The motor should move at a constant velocity.

⑩ Open the CCW limit switch. The motor should come to a stop at 10 rps^2.

If the motor continues to move, open the CW limit switch. If the motor stops when you open the CW limit switch, swap the CW and CCW limit input wires.

If neither of these limit switches stop the motor, recheck your switch wiring and refer to Chapter 7, *Troubleshooting*.

③ Test Trigger Input Connections

The six trigger inputs are typically connected to sensors or other signal sources to provide the indexer with information on external conditions. The indexer can be programmed to wait until one or more inputs switch to a desired state (on or off, 1 or 0). The `TR` command causes a pause in the execution of buffered commands if one or more trigger inputs are not in the state specified by the command. The `TS` command reports the state of all six trigger inputs.

Use the following steps to verify that you have wired the trigger inputs properly:

① Type `1TS`. If all the trigger inputs are open (not grounded), the response should be `1:000000`.

② Close the TRIG 1 input switch.

③ Type `1TS`. The response should be `1:100000`. This verifies that the TRIG 1 input is closed.

Repeat steps 1 through 3 for each trigger input.

④ Enter the following commands:

Command	Description
MN	Sets indexer to normal mode
A2	Sets acceleration to 2 rps^2
V2	Sets velocity to 2 rps
D25ØØØ	Sets distance to 25,000 steps
TR1XXXXX	Wait for Trigger Input 1 to be grounded
G	Executes the move (Go)

⑤ Ground trigger input 1 (motor will move).

④ Test POB Connections

The PC23 is equipped with six programmable output bits (two per axis). You may use them to signal a peripheral device that the PC23 has just completed some event. These outputs are controlled with the O Command.

Perform the following steps to verify that you have wired the outputs properly:

① Type `O111111`

The response from this command should be 5VDC, as measured between the POB (pin 18 or 20) and the external power supply ground.

② You may change the reading between the POB (pin 18 or 20) and the external power supply ground to logic low (ØVDC) by entering the following command: `OØØØØØØ`

⑤ Test Encoder Resolution

The PC23's encoder feedback functions will not operate properly if the encoder resolution or signal polarity are incorrect.

The Encoder Resolution (ER) command defines the number of positions (steps) received by the drive per revolution of motor movement. If you have an encoder, refer to the operator's manual for the resolution of your encoder. Use the following example as a guide for setting your encoder resolution:

Command	Description
ER4000	Tells the indexer that the encoder has a resolution of 4,000 steps/rev after quadrature (1,000 lines)

You may test the encoder resolution setting with the following procedure. The objective is to move the motor with open-loop moves and verify that the number of feedback encoder counts meet with your expectations.

① Issue the 1FR status request command to verify that all encoder functions are off for the test axis (axis #1). The response should be `1:00000000`. If the response is not `1:00000000`, refer to the FR and FS commands in *Chapter 5, Software Reference.*

② Use the following commands to make the motor move 1 revolution and set the absolute position counter to zero (example assumes use of a 25,000 step/rev motor):

Command	Description
MN	Sets mode to normal
FSBØ	Sets indexer to motor step mode
A1Ø	Sets acceleration to 10 rps^2
V2	Sets velocity to 2 rps
D25ØØØ	Sets distance to 25,000 steps
PZ	Sets absolute position counter to zero
G	Executes the move (Go)

③ Report the encoder position with the following command:

Command	Description
1PX	Reports Encoder Position

④ Issue the RSBØ command and repeat steps 1 through 3 to ensure that the backlash or motor windup are not interfering with the test. Issuing the encoder position report (PX) command should yield a number very close to the parameter that you specified with the encoder resolution (ER) command.

If the encoder position report is negative, encoder channels A and B are reversed. You can correct this by swapping encoder channel A+ with B+ and A- with B- (reversing encoder direction). ***Be sure to turn power off before reversing any wires.***

⑥ Test the Homing Function

This section provides procedures to test the home switch and the homing function for both open-loop and closed-loop operation. *Open-loop* homing uses only the home switch to establish the home reference position. *Closed-loop* homing uses the home switch and the Z (index) channel from an incremental encoder to establish the home reference position.

Test the Home Switch

Before testing the homing function, you should use the following procedure to test whether you have connected the home switch properly:

① Assuming the switch is normally-closed, manually open the switch and type 1IS. Assuming your end-of-travel limit switches are closed and all other inputs are open (not grounded), the response will be 1:000010. The 5th digit in the response represents the home input status.

② Close the home switch and type 1IS. The response should be 1:000000. This verifies that the switch is connected properly.

Go Home Process

① To initiate the homing function, you must issue the Go Home (GH) command. When you issue the Go Home command, you must include the direction and velocity that the motor should use to search for home. You may also specify the Go Home acceleration with the GA command.

② After you issue the GH command, the motor begins to move in the direction and at the velocity and acceleration that you specified. The PC23 waits for the home limit input to go active (usually activated by the load opening a normally-closed home switch). If the motor trips an end-of-travel limit switch while it searches for home, it will reverse direction and look for the home limit input to go active. If the motor trips the other limit switch before it detects the home signal, the Go Home move will be aborted and the motor will stop.

③ While the motor (and load) is moving, the PC23 recognizes the *home position* as the position where the home limit signal makes a transition from ON to OFF, or from OFF to ON, depending on if the osc setting is *active high* or *active low* (default is *active high*, home = transition from ON to OFF).

④ The motor decelerates to a stop.

⑤ The motor changes direction, passes the home position, and stops 1/32 of a revolution on the opposite side of the home position.

⑥ The PC23 then changes motor direction again and creeps the motor at 0.1 rps until it reaches the home position (open-loop) or until it receives the encoder's Z Channel pulse (closed-loop).

⑦ When the homing operation is complete, the indexer automatically resets its internal absolute position counter for that axis to zero (same as issuing PZ).

You can verify if the homing process was successful by issuing the Go Home Status (RC) command.

You must ensure that the final approach (step 4 above) is in the same direction you wish the motor to move after the homing function is complete. If the desired post-homing direction is CW, the 0.1 rps creep portion of the go home move must start from the CCW side of home. If there is significant *backlash* in the system, and the indexer is instructed to go home in the CW direction, the motor may end up on the wrong side of the home signal and execute its final approach in the wrong direction.

There are two situations in which the PC23 can lose track of the *home* position. They are (1), *overshoot* resulting from a Go Home velocity that is too high, and (2), a mechanical or electrical home switch signal delay. In such situations, you should reduce the Go Home velocity.

Test Open-Loop Homing

Use the following procedure to test the open-loop homing function:

① Issue the following commands:

Command	Description
`LDØ`	Enable the CW and CCW limits
`GA5`	Sets go home acceleration to 5 rps^2
`GH+2`	Instructs the motor to go home (CW) at 2 rps

The motor should move in the CW direction at a constant velocity of 2 rps. When the load opens the home switch (***If the motor is not connected to the load, you must manually open and close the switch***), the motor decelerates to a stop. The motor changes direction, travels CCW past the home position, and stops. The motor changes direction again and creeps CW at 0.1 rps and stops at the home position (***or when you manually open and close the home switch again***).

② Issue the `1PR` command to verify that the homing is complete and that the absolution position counter is reset to zero. The response should be `1:+00000000`.

③ Issue the `1RC` command to verify that the homing was successful. The response should be `1:*@`.

Test Closed-Loop Homing

The closed-loop homing function uses the encoder's Z channel and the Home switch for positioning. The Z channel is output once per revolution of the rotary encoder. *Home* is established when the load opens the normally-closed Home Enable switch, and while the switch is open the encoder outputs its Z Channel pulse. You can adjust them to coincide by (1), decoupling the encoder from the motor and turning the encoder until they coincide, or (2), moving the home switch until they coincide.

CAUTION

If the Home input is active for more than one encoder revolution, multiple home definitions will result and the repeatability of the homing function will be lost. When no Home Enable switch is needed, this input should be left unconnected.

You can perform closed-loop homing only if the PC23 is in the encoder step mode (set with the `FSB1` command).

You can change the active states (active high or low) of the home input and the Z channel with the `OSC` and `OSD` commands respectively (see the command descriptions in Chapter 5, *Software Reference*).

Use the following procedure to test the closed-loop homing function:

① Issue the following commands:

Command	Description
FSB1	Enable encoder step mode
LDØ	Enable the CW and CCW limits
GA5	Sets go home acceleration to 5 rps^2
GH+2	Instructs the motor to go home (CW) at 2 rps

The motor should move in the CW direction at a constant velocity of 2 rps. When the load opens the home switch (***If the motor is not connected to the load, you must manually open the switch***), the motor decelerates to a stop. The motor changes direction, travels CCW past the home position, and stops. The motor changes direction again and creeps CW at 0.1 rps and stops when the Z channel pulse from the encoder is received while the home switch is active (open).

② Issue the 1PX and the 1PR commands to verify that the homing is complete and that the absolution position counter is reset to zero. The responses to both commands should be 1:+00000000.

③ Issue the 1RC command to verify that the homing was successful. The response should be 1:*Ø.

⑦ Adjust the Joystick

Use the following procedure to adjust the joystick for proper operation:

① Issue the AV1 and the AV3 commands. Both channel voltage level responses should be between 0V (CH1:0.00V) and 0.005V (CH1:0.005V).

② Issue the AVØ and AV2 commands. Both channel voltage level responses should be greater than the responses from the AV1 and AV3 commands.

If the AV command responses do not provide the specified voltage values, check the joystick connections and the hardware configuration.

③ Issue the following command loop sequence:

Command	Description/Response
L	Begin loop
AVØ	Report analog voltage on channel Ø (the response should be CHØ:1.25V)
AV2	Report analog voltage on channel 2 (the response should be CH2:1.25V)
T1	Wait 1 second
N	End loop

The AVØ and AV2 responses should appear on the CRT at 1 second update intervals.

④ While the AV command responses are being updated on the computer CRT, mechanically adjust the potentiometer on the joystick assembly (**not the joystick itself**) until the responses indicate a voltage level of 1.25V (±0.25V). Issue the S command when you have completed the adjustment.

⑤ Issue the 1JZ and 2JZ commands.

C H A P T E R ④

Application Design

Chapter Objectives

The information in this chapter will enable you to the following:

- ❑ Recognize and understand important considerations that must be addressed before you implement your application
- ❑ Understand the capabilities of the system
- ❑ Customize the system to meet your requirements

The X language command examples in this chapter are tailored to single-axis operation. To use these examples for multi-axis simulation, you must change the axis-specific prefix for each command (1, 2, or 3). If you do not specify an axis, the PC23 defaults to axis #1.

Application Considerations

Successful application of a rotary indexer system requires careful consideration of the following important factors:

- ❑ Mechanical resonance
- ❑ Ringing or overshoot
- ❑ Move times (calculated vs. actual)
- ❑ Positional accuracy and repeatability

Mechanical Resonance

Resonance, a characteristic of all stepper motors, can cause the motor to stall at low speeds. Resonance occurs at speeds which approach the natural frequency of the motor's rotor and the first and second harmonics of those speeds. It causes the motor to vibrate at these speeds. The speed at which fundamental resonance occurs is typically between 0.3 and 0.8 revs per second and is highest for small motors and lowest for large motors.

Most full-step motor controllers *jump* the motor to a set minimum starting speed to avoid this resonance region. This causes poor performance below one rev per second. In nearly all cases, using a micro-stepping drive with the PC23 will overcome these problems.

Motors that will not accelerate past one rev per second may be stalling due to resonance. The resonance point may be lowered to some extent by adding inertia to the motor shaft. This may be accomplished by putting a drill chuck on the back shaft. *This technique is applicable only to double-*

shaft motors with the shaft extending from both ends of the motor. In extreme cases, you may also need a viscous damper to balance the load. One of the manufacturers of viscous dampers is listed below:

Ferrofluidics Corporation
40 Simon Street
Nashua, NH 03061
(603) 883-9800

Changing the velocity (V command) and acceleration (A command) may also help a resonance problem.

Ringing or Overshoot

The motor's springiness, along with its mass, form an underdamped resonant system that *rings* in response to acceleration transients (such as at the end of a move). Ringing at the end of a move prolongs settling time. *Overshoot* occurs when the motor rotates beyond the actual final position. The actual settling time of a system depends on the motor's stiffness, the mass of the load, and any frictional forces that may be present. By adding a little friction, you can decrease the motor's settling time.

Move Times: Calculated vs. Actual

You can calculate the time required to complete a move by using the acceleration, velocity, and distance values that you define with the A, V, and D software commands respectively. However, you should not assume that the values that you use will constitute the actual move time. After you issue the Go (G) command, the PC23 may take up to 50 ms to calculate the move before the motor starts moving. You should also expect some time to elapse for the motor and the load to settle.

Positional Accuracy vs. Repeatability

In positioning systems, some applications require high absolute accuracy. Others require repeatability. You should clearly define and distinguish these two concepts when you address the issue of system performance.

If the positioning system is taken to a fixed place and the coordinates of that point are recorded, the only concern is how well the system repeats when you command it to go back to the same point. For many systems, what is meant by accuracy is really repeatability. Repeatability measures how accurately you can repeat moves to the same position.

Accuracy, on the other hand, is the error in finding a random position. For example, suppose the job is to measure the size of an object. The size of the object is determined by moving the positioning system to a point on the object and using the move distance required to get there as the measurement value. In this situation, basic system accuracy is important. The system accuracy must be better than the tolerance on the measurement that is desired.

For more information on accuracy and repeatability, consult the technical data section of the *Compumotor Programmable Motion Control Catalog.*

Open Loop Accuracy

Open-loop absolute accuracy of a step motor is typically less than a precision-grade system, but is better than most tangential drive systems. When you *close the loop* with an incremental encoder, the accuracy of these systems is equivalent to the encoder's accuracy.

The worst-case accuracy of the system is the sum of the following errors. Accuracy = A + B.

A Uni-directional Repeatability: The error measured by repeated moves to the same point from different distances in the same direction.

B Hysteresis: The backlash of the motor and mechanical linkage when it changes direction due to magnetic and mechanical friction.

Closed-Loop Accuracy

Closed-loop accuracy is determined by the resolution of the encoder. The PC23 can position the motor within the encoder's specified dead band. Typically, this means the motor will be positioned to within one encoder step. To do this satisfactorily, the encoder must have a lower resolution than the motor. If the step size of the motor is equal to or greater than the step size of the encoder, the motor will be unable to maintain the position and may become unstable. In a system with adequate motor-to-encoder resolution (4:1), the motor is able to maintain accuracy within one encoder step. Refer to Selecting Encoder Resolution Values discussed later in this chapter.

Using the Terminal Emulation Mode

Compumotor recommends that you familiarize yourself with the PC23's X language commands in the *Terminal Emulation Mode* before you create your custom programs in another language. The terminal emulation mode is a user-friendly menu-driven program that allows you to directly use the X Language commands. After you are familiar with the X language, refer to the Programming section in this chapter to develop custom program routines that use X language sequences you have already tested in the terminal emulation mode.

To enter the terminal emulation mode, complete the following procedure:

① Insert support disk #1 into the computer disk drive .
② Type in `PC23TERM` at the `A>` prompt.
③ Enter device address `768`.
④ Answer `Y` (yes) to the reset question.

Positioning Modes

You can use one of three positioning modes to manipulate the motor with X language commands. The three modes are *normal* (preset), *alternating*, and *continuous*.

Normal (Preset) Mode

You can select preset moves by putting the PC23 into normal mode using the Mode Normal (`MN`) command. Preset moves allow you to position the motor in relation to the motor's previous stopped position (incremental moves) or in relation to a defined zero reference position (absolute moves). You can select incremental moves by using the Mode Position Incremental (`MPI` or `FSAØ`) command. You can select absolute moves using the Mode Position Absolute (`MPA` or `FSA1`) command.

Incremental Mode Moves

When using the Incremental mode (`MPI` or `FSAØ`), a preset move moves the shaft of the motor the specified distance from its starting position. For example, to move the motor shaft 1.5 revolutions, a preset move with a distance of +37,500 steps (1.5 revs @ 25,000 steps/rev) would be specified. Every time the indexer executes this move, the motor moves 1.5 revs from its resting position. You can specify the direction of the move by using the optional sign (`D+375ØØ` or `D-375ØØ`), or define it separately with the Set Direction (`H`) command (`H+` or `H-`) after the `D` command. Whenever you do not specify the direction (e.g., `D25ØØØ`), the unit defaults to the positive (CW) direction.

☛Sample Incremental Mode Moves

The moves shown below are incremental moves. The distance specified is relative to the motor's current position. **This is the default (power-up) positioning mode.**

Command	Description
MPI	Sets unit to Incremental Position Mode
A2	Sets acceleration to 2 rps^2
V5	Sets velocity to 5 rps
D25ØØØ	Sets distance to 25,000 steps
G	Executes the move (Go)
G	Repeats the move (Go)
H	Reverses direction of next move
G	Executes the move (Go)

The motor moves one CW revolution and stops. It then moves one more CW revolution in the same direction and stops. The motor changes direction and moves one CCW revolution.

Command	Description
D-25ØØØ	Sets the distance to 25,000 steps in the negative (CCW) direction.
G	Executes the move (Go)

The motor returns to its original starting position.

Command	Description
H	Toggles the motor direction of the next move, but maintains existing acceleration, velocity, and distance parameters.
G	Executes the same move profile as the previous move, but in the opposite direction (Go)

The motor moves 25,000 steps in the positive (CW) direction.

If you wish to load all the commands before executing them, you may use the Pause (PS) and Continue (C) commands.

Command	Description
PS	Pauses execution until the indexer receives a Continue (C) command
G	Executes a 25,000-step move (Go)
T3	Waits 3 seconds after the move
G	Executes another 25,000-step move (Go)
C	Cancels pause and executes the G T3 G commands

Absolute Preset Mode Moves

A preset move in the Absolute mode moves the motor the distance that you specify (in motor steps) from the *absolute zero position*. You can set the absolute position to zero with the Position Zero (PZ) command, issuing the Go Home (GH) command, or by cycling the power to the indexer.

The direction of an absolute preset move depends upon the motor position at the beginning of the move and the position you command it to move to. For example, if the motor is at absolute position +12,500, and you instruct the motor to move to position +5,000, the motor will move in the negative direction a distance of 7,500 steps to reach the absolute position of +5,000.

When you issue the Mode Position Absolute (MPA or FSA1) command, it sets the mode to absolute. When you issue the Mode Position incremental (MPI or FSAØ) command the unit switches to Incremental mode. The PC23 retains the absolute position, even while the unit is in the incremental mode. You can use the Position Report (PR) command to read the absolute position.

☛Sample Absolute Moves

The moves shown below are absolute mode moves. The distance specified is relative to the PC23's absolute zero position.

Command	Description
MN	Sets the PC23 to the normal mode
MPA	Sets the PC23 to the absolute positioning mode
PZ	Sets the current absolute position to zero
A5	Sets acceleration to 5 rps^2
V3	Sets velocity to 3 rps
D5ØØØ	Sets move to absolute position 5,000
G	Executes move (motor moves to absolute position 5,000)

Command	Description
D1ØØØØ	Sets the motor to absolute position 10,000. (Since the motor was already at position 5,000, it will move 5,000 additional steps in the CW direction.)
G	Executes the move (Go)
DØ	Sets the motor to absolute position 0. (Since the motor is at absolute position 10,000, the motor will move 10,000 steps in the CCW direction.)
G	Executes the move (Go)

Alternating Mode

You can select the Alternating mode with the MA command. Set-up for this mode is identical to that for Normal Mode (MN). The difference is that when you issue the Go (G) command, the motor shaft rotates to the commanded position that corresponds to the value set by the D command, and then retraces its path back to the start position. The shaft continues to move between the start position and the command position. Normally, the motor stops immediately when you issue an S command. However, if you issue the SSD1 command before the G command, then when you issue the S command, the motor completes the cycle and stops at the start position.

Another Go (G) command will repeat the same motion pattern.

This indexing mode is not functional when the absolute positioning mode is selected (MPA or FSA1).

Continuous Mode

The Continuous Mode (MC) is useful for applications that require constant movement of the load, when the motor must stop after a period of time has elapsed (rather than after a fixed distance), or when the motor must be synchronized to external events such as trigger input signals. You can manipulate the motor movement with either buffered or immediate commands. After you issue the G command , buffered commands are executed in the order in which they were programmed. While the motor is in continuous motion, you can change the velocity and acceleration by issuing a new V and A command followed by a G command.

☛Sample Continuous Mode Move

The following example demonstrates a continuous mode sequence.

Command	Description
MC	Sets mode to continuous
A1Ø	Sets acceleration to 10 rps^2
V1	Sets velocity to 1 rps
G	Executes the move (Go)

The motor accelerates to 1 rps and continues at that same rate until you issue the S command, or until the load triggers an end-of-travel limit switch.

The following example demonstrates how to change the acceleration and velocity while the motor is moving (*on-the-fly*) in continuous mode.

Command	Description
MC	Sets all moves to the Continuous mode
A1	Sets acceleration to 1 rps^2
V.5	Sets final velocity to 0.5 rps
G	Executes move (motor travels at 1 rps)
A10	Sets acceleration to 10 rps^2
V.4	Sets velocity to 0.4 rps
G	Changes velocity to 0.4 rps at 10 rps^2
V.1	Sets velocity to 0.1 rps
G	Changes velocity to 0.1 rps

Move Parameters

For rotary motors, velocity and acceleration parameters are expressed in units of motor revolutions. For accurate speed control, each axis needs to

know its motor's resolution. Use the MR command to change motor resolution from the default (25,000 steps/rev) as required. This parameter also controls step pulse width and velocity range. Refer to the description of the MR command in Chapter 5, *Software Reference.*

It is desirable in many applications for the distance parameter to be specified in units other than motor steps, such as hundredths of a degree or thousandths of an inch. If the desired units correspond to an integer number of motor steps, the indexer's position multiplier (US command) capability can used to scale the D command position parameter to those units for any axis.

Example

A 25,000-step/rev motor drives a 5-turn/inch leadscrew. The system resolution is 125,000 steps per inch. To express the distance parameter in thousandths of an inch, the operator uses the command US125 to set a distance multiplier of 125. If he then gives a distance command D1ØØØ, the motor will move 1,000 units of 125 steps each (one inch) on the next Go command. For more information, refer to the US command description in Chapter 5, *Software Reference.*

When scale factors greater than 1 are used, and an indexed move is interrupted by a Stop (S) command or by activation of a limit switch, the final position may not be a multiple of the scale factor. Consequently, the repeatability of that move is lost.

Closed-loop (Encoder-based) Operation

The PC23 may be operated in either the *open-loop* mode or the *closed-loop* mode.

Open-loop refers to operating without position feedback from an encoder. The move examples in the previous section, Positioning Modes, are examples of open-loop moves. This is the PC23's default operating mode. The Motor Step Mode (FSBØ) command sets the PC23 to the open-loop mode.

Closed-loop refers to operating with position feedback from an encoder. This section discusses how you can use an incremental encoder to perform closed-loop functions with the PC23. If the move examples below do not function as described, refer to Chapter 7, *Maintenance & Troubleshooting.*

Encoder Compatibility

The PC23 is designed to interface with an incremental rotary or linear encoder. Incremental encoders with quadrature (single-ended or differential) TTL square wave outputs may also be used. To implement the closed-loop functions, you must connect an encoder to the adaptor box as discussed in Chapter 3, Installation. When you use encoders with single-ended outputs, you must change the adaptor box jumpers as instructed in Chapter 6, *Hardware Reference.*

The functions that are added to a system when an encoder is used are listed below:

- Encoder-referenced positioning
- Encoder position servoing (*position maintenance*)
- Motor stall detection
- Higher accuracy homing
- Multi-axis stop on stall

Selecting Encoder Resolution Values

The number of encoder steps that the PC23 system recognizes is equal to four times the number of encoder *lines*. For example, a 1000-line encoder mounted directly on the motor will generate 4000 encoder steps per revolution of the motor shaft. A minimum of three motor steps per encoder step is required for successful operation of the position maintenance function. Position resolution (accuracy) is determined by the encoder resolution.

If the encoder is mounted on the load and there is gearing between the motor and the load, then the gear ratio must be considered when selecting the encoder resolution.

For example, using a 12,800 steps/rev motor, a 1,000-line encoder, and a 10:1 reducer, the ratio of motor revolutions to encoder steps would be changed as described in the table below.

Parameter	w/o Reducer 1:1 Ratio	w/ Reducer 10:1 Ratio
Number of encoder steps recognized by the PC23 (per motor rev)	4,000	400
Motor-to-encoder step ratio	3.2/1	32/1

Setting Encoder Resolution

Since there are many different encoders with different resolutions, you must specify for the PC23 what type of encoder you have connected to the system. Using the following example as a guide, you can specify the encoder's resolution to the indexer with the Encoder Resolution (ER) command.

Command	Description
ER4000	Tells the indexer that the encoder has a resolution of 4,000 steps/rev after quadrature (1,000 lines)

Encoder Step Mode

The indexer can perform moves in either motor steps or encoder steps. In Motor Step mode (FSBØ), the distance command (D) defines moves in motor steps. In Encoder Step mode (FSB1), the distance command defines moves in encoder steps.

The sample move below assumes the use of an incremental encoder with an motor-to-encoder step ratio of 4:1.

Command	Description
MN	Sets mode to normal
ER4ØØØ	Sets up encoder where 4,000 encoder steps (1,000 lines) are produced per 1 revolution of the motor.
FSB1	Sets move to encoder step mode
A1Ø	Set acceleration to 10 rps^2
V5	Set velocity to 5 rps
D4ØØØ	Set distance to 4,000 encoder steps
G	Executes the move (Go)

The motor will turn in the CW direction until 4,000 encoder steps (1 revolution) are received.

Position Maintenance

The Position Maintenance (FSC) command enables and disables the position maintenance function. You must enable position maintenance with the FSC1 command to activate closed loop servoing and ensure that encoder step moves are positioned to the exact encoder step commanded. You must have an encoder connected, and the indexer set in Encoder Step mode (FSB1) in order to enable position maintenance.

Enabling position maintenance causes the indexer to servo the motor until the correct encoder position is achieved. This occurs at the end of a move (if the final position is incorrect) or any time the indexer senses a

change in position while the motor is at zero velocity. The correction response calculation yields a correction velocity according to the gain parameter set with the Correction Gain (CG) command (refer to Chapter 5, *Software Reference* for description).

The sample move below assumes the use of an encoder with a motor-to-encoder step ratio of 4:1.

Command	Description
FSB1	Sets move to encoder step mode
FSC1	Enables position maintenance
MN	Sets indexer to normal mode
A1Ø	Sets acceleration to 10 rps^2
V5	Set velocity to 5 rps
D4ØØØ	Set distance to 4,000 encoder steps (1 rev)
G	Executes the move (Go)

The motor will turn in the CW direction until 4,000 encoder pulses (1 revolution) are received. The position maintenance function instructs the PC23 to servo the motor until the correct encoder position is achieved.

Stop-On-Stall

You can enable the Stop-on-Stall function with the FSD1 command. The move will terminate, without any delay, as soon as a stall is detected. This function works either in Motor Step or Encoder Step mode.

> ***CAUTION***
>
> **Disabling the Stop-on-Stall function with the FSDØ command will allow the PC23 to finish the move regardless of a stall detection, even if the load is jammed. This can potentially damage user equipment.**

The FSD1 command is valid only if the Enable Stall Detection (OSE1) command has been issued.

Example Set-up Commands

Command	Description
ER4ØØØ	Sets encoder resolution to 4,000 steps/rev
OSE1	Enables stall detect
FSD1	Enables stop on stall

Stall detection does not occur until the error exceeds the steps required to travel from one electrical pole of the motor to the next. In a 25,000 step/rev motor/drive system, this corresponds to 500 steps (25,000 ÷ 50 poles). In a 400 step/rev motor/drive system, this corresponds to 8 steps (400 ÷ 50 poles).

Therefore, if the above set-up commands are used and the PC23 and motor/drive resolution is 25,000 steps/rev (MR10 command = 25,000 steps/rev indexer resolution), then if the PC23 does not receive an encoder pulse after moving the motor 500 steps, the PC23 will detect a stall and stop the motor immediately.

Output-On-Stall

You can select the output-on-stall function with the Turn on Output #6 on Stall Detect (FSE) command. This is useful for signaling other components in your system that a stall has occurred.

If you enter the FSE1 command, the PC23 programmable Output #6 goes on (current flows) when a stall is detected and remains on until a new move begins. This command is valid only if the Stall Detection has been enabled (OSE1).

Example/Test

① Disconnect the encoder connector from the adaptor box.

② Issue the following commands:

Command	Description
MN	Sets the system in the preset mode
FSB1	Enables encoder mode
OSE1	Enables stall detect function
FSD1	Stops motor if a stall is detected
FSE1	Turns on Output 6 if stall is detected
A2	Sets acceleration to 2 rps^2
V.1	Sets velocity to 0.1 rps
D8000	Sets distance to 8,000 encoder steps
G	Execute the move (Go)

③ Since you disconnected the encoder, no encoder pulses are reaching the PC23. Consequently, after the motor moves 500 motor steps, the PC23 detects a *stall*, turns on Output #6, and stops the motor.

Multi-Axis Stop-on-Stall

On a multi-axis PC23 system, you may wish to have all axes stop motion if a stall is detected on any axis. You can select this function via the Kill Motion on Trigger (FSF) command. When selected with the FSF1 command, a signal on the Trigger #6 input terminates the move immediately, thus functioning as a remote stop input.

Perform the following steps to set up your PC23 system to terminate multi-axis moves when a stall is detected on any axis:

① Issue the FSE1 command for each axis to turn on output #6 if it detects a stall (e.g., 1FSE1 2FSE1 3FSE1).

