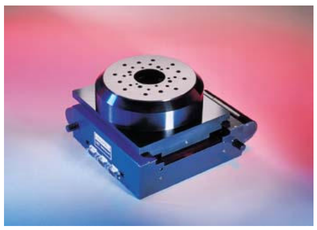
Newport
Experience | Solutions

PM500 SERIES






Operation and Command Reference Manual

Warranty

Newport Corporation warrants that this product will be free from defects in material and workmanship and will comply with Newport’s published specifications at the time of sale for a period of one year from date of shipment. If found to be defective during the warranty period, the product will either be repaired or replaced at Newport's option.

To exercise this warranty, write or call your local Newport office or representative, or contact Newport headquarters in Irvine, California. You will be given prompt assistance and return instructions. Send the product, freight prepaid, to the indicated service facility. Repairs will be made and the instrument returned freight prepaid. Repaired products are warranted for the remainder of the original warranty period or 90 days, whichever first occurs.

Limitation of Warranty

The above warranties do not apply to products which have been repaired or modified without Newport’s written approval, or products subjected to unusual physical, thermal or electrical stress, improper installation, misuse, abuse, accident or negligence in use, storage, transportation or handling. This warranty also does not apply to fuses, batteries, or damage from battery leakage.

THIS WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR USE. NEWPORT CORPORATION SHALL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM THE PURCHASE OR USE OF ITS PRODUCTS.

Trademarks

Windows is a registered trademark of Microsoft Corporation. LabVIEW and LabWindows are trademarks of National Instruments Corporation.

This manual has been provided for information only and product specifications are subject to change without notice. Any change will be reflected in future printings.

1791 Deere Ave.

Irvine, CA 92606

(800) 222-6440

P/N 22519-01, **Rev. A**

Confidentiality & Proprietary Rights

Reservation of Title:

The Newport programs and all materials furnished or produced in connection with them ("Related Materials") contain trade secrets of Newport and are for use only in the manner expressly permitted. Newport claims and reserves all rights and benefits afforded under law in the Programs provided by Newport Corporation.

Newport shall retain full ownership of Intellectual Property Rights in and to all development, process, align or assembly technologies developed and other derivative work that may be developed by Newport. Customer shall not challenge, or cause any third party to challenge the rights of Newport.

Preservation of Secrecy and Confidentiality and Restrictions to Access:

Customer shall protect the Newport Programs and Related Materials as trade secrets of Newport, and shall devote its best efforts to ensure that all its personnel protect the Newport Programs as trade secrets of Newport Corporation. Customer shall not at any time disclose Newport's trade secrets to any other person, firm, organization, or employee that does not need (consistent with Customer's right of use hereunder) to obtain access to the Newport Programs and Related Materials. These restrictions shall not apply to information (1) generally known to the public or obtainable from public sources; (2) readily apparent from the keyboard operations, visual display, or output reports of the Programs; 93) previously in the possession of Customer or subsequently developed or acquired without reliance on the Newport Programs; or (4) approved by Newport for release without restriction.

Service Information

This section contains information regarding factory service for the PM500 System. The user should not attempt any maintenance or service of the system or optional equipment beyond the procedures outlined in this manual. Any problem that cannot be resolved should be referred to Newport Corporation.

Technical Support Contacts

Newport Corporation Service Department.

1791 Deere Ave.

Irvine, CA 92606

Telephone: (800) 222-6440

Newport Corporation RMA Procedures

Each defective part must have an RMA number assigned by a Newport representative. The serial number of the damaged component must be provided to the Newport representative. Please refer to Appendix G of this manual for complete RMA procedure.

Packaging

Stages must be packaged in their original boxes and the stage locking tabs must be installed prior to shipping. There is a written procedure for removing, packaging, and shipping Newport stages. Please refer to Appendix F of this manual.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Scope of the Manual

Welcome to the PM500 Series Operations and Command Reference Manual.

This manual provides descriptions, operating procedures, and service requirements for the PM500 Linear and Rotary Stages.

Safety considerations, conventions and definitions are provided in Section 2, Safety Considerations.

Procedures for unpacking the equipment, inspection for damage are provided in Section 3 - Getting Started. Mounting requirements, descriptions of controls and indicators, computer interface, motion programming, setup procedures, and service and maintenance are provided in sections 1 through 19.

The Following Information Is Provided In The Appendices:

- Appendix A – Cabling And Communications
- Appendix B – Clean Room Compatibility And Outgassing
- Appendix C – Vacuum Compatibilities
- Appendix D – Memory Capabilities
- Appendix E – Motion Specifications
- Appendix F – Packaging
- Appendix G – Warranty Return
- Appendix H– Service Form
- Appendix I – System Settings

1.2 Manual Conventions

The following conventions and standards will be used in this manual.

1.2.1 Keys, Buttons and Icons

Computer keyboard keys and onscreen buttons and icons are used in the text to describe many user operations. The key-top symbol as it appears on the keyboard, the button or icon name is represented in **boldface type**. For example: **Ctrl** is used for the Control key or **Manual Operation** button.

1.2.2 Key Operation

Some operations require simultaneously use of two or more keys. Such operations are identified by the top symbols on the key separated by a plus (+) sign. For example, **Ctrl** + **Pause (Break)** means hold down the **Ctrl** key and at the same time press **Pause (Break)**. If three keys are used, hold down the first two and at the same time press the third.

1.2.3 User Input

Text that is required to be typed in, will be shown in the boldface type courier (new) font as shown below:

`DISKCOPY A: B:`

1.2.4 The Display

Text generated by the computer that appears on its display is presented in the typeface courier (new) font as shown below:

```
FORMAT complete
```

2 Safety Precautions

2.1 Definitions and Symbols

The following terms and symbols are used in this documentation and also appear on PM500 Series Products where safety-related issues occur.

2.1.1 General Warning or Caution

Figure 1: General Warning or Caution Symbol

The Exclamation Symbol in the figure above appears in Warning and Caution tables throughout this document. This symbol designates an area where personal injury or damage to the equipment is possible.

2.1.2 Grounding

Figure 2: Grounding Symbol

The Grounding Symbol in the figure above appears on labels affixed to the PM500 Series Products, and this symbol identifies terminal which is intended for connection to an external (ground) conductor for protection against electric shock in case of a fault, or the terminal of a protective earth (ground) electrode. Any mishandling could result in irreparable damage to the equipment, and personal injury or death.

2.1.3 Electric Shock

Figure 3: Electrical Shock Symbol

The Electrical Shock Symbol in the figure above appears on labels affixed to the PM500 Series Products, and this symbol indicates a hazard arising from dangerous voltage. Any mishandling could result in irreparable damage to the equipment, and personal injury or death.

2.2 Terminology

The following is a brief description of the terms specific to PM500 Series Controller and Stages.

Axis – a logical name for a stage/positioner/ motion device

Encoder – a displacement measuring device, term usually used for both linear and rotary models.

Fiducial (position) – the unique point in space that can be accurately found by an axis, also called origin.

Jog – a motion of undetermined-length, initiated manually

Motion device – electro-mechanical equipment. Used interchangeably with stage and positioner.

Move – a motion to a destination

Positioner – used interchangeably with stage and motion device

Stage – used interchangeably with motion device and positioner

2.3 Warnings and Cautions

The following are definitions of the Warnings, Cautions and Notes that are used throughout this manual to call your attention to important information regarding your safety, the safety and preservation of your equipment or an important tip.

WARNING

Situation has the potential to cause bodily harm or death.

CAUTION

Situation has the potential to cause damage to property or equipment.

NOTE

Additional information the user or operator should consider.

2.4 Safety Considerations

The following general safety precautions must be observed during all phases of operations of this equipment. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the equipment.

Disconnect or do not plug in the power cord in the following circumstances:

- If the power cord or any other attached cables are frayed or damaged.
- If the power plug or receptacle is damaged.
- If the unit is exposed to rain or excessive moisture, or liquids are spilled on it.
- If the unit has been dropped or the case is damaged.
- If you suspect service or repair is required.
- When you clean the case.

To protect the equipment from damage and avoid hazardous situations, follow these recommendations:

- Do not make modifications or parts substitutions.
- Return equipment to Newport Corporation for service and repair.
- Do not touch, directly or with other objects, live circuits inside the unit.
- Keep air vents free of dirt and dust.
- Do not block air vents.
- Keep liquids away from unit.
- Do not operate the equipment in an environment with humidity levels (<10% and >90% humidity).

WARNING

All attachment plug receptacles in the vicinity of this unit are to be of the grounding type and properly polarized. Contact an electrician to check faulty or questionable receptacles.

WARNING

This product is equipped with a 3-wire grounding type plug. Any interruption of the grounding connection can create an electric shock hazard. If you are unable to insert the plug into your wall plug receptacle, contact an electrician to perform the necessary alterations to assure that the green (green-yellow) wire is attached to earth ground.

WARNING

This product operates with voltages that can be lethal. Pushing objects of any kind into cabinet slots or holes, or spilling any liquid on the product, may touch hazardous voltage points or short out parts.

WARNING

When opening or removing covers observe the following precautions:

- **Turn power OFF and unplug the unit from its power source**
- **Remove jewelry from hands and wrists**
- **Use insulated hand tools only**
- **Maintain grounding**

WARNING

To protect operating and servicing personnel, this instrument is supplied with a three-prong power receptacle. The center prong of the receptacle connects the instrument's chassis, cabinet and panels to earth ground when used with a properly wired three-conductor outlet and cable. Improperly grounded equipment can result in hazardous electrical potentials.

3 Getting Started

3.1 Outline of Section 1

The following topics are covered in this section:

- Unpacking and inspection of the system
- Handling of motion devices

For additional setup and operation information for optional features, refer to the appropriate sections:

OPTION	SECTION
Vertical stages	6.5
Counter balance	6.5
Adding stages to an existing controller	19.3
Updating system firmware	19.4

WARNING

Read this entire section before assembling your system for the first time. System performance depends on many factors, including stage mounting and load attachment. Damage to the system can occur if devices are not connected or operated correctly.

3.2 Unpacking and Handling

	CAUTION
⚠	**All PM500 stages must be handled with care during shipping.**

Unpack your system carefully. PM500 stages are double-boxed and wrapped in plastic. ***Always handle stages by the base. Do not*** handle the stages by the motor or by the carriage. Such handling can damage the leadscrew bearings or the bearing ways.

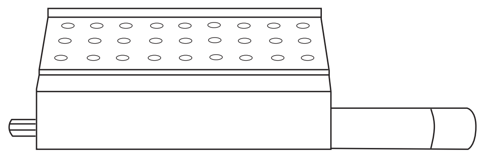


Figure 4: P500 Stage

To prevent damage during shipping, a temporary steel-shipping tab has been installed between the stage carriage and body on the integrated PM500-33LR and on all linear stages except the PM500-1L. **The shipping tab must be removed prior to operation.** The plate is located at the end of the stage opposite the motor and is fastened to the bearing way with two Allen screws and to the carriage with two more Allen screws (see Figure 5).

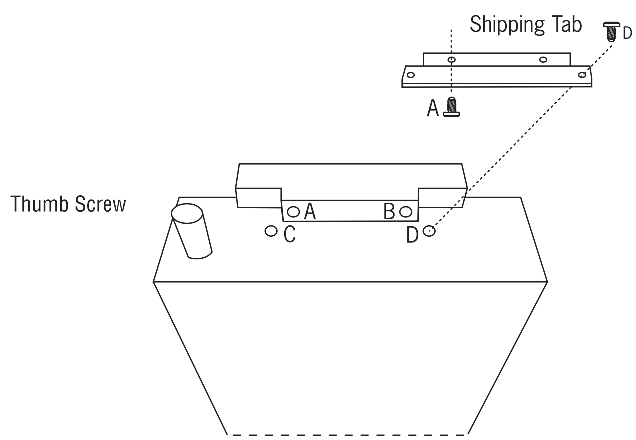


Figure 5: PM500 Stage Shipping Tab

Remove the screws, lift off the plate and replace the screws from the holes marked C and D in the picture above. The screws that were in A and B should be saved with the shipping tab. It is crucial that the shipping tab be used when shipping the stage, packing it for a move, or when placing large loads on the stage during setup. To replace the shipping tab, turn the thumbscrew until the carriage is positioned relative to the bearing way as shown above.

Save the packing material in case you need to ship the equipment. <u>If the material is unavailable then do not ship the stage</u>.

Packaging materials are available for purchase from Newport Corporation (800) 222-6440.

NOTE

The metal shipping brackets must be installed to prevent the possibility of damage and to preserve the warranty (if any).

CAUTION

Please request an empty box and/or stage locking tabs from Newport if you do not have them. Do not ship the stage without the locking tabs installed or without the proper shipping materials.

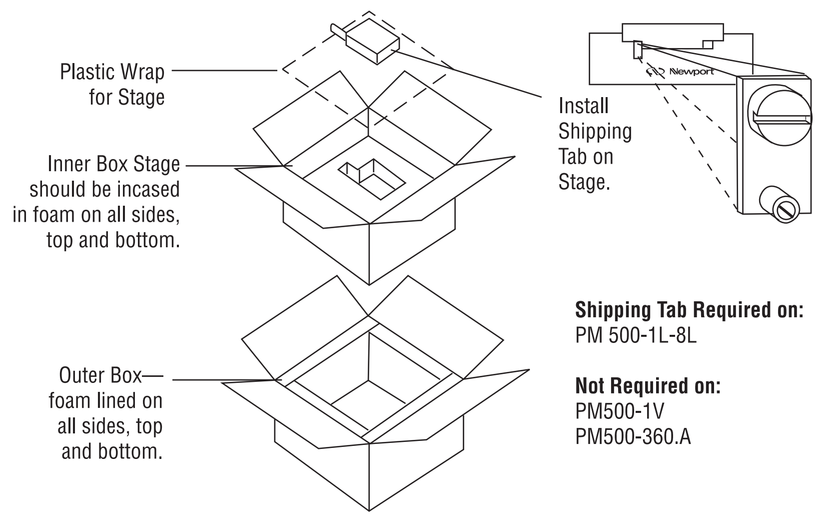


Figure 6: PM500 Stage Shipping Tab

3.3 Inspection and Damage

Inspect the shipping boxes and their contents for damage. Immediately report any damage to the shipper**.**

CAUTION

Handle the stages gently, especially when setting them on metal or other hard surfaces. Never allow the stage to drop or slap down. Mechanical shock is transmitted through the stage and can damage the bearing ways.

For your safety and the safety of your equipment you should need the following cautions:

CAUTION

- **Never attempt to power-up the system if there is evidence of shipping damage or you suspect internal damage to the unit.**
- **Do not power –up the system if it has been exposed to moisture or wetness.**
- **Do not power-up the system if any of the cables are frayed or damaged.**
- **Do not attempt to power-up the system with covers or safeties removed.**
- **Never operate the system if you suspect that it is damaged, malfunctioning or not working properly.**
- **Never insert or probe openings on the unit when plugged in or ON.**
- **Never expose the controller or devises to solvents or chemicals, especially when the system is plugged in or operating.**

If you have any concerns about the safe operation of this equipment contact Newport Corporation.

3.4 Linear Stage Shipping Tab Removal (Except PM500-1A, 1L, 1V)

To prevent damage during shipping, a temporary steel restraining plate has been installed between the stage carriage and body. The shipping tab must be removed prior to operation. The plate is located at the end of the stage opposite the motor and is fastened to the bearing way with one flat-head screw and one socket-cap screw. Remove the screws, lift off the plate and replace the screws and save the tab. Reinstall the tab when transporting the stage or when placing large loads on the stage during setup. *See* Figure 7.

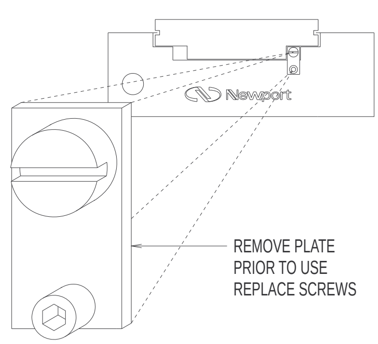


Figure 7: Removing the shipping tab before using the stage

4 Mounting Positioners and Actuators

4.1 Outline of Section 4

The following topics are covered in this section:

- Considerations for mounting motion devices
- Proper mounting of motion devices

Great care must be taken when attaching stages to work surfaces. PM500 stages are manufactured to better than micron-level flatness, whereas most optical benches and other common work surfaces are flat within only a few thousandths of an inch. Bolting a precision stage to such a (relatively) rough surface can twist or distort the bearings. This distortion can degrade the performance of the motion system.

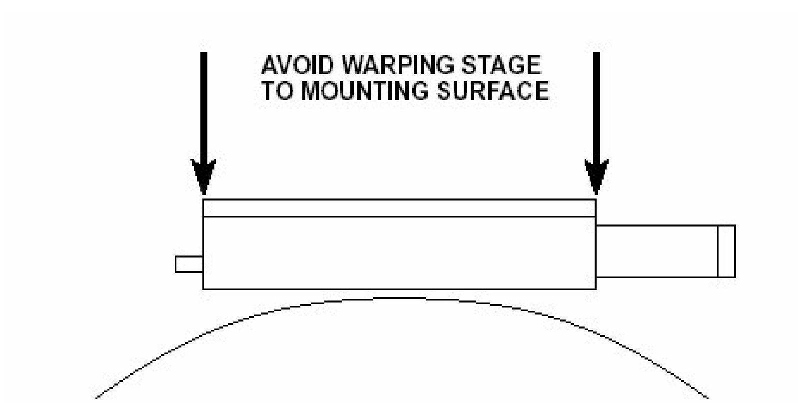


Figure 8: Considerations for mounting motion devices.

4.2 Preparing the Mounting Surface

Before mounting the stage to a surface, remove all dirt, dust, or particulate matter which may affect the flatness of the surface. For metal surfaces we recommend hand-lapping the surface with an Indian Oil Stone to remove any high spots, rough areas or burrs. Gently place the stage on its mounting surface and check that it does not rock on a high spot.

If a sufficiently flat surface is not available, we recommend not using/tightening the fourth bolt.

NOTE

For superior performance and stiffness, mount the stage to a precision-lapped plate or precision/inspection grade granite surface flat to within 5 microns Total Integrated Runout over the mounting area. If this is not possible, place three precision washers between the stage and mounting surface. If you use a forth bolt, do not tighten it.

CAUTION

PM500 stages are capable of 0.75-µm straightness of travel (over 100mm). Mounting the stage to an uneven surface can warp the stage body thus degrading the carriage trajectory. This could result in permanent damage to the stage. Any distortion of the stage body can cause the servo to "hunt" or make movement very slow.

4.3 Vertical Stage Mounting

Do not mount standard PM500 linear stages **(except the PM500-1L)** such that the carriage moves vertically (Figure 2.2) unless that stage has a factory-installed pneumatic counterbalance. The constant gravitational load on an unmodified stage may cause the servo to constantly adjust resulting in an oscillation. *See* Section 6.5 - Vertical Stages for more information on vertical stage mounting.

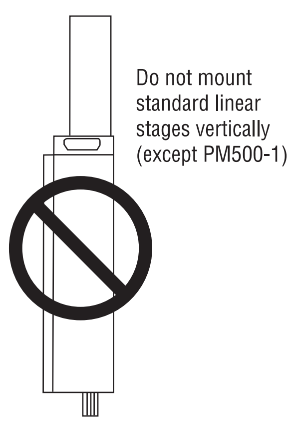


Figure 9: Do not mount standard Linear Stages vertically

4.4 Mounting PM500-1A Actuators

PM500-1A actuators are designed to replace 3/8-inch diameter micrometers and can operate a broad range of positioning components. Follow these guidelines for best performance:

- Install in positioning components that employ clamp or ring-nut mounting (ring-nut mounting is preferred avoid set screws) which impose a high point load on the actuator mounting barrel.
- Avoid over-tightening clamps on the actuator mounting barrel. This could bind the plunger, causing the system to oscillate or continuously servo.

CAUTION

Avoid mounting situations where the actuator's mounting barrel must support its own weight and/or the weight of the cable. This can cause binding of the plunger, resulting in system oscillation or continuous servoing. In some cases, damage may occur.

- In some motion systems, actuation causes the positioner and the attached actuator to move. When this cannot be avoided, support the actuator cable to allow free motion without restriction.
- Be sure to restrict full travel using soft limits, where maximum actuator travel can exceed the mechanical travel of the mount.

4.5 Mounting PM500-1A Linear Stages

PM500-1L Linear Stages may be ordered with an optional base plate for bolting the stage directly to the mounting surface (or mounting to the base stage in multistage configurations).

Mounting holes through the stage body are accessible through the carriage by aligning the carriage access holes with the mounting holes. Use the manual knob to move the stage carriage.

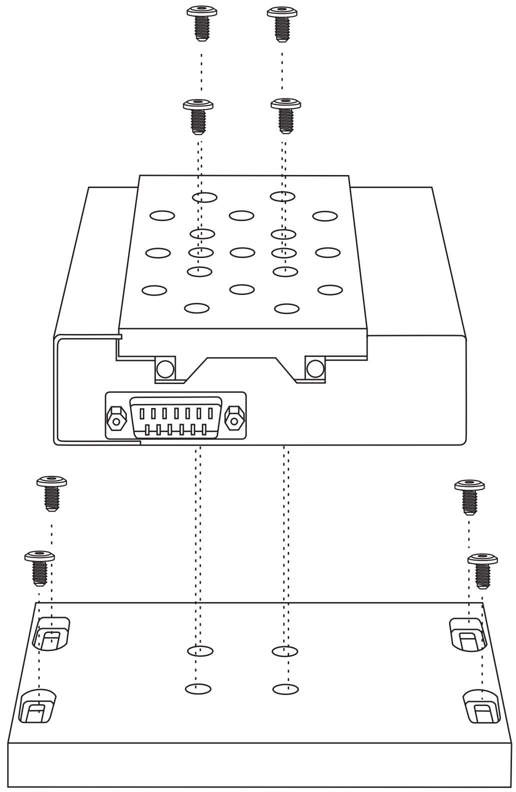


Figure 10: Mounting PM500-1 Linear Stage

The base plate will protect the stage from damage if it is affixed to an uneven mounting base. Motion Devices may be bolted together directly without the base plate.

4.6 Mounting PM500 Linear Stages (Except PM500-1L)

- Remove the button-head screws from the side covers running parallel to the stage carriage.
- Remove the side covers.
- Locate the two 1/4-20 or M6 countersunk clearance holes on each side of the stage.
- Replace the covers onto the correct sides. The cover on the scale side has a Teflon low-friction pad on its inner surface to prevent wear of the encoder read-head harness. Use extreme care to avoid contamination of the glass scale.