② Issue the FSF1 command for each axis to kill motion upon activation of the trigger #6 input (e.g., 1FSF1 2FSF1 3FSF1).

③ On the axis 3 auxiliary connector, connect Output #6 (pin 18) to Trigger #6 (pin 2).

Program Control

This section discusses the X language program control features of the PC23.

Loops

You may use the Loop (L) command to repeat certain sequences. You can nest loop commands up to 8 levels deep.

Command	Description
PS	Pauses command execution until the indexer receives a Continue (C) command
MPI	Sets mode to incremental
A5	Sets acceleration to 5 rps^2
V5	Sets velocity to 5 rps
L5	Loops 5 times
D2000	Sets distance to 2,000 steps
G	Executes the move (Go)
T2	Delays 2 seconds after the move
N	Ends Loop structure
C	Initiates command execution to resume

The example below shows how you can nest a small loop inside a major loop. In this example, the motor makes 2 moves and returns a carriage return. The unit repeats these procedures and will continue to repeat until you instruct the unit to stop immediately with an S(Stop) or K (Kill) command. If you issue a Y (Stop Loop) command, the PC23 finishes the current loop of commands and then stops the motor.

Command	Description
PS	Pauses command execution until the indexer receives a Continue (C) command
L	Loops indefinitely
1CR	Sends a carriage return
L2	Loops twice
G	Executes 2,000-step move
T.5	Waits 0.5 seconds
N	Ends loop
N	Ends loop
C	Cancels pause and initiates command execution

Loop

Loop

POBs (Programmable Output Bits)

You can turn the programmable outputs on and off using the Output (O)command. Zero (Ø) turns off a given output and one (1) turns the output on. The outputs conduct when they are on and do not conduct when they are off (see the O command description in Chapter 5).

Command	Description
MN	Set move to Mode Normal
PS	Pauses execution until indexer receives a Continue (C) command
A1Ø	Sets acceleration to 10 rps^2
V5	Sets velocity to 5 rps
D25ØØØ	Sets move distance to 25,000 steps
O1ØXXXX	Sets POB 1 on and POB 2 off and leaves the rest unchanged
G	Executes the move (Go)
C	Cancels the Pause and executes the move

Move Completion Signals

When you complete a move, you may use the PC23's programming capability to signal the end of the current move. In a preset move, you may use one of the following commands:

- Carriage return (CR)
- Output command (O)
- Status request commands such as PX(reports encoder's absolute position), PR(reports motor's absolute position), FR (reports which encoder functions are enabled), etc.

You may also use the moving/stopped bits of the status byte (SB) register to determine if an axis has completed its move.

Command	Description
PS	Pauses command execution until the indexer receives a Continue (C) command
A2	Sets acceleration to 2 rps^2
V.1	Sets velocity to 0.1 rps
D125ØØ	Sets distance to 12,500 steps
G	Executes the move (Go)
1CR	Sends a carriage return over the interface
OXX1XXX	Turns on Output #3 (ignores the other POBs)
1PR	Report absolute position
1PX	Report absolute encoder position
C	Cancels the Pause and executes the move

The motor moves 12,500 steps. When the move is complete, the PC23 issues a carriage return, turns on POB #3, reports the motor's absolute position, and reports the encoder's absolute position.

Custom Profiles

You can define a custom non-linear motion profile with the PC23's velocity streaming modes. The three velocity streaming modes that the PC23 offers are listed below.

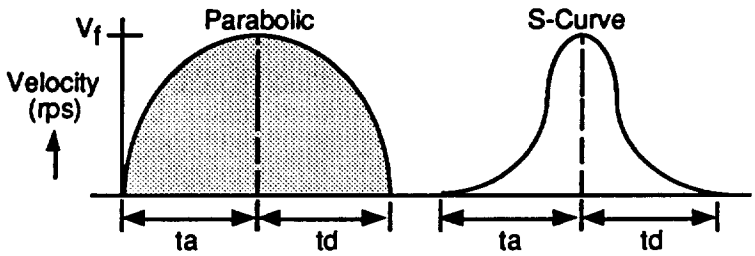
- Immediate Velocity (Q1 command)
- Time-Distance (Q2 command)
- Time-Velocity (Q3 command)

When you select the Velocity Streaming mode (Q1), the PC23 makes immediate velocity changes based on parameters you set with Set Immediate Velocity (RM) commands. The timing between issuing each RM command determines the exact move profile (refer to the Immediate Velocity Streaming section latter in this chapter).

The Time-Distance (Q2) mode allows you to specify an update period at which the PC23 travels the next distance that you specify.

The Time-Velocity (Q3) mode is very similar to the Time-Distance (Q2) mode, except that the variable parameter is the velocity.

By using these modes, you can define any motion profile. Examples are shown the the following figure. You can use a parabolic motion profile to optimize the motor's available running torque. An S-Curve profile provides smoother handling of sensitive loads by eliminating sudden acceleration changes



For more information on custom profiling modes, refer to the RM, Q2, and Q3 commands in Chapter 5, *Software Reference*.

Immediate Velocity Streaming Mode

In this mode, the host computer software supplies a stream of velocity data (X language RM commands) for real-time control of motor speed. The PC23 has no control of acceleration. **The operator controls all velocity changes and must take care that no change is too abrupt for the motor to follow. Otherwise the motor will stall or fault.**

Once this operating mode is entered with the Q1 command, the PC23 clears out any buffered commands waiting for execution and waits for Rate Multiplier in Velocity Streaming Mode (RM) commands. With these commands, the PC23 makes instantaneous changes to the specified velocity. The timing between issuing each RM command determines the exact move profile. Sending RM commands to the PC23 in rapid succession provides smoother motion by virtue of an S-curve acceleration. Testing and modification may be required to establish the correct sequence of RM commands.

CAUTION

Be careful not to make the velocity changes too abrupt. Otherwise, you may make the motor stall.

In the default high velocity range, with the default motor resolution (25,000 steps/rev) and a 25,000-step motor, a data increment of 1 produces a velocity increment of about 15.259 steps per second.

RM commands are issued in real-time. Consequently, it is not possible to download a list of RM values to the PC23, and execute it afterward.

To exit the velocity profiling mode, you must use the **Q0** command. When a **Q1** or **Q0** command is received, it is performed before any buffered commands.

Syntax, Range, and Output Control for RM Commands

The syntax for **RM** commands is **RM**$n_1n_2n_3n_4$ where $n_1n_2n_3n_4$ represents a 4-digit hexadecimal number. The most significant bit of n_1 is interpreted as a direction bit. If the most significant bit =1, the motor will turn in the CW direction. If the most significant bit = Ø, the motor will turn in the CCW direction.

For example, issuing **RM0666** changes the velocity of Axis #1 to 24,994 Hz (666 Hex ❑ 15.259 Hz) in the CCW direction. *This assumes a motor resolution of 25,000 steps/rev.*

To change the direction, you must enter a zero (Ø) point (**RM0000** if the motor was traveling CCW or **RM8000** if the motor was traveling CW). *If you do not enter the zero point, the new* **RM** *data point will be ignored.*

The following table summarizes the **RM** command parameter range.

	CW Rotation		CCW Rotation	
Parameter	Decimal	Hex	Decimal	Hex
Zero	32,768	8000	0	000
Maximum	65,523	FFF3	32,755	7FF3

The hex values 7FF4 through 7FFF, and FFF4 through FFFF are not listed above. These are special function values that do not produce a motor speed value. Instead, they provide signals on the programmable outputs in the middle of the velocity profiling operation. These values may be inserted in the data stream to generate a signal when the PC23 gets to critical points in the process. The functions of these values are shown in the following table.

Output	To set low	To set high
POB #1	7FF4 or FFF4	7FF5 or FFF5
POB #2	7FF6 or FFF6	7FF7 or FFF7
POB #3	7FF8 or FFF8	7FF9 or FFF9
POB #4	7FFA or FFFA	7FFB or FFFB
POB #5	7FFC or FFFC	7FFD or FFFD
POB #6	7FFE or FFFE	7FFF or FFFF

Immediate Streaming Example

Command	Description
Q1	Enter Velocity Profiling mode
RM0011	Go to RM velocity of 249 Hz in CCW direction
RM0055	Go to RM velocity of 1,105 Hz in CCW direction
RM0100	Go to RM velocity of 3,906 Hz in CCW direction
RM0055	Go to RM velocity of 1,105 Hz in CCW direction
RM0011	Go to RM velocity of 249 Hz in CCW direction
RM0000	Go to zero velocity (mandatory) to change direction
RM8011	Go to RM velocity of 249 Hz in CW direction
RM8055	Go to RM velocity of 1,105 Hz in CW direction
RM8100	Go to RM velocity of 3,906 Hz in CW direction
RMFFF5	Set Output #1 high
RM8055	Go to RM velocity of 1,105 Hz in CW direction
RM8011	Go to RM velocity of 249 Hz in CW direction
Q0	Exit velocity profiling mode

Timed Data Streaming Modes

The timed data streaming modes allow you precise multi-axis distance and velocity control. Data streaming is accomplished by dividing the profile into small straight-line segments, allowing you to control the profile shape with greater accuracy.

Time-distance streaming allows control over the number of steps output over a given period of time.

Time-velocity streaming allows control over the frequency of steps output over a given period of time.

In both time-distance and time-velocity streaming modes, outputs can be turned on or off *on-the-fly* (while the motor is in motion). Wait-on-trigger and looping are also possible in these timed data streaming modes.

To produce a data streaming profile, you must complete the following steps:

① Enter the timed data streaming mode for the appropriate axes.

② Define the update interval for each axis.

③ Identify the clock source for each axis.

④* Provide the time-distance or time velocity streaming data.

⑤* Start the master clock.

⑥ Exit the timed data streaming mode after the motion is completed.

** Steps 4 and 5 can be reversed, as discussed later in the Continuous Streaming section.*

Each of the above steps is described in detail in the following paragraphs.

① Enter a Timed Data Streaming Mode

There are two modes associated with timed data streaming. They are time-distance streaming and time-velocity streaming.

To enter the time-distance streaming mode, enter the `Q2` command for the appropriate axes. For example, entering `1Q2` and `2Q2` puts axes one and two in the time-distance streaming mode.

To enter the time-velocity streaming mode, enter the `Q3` command for the appropriate axes. For example, entering `2Q3` and `3Q3` puts axes two and three in the time-velocity streaming mode.

CAUTION

When either the `Q2` or `Q3` modes are entered, motion is killed and the command buffers are cleared on the specified axes.

② Define the Update Interval

The update interval is defined with the `TDnn` command, where `nn` is the update period ranging from 2 to 50 ms in 2-ms increments. Smaller update intervals produce finer discrete line segments. For example, issuing `1TD50` and `2TD50` establishes an update interval of 50 ms for axes one and two. The default update interval is 10 ms for each axis.

CAUTION

All axes in a master/slave relationship (using the `MSL` command) must have the same update interval and the same minimum pulse width (set with the `MR` command).

③ Identify the Clock Source

Using the `MSL` command, you must specify the clock source for each axis. In doing this, you determine the master/slave relationship of the axes to one another or to other boards. The `MSL` command must be specified in the form $\texttt{MSL}n_1n_2n_3$, where each n_n value represents the clock source for axes 1 through 3.

For Axis #1, the possible values for n_1 are as follows:

- **1** If axis #1 uses its own internal clock (master)
- **2** If axis #1 uses axis #2's internal clock (slave to axis #2)
- **3** If axis #1 uses axis #3's internal clock (slave to axis #3)
- **X** If axis #1 is not in the data streaming mode
- **Ø** If axis #1 uses an external clock connected to pins 1 and 2 on the RMCLK slave-to-external connector (refer to Chapter 5 for connections)

For Axis #2, the possible values for n_2 are as follows:

- **1** If axis #2 uses axis #1's internal clock (slave to axis #1)
- **2** If axis #2 uses its own internal clock (master)
- **3** If axis #2 uses axis #3's internal clock (slave to axis #3)
- **X** If axis #2 is not in the data streaming mode
- **Ø** If axis #1 and #2 uses an external clock (via the RMCLK connector)

For Axis #3, the possible values for n_3 are as follows:

- **1** If axis #3 uses axis #1's internal clock (slave to axis #1)
- **2** If axis #3 uses axis #2's internal clock (slave to axis #2)
- **3** If axis #3 uses its own internal clock (master)
- **X** If axis #3 is not in the data streaming mode
- **Ø** If axis #1, #2, and #3 uses an external clock (via the RMCLK connector)

> ***CAUTION***
>
> **Never overlap clock sources. For instance, do not issue `MSL21X`, where axis #2 would use axis #1's clock source and axis #1 would use axis #2's clock.**

When using the external clock, axis #1 must always use the external clock and be set to a data streaming mode (`1Q2` or `1Q3`). If the command string `1Q2 2Q2 3Q2 MSLXØØ` was issued, the `MSLXØØ` command would be invalid because axis 1 is not using an external clock.

The most common way to use the `MSL` command is to specify one axis as the master and slave the other axes to the master. For example, if you use the `MSL111` command, axis #1 is the master and axes #2 and #3 are slaved to axis #1.

④ Establish the Timed Data Streaming Format and Data

The Streaming Data (`SD`) command specifies either the distance to travel (in the Q2 mode) or the velocity output (in the Q3 mode) in one update period. Special data points implement certain functions such as loops, setting the POBs, and waiting for a specific trigger input pattern. *Data-intensive streaming on one axis is likely to reduce command throughput on the other axes that are not in the streaming mode.*

The format for the `SD` command is `SDnnnn(nnnn[nnnn])`, where `nnnn` is a 2, 4, or 6 byte HEX code. If one axis is in the Streaming mode, the format is `SDnnnn`. If two axes are in the Streaming mode, the format is `SDnnnnnnnn`. If all three axes are in the Streaming mode, the format is `SDnnnnnnnnnnnn`.

The order of the four HEX digits corresponds from the lowest to the highest axis number value. For example, if axes #2 and #3 are in the timed data streaming mode, the order is as follows:

If axes #1 and #3 are in the Streaming mode, the order is as follows:

The range of values for the four HEX digits is **ØØØØ - 7EFF** or **8ØØØ - FEFF**. These values can represent distance in the Q2 mode or velocity in the Q3 mode. HEX digits **ØØØØ - 7EFF** correspond to a data range of 0 thru 32511 (decimal).

The most significant bit of the four hex digits sets the direction. A 1 (binary) in this bit position means CW. The remaining three bits specify a magnitude. Therefore, the data range **ØØØØ - 7EFF** corresponds to the CCW direction, and **8ØØØ - FEFF** corresponds to the CW direction. For example, to specify a CW distance of 100 motor steps, the **SD** command would be **SD8Ø64**. To specify a CCW distance of 100 steps, the **SD** command would be **SDØØ64** (64 hex = 100 decimal).

Distance

*For an axis in the normal velocity range (**SSFØ**), the maximum distance that can be specified is calculated as follows:*

[(update interval in ms) * 1000/2] - 1

For example, if the update interval is 10 ms, the maximum distance is 4,999 steps (**SD1387** or **SD9387**).

*For an axis in the low velocity range (**SSF1**), the maximum distance that can be specified is calculated as follows:*

[(update interval in ms) * 100/2] - 1

For example, if the update interval is 10 ms, the maximum distance is 499 steps (**SDØ1F3** or **SD81F3**).

Velocity

For an axis in the normal velocity range, the velocity value for **nnnn** in the **SDnnnn** command is determined by dividing the desired velocity by 15.259 Hz (15.259 steps/rev). For example, to achieve 1,526 steps/sec in the CW direction, you would use the **SD8Ø64** command (1,526/15.259 = 100 = 64rps HEX). If you wanted to move in the CCW direction you would use the **SDØØ64** command.

For an axis in the low velocity range, the velocity value for **nnnn** in the **SDnnnn** command is determined by dividing the desired velocity by 1.526 Hz (1.526 steps/rev). For example, to achieve 153 Hz (steps/sec) in the CW direction, you would use the **SD8Ø64** command (153/1.526 = 100 = 64 HEX) for one update interval. If you wanted to move in the CCW direction you would use the **SDØØ64** command.

CAUTION

When specifying consecutive data points that are different in direction, you must insert a zero (Ø) data point between the two non-zero data points that are different in direction; otherwise, the timed data streaming mode for this axis will be exited.

Special Data Points

The maximum value for a normal data point is **7EFF** or **FEFF**. The range of values **7FØØ - 7FFF** and **FFØØ - FFFF** is used to perform specialized functions. *Data values in this range are interpreted **not** as data, but as special commands.*

☛Loops **SD7FØ1** thru **SD7F7F** and **SDFFØ1** thru **SDFF7F** are loop commands with the loop count as the two least significant hex digits. **SD7F8Ø** and **SDFF8Ø** are endloop commands. The most significant bit corresponds to the direction. Use **SD7FØØ** through **SD7F8Ø** for loops beginning in the CCW direction. Use **SDFFØØ** through **SDFF8Ø** for loops beginning in the CW direction.

The loop count range is 1 thru 127. **SD7FØØ** (CCW) and **SDFFØØ** (CW) are loop commands with an infinite loop count. For example, to create an infinite loop that causes 100 steps to be output for each update period, you could issue the following commands:

Command	Description
SDFFØØ	Loop an infinite number of times
SD8Ø64	Move CW 100 steps
SDFF8Ø	End loop

For a finite loop count of 5, you could issue the following:

Command	Description
SD7FØ5	Loop 5 times
SDØØ64	Move CCW 100 steps
SD7F8Ø	End loop

Any loop can be terminated on a *loop boundary* by sending another endloop data point. For instance, if the PC23 was half way through the third cycle of a 5-cycle loop and you issued another **SD7F8Ø** (or **SDFF8Ø**) command, the PC23 would finish the current loop cycle and then stop.

☛Outputs **SD7FCØ** thru **SD7FFF** are commands used to set or clear POBs 1 thru 6. In the **SD** data point, bit positions Ø thru 5 correspond to POBs 1 thru 6 respectively as follows:

SD Bit:	7	6	5	4	3	2	1	Ø
POB #:	X	X	6	5	4	3	2	1

To set or clear a POB, two POB data points are required: a set/clear mask and a don't care mask, specifying which bits are to be affected by the set/clear mask and which ones are to be unaffected.

When a pair of POB set/clear and don't care mask data points are encountered after the POB action is taken, the next distance or velocity data point is taken from the data buffer. ***This means that only 1 POB data point pair is allowed per update interval. If you specify more than one POB data point pair, the motion may not be smooth.***

The following example demonstrates how to calculate the two POB data points needed to set and clear POBs 1 through 6:

Desired outcome: POB1 = 0 POB2 = 1 POB3 = 1
POB4 = X POB5 = X POB6 = 1

① Calculate the first **SD7Fnn** command (*set/clear mask*). The purpose of the set/clear mask is to specify which bits (outputs) are not being turned off (output = Ø).

a. Since there are only six outputs, put 11 on the front of the POB pattern (1, 1, POB6, POB5, POB4, POB3, POB2, POB1). This is done to make two hex characters. The result is **11XX110**.

b. Replace Xs with 1s. The result is **11111110**.

c. Convert the 8-bit binary number to HEX (result is **FE**).

d. Thus, you should specify the data point as **SD7FE**.

② Calculate the second **SD7Fnn** command (*don't care mask*). The purpose of the don't care mask is to specify which bits (outputs) are to be left in their current state (output = X).

a. Replace 1s with 0s, and Xs with 1s in the POB pattern. The result is **011000**.

b. Put 11 on the front of the new POB pattern (to make two hex characters). The result is **11011000**.

c. Convert the 8-bit binary number to HEX (result is D8).

d. Thus, you should specify the data point as SD7FD8.

When axes are in a master/slave arrangement via the MSL command, SD data alignment is based on valid data points and not on Trigger, POB, or Loop/Endloop data points. Entering the following commands will output 100 steps on each axis for the first update interval. POB #1 will be high and both axes will output 50 steps during the second update interval:

Command	Description
SD8Ø648Ø64	Move both axes CW 100 steps
SD7FFF7FFF	Set POB #1 high and leave the rest SD7FFE7FFE unchanged (equivalent to 1XXXXX)
SD8Ø328Ø32	Move both axes CW 50 steps

☛Triggers SDFFCØ thru SDFFFF are commands used to implement a *buffer pause* based on the pattern of Trigger bits 1 thru 6. *Buffer paused* means that the previous data value is repeated until the trigger condition is satisfied. In the SD data point bit, positions Ø thru 5 correspond to Trigger's 1 thru 6 respectively (i.e., bit Ø is Trigger bit 1).

To implement a buffer wait function, two trigger *data* points are required. A *wait mask* and a *don't care mask* specify which bits are to be affected by the wait mask and which ones are to be unaffected. The calculations to compute each mask are performed exactly the same as the output mask calculations discussed in the previous section.

Once a valid trigger wait pair of data points is encountered, data points in the buffer are not fetched in subsequent update intervals (buffer-paused) until the pattern is satisfied. For example, to wait for trigger bits 2 and 3 to be set and bit 1 to be cleared while ignoring bits 4, 5, and 6, the following SD commands would be used:

Command	Description
SDFFFE	Wait for bit 1 to clear, bits 2 & 3 to be set, and SDFFF8 ignore bits 4, 5, & 6 (011XXX)

⑤ Start the Master Clock

The Master/Start (MSS) command initiates the start of the clock for the pulse generation circuitry on any axis that is designated as a master.

Data Streaming Restrictions

The following are restrictions to be considered when using the timed data streaming mode:

- ❑ If the master/slave relationship with another board is desired, then axis #1 must be included as a streaming axis, and must use the external clock.
- ❑ The minimum motor resolution required is 5,000 steps/rev.
- ❑ All master/slave axes must have the same velocity range (SSF command).
- ❑ If a limit is hit while in a timed data streaming mode (Q2 or Q3), the timed data streaming mode is exited for that axis only. This is equivalent to a QØ command.
- ❑ For update periods of 6 ms or less, the binary input mode is recommended (see below).
- ❑ While streaming in a master/slave relationship, all slave axes should be exited first, before the master axis. For example, when operating axis 1 through 3 under the command MSL111, you should exit as follows: 3QØ ... 2QØ ... 1QØ.
- ❑ Discrete data streaming is limited by the PC23 buffer size (see the section on Continuous Streaming discussed later in this chapter).

Time-Distance Streaming Example

An application requires axes one and two to run in the time-distance streaming mode and axis three to be independent. The commands used are as follows:

Command	Description
3A10	Acceleration = 10 rps^2
3V10	Velocity = 10 rps
3D25000	Distance = 25000 steps
3L200	Loop 200 times
3G	Start motion
3T1	Pause 1 second
3H	Change direction
3G	Start motion
3H	Change direction
3T2	Pause 2 seconds
3N	End loop
1Q2	Enter streaming mode axis #1
2Q2	Enter streaming mode axis 2
MSL22X	Synchronize axis 1 clock to axis 2, axis 3 independent
1TD10	Axis 1 update interval = 10 ms
2TD10	Axis 2 update interval = 10 ms
SD00640096	Move axis #1 100 steps CCW, move axis #2 150 steps CCW
SD00000000	Stop axis 1 and 2 - *required to change direction*
SD806481CA	Move axis 1 100 steps CW, move axis 2 458 steps C
SDFF05FF05	Loop axes 1 and 2, 5 times CW
SD80648064	Move axes 1 and 2 100 steps CW
SD7FF67FF6	Two commands required to
SD7FDØ7FDØ	Set POBs (6,5,4,3,2,1) to 1X0110
SD80968096	Move axis 1 and 2 150 steps CW
SD7FD97FD9	Two commands required to
SD7FDØ7FDØ	Set POBs (6,5,4,3,2,1) to 1X1001
SDFF80FF80	End loop
SD00000000	Stop axes 1 and 2
SD7F7F7F7F	Loop axis 1 and 2, 127 times CCW
SD010000FF	Move axis 1 256 steps CCW, move axes 2 255 steps CCW
SD00000000	Stop axis 1 and 2
SDFFFFFFFF	Both axes wait on trigger (6,5,4,3,2,1,)
SDFFFBFFFB	To be XXX1XX
SD7F807F80	End loop
SD00000000	Stop axis 1 and 2
MSS	Start master clock. Computer time delay needed (e.g., 1400 ms)
1QØ	Exit streaming on axis 1
2QØ	Exit streaming on axis 2

Time-Velocity Streaming Example

An application requires axes one and three to run in time-velocity streaming mode and axis two to run independently. The commands used are as follows:

Command	Description
2A10	Acceleration = 10 rps^2
2V10	Velocity = 10 rps
2D10000	Distance = 10000 steps
2L200	Loop 200 times
2G	Start motion
2T1	Pause 1 second
2H	Change direction
2G	Start motion
2H	Change direction
2T1	Pause 1 seconds
2N	End loop
SSF0	High velocity mode (default)
1Q3	Enter streaming mode axis 1
3Q3	Enter streaming mode axis 3
MSL1X1	Synchronize axis 1 clock to axis 3, axis 2 independent
1TD20	Axis 1 update interval = 20 ms
3TD20	Axis 3 update interval = 20 ms
SD7F7F7F7F	Loop axis 1 and 2, 127 times CCW
SD00050005	By end of update interval, velocity = 76.3 steps/sec
SD00100010	By end of update interval, velocity = 244.1 steps/sec
SD00000000	By end of update interval, velocity = 0 steps/sec
SD7F807F80	End loop
MSS	Start master clock) computer delay needed (e.g., 60 ms)

```
1Q0                 Exit streaming mode axis 1
3Q0                 Exit streaming mode axis 3
SSF1                Low velocity mode
1Q3                 Enter streaming mode axis 1
3Q3                 Enter streaming mode axis 2
MSL1X1              Synchronize axis 1 clock with axis 3, axis 2 independent
1TD50               Axis 1 update interval = 50 ms
3TD50               Axis 3 update interval = 50 ms
SDF005F005          By end of update interval, velocity = 76.3 steps/sec
SD7FF67FF6          Two commands required to
SD7FD87FD8          Set POBs (6,5,4,3,2,1,) to 1X0110
SD7F287F28          Loop axes 1 and 2, 20 times CCW
SDF0085F085         By the end of update interval velocity = 76.3 steps/sec
SD7F887F88          End loop
SD7FD97FD9          Two commands required to
SD7FD87FD8          Set POBs (6,5,4,3,2,1,) to 1X1001
SD80108010          By end of update interval, velocity = 24.41 steps/sec
SD00000000          By end of update interval, velocity = 0 steps/sec
SD7F807F80          End loop
MSS                 Start master clock. Computer delay needed (e.g., 150 ms)
1Q0                 Exit streaming mode axis 1
3Q0                 Exit streaming mode axis 3
```

Motion stops when the time velocity buffer is empty.

Discrete Data Streaming

The time-distance and velocity-distance examples on the previous two pages use what is referred to as *discrete data streaming*, which follows this pattern:

①	Q2 or Q3	Enter time-distance or velocity-distance mode
②	TD	Provide time-distance update rate or period
③	MSL	Specify master/slave clock source
④	SD	Specify stream data or send distance
⑤	MSS	Master/slave start (start movement)
⑥	Q0	Exit time-distance mode

This form of streaming is useful when the number of SD data points does not exceed the PC23's buffer capacity. Since each axis buffer can hold 1,000 characters, this corresponds to about 200 SD data points per axis (1,000 bytes @ 5 bytes per SD data point).

By using discrete streaming, the standard communication handshaking can be used (refer to **SIMPLEX.BAS**, **PC23P.PAS**, or **PC23C.C** on the support disks). Discrete streaming also allows update intervals of 2 ms to be used without using the binary input mode or assembly language drivers.

Continuous Data Streaming

Continuous data streaming differs from discrete data streaming in that the SD data points are sent to the PC23 **after** starting the master clock, instead of **before** starting the master clock. By choosing continuous streaming over discrete streaming, you are not limited by the PC23 buffer. Therefore, you can send an infinite number of SD data points to the PC23.

CAUTION

You must check bit 7 of the status byte to ensure that the PC23 data buffer is able to handle the data. The buffer cannot accept new data until bit 7 changes from Ø to 1. You can send up to 200 data points (2 bytes per point) at one time and wait for bit 7 to change from Ø to 1 to send 200 more.

The command sequence required to conduct a continuous streaming operation for each axis is as follows:

①	Q2 or Q3	Enter time-distance or velocity-distance mode
②	MSL	Specify master/slave clock source
③	TD	Provide time-distance update rate or period

④ `MSS` Master/slave start (start movement)
⑤ `SD` Specify velocity or # of steps to output
⑥ `Q0` Exit time-distance mode

Step 5 of the above sequence can be accomplished in two different ways. One way is by repeatedly using the standard PC23 write drivers (refer to the `SIMPLEX.BAS`, `PC23P.PAS`, or `PC23C.C` programs on the support disks). The other approach is to use the PC23's binary input capability, as described below and reference on support disk #2, files `PC23TDP.PAS` and `PC23TDC.C`.

Binary Input Mode

By using the binary input mode, you are able to use a much faster update rate. Having a shorter update interval allows more precise control of the motion segments. Within the binary input mode, the data points are entered as binary words. Therefore, when you enter data this way, only two, four, or six bytes (for one, two, or three axes) per input session are allowed. During each input session, one SD data point (for 1, 2, or 3 axes) is transferred to the PC23.

The steps required to support a binary data input mode session are as follows:

① Read the status byte (base address + 1) until bit 4 is set. (PC23 ready to receive a byte)

② Send one byte of the 2, 4, or 6-byte data packet to the PC23 data register (base address).

③ Send HEX 71 to the control byte (base address +1). This puts the PC23 in binary input mode and informs the PC23 that a data byte has been transferred to it.