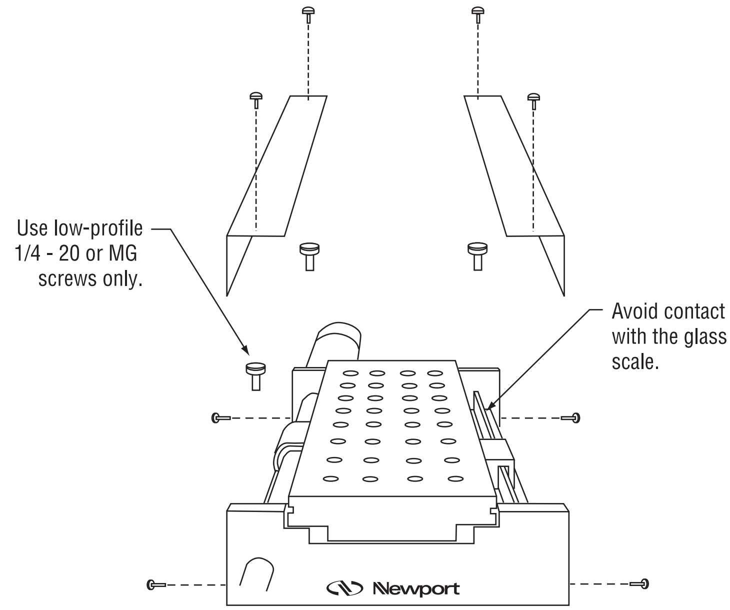
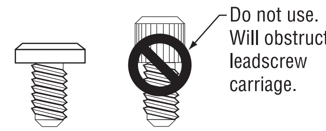


Figure 11: Exploded view of linear stage.

	CAUTION **Use only <u>low-profile</u> 1/4-20 or M6 bolts for mounting. Standard socket-head cap screws will interfere with carriage motion, damaging the stage. If these screws are not available to you, contact Newport.**	

PM500 rotary stages should be mounted with 1/4-20 x 1 1/2" screws via the four mounting thru-holes provided. Caution should be exercised that mounting hardware will not obstruct motion.

4.7 Mounting PM500 Rotary Stages

PM500 rotary stages can be mounted vertically but are not designed to handle high-torque loads. In extreme cases, such as when the load mass isn't placed about the center of rotation, the stage will stall, oscillate or continuously servo to position.

In instances where the load mass can't be centered about rotation, the load must be counterbalanced to cancel any torque effects. Stiffness of the load fixturing in this situation is a critical factor in the positioning stability of the system. For more information, see Section 6, "Attaching loads to positioners".

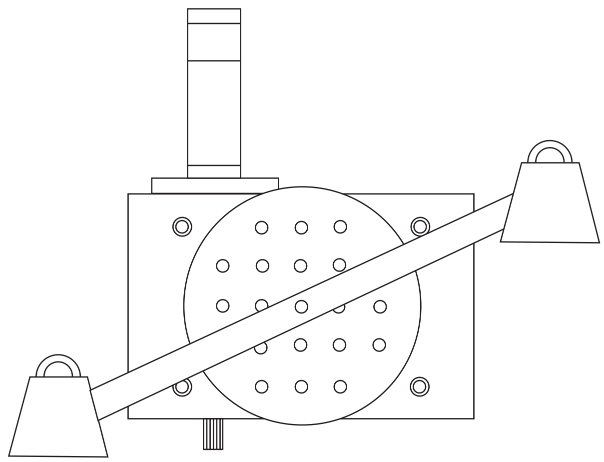


Figure 12: PM500 Rotary Stages are sensitive to high-torque loads.

NOTE

Multistage stacks using the PM500-360A
In multistage stacks, the PM500-360A should always be the top stage.

---

5 Device Controller Interconnection

5.1 Outline of Section 5

The following topics are covered in this section:

- Connection of motion devices
- Cabling considerations
- Site requirements
- Controller connections
- Placement and attachment of loads

CAUTION

Most PM500 motion devices are electrically matched to specific axis driver cards installed in your controller. Refer to Appendix Z at the back of the manual to see which device should be connected to which controller axis card. Device damage or improper operation may result if motion devices and axis cards are mixed.

Newport strongly cautions against attempted use of PM500 motion devices with other controllers.

CAUTION

Linear motion devices (stages and actuators) are supplied with PM500-L16 cables. Rotary stages are supplied with PM500-L16-R cables, which are electrically different. Rotary cables are identified by a label. It is very important that all cable-clamping screws are tightened, both at the controller and at the motion device. If the screws are not tightened, the cable connectors may loosen imperceptibly, causing erratic or uncontrolled motion.

5.2 Matching Devices and Axis Cards

Each motion device consists of a stage and an axis card. These are tuned and matched as a system.

NOTE

Be certain that the proper stage is connected to its corresponding axis card

The system settings sheet in Appendix Z will identify the axis that a particular stage is tuned with.

If you wish to change the axis assignment of a particular device, the *stage* ***and the axis card*** must be moved to the desired controller slot. Refer to Section 19 for axis card removal and replacement procedures.

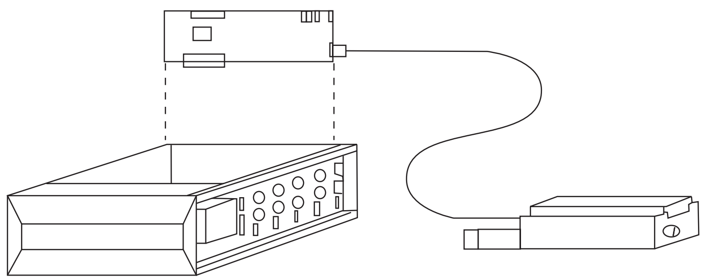


Figure 13: Assignment of axes change.

5.3 Rack Mounting

The PM500 can be rack-mounted using either rack slides or rack ears. Rack ears (PM500-R) are available from Newport. The unit also has provisions for rack slides, accessible by removing the side plates via the button-head screws. The PM500 feet can be removed for rack mounting.
Only rear panel clearance is required for cooling.

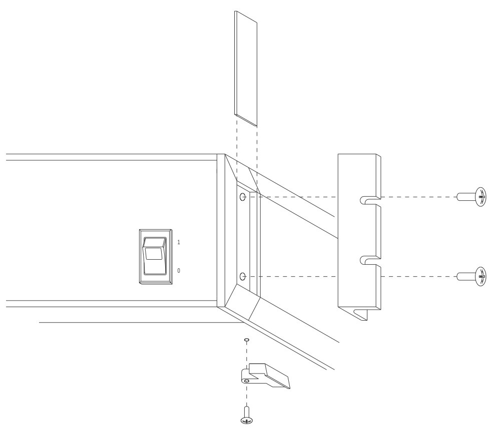


Figure 14: Exploded view of attachment of rack ears on controller.

5.4 Back Panel Connectors

Locate the following connectors on the controller's rear panel (refer to Figure 15).

- Voltage selector (slide switch)
- Remote motor shutdown (BNC connector)
- Circuit breaker
- Power cord receptacle
- Remote console operator interface (9-pin D-style connector)

WARNING

Dangerous Voltages Present.

- RS-232 port (25-pin D-style connector)
- Auxiliary port for axis options, if present (DAI, RTU)
- GPIB (IEEE-488) port (conventional Centronics parallel connector)
- Axis ports (25-pin D-style connector)

Axis assignments are X, Y, Z, A, B, C from right to left. PM500-C6 controllers have five axis ports; PM500-C6 controllers have six.

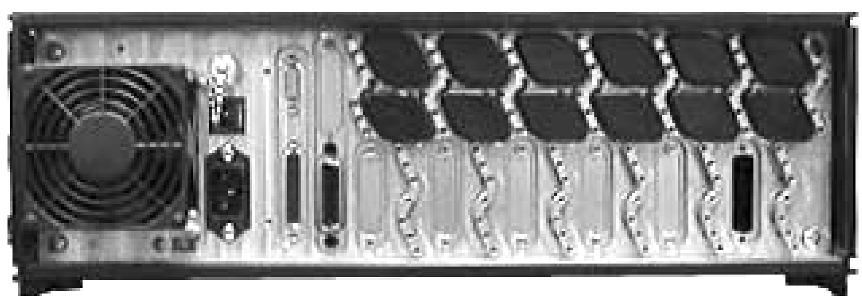


Figure 15: Rear view of the PM500-C6 6-axis controller.

CAUTION

Do not attempt to interface into controller ports with other than the proper motion device or interfaces. The PM500 controller is designed only for use with PM500 devices.

Motion devices should be connected to their specific axis cards, as recorded in the System Setting Sheet located at the back of this manual. PM500 motion devices are electrically matched to their specific axis cards and are not interchangeable without adjustment. Rotary device axis cards are not interchangeable with linear device axis cards.

5.5 Connecting Cables

⚠	**CAUTION** **Controller power should be off before connecting cables.**

Cable connections should be made with the power OFF to prevent possible damage. Cables should be securely fastened with both screws on the D-style connector. Loose screws/connections can result in erratic or uncontrolled motion. *See* Appendix A for cable pin-outs and specifications.

5.5.1 Cable types

Rotary and linear cables are not interchangeable.

- Rotary cables are marked “ROTARY”.
- Linear cables are not marked.

5.5.2 Cable routing

Tension and rubbing of the cables during motion can induce micron-scale disturbances in your setup. It is recommended that the cables be secured but not restricted. Pay particular attention to:

- Avoid mounting actuators in stages which form long lever arms.

See Section 6, “Attaching loads to positioners,” for more information.

Due to the precision signal and power requirements of the motion devices, it is not possible to reduce the cable size or increase flexibility.

No attempt should be made to modify or replace the factory provided cable. Poor performance or erratic motion can occur from improper shielded cabling.

⚠	**CAUTION** **Your warranty is voided should stage damage occur from non-factory cabling malfunctions.**

5.6 Remote Motor Shutdown Connector

An emergency motor stop BNC connector is provided on the back panel below the cooling fan. It is an emergency motor shutdown for the entire PM500 system. When tied to ground, either remotely or via the permanently attached BNC shorting cap provided, motor power is *enabled*. With the short removed, motor power is shut off, and the motors will coast to a stop. Motion will have to be re-commanded, e.g. `<axis>T`, for the axis to servo to position. The motor status upon reconnection for each axis can be defined by the **ESP** command *(see* the Command Reference Section for details).

5.7 Cooling

When installing the PM500-C6, care must be taken to avoid blocking the air intakes and exits on the rear panel. For proper operation and reliability, the top and side covers must be installed to provide adequate cooling. For servicing and adjustment, the top cover may need to be removed to operate the instrument **<u>for a short period of time</u>**. In any case, the ambient air temperature for proper operation should not exceed 50° C. There is no clearance required above, below, or to the sides for cooling.

This page is intentionally left blank

6 Attaching Loads to Positioners

6.1 Outline of Section 6

This section covers:

- Effects of vibration
- Constructing fixtures for stages
- Attaching cables and feeds

The placement, attachment, and location of loads is a critical factor in sub-micron motion control. Loads can include:

- Additional stages
- Cabling, wires or tubes that attach to the load or the sample
- Fixturing to support the sample

Follow these guidelines for attaching loads to positioners:

- Fasten loads securely, and as close as possible to the stage carriage. Avoid cantilevers or lever arms.
- Use bolts, vacuum chucks or other stable forms of attachment. Do not use tape or other temporary methods.
- Be sure the center of ***mass*** of the load corresponds to the center of the stage carriage (*see* Figure 16).
- Avoid stacking multiple stages. If you need XYZ, rotary or complex movement, consider moving the measurement assembly or the optical assembly rather than the sample.
- Avoid motion that drags or pulls cables or hoses attached to the stage or load.

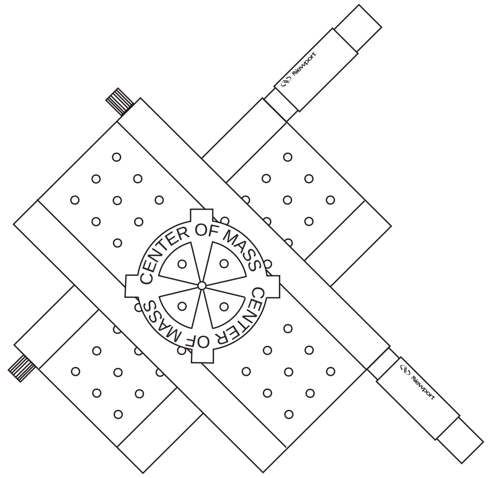


Figure 16: Placeing loads.

6.2 Effect of Vibrations

The PM500 servo system senses stage vibration if the amplitude of the vibration is greater than the system resolution. Such vibration may cause oscillations as the servo tries to compensate. This usually occurs after a movement (but not necessarily a movement of the oscillating stage). It indicates vibration or flexing of the stages or fixturing due to:

- Improper or non-rigid components
- Cantilevered loading
- Unstable load/fixturing attachment
- Dragging cables or hoses

6.3 Constructing fixtures

There are three critical factors to consider when designing and building your own fixturing for micron-scale motion: rigidity, weight, and center of mass.

6.3.1 Rigidity

Fixture rigidity is fundamental for accurate micron-scale motion. If the fixture bends or flexes during movement, the servo can overshoot or vibrate. Aluminum is lightweight and easy to machine, and is a good choice for fixtures. It also has a high inherent resonant frequency which minimizes vibration.

6.3.2 Weight

The combined load of the fixture and your sample (and the top stage in multistage configurations) should not exceed the load capacity of the stage that supports it (*see* Figure 17). When cantilevered, even lighter loads can exceed the stage capacity. Use the following formula to assess the actual load induced by cantilevered masses.

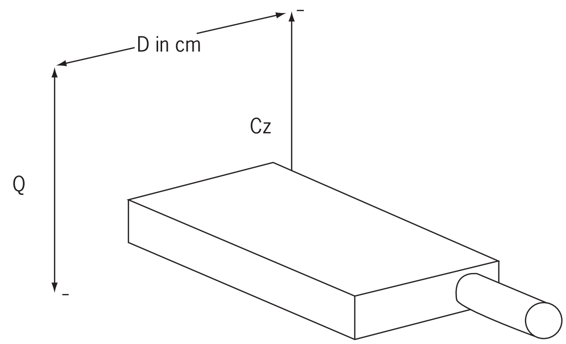


Figure 17: Calculating cantilevered loads

Formula 4.1 Calculating Cantilevered Loads
Cz (Allowable load (kg)) = Q (1 + 0.2D)
where:
Q = weight of load in kilograms. D = Distance load is from carriage center in centimeters
The load must always be smaller than the load capacity of the stage.

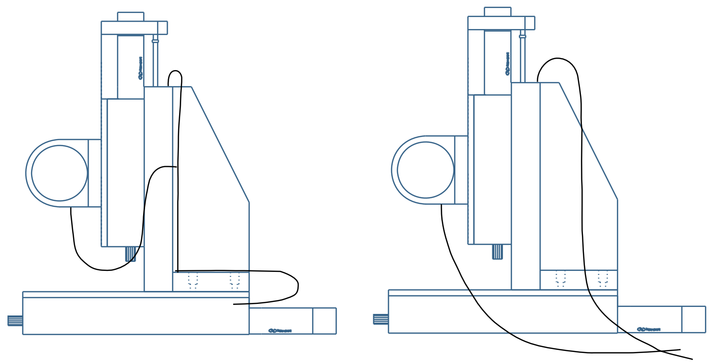
6.3.3 Locating the load

The location of the center of mass of the fixturing and load should be positioned over the center of the base stage. The load and fixture should be considered as one unit when calculating the center of mass.

When the setup prohibits the centering of the mass, a counter weight must be added to “balance” the load and bring the center of mass over the stage carriage. Use the formula in the preceding section to verify that the load capacity of the stage is not exceeded.

6.4 Attaching Cables and Feeds

Never allow stage motion to drag cables or feeds. Always secure, but do not restrict cables. Support cable from above if possible (*see* Figure 17). The dragging of cables can cause micron-level disturbances in the motion system. This can cause the motion system to attempt to compensate for the disturbance. This may appear as oscillation or vibration of a stage.



RIGHT **WRONG**

Figure 18: Attaching cables and feeds.

⚠	**CAUTION** **When tightening or placing loads on the stage, be careful when tightening screws. The tightening torque is supported by the carriage and bearings ways of the stage, not the base. Severe mechanical shock from torquing or impact can damage the bearing ways.**

NOTE

For placement of heavy loads, temporarily reinstall the shipping tab. Be sure to remove the tab before using the device .

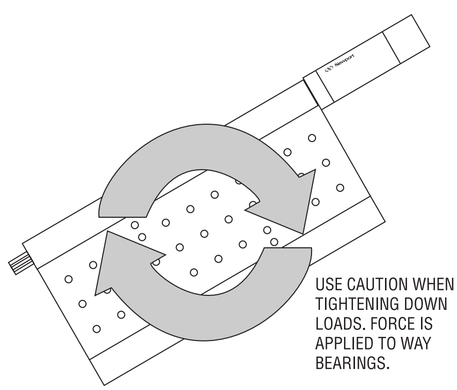


Figure 19: Placing loads or tightening screws to fasten loads.

6.5 Vertical Stages

Special PM500 linear stages are available for applications where vertical motion greater than 1 inch of travel is required. (The standard PM500-1 Linear Stage, which offers 1-inch travel, can be operated in vertical applications without modification.) There are two options that can be used separately or in combination.

- Pneumatic load compensation piston
- Ultra-precision Z-bracket

We recommend that the pneumatic load-compensation piston be used in all vertical applications. Should you decide to build your own Z-bracket, contact Newport for design guidelines.

⚠	**CAUTION** **Proper preparation of the mounting surface for Z-bracket installation is essential. Z-bracket surfaces are machined to micron-level flatness, whereas optical bench tops and other common work surfaces are flat to only a few thousands of an inch.**

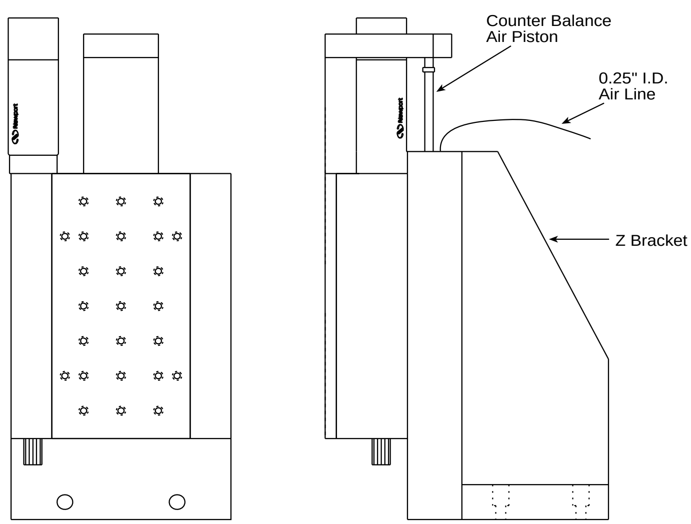


Figure 20: A typical installation of a Z-bracket.

6.5.1.1 Z-bracket mounting

Mounting the Z-bracket requires care to avoid twisting or distorting of the bracket which can compromise motion performance.

6.5.1.2 Mounting directly to the work surface

The bracket should be securely fastened to a super flat or lapped surface and attached using four bolts. Check that the stage carriage motion is unobstructed.

6.5.1.3 Mounting to another stage

Mounting the bracket to the top of another stage requires that the bracketed stage and load be treated as one unit. The bracketed stage and load should be mounted so its center of mass is centered on the carriage/carriages of the base stages. The load should be over the base stage carriage and not suspended over the side of the carriage. Avoid cantilevering the bracket stage or the load.

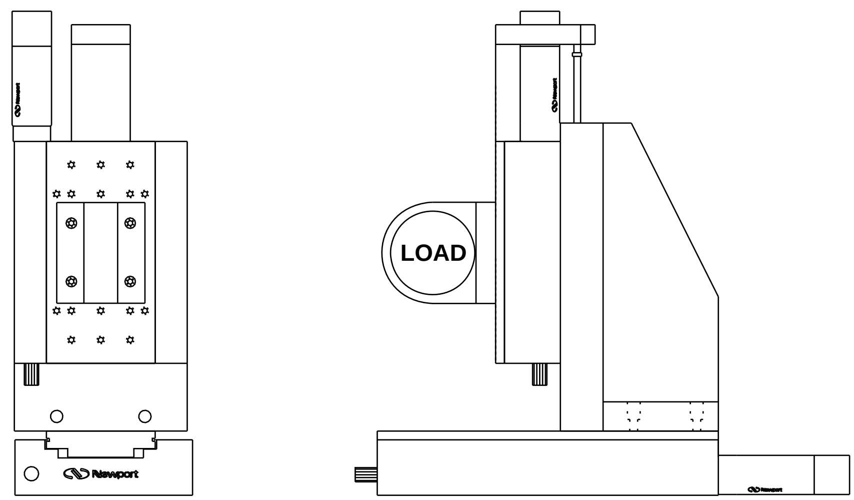


Figure 21: Mounting to another stage

⚠	**CAUTION** **Avoid exceeding the load capacity of the base stages.**

6.6 Pneumatic Load Compensation Piston

The pneumatic load-compensation piston uses air pressure to counterbalance the load on the stage. The air regulator must be precisely adjusted to counter the stage load. The regulator setting is usually made at the factory using the weight of the load given by the user.

6.6.1 Regulator compensation adjustment

The regulator can be user-adjusted by monitoring the motor current as the stage moves , increasing or decreasing the air pressure until the motor current is the same on both directions.

6.6.1.1 What you'll need

- Filtered air or nitrogen source
- Computer interface to PM500 via RS-232 or IEEE
- Actual load mounted to the stage

Before starting, set up and connect PM500 systems and stages per Sections 1–4 of this manual, turn the controller ON and verify that the computer interface is operating.

The volume of the air cylinder is approximately 1 cubic inch, thus the approximation of air pressure equals load can be used.

1. Attach and hookup regulator and air lines to a filtered air source. Air OFF.
2. Mount the actual load on the stage as it will be in your application.
3. Turn the regulator to OFF or zero pressure.
4. Turn air source ON.
5. Adjust regulator to 5 psi or slightly less.
6. Issue the command <axis letter> FØ. The stage carriage should move to center of travel.
7. Issue <axis letter>LGR-125ØØ. The stage should slowly move. (Should the stage stall, increase air pressure to lighten load.)
8. Issue <axis letter> RDCUR? Read response. Record value.
9. Issue <axis letter> M. Motor is now off, stage stops.
10. Issue <axis letter> LGR-125ØØ. The stage carriage should move in the opposite direction.
11. While the stage is underway issue <axis letter> RDCUR? Read response. Record value.