④ Read the status byte until bit 4 is cleared. (data byte accepted by PC23)

⑤ If the data byte that you sent in step 2 is the last byte of the 2, 4, or 6-byte packet, send HEX 6Ø to the control byte (exit binary input mode). If the data byte sent in step 2 is not the last byte, send HEX 61 (remain in binary input mode) to the control byte and repeat steps 1 through 5.

The binary input mode will work only in continuous streaming.

Example

Axes #1 and #2 are in the time-distance mode (Q2) with axis #1 as the master and axis #2 as the slave, using the step pulse clock from axis #1 (i.e., `MSL11X`). The update rate is 2 ms (TD2).

To input a data point such that axis #1 outputs 100 steps in the CW direction for one update period and axis #2 outputs 200 CW steps for the same period, two binary words (4 bytes) are sent in one binary input mode session as follows (the sequential order is from top to bottom):

Command	Description
`MSB1`	HEX 80 (binary, not ASCII)
`LSB1`	HEX 64
`MSB2`	HEX 80
`LSB2`	HEX C8

On software support disk #2, binary input mode examples are written in two different program languages: PASCAL and C.

Language Considerations

When creating a program in the timed data streaming mode, you must consider the programming language you are using. Certain languages, such as C, run considerably faster than BASIC. PASCAL also runs faster than BASIC, even a compiled BASIC, such as Microsoft QuickBASIC.

On tests performed with an IBM compatible (80286 microprocessor), Borland TURBO C 2.0 was able to run a continuous timed data streaming program with an update interval of 6 ms (`TD6`). This program used the binary input mode to communicate `SD` data points to the PC23. Using Borland TURBO PASCAL 5.0, a similar program ran at 12 ms update intervals (`TD12`). Using Microsoft Quick BASIC 4.5, a comparable program ran at 30 ms update intervals (`TD30`).

If you require update intervals of 2 ms, you must use assembly language binary input mode drivers. By linking these assembly language routines into your BASIC, C, or PASCAL program, you will be able to obtain 2 ms update intervals, even on a standard PC with an 8088 microprocessor.

Examples of how to link the assembly language binary input mode drivers are provided on support disk #2. For BASIC, reference the `\BAS_ASM` subdirectory. For C, reference the `\C_ASM` subdirectory. For PASCAL, reference the `\PAS_ASM` subdirectory.

Communicating with the PC23

This section describes how the computer communicates with the PC23.

Read and Write Registers

To operate the PC23 with X language commands and sequences, you must be able to communicate with it over the computer's I/O bus. Communication with the PC23 involves two pairs of *registers*. A register refers to a temporary storage area for holding one character (or one eight-bit byte). Data transfer to and from the register takes place one character at a time.

Register pair #1 resides at the PC23's bases address set with the 8-position DIP switch and consists of the input data buffer (IDB) and the output data buffer (ODB).

Register pair #2 resides at one address location above the PC23's bases address and consists of the control byte (CB) and the status byte (SB).

The ODB and the SB are *read-only registers*. The IDB and the CB are *write-only registers*. Sample *read* and *write* routines that access the computer's I/O bus are provided on the demonstration diskettes which accompany the PC23. The routines are written in BASIC, C, and PASCAL.

Control Byte and Status Byte

X language commands are *strings* of ASCII characters. Sending a command to the PC23 requires transferring each character in the string one at a time. Each character transfer requires that the sender notify the receiver that a character is ready, and that the receiver notify the sender that the character has been received. This notification process involves setting or clearing control bits (flags) in the 8- bit SB and CB registers.

Set = high = binary value of **1**
Clear = low = binary value of **Ø**

Control Byte flags allow the host program to signal the PC23 with messages such as "`A10 V10 D25ØØØ G`". The Status Byte flags allow the host program to check on the PC23 operating conditions such as if motor C or axis #2 is moving .

Control Byte (CB) Flags

The following table shows the control byte flags available to the programmer for signalling the PC23. Both this register and the status byte are used to communicate with the PC23.

Bit	Definition
0	Binary Input Mode (active high)
1	*Unused*
2	Stop watchdog timer (active high)
3	Acknowledge interrupt (active high)
4	IDB command character ready (active high)
5	Restart watchdog timer (active low)
6	Reset interrupt output (active low)
7	ODB message character accepted (active high)

Bit 0, when set, indicates that the binary input mode of data streaming is being used.

Bit 2, when set, causes the PC23's watchdog timer to time out and stop. When the timer stops, it forces a hardware reset. The reset condition may be cleared by cycling power or restarting the timer (see Bit 5 below).

Bit 3, when set, tells the PC23 that its interrupt signal to the computer has been noted and is no longer needed (see Bit 6 below).

Bit 4, when set, tells the indexer that a command character has been put into the IDB. The PC23 then clears Bit 4 of the status byte (SB) to indicate that the IDB is unavailable, reads the character in the IDB , and then sets Bit 4 of the SB to indicate to the host that the IDB is again ready for a new character.

Bit 5 restarts the watchdog timer. It must first be cleared, then the timer will start up when the bit is set again. This bit should never be toggled unless the timer has timed out.

Bit 6 resets the hardware interrupt latch and thus the interrupt output. The interrupt output cannot be reset unless the interrupt is first acknowledged with Bit 3 above. These bits should be cleared during Reset or Interrupt acknowledge.

Bit 7, when set, tells the PC23 that a response character, previously placed in the ODB by the PC23, has been received by the host, and a new character may be placed in the ODB.

Status Byte (SB) Flags

The status byte provides several information flags for the programmer as shown in Table 4-5. You can use the status byte to assist in the communications process and to provide run-time status information without the need to burden the indexer with routine status request commands.

Bit	Definition	Power-up State
0	Axis 2 stopped (active high)	Set
1	Axis 1 stopped (active high)	Set
2	Axis 3 stopped (active high)	Set
3	ODB ready (active high)	Cleared
4	IDB ready (active high)	Set
5	Board fail (active high)	Cleared
6	Interrupt active (active high)	Cleared
7	Timed data buffer > 1/2 full (active low)	Cleared

Bits 0, 1, and 2 indicate whether the motors for the three axes are moving. At the beginning of any move, the appropriate bit is cleared. Specifically, these bits indicate whether or not the indexer is sending step pulses to the drives. *These bits do not indicate if position maintenance is in effect.* Bits 3 and 4 are set when their corresponding data buffer is ready.

Bit 3 is set when the ODB contains an output character for the host, signalling the host to read the information it contains.

Bit 4 is set when the IDB is ready (computer may write a character to the IDB).

Bit 5, when set, tells the PC that the PC23's watchdog timer has *timed out*, possibly indicating an internal failure from which it cannot recover. The only way to clear this bit is to reset the indexer by cycling power to the PC23 or by sending hex 6Ø to the CB to restore it. Resetting the PC23 is discussed in the Programming section of this chapter. Exercising the self-test function will also set this bit. When bit 5 is set, the drive shutdown output goes active, removing motor torque and generating a drive fault.

Bit 6 indicates to the host that a conditional interrupt has been armed and that an interrupt has occurred. If either jumper JU27 (selects interrupt 3) or JU28 (selects interrupt 4) on the indexer board is installed, then the PC23 has generated a hardware interrupt signal.

Bit 7, when cleared, indicates that the timed data streaming buffer is over half full. This bit is only active when in time-velocity or time-distance streaming mode. At all other times, the bit is cleared.

Communication Process

The following sequence of events occurs when sending a character to the PC23:

① Read the SB until bit 4 is set.
② Write a byte (character) into the IDB.
③ Write to the CB and set bits 4, 5, and 6.
④ Read the SB until bit 4 is cleared.
⑤ Write to the CB to clear bit 4 and set bits 5 and 6.

The following sequence of events occurs when receiving a character from the PC23:

① Read the SB until bit 3 is set.
② Write a byte (character) from the ODB.
③ Write to the CB and set bits 5, 6, and 7.
④ Read the SB until bit 3 is cleared.
⑤ Write to the CB to clear bit 7 and set bits 5 and 6.

Programming

The PC23 comes supplied with two floppy diskettes containing support routines written in BASIC, C, and PASCAL. In addition, ASSEMBLY routines to reset, read, and write to the PC23 are included.

The support disks are divided logically into sub-directories. To go between these sub-directories type: `cd[subdirectory name]`. To back up one subdirectory type: `cd..`

The BASIC support routine is contained in the **BASIC** subdirectory.

The C support routine is contained in the **C** subdirectory.

The PASCAL support routine is contained in the **PASCAL** subdirectory.

The ASSEMBLY routines are contained in the **BAS_ASM**, **C_ASM**, and **PAS_ASM** sub-directories. Each of these sub-directories show how to link ASSEMBLY code to the appropriate programming language (BASIC, C, or PASCAL).

Information about the support diskettes is contained in a menu-driven program called **INFO.EXE**. To run this program, simply type **INFO** at the DOS prompt. If you would like a hard copy of the information presented in **INFO.EXE**, print the file called **INFO.DAT**.

A terminal emulator program has also been included with the support code. This program (**PC23TERM.EXE**) can be used to talk to the PC23 directly. It is a good program to practice making motion with the PC23.

Use the following procedure to copy the support routines to your hard disk:

① Create a sub-directory in which to put the routines (**mkdir PC23**).

② Change directories to the newly created subdirectory (**cd PC23**).

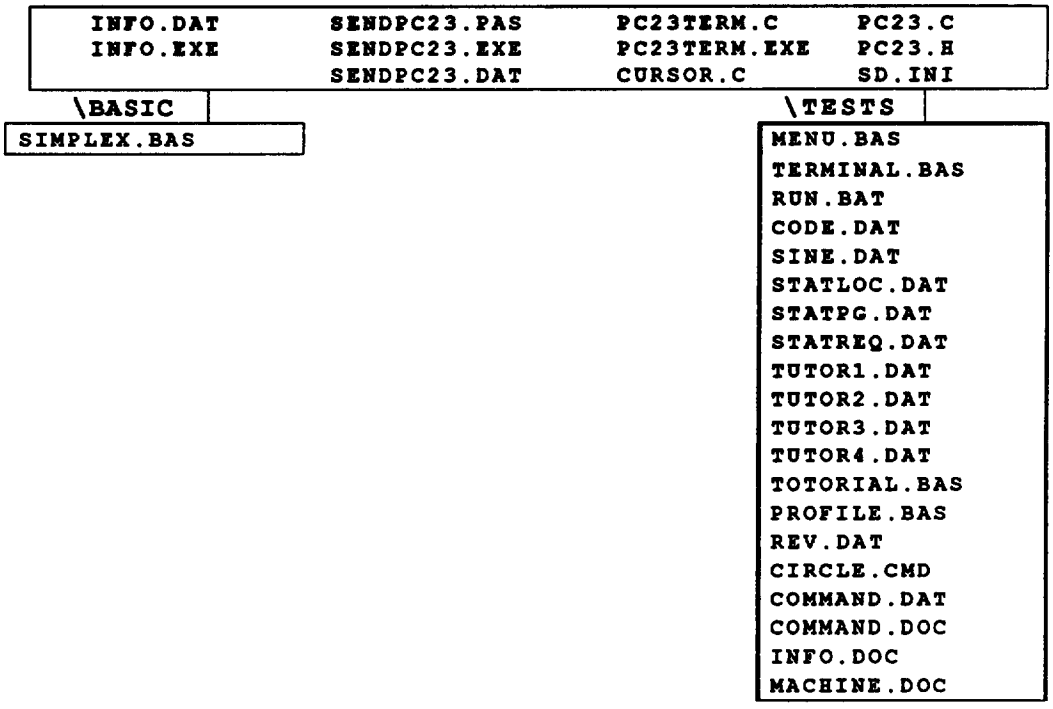
③ Copy the file **INSTALL.BAT** to this sub-directory (copy a:**INSTALL.BAT**).

④ Type: **INSTALL**.

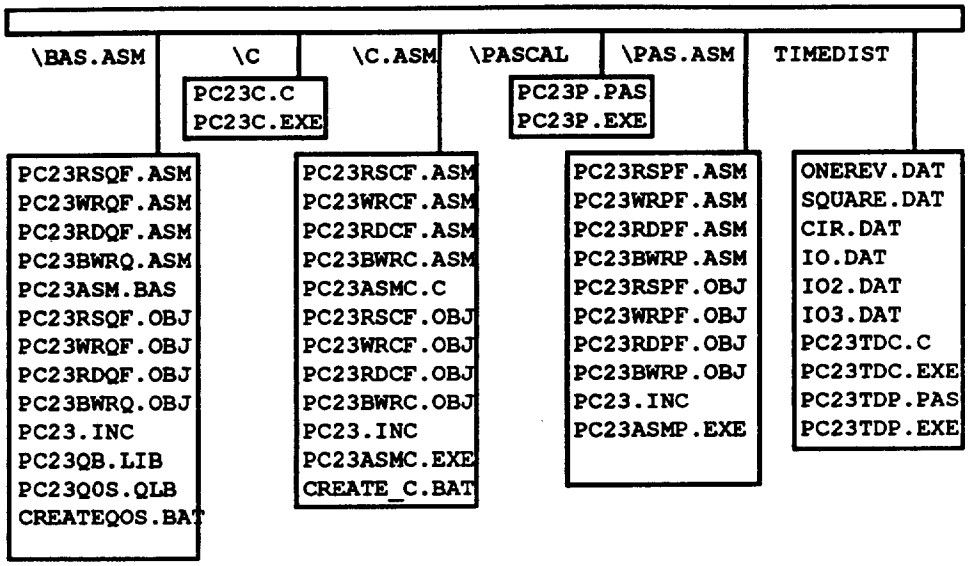
Support Disk Files

To install the support disk onto your hard drive. Type **INSTALL** at the DOS prompt.

Support Disk File Structure



Support Disk File Structure



Designing the Computer Program

Responsibility for creating computer programs lies with the user. Creating an interactive program should pose no problem for an experienced programmer. If you are not familiar with programming for your computer, it would be prudent to seek the advice or assistance of someone familiar with both programming and your computer.

The PC23 is designed to operate motor axes in a fashion largely independent of the computer, requiring only a small number of high level commands and interaction. This interaction is almost exclusively in the form of characters and strings rather than numbers. The programmer will need knowledge of string handling in the programming language to be used. Regardless of the intended purpose of the program, it must include subroutines or procedures to do the following (in order of importance):

- Reset the indexer
- Send a command string to the indexer
- Receive a character string from the indexer
- Decode Status Request responses
- Allow input of indexer command files

Ideally, the programs should be set up so that a non-programmer operator can routinely modify appropriate parameters or otherwise change the process being controlled without requiring that the program be rewritten or compiled. The examples in this chapter are designed to illustrate the approach you would use to communicate with the indexer and are not necessarily optimum for any specific application.

Sending and Receiving Strings

The basic procedure for sending and receiving single characters is shown below. The algorithms discussed later in this chapter are required building blocks for any program.

Sending command strings of varying lengths is easy because the length of the string is easily known. Any general-purpose programming language will have the string length and string pointer functions needed to put together an iterative algorithm to send a string one character at a time.

Receiving response strings is complicated by the fact that not all responses are the same length. It is not always possible to predict when a response

will occur, since some responses are strings representing numbers and some responses are codes represented by strings.

The choice of commands will influence the kind of data returned by the indexer. Therefore, you must make provisions to interpret the indexer responses to status requests. When the response is a single character such as **@**, **A**, **B**, **M**, etc., the meaning of the character is a function of the requesting command.

There is nothing inherent in a response that identifies which process of decoding should be applied to the single character of the response. Position report responses are a function of the specific command to the PC23 and can be decimal, hexadecimal, or binary data, with varying zero reference positions.

In the default report format, Axis 1 can respond in a format different from the other two axes. In general, it is always a good programming practice to precede the status request commands with the appropriate axis specifier.

Receiving Status Information and Data

It is comparatively simple to read status with an `INP` instruction like the following:

```
BYTE  =  INP(ADDRESS+1)      BASIC
BYTE  =  INPORTB(ADDRESS+1)        TURBO C
BYTE  =  INP(ADDRESS+1)      MICROSOFT C
BYTE:  =  PORT(ADDRESS+1) TURBO PASCAL
```

The numeric variable named **ADDRESS** has previously been set equal to the base address of the PC23. This instruction sets the numeric variable named **BYTE** equal to the binary number corresponding to the bit pattern of the PC23's status byte.

This instruction may be executed at any time regardless of what the PC23 is doing. The resulting variable **BYTE** may be used as a regular number or as a logic value. As a logic value it can be logically **AND**ed with a *mask* logic value to reset all the bits in **BYTE** other than the status bit of interest.

Acquiring data bytes from the PC23, such as a position or specific status report, calls for the steps detailed in the following section. The **INP** instruction is also used to read response characters. They must be converted from numbers with the **BASIC VAL** instruction, and linked to form the entire response string. A sample **BASIC** input driver is shown in the BASIC example in the **SIMPLEX.BAS** program on the support disk.

Testing Individual Status Bits

In the definition of the status byte (SB) above, Bit 3 of the 8-bit status byte will be a one (**1**) if an output character is waiting, zero if not. (The bits are numbered 0 through 7). To test this bit, set all the other bits to zero by **AND**ing the byte with a mask and perform a logical test on the result for zero (false). Bit 3 has a binary weight of 8.

Example: Logical Masking

Suppose the PC23 returned a status byte value of 89, Bit 3 is high:

```
          Dec     Hex     Binary
BYTE =     89 =    59 =   01011001 \
                                   > logic AND = 00001000
Mask =      8 =    08 =   00001000 /      (TRUE)
```

If the PC23 returns the value 70. A check on Bit 3 reveals that it is low, as follows:

```
        Dec     Hex     Binary
BYTE =      70 =    46 =    01000110 \
                                      > logic AND = 00000000
Mask =       8 =    08 =    00001000 /       (FALSE)
```

In **BASIC**, this check might appear as follows:

```
3000  BYTE = INP ( ADDRESS+1 )
3010  IF ( BYTE AND 8 ) > 0 THEN GOTO 3020 ELSE GOTO 3000
3020  CHAR = INP (ADDRESS)
3030  CHAR$ = CHR$ ( CHAR )
3040  ANSWER$ = ANSWER$ + CHAR$
3050  etc
```

This instruction keeps sending the program back to read the PC23 status until a character is ready. Then the character may be read. Note that this example will *trap* the computer if the PC23 has not received a status command.

Sending Control Information and Data

To write bytes to the PC23, you can use **OUT** instructions such as the following:

```
OUT  ADDRESS,  ALPHA          BASIC
OUTPORTB(ADDRESS,  ALPHA);        TURBO C
OUTP(ADDRESS,  ALPHA);     MICROSOFT C
PORT[ADDRESS]:  =  ALPHA;  PASCAL
```

Resetting the PC23

The following are step-by-step procedures for writing your own subroutines for resetting the PC23:

① Write 64 Hex to the Control Port (Board Address +1)

② Read the Status Port (Board Address +1) until (the status byte AND 20 Hex)> 0

③ Write 40 Hex to the Control Byte (Board Address +1)

④ Write 60 Hex to the Control Byte (Board Address +1).

⑤ Read the Status Port (Board Address +1) until (the status byte and 7F Hex) = 17 Hex.

⑥ Write 20 Hex to the Control Port (Board Address +1)

⑦ Write 60 Hex to the Control Port (Board Address +1).

Reading Characters From the PC23

Use the following procedure to read characters from the PC23:

① Initialize the ASCII variable to null (Ø).

② Read the Status Port (Board Address +1) until (Status Byte AND 8 Hex) >0.

③ Read the Data Port (Board Address) in to the ASCII variable.

④ Write EØ Hex to the Control Port (Board Address +1)

⑤ Read the Status Port (Board Address +1) until (the status byte AND 8 Hex) = 0.

⑥ Write 60 Hex to the Control Port (Board Address +1)

Writing Characters to a PC23

Use the following procedure to write characters to the PC23:

① Convert the character to ASCII. This may not be necessary in some programming languages such as C (except for axes in the binary input data streaming mode).

② Read the Status Port (Board Address +1) until (the Status Byte AND 10 Hex) > 0.

③ Write the ASCII character to the Data Port (Board Address+1).

④ Write 70 Hex to the Control Port (Board Address +1).

⑤ Read the Status Port (Board Address +1) until (the Status Byte AND 10 Hex) = 0.

⑥ Write 60 Hex to the Control Byte (Board Address +1).

Program Examples

The following support disk files provide read, write, and reset routines for the PC23:

- **SIMPLEX.BAS** (written in BASIC)
- **PC23P.PAS** (written in PASCAL)
- **PC23C.C** (written in C)

These files provide the foundation from which you can design a motion control program for the PC23. The following are examples of how to use these files to create your own PC23 program.

BASIC Program Example

Use the following steps as a guide to develop your custom PC23 BASIC program:

① Make a copy of **SIMPLEX.BAS**.

② Edit the copy starting at line # 5000.

③ To send a command, place the X language commands in CMD$ (CMD$ = **A1Ø V1Ø D25ØØØ G**)

④ To get a response from the PC23, use the GOSUB 3000. The response is returned in ANSWER$.

```
5000   CMD$ = "1PR "
5010   GOSUB 1000          'write to PC23
5020   GOSUB 3000          'read from PC23
5030   IF ANSWER$ = "" THEN 5020
5040   PRINT ANSWER$
```

⑤ Compile and run the program.

C Program Example

Use the following steps as a guide to develop your custom PC23 C program:

① Make a copy of **PC23C.C**.

② Edit the main program of the copied file.

③ To send a command, place the X language in message (message = **A1Ø V1Ø D25ØØØ G**). Then make a call to procedure writecmd (message).

④ To get a response from the PC23, use procedure readanswer (answer). The response is returned in answer.

⑤ Example:

```
main()  {
char    *message,*answer;
answer="";
initialize();                /*resets  PC23*/
message = " 1PR ";
writecmd  (message);
readanswer  (answer);
printf  (answer);
}    /* end of main*/
```

⑥ Compile and run the program.

PASCAL Program Example

Use the following steps as a guide to develop your custom PC23 PASCAL program:

① Make a copy of **PC23P.PAS**.

② Edit the main program of the copied file.

③ To send a command, place the X language commands in message (cmd = A1Ø V1Ø D25ØØØ G). Then make a call to procedure writecmd (768,cmd) where 768 is the PC23 board address.

④ To get a response from the PC23, use procedure readanswer (768,answer). The response is returned in answer.

⑤ Example:

```
begin
answer:='';
Initialize  (768);
cmd:=' A1Ø  V1Ø  D25ØØØ  G  PR  ';
Writecmd  (768,cmd);
while answer='' do Readanswer (768,answer);
writeln  (answer);
End
```

⑥ Compile and run the program.

Special Modes of Operation

This section discusses special modes of operation for the PC23. These special modes include the following:

- Joystick operation
- Using multiple PC23s with one computer
- X-Y linear interpolation
- Using interrupts

Joystick Mode

The PC23 uses an 8-bit analog-to-digital (A/D) converter to accept an analog voltage input (generally from a 5kΩ potentiometer joystick) to control motor velocity on axes 1 and 2.

Joystick connection and adjustment instructions are provided in Chapter 3, *Installation*.

You can enter input for the Joystick mode with the **J1** command, which assigns two A/D inputs (used in differential mode) to the specified axis. For example, if no axes are in the joystick mode, **CHØ** and **CH1** of the A/D are assigned to the first axis that enters the Joystick mode. Once you enter the Joystick mode, steps are output in a CW or CCW fashion, depending on the magnitude of the voltage input and the parameters that you set via the various joystick commands (refer to the **JØ**, **J1**, **JB**, **JD**, **JV**, and **JZ** command descriptions in Chapter 5, *Software Reference*).

The voltage range for the A/D is 0 - 2.5V. This range is divided into two regions. One region is from 0 - 1.25V and the other is from 1.25 - 2.5V. If the input voltage is exactly 1.25V, no steps will be output. If the voltage level is from 0V - 1.24V, steps will be output in the CCW direction, with the maximum velocity at 0V. The maximum joystick velocity is determined (1), by the value of the last **V** command that was issued *prior to entering the joystick mode*, or (2), with the **OSF** command (refer to the **OSF** command description in Chapter 5).

If the input voltage is between 1.26V - 2.5V, steps will be output in the CW direction. Zero step offset can be performed with the **JZ** command. Zero step output can be set to 1.25V (±0.25V). A zero step dead band symmetrical about the zero step set point established with the **JZ** command can be invoked with the **JD** command. If the motor drifts, exit the joystick mode and issue an appropriate **JD** command.

You can establish a trapezoidal backlash move that will occur whenever the dead band region is traversed (i.e., a change direction). You can set the distance and velocity for this move with the **JB** and **JV** commands. The acceleration for this move is determined by the last **A** command that was issued before the Joystick mode was entered.

Joystick Commands

Command	Description
AVn	Analog voltage request from channel n. Response will range from 0.000V to 2.500V.
JBnnnn	Joystick backlash of nnnn steps
JDØ.nnn	Joystick dead band in Ø.nnnV
JVnn.nnn	Joystick backlash compensation velocity in nn.nnn rps
JZ	Joystick Zero
J1	Joystick Enable
JØ	Joystick Disable
OSF	Set maximum joystick velocity

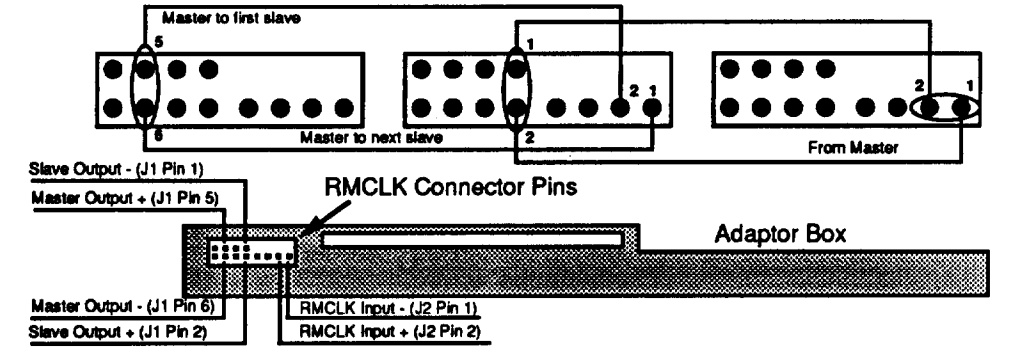
Multiple PC23 Addressing

Multiple PC23s can be synchronized by using one master clock. By connecting the adapter boards of each PC23 via the RMCLK connectors and by using the MSL command to specify the master/slave relationship of the different axes, any number of PC23s can then be made synchronous. ***As identified in the following figure, synch connections for PC23s with serial numbers ≤3490 are different than for those with serial numbers ≥3491.***

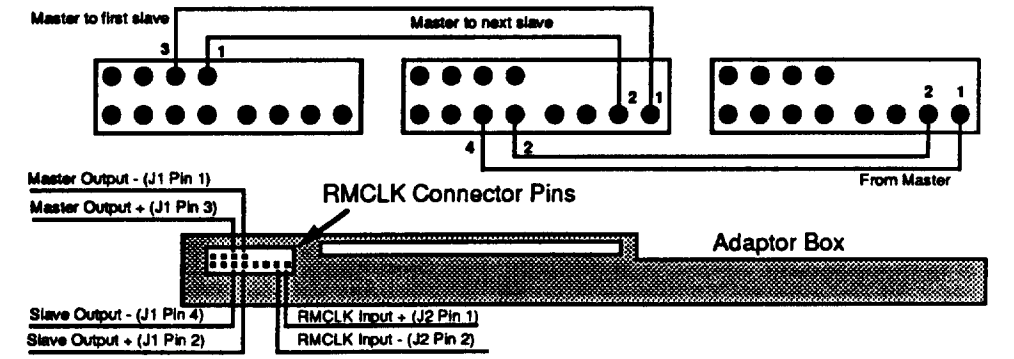
The following table lists parts and tools needed to make your synchronization cable.

20-22 AWG wire
AMP connectors
Tools:
- Manual tool for double row connector
- Conversion kit for single row connector
- Single row 4 position (cover)
- Single row 4 position (housing)
- Double row 8 position (kit)

FOR PC23s WITH SERIAL NUMBERS 3490 AND BELOW:



FOR PC23s WITH SERIAL NUMBERS 3491 AND ABOVE:



Your program will need to write to a different address for each PC23 used. For example, by writing the following commands to address 768, you can designate axis 1 of this address as the master:

Command	Description
Q2	Enter axis 1 in the time-distance streaming mode
TD10	Set update to once every 10 ms
MSL1XX	Use the axis 1 internal clock
SDFF00	Loop an infinite number of times
SD8064	Move CW 100 steps
SDFF80	End loop

These commands will cause axis 1 to enter the streaming mode and designate it as the master axis. Axes 2 and 3 are not in the streaming mode and no master/slave relationship exists. The update period is set to 10 ms and 100 CW steps will loop indefinitely. Notice that the clock has not been set yet, (with the MSS command).

By writing the following commands to address 772, you can designate axes 1, 2, and 3 of this address as slave axes:

Command	Description
1Q2	Enter axis 1 in the time-distance streaming mode
2Q2	Enter axis 2 in the time-distance streaming mode
3Q2	Enter axis 3 in the time-distance streaming mode
1TD1Ø	Set axis 1 update interval to 10 ms
2TD1Ø	Set axis 2 update interval to 10 ms
3TD1Ø	Set axis 3 update interval to 10 ms
MSLØØØ	Slave axes 1, 2, & 3 to an external clock
SD8Ø648Ø648Ø64	Move all axes 100 CW steps

These commands are identical to the commands given to the master axis, except that these 3 axes are slaved to an external clock, (i.e. the other PC23). Note that if you enter an MSS command to start the master clock of the PC23 at this address (772), nothing would happen because the master clock is on the PC23 at the other address (768).