12. Adjust air pressure. Increase if motor current is higher moving up then down, decrease if motor current is higher moving down then up.

13. Repeat steps 7 through 12 until motor current is within 8 to 10 DAC counts when traveling in both directions.

6.6.2 Routing of air supply line

Secure the air line from the bracket stage to the bodies of the base stages. Allow enough tube for freedom of travel in all axis without restriction.

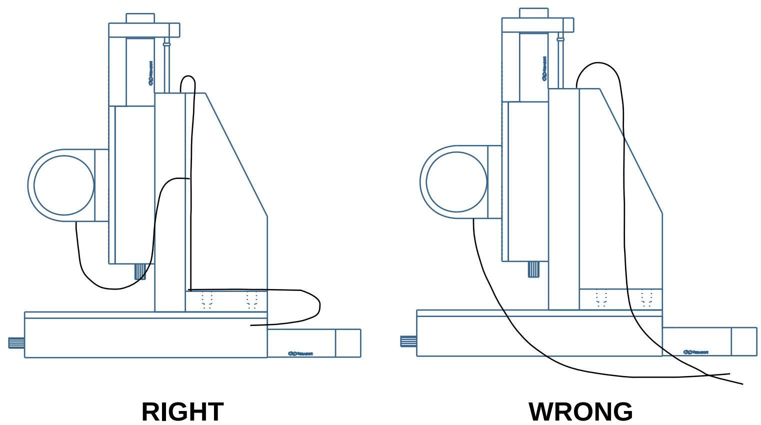


Figure 22: Routing of air supply line

7 Operation

7.1 Outline of Section 7

The following topics are covered in this section:

- Choosing an interface—RS-232/GPIB

7.2 Choosing an Interface

In general, the PM500-K Remote Control Console is useful only for manual control of very simple positioning tasks. To take full advantage of the advanced features and capabilities of the PM500 system, we recommend operation using the RS-232 or GPIB (IEEE-488) communication interfaces. The PM500-K6 will operate in conjunction with the RS-232 or GPIB interface.

Either interface is suitable for the vast majority of applications. The RS-232 interface allows the PM500 to communicate with a variety of computers with a minimum amount of effort. The GPIB interface offers a speed advantage in data intensive applications where large blocks of data must be transferred. The GPIB interface is also the best choice for situations requiring support of multiple instruments, coordination/synchronization of multiple systems and status-reporting capabilities.

The following factors should be considered when choosing which interface to use:

7.2.1 RS-232

Advantages

- Inexpensive hardware and cabling
- Ports standard on most personal computers
- Relatively easy to program

Disadvantages

- Low to moderate speed (up to 19k baud or 2k cps)
- Supports only one instrument per port
- No synchronization or coordination facilities
- Cabling, pin-outs, handshaking and protocols not standardized

7.2.2 GPIB

Advantages

- Moderate to high speed (up to 10^6 characters per second)
- Supports many instruments per GPIB controller card
- Excellent synchronization/coordination facilities
- Standardized cabling and pin-outs
- Standardized protocols

Disadvantages

- GPIB controller cards cost extra
- Cabling lengths are limited, and cabling is costly
- Programming is somewhat more complex (but more powerful) than RS-232

8 PM500-K6 Remote Control Console

8.1 Outline of Section 8

The following topics are covered in this section:

- Connecting the console
- Operating modes
- Controlling the PM500-K6
- Menu key functions

Features

The PM500-K6 Remote Control Console provides manual control over most motion operations and continuous position display for three selected axes. The PM500-K6 provides control for up to six axes and will operate earlier 5-axis PM500 controllers without modification. The PM500-K6 can be used in conjunction with GPIB or RS-232 interface control.

- 6-axis control key sets FWD/REV for each axis
- Settable incremental and slewing axis motions
- Three velocity range switches
- 10X multiplier key allows selection of six step sizes and six velocities.
- Displays up to three axis positions via 40-character backlit LCD display
- Self-prompting programming menu
- Selectable parameters include speed values, key direction, and display sign conventions—all of which may be stored in non-volatile memory
- Complete configurability via RS-232 or GPIB interfaces

8.2 Connection

WARNING

Do not attempt to connect anything other than the PM500-K or K6 Remote Control Console to the operator interface port. Voltages are present at this interface which can cause personal injury. Uncontrolled motion or damage to system may also result.

Follow these steps to connect the PM500-K6 Remote Control Console to the PM500-C6 Controller:

- Turn the PM500 controller OFF.
- Connect the Operator Interface Module cable to the PM500-K6 and the PM500 Controller rear port marked “Operator Interface”. Secure using both screws.

DO NOT FORCE THE CONNECTOR.

- Leave the RS-232 or GPIB interface in place.
- Turn the PM500 controller ON. An identification message will appear momentarily on the PM500-K6 display.

8.3 Operating Modes

The PM500-K6 has two operating modes:

Execute Mode: Keypad keys directly execute motion; display shows position of selected axes.

Menu Mode: Keypad keys are mapped per menu instructions; display shows menu options.

To enter Menu mode, press the UP or DOWN keys in the Menu/Set keypad. To return to Execute mode without saving, press the CLEAR key.

8.3.1 Execute mode operation

Briefly pressing a keypad key causes a jog motion; if you hold the keypad key down, after a moment continuous slewing will occur. The displays will show the position of selected axes.

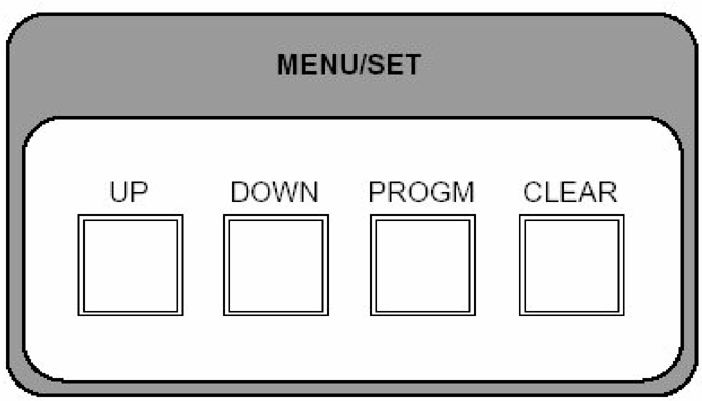
The size of the jog and slew speed depend on the speed range selected. Step size and slew speed for each speed range may be adjusted via the PM500-K6 setup menu, GPIB interface or RS-232 interface. Default axes are X (horizontal) and Z (vertical). Axes can be assigned to any key via the setup menu.

NOTE

The PM500-C6 will ignore any instructions from the PM500-K6 when executing any command from the GPIB or RS-232 interface. Position update will always function unless disabled, regardless of the origin of system control.

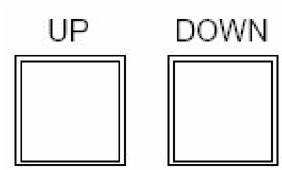
8.3.2 Menu mode operation

The PM500-K6 menu system enables the user to perform basic operating functions from the remote console, such as turning motors ON and OFF, clearing error conditions and Auto-Zeroing axes. A setup menu, which is accessed through the main menu, also permits the modification of certain key functions and the selection and storing of menu parameters.

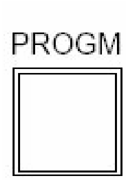


8.4 Menu Key Functions

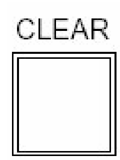
The menu/Set functions are as follows:



Pressing the UP or DOWN key enters the menu system. Each subsequent UP or DOWN keystroke sequences through the menu options.



The PROGM key executes and stores the current selection.



The CLEAR key exits the menu system (without saving if menu instructions do not request it) and the PM500-K6 returns to Execute mode.

8.4.1 Main Menu

From the Main Menu, you can perform basic operating functions such as turning motors on and off, clearing error conditions and Auto-Zeroing axes. The Main Menu also provides access to the setup menu.

To enter Menu mode from Execute mode, press the UP or DOWN key in the Menu/Set keypad. The display will show a Main Menu selection. Continue pressing the UP or DOWN keys to cycle through the selections shown on the left below:

Main Menu	***Setup Menu***
1. Press **PROGM** to enter Setup Menu	
2. Press **PROGM** to initialize stage	8. Press **PROGM** to save stage locations
3. Press **PROGM** to clear position of axis	7. Press **PROGM** to init stage locations
4. Press **PROGM** to Auto Zero all axes	6. Press **PROGM** to default stage locations
5. Press **PROGM** to clear positions of all axes	5. Press **PROGM** to save new OIM parameters
6. Press **PROGM** to read error condition	4. Press **PROGM** to init OIM parameters
7. Press **PROGM** to clear error condition	3. Press **PROGM** to default OIM parameters
8. Press **PROGM** to turn all motors ON	2. Press **PROGM** to change axes displayed
9. Press **PROGM** to turn all motors OFF	1. Press **PROGM** to change position display

**** Press appropriate speed key to modify parameter***

Press the PROGM key to execute the selection in the display.

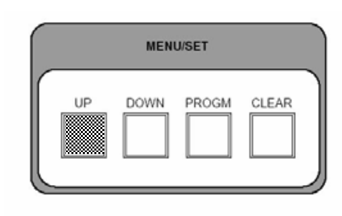
8.4.2 Setup Menu

The Setup Menu lets you modify certain key functions and facilitates the selection and storing of menu parameters. To access the Setup Menu, cycle through the main menu until the display reads `“Press PROGM to enter Setup Menu.”` Pressing the PROGM key opens the setup menu.

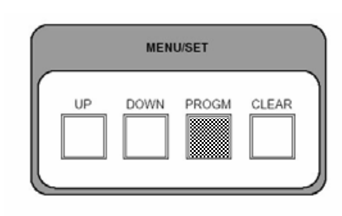
All Setup Menu selections that cause move changes, speed changes, or direction changes are permanently stored in the controller’s non-volatile memory and will be used for all subsequent power-ups. The parameters can be cleared by selecting `“Initialize PM500-K6 parameters”` or via the **DEFOM 1** command.

All of the above functions for configuring the Remote Control Console can be implemented remotely through the communications interfaces via appropriate commands. *See* the PM500-K6 portion of Section 16 - Command Reference for a complete list of commands.

Example: Turn motors OFF

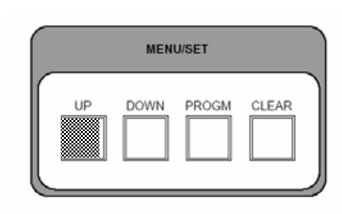


Press the UP key in the Menu/Set keypad to enter Menu mode, then sequence through the main menu by pressing the UP key repeatedly until the display selection reads “`Press PROGM to turn all motors OFF.`”

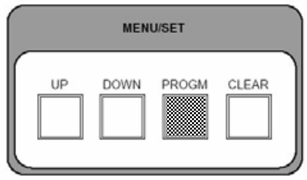


Press the PROGM key in the Menu/Set keypad to execute the motor off command. The motors will be turned off for all axes, allowing you to move the stage via the manual knob opposite the motor. The PM500-K6 display will update position and return to execute mode

Example: Turn motors ON

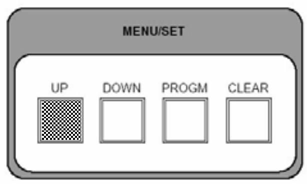


Press the UP key in the Menu/Set keypad to enter Menu mode, then sequence through the main menu by pressing the UP key until the display selection reads `“Press PROGM to turn all motors ON.”`

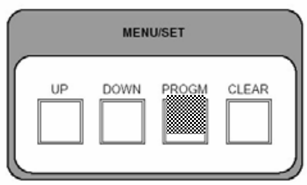


Press PROGM to execute clear position command. The PM500-K6 will clear the position register of the PM500 and return to execute mode.

Example: Clear Axes Position



Press the UP key in the Menu/Set keypad, and sequence through the Main menu until the display selection reads `“Press PROGM to clear position of all axes.”`



Press PROGM to execute clear position command. The PM500-K6 will clear the position register of the PM500 and return to execute mode.

8.5 Controlling the PM500-K6 via RS-232/GPIB

CPM500-K6 functions may also be modified via RS-232 or GPIB communication. User settings are saved in non-volatile memory for all subsequent power-ups. The PM500-K6 is always active when enabled regardless of the origin of system control. The PM500-K6 will be ignored when the system is executing commands from RS-232 or GPIB control. The PM500-K6 has no “go to local” mode.

For details on PM500-K6 commands, refer to the PM500-K6 commands in Section 16 - Command Reference section of this manual. Commands for the PM500-K6 Remote Control Console can be identified by their **“OM”** prefix.

9 RS-232 Computer Interface

9.1 Outline of Section 9

The following topics are covered in this section:

- Setup
- Operation

9.2 Setup and Operation

The RS-232 port allows the PM500 to communicate with computers having a RS-232 port. High-level language programs can be written to customize control of the PM500.

The PM500 RS-232 port is a female 25-pin D-style RS-232.

9.2.1 What you'll need

- RS-232 port or add-on card installed and setup properly in your computer
- RS-232 cable
- Terminal program or other programming language to interface to the RS-232 port

9.2.2 Setting the RS-232 parameters

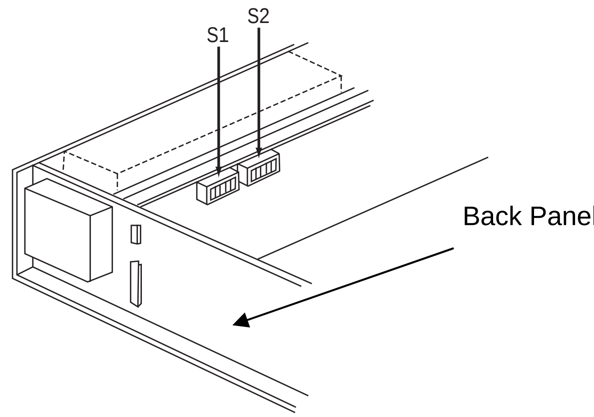


Figure 23: DIP switch location in PM500-C6

9.2.3 Configuring for RS-232 communications

The PM500 is shipped configured for GPIB communication. You'll need to reconfigure the controller for RS-232. This is done by the mode switches located on the SBC board inside the controller.

- Turn the controller OFF.
- Remove the controller cover via the four Phillips-head screws.
- The controller board is located next to the controller power supply.
- On the controller locate the two banks of mode switches along the top of the board near the rear of the controller.
- The RS-232 is enabled/configured by the switches on the S2 bank.

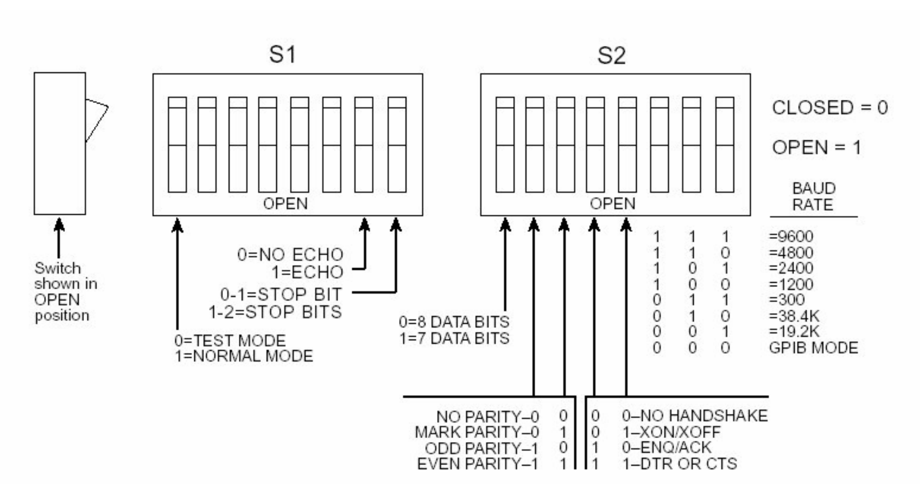


Figure 24: Changing the RS-232 parameters.

9.2.4 RS-232 interface specifications

The RS-232 port on the back panel of the PM500 controller is a 25-pin female connector.

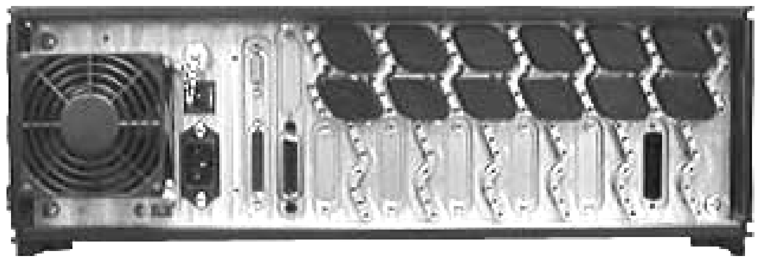


Figure 25: Back panel of the PM500 controller.

9.2.5 Description of the PM500 RS-232 port

The PM500 RS-232 port is a db25-pin female connector.

Pin #	***Description***
1	**Protective ground.** Electrical ground frame and AC power ground.
2	**Transmitted Data.** Data originated by the terminal or computer to be received by the PM500 (input).
3	**Received Data.** Data transmitted from the PM500 to the terminal or computer (output).
4	**Request to Send.** Indicates to the PM500 that the terminal or computer is ready to transmit data (output).
5	**Clear to Send.** Indicates to the terminal or computer that the PM500 is ready to transmit data (output).
6	**Data Set Ready.** Indicates to the computer or terminal that the PM500 is ON (output).
7	**Signal Ground.** Establishes a common electrical reference between the PM500 and the terminal or computer.
8	**Received Line Signal detector.** Indicates to the terminal or computer that the PM500 is ON (output).
20	**Data Terminal Ready**. Indicates to the terminal or computer that the PM500 is ready to receive data (input).

Voltage specifications for the RS-232 interface	
Open-Circuit Voltage Permissible	+25 volts +15 volts
Logical“0” “ON” Condition or “Space	+5 volts
Noise margin	+3 volts
Transition Region	**0 volts**
Noise Margin	-3 volts
Logical “1” “OFF” Condition or “MARK”	-5 volts
Permissible Open-Circuit Voltage	-15 volts -25 volts

9.2.6 RS-232 cable length restrictions

The PM500 RS-232 connector follows the standard EIA RS-232-C protocol. The length of the cable should be no longer than 50 feet (15 m), assuming the load capacitance at the interface point is the worst-case value of 2,500-pf. Longer cables are routinely used, especially in point-to-point configurations, when you know that total load capacitance will not exceed the 2,500-pf maximum.

9.2.7 Sample RS-232 cable configurations

Three examples of typical RS-232 interface configurations are presented here. Note that in all cases, pin 1, 2, 3, and 7 of the PM500 must be connected for the interface to operate safely. A “+” signifies the electrical connection of two pins at the device. Note that “transmitted data” pins and “received data” pins vary among computers (DCE) and terminals (DTE).

Example 1: Three-wire hookup		
PM500 pins	***Host device (DTC or DTE)***	***Description***
1	1	Protective ground
7	7	Signal ground
2	3 or 2	Data to PM500
3	2 or 3	Data from PM500

Example 2: Three-wire hookup with XON / XOFF		
PM500 pins	**Host device (DTC or DTE)**	**Description**
1	1	Protective ground
7	7	Signal ground
2	3 or 2	Data to PM500
3	2 or 3	Data from PM500
4+5	N/C	Enable transmitter on PM500
N/C	4+5	Enable transmitter on DCE/DTE

Example 3: Full hardware handshake		
PM500 pins	**Host device (DTC or DTE)**	**Description**
1	1	Protective ground
7	7	Signal ground
2	3 or 2	Data to PM500
3	2 or 3	Data from PM500
4	20	Shows PM500 is ready
5	5	Shows PM500 output
20	4	Stops host output

9.3 RS-232 Operation

9.3.1 RS-232 Handshaking Modes

The PM500 has four modes of RS-232 handshaking. These modes may be selected by changing the switch settings on the controller board. The switch locations are documented in Figure 23. The four modes are:

- No handshaking
- XON/XOFF
- ENQ/ACK
- Hardware: RTS/CTS or DTR

Handshaking is generally not required in a buffered system. The commands are short and the processing time (servo time) for most commands is longer than the transmission time. The PM500 is delivered in the following configuration (when RS-232 baud switch is set in an RS-232 mode):

- baud rate–user select
- 7 data bits
- Even parity
- 2 stop bits
- Hardware RTS/CTS handshaking
- Echo mode on

9.3.1.1 Considerations for configuring the RS-232 handshaking mode

When using the system in RS-232 mode, overall system goals must be considered. If the host computer interface is a simple, one-action/one-reaction type of system, then sending commands and acknowledging them via the STAT command is the easiest method of interfacing. This method is referred to as “synchronous” interfacing.

If the host computer has an interrupt-driven serial interface which is accessible to your program, then sending commands, acknowledging them, and receiving the `DONE` status is the most efficient method of interfacing. This method is referred to as “asynchronous” interfacing. The default configuration allows this type of interfacing with the echo ON.

9.3.2 Synchronous RS-232 Interfacing

This method involves sending commands, acknowledging them, and polling the status of an axis via the **STAT** command. This is the simplest method of interfacing. The required configuration command is:

SCUM1
SENAINT $87E
(*See* **SENAINT** command—Section 16 - Command Reference.)

- CR LF w/CR terminator
- no echo
- sign on
- acknowledge response
- no done response
- fixed format response
- no immediate limit reporting
- all axes on limit of one or more axes stop

A sample interface process for the synchronous communication is:

User Sends	***User Reads***	***Response Means***
`XG 1000`	`XA`	Command accepted
`YG 1000`	`YA`	Command accepted
`XSTAT`	`XB`	X axis busy
`YSTAT`	`YB`	Y axis busy
`XSTAT`	`XB`	X axis busy
`YSTAT`	`YB`	Y axis busy

User repeats **STAT** queries until **STAT** return is “`D`” (done)

	`XD`	X axis done
	`YD`	Y axis done

9.3.2.1 Asynchronous RS-232 interfacing

This method involves sending commands, acknowledging them, and waiting for the status of an axis response. This is the most efficient method of interfacing. The required configuration command is:

SCUM1
SENAINT $876 (*See* **SENAINT** command—Section 16 - Command Reference.)

- CR LF w/CR terminator
- sign on
- acknowledge response
- done response
- fixed format response
- no immediate limit reporting
- all axes on limit of one or more axes stop

A sample process for an asynchronous communication interface is:

User Sends	*User Reads*	*Response Means*
`XG1000`	`XA`	Command accepted
`YG1000`	`YA`	Command accepted
	`XD`	X axis done
	`YD`	Y axis done

Notice that this method uses the least amount of interface overhead. This is why it's more efficient than the synchronous method. However, if the computer is simply waiting for the **XD** or **YD** response, using the synchronous method may be easier.

9.3.3 RS-232 default configuration interfacing

This method involves sending commands, receiving the command echo, acknowledging commands, and waiting for the status of an axis to respond asynchronously. The default configuration command is:

SCUM1
SENAINT $836 (*See* **SENAINT** command—Section 16 - Command Reference.)

- CR LF w/CR terminator
- echo
- sign on
- acknowledge response
- done response
- fixed format response

- no immediate limit reporting
- transfer all axes on limit

The interface process is:

User Sends	*User Reads*	*Response Means*
`XG 1000`	`XG 1000`	Echo
	`XA`	Command accepted
`YG 1000`	`YG 1000`	Echo
	`YA`	Command accepted
	`XD`	X axis done
	`YD`	Y axis done

NOTE

An important point about echoes: If a command entry is in process, each character is echoed as it is received. When a command becomes complete and the command entry is in process, the [axis]A or the [axis]D responses will be delayed until the command being entered is terminated.

9.3.4 Disabling RS-232 responses

This method involves sending the command, then waiting for the done response to that command. The read responses are also different with this method.

A sample interface process with disabled responses is:

User sends	*User reads*	*Response means*
`XG 1000`	Nothing	No immediate response
`YG 1000`	Nothing	No immediate response
	`XD`	Response when done
	`YD`	Response when done

- This method may seem simple, but it has some drawbacks:
- The command has no immediate response. The host computer has no knowledge of the action being taken by the command, so the echo mode must normally be used.
- If a limit occurs, the response will be: `XL.` This requires a more sophisticated response-checking routine.
- The read response does not indicate limit or motor off conditions.

10 RS-232 Quick Start

10.1 Outline of Section 10

The following topics are covered in this section:

- PM500-C6 configuration

10.2 PM500-C6 configuration for RS-232 (Quick Start)

10.2.1 What you'll need

RS-232 cable (A 25-pin female connector is required for the PM500 back panel; the cable configuration at other end will depends on your computer. *See* Section 9 - RS-232 Computer interface, for RS-232 cabling configuration.)

The PM500-C also requires configuration for RS-232 communication. This is done via the S1 and S2 DIP switches located on the top edge of the controller board, next to the power supply (Figure 26).

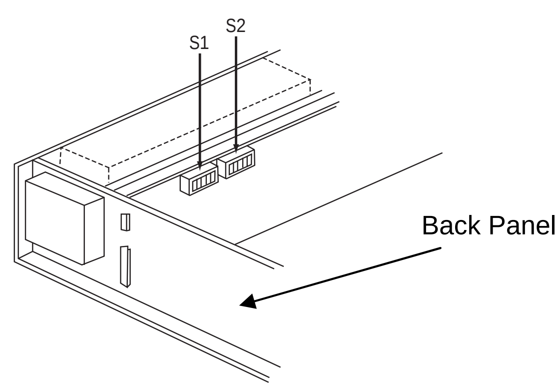


Figure 26: DIP switch location in PM500-C6

CAUTION

The PM500-C6 must be turned off before changing DIP switch settings

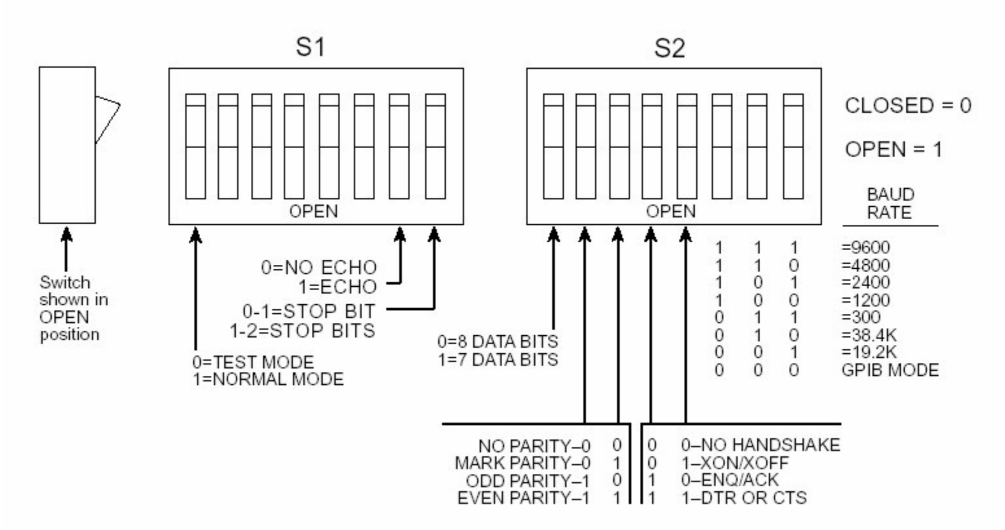


Figure 27: The RS-232 parameters are changed via S1 and S2 switch .

After replacing the cover and restarting the PM500-C6, launch the Hyper Terminal Program, type the PM500 command `COMOPT3`, and then press Enter or Return. This command enables completion of signaling and echoing.

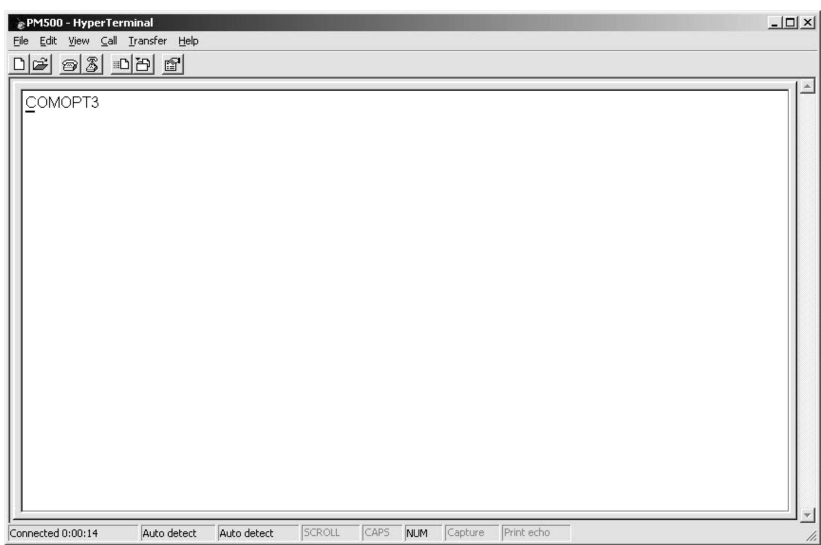


Figure 28: Command to complete signaling and echoing.

10.2.2 Verifying communication

To test successful configuration of the Terminal Program and the PM500 motion controller, type `SCUM0` and press Enter or Return. The PM500 should respond with the firmware version as shown in Figure 29.

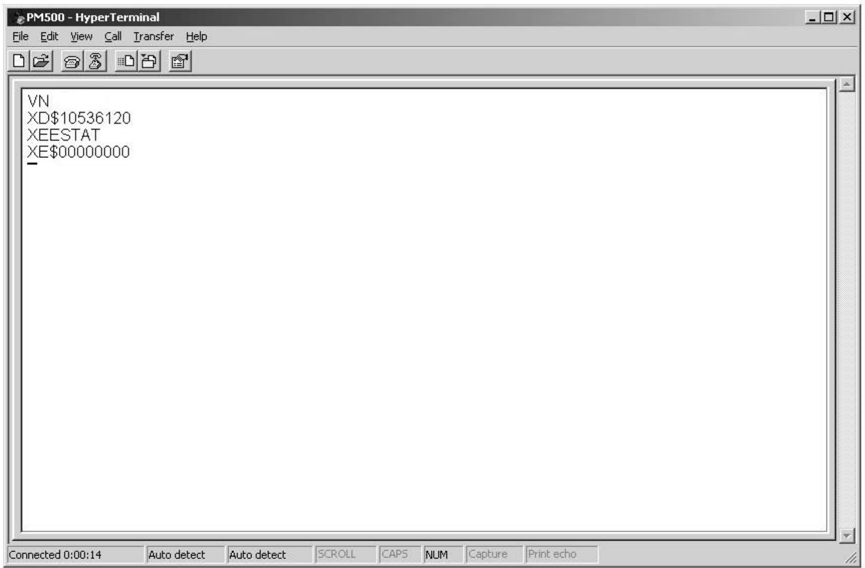


Figure 29: PM500 response to firmware version query.

Refer to Section 16 - Command Reference for a complete list of RS-232 commands.

This page is intentionally left blank

11 GRIP (IEEE-488) Computer Interface

11.1 Outline of Section 11

The following topics are covered in this section:

- Setup and operation
- GPIB Quick Start with National Instruments' Interactive Control Software

11.2 Setup and Operation

The IEEE- 488 port allows the PM500 to communicate with computers and other devices that have a GPIB port. Free instrument drivers for the PM500 and other GPIB-compatible Newport instruments are available for LabVIEW, in Windows.

Commands are sent to the PM500 controller through the GPIB port via a standard IEEE-488 cable.

11.2.1 What you'll need

- GPIB card installed and setup properly in your computer
- GPIB cable
- Terminal program or other programming language to interface to the GPIB card

11.2.2 Setting the GPIB address

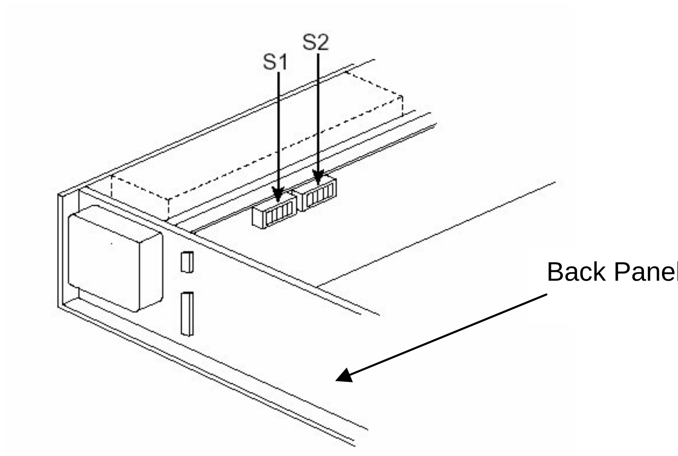


Figure 30: DIP switch location in PM500-C6

The default GPIB address for the PM500 as shipped from the factory is 1. Should the default address be in conflict with another GPIB device address, you may change the PM500 controller address to any other valid GPIB address by setting the mode switches located on the controller board inside the controller. Follow these steps to change the GPIB address:

- Turn the controller OFF.
- Remove the controller cover via the four Philips-head screws.
- The controller board is located next to the controller power supply.
- On the controller locate the two banks of mode switches along the top of the board near the rear of the controller.
- The GPIB address is selected by switches 2–6 on the S1 bank (closest to rear of controller).

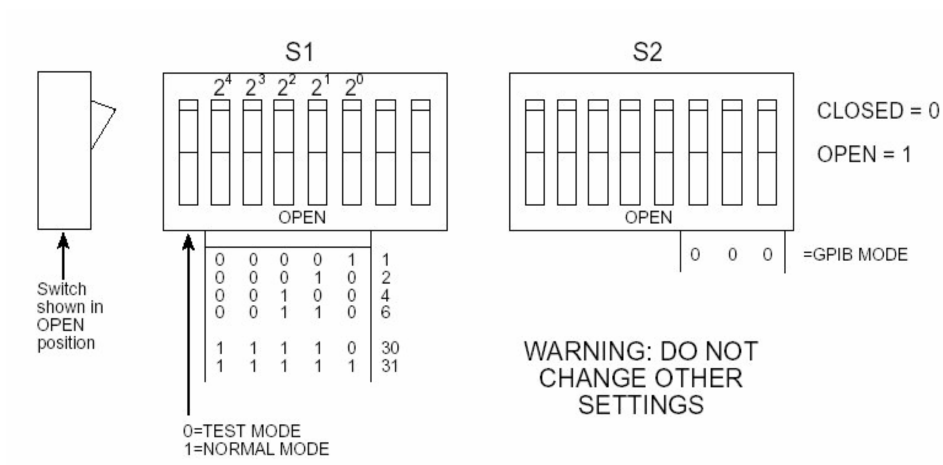


Figure 31: The GPIB address is changed via S1 and S2 switch banks.

11.3 GPIB Quick Start with Interactive Control Software

In order to begin communicating with your PM500 quickly, try the following example using Interactive Control Software by National Instruments (starting from version 1.6), a terminal emulation program included with National Instruments' IEEE-488 boards.

⚠	**CAUTION** **The following example is for use with a National Instruments IEEE-488 interface card only.**

Windows: Double click on the Interactive Control icon.

Interactive Control should display a colon prompt (**:**). Type in the underlined text; lower case text are returns from Interactive Control . An explanation of the commands is given in the right-hand column.

`:IBFIND` `enter board/device name: DEV1`	*DEV1 is the default name given to address 1, unless it has been changed by the user.*
`ID=12345-`should be a number	*Should you encounter an error, check the error codes and refer to your GPIB manual for correction procedures.*
`:IBWRT` `enter string: "XR\r\n"`	*Read position of X axis. Use quotations and proper case*
`:IBRD` enter byte count: `25`	*How many bytes to read back*
`4e 57 32 5d 7e 0d  XD+123456.7` `30 36 45 2d 39 0d`	*Should return with position reading in HEX and ACSII*

Refer to Section 16 - Command Reference for a complete list of GPIB/RS-232 commands for the PM500.

11.4 4.2.1.2 IEEE-488 Operation

The following capabilities are supported in the PM500's IEEE-488 interface:

SH1	Complete source handshake
AH1	Complete acceptor
T6	Basic talker, serial poll-unaddressed if MLA
L4	Basic listener, serial poll-unaddressed if MTA
SR1	Complete service request capabilities
RL2	Remote local, no local lockout
PP1	Parallel poll, remotely configurable
DC1	Device clear, complete capability
DT1	Device trigger, complete capability
C0	No controller
E1/2	Tri-state drivers for parallel poll

In addition to the basic hardware requirements, close attention to IEEE Std 728-1982, Recommended Practice for Code and Format Conventions, has been followed.

The IEEE-488 interface is shipped to the customer as the active interface. The factory default address is 1. The address may be changed by changing the switch settings on the Controller Board.

When using the system in IEEE-488 mode, the only considerations for interface functions are related to how the user wishes his response to be made. If the following recommended procedures are followed, the simplest interfacing will result.

The most basic operation uses the **STAT** command to determine the condition of an axis. The following example shows the sequence which is used:

User sends	***User reads***	***Response means***
`XG 1000`	`Nothing`	No immediate response
`XG 1000`	`Nothing`	No immediate response
`XSTAT`	`XB`	x axis busy
`YSTAT`	`YB`	y axis busy
`XSTAT`	`XB`	–
`YSTAT`	`YB`	–
	–	–
	–	–
	–	–
`XSTAT`	`XD`	x axis done
`YSTAT`	`YD`	y axis done

The most powerful interface method is the IEEE-488 service request function. This method reduces your computer overhead and also reduces the bus activity considerably. This method can be enhanced by using parallel poll in conjunction with serial poll. The following sequence shows an application:

User sends	***User reads***	***Response means***
`SRQCTL $EF`	`Nothing`	Enables SRQ
User commences motion:		
`XG 100`	`Nothing`	No immediate response
`YG 134`	`Nothing`	No immediate response
User gets SRQ:		
`Parallel poll`	`$01`	This device SRQed
`Serial poll`	`$80`	X axis done
User gets SRQ:		
`Parallel poll`	`$01`	This device SRQed
`Serial poll`	`$81`	Y axis done

This page is intentionally left blank

12 Motion Programming

12.1 Outline of Section 12

The following topics are covered in this section:

- Command Types and their Responses
- Command Syntax
- Status Characters
- GPIB basics
- SRQ Assertion (GPIB)
- Serial Polling (GPIB)

12.2 Command Types and Their Responses

There are two types of commands as defined by IEEE-488.2 standard.

12.2.1 Device Independent commands

These are commands that apply to the IEEE-488.2 software standard and are common to all IEEE-488.2 compliant instruments, *hence the term device independent.* The PM500 does not support device independent commands but is Blue Ribbon compliant to the IEEE-488.1 hardware standard.

12.2.2 Device dependent commands

These are commands that are specific to a particular instrument. Device dependent commands are not compatible with other instruments.

There are 4 command types in the system:

- System commands
- Motion commands
- Read/status commands
- Parametric data commands

12.2.3 Command Types

12.2.3.1 System Control Command

System Control commands allow global recovery/modification of the system's configuration/operation.

12.2.3.2 RS-232

In RS-232 mode, system commands do not allow acknowledge reporting upon receipt of a command. When the command is complete or an error has occurred, the system will return a status message. The presence of the status responses are controlled using the **ENAINT** or **COMOPT** command. The **COMOPT** command offers a preselected menu of various communication options. The **ENAINT** allows customization of all communication parameters.

12.2.3.3 IEEE-488

In IEEE-488 mode, the responses from these commands may be determined by issuing a read/status command or performing an interface service request (SRQ) function.

12.2.4 Motion Response

The second type of command is a motion command. The commands cause servo motion to occur in the specified axis.

12.2.4.1 RS-232

RS-232 system returns and responses to motion commands will perform as follows:

Mode	System Return	Comments
Command received	`<axis><A>`	"A" Acknowledge
Command completed	`<axis><D>`	"D" Done
Limit event occurs	`<axis><L>`	"L" Limit
Motor off (caused externally)	`<axis><M>`	"M" Motor Off
Error occurs*	`<axis><E>`	"E" Error

When an error occurs the user should query the device's STAT, ESTAT responses respectively, If the ESTAT register bit 22 is high, the EESTAT register should be queried to identify the system level error.

The presence of the "acknowledge" and "done" response can be controlled by the **ENAINT** command. *See* the **ENAINT** command in the Command Reference section for a further description of command acknowledgment.

12.2.4.2 IEEE-488

In IEEE-488 mode the response from these commands may be determined by issuing a "READ/STATUS", i.e. `<axis>R?`, `<axis>STAT?`, command or by performing an interface serial poll function.

The serial poll will generate a response that has the following bit definitions:

Bits	***Decimal***	***Description***
7	128	"Done" motion completed successfully
6	64	"SRQ" Service Request
5	32	"Fault" limit, or hardware failure.
Query ESTAT-EESTAT		
4	16	MAV: Message Available
3	8	axis ID
2	4	axis ID
1	2	axis ID
0	1	axis ID

The ID specifier is interpreted as follows:

ID	**Subsystem**
$01	X axis
$02	Y axis
$04	Z axis
$08	A axis
$10	B axis
$1F	System - all bits on

A serial poll response with the "Fault" bit (5) set indicates that error status is available from querying the **ESTAT** value.

NOTE

System default at power-on is Serial Poll Register disabled (SRQCTL$Ø) thus serial polls will generate no response under any circumstance. The user must enable the Serial Poll Register each time after the system is powered-on or restarted. Recommended masking value of the Serial Poll Register is SRQCTL$FF/SSRQCTL$EF, which enables all responses and axes.

12.2.5 Read/Status Commands

The third type of command is a read status command that requests real-time system states. The placement of a query `(?)` is required with these commands.

12.2.5.1 RS-232

In RS-232, the “acknowledge” reporting is suppressed upon receipt of a Read/Status command. The system returns a response that includes the system status as well as any additionally requested data. The output is buffered to allow operation of interrupt driven systems with independent input/output data paths.

12.2.5.2 IEEE-488

In IEEE-488 mode, status reporting is the same as in RS-232. In order to assure valid data responses, the user must read data from the IEEE-488 interface every time a read/status command is sent to the controller.

12.2.6 Read response formats

The numeric response will have the following formats:

12.2.6.1 Decimal

```
<ID><status><+/-><8 digits><decimal point>
<2 or 3 digits>\terminator
```

```
XD+0030000.00
```

12.2.6.2 Hexadecimal

```
<ID><status><$><8 hex digits>\terminator
```

The **STAT** command is used to request the instantaneous subsystem status. The format of a **STAT** command response is:

```
<ID><status>\terminator
```

See the “Command Status Characters” in the latter half of this section for a list and definition of status characters.

12.2.7 Using hexadecimal numbers

Hexadecimal (hex) numbers can be used for representing a position, velocity, or acceleration in counts. One count is defined as the smallest move size allowed by the axis servo system. (Example: 4 counts in a .5 micron system is equal to 2 microns.) Since various resolutions may be present on different axis boards in a multiple axis system, different count-to-distance relationships may exist in one system. It’s recommended that you stay in decimal entry unless hex entry is mandatory for a given application. Numerical formats are defined via the **ENAINT** command. The default format is decimal.

Velocities and accelerations may also be specified in hex. The default units for velocities and accelerations when specified in decimal are:

Velocities:	Millimeters (k arc-sec) per second
Accelerations:	Millimeters (k arc-sec) per second^2

Low-speed loop velocity and acceleration units are:

Velocities:	µm (arc-sec) per second
Accelerations:	µm (arc-sec) per $second^2$

The default units for velocities and accelerations when specified in hexadecimal are:

Velocities:	Counts per millisecond * 65536
Accelerations:	Counts per $millisecond^2$ * 65536

This hexadecimal representation allows fractional entry of counts per millisecond by observing the following format:

V $108000

This sets the maximum high speed loop velocity to 16.5 counts per millisecond.

12.2.8 Parametric Data Commands

The last type of command is a parametric data command. These commands enter system parameters such as motion parameters, user defined locations, etc. Acknowledge reporting is suppressed on parametric entry commands.

12.2.8.1 RS-232

In RS-232 mode the system will respond with a single character response which includes the ID addressed and its current status. For a description of possible status responses, *see* the “Command Status Characters” in the latter half of this section.

12.2.8.2 IEEE-488

In IEEE-488 mode, the response for parameter modification commands will be the command handshake over the IEEE-488 bus.

12.2.9 Command Error Messages

An error response may occur if an incorrect or unrecognized command is issued to the system. The syntax of an error response is:

```
<ID><E>\terminator
```

The actual cause of the error may be determined by examining the error status registers by querying `<device>ESTAT?` where `<device>` is the specified axis of the system.