To start the master clock by writing to address 768, enter the MSS command. This will initiate movement on Axis 1 (at address 768) and axes 1, 2, and 3 at address 772. Send an MSLXXX command to address 768 to stop both the master and slave axes. MSS will start them both moving again: however, true synchronicity will be lost. To maintain true synchronicity, MSLXXX should be sent to both the master and slave addresses. Then, when the MSS command is sent to the master address, they will both be synchronous again.

X-Y Linear Interpolation

To move multiple orthogonal linear axes in a straight line, you must make all the axes start, finish accelerating, start decelerating, and stop, in a synchronized fashion.

The simplest case involves producing a 45° angle line of movement with two axes. Both axes (X and Y) are given the same velocity, acceleration, and distance parameters (*to produce other angles, these three parameters must be proportionally scaled*).

Typically, the task is to derive appropriate move parameters to get from the current location to a new location, where each position is specified by a set of *Cartesian coordinates*. Linear acceleration and velocity are specified.

In the following example, the incremental distance parameter for each axis is the difference between the target position coordinate and the current position coordinate for that axis. The ratio of incremental distance for one axis to that of the next establishes the ratio of the respective accelerations and velocities. Linear acceleration and velocity is

the *vector sum* of these components. The *Pythagorean theorem* provides the formula for calculating the velocity.

Example

The two-axis X-Y positioning system must move from its current position to a new position at a linear speed of 1 inch/second (V_L = 1). A motor velocity of 4 rps translates to 1 inch/sec (ips) on both axes. Acceleration on any axis must not exceed 100 rps^2. The following procedure is used to determine the move parameters:

Current Position: X_0 = -60,000 **Target Position:** X_1 = 180,000
Y_0 = 200,000 Y_1 = 20,000

① Determine incremental distances (D) and the ratio:

X axis distance	= $X_1 - X_0$	= +240,000	= **D+240000**
Y axis distance	= $Y_1 - Y_0$	= -180,000	= **D-180000**
Ratio	= Y/X	= -0.75	

② Determine the velocity (V) settings:

Velocity ratio	= V_Y	= $0.75(V_X)$	
Pythagoras	= V_L^2	= $V_X^2 + V_Y^2$	
Substitution	= V_L^2	= $V_X^2 + (0.75(V_X))^2 = 1.56(V_X^2)$	
V_X^2	= $V_L^2/1.56$	= 0.8006ips	= 3.20rps = **V3.20**
V_Y	= $0.75(V_X)$	= 0.6004ips	= 2.40rps = **V2.40**

③ Determine the acceleration (A) settings:

Acceleration ratio = A_Y = $0.75(A_X)$

A_X = 100 rps^2 = **A100**

A_Y = 75 rps^2 = **A75**

(Specific proportional acceleration values may be calculated in the same way the velocity is calculated above)

④ Enter the commands derived from steps 1 through 3 above:

Command	Description
1A100	Set acceleration on axis 1 to 100 rps^2
1V3.024	Set velocity on axis 1 to 3.024 rps
1D240000	Set distance on axis 1 to 240,000 steps CW
2A75	Set acceleration on axis 2 to 75 rps^2
2V2.268	Set velocity on axis 2 to 2.268 rps
2D-180000	Set distance on axis 2 to 180,000 steps CCW

Once these commands are entered, the PC23 is ready to make the synchronous move (by issuing the G12 synchronous Go command) subject to the requirements discussed below.

When polar coordinates are used, the distance to the endpoint (radius) and the angle from the X axis are known. In this case, the distance (D) for the X axis is equal to the radius multiplied by the cosine of the angle. The distance for the Y axis is equal to the radius multiplied by the sine of the angle. The X and Y velocity and acceleration parameters may be derived from linear parameters in the same way.

Using Interrupts

The PC23 has one interrupt signal available to be activated upon specific conditions. This interrupt signal can be directed at either of two interrupt request lines, IRQ3 or IRQ4. Jumper 27/28, located just above the edge connector on the PC23 indexer card, determines which interrupt request line is activated. Installing a jumper on the JU27 pins selects IRQ3, and installing a jumper on the JU28 pins selects IRQ4.

You, the user, are responsible to enable the 8259 interrupt controller chip in the PC to accept the interrupt request. You must also set the corresponding interrupt vectors to accommodate the interrupt request.

8259 Interrupt Controller Chip

Use the following procedure to enable the 8259 interrupt controller chip:

① Read in the current interrupt mask register (IMR) located at the I/O address (21 HEX).

② Clear the appropriate bit to enable the interrupt hardware (IRQ3 = bit #3, IRQ4 = bit #4).

③ Write the IMR back to the 8259 chip.

```
unsigned char int_8259;              /* Byte to read IMR into */
int_8259 = inportb(0x21);            /* Read IMR for current settings */
int_8259 = int_8259 & 0xF7;          /* Clear bit 3 */
outportb(0x21,int_8259);             /* Write new IMR */
```

Interrupt Vector

The IRQ3 interrupt corresponds to the 11th interrupt vector. Since each vector is 4 bytes long, the user must modify the 4 bytes located at memory address ØØØØ:ØØ2C to point to he interrupt service routine. For IRQ4, modify the 4 bytes at memory address ØØØØ:ØØ3Ø.

Usually, IRQ3 and IRQ4 are assigned to the computer's serial ports, COM 2 and COM 1 respectively. Therefore, when making changes to these vectors, **take care to avoid conflicts with communication cards that may require these vectors**.

The following is an example of changing the interrupt vector for IRQ3:

```
struct address { char far *p; };
struct address far *addr = (struct address far *) 44;/* 2c HEX = 44 decimal */
addr->p = (char far *) pc23_irq3;     /* pc23_irq3 is the name of the interrupt
                                         service routine */
```

Interrupt Commands

You can activate the PC23's interrupt output based on several conditions for each axis. The `QS` command allows you to enable or disable the *interrupt-on-condition* functions. No hardware interrupt will be generated until a specific `QS` command has been issued and the corresponding condition is met. The possible interrupt-on-condition functions are as follows:

- Interrupt on Trigger #1 high (`QSA` command)
- Interrupt on Move Complete (`QSB` command)
- Interrupt on Encountered Limit Switch (`QSD` command)
- Interrupt on Ready to Respond (`QSE` command)
- Interrupt on Command Buffer Full (`QSG` command)
- Interrupt on Motor Stall (`QSH` command)

The `QR` command reports which interrupt-on-condition functions have been selected with the `QSA` through `QSH` commands. The `QI` command reports which interruptible conditions are active. *The `QI` command always shows a response to in position (`QSB1`), even if the interrupt is not enabled (`QSBØ`).*

You can take advantage of the PC23's interrupt capabilities without actually generating any host interrupts. This is done by arming whatever interrupt conditions are desired for each axis, and then polling the status byte (SB) to see if the *Interrupt Active* bit (bit #6) is set. Then, each axis must be polled with the `QI` command to determine the source of the signal.

Clearing the Interrupt Signal

Once an interrupt signal is generated, two steps are required to remove that signal. First, the hardware device (*latch*) which holds the signal must be cleared. This calls for toggling the *Reset Interrupt Output* bit (bit #6) in the control byte (CB). Second, the host must acknowledge the interrupt. This calls for toggling the PC23's Interrupt Acknowledge bit (bit #3) in the CB.

The following is example code used to clear an interrupt:

```
int pc23addr = 0x300;                    /* HEX 300 = decimal 768 */
int i;
outportb(pc23addr+1, 0x20);              /* clear bit 6 to restore hardware
                                            interrupt */
for(i=1;i<200;i++);                      /* time delay for strobing bit 6 low */
outportb(pc23addr+1, 0x68);              /* set bit 4 to acknowledge interrupt */
while((inportb(pc23addr+1) & 0x40));     /* wait for interrupt signal to go low */
outportb(pc23addr+1, 0x60);              /* normal state of CB */
```

When your interrupt service routine is complete, remember to send an EOI (end-of-interrupt) signal to the 8259 chip, as exemplified below:

```
outportb(0x20, 0x20); /* clear interrupt from 8259 chip */
```

C H A P T E R ⑤

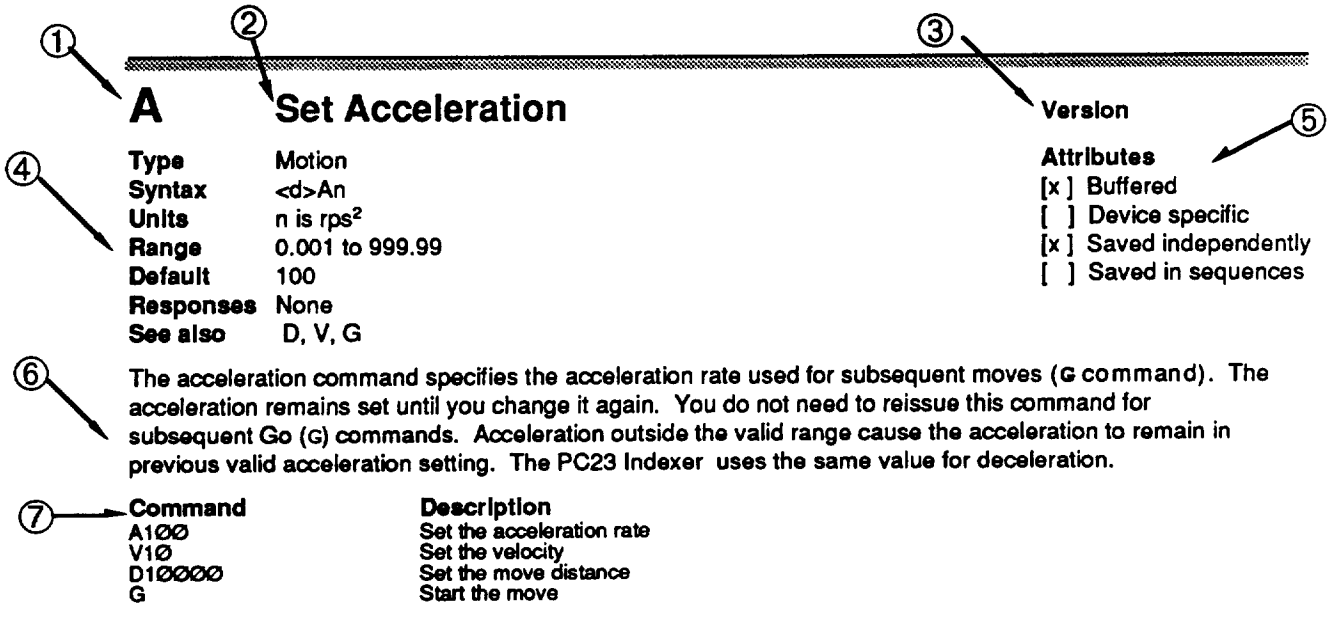
Software Reference

The information in this chapter will enable you to:

- Identify the four types of commands in Compumotor's X-Series Language
- Use this chapter as a reference for the function, range, default, and sample use of each command

Command Format Description

The following section describes the format of the command descriptions used in this chapter. The numbered arrows refer to the numbered sections below the drawing.



① Command Identifier

The letter or letters used to represent the command.

② Command name

This name used to refer to the command. For example, Acceleration for the A command.

③ Version

The revision of software in the PC23 Indexer when the described command was first introduced or last modified. If the revision level of the software you are using is equal to or greater than the revision level listed here, the command is available in your unit. You can determine the level of software in your PC23 Indexer by issuing the Revision Level (RV) command.

④ Characteristics

The following sections describe the main characteristics of the command.

Type

This portion of the box contains the command's type. The four command types are listed below.

Set-Up: These commands define Set-Up conditions for the application. Set-Up commands include the following types of commands:

- Homing (go home acceleration and velocity, etc.)
- Input/Output (limits, scan time, in-position time, etc.)
- Tuning (servo or position tracking)
- General (set switches, return to factory settings, etc.)

Programming: Programming commands affect programming and program flow. For example, trigger, output, all sequence commands, quote, time delays, pause and continue, enable and front-panel, loop and end loop, line feed, carriage return, and backspace.

Status: Status commands respond (report back) information.

Motion: Motion commands affect motor motion (for example, acceleration, velocity, distance, go home, stop, direction, mode, etc.)

Syntax

This field shows the syntax for the command. PC23 Indexer commands use the following generic syntax: `acspd`

Variable **a** This variable is the device address. If the address is optional it is shown in angle brackets: <d>. Only commands which require the PC23 Indexer to send a response require a device address. All commands may use a device address to designate which unit on a daisy chain the command is intended for.

Variable **c** This variable is the command identifier, which is one or more letters.

Variable **s** This variable represents a sign. A sign is not allowed for all commands. The **s** is not shown in the syntax if not allowed.

Variable **p** This variable represents the parameters the command requires. There may be zero or more parameters. If the number of parameters is zero **n** is not shown in the syntax

Variable **d** This variable is the end of command delimiter. This is always required and is not shown in the following descriptions for clarity. The delimiter may be a space character or a carriage return.

Units

This field describes what unit of measurement the parameter in the command syntax represents.

Range

This is the range of valid values that you can specify for **n** (or any other parameter specified).

Default

The default setting for the command is shown in this box. A command will perform its function with the default setting if you do not provide a value.

Response

The response to the command is shown in this box. Status commands report a condition in the indexer. Status commands do not affect the status they read.

Commands that set parameters report the parameters when the command is issued without a parameter. For example, **A1ØØ** sets the acceleration to 100 rps, but **1A** returns the current setting. ***Note:*** *To receive a response, a device address is required.*

See Also

Commands that are related or similar to the command described are listed here.

⑤ Attributes

Each command has attributes as shown below.

Attributes
[x] Buffered
[ ] Device specific
[x] Independently saved
[ ] Saved in sequences

Buffered

If the Buffered box is checked the command is buffered. If it is not checked the command is acted on immediately. Buffered commands are executed in the order they are received. An internal buffer, or storage area, holds the commands in a queue until the previous command has been executed.

Immediate commands are executed as they are received. Immediate commands are executed even if the command buffer has commands in it. For example, the Stop (**S**) command is immediate. When a Stop command is received the motor is stopped as soon as the command is received. The PC23 Indexer does not process the commands in its command buffer before stopping the motor.

Device specific

If the Device specific box is checked the command requires a device identifier. If it is not checked the command may be used with or without a device identifier. Commands which are device specific are normally Status commands. Device specific commands have a syntax description with a **d** by itself before the command. If it is not device specific the command syntax description has a **<d>** in angle brackets before the command.

Saved always

If the Independently saved box is checked the parameter controlled by the command is always saved. This differs from commands which may only be saved in sequences and those which are never saved. If neither the Saved always nor the Saved in sequences box is checked the command is never saved.

Saved in sequences

If the Saved in sequences box is checked the command will be saved only if it is in a sequence and you issue the Save command (**SV**). If neither the

Saved always nor the Saved in sequences box is checked the command is never saved.

⑥ Description

A description of the command appears in this area along with any special considerations you should know about.

⑦ Example

An example of how to use the command appears in this area. The left column contains the commands you would issue to the PC23 Indexer. The right column contains descriptions of what the commands do in the program.

Alphabetical Command List

A — Set Acceleration

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Motion
Syntax aAn
Units n = rps^2
Range 0.01 to 999.99 (Motor dependant)
Default 100
Response None
See also D, V, G

The Acceleration command specifies the acceleration rate to be used upon executing the next Go (G) command. The acceleration remains set until you change it. You do not need to reissue this command for subsequent G commands. Accelerations outside the valid range cause the acceleration to remain in previous valid A setting.

Command	Description
>MN	Set to Normal mode
>A5	Set acceleration to 5 rps^2
>V0	Set velocity to 10 rps
>D10000	Set distance to 10,000 steps
>G	Execute the move (Go)

AB — Report Analog Voltage (Binary)

Version B3X

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Status
Syntax aABn
Units n = channel
Range 0 - 3
Default None
Response nn (two ASCII characters)
See also AV

Read the analog input voltage on channel number n referenced to analog input ground and respond in binary value. The analog voltage request responds with two bytes. The first byte is the A/D channel number (analog channel 0 through 3). The second byte is the 8 bit number for that channel on the joystick's A/D converter.

The 8 bit number corresponds to values between 0 and 255. Since the analog input voltage ranges between 0-2.5VDC, the binary reading will approximately correspond to the voltage on the input.

This command is very useful for quickly reading the analog input voltage.

Command	Description/Response
AB1	Reports voltage on Channel 1

AV Report Analog Voltage (ASCII)

Version A

Type	Status
Syntax	**aAV**n
Units	None
Range	None
Default	None
Response	CHa:n.nnV
See also	AB

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Read the analog input voltage on a channel number (n) referenced to analog input ground. The response is in ASCII. This command is necessary for joystick setup. After hooking up the joystick, read each channel and verify that Channel 0 - Channel 1 and Channel 2 - Channel 3 are both positive. Refer to the joystick installation procedure for test and calibration.

Command	Response
AVØ	CH0:1.45V

B Report Buffer Status

Version A

Type	Status
Syntax	**aB**
Units	None
Range	None
Default	None
Response	*R or *B
See also	None

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The buffer status command will report the status of the command buffer. The command buffer is 1,000 bytes long. The response to this command is:

*R = More than 31 bytes are free
*B = Less than 32 bytes are free

This command is commonly used when a long series of commands will be loaded remotely. If the buffer size is exceeded, the extra commands will not be received by the Controller.

Command	Response
1B	1:*R (more than 31 bytes of the buffer are free)

C Continue

Version A

Type	Programming
Syntax	**aC**
Units	None
Range	None
Default	None
Response	None
See also	PS, U

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Continue (C) command ends a pause state. It enables your indexer to continue executing buffered commands. After you initiate a pause with the Pause (PS) command or the Pause and Wait for Continue (U) command, you can clear it with a C command. This command is useful when you want to transmit a string of commands before you actually need to execute them.

Command	Description
PS	Pause execution on axis #1 until the indexer receives a C command
MC	Set to Continuous mode
A5	Set acceleration to 5 rps^2
V5	Set velocity to 5 rps
G	Execute the move (Go)
T1Ø	Wait 10 seconds after the move
VØ	Set velocity to zero
G	Decelerate the motor to zero velocity
C	Start executing commands in buffer

CG Set Correction Gain

Version A

Type	Set-Up
Syntax	aCGn
Units	n = gain
Range	0 - 8
Default	8
Response	None
See also	FSB, FSC, CM, MV

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command allows you to set the amount of error (steps) that should be corrected on the initial position maintenance (FSC1 command) correction move (which takes place whenever the motor is stationary). This function is only valid in the Encoder Step mode (FSB1 command).

The percentage of error that the Position Maintenance function will attempt to correct on each correction move is n/8 • 100%. If you set n to 1, the system will correct the error slowly (1/8 of the error is corrected on each try). This type of correction is performed smoothly. If you set n to 8, the system will correct the error faster. However, there may be more overshoot and ringing at the end of this type of correction move.

Command	Description
FSB1	Set to Encoder Step mode
FSC1	Enable position maintenance while the motor is stationary
CG3	The system corrects 3/8 of the final-position error on the initial correction move

CR Carriage Return

Version A

Type	Set-Up
Syntax	aCR
Units	None
Range	None
Default	None
Response	None
See also	None

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Carriage Return (CR) command determines when the indexer reaches a particular point in the execution buffer. When the indexer reaches this command in the buffer, it responds by issuing a carriage return (ASCII 13) over its interface back to the host computer. If you place the CR command after a Go (G) command, it will indicate when a move is complete. If you put the CR command after a Trigger (TR) command, it will indicate when the trigger condition is met.

Command	Description
MN	Set to Normal Mode
MPA	Set to Absolute Position mode
A5	Set acceleration to 5 rps^2
V5	Set Velocity to 5 rps
D5000	Set distance to 5,000 steps
G	Execute the move (Go)
CR	Send a carriage return at the end of the move

D Set Distance

Version A

Type	Motion
Syntax	aDn
Units	n = steps
Range	0 - ±99,999,999
Default	25,000
Response	None
See also	A, V, G, MN

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Distance (D) command defines either the number of steps the motor will move or the absolute position it will seek after a Go (G) command is entered. In incremental mode (MPI or FSA0), the value set with the D command will be the distance (in steps) the motor will travel on all subsequent G (Go) commands.

In absolute mode (MPA or FSA1), the distance moved by the motor will be the difference between the current motor position and the position (referenced to the zero position) set with the D command The Distance (D) command has no effect on continuous moves (MC).

Command	Description
2MN	Set to Normal mode
2A5	Set acceleration to 5 rps^2
2V10	Set velocity to 10 rps
2D50000	Set distance to 50,000 steps
2G	Execute the move (Go)

DB Set Dead Band

Version A

Type Set-Up
Syntax <a>DBn
Units n = Encoder Counts
Range 0 - 999999
Default 0
Response None
See also DW, FS

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Once the initial move is completed, position maintenance will determine if a position error condition exists. The error is expressed in terms of Encoder Counts. If the error exceeds the DB setting (in either direction) position maintenance will generate a correction move, in the appropriate direction, based on the DGn and CMn settings.

Position maintenance will remain active and correction moves will continue to be generated until a FSCØ command is issued by the host.

The maximum allowable DB setting is 999999. Any attempt to exceed the maximum will leave the previous setting unchanged.

DPA Display Position Actual

Version A

Type Status
Syntax <a>DPA<n>
Units n = steps
Range None
Default None
Response None
See also PR, P, PX, W

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Display Position Actual (DPA) command displays the actual position in feedback steps. The function is canceled by sending any single ASCII character to the indexer.

Command	Description
DPA	Continually report the actual position of axis #1
2DPA1	Reports the actual position of axis 2, one time

DW Set Dead Band Window

Version A

Type Set-Up
Syntax <a>DWn
Units n = Motor steps (FSBØ) or Encoder Counts (FSB1)
Range 0 - 999999
Default 250
Response None
See also FSB

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command specifies a backlash deadband. The deadband compensates for mechanical systems with backlash and allows use of the Stall Detection features. Stall detection occurs when the position error exceeds the backlash setting.

Command	Description
FSB1	Enter Encoder Step mode
D25ØØØ	Set distance to 25,000 encoder counts
DW1ØØ	Allow 100 counts of error
G	Execute the move (Go)

ER Set Encoder Resolution

Version A

Type Set-Up
Syntax a**ER**n
Units n = encoder steps
Range 1 - 50,000
Default 4,000
Response None
See also FS commands

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command defines the number of encoder counts per one revolution of the motor. The number of lines on an encoder should be multiplied by 4 (quadrature) to arrive at the correct encoder resolution value per revolution of the motor. In other words, one line of an encoder produces 4 encoder steps.

If you are not sure what the resolution of the encoder is, you may do the following to find the encoder resolution.

① Zero the position counter using the **PZ** command.

② Move the motor 1 revolution in motor mode (**FSBØ**).

③ Read the encoder position (**PX**) command. This reading indicates what the resolution will be.

Repeat this several times and use the average value as the **ER** command.

A 4:1 ratio of motor steps to encoder steps is necessary for proper closed-loop operation. If a lower ratio is used, it may be difficult to tune the position maintenance (Servoing) feature of your indexer.

Command	Description
2ER8ØØØ	Set encoder resolution to 8,000 steps on axis #2.

FR Report Encoder Functions

Version A

Type Status
Syntax a**FR**
Units None
Range 0 = function off, 1 = function on
Default 0
Response a:nnnnnnnn
See also FS commands

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command allows you to request the status of **FS** command functions. The response contains one ASCII digit per function (zero [Ø] or one [1]). The digits (n) correspond to the functions, (A - H, left to right). One (1) means an **FS** function is on. Zero (Ø) means an **FS** function is off.

A *Incremental = OFF (Ø), Absolute = ON (1).* Defines the move distances (D) as either incremental from current position, or as absolute referenced to the absolute 0 position).

B *Motor step mode = OFF (Ø), Encoder step mode = ON (1).* Defines the distance in motor steps or encoder steps.

C *Position Maintenance = OFF (Ø), = ON (1).* Enables position maintenance. This causes the indexer to servo the motor to the desired position if it is not in the correct position at the end of a move, or, if the motor is forced out of position while at rest.

D *Terminate Move on Stall Detect = OFF (Ø), = ON (1).* Instructs the indexer to abort a move if it detects a stall.

E *Turn on Output on Stall Detect = OFF (Ø), = ON (1).* Instructs the indexer to turn an output on if it detects a stall.

F *Multiple axis stop = OFF (Ø), = ON (1).* Instructs the indexer to abort any move if a signal is received on the Trigger #6 input . If output on stall is enabled (**FSE1**), the indexer will also turn on an output when a trigger is seen.

G ***Reserved***

H ***Reserved***

Command	Response
1FR	1:11000000. Axis 1 is in absolute encoder step mode. All other **FS** functions are OFF.

FSA Set Absolute/Incremental Positioning Mode

Version A

Type	Set-Up
Syntax	a**FSA**n
Units	n = function
Range	0 = off, 1 = on
Default	0
Response	None
See also	FR, MPI, MPA, PZ, PR

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command sets the indexer to perform its moves in either absolute or incremental positioning mode.

FSAØ = Incremental mode (equivalent to **MPI** command)
FSA1 = Absolute mode (equivalent to **MPA** command)

In Incremental mode (**FSAØ** or **MPI**), all moves are made with respect to the position at the beginning of the move. This mode is useful for repeating moves of the same distance.

In Absolute mode (**FSA1** or **MPA**), all moves are made with respect to the absolute zero position. The absolute zero position is set to zero when you power up the indexer or execute the Position Zero (PZ) command.

Command	Description
MN	Set to Normal mode
FSA1	Set Indexer to absolute mode
PZ	Reset the absolute counter to zero
A5	Set acceleration to 5 rps^2
V5	Set velocity to 5 rps
D25ØØØ	Move motor to absolute position 25,000
G	Execute the move (Go)
D5ØØØØ	Move motor to absolute position 50,000
G	Execute the move (Go)

The motor moves 25,000 steps, and an additional 25,000 steps to reach the absolute position of 50,000.

FSB Set Motor/Encoder Step Mode

Version A

Type	Set-Up
Syntax	a**FSB**n
Units	n = function
Range	0 = off, 1 on
Default	0
Response	None
See also	D, ER, FR, FSC, MR

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command sets up the indexer to perform moves in either motor steps or encoder steps.

FSBØ = Motor step mode
FSB1 = Encoder step mode

In Motor Step mode, the distance command (D) defines moves in motor steps.

In Encoder Step mode, the distance command defines moves in post-quadrature encoder steps. You must set up the indexer for the correct encoder resolution The Encoder Resolution (ER) command is used to define the post-quadrature encoder resolution.

If you enable encoder step mode (**FSB1**), without having the encoder connected to the PC23, upon receiving a Go (G) command, the motor will drift at low velocity. This drift is a result of the PC23 not receiving encoder pulses properly.

Enabling Encoder Step mode does not guarantee that your moves will position to the exact encoder step commanded. Position maintenance (FSC) must be enabled to activate closed loop servoing.

Command	Description
ER4ØØØ	Set encoder resolution to 4,000 post-quadrature encoder pulses
FSB1	Set to Encoder Step mode
A5	Set acceleration to 5 rps^2
V5	Set velocity to 5 rps
D4ØØØ	Set distance to 4,000 encoder steps
G	Execute the move (Go)

FSC Enable Position Maintenance

Version A

Type Set-Up
Syntax aFSCn
Units n = function
Range 0 = off, 1 = on
Default 0
Response None
See also ER, FR, FSB, FSD, CM

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command enables and disables the position maintenance function.

FSC1 = Enable Position Maintenance
FSCØ = Disable Position Maintenance

Enabling position maintenance will cause the indexer to servo the motor until the correct encoder position is achieved. This occurs at the end of a move (if the final position is incorrect) or any time the indexer senses a change in position while the motor is at zero velocity. You must have an encoder connected, and set the indexer in Encoder Step mode (FSB1) in order to enable position maintenance.

If you enabled position maintenance (FSC1) and the motor drifts, the encoder may not be connected properly. The motor will drift at 0.1 rps per revolution.

CAUTION

If you are making a move with position maintenance enabled and the encoder is disconnected, the PC23 will continue to output pulses trying to find the desired position. Consequently, the PC23 could conceivably output pulses forever, or until a limit is encountered. For safety reasons, you should enable Stall Detection (OSE1) and Stop On Stall (FSD1) to ensure that the motor will stop if such a situation occurs.

Command	Description
ER4ØØØ	Set encoder resolution to 4,000
FSB1	Set to encoder step mode
OSE1	Enable stall detection
FSD1	Enable stop on stall
FSC1	Enable position maintenance

FSD Enable Stop on Stall

Version A

Type Set-Up
Syntax aFSDn
Units n = function
Range 0 = off, 1 = on
Default 0
Response None
See also ER, FR, SS, OSE

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command enables and disables the Stop on Stall function.

FSD1 = Enable Stop on Stall
FSDØ = Disable Stop on Stall

Entering FSD1 will cause the indexer to stop the move in progress when a stall is detected. The move is stopped immediately; no deceleration. This command is only valid if stall detection has been enabled (OSE1). It will have no effect otherwise.

Stall detection will work in either motor step mode (FSBØ) or encoder step mode (FSB1).

Entering FSDØ will cause the indexer to attempt to finish the move when a stall is detected, even if the load is jammed.

Command	Description
ER2ØØØ	Set encoder resolution to 2,000 steps/rev
OSE1	Enable stall detect function
FSD1	Enable stop on stall

FSE Enable Output #6 on Stall

Version A

Type	Set-Up
Syntax	a**FSE**n
Units	n = output, on/off
Range	0 or 1
Default	0
Response	None
See also	SS, ER, FR, FSF

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

FSE0 = Do not enable Output #6 on stall
FSE1 = Enable Output #6 on stall

Entering **FSE1** will cause the indexer to turn on Output #6 when a stall is detected. This is useful for signaling other components in you system that a stall has occurred. This command will only be valid if Stall Detect (**OSE1**) has been enabled.