12.3 Command Syntax

Commands for the PM500 have the following syntax construction:

12.3.1 For system or motion commands:

```
<ID> <COMMAND>[1 space][parameter]\terminator
```

No space is allowed between the `<ID>` and `<COMMAND>` else a command error will result and the command will be ignored. A `[space]` is allowed between the `<COMMAND>` and the `[parameter]` but is not required. The proper terminator must be appended to all commands, as defined by the **COMOPT** or **ENAINT** commands.

12.3.2 For multiple system or motion commands

```
<ID><COMMAND>[1space][parameter][*delimiter]<ID>
<COMMAND>[1 space][parameter]\terminator
```

Example:

```
XGR 5.0;YGR 10.0;Zgr 15.0
```

<delimiter> character is semicolon (;)

For multiple command strings, the delimiter character (;) must immediately follow the parameter of the first command. No space is allowed before or after the delimiter, else the secondary command will be ignored. The proper terminator must be appended to all commands, as defined by the **COMOPT** or **ENAINT** commands.

12.3.3 For Read/Status and Queries

```
<ID><COMMAND>[1 space][?]\terminator
```

No space is allowed between the `<ID>` and `<COMMAND>` else a command error will result and the command will be ignored. A `[space]` is allowed between the `<COMMAND>` and the `[?]` but is not required. The proper terminator must be appended to all commands, command terminations are defined by the **COMOPT** or **ENAINT** commands.

12.3.4 Syntax Definitions

`ID` refers to a subsystem specifier. The valid range of the specifiers are the following single characters (letters).

5-axis controllers		6-axis controllers	
ID	***Description***	***ID***	***Description***
X	X axis	X	X axis
Y	Y axis	Y	Y axis
Z		Z axis	ZZ axis
A		A axis	A A axis
B	B axis	B	B axis
S*	System	C	C axis
		S	System

** 5 axis controller under SCUM1 mode*

`COMMAND` references the command issued, *see* Command Reference section. This is a variable length field which adheres to IEEE-728 HR1 and HR2 formats, and means the command may be preceded and followed by spaces with no adverse affects.
NOTE: ID specifiers are part of the COMMAND field; therefore no spaces are allowed between the ID and COMMAND fields.

`[space]` is a separator. Spaces may appear in between the command and any of its fields to improve user clarity, but are not required. Spaces are not allowed between the ID and COMMAND fields.

`[delimiter]` refers to a delimiter. The semicolon (`;`) delimiter is used to signify separate concatenated commands to the controller. No space is allowed before or after the command concatenation delimiter.

`[parameter]` refers to the parameter modifier for the specified COMMAND. The range and data type of the parameter is dependent upon the specific COMMAND. Refer to the Range definition for specific commands in the Command Reference section.

`[query]` is used to return a value as specified by the COMMAND. A question mark character (`?`) is used to indicate a query request. The query is optional on read types of commands.

`\terminator` refers to the terminating string or character which signifies the end of a command. All commands should be terminated with a Carriage Return\Linefeed (ASCII: `\r\n`) (Hex: `ØDØA`). When using the IEEE-488 interface, the EOI (End Or Identify) instruction will always be recognized as a terminator regardless of the character sent.

12.3.5 Numeric Formats

12.3.5.1 Decimal Number Entry

Decimal number entry is supported by the following IEEE-728 formats: NR1, NR2, and NR3

Exponent notation is accepted by the controller. Twelve digits are allowed (excluding the decimal point) before the exponent. The exponent is optional and is invoked by an `"E"` or `"e"`. The exponent value may consist of a sign and up to 4 digits. The maximum total acceptable field length is 18 characters.

Example: `119876543210.e-512`

12.3.5.2 Hexadecimal Numbers

Hexadecimal Numbers have the following characteristics:

- Digits 0–9
- Letters A–F, a–f
- Must be pre-pended with a dollar sign ($)
- No more than 8 digits/characters
- 32 bit signed 2's complement format (each digit = 4 bits)

When using hexadecimal numbers in motion commands and returns, the numeric representation is in "resolution counts" (i.e. 4 counts in a 0.1 micron system is 0.4 microns). Hex numbers are also useful when entering a "bitfield". An example of this is the **SENAINT** command.

Example of a hexadecimal command parameter entry: `SENAINT $EF`

12.3.5.3 Units

The PM500 handles units which are compatible with the encoder scale type.

Linear systems use metric scales and return values as follows:

- Position in microns
- Velocities in millimeters per second
- Low-speed-loop velocities in microns per second
- Acceleration and Deceleration in millimeters per second per second

Rotary systems use angular units:

- Position in arc-seconds
- Velocities in degrees per second
- Acceleration and Deceleration in degrees per second per second

12.4 Command Status Characters

PM500 returns can be customized to include a single ASCII status character with each return. The status character field is a fixed length of 1. The format of returns will be:

```
<axis or system ID> <status character> <response value-numeric>
```

The first return field will be 1 character; axis ID (X, Y, Z, etc) or system ‘S’. The acknowledgement field (if enabled) via **ENAINT** will have a field length of 1. The interpretation of the status character must be performed to handle 6 possible string values, the status character field length is fixed at 1. Thus, the user may extract the first value of the returned string to derive the axis, the second value of the string to read acknowledgement and the third value to derive status. The return string can also be offset by 3 when converting the response value to a numeric data type. The table below describes the meaning of the status characters.

Status Character	***Description***	***Comments***
A	Acknowledge	When enabled (ENAINT bit 4 = 1) System will return `<axis>A:` acknowledge for axis specific commands and `SA:` System acknowledge for system commands
B	Busy	Axis specified is still in process of completing last command
D	Done	Axis specified has completed the previous command. The declaration of the DONE response can
E	Error	Follow with **ESTAT**, and **EESTAT** if **ESTAT** bit 22 is high for definition of error.
L	Limit	Axis specified is in limit or previous motion command has violated a limit. Limits can be user defined by the **PSLIM** and **NSLIM** commands. PM500 motion devices also have hard limits at the mechanical extremes of device travel.
M	Motor OFF	Motor is OFF for the axis specified. Motor OFF can occur by the issue of the “**M**” motor off command or open circuit condition of the controllers emergency BNC (emergency stop).

Status characters are invaluable for RS-232 programming as RS-232 lacks the sophisticated status monitoring of GPIB. However, the user will find the status characters useful in simple status checking of GPIB routines.

Enabling Status Characters

The Status and acknowledge characters are enabled via the **COMOPT** (1–7) or **ENAINT** Command. The **COMOPT** command offers a pre-selected menu of various communications protocols. The **ENAINT** command allows custom control of each of the command parameters. See the Command Reference section for more information on the appropriate command.

12.5 GPIB Basics

This section gives a general overview of GPIB programming and definition of terms for users with little or no knowledge of GPIB interfacing. An advanced user can skip this section. This is not intended as a complete description of, or introduction to GPIB interfacing. For more comprehensive information users should refer to texts on GPIB interfacing.

12.5.1 Programming the PM500

Instrument communication programming (RS-232 or GPIB) is fundamentally the execution of 2 basic communication functions:

- Writing strings (commands) across the communications lines
- Reading strings (data) from the instrument

Important to these functions is the synchronization and proper orchestration to external events and events within the instrument under control.

This section focuses on GPIB communication, which possesses a more sophisticated status and signaling method in both hardware and software than RS-232. GPIB allows the user to synchronize instruments and events to program execution.

The PM500 has several methods of communicating system events, status or errors that may have occurred. They are logically divided by the type of event and the way in which status can be polled. The polling method of events can be customized by the user to take many forms and lend themselves to particular programming methods. The users should begin by outlining the requirements of their application, the program method that will best suit that application and configure the PM500 communications settings appropriately.

12.5.2 Service Request

The most common form of GPIB instrument status reporting is known as “Request for Service.” This simply means that a particular instrument requests the attention of the GPIB controller (in most cases your computer). The (SRQ) Service Request is a single hardware line on the GPIB bus. Any instrument on the GPIB bus can request service via this line. A parallel poll or other routine that polls each instrument on the bus is then employed to identify which instrument is “Requesting Service”. The advantage to this is that bus speed is not loaded down by the addition of instruments on the bus.

The PM500 supports the IEEE-488 GPIB (SRQ) Request for Service function. The user can define what events, if any, occurring within the PM500 will trigger a (SRQ) Service Request. This is done via the PM500's SRQ register. This register is software configured and logically partitioned by providing axis ID bits and event bits. Each bit has 2 states: ON or OFF. (Setting the various register bits ON or OFF is known as "masking".)

The following table shows the serial poll of the PM500 (under SCUM1 mode). This register is 8 bits wide.

Serial Poll Register (under SCUM1 mode)

S Poll Bit	Meaning	Comments
0	Axis bitmap	Not maskable
1	Axis bitmap	Not maskable
2	Axis bitmap	Not maskable
3	Axis bitmap	Not maskable
4	MAV	Message available. Read message buffer
5	FAULT*	See bits 0–3 for specified axis in fault condition
6	SRQ	Not maskable
7	DONE/ACK*	Signals done when a specific axis is specified Signal command acknowledge when bits 0–3 are ON: system.

****When the unit requests service with both bits 5 (FAULT) and 7 (ACK) off, completion is indicated for the specified axes.***

Table 1: A graphical representation of the PM500 SRQ register.

NOTE

By default the PM500 SRQ mask is set to 0, turning all bits OFF. No SRQ events will occur under any circumstance. The recommended setting of the Serial Poll register is SRQCTL$FF, all bits on.

The Serial Poll register is configured using the software command **SSRQCTL**. Register bits 0–3 are devoted to identifying the axis or axes for which an event has occured. The user can "mask" (turn bits OFF=0) axes that are not installed in the system or that are not under program control. There are 3 types of event reporting: FAULT, MAV (Message Available), and DONE, which in conjunction with an axis bit 0-4 will describe the condition of a particular axis e.g. if bit 1 and 7 are ON, axis Y is DONE.

In addition to the Serial Poll register, the PM500 has a hierarchical register system for establishing detailed information on status and errors. These are the **STAT**, **ESTAT**, and **EESTAT** registers:

- **STAT—**single character status
- **ESTAT—**device level error register value

When a FAULT condition occurs the user can refer to the **STAT**, **ESTAT**, and **EESTAT** registers respectively to derive the specific nature of an error or status condition.

12.6 SRQ Assertion/Serial Polling methods

Once the Serial Poll register is properly configured, the user can employ 2 different methods of "polling" the register for system status. The method of polling has an effect on the execution of the users program.

12.6.1 Serial Program Execution (Wait for SRQ/RSQ)

In this application the user will wish to trigger an event and wait for that event to complete before executing the next program routine.

For Example: Initiate a motion and wait for motion to complete before reading position.

The technique for this type of SRQ polling is "Wait for SRQ." This is a IEEE-488.2 compliant software function that should be available from the software library of your GPIB card.

12.6.2 Parallel Program Execution (Serial Poll)

In this application the user would wish to trigger an event and proceed with another program routine and be notified when the event is complete or perhaps terminate the secondary event upon the primary events completion.

For example: Initiate a motion, then take readings from another instrument, i.e. Optical power meter, until the motion is complete.

12.7 Serial Polling

The Serial Poll of your GPIB card differs from "Wait for SRQ" in that it allows concurrent program execution during its execution. Serial polling will return the decimal value of the PM500 Serial Poll register and clear the current SRQ assertion. The user must convert the returned decimal value into the binary representation of the Serial Poll and check for the SRQ bit (6) in the ON state or perform a logical AND for 64. Bit 6 (SRQ) is equal to decimal 64 (2^6).

13 Programming the PM500-C6

13.1 Outline of Section 13

System Software of PM500 Controllers

Checking your PM500-C6 System Software

The Difference Between the Modes

Status and Error Reporting System of the PM500

Error Handling

Program Flow charts:

- System Initialization
- General Communication and Handshaking
- Simultaneous Preset Moves
- Pipeline Command Flow

Motion Profile

Motion Speed Loops

Motion Device Dependent commands

- Linear
- Rotary
- Vertical

Essential Motion Commands

13.2 System Software of PM500 Controllers

System software (firmware) differs between the 5 axis and 6 axis PM500 controllers. With PM500 6 axis controllers the serial poll register was updated to support event reporting for greater than 5 axis. This system software also adds additional “system-level” functionality which:

- Facilitates highly synchronized, simultaneous multi-axis motions. By offering new commands only available in the system level communication mode.
- Reduced GPIB overhead in applications with rapid coordinated motion requirement. Via the new SMP Command.
- Implements a MAV (message available) indicator bit in the serial poll register.
- Implements a selectable command acknowledgment capability.
- Provides a consistent and universal interface to 5 and 6 axis versions of the PM500-C.

NOTE

6 axis controllers ONLY operate in the “System Level Communication Mode” (SCUM1). 5 axis controllers with firmware later than 1104 can operate in either mode. The System Communications Mode is enabled via the SCUM command.

13.3 Checking Your System Software Version

If you have an older PM500 System, you can check if your firmware supports the new system level communications mode. Firmware versions later than 1104 support this mode. To query your systems firmware, execute the following query via GPIB:

Send: `SVN?`

Response: `XD+001133456.00`

The version number is the underlined portion of the returned string. Should your system support the new system communications mode, you are urged to use it over the earlier mode so that you maintain forward compatibility with newer PM500 systems.

Should you wish, you can upgrade your PM500 firmware for a nominal fee. This usually encompasses a simple firmware (EPROM) change. However, some earlier PM500 systems, pre 1990 may also require a hardware upgrade. Contact Newport with your systems serial number for more information about upgrading your system firmware.

13.4 The Difference Between the Modes

There are 3 significant differences between the system level communications mode and the earlier PM500 communications mode:

1. A new serial poll register format is available - via the SENAINT command to provide event reporting for up to 6 axes.

Old 5 axis serial Poll register

X	Y	Z	A	B	FAULT	SRQ	DONE	Definition
0	1	2	3	4	5	6	7	Bit No.
1	2	4	8	16	32	64	128	Dec. Val

New 6 axis serial Poll register

Bit map	Bit map	Bit map	Bit map	MAV	FAULT	SRQ	ACK	Definition
0	1	2	3	4	5	6	7	Bit No.
1	2	4	8	16	32	64	128	Dec. Val

The new serial poll format bitmaps bits 0 - 3 which allows the specification of up to 6 axes plus “S” the system specifier. No bits ON across 0 - 3 indicate the X axis.

2. Preset motion commands are available under the new mode to allow execution of motion for multiple axes with one command. These are:

- SMP: Simultaneous Move Preset (Absolute)
- SMRP: Simultaneous Move Preset (Relative)

3. The system now has ID specifier “S”. The system ID specifier must pre-pend system level commands i.e. those commands that do not directly address an axis. <u>This only pertains to operation in **SCUM1** mode:</u>

System Level commands under the modes:*

Mode	**SCUM 0**	**SCUM1**	**Comments**
Command	RSTART	SRSTART	Restart system
	DEFEE	SDEFEE	Default system
	SRQCTL	SSRQCTL	SRQ masking
	ENAINT	SENAINT	Communications configuration

** This is not a complete list: refer to the Command Reference section.*

13.5 Status and Error Report Flowchart for the PM500

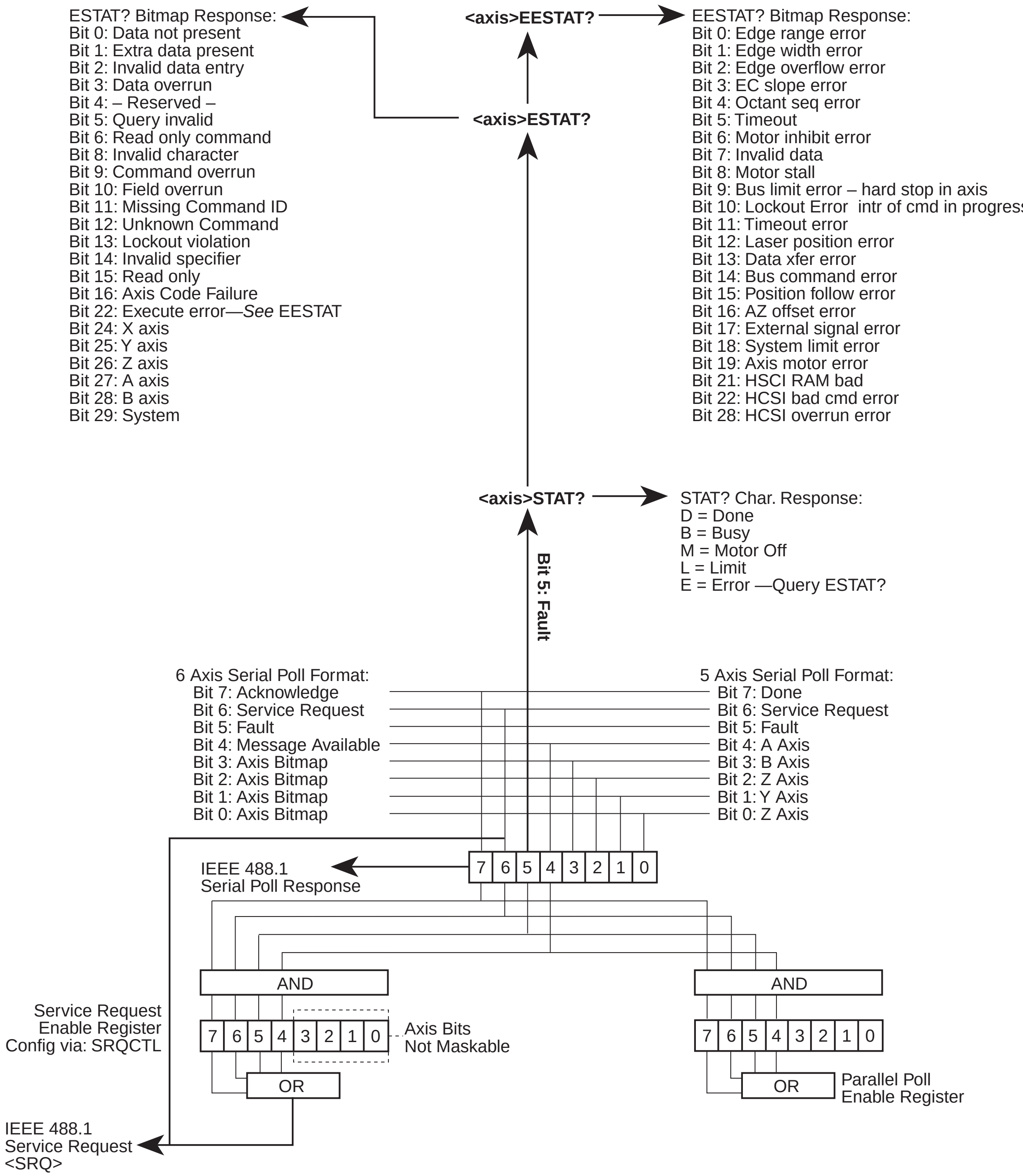


Figure 32: Flowchart: Status and error reporting registers of the PM 500.

NOTE

Note the serial poll format differences between SCUM0 and SCUM1 modes.6=0

13.6 Error Handling

Good Programming practice demands that immediately upon receipt of an error indicator the user investigate the status of the system. The PM500 indicates errors in two ways:

- Fault Bit (5) of the Serial Poll Register.
- "E(error) or L(limit)" ASCII Status character with return strings

On encountering an error the user should do the following:

1. Query the Error Status Register via the **ESTAT?** Command for the indicated axis - this includes the system should the serial poll indicate a system level error.
2. If Bit 22 of the **ESTAT** register is on, query the Execution Error Register via the **EESTAT?** Command for the indicated axis.
3. Take corrective action to clear the error condition.

The system latches error states until queried. The system does not stack error states. The most recent error state will overwrite older error information.

System responses to certain error conditions can be controlled via the **ENAINT** command i.e. Bit 0 of the **ENAINT** register controls whether the system will halt all axes on an single axis limit or only the effected axis.
Lab View examples can be downloaded from Newport Website-www.newport.com

13.6.1 Initializing the PM500 (SCUM1 or 6axis controllers) Program flowchart

This initialization procedure should be done at the start of all your PM500 programs to ensure reliable and repeatable GPIB performance. The procedure below initializes the PM500 SRQ register enabling all axes and all events to trigger SRQ's. The **SENAINT** register configures the communication parameters. The serial poll is repetitively done until bit 16 is clear, then a GPIB read is done until the return string is empty to ensure the message buffer in the PM500 is clear.

This will bring the PM500 controller to a know state, ensuring no previous SRQ's or messages are in the buffers, which can cause program synchronization problems.

NOTE

This applies to GPIB communication only—for RS-232, see the RS-232 Section.

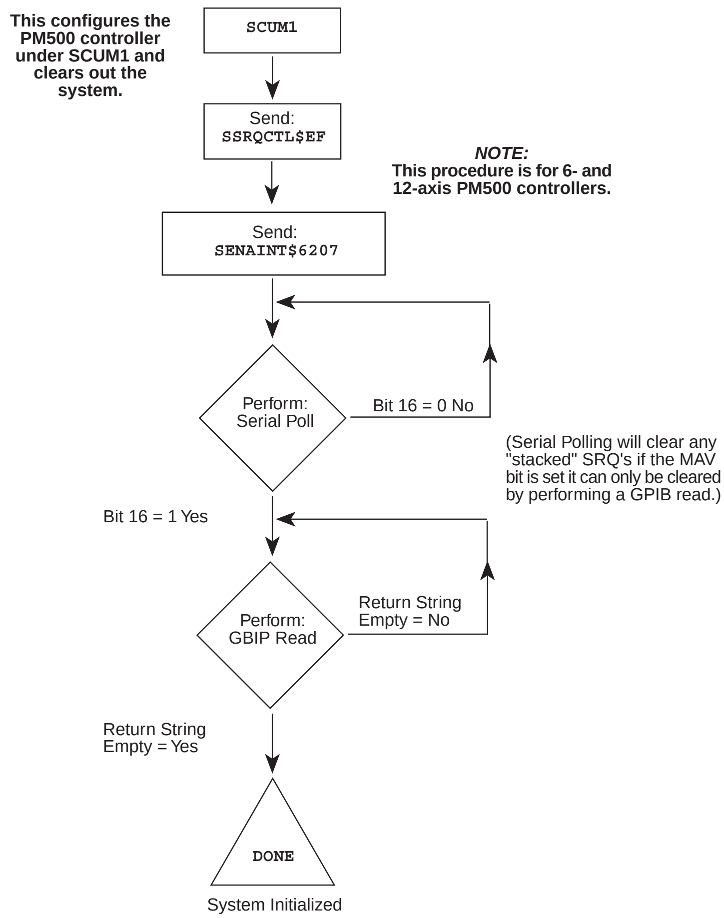


Figure 33: Flowchart: An initialization procedure for the PM 500.