Output #6 is unaffected by a stall when **FSE0** and **OSE1** are entered.

Command	Description
ER4000	Set encoder resolution to 4,000 steps/rev
OSE1	Enable stall detect
FSE1	Turn on output number 4 when a stall is detected

FSF Enable Stop on Trigger #6

Version A

Type	Set-Up
Syntax	a**FSF**n
Units	n = function
Range	0 = off, 1 = on
Default	0
Response	None
See also	FR, TR, TS

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command enables and disables the Stop on Trigger function.

FSF0 = Do not terminate move on Trigger #6
FSF1 = Terminate move when Trigger #6 is high

Entering **FSF1** will cause any move in progress to be stopped whenever Trigger #6 is brought high. For multiple axis application, setting up another unit to turn on Output #6 when it detects a stall, enables the user to implement a multi-axis stop on stall by connecting the output of one axis to the trigger of the other. The move will be decelerated at the maximum acceleration rate.

Entering **FSF0** causes the indexer to treat Trigger #6 as a standard trigger input.

Command	Description
FSF1	Trigger #6 is now dedicated as a remote stop input

G Go

Version A

Type	Motion
Syntax	a**G**
Units	None
Range	None
Default	None
Response	None
See also	A, D, V, FSA, FSB, MA, MC, MN

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Go (**G**) command instructs the motor to make a move using motion parameters that you have previously entered. You do not have to re-enter Acceleration (**A**), Velocity (**V**), Distance (**D**), or the current mode (**MN**, **MA**, or **MC**) commands with each **G** command. In the Incremental Preset mode (**MPI**), a **G** command will initiate the steps you specified with the **D** command.

A **G** command in the Absolute Preset mode (**MPA**) will not cause repeated motion unless you enter a change in distance (**D** command).

In the Continuous mode (**MC**), you only need to enter the Acceleration (**A**) and Velocity (**V**) commands prior to the **G** command. The system ignores the Distance (**D**) command in this mode.

No motor motion will occur until you enter the G command in the Normal (MN), the Continuous (MC), or Alternating (MA) mode.

If motion does not occur with the G command, an activated end-of-travel limit switch may be on, the indexer is waiting for a trigger input (TR), or the indexer is in a pause state (PS, U) and waiting for a continue (C). Check the limit switches.

Command	Description
1MN	Set to Normal mode
FSAØ	Set to Incremental mode
FSBØ	Set Motor Step mode
1A5	Set acceleration to 5 rps^2
1A5	Set acceleration to 5 rps^2
1V5	Set velocity to 5 rps
1D25ØØØ	Set distance to 25,000 steps
1G	Execute the move (Go)
1A1	Set acceleration to 1 rps^2
1G	Execute the move (Go)

Assuming the indexer is in Incremental Preset mode, the motor turns 25,000 steps and repeats the 25,000-step move using the new acceleration value of 1 rps(Total distance moved = 50,000 steps).

Gnnn Go (Synchronized)

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Motion
Syntax aGnnn
Units n = axis
Range None
Default None
Response None
See also D, V, G, MN, I

This command allows you to put a special synchronized go command in each specified buffer. Each buffer will wait until all specified axis buffers have reached the synchronized go command. Each axis should start within 150 µsec of one another. Typically it is used to synchronize moves between more than one axis but it may also be used to synchronize two axes buffers (by issuing a zero distance move). The command may also be used to do simple multi-axis linear interpolation.

You should not send a new G or G123 command until motion is completed.

Command	Description
G12	Synchronize Axis #1 and #2 to start moving together

GA Go Home Acceleration

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Motion
Syntax aGAn
Units n = rps^2
Range 0.01 to 999.99
Default 10
Response None
See also GH, OS

The Go Home Acceleration (GA) command sets the linear acceleration value that the motor will use during any subsequent Go Home (GH) moves.

Command	Description
2GA5	Sets go home acceleration on axis #2 to 5 rps^2
2GH-5	The motor accelerates at 5 rps^2 to 5 rps in the CCW direction and searches for home

GH Go Home

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type	Motion
Syntax	aGHsn
Units	n = rps^2, s = ±
Range	0.001-20 (25,000 step/rev motor)
Default	
Response	None
See also	A, GA, OS, PZ

The Go Home (GH) command instructs the controller to search for the home position, either in the motor step mode or the encoder step mode. When in the motor step mode, the controller looks only at the Home Limit input. It will define Home as the CCW edge of the Home Limit signal (the edge closest to the CCW limit input).

The process is the same in encoder step mode, except the controller also looks for the Z channel input as well as the Home Limit input to be active at the same time. This means that the Z channel pulse must be *enveloped* by the active region of the Home Limit input. When both are active, the indexer defines that position as the home position.

The indexer will reverse direction if an end-of-travel limit is activated while searching for Home; however, if a second end-of-travel limit is encountered in the new direction, the Go Home procedure will stop and the operation will be aborted.

After the GH command is issued, the motor will run in the direction and velocity specified. The motor will keep running after the home switch is activated until it is deactivated. It will then decelerate and reverse direction.

The position counter is set to zero at the conclusion of the go home move.

Command	Description
GH-2Ø	The motor moves in the CCW direction at 20 rps and looks for the Home Limit input to go active

^H Backspace

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type	Programming
Syntax	^H
Units	None
Range	None
Default	None
Response	None
See also	None

This command allows you to delete the last character that you entered (unless it was a delimiter). The ^H command will not prevent execution of an immediate command. A new character may be entered at that position to replace the existing character. (^H indicates that the Ctrl key is held down when the H key is pressed.) This command prompts the indexer to backup one character in the command buffer, regardless of what appears on the terminal. On some terminals, the Ctrl and the left arrow <-- keys produce the same character. *Pressing the delete key does not delete the previous character.*

H Set Direction

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type	Motion
Syntax	<a>H<s>
Units	s = direction
Range	±
Default	+
Response	None
See also	None

The Set Direction (H) command changes or defines the direction of the next move that the system will execute. This command does not effect moves already in progress.

- ❑ H+ Sets move to CW direction
- ❑ H- Sets move to CCW direction
- ❑ H Changes direction from the previous setting
- ❑ CW Direction of motor rotation when viewed from the front face

In preset moves, a Distance (D) command entered after the H command overrides the direction set by the H command. In Continuous mode (MC) only the H command can set the direction of motion.

Command	Description
MN	Set to Normal mode
A5	Set acceleration to 5 rps^2
V5	Set velocity to 5 rps
D25000	Set distance to 25,000 steps
G	Execute the move (Go) in + direction
H	Reverse direction
G	Execute the move (Go) in - direction
MC	Set to Continuous mode
H+	Set direction to + direction
G	Move continuously in + direction

I Load Move Data

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Programming
Syntax <a>I
Units None
Range None
Default None
Response None
See also SSC, Gnnn

The Load Data (I) command allows the indexer to precalculate move data sent, so that execution of the move will begin within 10 ms of receiving a Go (G) command. The move profile may be repeatedly executed with only a 10 ms calculation delay until any one of the three move parameters: Acceleration (A), Velocity (V), or Distance (D) is changed.

Without use of the I command, a delay of up to 30 ms per axis will occur before execution of the move.

You must keep in mind that if you issue an Acceleration (A), Velocity (V), and Distance (D) command the PC23 will take up to 30ms to calculate the open-loop move profile. Therefore, the load data will be useful if you can send the I command, and go elsewhere to read data, then come back to execute the Go (G) command.

Command	Description
2MN	Set to Normal mode
2A5	Set acceleration to 5 rps^2
2V10	Set velocity to 10 rps
2I	Load move data (precalculates move profile)
2TR1XXXXX	Wait for trigger #1 to go high
2G	Execute the move (Go)

IO Immediate Output

Version F

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Status
Syntax aIOnnnnnnn
Units None
Range o = off, 1 = on, x = don't care
Default None
Response a:nnnnnnnn
See also O, OS, SS

The Immediate Output (IO) command turns the programmable output bits (POBs) on and off. This can be used for signaling remote controllers, turning on LEDs, or sounding whistles. POB #1 is controlled by the first position after the IO, POB #2 is controlled by the second position, etc.

IS Input Status

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Status
Syntax aIS
Units None
Range None
Default None
Response None
See also LD, TR, TS

This command reports the status of the inputs for the specified axis. The response is 8 characters terminated with a carriage return. The first two characters are the axis number followed by a colon. The

next characters are the status of 2 trigger bits, the CW limit, CCW limit, Home input and Z channels respectively.

Axis #1:	TRIG1, TRIG2, CW Limit 1, CCW Limit 1, HOME1, Z channel 1
Axis #2:	TRIG3, TRIG4, CW Limit 2, CCW Limit 2, HOME2, Z channel 2
Axis #3:	TRIG5, TRIG6, CW Limit 3, CCW Limit 3, HOME3, Z channel 3

Command	Response
1IS	1:000011 Axis 1 has the home and z channel active
2IS	2:100100 Axis 2 reports TRIG3 active and that its CCW limit is active
3IS	3:010010 TRIG6 is active and the HOME switch is active

J Enable/Disable Joystick

Version A

Type Set-Up
Syntax <a>Jn
Units n = function
Range 0 = disable, 1 enable
Default JØ = Immediate
Response None
See also JB, JD, JV, JZ, OSF

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command allows you to enter and exit the Joystick mode.

JØ = Disable Joystick Mode
J1 = Enable Joystick Mode

When you enter the Joystick mode (J1), the command clears the buffer before entering the mode. The joystick is slew-rate limited by the last acceleration command. If the address preceding the command is the first to enter the Joystick mode, the differential signal between analog channel 0 and analog channel 1 will serve as the axis reference. In joystick mode, the last acceleration and velocity rates that you specify will be used to determine both axis' maximum velocity with full deflection and how rapidly the motor will track the joystick movements. High velocities will result in coarse velocity resolution. Compumotor recommends that you set the acceleration rate high to follow the joystick's movement as stiffly as possible.

The first axis to request a joystick receives channels Ø and 1. The second axis to request a joystick receives channels 2 and 3. To ensure proper assignment of an axis to the different channels, wait at least 25 ms between J1 commands.

Command	Description
1J1	Axis 1 uses differential voltage Channels 0 & 1
2J1	Axis 2 uses differential voltage Channels 2 & 3

JB Set Joystick Backlash

Version B

Type Set-Up
Syntax <a>JBn
Units n = steps
Range 0 - 999
Default 0
Response None
See also JB, JD, JV, JZ

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command allows you to set the joystick backlash compensation distance in steps. If you change the direction of the joystick, a corrective move of nnn steps will be made in the new direction at the velocity you specified with the (Joystick Backlash Compensation Velocity (JV) command. This command overcomes the lag time that exists between changing the joystick's direction (when backlash is present) and actually moving in the new direction.

Command	Description
1JV2	Set the maximum backlash velocity to 2 rps
1JB25Ø	Set the backlash corrective move to 250 steps

JD Set Joystick Dead Band

Version B

Type	Set-Up
Syntax	<a>JDn
Units	n = volts
Range	0.000 to 0.500
Default	0.500
Response	None
See also	JB, JD, JV, JZ

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command allows you to set the the dead band value (in volts). You should set the joystick dead band value to a voltage that allows for the mechanical hysteresis in the resistive joystick assembly.

There will be no movement between zero point set by the JZ command and this dead band voltage.

Command	Description
1JZ	Set joystick's 0 point
1JD.Ø5	Set the joystick dead band 50mV above and below the joystick zero point.

JV Set Joystick Backlash Compensation Velocity

Version B

Type	Set-Up
Syntax	<a>JVn
Units	n = rps
Range	00.000 - 99.000
Default	0
Response	None
See also	JB, JD, JZ

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command allows you to set the joystick's backlash compensation peak velocity. This is the velocity at which the motor will travel when it corrects a position error (backlash). You can determine the distance of any joystick backlash move with the JB command.

Command	Description
1JV2	Set maximum backlash velocity to 2 rps
1JB25Ø	Set the joystick backlash move to 250 steps

JZ Set Joystick to Zero

Version B

Type	Set-Up
Syntax	<a>JV
Units	None
Range	None
Default	None
Response	None
See also	JB, JD, JV

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command allows you to initialize the current joystick position as the zero or no movement position. The position must be set with a differential channel voltage (restricted to 1.25 ±0.25V). If you send the command while the voltage is greater than the specified range, the PC23 will generate an error message. Use the following equations to calculate CCW and CW velocity resolution.

$$\text{CCW Velocity Resolution} = \frac{\text{Last Velocity Specified}}{\text{Zero Point} - 0.01}$$

$$\text{CW Velocity Resolution} = \frac{\text{Last Velocity Specified}}{2.5 - \text{Zero Point}}$$

Command	Description
1JZ	Set axis #1 zero velocity point
2JZ	Set axis #2 zero velocity point

K Kill Motion

Version A

Type	Motion
Syntax	<a>K
Units	None
Range	None
Default	None
Response	None
See also	S

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Kill (K) command is an emergency stop command and should only be used as such. This command causes indexing to cease immediately. There is *no* deceleration of the motor. If Kill causes the motor to slip (i.e., large loads at high speed), the load could be driven past limit switches and cause damage to the mechanism and possibly to the operation.

In addition to stopping the motor, the K command will terminate a loop, end a time delay and clear the command buffer.

Command	Description
1A5	Set acceleration on axis #1 to 5 rps^2
1V2	Set velocity to 2 rps
1MC	Set to Continuous mode
1G	Execute the move (Go)
•	
•	
•	
1K	Stop the motor instantly

L Loop

Version A

Type	Programming
Syntax	<a>Ln
Units	n = number of loops
Range	0 = infinite, otherwise: 1 - 4,299,467,294
Default	0
Response	None
See also	C, N, U, Y

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

When you combine the Loop (L) command with the End-of-Loop (N) command, all of the commands between L and N will be repeated the number of times indicated by n. If you enter the L command without a value specified for n, or with a Ø, subsequent commands will be repeated continuously.

The End-of-Loop command prompts the indexer to proceed with further commands after the designated number of loops have been executed. The Stop Loop (Y) command indicates where execution will stop. The Immediate Pause (U) command allows you to temporarily halt loop execution. You can use the Continue (C) command to resume loop execution. Nested loops are allowed up to 8 levels deep.

Command	Description
L5	Loop 5 times on axis #1.
A5	Set acceleration to 5 rps^2
V1Ø	Set velocity to 10 rps
D1ØØØØ	Set distance to 10,000 steps
G	Execute the move (Go)
N	Specifie the above 10,000-step move to be repeated five times

LA Limit Acceleration

Version F

Type	Set-Up
Syntax	<a>LAn
Units	n = rps^2
Range	.01 - 999.99
Default	999.99
Response	None
See also	A, LD

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Limit Acceleration (LA) command allows you to define the deceleration rate that should be used when an end-of-travel limit is encountered. This command is useful if you do not want an abrupt stop upon encountering a limit. However, you should be careful to specify a deceleration rate that will stop the load before it can do any damage. Normally, limit switches are placed so that the motor has room to safely decelerate the load.

Command	Description
LA5Ø	The motor on axis #1 decelerates at 50 rps^2 when it encounters an end-of-travel limit.

LD Limit Disable

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Set-Up
Syntax <a>LDn
Units n = enable/disable limits
Range 0 - 3
Default 3
Response None
See also RA

The Limit Disable (LD) command allows you to enable/disable the end-of-travel limit switch protection. The LDØ condition does not allow the motor to turn without properly installing the limit inputs. If you want motion without wiring the limits, you must issue the LD3 command.

Enable CCW and CW limits	n = 0
Disable CW limits	n = 1
Disable CCW limits	n = 2
Disable CCW and CW limits	n = 3 (Default)

If you wire the limit switches, these switches will be ignored unless you enable the limit switch inputs using the LDØ command.

Command	Description
1LDØ	Enables CW and CCW limits on axis #1.
1LD3	Allows you to make any move, regardless of the limit input state.

MA Set Mode Alternate

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Motion
Syntax <a>MAn
Units None
Range None
Default None
Response None
See also A, D, V, G, MC, MN, SSD

The Mode Alternate (MA) command, like the Mode Normal (MN) and the Mode Continuous (MC) commands, effects the way moves are preformed. In Mode Alternate (MA), a move is made to a position that you define with the Distance (D) command. When the motor reaches the specified distance it reverses direction and returns to the starting position. This cycle will continue until you issue a Stop (S) or Kill (K) command.

The way the motor stops when a Stop (S) command is issued can be set by using the SSD command. The default is so immediately stop.

Command	Description
MA	Set to Alternate mode
A5	Set acceleration to 5 rps^2
V1	Set velocity to 1 rps
D1ØØØ	Set distance to 1,000 steps
G	Execute the move (Go)

The motor makes a positive move 1,000 units in the CW direction. It then reverses direction and moves 1,000 steps in the CCW direction. This motion continues until you issue a Stop (S) or Kill (K) command.

MC Set Mode Continuous

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Motion
Syntax <a>MCn
Units None
Range None
Default None
Response None
See also A, G, H, MA, MN, T, TR, V

The Mode Continuous (MC) command causes subsequent moves to ignore any distance parameter and move continuously. You can clear the MC command with the Move Normal (MN) or the Mode Alternate (MA) command.

The indexer uses the Acceleration (A) and Velocity (V) commands to reach continuous velocity. The direction of the move should be specified with the H+ or H- command. Using the Time Delay (T), Trigger (TR), and Velocity (V) commands, you can achieve basic velocity profiling.

Command	Description
MC	Set to Continuous mode
A5	Set acceleration to 5 rps^2
V5	Set velocity to 5 rps
T2	Move at 5 rps for 2 seconds
VØ	Change velocity to 0.00 rps
H+	Set move to CW
G	Execute the move (Go)

The motor turns CW at 5 rps until it is halted by the Stop (S) command, Kill (K) command, a limit switch, or by a new velocity specification (VØ followed by a G command).

MN Set Mode Normal

Version A

Type	Motion
Syntax	<a>**MN**
Units	None
Range	None
Default	None
Response	None
See also	A, D, V, G, MC, MPI, MPA

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Mode Normal (MN) command sets the system to mode normal. In Mode Normal, the motor moves the distance specified with the distance (D) command. To define the complete move profile, you must define Acceleration (A), Velocity (V), and Distance (D). The MN command is used to change the mode of operation from Mode Continuous (MC) or Mode Alternating (MA) to the preset mode.

Command	Description
2MN	Set to Normal mode on axis #2
2A5	Set acceleration to 5 rps^2
2V5	Set velocity to 5 rps
2D1ØØØ	Set distance to 1,000 steps
2G	Execute the move (Go)

MPA Set Position Absolute Mode

Version A

Type	Set-Up
Syntax	<Ma>**MPA**
Units	None
Range	None
Default	None
Response	None
See also	D, FSA, MC, MN, MPI

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command sets the positioning mode to absolute. In this mode all move distances are referenced to absolute zero. Note that in Mode Absolute (MPA or FSA1), giving two consecutive go (G) commands will cause the motor to move once, since the motor will have achieved its desired absolute position at the end of the first move.

Mode Position Absolute (MPA) is most useful in applications that require moves to specific locations.

You can set the absolute counter to zero by cycling power or issuing a Position Zero (PZ) command.

Command	Description
MN	Set to Normal mode
MPA	Set to Absolute Position mode
A5	Set acceleration to 5 rps^2
PZ	Set absolute counter to zero
V1Ø	Set velocity to 10 rps
D25ØØØ	Set distance to 25,000 steps
G	Move motor to absolute position 25,000
D125ØØ	Set absolute position to +12,500 steps
G	Move motor to absolute position +12,500 steps

MPI Set Position Incremental Mode

Version A

Type Set-Up
Syntax <a>MPI
Units None
Range None
Default None
Response None
See also D, FSA, MN, MPA

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command sets the positioning mode to incremental. In incremental mode, all move distances specified with the Distance (D) command are referenced to the current position. Position Incremental mode (MPI or FSA0) is useful in applications that require repetitive movements, such as feed-to-length applications.

Command	Description
1MN	Set to Normal mode
1MPI	Set to Positioning Incremental mode
1A5	Set acceleration to 5 rps^2
1V10	Set velocity to 10 rps
1D10000	Set distance of move to 10,000 steps
1G	Move 10,000 steps CW
1G	Move another 10,000 steps CW

MR Select Motor Resolution

Version A

Type Set-Up
Syntax <a>MRn
Units n = resolution codes
Range 0 - 46
Default 10
Response None
See also A, V, ER

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Motor Resolution (MR) command sets the number of steps per revolution according to the numeric suffix n (see table below). This command allows the indexer to control drives of different resolutions while maintaining the commanded acceleration and velocity.

Motor Resolution	Velocity Max (rps)	Pulse W. (µsec)	PC23 command	Motor Resolution	Velocity Max (rps)	Pulse W. (µsec)	PC23 command
200	160	15	MR0	101600	19	.25	MR24
400	80	15	MR1	63488	15	.50	MR25
800	78	7.5	MR2	76800	13	.50	MR26
1000	100	4	MR3	81920	24	.25	MR27
1600	60	4	MR4	102400	19	.25	MR28
3200	78	2	MR5	126976	15	.25	MR29
5000	100	1	MR6	128000	15	.25	MR30
6400	78	1	MR7	153600	13	.25	MR31
10000	50	1	MR8	163840	12	.25	MR32
21600	23	1	MR9	204800	9.7	.25	MR33
25000	20	1	MR10	253952	7.8	.25	MR34
25400	19.5	1	MR11	256000	7.8	.25	MR35
36000	13.8	1	MR12	307200	1.6	1	MR36
50000	10	1	MR13	327680	3.0	.50	MR37
50800	9.8	1	MR14	409600	2.4	.50	MR38
4096	122	1	MR15	507904	1.9	.50	MR39
12800	39	1	MR16	521000	1.9	.50	MR40
25600	19.5	1	MR17	614400	1.6	.50	MR41
12500	40	1	MR18	655360	3.0	.25	MR42
16384	30	1	MR19	819200	2.4	.25	MR43
20000	25	1	MR20	1024000	1.9	.25	MR44
25000	80	.25	MR21	2000	50	4	MR45
50000	40	.25	MR22	4000	125	1	MR46
100000	20	.25	MR23				

The motor resolution you select does not determine the resolution of the motor. The motor/drive determines the resolution of the system. You use the MR command to match the motor/drive to your system, so that you can program your acceleration (A), and Velocity (V) in revolutions.

Command	Description
MN	Set to Normal mode
MR1	Set motor resolution to 400 steps/rev
A5	Set acceleration to 5 rps^2
V10	Set velocity to 10 rps
D800	Set distance of move to 800 steps
G	Execute the move (Go)

A 400 step per revolution motor/drive will turn 800 steps (two revs) CW at an acceleration of 10 rps^2 and a velocity of 10 rps after the G command.

If this command set is sent to a motor/drive with a resolution of 4,000, the motor will still turn 800 steps (1/5 of a rev). However, the actual acceleration would only be 0.5 rps^2 and the actual velocity would only be 1 rps. Indexer resolution and motor/drive resolution must match to get the commanded velocity and acceleration. **This command does NOT affect distance.**

MSL — Identify Clock Source for Timed Data Streaming Mode

Version A

Type Set-Up
Syntax <a>**MSL**$n_1n_2n_3$
Units Value for each
Range 0, 1, 2, 3, or x
Default None
Response None
See also MSS, TD, SD, Q0, Q2, Q3

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Identify Clock Source for Timed Data Streaming Mode (MSL) command sets the Source/Receiver relationship for clock source for each axis in the timed data streaming mode. The following table defines the relationship for each axis. Any axis not in timed data streaming mode can still receive and execute other motion commands.

n	Value of n	Mode
n_1	Ø	Axis #1 is a slave and uses the external rmclk
n_1	1	Axis #1 is a master and uses its internal rmclk
n_1	2	Axis #1 is a slave and uses axis #2's rmclk
n_1	3	Axis #1 is a slave and uses axis #3's rmclk
n_1	X	The X axis is not in the timed data streaming mode and no master/slave specification is necessary
n_2	Ø	Axis #2 is a slave and uses the external rmclk
n_2	1	Axis #2 is a slave and uses axis #1's rmclk
n_2	2	Axis #2 is a master and uses its internal rmclk
n_2	3	Axis #2 is a slave and uses axis #3's rmclk
n_2	X	Axis #2 is not in the timed data streaming mode and no master/slave specification is necessary
n_3	Ø	Axis #3 is a slave and uses the external rmclk
n_3	1	Axis #3 is a slave and uses axis #1's rmclk
n_3	2	Axis #3 is a slave and uses its internal rmclk
n_3	3	Axis #3 is a master and uses axis #3's rmclk
n_3	X	Axis #3 is not in the timed data streaming mode and no master/slave specification is necessary

Command	Description
MSL12X	Axes #1 and #2 are in Timed Data Streaming mode. Both are masters, Axis #3 is independent.
MSLØXX	Axis #1 is in Timed Data Streaming modeAxis #1 is a slave to the external RMCLK.

MSS — Start Master Clock for Timed Data Streaming Mode

Version A

Type Programming
Syntax <a>**MSS**
Units None
Range None
Default None
Response None
See also MSL, TD, SD, Q0, Q2, Q3

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Start Master Clock (MSS) command will cause the clocks on any master axes to start (Initiate Motion). If Profiling (SD) commands are stored in the buffer, entering the MSS command causes execution of these moves. Subsequent Streaming Data (SD) commands will be executed sequentially in a buffered manner. Care must be taken to not overflow these buffers. If the buffer is empty, no motion will result during the particular update interval.

Command	Description
1Q2	Set axis #1 to Time Distance mode
1TDØ6	Set update interval to 6 ms
MSL1XX	Set axis #1 as a Master, using its internal rate multiplier clock. Axis #2 and #3 are not in the time distance mode.
SD8Ø28	Set distance to be moved during the first update interval.
SD8Ø45	Set distance to be moved during the next update interval
SD8Ø45	Set distance to be moved during the next update interval.
SD8Ø28	Set distance to be moved during the next update interval.
SD8Ø1D	Set distance to be moved during the next update interval.
MSS	Start Master Clock

MV Set Maximum Correction Velocity

Version A

Type Set-Up
Syntax <a>MVn
Units n = rps
Range 0.01 - 20.00
Default 0.2
Response None
See also CG, CM, ER, FS

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

If the motor is running with the encoder feedback (FSB1), and position maintenance is turned on FSC1, this command will specify the maximum velocity for correction moves.

The correction move is performed after the preset move is finished, and only when the motor is positioned outside the dead band.

Command	Description
MN	Set to Normal mode
MV2	Set the maximum correction velocity to 2 rps
FSB1	Set motion in encoder mode
FSC1	Set position maintenance on

N End of Loop

Version A

Type Programming
Syntax <a>N
Units None
Range None
Default None
Response None
See also C, L, PS, Y

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command marks the end of loop. Use this command in conjunction with the Loop (L) command. All buffered commands that you enter between the L and N commands are executed as many times as the number that you enter following the L command. You may nest loops 8 levels deep.

Command	Description
PS	Pause the execution of buffered commands
MN	Set to Normal mode
A5	Set acceleration to 5 rps^2
V5	Set velocity to 5 rps
D1ØØØØ	Set move distance to 10,000 steps
L5	Loop the following commands five times
G	Execute the move (Go)
N	End the loop
C	Clear pause and executes all the buffered commands

O Output

Version A

Type Programming
Syntax <a>Onnnnnn
Units n = output
Range 0 = off, 1 = on, x = don't change
Default None
Response None
See also IO, FSE

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Output (o) command turns the programmable output bits (POBs) on and off. This is used for signaling remote controllers, turning on LEDs, or sounding whistles. The output can indicate that the motor is in position, about to begin its move, or is at constant velocity, etc. POB #1 is controlled by the first position after the o, POB #2 is controlled by the second position, etc.

Command	Description
A5	Set acceleration to 5 rps^2
V5	Set velocity to 5 rps
D2ØØØØ	Set move distance to 20,000 steps
OØ1XXXX	Set programmable output 1 off and output 2 on (outputs 3 through 6 are the same)
G	Execute the move (Go)
OØØXXXX	After the move ends, turn off output 2

O,D Output on the Fly

Version

Type	Programmable
Syntax	<a>Op,Dn (see below)
Units	see below
Range	see below
Default	None
Response	None
See also	O, D

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Syntax: **<axis>Opppppp,Dnnnnnn Where p = bit pattern and n = distance**
Units: **p = O (off), 1 = (on), or X (don't change) for each of the programmable output bits. n = steps**
Range: **<a> = axis 1, 2, or 3.**
p = any combination of Ø's, 1's, or x's for a total of 6 bits.

The O,D command allows the user to program a specified bit pattern to appear on the outputs, at a specified distance.

Command	Description
A5	Set acceleration to 5 rps^2
V5	Set velocity to 5 rps
D25ØØØ	Set distance to 25,000 steps
O101010	Output the pattern 101010
OXXXXX1,D15ØØØ	Change the output LSB to a 1 at 15,000 steps
G	Execute the move (Go)
O1XXXXX	Change the MSB to a 1 when the move is complete

The pattern 101010 will appear and remain on the outputs, prior to the move. When 15,000 steps have been generated the output pattern will change to 101011. When the move is complete, the pattern will change to 001011.