13.6.2 General communication and 'handshaking' with PM500 Program Flowchart (under SCUM1) operation mode

We suggest the following process be followed for all programs. This process handles both types of commands—queries and motion—and their proper completion signaling and error handling. The user can use the "Wait for SRQ" function or Serial Poll interchangeably (Serial Poll offers more information than simple completion (SRQ) signaling). Error handling is also detailed.

NOTE

The initialization procedure (Section 12.2) should precede this process.

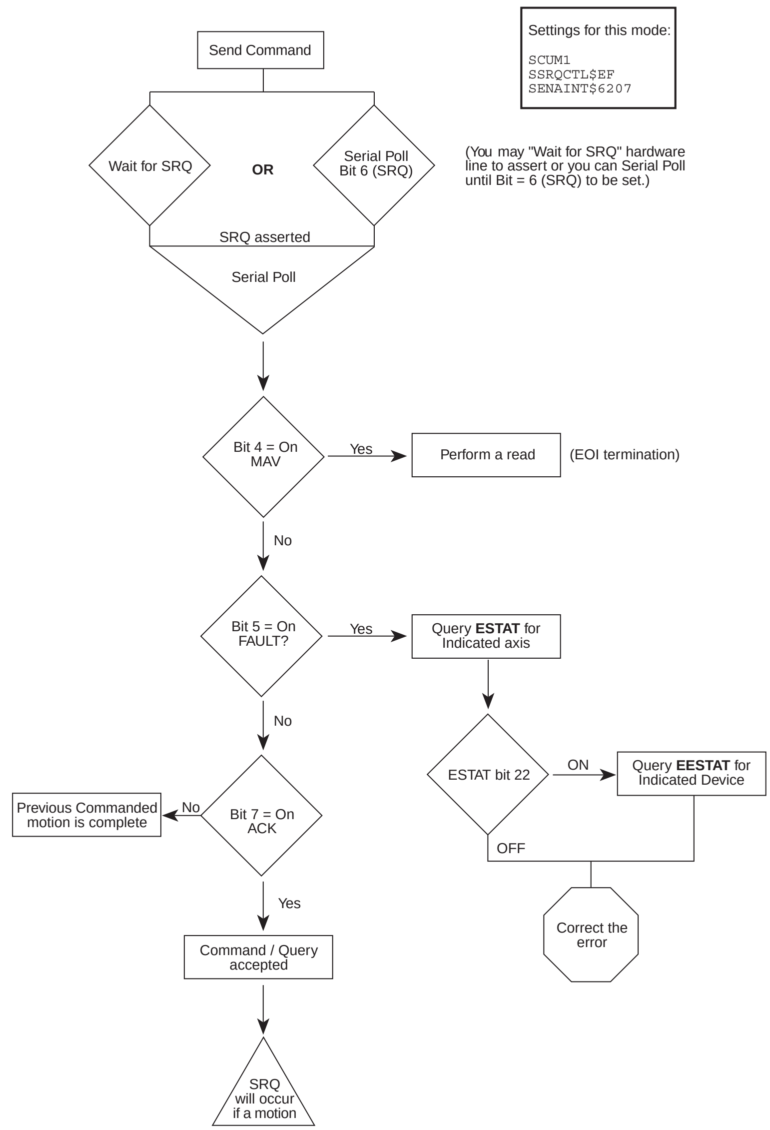


Figure 34: *A general handshaking and communication method.*

13.6.3 Simultaneous Preset Motions Program Flowchart

The following is the recommended program flow for the setup and execution of the simultaneous preset motion function. The **SMPL** register must first be masked to include the desired axes to move. If you wish to specify a different acceleration, deceleration, and velocity for the preset move, it can be done by first enabling that function via the **SMPL** register then setting the parameters using the **MPACC**, **MPDEC**, and **PMVEL** commands.

NOTE

Only for use with PM500-C6 controllers operating in SCUM1.

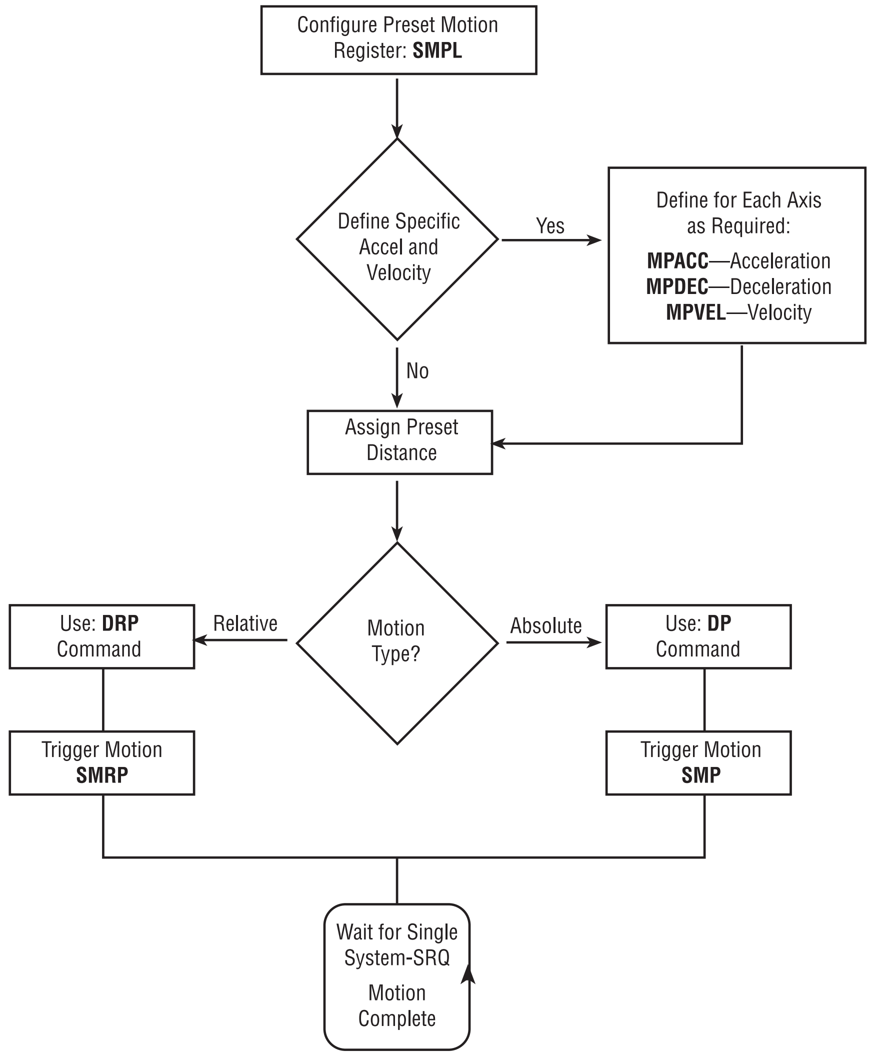


Figure 35: PM500's simultaneous preset motion.

NOTE

Specific acceleration, deceleration, and velocity can be specified for the preset move. The move definition and trigger command used depends upon the move type to be initiated.

There are two sets of preset move definition commands and trigger commands which depend upon the type of move (relative or absolute) to be initiated. The system will assert only one System SRQ upon completion of the simultaneous move, regardless of the number of axes involved.

13.7 The Motion Profile And Related Motion Commands

The PM500 has a unique self-adjusting motion profiling and servo system. The command set is intuitive and easily related to the classical "trapezoidal move profile." Take time to read through this section, as it will provide valuable insight into the significance and relationships of the PM500 motion command set.

The following diagram outlines the basic PM500 motion commands and their relation to a typical motion profile.

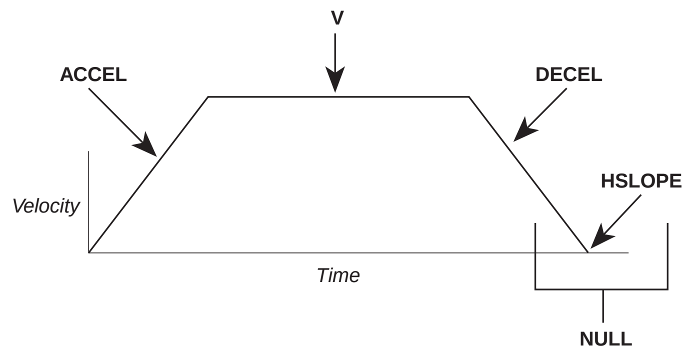


Figure 36: Trapezoidal move profile with corresponding commands.

Command	***Description***
ACCEL	Rate of acceleration until velocity is achieved. *(Not actual value set dependent upon load.)*
V	Velocity of move. *(The desired velocity may not be achieved depending upon the size of the move and the programmed rate of acceleration and deceleration.)*
DECEL	Rate of deceleration to stop at desired position. *(Not actual value set dependent upon load.)*
HSLOPE	Position holding/servoing stiffness.
NULL	Window which defines when the axis signals motion complete.

13.8 Motion Speed Loops

The PM500 has three motion step/speed ranges:

- Ultra-low speed loop
- Low speed loop
- High speed loop

The setting of the **LSIZE** (low size) parameter command allows the controller to automatically scale the speed of the move range to execute a motion ***dependent on the size of the motion***. This allows the user to execute motions of various sizes without specifying velocity and accelerations.

The function of these three speed ranges is analogous to driving a car. To travel a few meters one would not attempt to cover that distance at maximum acceleration, but rather at a lower speed to provide better accuracy with no overshoot.

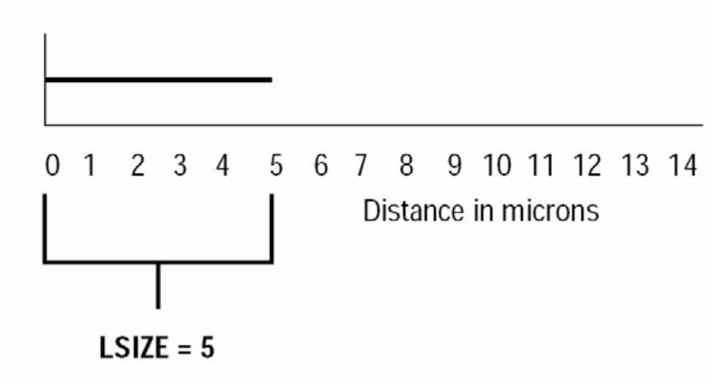


Figure 37: Setting the LSIZE.

Using the **LSIZE** setting in the figure above, any motion under 5-microns will be executed in the low speed loop. Low speed motions parameters are controlled by an identical set of motion commands as mentioned above but are preceded by an **L**, i.e., **LACCEL**, **LDECEL**, **LV**, etc.

NOTE

Do NOT set LSIZE less than system resolution or zero. Since position holding is performed within the low speed loop, doing so will disable the axis ability to hold and repeat position.

13.9 Motion Device Dependent commands & PM500 devices

13.9.1 Linear motion device commands

The following illustration demonstrates various commands that relate to mechanical properties of PM500 Linear stages.

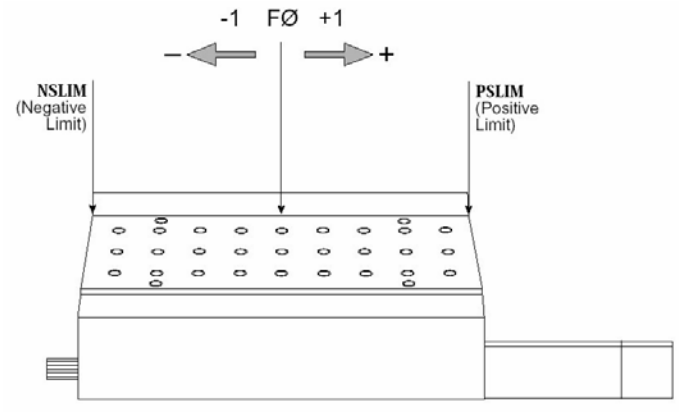


Figure 38: Linear motion device commands forPM500 Linear Stages.

All motions are centered about the stage fiducial point (center of travel). You must specify motions in (+) positive or (–) negative for motions relative to this point. The default direction of motion (+direction) is towards the motor.

Units

Position: Position commands should be in microns. All returns from position queries are in microns.

Velocity: In mm/sec

Acceleration: In mm/sec^2

Limits: PM500 Linear stages possess settable software limits at each end of travel, specified as negative and positive values. Note that the limits are active even when the motor has been commanded off (**M**). If the limit is encountered due to manual motion, the motor will turn on and servo to position to prevent violation of the limit.

This will change if the directional coordinate system is reversed using the **SIGN** command.

Directional Defaults: (+) Positive moves towards motor (towards knob on PM500-1)

(–) Negative moves away from motor

The directional coordinate system can be changed using the **SIGN** command and stored as power-on default.

13.9.2 Rotary motion device commands

The following illustration demonstrates various commands that relate to mechanical properties of PM500360-A Rotary stages.

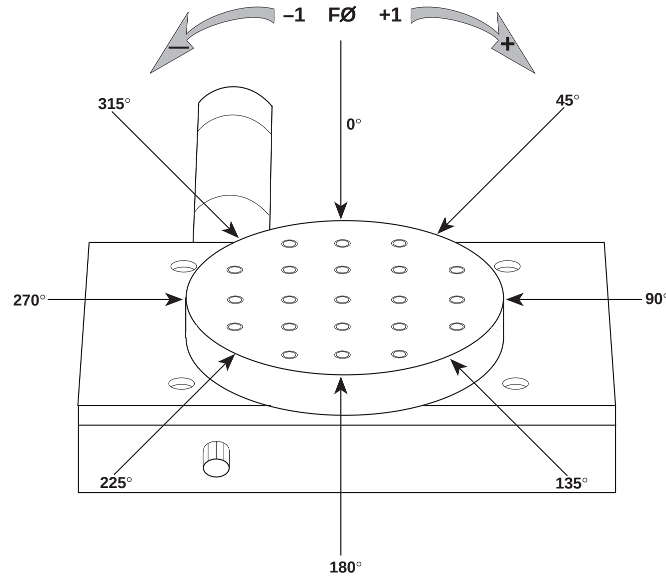


Figure 39: PM500-360/A Rotary stages.

Units

Position: Position commands should be in arc-seconds. All returns from position queries are in arc-seconds.

Velocity: In K arc-sec/sec

Acceleration: In K arc-sec/sec^2

Fiducials: The PM500-Rotary stages have 8 fiducial points located every 45º. You can define any of the 8 fiducials as the default for the Fiducial "**F**" Seek command via the **FNUM** command. *See* the Command Reference Section for details. A sign (+/–) defines the direction the fiducial seek will travel.

Home: By default the direction for travel for the **H** (Home) command is clockwise. This will change if the rotary directional coordinate system is reversed using the **SIGN** command.

Limits: The rotary stages have no software limits.

13.9.3 Vertical motion device commands

The following illustration demonstrates various commands that relate to mechanical properties of PM500-1V Vertical linear stage. The default direction of motion (+ direction) is up.

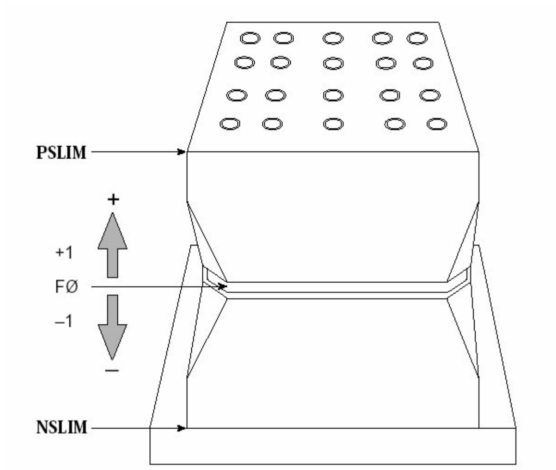


Figure 40: PM500-V1 Vertical stages.

Units

Position: Position commands should be in microns. All returns from position queries are in microns.

Velocity: In mm/sec

Acceleration: In mm/sec^2

Limits: PM500 Linear stages possess settable software limit at each end of travel, specified as negative and positive values. Note that the limits are active even when the motor has been commanded OFF (**M**). If a limit is encountered due to manual motion the motor will turn on and servo to position to prevent violation of the limit.

Fiducial: All motions are centered about the stage fiducial point (center of travel). You must specify motions in (+) positive or (–) negative for motions relative to this point.

Directional Defaults: (+) Positive moves the platform up. (motor side is knob on PM500-1V)

(–) Negative moves down.

The directional coordinate system can be changed using the **SIGN** command and stored as power-on default.

13.10 Essential Motion Commands

The following is a list of the most often used commands of the PM500 command set:

Initialization	***Commands***	***Page***
ENAINT	Communications Configure	135
SCUM	System Communications Mode Enable	246
SRQCTL	Service Request Assertion Masking	254

Motion	***Commands***	***Page***
G	Go Absolute	147
GR	Go Relative	150
MR	Move Relative	177
S	Scan axis at velocity	237
ULS	Ultra-low speed scan	262

Motion Profile	***Commands***	***Page***
ACCEL	Acceleration – High Speed Loop	105
DECEL	Deceleration – High Speed Loop	123
LV	Velocity – Low Speed Loop	167
LACCEL	Acceleration – Low Speed Loop	159
LDECEL	Deceleration – Low Speed Loop	161
LSIZE	Defines Low Speed Loop Threshold	164
V	Velocity – High Speed Loop	265

Homing and Limits	***Commands***	***Page***
F	Fiducial Seek	142
H	Home axis (GO TO 0.0)	153
NSLIM	Negative Soft Limit	179
PSLIM	Positive Soft Limit	233

Miscellaneous	***Commands***	***Page***
SDEFEE	Default System to Factory Settings	126
SRSTART	Reboot System	256
SSAVEAX	Saves User Settings	238

14 Command Index (by Type)

Motion commands

Absolute motion commands Page

Essential motion commands

Please refer to the corresponding pages for complete syntax information

15 Command Index (Alphabetical)

In a PDF format you may click on a line to be automatically taken to the corresponding page.

This page is intentionally left blank

16 Command Reference

ACCEL - Define Acceleration for High-Speed Loop

Syntax: [n]ACCEL <acceleration>

Parameters: [n] defines the axis identifier, n = X, Y, Z, A, B, C whose High speed Loop acceleration is to be set. If the axis identifier [n] is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier [n] and the command.

The acceleration value <acceleration> must be a string type. No quotation marks should frame the acceleration value, otherwise a command error will result.

Function: This command defines the acceleration for the specified axis in mm/sec^2 or k arc-sec/sec^2 for the High Speed Loop. This parameter is used for controlling the velocity ramp up profile of a move. Adjusting this parameter allows very soft or harsh moves to be programmed. Actual acceleration is somewhat dependent upon system load; no absolute calibration is implied in this parameter. This parameter has no effect on motions executed in the Low Speed Loop.

Returns: NONE

Range: Device dependent

Examples:

Send:	XACCEL 500
Send:	XACCEL?
Receive:	XD+000500.0

The above example sets the acceleration for the X axis High Speed Loop at 500mm/sec^2.

User set values can be stored in non-volatile memory using the [n]SAVEAX command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. For large or heavy loads the **ACCEL** should be decreased to avoid long settling times or vibration caused by sudden acceleration. *See* the System Settings section in this manual for factory settings.

Related commands: **ACCEL?**, **DECEL**, **V**, **LACCEL**, **SAVEAX**

ACCEL? - Acceleration for High-Speed Loop Query

Syntax: [n]ACCEL?

Parameters: [n] defines the axis identifier, n = X, Y, Z, A, B, C whose High speed Loop acceleration is to be queried. If the axis identifier [n] is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier [n] and the command.

Function: This query returns the acceleration for the specified axis [n] in mm/sec^2 or k arc-sec/sec^2 for the High Speed Loop. Actual acceleration is somewhat dependent upon system load; no absolute calibration is implied in this parameter. This parameter has no effect on motions executed in the Low Speed Loop.

Returns: <[Axis]><*[Status]><[n]><+/->
<acceleration>
** If status character enabled*

Examples:
Send: XACCEL?
Receive: XD+000500.0

In the above example the acceleration for the X axis High Speed Loop was set at 500mm/s^2.

Recommendations: Actual acceleration during motion cannot be queried on-the-fly; the return would be the set **ACCEL** value.

Related commands: **ACCEL**, **DECEL**, **V**, **LACCEL**, **SAVEAX**

AZ - Auto Zero Self Calibration

Syntax: `[n]AZ`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose axis is to be self calibrated. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command commences an automatic calibration process, which takes approximately 45 seconds to complete (a unique capability of the PM500 controller). When the AutoZero command is issued to an axis, the controller halts all motion and enters a monitoring routine which rebalances ("zeros") the analog servo-loop circuitry if necessary.

NOTE

The motion devices should not be disturbed during the AutoZero process, which takes approximately 45 seconds.

The PM500 controller's axis cards incorporate advanced, proprietary analog and digital servo-loops. The controller has an automatic capability to "self-adjust" to compensate for minor component variations due to aging, environmental differences versus factory conditions, or vibration and handling. It is recommended that each axis be AutoZeroed prior to each use to achieved maximum performance.

Returns: `NONE`

Examples:

Send:	XAZ
Send:	XR?
Receive:	XB+00000.0

In the above example an Auto Zero is commenced on the X axis, then read via the **R** command. The return includes the axis status "B" which indicates the axis is Busy. When the status character changes to a "D" the **AZ** is done.

The results of the AutoZero process should be saved in non-volatile memory using the `[n]SAVEAX` command.

User set SAVEAX values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The AutoZero command should be issued shortly after system warm-up prior to use. The **SAVEAX** command should be issued to save the AutoZero parameters.

Related commands: **AZVAL?**

C - Clear Position

Syntax: `[n]C`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose position is to be cleared. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command clears the position and defines it as 0.0. When the command is issued it stops any move in progress, clears the position register generating a new home and servos to this position.

Returns: `NONE`

Examples:

Send:	`XR?`
Receive:	`XD+002500.0`
Send:	`XC`
Send:	`XR?`
Receive:	`XD+000000.0`

In the above example the current position of the X axis is read, then cleared using the **C** command.

Recommendations: The home created by the **C** command is used as the origin for the **H** “Home” command.