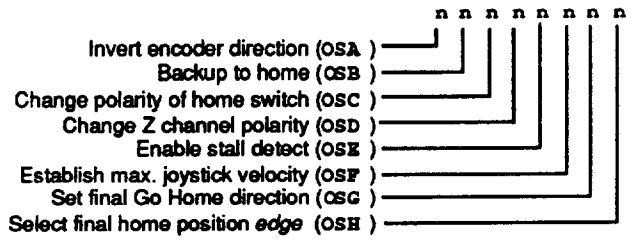
OR Report Function Set-Ups

Version A

Type	Status
Syntax	aOR
Units	None
Range	None
Default	None
Response	a:nnnnnnnn (a = axis number, n = 0, 1)
See also	OS

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command results in a report of which software switches have been set by the OS command. The reply is eight digits. This command reports OSA through OSF setup status in binary format.



The digits (n) correspond to OSA through OSH commands (left to right). One (1) means an OS function is on.

OSA: Encoder Direction = Normal (Ø), = Inverted (1).
OSB: Backup to Home = Disabled (Ø), = Enabled (1).
OSC: Active State of Home Input = High (Ø), = Low (1).
OSD: Active State of Z Channel = High (Ø), = Low (1).
OSE: Enable Stall Detect = Disabled (Ø), = Enabled (1).
OSF: Set Max. Joystick Velocity = Use max. velocity (Ø), = Use last specified velocity (1).
OSG: Final Go Home Direction = CW (Ø), = CCW (1).
OSH: Select Final Home Position Edge = CW (Ø), = CCW (1).

Command	Description
1OSA1	Inverts the encoder direction for axis #1.
1OR	Response is 1:1ØØØØØØØ

OSA Set Encoder Direction

Version A

Type	Set-Up
Syntax	<a>OSAn
Units	None
Range	0 or 1
Default	0
Response	None
See also	FSB, OR

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Entering OSA1 inverts the up/down count direction of the encoder. It causes the motor to expect encoder pulses back in the opposite direction of the motor pulses being sent out. This is useful if the encoder is on the load and is turning in the opposite direction of the motor due to the mechanical configuration of the system. A typical symptom of the encoder counting in the opposite direction is, if the motor drifts at a low velocity in encoder mode (FSB1).

Entering OSAØ will change the up/down count of the encoder back to normal if it has been inverted with the OSA1 command.

Command	Description
1OSA1	Invert the encoder direction for axis #1.
FSB1	Enable encoder mode

OSB Backup to Home Switch

Version A

Type	Set-Up
Syntax	<a>OSBn
Units	None
Range	0, 1
Default	1
Response	None
See also	FS, GH, OR

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

OSBØ = Do not back up to home switch
OSB1 = Back up to home switch

If this function is disabled (OSBØ), the PC23 will consider the motor at Home if the home input is active at the end of deceleration after encountering the active edge of Home region. If this function is enabled (OSB1), the PC23 will decelerate the motor to a stop after encountering the active edge of the Home region, and then move the motor in the opposite direction of the initial Go Home move at .1 rev/sec until the active edge of the Home region is encountered. The PC23 will then consider the motor at Home. This will occur regardless of whether or not the home input is active at the end of the deceleration of the initial Go Home move.

OSC Define Active State of Home Switch

Version A

Type	Set-Up
Syntax	<a>OSCn
Units	n = active state, high/low
Range	0 or 1
Default	0
Response	None
See also	GH, OSD, OR

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

OSCØ = Active state of home input is high
OSC1 = Active state of home input is low

This command inverts the active state of the home input. It enables you to use either a normally closed or a normally open switch for homing. OSCØ requires that a normally closed switch be connected to the home limit input.

OSC1 requires that a normally open switch be connected to the home limit input.

Command	Description
OSC1	Set the active state of the home input to low

OSD Define Active State of Z Channel

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Set-Up
Syntax <a>**OSD**n
Units n = active state high/low
Range 0 or 1
Default 0
Response None
See also GH, OSC, OR

OSDØ = Active state of Z Channel is high
OSD1 = Active state of Z Channel is low

This command inverts the active state of the encoder Z channel input. OSDØ requires that the Z Channel input be an active high input. OSD1 requires that the encoder Z Channel input be an active low.

The Z channel input can be used for encoder based homing (GH) only. In encoder mode, you must have the home switch and the Z channel active at the same time to successfully home the motor.

Command	Description
OSD1	Set the active state of the encoder Z Channel to low

OSE Enable Stall Detect

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Set-Up
Syntax <A>**OSE**n
Units n = enable/disable
Range 0 or 1
Default 0
Response None
See also FSB1, FSC1, FSD, FDE, OR QSH

OSEØ = Stall detect disabled
OSE1 = Stall detect enabled

When enabled (OES1), a stall is detected if the indexer receives no encoder pulses after moving the motor 1/50th of a rev (corresponds to 1 mechanical pole of a 50-pole motor). Thus, when the indexer and motor/drive system resolution is 25,000 steps/rev, 500 steps (25,000 ÷ 50) are output before a stall is detected.

To stop motor on a stall (FSD1), or to turn on output on stall (FSE1), enable stall detection (OSE1).

Command	Description
OSEØ	Disable Stall Detection

OSF Establish Maximum Joystick Velocity

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Set-Up
Syntax <a>**OSF**n
Units None
Range 0 or 1
Default 1
Response None
See also OR, V

OSFØ = Use the maximum system velocity as the joystick maximum velocity
OSF1 = Use the previously defined velocity (V) as the joystick maximum velocity

OSG Set Final Go Home Direction

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Set-Up
Syntax <a>**OSG**n
Units None
Range 0, 1
Default 1
Response None
See also GH, OR, OSB, OSC

OSGØ = Sets the final portion of the go home move sequence to CW
OSG1 = Sets the final portion of the go home move sequence to CCW

OSH Reference Edge of Home Switch

Version A

Type	Set-Up
Syntax	<a>ODBn
Units	None
Range	0, 1
Default	1
Response	None
See also	OR, OSG

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

OSB0 = Selects the CW side of the Home signal as the *edge* on which the final approach will stop
OSB1 = Selects the CCW side of the home signal as the *edge* on which the final approach will stop

The CW edge of the Home switch is defined as the first switch transition seen by the indexer when traveling off of the CW limit in the CCW direction. If n = 1, the CCW edge of the Home switch will be referenced as the Home position. The CCW edge of the Home switch is defined as the first switch transition seen by the indexer when traveling off of the CCW limit in the CW direction.

P Report Incremental Position

Version A

Type	Status
Syntax	aP
Units	None
Range	None
Default	None
Response	a:±nnnnnnnn (n = 0 - 9)
See also	PB, PR, PX, PXB, W1, W3

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Reports incremental distance traveled during the last move in decimal format. The command reports positions in an 8-digit number preceded by sign (+/-), and followed by a carriage return. The range for the position report is 0 - ±99,999,999. If the indexer is in the encoder mode, the report is in encoder steps. If the indexer is in motor mode, the report is in motor steps.

Command	Description
MN	Set to Normal mode
FSB0	Set to Motor Step mode
A10	Set acceleration to 10 rps^2
V5	Set velocity to 5 rps
D25000	Set move distance to 25,000 steps
G	Execute the move (Go)
1P	Request Position Response = 1:+00025000

PB Report Incremental Position (Binary)

Version A

Type	Status
Syntax	aPB
Units	None
Range	None
Default	None
Response	nnnnn
See also	W1, W3, P, PR, PX, PXB

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command reports back the encoder's (in FSB1 mode) or the motor's (in FSB0 mode) incremental position in binary mode. The binary response corresponds to a decimal range of 0 - ±2,147,483,647. The response format is five bytes ($n_1n_2n_3n_4n_5$). The first byte (n_1) is the axis number (Axis 1 - 3). The next four bytes ($n_2n_3n_4n_5$) are the most significant bit to the least significant bit in 2's compliment notation. These four bytes give the 32-bit encoder position.

Converting 2's Compliment to Decimal

If the most significant byte is in the range 00 - 7F, the result is positive. If the most significant byte is in the range 80 - FF, the result is negative.

For positive results, use the following conversion procedure:

① Convert each byte (ASCII character) to decimal.

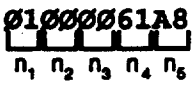
② Multiply each decimal number by the following:

n_2 x 16,777,216
n_3 x 65,536

n_4 x 256
n_5 x 1

③ Add together the products of step 2.

Response from the 1PB command is as follows:



① n_2 = ØØ hex = ØØ decimal
n_3 = ØØ hex = ØØ decimal
n_4 = 61 hex = 97 decimal
n_5 = A8 hex = 168 decimal

② n_2 (ØØ) x 16,777,216 = Ø
n_3 (ØØ) x 65,536 = Ø
n_4 (97) x 256 = 24,832
n_5 (168) x 1 = 168

③ Ø + Ø + 24,832 + 168 = 25,000. Thus, the last move was 25,000 steps in the CW direction.

Binary Approach

① Convert the hexadecimal response to binary form.
② Complement the binary number.
③ Add 1 to the binary number.
④ Convert the binary result to a decimal value.

The response from the 1PB command is Ø1EFFE271Ø. Ø1 is the axis number and EFFE271Ø is the negative incremental position report.

1	2	3	4
E = 1110	1110 = ØØØ1	ØØØ1 = ØØØ1	ØØØ1 = 1
F = 1111	1111 = ØØØØ	ØØØØ = ØØØØ	ØØØØ = Ø
F = 1111	1111 = ØØØØ	ØØØØ = ØØØØ	ØØØØ = Ø
E = 1110	1110 = ØØØ1	ØØØ1 = ØØØ1	ØØØ1 = 1
2 = ØØ1Ø	ØØ1Ø = 11Ø1	11Ø1 = 11Ø1	11Ø1 = D
7 = Ø111	Ø111 = 1ØØØ	1ØØØ = 1ØØØ	1ØØØ = 8
1 = ØØØ1	ØØØ1 = 1110	1110 = 1111	1110 = F
Ø = ØØØØ	ØØØØ = 1111	1111 = ØØØØ	1111 = Ø

1ØØ1D8FØ hex = 268,556,528 decimal. The last move was 268,556,528 steps in the CCW direction.

Computer Approach

① Convert the hexadecimal number to a decimal number.
② Subtract 168 (= 232 = 4,294,967,296) from the decimal number derived from step 1.

The response from the 1PB command is Ø1EFFE271Ø. Ø1 is the axis number and EFFE271Ø is the negative incremental position report.

① EFFE271Ø hex = 4,026,410,768 decimal
② 4,026,410,768 - 4,294,967,296 = 268,556,528
The last move was 268,556,528 steps in the CCW direction.

PR Report Absolute Position

Version A

Type	Status
Syntax	aPR
Units	None
Range	None
Default	None
Response	a:±nnnnnnnn (n = 0 - 9)
See also	D, MN, MPA, MPI, P, PB, PZ, W

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This is an absolute position counter. It reports motor position with respect to power-up position. The absolute position counter can track up to ± 99,999,999 steps. If the counter is overrun in the relative position mode, the absolute position will be invalid.

If in the encoder mode, position will be reported in encoder steps. If you are in motor mode, position will be reported in motor steps. In preset mode, response to this command will be reported after the move is done. In continuous mode, the response to this command will be reported after the motor reaches constant velocity.

The Position Report (PR) command responds with the cumulative position of the motor with respect to the zero position. The zero position can be defined by the position of the motor after a Position Zero (PZ) command is issued or by the successful completion of a Go Home (GH) command.

This command can only respond when the motor is not being commanded to move. Should you need relative positional information when the motor is moving, see the Report Position Relative to Start of Current Move (W) command.

Command	Description
PZ	Set current position to absolute zero
MN	Set to Normal mode
FSBØ	Set to Motor Step mode
A1Ø	Set acceleration to 10 rps^2
V5	Set velocity to 5 rps
D25ØØØ	Set move distance to 25,000 steps
G	Execute the move (Go)
G	Execute the move (Go)
1PR	Request Position Response = 1:+ØØØ5ØØØØ

The motor on axis #1 will move 50,000 motor steps, then the indexer will report the distance moved since the last PZ command.

PS Pause

Version A

Type	Programming
Syntax	<a>PS
Units	None
Range	None
Default	None
Response	None
See also	C, U

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command pauses execution of a command string or sequence following the Pause (PS) command until the indexer receives a Continue (C) command. This command is useful if you need to enter a complete string of commands before you can execute your other commands.

This command is useful for interactive tests and in synchronizing multiple indexes that have long command strings.

Command	Description
PS	Pause execution of following commands until axis # 1 receives the Continue (C) command
A5	Set acceleration to 5 rps^2
V5	Set velocity to 5 rps
D25ØØØ	Set move distance to 25,000 steps
G	Execute the move (Go)
T2	Delay the move for 2 sec
G	Execute the move (Go)
C	Continue Execution

When axis #1 receives the C command, the motor moves 25,000 steps twice with a two second delay between moves.

PX Report Encoder Absolute Position (ASCII)

Version A

Type	Status
Syntax	aPX
Units	None
Range	None
Default	None
Response	a:±nnnnnnnn (n = 0 - 9)
See also	W1, W3, P, PB, PR, PXB

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command reports encoder position with respect to power-up position. The absolute position counter can track up to ±99,999,999 encoder steps. If the counter is overrun in the relative position mode, the absolute position will be invalid.

Whether in encoder step or motor step mode position will be reported in encoder steps. In preset mode, response to this command will be reported after the move is done. In continuous mode, the response to this command will be reported after the motor reaches constant velocity.

The Position Report (PX) command responds with the cumulative position of the encoder with respect to the zero position. The zero position can be defined by the position of the encoder after a Position Zero (PZ) command is issued or by the successful completion of a Go Home (GH) command.

This command can only respond when the motor is not being commanded to move. Should you need relative positional information when the motor is moving, see the Report Position Relative to Start of Current Move (W) command or the absolute encoder position binary (PXB)command below.

Command	Description
MN	Set to Normal mode
A1Ø	Set acceleration to 10 rps^2
V5	Set velocity to 5 rps
D4ØØØ	Set move distance to 4,000 encoder steps
G	Execute the move (Go)
1PX	Request position of axis 1
	Response = 1:+ØØØØ4ØØØ

Axis 1# moves move 4,000 encoder steps, then the indexer reports the distance that axis #1 just moved.

PXB Report Encoder Absolute Position (Binary)

Version B3

Type	Status
Syntax	aPXB
Units	None
Range	None
Default	None
Response	nnnnn
See also	W1, W3, PB, PR, PX

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command reports back the encoder absolute position in binary mode. The binary response corresponds to a decimal range of 0 - ±2,147,483,647. The response format is five bytes ($n_1n_2n_3n_4n_5$). The first byte (n_1) is the axis number (Axis 1 - 3). The next four bytes ($n_2n_3n_4n_5$) are the most significant bit to the least significant bit in 2's compliment notation. These four bytes give the 32-bit encoder position. Refer to the PB command for procedures and examples of how to convert the binary position report.

PZ Set Zero Position

Version A

Type	Status
Syntax	<a>PZ
Units	None
Range	None
Default	None
Response	None
See also	D, MN, MPI, MPA, PR, GH

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command sets the absolute position counter to zero. When your indexer powers up, the absolute counter sets to zero. After moving the motor, the PZ command is used to reset the absolute counter to zero.

In Absolute Mode (MPA), all the moves will be made with respect to the absolute counter. When you execute this command, the Position Report (PR), Pause (P), and Report Absolute Encoder Position (PX) commands will report the zero position.

Command	Description
PZ	Set the absolute counter to zero
MPA	Set to Preset mode with respect to Absolute zero position
A5	Set acceleration to 5 rps^2
V5	Set velocity to 5 rps
D25ØØ	Set move distance to 2,500 steps
G	Execute the move (Go)
1PR	Report Absolute Position
	Response = 1:+ØØØ25ØØØ
PZ	Set the absolute counter to zero
1PR	Report absolute position
	Response = 1:+ØØØØØØØØ

Q — Complete Current Command and Clear Buffer

Version A

Type Programming
Syntax <a>Q
Units NOne
Range None
Default None
Response None
See also K, S

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Q command completes the current command being executed and clears the remainder of the command buffer on the specified axis.

This command is useful if you do not wish to interrupt the current move, or any other command being executed.

The Kill (K) command will stop any command being executed and clears the remainder of the commands in the buffer on the specific axis.

Command	Description
Q	Completes current command and clears command buffer

QØ — Exit Streaming Mode

Version A

Type Set-Up
Syntax <a>QØ
Units None
Range None
Default None
Response None
See also Q1, Q2, Q3, RM

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The QØ command is an exit command for all streaming modes. The motor will stop when QØ is issued.

Refer to Q1 and Q2 examples.

Q1 — Enter Immediate Velocity Streaming Mode

Version A

Type Set-Up
Syntax <a>Q1
Units None
Range None
Default None
Response None
See also QØ, RM

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Q1 command enters the immediate velocity streaming mode. *The command buffer is cleared and any motion is killed on the specified axis.* Subsequent RM commands will cause an immediate change in motor velocity. Use QØ to exit this mode.

Command	Description
Q1	Enter Velocity Profiling mode
RMØØ11	Go to RM velocity of (11 hex) RM rps
RMØØ55	Go to RM velocity of (55 hex) RM rps
RM01ØØ	Go to RM velocity of (100 hex) RM rps
RMØØ55	Go to RM velocity of (55 hex) RM rps
RMØØ11	Go to RM velocity of (11 hex) RM rps
QØ	Exit Velocity Profiling mode

Motor movement will stop when the QØ command is entered. See RM command for more details.

Q2 Enter Time-Distance Streaming Mode

Version A

Type Set-Up
Syntax <a>Q1
Units None
Range None
Default None
Response None
See also QØ, MSL, MSS, TD, SD

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Enter Time-Distance Streaming mode (Q2) command causes the indexer to enter the Time-Distance Streaming mode for the specified axis. The Time-Distance Streaming mode will interpret values entered with the SD command as the number of motor steps to be output in the update interval specified by the TD command. *The command buffers are cleared and motion is killed on the specified axis when the Q2 command is issued.*

This mode is useful in applications that require multi-axis contouring, synchronization, or custom move profiles.

Refer to *Chapter 4, Application Design*, for a detailed discussion of the Time-Distance Streaming mode.

Q3 Enter Time Velocity Streaming Mode

Version A

Type Set-Up
Syntax <a>Q1
Units None
Range None
Default None
Response None
See also QØ, MSL, MSS, TD, SD

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Enter Time-Velocity Streaming mode (Q3) command causes the indexer to enter the Time-Velocity Streaming mode for the specified axis. The Time-Velocity Streaming mode will interpret values entered with the SD command as the velocity in motor steps to be output in the update interval specified by the TD command. *Note that the command buffers are cleared and motion is killed on the specified axis when the Q3 command is entered.*

Refer to *Chapter 4, Application Design*, for a detailed discussion of the Velocity Streaming mode.

QI Interrupt Status Report

Version A

Type Set-Up
Syntax <a>QI
Units None
Range None
Default None
Response nnnnnnnn (n = 0 or 1)
See also QS

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

QI command indicates any active interrupt conditions. The response is the address followed by 8 bits (A through H) as described in the table below. The bits do not need to be enabled with the QS commands. For example, bit A will be high if trigger 1 is active, whether or not QSA1 was entered.

Bit	Function	Options	Value
A	Trigger 1 active	NO (Ø) or YES (1)	n
B	Move Complete	NO (Ø) or YES (1)	n
C	*Not Used*	Ø	Ø
D	Limit Encountered	NO (Ø) or YES (1)	n
E	Ready to Respond*	Ø	0
F	*Not Used*	Ø	Ø
G	Command Buffer Full	NO (Ø) or YES (1)	n
H	Motor Stall	NO (Ø) or YES (1)	n

*The ODBRDY bit in the status register is set. Refer to *Chapter 4, Application Design*, for more on communication.

Move complete will always display a result, even if the QSB command is not enabled.

Command **Response**
1QI Request source interrupt status for axis #1. Response = 1:Ø1ØØØØØØ

QIB Interrupt Status Report (Binary)

Version A

Type Set-Up
Syntax <a>QIB
Units None
Range None
Default None
Response a:n
See also QI

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Interrupt Status Report (QIB) command responds with two bytes. The first byte is axis number (1-3). The second byte is the binary equivalent of the QI response (range is 0 - 255). The format for the second byte functions are shown below.

Since the response in in the binary format, the user must convert the binary value to ASCII value of you wish to print the value to the monitor.

Function	Options		Value
A Trigger 1 active	NO/YES	(n = 0,1)	n
B Move Complete	NO/YES	(n = 0,1)	n
C Not Used		(0)	0
D Limit Encountered	NO/YES	(n = 0,1)	n
E Ready to Respond*		(0)	0
F Not Used		(0)	0
G Command Buffer Full	NO/YES	(n = 0,1)	n
H Motor Stall	NO/YES	(n = 0,1)	n

*The ODBRDY bit in the status register is set. Refer to *Chapter 4, Application Design* for more on communication.

Move complete will always display a result, even if the QSB command is not enabled.

QR Report QS Command Function Enable Status

Version A

Type Status
Syntax aQR
Units None
Range None
Default None
Response a:nnnnnnnn (n = 0 or 1)
See also QS, QI

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Request for the functions enabled or disabled by the QSA through QSH commands. The reply is eight digits, indicating which signal functions are active, as shown below:

Function	Options		Value
A Trigger 1 high	OFF/ON	(0)	n
B Move Complete	OFF/ON	(n = 0,1)	n
C Not Used		(0)	0
D Limit Encountered	OFF/ON	(n = 0,1)	n
E Ready to Respond		(0)	0
F Not Used		(0)	0
G Transmit on Command Buffer Full	OFF/ON	(n = 0,1)	n
H Motor Stall	OFF/ON	(n = 0,1)	n

QS Interrupt on Signal Commands

Version A

Type Set-Up
Syntax <a>QSxn
Units x = function
Range x = A, B, C, D, E, F, G, H n = on/off
Default None
Response None
See also QR

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The QS commands allow the PC23 to send an interrupt to the host computer when attention is required. Once enabled, QS commands reside in the PC23's background during normal operation. When the condition that prompts the interrupt is met, the interrupt is generated. The interrupt is determined by the host computer via the QI or QIB command. The interrupt will be generated each time the condition occurs, until

the function is disabled (Refer to the parameter n in the description box above. When n = ø (off). When n = 1 (on). If more than one interrupt is pending, the rest are ignored (if one is buffered, the rest are lost).

Refer to *Chapter 4, Application Design*, for information on setting hardware interrupts on the PC bus.

Bit	Function
A	Trigger 1 active
B	Interrupt on Move Complete
C	Not Used
D	Interrupt on Limit Encountered
E	Interrupt on Ready to Respond
F	Not Used
G	Interrupt on Command Buffer Full
H	Interrupt on Motor Stall

QSA Interrupt on Trigger #1 High

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Set-Up
Syntax <a>QSAn
Units n = function on/off
Range 0 or 1
Default 0
Response None
See also TR, TS, QR

QSAø = Do not send interrupt on Trigger #1 being high
QSA1 = Interrupt signal on Trigger #1 being high

Entering QSA1 causes the indexer to transmit an interrupt signal to the host computer whenever trigger #1 goes high. How the interrupt is handled is dependent upon the host computer and your interrupt routine within the host.

QSB Interrupt on Move Complete

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Set-Up
Syntax <a>QSBn
Units n = function on/off
Range 0 or 1
Default 0
Response None
See also QR

QSBø = Do not send interrupt on move complete
QSB1 = Interrupt signal on move complete

Entering QSB1 causes the indexer to transmit an interrupt signal to the host computer whenever a move has been completed. How the interrupt is handled is dependent upon the host computer and your interrupt routine within the host.

QSD Interrupt on Limit Encountered

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Set-Up
Syntax <a>QSDn
Units n = function on/off
Range 0 or 1
Default 0
Response None
See also QR

QSDø = Do not interrupt signal upon reaching a limit
QSD1 = Interrupt signal upon reaching a limit

Entering QSD1 causes the indexer to transmit an interrupt signal to the host computer whenever a CW or CCW limit has been reached. How the interrupt is handled is dependent upon the host computer and your interrupt routine within the host.

QSE Interrupt on Ready to Respond

Version A

Type Set-Up
Syntax <a>QSEn
Units n = function on/off
Range 0 or 1
Default 0
Response None
See also QR

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

QSE0 = Do not interrupt when indexer is ready to respond
QSE1 = Interrupt when indexer is ready to respond

Entering QSE1 causes the indexer to transmit an interrupt signal to the host computer whenever the indexer is ready to respond from a status command. How the interrupt is handled is dependent upon the host computer and your interrupt routine within the host.

QSG Interrupt on Command Buffer Full

Version A

Type Set-Up
Syntax <a>QSGn
Units n = function on/off
Range 0 or 1
Default 0
Response None
See also QR

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

QSG0 = Do not interrupt on buffer full (less than 32 bytes free)
QSG1 = Interrupt on buffer full

Entering QSG1 causes the indexer to transmit an interrupt signal to the host computer whenever the command buffer is full. How the interrupt is handled is dependent upon the host computer and your interrupt routine within the host.

QSH Interrupt on Motor Stall

Version A

Type Set-Up
Syntax <a>QSAn
Units n = function on/off
Range 0 or 1
Default 0
Response None
See also DQ, OSE1, OR

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

QSH0 = Do not interrupt on stall
QSH1 = Interrupt on stall detect

Entering QSH1 causes the indexer to transmit an interrupt signal to the host computer whenever a stall is detected. How the interrupt is handled is dependent upon the host computer and your own program within the host.

This command is functional only if you enable stall detection with the OES1 command. When enabled, a stall is detected if the indexer receives no encoder pulses after moving the motor 1/50th of a rev (corresponds to 1 mechanical pole of a 50-pole motor). Thus, when the indexer and motor/drive system resolution is 25,000 steps/rev, 500 steps (25,000 ÷ 50) are output before a stall is detected.

Entering QSH0 causes no interrupt to be transmitted by the indexer when a stall is detected.

Command	Description
QSH1	An interrupt will be transmitted to the host when a stall occurs.

R Report Indexer Status

Version A

Type	Status
Syntax	aR
Units	None
Range	None
Default	None
Response	*x (x = R, B, S, or C)
See also	RA, RB, RC

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

You can use the Request Indexer Status (R) command to review the status of the indexer. Possible responses are:

Response	Definition
*R	Ready
*S	Ready, Attention Needed
*B	Busy
*C	Busy, Attention Needed

When the indexer is prepared to execute an immediate command, the following conditions delay the execution:

- ❑ Performing a preset move
- ❑ Accelerating/decelerating during a continuous move
- ❑ A time delay is in progress (T command)
- ❑ Paused (PS)
- ❑ Waiting on a Trigger (TR)
- ❑ Going Home
- ❑ Executing a loop

The following conditions cause error responses.

- ❑ A feedback error condition exits.
- ❑ Go home failed
- ❑ Limit has been encountered

You can obtain further details on the error condition with the RA, RB, or RC commands. *Compumotor does not recommended that you use this command in tight polling loops. It may overload the microprocessor. Time delays can alleviate this problem.*

Do not use this command to determine if a move is complete. Use it after a move is complete to determine if other errors or faults exist. Use a buffered status request command such as a CR (Carriage Return) or a programmable output to verify move completion.

Command	Response
1R	1:*R Axis #1 is ready to accept a command—no error conditions require attention

RA Report Limit Switch Status

Version A

Type	Status
Syntax	aR
Units	None
Range	None
Default	None
Response	*x (x = @, A, B, C, D, E, F, G, H, I, J, K, L, M, N, or O)
See also	R, RB, RC, LD

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Limit Switch Status Report (RA) command responds with the status of the end-of-travel limits during the last move as well as the present condition. This is done by responding with one of 16 characters representing the conditions listed below.

Response Character	Last Move Terminated By CW Limit	Last Move Terminated By CCW Limit	Current Move Limited By CW Limit	Current Move Limited By CCW Limit
*@	NO	NO	NO	NO
*A	YES	NO	NO	NO
*B	NO	YES	NO	NO
*C	YES	YES	NO	NO
*D	NO	NO	YES	NO
*E	YES	NO	YES	NO
*F	NO	YES	YES	NO
*G	YES	YES	YES	NO
*H	NO	NO	NO	YES
*I	YES	NO	NO	YES
*J	NO	YES	NO	YES
*K	YES	YES	NO	YES
*L	NO	NO	YES	YES
*M	YES	NO	YES	YES
*N	NO	YES	YES	YES
*O	YES	YES	YES	YES

The RA command is useful when the motor will not move in either or both directions. The report back will indicate whether or not the last move was terminated by one or both end of travel limits.

If you wish to disable the limit inputs, you may issue the LD3 (Disable both limits) command.

Command	Response
2RA	2:*@

This response indicates that the last move on axis 2 was not terminated by a limit and that no limits are currently active.

RB Report Loop, Pause, Shutdown, and Trigger Status

Version A

Type	Status
Syntax	aRB
Units	None
Range	None
Default	None
Response	*x (x = @, A, B, C, D, E, F, G, H, I, J, K, L, M, N, or O)
See also	L, PS, R, RA, RC, ST, TR

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The RB command responds with the status of the command buffer if it is presently executing a Loop (L) or a Shutdown (ST) command. It will also indicate if a Pause (PS) command is being executed or if a Trigger (TR) condition is presently being waited on.

The controller responds with one of 16 different characters, each of which represents one of the conditions listed below.

Response Character	Loop Active	Pause Active	Shutdown Active	Trigger Active
*@	NO	NO	NO	NO
*A	YES	NO	NO	NO
*B	NO	YES	NO	NO
*C	YES	YES	NO	NO
*D	NO	NO	YES	NO
*E	YES	NO	YES	NO
*F	NO	YES	YES	NO
*G	YES	YES	YES	NO
*H	NO	NO	NO	YES
*I	YES	NO	NO	YES
*J	NO	YES	NO	YES
*K	YES	YES	NO	YES
*L	NO	NO	YES	YES
*M	YES	NO	YES	YES
*N	NO	YES	YES	YES
*O	YES	YES	YES	YES

This command is useful to determine the present status of the execution buffer, especially when execution is held up or the response is unclear.