Related commands: **H**

S/COMOPT – Communication Setup

Syntax: (S)*COMOPT <option>

Parameters: <option> must be an interger within the range as defined by the options listed. The <option> will be rounded to the nearest interger or ignored if out of range. A space is allowable between the command and the parameter but is not required. This command has no axis identifier [n].
**"S"system specifier required under SCUM1 mode.*

The <option>, must be a string type. No quotation marks should frame the option value, else a command error will result.

Function: This command configures the RS-232/IEEE interface communications.

This parameter is used for selecting the response/acknowledgment modes. Adjusting this parameter allows you to enable or disable completion /termination or tailor controller responses to include/exclude status information.

Returns: NONE

Defined S/COMOPT
Options: Indicates enabled - x

S/COMOPT	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*10**
RS232	x	x	x	x	x	x			
GPIB IEEE							x	x	x
ECHO		x		x					
SIGN-ON MESSAGE				x		x			
ACKNOWLEDGE. REPORTING				x		x		x	x
STATUS CHAR. RESPONSE	x	x	x	x	x	x	x	x	x
COMMAND COMPLETION			x	x	x	x			
ACK. REPORT VIA HANDSHAKE							x	x	x
CR LF TERMINATION	x	x	x	x	x	x	x	x	x
EOI ON LAST CHAR.							x	x	x

****10 available under SCUM1 mode.***

Examples:

Send: `COMOPT 1`

The above example sets communication for RS232, echo disabled, sign on message disabled, acknowledge reporting enabled, command completion/termination disabled, **CR LF** termination on.

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations:

It is recommended that status reporting and status character response be enabled. This allows better “visibility” of the system status during communication interaction. The **ENAINT** command has greater flexibility in the configuration of system communications.

COMOPT Equivalents to ENAINT:

COMOPT 0 fi	ENAINT$85F
COMOPT 4 fi	ENAINT$847
COMOPT 6 fi	ENAINT$607
SCOMPT 10	SENAINT$6217

Related commands: **ENAINT**

CONFIG - Report Axis Configuration Query

Syntax: [n]CONFIG?

Parameters: [n] defines the axis identifier, n = X, Y, Z, A, B, C whose configuration is to be queried. If the axis identifier [n] is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier [n] and the command.

Function: This command queries the axis configuration, which returns a configuration code.

Returns: Axis configuration code

Examples:

Send: XCONFIG?
Receive: XD+000400.0

The above example queries the X axis configuration. The return is an axis configuration code.

Recommendations: This command is useful when axis configuration is unknown. Contact Newport for the type of interface option (if present) that the code represents.

Related commands: **VN? SVN?**

CSCUR - Define Motor Current Limit

Syntax: `[n]CSUR <-limit>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose current limit value is to be modified. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command. If the limit is not prefaced with the (-) negative sign the command will not be executed and a command error will be posted.

The `<limit>` must be a string type and a negative value. No quotation marks should frame the limit value, otherwise a command error will result.

Function: This command defines the current limiting value overload protection for the specified axis. **CSCUR** is entered as a negative number.

The PM500 controller has the ability to sense motor overload and maintain proper system control. If an overload condition is sensed, the controller will de-energize the servo drive and post an error to you in accordance with the reporting protocol selected.

Overload conditions are determined by a real-time algorithm which uses motor current level, position, and time. A current sense error will be detected if the drive current exceeds a programmed level (**CSCUR**) for a programmed time (**CSTIME**) and moves less than a programmed distance (**CSMOVE**).

CAUTION

This parameter is used for applications where motion can cause objects to come in physical contact and the amount of force, i.e. motor current must be limited to prevent damage.

Range:	-511 to 0 Each bit equals 0.02 amperes of motor current.
Defaults:	Linear and rotary translators: -150 (3 amps) PM500-1 Mini-Stage: PM500-1A Actuators: -50 (1 amps)
Returns:	`NONE`
Examples:	Send: `XCSUR -100` The above example sets the X axis motor current to 2 amps. User set values can be stored in non-volatile memory using the `[n]SAVECS` command. User set **SAVECS** values are restored after system power-up. *See* **SAVECS** command for details.
Recommendations:	It is not recommended that this parameter be adjusted. The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. Any modification to this command should be in conjunction with changes to the related commands listed below. *See* the System Settings section in this manual for factory settings.
Related commands:	**SAVECS**, **CSMOVE**, **CSTIME**, **DEFCS**, **INITCS**

CSCUR? - Define Motor Current Limit Query

Syntax: `[n]CSUR?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose current limit value is to be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the current limiting value overload protection for the specified axis. **CSCUR** value returned will be a negative number.

The PM500 controller has the ability to sense motor overload and maintain proper system control. If an overload condition is sensed, the controller will de-energize the servo drive and post an error to you in accordance with the reporting protocol selected.

Overload conditions are determined by a real-time algorithm which uses motor current level, position, and time. A current sense error will be detected if the drive current exceeds a programmed level (**CSUR**) for a programmed time (**CSTIME**) and moves less than a programmed distance (**CSMOVE**).

Range: -511 to 0 Each bit equals 0.02 amperes of motor current.

Defaults: Linear and rotary translators: -150 (3 amps)
PM500-1 Mini-Stage: -50 (1amps)
PM500-1A Actuators: -50 (1 amps)

Returns: Current **CSCUR** value

Examples:

`Send:`	`XCSUR?`
Receive	`XD-000100.00`

The above example reads the X axis motor current which was set to 2 amps.

Recommendations:

It is not recommended that this parameter be adjusted. The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. Any modification to this command should be in conjunction with changes to the related commands listed below.

See the System Settings section in this manual for factory settings.

Related commands:

SAVECS, **CSMOVE**, **CSTIME**, **DEFCS**, **INITCS**

CSMOVE - Define Motor Current Limit Sense Move Value

Syntax: `[n]CSMOVE <distance>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose motor limit move value is to be defined. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

The `<distance>,` must be a string type.

Function: This command defines the minimum distance the axis must move to avoid a current sense error. **CSMOVE** is entered in position units µm or arc-sec as a positive number.

The PM500 controller has the ability to sense motor overload and maintain proper system control. If an overload condition is sensed, the controller will de-energize the servo drive and post an error to you in accordance with the reporting protocol selected.

Overload conditions are determined by a real-time algorithm which uses motor current level, position, and time. A current sense error will be detected if the drive current exceeds a programmed level (**CSUR**) for a programmed time (**CSTIME**) and moves less than a programmed distance (**CSMOVE**).

Returns: `NONE`

Examples:

Send:	`XCSMOVE 5.0`
Send:	`XCSMOVE?`
Receive:	`XD+000005.0`

In the above example the current sense move distance is set to 5µms.

User set values can be stored in non-volatile memory using the `[n]SAVECS` command. User set **SAVECS** values are restored after system power-up. *See* **SAVECS** command for details.

Recommendations:	It is not recommended that this parameter be adjusted. The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. Any modification to this command should be in conjunction with changes to the related commands listed below. *See* the System Settings section in this manual for factory settings.
Related commands:	**CSCUR**, **CSTIME**, **SAVECS**, **DEFCS**, **INITCS**

***CSTIME** - Define Motor Current Limit Sense Time Value*

Syntax: `[n]CSTIME <time>`

Parameters: [n] defines the axis identifier, `n = X, Y, Z, A, B, C` whose Motor Current Limit Sense time is to be modified. If the axis identifier [n] is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier [n] and the command.

The `<time>,` must be a string type. No quotation marks should frame the value, otherwise a command error will result.

Function: This command sets the current sense time period for overload protection for the axis specified. **CSTIME** is entered in milliseconds.

The PM500 controller has the ability to sense motor overload and maintain proper system control. If an overload condition is sensed, the controller will de-energize the servo drive and post an error in accordance with the reporting protocol selected.

Overload conditions are determined by a real-time algorithm which uses motor current level, position, and time. A current sense error will be detected if the drive current exceeds a programmed level (**CSUR**) for a programmed time (**CSTIME**) and moves less than a programmed distance (**CSMOVE**).

Range: 0 to 32767 in milliseconds

Default: System Dependent

Returns: `NONE`

Examples:

Send:	`XCSTIME 200`
Send:	`XCSTIME?`
Receive:	`XD+000200.0`

In the above example **CSTIME** for the X axis is set to 200 milliseconds.

User set values can be stored in non-volatile memory using the `[n]SAVECS` command. User set **SAVECS** values are restored after system power-up. *See* **SAVECS** command for details.

Recommendations: It is not recommended that this parameter be adjusted. The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. Any modification to this command should be in conjunction with changes to the related commands listed below. *See* the System Settings section in this manual for factory settings.

Related commands: **CSCUR**, **CSMOVE**, **SAVECS**, **DEFCS**, **INITCS**

CSTIME? - Motor Current Limit Sense Time Query

Syntax: `[n]CSTIME?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Motor Current Limit Sense time is to be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the current sense time period of overload protection for the axis specified. **CSTIME** return is in milliseconds.

The PM500 controller has the ability to sense motor overload and maintain proper system control. If an overload condition is sensed, the controller will de-energize the servo drive and post an error to you in accordance with the reporting protocol selected.

Overload conditions are determined by a real-time algorithm which uses motor current level, position, and time. A current sense error will be detected if the drive current exceeds a programmed level (**CSUR**) for a programmed time (**CSTIME**) and moves less than a programmed distance (**CSMOVE**).

Returns: **CSTIME** value in milliseconds

Examples:

Send: `XCSTIME?`
Receive: `XD+00200.0`

In the above example **CSTIME** for the X axis is set to 200 milliseconds.

Recommendations: It is not recommended that this parameter be adjusted. The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. Any modification to this command should be in conjunction with changes to the related commands listed below. *See* the System Settings section in this manual for factory settings.

Related commands: **CSCUR**, **CSMOVE**, **SAVECS**, **DEFCS**, **INITCS**

DECEL - Define Deceleration Value for Large Moves

Syntax: `[n]DECEL <value>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose deceleration is to be modified. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

The `<value>`, must be a string type. No quotation marks should frame the acceleration value, otherwise a command error will result.

Function: This command defines the High Speed Loop deceleration value for the specified axis in mm/sec^2 or k arc-sec/sec^2. This parameter is used for controlling the ramp-down profile of motion. Adjusting this parameter allows very soft or harsh moves to be programmed. The actual deceleration is somewhat dependent upon system load; no absolute calibration is implied in this parameter.

Returns: NONE

Range: Device dependent

Examples:

Send: `XDECEL 300`
Send: `XDECEL?`
Receive: `XD+0000300.0`

In the above example the deceleration for the X axis is set to 300mm/sec^2.

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. For large or heavy loads the **DECEL** should be decreased to avoid long settling times or vibration caused by sudden deceleration. *See* the System Settings section in this manual for factory settings.

Related commands: **ACCEL, DECEL?, V, SAVEAX**

DECEL? - Deceleration Value for Large Moves Query

Syntax: `[n]DECEL?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose deceleration is to be modified. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the High Speed Loop deceleration value for the specified axis in mm/sec^2 or k arc-sec/sec^2. The actual deceleration is somewhat dependent upon system load; no absolute calibration is implied in this parameter.

Returns: Current set **DECEL** value in mm/sec^2 or k arc-sec/sec^2

Range: Device dependent

Examples:

Send:	`XDECEL?`
Receive:	`XD+0000300.0`

In the above example the deceleration for the X axis was set to 300mm/sec^2.

Recommendations: *See* the System Settings section in this manual for factory settings of this value.

Related commands: **ACCEL**, **DECEL**, **V**, **SAVEAX**

DEFCS - Default Current Sense Parameters

Syntax: `[n]DEFCS <option>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose current sense parameters are to be defaulted. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command restores the Current Sense parameters to your settings or factory default depending upon which option is selected.

Range: 0 = restore to User settings from last **SAVECS**
1 = restore to factory system default

Returns: `NONE`

Examples: Send: `XDEFCS 1`

In the above example the X axis current sense parameters are restored to factory default.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **INITCS, SAVECS, CSTIME, CSMOVE, CSCUR**

SDEFEE - System Reset to Factory Defaults

Syntax: `(S)*DEFEE`

Parameters: This is a system level command. This command under SCUM1 operation will require the system ID specifier. No axis identifier or parameter is allowed with this command else the command will not be executed and a command error will be posted.
*"S" system specifier required under SCUM1 mode.

WARNING

All user settings will be cleared and the system will be restored to original factory parameters. If you do not wish to reset axis cards parameters use the (S)DEFLT command.

Function: This command resets the entire system to factory defaults as stored in system firmware. All user set values will be cleared and restored to original factory settings. This command should be followed by the (S)RSTART command.

Range: 0 = restore to User settings from last **SAVECS**
1 = restore to factory system default

Returns: `NONE`

Examples:

Send: `DEFEE` (under SCUM 0)

Send: `SDEFEE` (under SCUM 1)

In this example the system is restored to original factory settings and restarted, under both modes of operation.

Recommendations: The system should be restored to original factory settings if parameter adjustments you made have caused erratic system behavior or poor performance. Restoring the system to original settings via (S)DEFEE should be attempted before any servo tuning or other adjustments are made to the system. Refer to the Factory Setting section in this manual for the original factory setting parameters.

Related commands: **SDEFLT**

DEFLM - Default Axis Soft Limits

Syntax: `[n]DEFLM <option>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose axis soft limits is to be defaulted. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command restores the soft limits for the specified axis to user settings as saved via **SAVELM** or factory default settings depending upon the parameter specified.

Parameters: 0 = restore to users settings stored via **SAVELM**
1 = restore to factory default settings.

Returns: `NONE`

Examples: Send: `YDEFLM`

In the above example the Y axis soft limit is returned to original factory settings.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **SAVELM**, **NSLIM**, **PSLIM**, **RPOS**

SDEFOM - Default PM500-K6 Settings

Syntax: `(S)*DEFOM <option>`

Parameters: This is a system level command. This command under SCUM1 operation will require the system ID specifier. No axis identifier or parameter is allowed with this command otherwise the command will not be executed and a command error will be posted.
**"S" system specifier required under SCUM1 mode.*

Function: This command defaults the PM500-K6 to user stored or factory settings dependent upon the option selected.

Options: 0 = restore to user settings from last SAVEOM
1 = restore to factory settings

Returns: `NONE`

Examples: Send: `DEFOM 0`

The above example defaults the PM500-K6 to user stored settings.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **INITOM**, **SAVEOM**

DP - Define Preset Position

Syntax: `[n]DP <position>`

Parameters: [n] defines the axis identifier, `n = X, Y, Z, A, B, C` whose position is to be predefined. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command defines the preset position for the specified axis in µms or arc-sec. The value of **DP** is stored and used as the move size for the **GP** command.

Range: Any valid absolute position value + or - in µm or arc sec.

Returns: `NONE`

Examples:

Send:	`XDP 12500.0`
Receive:	`XGP`

In the above example the predefined move for the X axis is 12500.0 microns. The issuing of the **GP** (Go Preset) sends the device to the preset position of 12500.0.

Related commands: **C**, **GP**, **SMP**, **SMPL**

DP? - Preset Position Query

Syntax: [n]DP?

Parameters: [n] defines the axis identifier, n = X, Y, Z, A, B, C whose predefined position is to be queried. If the axis identifier [n] is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier [n] and the command.

Function: This command queries the preset position for the specified axis in µms or arc-sec. This is the value that is used as the move size for the **GP** command.

Returns: Currently stored predefined position in µm or arc-sec.

Examples:

Send:	XDP?
Receive:	XD+0012500.0

In the above example the predefined move for the X axis is 12500.0 microns. The issuing of the **GP** (Go Preset) sends the device to the preset position of 12500.0.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **C**, **GP**, **SMP**

DRP - Define Relative Preset Position

Syntax: `[n]DRP <position>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose position is to be predefined. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command defines the relative preset position for the specified axis in µms or arc-sec. The value of **DRP** is stored and used as the move size for the **GRP** and **SGRP** commands.

Range: Any valid position value + or - in µm or arc-sec

Default: 0

Returns: NONE

Examples:

Send: `XDRP 500.0`
Receive: `XGRP`

In the above example the predefined relative move for the X axis is set at 500.0 microns. The issuing of the **GRP** (Go Relative Preset) sends the device to the relative preset position of 500.0 µm to the current position of the device.

Recommendations: The **DRP** command provides faster response and better synchronization between axes than the standard single axis Direct Execute Relative Motion command.

Related commands: **GRP**, **SMRP**, **SMPL**

DRP? - Relative Preset Position Query

Syntax: `[n]DRP?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose position is to be predefined. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the relative preset position currently set for the specified axis in µm or arc-sec. This is the value that is used as the move size for the **GRP** and **SGRP** commands.

Default: `0`

Returns: Currently stored relative preset position

Examples:

Send: `XDRP?`
Receive: `XD+000500.0`

In the above example the predefined relative move for the X axis is set at 500.0 microns. The issuing of the **GRP** (Go Relative Preset) sends the device to the relative preset position of 500.0 µm to the current position of the device.

Recommendations: The **DRP** command provides faster response and better synchronization between axes than the standard single axis Direct Execute Relative Motion command.

Related commands: **DRP**, **SMRP**, **SMPL**

EESTAT - Read Execution Error Register

Syntax: `[n]EESTAT`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Execution Error Register is to be queried. If the axis identifier [n] is omitted the command will not be executed and a command error will be posted. No parameter is allowed with this command.

The `“?”` is optional, the system will always return a response from the **EESTAT** command.

Function: This reads and returns the values in the Execution Error register. The **EESTAT** function should be used when **ESTAT** returns an `“Execute Error”` bit 22 set response. This error register defines the nature of an Execute Error. The **EESTAT** Register is bit mapped. The bits are listed below; most significant bit first:

0	Edge Range Error
1	Edge Width Error
2	Edge Overflow Error
3	EC Slope Error
4	Octant Seq Error
5	Timeout
6	Motor Inh Error
7	Invalid Data
8	Motor Stall
9	Bus Limit Error - Hard stop in Axis
10	Lockout Error - Interruption of AZ
11	Timeout Error
12	Laser Position Error
13	Data Xfer Error
14	Bus Command Error
15	Position Follow Error
16	AZ Offset Error
17	External Signal Error
18	System Limit Error
19	Axis Motor Error
20	HSCI RAM Bad
21	HSCI Bad Cmd Error
22	HSCI Overrun Error

The **EESTAT** register reports systems error for axes and optional hardware and enhancements for the PM500. Not all errors may be germane to your particular system configuration.

Refer to Section 19.7 - Troubleshooting Tables for correction procedures of system errors.

Returns: Bit mapped value in hexadecimal format

Examples:

Send: `AR`
Receive: `AE+000456.0`
an “`E`*” error, check* ***ESTAT***
Send: `ESTAT`
Receive: `AE$400000`
The error returned is Hex 400000 which equalsdecimal 4194304, which is bit 22, thus the error is an Execute Error. Refer to ***EESTAT****.*

Send: `EESTAT`
Receive: `AE$000100`
Hex 100 equals decimal 256, which is bit 8 on the ***EESTAT*** *register—Motor Stall*

Recommendations: Whenever the status character returns an “E” you should follow with an **ESTAT** query to identify the nature of the error. Should the **ESTAT** response indicate an “`Execute Error`”, bit 22, you should query the **EESTAT** to identify the nature of the error.

If STAT? returns an “`E`”

Read **ESTAT** response

Error defined? Then done.

s bit 22 = “`Execute Error`” set?

Read **EESTAT** response to identify error.

Related commands: **STAT?**, **ESTAT**

ENAINT - *Control Internal Configuration*

Syntax: `(S)*ENAINT <$><Hexadecimal bitmap value>`

Parameters: This is a system level command. No axis identifier is allowed else the command will not be executed and a command error will be posted. The parameter must be in a hexadecimal format and thus requires the value to be pre-pended with a dollar sign ($). A space is allowable between the command and the parameter but is not required.

*"S" system specifier required under SCUM1 mode.

NOTE

This command is only for use in conjunction with system communication mode of SCUM 1 with 11.14 or later firmware.

Function: This command allows custom selection of communications options and provides a wider variety of options than the pre defined modes or **SCOMOPT** commands. The parameter which represents the bitmap corresponding to your preferred communications methodology, $#, is an ASCII character string representing a *hexadecimal* integer (e.g., $0 through $FFFF). The dollar sign which specifies that the number in hexadecimal format *must* be included in the command string sent to the controller. The communications interface specified in the descriptions should correspond to the DIP switch settings which select the hardware interface. Refer to the RS-232 or GPIB section of this manual for proper DIP setting support of these communication modes.

NOTE

When you send a S/ENAINT command, it is automatically stored in non-volatile memory and becomes the power-up default for the controller. For 5-axis controllers however, SCUM 0 is always the power-up default, so if you wish to utilize the new features available in SCUM 1 mode, be sure to issue SCUM 1 as your first command after power-up or mode of SCUM 1 with 11.14 or later firmware.

Bits 7 through 11 specify the command and response terminator. One of these *must* be selected.