When you send a buffered command and the indexer does not execute the command, you may execute this command to receive a status of the indexer.

Command	Response
1RB	1:*A Axis #1 is currently executing a loop

RC Report Closed Loop and Go Home Status

Version A

Type Status
Syntax aRC
Units None
Range None
Default None
Response *x (x = @, A, B, or C)
See also R, RB, OSE, GH, L, FSC

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Report Closed Loop And Go Home Status (RC) command responds with a character that represents one of the conditions described below.

- ***Homing Function Failure*** — In this condition, the controller reaches both end-of-travel limits or one of several possible Stop commands or conditions. The Go Home motion was concluded, but not at home.
- ***Stall*** — In this condition, the controller detects either a deviation between motor and encoder position that is larger than one pole of the motor while running, or a deviation larger than one pole of the motor plus the backlash parameter following a direction change.

Response Character	Stall Detected	Go Home Unsuccessful	Static Loss Detected
*@	NO	NO	NO
*A	YES	NO	NO
*B	NO	YES	NO
*C	YES	YES	NO
*H	NO	NO	YES
*I	YES	NO	YES
*J	NO	YES	YES
*K	YES	YES	YES

Command	Response
1RC	1:*C

This means that while attempting its last move, axis 1 detected a stall.

RM Rate Multiplier in Immediate Velocity Streaming Mode

Version A

Type Set-Up
Syntax <a>RMxxxx
Units xxxx = rate multiplier value (hexadecimal)
Range 0000 - FFFF
Default 0000
Response None
See also Q1, Q0

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The RM command, followed by 4 hexadecimal digits, represents an immediate velocity setting. The variable nnnn is a hex value (in ASCII format) ranging from 0000 - FFFF which corresponds to 15.259 Hz - 500 kHz in increments of 15.259 Hz. The velocity change is instantaneous. There is no acceleration/ deceleration ramp between velocities. A limit switch-closure stops movement while in the axis is in velocity profiling mode, but does not cause the PC23 to exit Immediate velocity streaming mode. Hex values 0000 - 7FF3 result in CCW rotation. Hex values 8000 to FFF3 result in CW rotation. Hex values 7FF4 - 7FFF and FFF4 - FFFF are special values that you can use to set POBs (refer to Chapter 4, Application Design for detailed information on contouring).

Command	Description
Q1	Enter Velocity Streaming mode on axis #1
RM0666	Jump to 1 rps in the CCW direction
RM0CCC	Jump to 2 rps
RM1332	Jump to 3 rps
RM1998	Jump to 4 rps
RM1332	Jump to 3 rps
RM0CCC	Jump to 2 rps
RM0666	Jump to 1 rps
RM0000	Jump to 0 rps
Q0	Exit Velocity Streaming mode

The motor jumps to 1 rps when you issue the first RM value. The velocity increases by 1 rps when you issue the next three RM commands. The subsequent RM commands decrease the motor velocityin 1 rps increments until the motor stops. The amount of time that elapses between the issuance of each RM command determines the exact motion profile.

RV Report Software Part Number

Version A

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Status
Syntax <a>RV
Units None
Range None
Default None
Response 92-nnnnnn-nn(xn) (n = 0 - 9, x = A - Z)
See also None

The RV command responds with the software part number and its revision level. The response is in the form shown below:

```
92-nnnnnn-nn<xn>
part number <revision level>
```

The part number identifies which product the software is written for, as well as any special features that the software may include. The revision level identifies when the software was written. You may want to record this information in your own records for future use. This information is useful when you consult Parker Compumotor's Applications Engineering Department.

Command	Response
1RV	92-006887-01K

The product is identified by 92-006887. The revision level is identified by 01K.

S Stop

Version A

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Motion
Syntax <a>S
Units None
Range None
Default None
Response None
See also K, SSD, SSH, A

This command decelerates the motor to a stop using the last defined Acceleration (A) command. This command normally clears any remaining commands in the command buffer, unless prevented from doing so by the Save Command Buffer On Stop (SSH1) command. When the SSH1 command is executed, the S command stops only the current move and goes on to the next command in the buffer.

Command	Description
MC	Set to Continuous mode
A1	Set acceleration to 1 rps^2
V1Ø	Set velocity to 10 rps
G	Execute the move (Go)
S	Stop motor—motor comes to 0 rps at a deceleration rate of 1 rps^2

The S command cannot be put into the buffer since it is an immediate command. As soon as the indexer receives the S command, it stops motion.

SA Stop All

Version A

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Motion
Syntax <a>SA
Units None
Range None
Default None
Response None
See also S, K

The Stop All command is equivalent to a 1S, 2S, 3S.

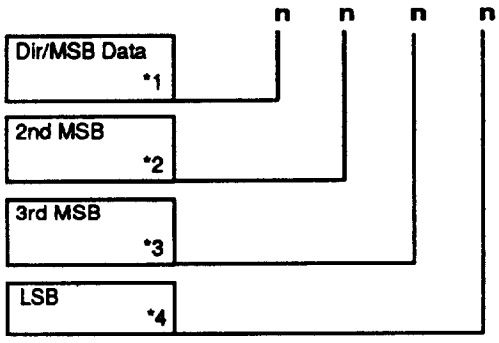
SD Define Timed Data Mode Streaming Data Version A

Type Programming
Syntax <a>SDn
Units n = hex digit up to 3 sets of 4
Range 0 - F
Default None
Response None
See also Q2, Q3, TD, MSL, MSS

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command allows you to define streaming data. In the time-distance (Q2) mode, you can specify the number of steps that the motor will move per interval with this command. In the time-velocity (Q3) mode, you can specify the velocity that the motor will move per interval. You can set the time interval with the TD command.

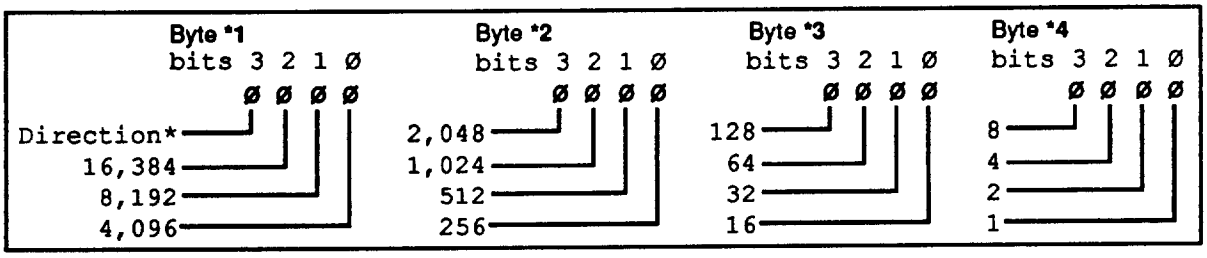
You may define streaming data with one or two or three four-digit hexadecimal numbers. If only one axis is in a timed data streaming mode, only one four-digit hex number is required (SDnnnn). If two axes are in a timed data streaming mode, two four-digit hexadecimal numbers are required (SDnnnnnnnn). If you have a 3 axis system, you need to send 3 four digit hexadecimal numbers (SDnnnnnnnnnnnn) You can define data for all three axes with a single SD command. Each SD command is only executed during a single TD interval.



If you want to define data for one axis in a timed data streaming mode, use the configuration SDnnnn. If you want to define data for two axes in a timed data streaming mode, use the configuration SDnnnnnnnn. The first four hex numbers define data for the lower numbered axis. The next four hex numbers define data for the higher numbered axis.

The function of each byte (*1, *2, *3, and *4) is described below.

*1 sets the direction in which the axis will move. A Ø in the most significant bit (MSB) specifies CCW motion. A 1 in this position indicates CW motion. The remaining three bits of this byte are the 3 MSBs to define the magnitude of distance (Q2 mode) or velocity (Q3 mode). *2 is the most significant *full* byte of segment data. *3 is the next most significant *full* byte of segment data. *4 is the least significant *full* byte of segment data. The weight of each bit in bytes *1, *2, *3, and *4 is as follows:

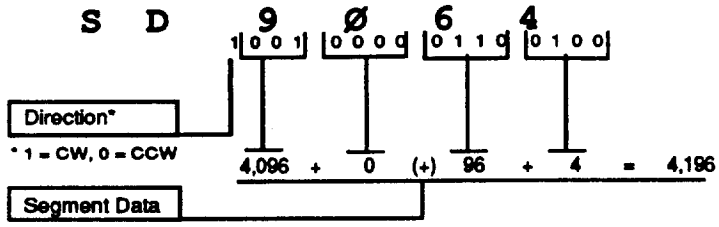


In Q2 mode, this byte indicates the number of steps the motor will move per time interval. In Q3 mode, this byte indicates the velocity that the motor will move per time interval. You can set the time interval with the TD command. In the Q3 mode, the velocity desired per update interval is related to the four hex digits as follows (disregarding the most significant bit which is the direction bit):

```
Vel (in rps for a 25000 step/rev motor) = (nnnn * 15.259)/25000
```

Command
`1SD9064`

Response
Assuming that axis 1 is in a timed data streaming mode, this command specifies a distance of 4,196 CW steps for axis 1. Refer to the graphic depiction below.



Certain SD data points allow you to turn on outputs, wait on triggers, and loop. Refer to Chapter 4, *Application Design*, for more information.

SR Report Configuration Status

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Status
Syntax aSR
Units None
Range None
Default None
Response a:nnnnnnnn (a = axis number, n = 0 or 1)
See also SS

The SR command is a special status request command. This command provides information on the status of up to 5 software switches that you use to turn configuration options on and off. You can set these options with the SS commands.

The table below lists the response functions that you can receive status information on.

Code	Function	On/Off Status
A	*Not Used*	*N/A*
B	*Not Used*	*N/A*
C	Load and Go Mode	Ø = Disabled, 1 = Enabled
D	Alternate Mode Stop	Ø = Cycle End, 1 = Immediate
E	*Not Used*	*N/A*
F	Velocity Range	Ø = Normal, 1 = Low
G	Command Buffer on Limit	Ø = Purge, 1 = Save
H	Command Buffer on Stop	Ø = Purge, 1 = Save

Command	Response
1SR	1:ØØØØØ1ØØ This indicates that axis #1 is in the low-velocity range.

SSD Mode Alternate Stop Mode

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Set-Up
Syntax <a>SSDn
Units n = function (on/off)
Range 0 or 1
Default 0
Response None
See also MA, SR

This command determines the method of stopping when in the MA move mode.

SSDØ = Stop immediately
SSD1 = Stop at the end of the current loop

If you enable SSD1, upon receiving the Stop (s) command, the motor will move back to the starting position then stop motion. If you use the SSDØ command, the motor will decelerate to a stop immediately.

Command	Description
SSD1	Stop at end of loops in mode alternate

SSF Normal/Low Velocity Range

Version A

Type Set-Up
Syntax <a>SSFn
Units n = function (on/off)
Range 0 or 1
Default 0
Response None
See also SR, V

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command sets the velocity range for the Velocity (V) command. The normal range (SSFØ) for a 25,000-step/rev motor is .01 - 20 rps. The low range (SSF1) for a 25,000-step/rev motor is .001 - 2.00 rps.

Command	Description
SSF1	Set low velocity range for axis #1

SSG Clear/Save the Command Buffer on Limit

Version A

Type Set-Up
Syntax <a>SSGn
Units n = function (on/off)
Range 0 or 1
Default 0
Response None
See also LD, SR

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

In most cases, it is desirable that upon activating an end of travel limit input, all motion should cease until the problem causing the over-travel is rectified. This will be assured if all commands pending execution in the command buffer are cleared when hitting a limit. This is the case if SSGØ is specified. If SSG1 is specified and a limit is activated, the current move is aborted, but the remaining commands in the buffer continue to be executed.

Command	Description
SSG1	Save buffer on limit
A5	Set acceleration to 5 rps^2
V5	Set velocity to 5 rps
D25ØØØ	Set distance to 25,000 steps
G	Execute the move (Go)
O11	Turn on outputs 1 and 2

If a limit switch is encountered while executing the move, outputs 1 and 2 will still go on.

SSH Clear/Save the Command Buffer on Stop

Version A

Type Set-Up
Syntax <a>SSHn
Units n = function (on/off)
Range 0 or 1
Default 0
Response None
See also S, SR

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

SSHØ = Clears command buffer on stop
SSH1 = Saves command buffer on stop

In Normal Operation (SSHØ) the Stop (S) command will cause any commands in the command buffer to be cleared. If you select the Save Command Buffer On Stop (SSH1) command,a Stop (S) command will only stop execution of a move in progress. It will not stop execution of any commands that remain in the buffer.

Command	Description
SSHØ	Clear command buffer on stop
A5	Set acceleration to 5 rps^2
V5	Set velocity to 5 rps
D25ØØØ	Set distance to 25,000 steps
L5Ø	Loop 50 Times
G	Execute the move (Go)
T.5	Pause the motor 500 ms
N	End Loop
S	Stop motion

When you issue the S command, the indexer will clear the buffer and stop the move.

ST Shutdown

Version A

Type	Set-Up
Syntax	<a>STn
Units	n = function (on/off)
Range	0 or 1
Default	0
Response	None
See also	None

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Shutdown (ST1) command rapidly decreases the motor current to zero. The system ignores move commands that you issue after the ST1 command. Torque on the motor is not maintained after you issue the ST1 command.

The STØ command rapidly increases the motor current to normal level. Once you restore the current, you can execute moves.

This command is useful for reducing motor heating and allows you to manually position the load. The motor position counter is set to zero when you re-energize the motor using the STØ command.

Most Compumotor drives have a Shutdown Input along with Step and Direction inputs. The ST1 command activates the shutdown input of the drive, disabling the current going through the motor.

Command	Description
ST1	Shuts off current to the motor

T Time Delay

Version A

Type	Programming
Syntax	<a>Tn
Units	n = seconds
Range	0.01 to 999.99
Default	None
Response	None
See also	None

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Time Delay (T) command causes the indexer to wait the number of seconds that you specify before it executes the next command in the buffer. This command is useful whenever you need to delay the motor's actions.

Command	Description
MN	Set to Normal mode
A5	Set acceleration to 5 rps^2
V5	Set velocity to 5 rps
D25ØØØ	Set distance to 25,000 steps
T1Ø	Pause motor movement 10 seconds
G	Execute the move (Go)
T5	Pause the motor for 5 seconds after the move ends
G	Execute the move (Go)

TD Set Time Interval for Timed Data Streaming Mode

Version A

Type	Motion
Syntax	<a>TDnn
Units	nn = ms
Range	02 - 50
Default	10
Response	None
See also	MSL, MSS, SD, QØ, Q2, Q3

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command sets the time interval for execution of segments defined with the SD command. Each SD command will be executed during one time interval only. In Q2 mode, the segment profile will be derived from the time (in milliseconds—ms) specified by nn, where nn = 02 to 50 (2 to 50 ms **in increments of 2 ms**) , and the segment distance specified with an SD command (i.e., the motor will move the distance—in steps—specified with the SD command in the time set with the TD command). In Q3 mode, the segment profile will be derived from the time specified by nn, and the velocity (in steps per second) specified by the SD command. i.e. The motor will achieve the velocity specified with the SD command in the time set with the TD command.

Command	Description
Q2	Enter Time Distance mode
TD04	Sets update time to 4 milliseconds
MSL11X	Axis #1 and axis #2 are in Time Distance mode
SD0000802800	Move axis #2 40 steps within the update time
SD0100002800	Move axis #1 256 steps, axis #2 40 steps within 4 ms
MSS	Start master clock (Upon entering this command, moves defined by SD command will start)

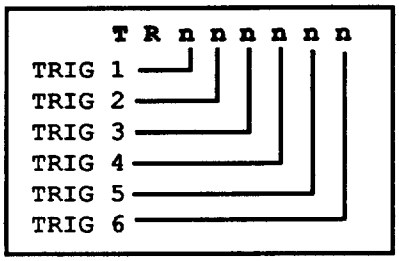
TR Wait for Trigger

Version A

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Programming
Syntax <a>TRnnnnnn
Units n = function
Range (0 = open-on/off, 1 = closed, x = don't care)
Default 0
Response None
See also IS, TS

Triggers are used to synchronize indexer operations with external events. They can be used to implement a handshaking function with other devices. There are six triggers (see below).



When TR command is used in a buffer, the indexer will get to this command and wait until the input pattern is matched before going on to the next command.

Command	Description
TR10XXXX	Wait for input #1 to be grounded and input #2 to be opened before going on to the next command. Inputs #3 - #6 are ignored.
A5	Set acceleration to 5 rps^2
V5	Set velocity to 5 rps
D25000	Set distance to 25,000 steps
G	Execute the move (Go)

TS Report Trigger Input Status

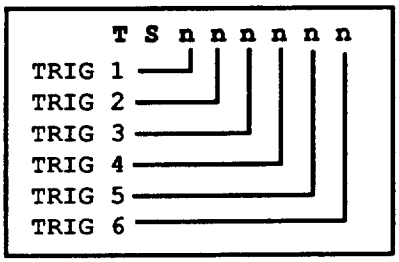
Version A

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type Status
Syntax aTS
Units None
Range None
Default None
Response a:nnnnnn (a = axis number, n = o, 1)
See also IS, TR

The Trigger Status command retrieves the status of the trigger inputs.

- n= 1 (Input is ON)
- n = 0 (Input is OFF)



Since the TS command is immediate, the host controller can determine the status of the trigger inputs at any time, even during execution of other commands. You can use this command to make sure that your trigger pattern is met, when you have issued the Trigger (TR) command.

Command	Response
1TS	1:1Ø1Ø11

Trigger bits 1, 3, 5 & 6 are active. Trigger bits 2 and 4 are inactive.

U Pause and Wait for Continue

Version A

Type Programming
Syntax <a>U
Units None
Range None
Default None
Response None
See also C, PS

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command causes the indexer to complete the move in progress, then wait until it receives a Continue (C) to resume processing. Since the buffer is saved, the indexer continues to execute the program (at the point where it was interrupted). The indexer continues processing when it receives the C command. This command is typically used to stop a machine while it is unattended.

Command	Description
MN	Set to Normal mode
A5	Set acceleration to 5 rps^2
V5	Set velocity to 5 rps
LØ	Loop indefinitely
D256ØØ	Set distance to 25,600 steps
G	Execute the move (Go)
T1Ø	Wait 10 seconds after the move
N	End loop
U	Halt execution until the indexer receives the Continue command.

This command string pauses at the point where the U command is entered. A Continue (C) command causes execution to resume at the point where it was paused. In this example, the loop stops at the end of a move, and resumes when the indexer receives the C command.

UR Report Scale Factor Status

Version A

Type Status
Syntax aUR
Units None
Range None
Default None
Response a:nnn (nnn = 001 to 255)
See also US

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Reports the scale factor set using US command. The actual number of steps sent to the motor drive during a Preset move will be the current distance parameter setting multiplied by this scale factor. The range for nnn is 001 to 255.

Command	Description
MN	Set to Continuous mode
A1Ø	Set acceleration to 10 rps^2
V5	Set velocity to 5 rps
US2Ø	Set scale factor to 20
1UR	Request scale factor. The response is 1:020 (verifies that scale factor is set to 20)
D2ØØØØ	Set distance to 20,000 steps
G	Execute the move (Go)

Axis #1 sends out 20,000 • 20 = 400,000 steps.

US Set Position Scale Factor

Version A

Type Set-Up
Syntax <a>USn
Units n = distance multiplier
Range 1 - 255
Default 1
Response None
See also UR

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

This command sets the distance scale factor from 1 to 255. Any distance value set with the D command will be multiplied by the value set by US command. This value will remain valid until the indexer is reset or until a new US value is defined.

For the US command to be effective, you must issue a distance (D) command after the US command.

If 200 steps will yield 0.001 inches of rotary movement, you can issue a US200 command and program distance in 0.001-inch increments.

Command	Description
MN	Set to Continuous mode
A10	Set acceleration to 10 rps^2
V5	Set velocity to 5 rps
US200	Set scale factor to 200
D10000	Set distance to 10,000 steps
G	Execute the move (Go)

V Set Velocity

Version A

Type Motion
Syntax <a>Vn
Units n = rps
Range 0.01 to 160.000 (Motor dependant)
Default 1
Response None
See also A, D, G, MR

Attributes
[x] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Velocity (V) command defines the maximum speed at which the motor will run when given the Go (G) command. The actual speed of the motor or output frequency of the indexer will vary, depending on the motor drive resolution. The following formula is used to determine the output frequency of the indexer:

The motor resolution is a function of the motor/drive. However, you may match the PC23 to the motor/drive's resolution using the Motor Resolution (MR) command.

Frequency = (n) • (Motor Resolution) in *steps/rev.*

The top speed of the motor drive is limited by the motor type. Entering a velocity higher than the top speed of a motor drive system will cause the motor to stall and may cause the drive to fault.

Command	Description
MC	Set to Continuous mode
A5	Set acceleration to 5 rps^2
V5	Set velocity to 5 rps
G	Execute the move (Go)

In preset mode, Mode Normal (MN) the maximum velocity may also be limited when the resulting move profile is triangular. In Mode Continuous (MC), a Go (G) command is completed—the indexer moves on to the next command in the buffer once the specified velocity is reached.

W Report Immediate Position

Version A

Type Status
Syntax aWn
Units n = form of response
Range 1 = binary, 3 = hex ASCII
Default 1
Response 1W1 = a:nnnnn or xxxxxxxx (n = position in binary)
1W3 = a:xxxxxxxx (n = axis number, x = position in hex ASCII)
See also FSB, P, PB, PR, PX, PXB

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

The Immediate Position Request (W) command provides a position report of a specified axis while the motor is moving. The report indicates position relative to the start of the current move or the completion of the last move. This command works in both encoder step mode (FSB1) and motor step mode (FSB0).

If you specify the variable n as 1, the response format will be a 5-byte binary number (with no carriage return). The first byte is the axis number and the remaining bytes is the position in 2's compliment notation.

If you specify the variable n as 3, the response format will be an optional axis number, a colon, and eight hex ASCII characters in two's complement signed notation (with a carriage return).

Interpreting Hexadecimal Position Reports

This form of position report (a:xxxxxxxx) is generated by the W3 command. It consists of an an optional axis number and a colon, followed by eight hexadecimal characters; 0 - 9 and A - F. The position report is followed by a carriage return.

The decimal value of the hexadecimal expression can be determined using the technique demonstrated in Tables 5-1 and 5-2. The response is in 2's complement notation reflecting direction; negative numbers imply CCW motion. Both commands are designed for computer controlled situations where the computer can translate the hexadecimal.

Interpreting Binary Position Reports

This form of position report (nnnnn), consists of five bytes. The first is the axis number, followed by four bytes that must be linked together (concatenated) to form a 32-bit binary number. A typical PC23 communications algorithm expects to handle characters rather than binary numbers and may have problems with this kind of response. Assume that a response equivalent to the ASCII characters ^@, #, 0, and / (^@ refers to the CTRL key and @, an unprintable character) is given. The binary code for this response should be:

```
   ^@        #         0         /
00000000  00100011  00110000  00101111
```

This code has to be interpreted by the computer. The four characters must be converted to their ASCII code numbers and multiplied by the appropriate power of 256. The first character received is the most significant byte. Refer to the following table for a conversion technique for ^@ # 0 /. The formula used for the binary conversion is:

ASCII Value • Character Multiplier = Character Value

Response	ASCII Hex Value	ASCII Decimal Value	Character Multiplier	Conversion (steps)
^@	Ø	0	16,777,216 (256^3)	0
#	23	35	65,536 (256^2)	2,293,760
0	3Ø	48	256 (256^1)	12,288
/	2F	47	1 (256^0)	47
			Position Total:	2,306,095

If the position total is ≥2,147,483,648, then the position total is showing a 2's compliment negative number. To establish the correct position total, subtract 4, 294, 967, 296 (= 2^{32} = 16^8) from your result to receive the correct negative number. For example, if the response is 1(smiley face)+Σa8, the conversion would be as follows:

Response	ASCII Hex Value	ASCII Decimal Value	Character Multiplier	Conversion (steps)
+	F6	246	16,777,216 (256^3)	4,127,195,136
Σ	E4	228	65,536 (256^2)	14,942,208
a	61	97	256 (256^1)	24,832
8	38	56	1 (256^0)	56
			Position Total:	4,142,162,232

To establish the correct position, subtract 4, 294, 967, 296 from 4,142,162,232. The result is -152,805,064. Therefore, the last position total indicated 152,805,064 steps were moved in the CCW direction.

Positive W3 Response Interpretation

The system provides responses in the following format:

```
 MSD       LSD
xxxxxxxx
```

The first digit is the most significant digit (MSD). The last digit is the least significant digit (LSD). Refer to the following table for the value of each digit.

Digit	Digit Multiplier
x (MSD)	x •16^7 = h • 268,435,456 = _____
x	x •16^6 = h • 16,777,216 = _____
x	x •16^5 = h • 1,048,576 = _____
x	x •16^4 = h • 65,536 = _____
x	x •16^3 = h • 4,096 = _____
x	x •16^2 = h • 256 = _____
x	x •16^1 = h • 16 = _____
h (LSD)	x •16^0 = h • 1 = _____

The decimal (h) may have one of the values shown in Table 5-3.

Decimal Value	Hexadecimal Value	Decimal Value	Hexadecimal Value
0	0	8	8
1	1	9	9
2	2	10	A
3	3	11	B
4	4	12	C
5	5	13	D
6	6	14	E
7	7	15	F

Using the previous tables, review the decimal value that would be calculated if the hexadecimal response was **ØØØ433AE**.

Hexadecimal	Character Multiplier	Conversion (steps)
Ø	0 • 268,435,456	0
Ø	0 • 16,777,216	0
Ø	0 • 1,048,576	0
4	4 • 65,536	262,288
3	3 • 4,096	12,288
3	3 • 256	768
A (= 10)	10 • 16	160
E (= 14)	14 • 1	14
	Total Steps:	275,374

Negative W3 Response Interpretation

If the first digit of the position portion of the response **a:xxxxxxxx** is **8**, **9**, **A**, **B**, **C**, **D**, **E**, or **F**, the response represents a *two's complement* negative number. Any other response should be interpreted per Table 5-1. There are several ways to convert an 8-digit two's complement hexadecimal number to decimal.

The Binary Approach

① Convert the hexadecimal response to binary form.

② Complement the binary number.

③ Add 1 to the binary result.

④ Convert the binary result to decimal value.

Response to **1W3** is **1:EFFE271Ø**.

①	E	F	F	E	2	7	1	Ø
	1110	1111	1111	1110	0010	0111	0001	0000
②	0001	0000	0000	0001	1101	1000	1110	1111
③	0001	0000	0000	0001	1101	1000	1111	0000
④	1	Ø	Ø	1	D	8	F	Ø

1ØØ1D8FØ hex = **268,556,528** decimal.

Therefore, the result is 268,556,528 steps in the CCW direction.

The Computer Approach

① Convert the hexadecimal response to decimal.

② Subtract 4,294,967,296 (= $2^{32} = 16^8$) from the decimal number.

Response to **1W3** is **1:EFFE271Ø**.

① Convert hex to decimal as follows:

Hexadecimal	Character Multiplier	Conversion (steps)
E (= 14)	14 • 268,435,456	3,758,096,384
F (= 15)	15 • 16,777,216	251,658,240
F (= 15)	15 • 1,048,576	15,728,640
E (= 14)	14 • 65,536	917,504
2	2 • 4,096	8,192
7	7 • 256	1,792
1	1 • 16	16
Ø	Ø • 1	0
	Total Steps:	4,026,410,768

② 4,026,410,768 - 4,294,967,296 = ⁻268,556,528. Therefore, the result is 268,556,528 steps in the CCW direction.

Y Stop Loop

Version A

Attributes
[ ] Buffered
[ ] Device specific
[ ] Saved independently
[ ] Saved in sequences

Type	Programming
Syntax	<a>Y
Units	None
Range	None
Default	None
Response	None
See also	L, N

The Stop Loop (Y) command takes you out of inner loop when the loop completes its current pass. This command does not halt processing of the commands in the loop until the indexer processes reach the last command of the current loop. At that time, the indexer executes the command that follows the End Loop (N) command.

Command	Description
L	Loop indefinitely
A5	Set acceleration to 5 rps^2
V5	Set velocity to 5 rps
D25000	Set distance to 25,000 steps
T2	Wait 2 seconds
G	Execute the move (Go)
N	End loop
Y	Stop loop

The loop requires the motor to move 25,000 steps and then wait for two seconds. The loop terminates at the end of the loop cycle it is executing when it receives the Y command.

C H A P T E R ⑥

Hardware Reference

Chapter Objectives

This chapter is designed to function as a quick-reference tool for the following information:

- ❑ System specifications (dimensions and performance)
- ❑ Jumper and DIP switch settings
- ❑ I/O connections and specifications

PC23 System Specifications

Refer to the following table for PC23 System Specifications.

	Parameter	Value
Performance	Stepping Accuracy	±0 steps from preset total
	Velocity Accuracy	±0.02% of maximum rate above 0.01 revs/sec
	Velocity Repeatability	±0.02% of maximum rate
	Velocity Range:	25,000 steps/rev 0.01 - 20.00 rps 5,000 steps/rev 0.01 - 100.00 rps
	Acceleration Range:	
	25,000 steps/rev	0.01 - 999.9 rps^2
	5,000 steps/rev	0.05 - 4,999.95 rps^2
	Position Range (all resolutions)	0 - 99,999,999 steps
Power Input	Indexer Board	+5VDC @ 1.2A typical, 1.5A maximum (from user's PC)
	User-supplied to Adaptor Box	+5VDC @ 1A for PC23 optos without encoders, 2A for PC23 optos with encoders
Interface	IBM PC interface in accordance with IBM I/O channel signal specifications and definitions. Uses the following I/O channel signal lines:	
	+A0, +A2 through +A9	-IOW
	+D0 through +D7	+IRQ3
	+AEN	+IRQ4
	+RESET	+5V
	+IOR	GND
Inputs	***User must provide +5VDC (±10%) to power all input opto couplers***	
	CCW, CW Limits, Home Enable, and Triggers	3.5VDC (minimum) to 10VDC (maximum) differential voltage, 10 - 40MA source Signal duration≥ 4 ms
	Encoder	< 0.5VDC = low, > 3.0VDC = high, -4.5mA sink 25KHz max input frequency (pre-quadrature)
	Joystick	0 to 2.5VDC, 5kΩ impedance recommended

Outputs (Step, Direction, Shutdown, and POBs)	Differential outputs can be changed via adaptor box jumpers to single-ended.	
	Signal high	+3.0VDC minimum @ +30mA
	Signal low	+1.0VDC maximum @ -30mA
	Step signal duration:	15 µs nominal @ 200 - 400 steps/rev
		1 µs nominal @ 21,600 - 50,000 steps/rev
Environmental	Operating Temperature	32°F to 122°F (0°C to 50°C);
	Humidity	0 to 95% (non-condensing)
	Storage	-22°F to 185°F (-30°C to 85°C)

I/O Connections

This section provides PC23 Auxiliary Box I/O connection pinouts and circuit drawings.

Auxiliary Connector Pinouts

Pin #	Axis #1	Axis #2	Axis #3
1	Trigger #1 Input +	Trigger #3 Input +	Trigger #5 Input +
2	Trigger #2 Input +	Trigger #4 Input +	Trigger #6 Input +
3	Home Enable Input +	Home Enable Input +	Home Enable Input +
4	CCW Limit Input +	CCW Limit Input +	CCW Limit Input +
5	CW Limit Input +	CW Limit Input +	CW Limit Input +
6	CCW Limit Return -	CCW Limit Return -	CCW Limit Return -
7	CW Limit Return -	CW Limit Return -	CW Limit Return -
8	*Not Connected (NC)*	*NC*	*NC*
9	*NC*	*NC*	Joystick Analog GND
10	*NC*	*NC*	Joystick Channel Ø
11	*NC*	*NC*	Joystick Channel 1
12	*NC*	*NC*	Joystick Channel 2
13	*NC*	*NC*	Joystick Channel 3
14	Trigger #1 Input Return	Trigger #3 Input Return	Trigger #5 Input Return
15	Trigger #2 Input Return	Trigger #4 Input Return	Trigger #6 Input Return
16	Home Enable Return	Home Enable Return	Home Enable Return
17	*NC*	*NC*	*NC*
18	Programmable Output #2	Programmable Output #4	Programmable Output #6
19	DC Common	DC Common	DC Common
20	Programmable Output #1	Programmable Output #3	Programmable Output #5
21	DC Common	DC Common	DC Common
22	*NC*	*NC*	*NC*
23	Ext +5VDC Input*	Ext +5VDC Input*	Ext +5VDC Input*
24 & 25	*NC*	*NC*	*NC*

* User Supplied

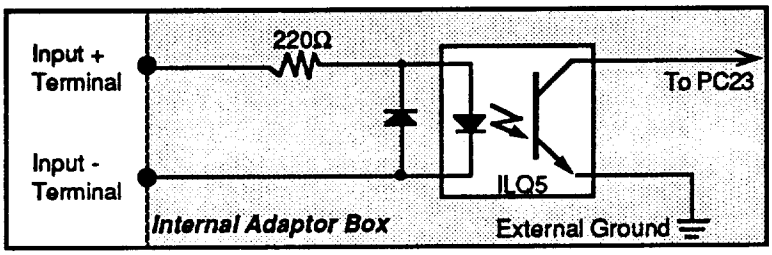
Motor Driver Connector Pinouts

Pin #	Wire color	Function
1	Red	Step +
2	Green	Direction +
5	-	Shield
14	Black	Step -
15	White	Direction -
16	Blue	Shutdown +
17	White/Red	Shutdown -

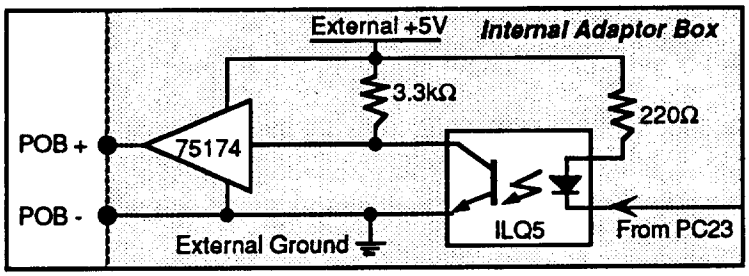
Encoder Connector Pinouts

Pin #	Function
1	Quadrature Channel A+
2	Quadrature Channel A- (optional)
3	Quadrature Channel B+
4	Quadrature Channel B- (optional)
5	Channel Z+ (index)
6	Channel Z- (optional)
7	*NC*
8	Shield
9 - 13	*NC*
14 - 20	Ground
21 & 22	NC
23 - 25	+5VDC (250mA max), user supplied

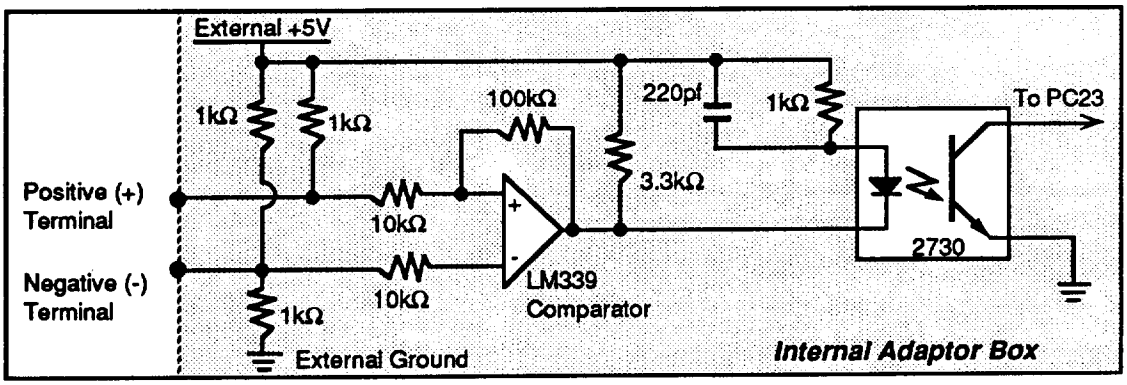
Auxiliary Connector: Typical Input Circuit



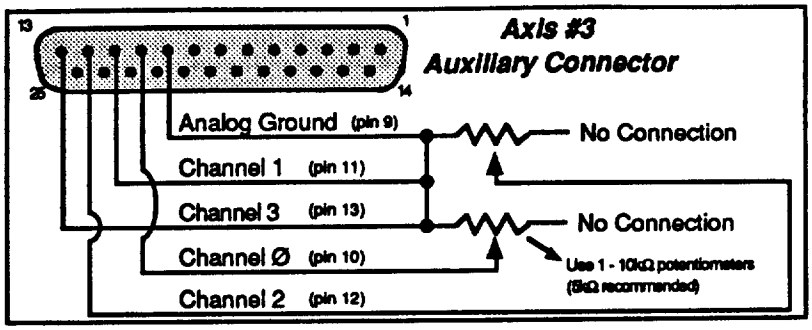
Auxiliary Connector: Typical Output Circuit



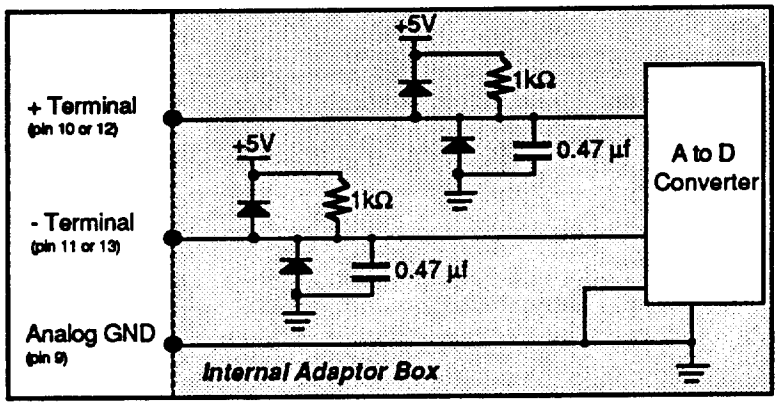
Encoder Connector: Typical Encoder Input Circuit



Joystick Connections



Axis 3 Aux. Connector: Typical Joystick Input Circuit



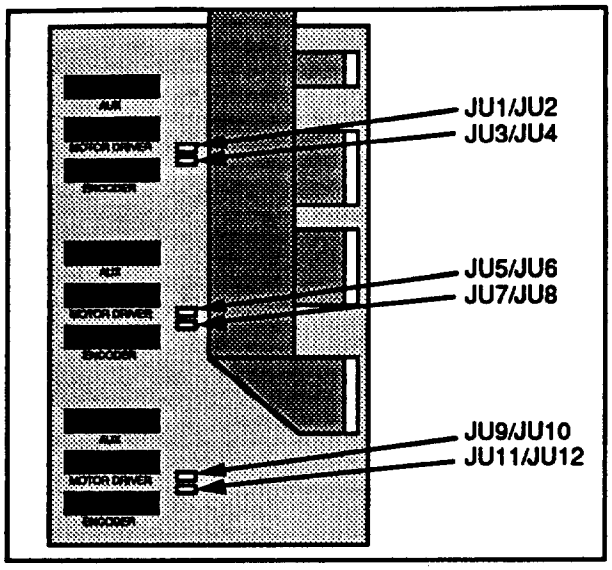
Jumper Settings

This section discusses jumper settings for the PC23 Indexer board and adaptor box. The following table identifies the function and default settings for each jumper.

Jumper	Function	Default Setting
PC Board Jumpers:		
JU13	Enable/disable self test function	Installed (disables self test)
JU27/JU28	Select interrupt 3 (JU27) or 4 (JU28) on PC I/O bus	Removed (no interrupts)
Adaptor Box Jumpers:		
JU1/JU2 (axis 3)	Select differential or single-ended output (Step)	Position JU2 (differential)
JU3/JU4 (axis 3)	Select differential or single-ended output (Direction)	Position JU4 (differential)
JU5/JU6 (axis 2)	Select differential or single-ended output (Step)	Position JU6 (differential)
JU7/JU8 (axis 2)	Select differential or single-ended output (Direction)	Position JU8 (differential)
JU9/JU10 (axis 1)	Select differential or single-ended output (Step)	Position JU10 (differential)
JU11/JU12 (axis 1)	Select differential or single-ended output (Direction)	Position JU12 (differential)

Changing Outputs from Differential to Single-Ended

The PC23 is shipped with its step and direction outputs configured as TTL-compatible differential. If your system accepts only single-ended indexer outputs, change the outputs from differential to single-ended using the jumpers on the adaptor box. To access these jumpers, you must first remove the adapter box cover. The following figure illustrates the jumper locations. Refer to the previous table for jumper settings.



DIP Switch Settings: (Indexer Card Address)

The address is set using the 8-position DIP switch located on the PC23 indexer card. The DIP switches are *negative true* in that any switch in the position marked ON has a binary value of zero. Switches that are OFF have a non-zero binary value. The sum of the binary values of DIP switches 1 through 8 is the board's *base address*. The binary values assigned to each of the eight DIP switches are listed in the following table.

Switch #	Address	Binary Value (OFF) Decimal	Hex	Default Setting
1	9	512	200	OFF
2	8	256	100	OFF
3	7	128	80	ON
4	6	64	40	ON
5	5	32	20	ON
6	4	16	10	ON
7	3	8	8	ON
8	2	4	4	ON

The PC23 is shipped from the factory with switches 1 and 2 in the OFF position, all others are ON . The board address then is 256 + 512 = 768 (or 100 + 200 = 300 hex). As such, the PC23 is configured to occupy I/O address locations 300 hex through 303 hex. The control and status registers are at the odd address location (301 hex), and data registers are at the even address location (300 hex).

The allowable address locations are highlighted in the following table.

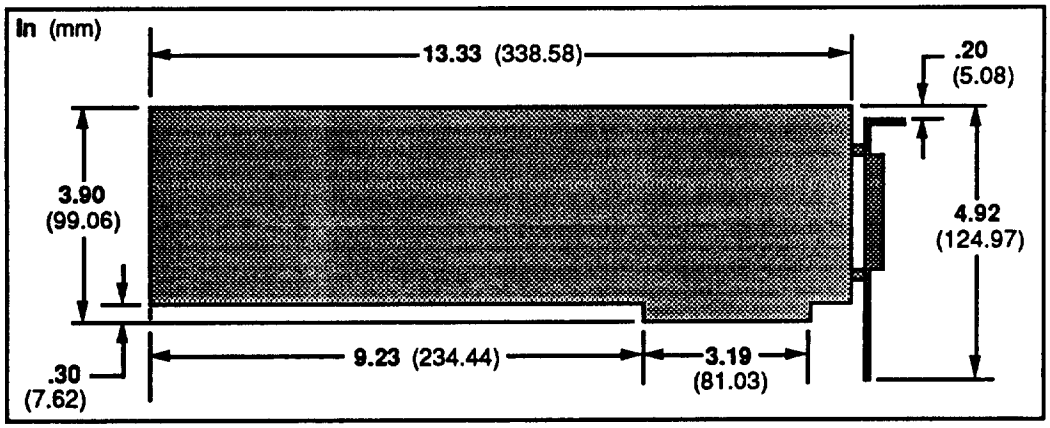
Hex Range	Device
000-01F	DMA Controller 1, 8237A-5
020-03F	Interrupt controller 1, 8259A, Master
040-05F	Timer, 8254-2
060-06F	8042 (Keyboard)
070-07F	Real-time clock, NMI (non-maskable interrupt) mask
080-09F	DMA page register, 74LS612
0A0-0BF	Interrupt Controller 2, 8259A
0C0-0DF	DMA controller 2, 8237A-5
0F0	Clear Math Coprocessor Busy
0F1	Reset Math Coprocessor
0F8-0FF	Math Coprocessor
1F0-1F8	Fixed Disk
200-207	Game I/O
20C-20D	Reserved
21F	Reserved
278-27F	Parallel printer port 2
2B0-2DF	Alternate Enhanced Graphics Adapter
2E1	GPIB (Adapter 0)
2E2 & 2E3	Data Acquisition (Adapter 0)
2F8-2FF	Serial Port 2
300-31F	**Prototype Card**
360-363	PC Network (low address)
364-367	Reserved
368-36B	PC Network (high address)

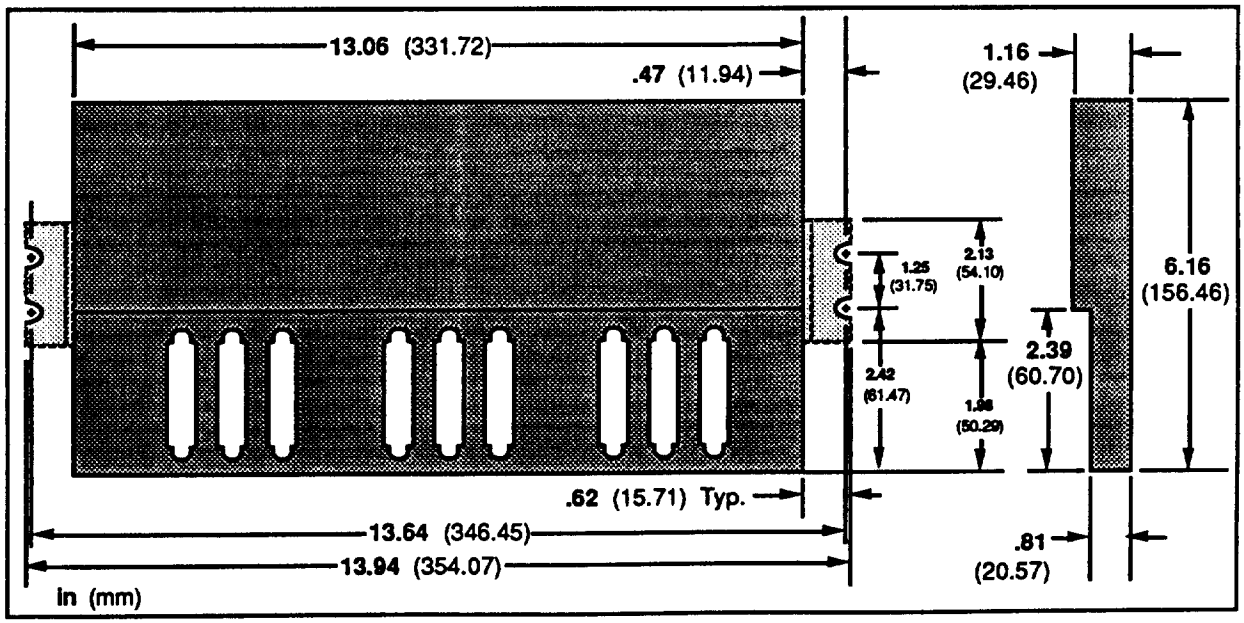
Hex Range	Device
36C-36F	Reserved
378-37F	Parallel printer port 1
380-38F	SDLC, bisynchronous 2
390-393	Cluster
3A0-3AF	Bisynchronous 1
3B0-3BF	Monochrome Display and Printer Adapter
3C0-3CF	Enhanced Graphics Adapter
3D0-3DF	Color/Graphics Monitor Adapter
3F0-3F7	Diskette Controller
3F8-3FF	Serial port 1
6E2 & 6E3	Data acquisition (Adapter 1)
790-793	Cluster (Adapter 1)
AE2 & AE3	Data acquisition (Adapter 2)
B90-B93	Cluster (Adapter 2)
EE2 & EE3	Data acquisition (Adapter 3)
1390-1393	Cluster (Adapter 3)
22E1	GPIB (Adapter 1)
2390-2393	Cluster (Adapter 4)
42E1	GPIB (Adapter 2)
62E1	GPIB (Adapter 3)
82E1	GPIB (Adapter 4)
A2E1	GPIB (Adapter 5)
C2E1	GPIB (Adapter 6)
E2E1	GPIB (Adapter 7)

Note: I/O Address, hex 000 to 0FF, are reserved for the system board I/O. Hex 100 to 3FF are available on the I/O channel.

Dimensional Drawings

Refer to the following figures for dimensions of the PC23 indexer board and the adaptor box.





C H A P T E R ⑦

Troubleshooting

Chapter Objective

The information in this chapter is provided to help you to isolate and resolve system hardware and software problems .

Spare Parts List

Description	Part Number
25-Pin D connector plug	43-001989-01
25-Pin D connector shell	43-001990-01
Phillips screws 4-40 x 1/4	51-006021-01
Phillips screws 6-32 x 1/4	51-006037-01
Adaptor box mounting bracket	53-006007-01
Card guide	58-004144-01

Troubleshooting

This section discusses methods to identify, isolate, and resolve indexer-related problems that may occur with your PC23 Indexer.

Isolating Problems

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 accomplish this, you can effectively begin to resolve the problem.

Try to 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 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.

Isolate each system component and ensure that each component functions properly when it is run independently. You may have to remove your system components and re-install them component-by-component to

detect the problem. If you have additional components available, you may want to use them to replace existing components in your system to help identify the source of the problem.

> ***WARNING***
>
> **Make sure to remove power before disconnecting system components or changing wiring.**

Once you have isolated the problem, take the necessary steps to resolve it. Refer to the problem solutions contained in this chapter. If your system's problem persists, contact Parker Compumotor's Applications Department at (800) 358-9070.

Problems & Solutions

Problem	Cause	Solution
No Communication	1. Wrong Address	1. Change jumpers or software address
	2. Two boards have same address	2. Change address on one board (unique addresses)
	3. Board not seated	3. Seat board in IBM Bus
No Motion	1. 5V external power supply to AUX Board is missing	1. Connect 5V external power supply to pin 23 and ground to pin 21 on any AUXILIARY connector.
	2. Step pulse width too narrow for drive to recognize	2. Issue MR command to set motor resolution.
	3. Limits engaged	3. Issue LD3 or move load off of limit.
	4. Improper Wiring: Drive Faults Motor plugged to encoder Encoder to wrong axis Shutdown line miswired	4. Check wiring, be sure the self-test jumper (JU13) is not removed. Refer to Chapter 6.
	5. Load is jammed	5. Remove power and manually move load away from jam.
No Torque	1. Improper wiring	1. Check wiring. Refer to Chapter 6.
	2. Blown drive	2. Return drive for repair.
	3. Drive faulted	3. Remove fault condition.
	4. Drive Shutdown	4. Issue the ST0 command to energize the drive.
Fault Light Comes On	1. Jumper JU13 removed	1. Ensure the self-test jumper JU13 is not removed.
	2. Hardware failure	2. Return unit for repair.
Wrong Speed or Distance	1. Wrong resolution setting	1. Issue MR command to set motor resolution.
Missing Encoder Counts	1. Miswiring	1. Check wiring. Refer to Chapter 6.
	2. Encoder slipping	2. Tighten encoder coupler.
	3. Encoder too hot	3. Reduce encoder temperature with heatsink, thermal insulator, etc.
Motor Creeps	1. Jumper JU13 removed	1. Install self-test jumper (JU13).
	2. Wrong encoder/motor polarity	2. Swap Channel A & Channel B of encoder, or issue OSA1 command.
	3. Encoder connected to wrong axis	3. Swap encoder to indexer connection.
No POB Operation	1. Improper wiring	1. Check wiring. Refer to Chapter 6.
	2. Shorted POB	2. Check for shorts on open collector POB outputs.
	3. No power to pull-up	3. Connect external 5VDC to pin 23 and GND to pin 21 on any AUXILIARY connector.
No Trigger Inputs	1. Improper wiring	1. Check wiring. Refer to Chapter 6.
Motor Creeps in Joystick Mode	1. Pot not centered to give 1.25V in	1. Perform mechanical adjustment on joystick. Use J2 and J0 commands.

Reducing Electrical Noise

If you operate the PC23 in an environment in which there is an excessive amount of electrical noise, try to eliminate sources of possible noise interference. Potential sources of electrical noise include inductive devices such as solenoids, relays, motors, and motor starters when they are operated by a hard contact.

For more information on identifying and suppressing electrical noise, refer to the Technical Data section of the *Compumotor Programmable Motion Control Catalog*.

Returning The System

If you must return your PC23 Indexer to effect repairs or upgrades, use the following procedure:

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 Parker Compumotor to be out of warranty.

② Before you ship the indexer to Parker Compumotor, have someone from your organization with a technical understanding of the PC23 Indexer and its application include answers to the following questions:

- What is the extent of the failure/reason for return?
- How long did it operate?
- How many units are still working?
- How many units failed?
- What was happening when the unit failed (i.e., installing the unit, cycling power, starting other equipment, etc)?
- How was the product configured (in detail)?
- What, if any, cables were modified and how?
- With what equipment is the unit interfaced?
- What was the application?
- What was the system sizing (speed, acceleration, duty cycle, inertia, torque, friction, etc.)?
- What was the system environment (temperature, enclosure, spacing, unit orientation, contaminants, etc.)?
- What upgrades, if any, are required (hardware, software, user guide)?

③ 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 Applications Department is (800) 358-9070.

④ Ship the unit to:

Parker Compumotor Corporation
5500 Business Park Drive, Suite D
Rohnert Park, CA 94928
Attn: RMA # xxxxxx

APPENDICES

Command Listing

A	(ACCELERATION)
AB	(REPORT ANALOG VOLTAGE, BINARY)
AV	(REPORT ANALOG VOLTAGE, ASCII)
B	(BUFFER STATUS)
C	(CONTINUE)
CG	(CORRECTION GAIN)
CM	(SET CORRECTION MODE FOR POSITION MAINTENANCE)
CR	(CARRIAGE RETURN)
D	(DISTANCE)
DB	(DEAD BAND)
DPA	(DISPLAY POSITION ACTUAL)
DW	(SET DEAD BAND WINDOW)
ER	(ENCODER RESOLUTION)
FR	(ENCODER FUNCTIONS REPORT)
FSA	(SET INCREMENTAL/ABSOLUTE MODE)
FSB	(SET INDEXER TO MOTOR/ENCODER MODE)
FSC	(POSITION MAINTENANCE)
FSD	(STOP ON STALL)
FSE	(ENABLE OUTPUT #6 ON STALL)
FSF	(ENABLE STOP ON TRIGGER #6)
G	(GO)
GNNN	(SYNCHRONIZED GO)
GA	(GO HOME ACCELERATION)
GH	(GO HOME)
^H	(BACKSPACE)
H	(SET DIRECTION)
I	(LOAD MOVE DATA)
IO	(IMMEDIATE OUTPUT)
IS	(INPUT STATUS)
J	(ENABLE/DISABLE JOYSTICK)
JB	(SET JOYSTICK BACKLASH)
JD	(SET JOYSTICK DEAD BAND)
JV	(SET JOYSTICK BACKLASH COMPENSATION VELOCITY)
JZ	(SET JOYSTICK TO ZERO)
K	(KILL)
L	(LOOP)
LA	(LIMIT ACCELERATION)
LD	(LIMIT DISABLE)
MA	(MODE ALTERNATE)
MC	(MODE CONTINUOUS)
MN	(MODE NORMAL)
MPA	(MODE POSITION ABSOLUTE)
MPI	(MODE POSITION INCREMENTAL)
MR	(SELECT MOTOR RESOLUTION)
MSL	(IDENTIFY CLOCK SOURCE FOR TIMED DATA STREAMING)
MSS	(START MASTER CLOCK FOR TIMED DATA STREAMING)
MV	(SET MAXIMUM CORRECTION VELOCITY)
N	(END OF LOOP)
O	(OUTPUT)
OR	(REPORT FUNCTION SETUPS)
OSA	(SET ENCODER DIRECTION)
OSB	(BACKUP TO HOME)
OSC	(DEFINE ACTIVE STATE OF HOME SWITCH)
OSD	(DEFINE ACTIVE STATE OF ENCODER'S Z CHANNEL INPUT)
OSE	(ENABLE STALL DETECT)
OSF	(SET MAXIMUM JOYSTICK VELOCITY)
OSG	(SET FINAL GO HOME DIRECTION)
OSH	(REFERENCE EDGE OF HOME SWITCH)
P	(REPORT INCREMENTAL POSITION, ASCII)
PB	(REPORT INCREMENTAL POSITION, BINARY)
PR	(REPORT ABSOLUTE POSITION)
PS	(PAUSE)
PX	(REPORT ENCODER ABSOLUTE POSITION, ASCII)
PXB	(REPORT ENCODER ABSOLUTE POSITION, BINARY)
PZ	(POSITION ZERO)
Q	(COMPLETE CURRENT COMMAND AND CLEAR COMMAND BUFFER)
QØ	(EXIT STREAMING MODE)
Q1	(ENTER IMMEDIATE VELOCITY STREAMING MODE)
Q2	(ENTER TIME-DISTANCE STREAMING MODE)
Q3	(ENTER TIME-VELOCITY STREAMING MODE)
QI	(REPORT STATUS OF QS COMMANDS)
QIB	(INTERRUPT STATUS REPORT, BINARY)
QR	(REPORT QS COMMAND FUNCTION ENABLE STATUS)
QS	(INTERRUPT ON SIGNAL COMMANDS)
QSA	(INTERRUPT ON TRIGGER #1 HIGH)
QSB	(INTERRUPT ON MOVE COMPLETE)
QSD	(INTERRUPT SIGNAL ON LIMIT ENCOUNTERED)
QSE	(INTERRUPT ON READY TO RESPOND)
QSG	(INTERRUPT ON COMMAND BUFFER FULL)
QSH	(INTERRUPT ON MOTOR STALL)
R	(REQUEST INDEXER STATUS)
RA	(LIMIT SWITCH STATUS REPORT)
RB	(REPORT LOOP, PAUSE, SHUTDOWN, TRIGGER STATUS)
RC	(REPORT CLOSED LOOP AND GO HOME STATUS)
RM	(RATE MULTIPLIER IN IMMEDIATE VELOCITY STREAMING MODE)
RV	(REPORT SOFTWARE PART NUMBER)
S	(STOP)
SA	(STOP ALL)
SD	(DEFINE TIMED DATA STREAMING MODE STREAMING DATA)
SR	(REPORT CONFIGURATION STATUS)
SSD	(MODE ALTERNATE STOP MODE)
SSF	(NORMAL/LOW VELOCITY RANGE)
SSG	(CLEAR/SAVE THE COMMAND BUFFER ON LIMIT)
SSH	(CLEAR/SAVE THE COMMAND BUFFER ON STOP)
ST	(SHUTDOWN)

T	(TIME DELAY)
TD	(SET TIME INTERVAL FOR TIMED DATA STREAMING MODE)
TR	(WAIT FOR TRIGGER)
TS	(REPORT TRIGGER STATUS)
U	(PAUSE AND WAIT FOR CONTINUE)
UR	(REPORT SCALE FACTOR STATUS)
US	(SET POSITION SCALE FACTOR)
V	(VELOCITY)
W1	(SIGNED BINARY POSITION REPORT)
W3	(HEXADECIMAL POSITION REPORT)
Y	(STOP LOOP)

I N D E X