Bit	Description	Options	Comments
0	Limit Halt	0: All axes halt when any axis encounter limit 1: Only affected axis will halt	GPIB & RS-232
1	Limit out message	0: System will respond with <axis>L when commanded out of limit condition. 1: No message issued when axis move out of limit	RS-232 only
2	Query Echo	0: System will respond to queries by echoing the query and appending the numeric response 1: No query echo; will prepend a status character to axis or system specifier: D:Done, L:Limit, E:Error, M:Motor off.	GPIB & RS-232
3	ASCII Status Character	0: All status returns from the PM500 will have a status character inserted immediately after the axis specifier. 1: No status character inserted in responses.	RS-232 only
4	ASCII Command	0: No acknowledgement when a command is received. 1: ASCII character <axis>A or SA (system acknowledge) will be returned after each command received.	GPIB/RS-232
5	Sign-on message at power-up	0: Disable Sign-on message at power-up 1: Enable Sign-on message at power-up	RS-232 only
6	RS-232 Echo	0: No echo: 1: Echo enabled	
7	Carriage Return Command Termination	0: Command termination to be defined by user via bits 8, 9, 10, 11 (other than Carriage Return(CR)) 1: Carriage return (CR, ASCII 0DH) Command termination. Responses will also be terminated by CR.	GPIB
8	Line Feed Command Termination	0: Command termination to be defined by user via bits 7, 9, 10, 11 (other than Line Feed (LF)) 1: Line Feed termination (LF) command termination. Response will also be terminated by line feed. (No carriage return)	GPIB
9	EOI Command	0: No EOI sent 1: Commands termination will be terminated by CR\LF\EOI.* EOI is always recognized as a command terminator, regardless of the status of this bit.	Recommended under GPIB*
10	CR\LF	0: Command termination to be defined by user via bits Command 7, 9, 10, 11 (other than CR\LF) 1: Commands will be terminated by CR\LF	GPIB
11	CR commands CR\LF response Termination	0: Command termination to be defined by user via bits 7, 9, 10, 11 (other than CR for commands, CR\LF for responses) 1: Commands will be terminated by CR Responses will be terminated by CR\LF	GPIB

Bit		*Description*	*Options*	*Comments*
12*	Hexadecimal	0: Numbers will be specified in decimal format unless Format one format or the other is required by the command. When specifying Hexadecimal numbers, you must prepend the number with a dollar sign ($) 1: All numbers will be specified in Hexadecimal format. This improves system speed slightly. Hexadecimal significant digit of a hexadecimal position number is equivalent to the axis resolution.		GPIB\RS-232
13*	2nd Generation Serial Poll Bit Format	0: Early Serial Poll Bit mapping (5 axis controllers) used. See Serial Poll Format "A". ***Note: System must be in SCUM 0 mode in conjunction with this designation.*** 1: New Serial Poll Bit mapping (6 & 12 axis controllers) used. See Serial Poll format "B". ***Note: System must be in SCUM 1 mode in conjunction with this designation.***		GPIB
14*	SRQ on Message	0: No SRQ asserted when characters are present in the PM500 output buffer. (MAV bit off) 1: SRQ asserted when characters are present in the PM500 output buffer (MAV bit on). We recommend *not* masking this and SSRQCTL bit 4 both on.		GPIB

***Not available under SCUM 0.**

Returns: Current **ENAINT** register mask value in hexadecimal

Range: In Hexadecimal format $0 - $FFF (under SCUM0)
In Hexadecimal format $016- $EF (under SCUM1)

Default: For RS-232 communications:

CMD	Bit	Value	Description
ENAINT $836	0	0	Global axes halt on fault/limit
	1	1	No Limit out message
	2	1	Query Echo
	3	0	Status Character
	4	1	Acknowledge
	5	1	Sign-on enabled
	10	1	CR LF

For IEEE-488 Communications:

CMD	Bit	Value	Description
ENAINT $606	0	0	Global axes halt on fault/limit
	1	1	No Limit out message
	2	1	Query Echo
	4	0	Acknowledge disabled
	5	0	Sign-on disabled
	7	0	No Carriage Return
	9	1	CR\LF\EOI
	10	1	CR LF with CR terminator

Example: Send: `ENAINT $30F`
Receive: `NONE`

The above example sets ENAINT bits 0,1,2,4,7 and 9 ON.

CMD	Bit	Value	Description
ENAINT $30F	0	1	Only faulted axis halt
	1	1	No Limit out message
	2	1	No Query Echo
	3	1	No Status
	8	1	LF Terminator
	9	1	CR\LF\EOI terminator

Recommendations: **SCUM 0** `ENAINT$606`
SCUM 1 `SENAINT$2606`

Related commands: **S/COMOPT**, **S/SRQCTL**

ESP - Emergency Shutdown Protocol

Syntax: `[n]ESP <n>`

Parameters: `[n]` defines the axis `n = X, Y, Z, A, B, C` whose emergency shutdown protocol is to be defined. If no axis identifier is specified the command will be ignored and an error will be posted.

`<N> Parameter:` numeric value based on the desired configuration of the **ESP** register.

Definition: This command defines the action a specified axis will take upon the shunting (reconnection) of the controllers emergency shutdown BNC after an open (motor shutdown) condition. Modifying this parameter allows the user to define the state of the specified axis upon reconnection. Upon reconnection (shunting) of the controllers emergency stop BNC the system will transfer the designated command into specified axis command register.

Range:

Value	*Command*	*Definition*
0	M	Motor OFF
1	T	Transfer Position

Default: ESP 0:
”M” command transferred; Motor OFF command will be transferred upon reconnect.

Examples: `XESP 1`

X axis will “T” transfer position upon emergency shutdown reconnect.

Recommendations: This command should only be used in conjunction with the hardware connection with the PM500 controllers Emergency Shutdown BNC.

ESTAT *- Read Error Register*

Syntax: `[n]ESTAT`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose error register is to be read. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command. No parameter is allowed with this command.

Function: This reads and returns the values in the error register for the specified axis. The **ESTAT** function should be used when **STAT?** returns an `“E”` error message. This error register defines the nature of the error. In the instance of `“Execute Error”` will refer you to the **EESTAT** Execute Error register. The **ESTAT** register is bit mapped. The bits are listed below, most significant bit first. The **ESTAT** register is divided into four categories:

Bit	***Data Error***
0	Data not present
1	Extra data present
2	Invalid data entry
3	Data overrun
4	Invalid data type
5	Reserved
6	Query invalid
7	Read only command

Bit	***Parser Errors***
8	Invalid character
9	Command overrun
10	Field overrun
11	Missing command
12	Unknown command
13	Lockout violation
14	Invalid specifier
15	Read only

Bit	***Motion Errors***
16–22	Axis failure code
23	Execute error - *refer to **EESTAT***

Bit	***Specifier***	***Under Scum1***
24	X Axis	Axis Bitmap
25	Y Axis	Axis Bitmap
26	Z Axis	Axis Bitmap
27	A Axis	Axis Bitmap
28	B Axis	Axis Bitmap
29	System	Axis Bitmap
30	Not used	
31	Not used	
32	Not used	

Returns: Bit mapped value in hexadecimal format.

Examples:

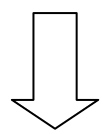
Send: `AR`
Receive: `AE+000456.0`
an `"E"` *error, check* ***ESTAT***

Receive: `AE$400000`
The error returned is Hex 400000 which equals decimal 4194304, which is bit 22, thus the error is an Execute Error. Refer to ***EESTAT****.*

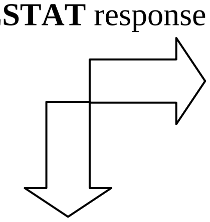
Send: `EESTAT`
Receive: `AE$000100`
Hex 100 equals decimal 256, which is bit 8 on the ***EESTAT*** *register—Motor Stall.*

Recommendations: Whenever the staus character returns an `"E"` you should follow with an **ESTAT** query to identify the nature of the error. Should the **ESTAT** response indicate an `"Execute Error",` bit 22, you should query the **ESTAT** to identify the nature of the error.

If **STAT?** returns an `"E"`

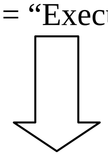


Read **ESTAT** response



Error defined? Then done.

Is bit 22 = "Execute Error?" set?



Read **EESTAT** response to identify error.

Related commands: **STAT?, ESTAT, EESTAT**

***F* - Fiducial Seek**

Syntax: `[n]F <option>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` which will be commanded to conduct the fiducial search. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

WARNING

This command will initiate the motion of the specified axis. Be certain that the device is clear of obstructions before issuing this command.

Function: This command instructs the specified axis to conduct a fiducial search.

This will cause the specified axis to move to the fiducial point (center of travel) either directly or *See*k the limit of travel first depending upon the option selected. After the fiducial is achieved the position registers are cleared, and this point becomes zero (home) position for subsequent motions. The fiducial reference frame is always retained (even when the position register is cleared or offsets are used) until power-off or system restart.

The glass scale on linear motion devices have a fiducial track located on the scale in the center of the stage travel. This provides a mechanical reference point that can be used as a "home" when the position registers are cleared or as a reference point for your experimental setup. The repeatability of the fiducial is equal to the system resolution.

Options:

0	=	fiducial *See*k directly (most efficient method)
-1	=	Move towards motor away limit then fiducial *See*k
+1	=	Move towards motor side limit then fiducial *See*k

Returns: `NONE` - Specified axis carriage should move to center of travel (fiducial) location.

Examples: Send:`XF-1.`

In the above example the X axis would move to the extreme of travel in the direction away from the motor then *See*k fiducial at center of travel.

Recommendations: The fiducial should be used a reference point in your experimental setup. This will provide a mechanical reference between the stage and external components of your experiment. For example, a starting point in a run, or a reference when the position registers have been cleared. The **OFFSET** command allows you to create a “virtual home” relative to the actual zero position from where subsequent motions can be referenced.

You should allow one hour system warm-up time for the highest repeatability of the fiducial reference point to their mechanical setup.

Related commands: **FV**, **FNUM, OFFSET**

FNUM - Define Fiducial Default Number (Rotary Stages Only)

Syntax: `[n]FNUM <parameter>`

Parameters: `[n]` defines the axis `n = X, Y, Z, A, B, C` whose fiducial number is to be defined. If no axis identifier is specified the command will be ignored and an error will be posted.

`<N> Parameter:` numeric value based on the desired range defined below.

Definition: This command defines the default Fiducial point for the "F" Fiducial Seek command for Rotary Devices. There are 8 fiducial locations at 45¼ increments about the stage. Modifying this parameter allows the user to define which Fiducial reference point the "F" fiducial seek will locate to.

Range:

Value	***Location Degrees***
0*	0¼
1	45¼
2	90¼
3	135¼
4	180¼
5	225¼
6	270¼
7	315¼

**Default*

Default: 0

Examples: `XFNUM 4`

This example defines the default Fiducial location to 4 - 180°relative to the origin Fiducial 0.

Recommendations: This command allows the user to customize the mechanical reference point (Fiducial) for a rotary device. The Fiducial provides a repeatable mechanical reference for experimental setups. To seek the specified Fiducial location use the "F" Fiducial seek command.

Related commands: **F**

FV - Fiducial Velocity

Syntax: `[n]FV <option>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose fiducial search velocity will be modified. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command defines the velocity used during a fiducial search. The fiducial search is done in the Low Speed Loop. Programming velocities above maximum will be ignored. Maximum Fiducial search velocity is the current **VL** (velocity) value.

Range: 0 to the current **V** (velocity) setting

Returns: `NONE`

Examples: Send: `XFV 150`

In the above example the X axis fiducial search velocity is set to 150 µm/sec.

Recommendations: The fiducial search velocity should be reduced when reduction in **LACCEL** and **LDECEL** Low Speed Loop parameters are required due to the weight or size of the load.

Related commands: **FV**, **OFFSET**

FV? - Fiducial Velocity Query

Syntax: `[n]FV?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose fiducial search velocity will be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the current fiducial search velocity setting. Maximum Fiducial search velocity is the current **VL** (velocity) value.

Returns: Current Fiducial Search velocity in µm/sec or arc-sec

Examples:

Send: `XFV?`
Receive: `XD+000150.0`

In the above example the X axis fiducial search velocity was set to 150 µm/sec.

Recommendations: The fiducial search velocity should be reduced when reduction in **LACCEL** and **LDECEL** Low Speed Loop parameters are required due to the weight or size of the load.

Related commands: **F**, **FV**, **OFFSET**

G - Go to Position Absolute

Syntax: `[n]G <position>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose absolute position is to be commanded. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command moves the specified axis to the absolute position as specified and servos to that location. The displacement is an absolute measurement from the zero (home) position. The position can be read using the **R** or **G**? command.

Returns: `NONE`

Examples: Send: `XG 25000.0`

The above example commands the X axis to move to 25000.0 µms absolute position.

Recommendations: After issuing this command use the status character to tell when the motion is complete or via GPIB enable the status registers to signal an SRQ when motion is complete. Via GPIB you can Serial Poll or Wait for SRQ to tell when motion is complete.

Related commands: **C**, **R**, **G**?

G? - Position Absolute Query

Syntax: `[n]G?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose absolute position is to be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries and reports the position value of the **G** command from the command register for the specified axis. The **G**? command does not report the contents of the position register but rather the command register only providing the last **G** value specified for that axis.

Returns: Current absolute position value specified by the **G** command in µm or arc-sec.

Examples:

Send: `XG?`
Receive: `XD+000250.0`

In the above example the **G** value for the X axis is queried and found to be 250 µm.

Recommendations: The **R**? command should be used when axis position information is required. The **G**? should be used to verify the values defined by the **G** command.

Related commands: **R**, **G**

GP - Go to Preset Absolute Position

Syntax: [n]GP <position>

Parameters: [n] defines the axis identifier, n = X, Y, Z, A, B, C whose preset position is to be defined. If the axis identifier [n] is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier [n] and the command.

Function: This commands the specified axis to move to the preset absolute position as specified for the DP command. If no value is set for the DP command the move will default to 0.

Returns: NONE

Examples: XDRP 100.0
Send: XGP (X will commence motion as specified by DRP)
Receive:

The above example defines an absolute position of 100.0 for the X axis. The **GP** command is the trigger to execute the preset move.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **DP**, **DRP**, **SMP**, **SMPL**

GR - Go Relative Move

Syntax: `[n]GR <distance>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose axis is to be moved. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command moves the specified axis to a relative position of the last commanded position and servos to that location. It is important to note that this command always adds the relative position to the current position in the command register instead of the actual position of the axis. If the latter were true, cumulative position errors would result.

The move will take place in the high speed loop if the distance of the move is larger than the setting of **LSIZE**, otherwise it will default to the low speed loop. The **LGR** command forces the move to occur within the Low Speed Loop regardless of size. *See* **LGR**.

NOTE

This commands motion to the last commanded position. If the current position was arrived at manually the stage will first return to the last commanded position then move the specified relative distance from that point.

Range: Any valid position value in µm or arc-sec

Returns: `NONE`

Examples: Send: `XGR 3600`

The above example sends the X axis 3600 arc-sec or 1 deg relative to the last commanded position.

Recommendations: Due to the unique way the PM500 handles relative motion commands, absolute commands **G** and the relative command **GR** can be used interchangeably without the concern of accumulating errors which are normally associated with relative move commands.

Related commands: **LGR**, **G**

GRP - Go to Preset Relative Position

Syntax: `[n]GRP <position>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose preset position is to be defined. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This commands the specified axis to move to the preset relative position in µm or arc-sec as specified for the **DRP** command. The move will take place in the high speed loop if the distance of the move is larger than the setting of **LSIZE**, otherwise it will default to the low speed loop. If no value is set for the **DRP** command the move will default to 0 or "home" position.

Returns: NONE

Examples:

Send: `XDRP 500.0`
Send: `XGRP`

In the above example the preset position for the X axis is set to 500 µms via the **DRP** (Define Relative Preset position) command. The **GRP** command triggers the move as set by **DRP**.

Recommendations: NONE

Related commands: **DP**, **DRP**, **SMPL**, **SMRP**

H - Home

Syntax: [n]H

Parameters: [n] defines the axis identifier, n = X, Y, Z, A, B,… whose axis is to be commanded. If the axis identifier [n] is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier [n] and the command. No parameter is allowed with this command.

Function: This command moves the specified axis to its home position which is defined as 0.0. 0.0 in the position register. 0 can be defined in several ways, therefore the actual location of the “Home” can change with exception of fiducial. The following commands allow you to define home in several ways.

F = Fiducial command finds mechanical home.
C = Clear commands defines the current location as 0.0
OFFSET= Offset command defines a “virtual home” which can be offset from a fiducial home or Clear defined home.

Returns: `NONE`

Examples: Send: `YH`

In the above example the Y axis is commanded to the 0.0 or home position.

Recommendations: You should note that the “Home” position is 0.0 and its mechanical location relative to the stage or your setup is defined by how the 0.0 is defined. The F0 fiducial will place 0.0 at the center of stage travel.

The **C** (clear) command will declare 0.0 at the current carriage location. Once the system has been turned OFF, the position register is cleared.

Related commands: **C**, **OFFSET**, **RPOS**

HSLOPE - Define Holding Slope

Syntax: `[n]HSLOPE <value>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Holding SLOPE is to be modified. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command defines the holding slope parameter for the specified axis. **HSLOPE** effects the positioning holding stiffness of the servo system. If **HSLOPE** is set too large, motion commands may not execute to completion - the axis will be "busy" for long periods of time. If the **HSLOPE** is set too small, the system may oscillate when excited.

HSLOPE is in units of distance per **DAC count.**

Values: 0.1 to 0.015

Examples: Setting **HSLOPE** is straightforward if the following procedure is observed.

NOTE

This procedure assumes the system has been properly installed and that the adjustments are correct.

If the stage stalls: decrease **HSLOPE**
If the stage oscillates: increase **HSLOPE**

- Set **HSLOPE** to one-half the resolution of the system. Example: In a .05-micron system, use .025 **HSLOPE** to start with.
- Adjust **HSLOPE** until oscillations disappear (each time **HSLOPE** is changed, lightly tap the end of the stage). Note the value of **HSLOPE** determined at this time, and proceed to Step 4.
- If the initial value of **HSLOPE** set in Step 2 caused no oscillations, decrease **HSLOPE** until oscillations appear (each time **HSLOPE** is changed, lightly tap the end of the stage). Note the value of **HSLOPE** that was used just before the oscillations appeared.

- Take the value of **HSLOPE** noted in Steps 2 or 3 and increase it by approximately 15% and enter it. Check to see if the stage oscillates by tapping on the stage lightly. If oscillations persist, repeat Steps 2 and 4 using an oscilloscope to look for oscillations.

User set values can be stored in non-volatile memory using the [n]SAVEAX command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: NONE

INITCS - Initialize Axis Current Sense Parameters

Syntax: `[n]INITCS`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose current sense parameters is to be initialized. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command initializes the current sense parameters to power-up values as stored by the **SAVECS** command. If no values are stored via **SAVECS** the values will return to factory default.

Returns: `NONE`

Examples: Send: `XINITCS`

The above example restores the current sense parameters to power-up value or those stored via the **SAVECS** command.

Recommendations: NONE

Related commands: **DEFCS**, **SAVECS**

INITLM - Restore Axis Soft Limits

Syntax: `[n]INITLM`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose axis soft limits are to be restored. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command. No parameter is allowed with this command.

Function: This command restores the soft limits for the specified axis to non-volatile stored settings as saved by **SAVELM**.

Returns: `NONE`

Examples: Send: `YINITLM`

The above example restores the Y axis soft limits to the values as stored by **SAVELM**.

Recommendations: NONE

Related commands: **SAVELM**, **NSLIM**, **PSLIM**, **RPOS**

INITOM - Initialize PM500-K6 to User Saved Settings

Syntax: `INITOM`

Parameters: No axis identifier is required for this command. No parameters are allowed with this command otherwise a command error will occur.

Function: This command initializes the PM500-K6 to power-up values as stored by the SAVEOM command. If no values are stored via SAVEOM the values will return to factory default.

Returns: `NONE`

Examples: Send: `XINITOM`

The above example restores the current sense parameters to power-up value or those stored via the **SAVEOM** command.

Recommendations: NONE

Related commands: **DEFOM**, **SAVEOM**

LACCEL - Define Low Speed Loop Acceleration

Syntax: `[n]LACCEL <accel>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Low Speed Loop acceleration is to be modified. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command defines the Low Speed Loop Acceleration for the specified axis in mm/sec^2 or k arc-sec/sec^2. This parameter is used for controlling the ramp up profile of moves which are less than or equal to the **LSIZE**. Adjusting this parameter is useful for improving step and settle responses for small moves. Actual acceleration is somewhat dependent upon system load; no absolute calibration is implied in this parameter.

Range: Device dependent; Any valid value in mm/sec^2 or k arc-sec/sec^2

Returns: `NONE`

Examples: Send: `XLACCEL 20.0`

The above example sets the Low Speed Loop acceleration to 20mm/sec^2

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **LDECEL**, **SAVEAX**, **LV**

LACCEL? - *Define Low Speed Loop Acceleration Query*

Syntax: `[n]LACCEL?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Low Speed Acceleration is to be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the Low Speed Loop Acceleration value for the specified axis in mm/sec^2 or k arc-sec/sec^2. The **LACCEL** parameter is used for controlling the ramp up profile of moves which are less than or equal to the **LSIZE**. Adjusting the **LACCEL** parameter is useful for improving step and settle responses for small moves. Actual acceleration is somewhat dependent upon system load; no absolute calibration is implied in this parameter.

Range: Device dependent; any valid value in mm/sec^2 or k arc-sec/sec^2

Returns: Current **LACCEL** value in mm/sec^2 or k arc-sec/sec^2

Examples:

Send: `XLACCEL?`
Receive: `XD+000050.0`

The above example queries the Low Speed Loop acceleration for the X axis, finding it at 50mm/sec^2.

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The **LACCEL** value cannot be queried on-the-fly—the return would be the set **LACCEL** value. The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **LDECEL**, **SAVEAX**, **LV**

LDECEL - *Define Low Speed Loop Deceleration*

Syntax: `[n]LDECEL <decel>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Low Speed Loop Deceleration is to be modified. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command defines the Low Speed Loop deceleration for the specified axis in mm/sec^2 or k arc-sec/sec^2. This parameter is used for controlling the ramp down profile of moves which are less than or equal to the **LSIZE**. Adjusting this parameter is useful for improving step and settle responses for small moves. Actual deceleration is somewhat dependent upon system load; no absolute calibration is implied in this parameter.

Range: Device dependent; Any valid value in mm/sec^2 or k arc-sec/sec^2

Returns: `NONE`

Examples: Send: `XLDECEL 20.0`

The above example sets the Low Speed Loop deceleration to 20mm/sec^2

User set values can be stored in non-volatile memory using the [n]SAVEAX command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **LACCEL**, **SAVEAX**, **LV**

LDECEL? - Low Speed Loop Deceleration Query

Syntax: [n]LDECEL?

Parameters: [n] defines the axis identifier, n = X, Y, Z, A, B, C whose Low Speed Loop Deceleration is to be modified. If the axis identifier [n] is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier [n] and the command.

Function: This command queries the Low Speed Loop deceleration for the specified axis in mm/sec^2 or k arc-sec/sec^2. The **LDECEL** parameter is used for controlling the ramp down profile of moves which are less than or equal to the **LSIZE**. Adjusting this parameter is useful for improving step and settle responses for small moves. Actual deceleration is somewhat dependent upon system load; no absolute calibration is implied in this parameter.

Returns: Low Speed Deceleration value in mm/sec^2 or k arc-sec/sec2

Examples:

Send:	XLDECEL?
Receive:	XD+00020.0

The above example queries the Low Speed Loop deceleration at 20mm/sec^2

Recommendations: *See* the System Settings section in this manual for factory settings.

Related commands: **LDECEL**, **LACCEL**, **SAVEAX**, **LV**

LGR - Relative Move in Low Speed Loop

Syntax: `[n]LGR <rel move>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose axis is to be commanded to move. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command defines the relative motion for the specified axis in µms or arc-sec. The motion will be restricted to occur only in the Low Speed Loop. The acceleration, deceleration and velocity for this motion are defined by the values of **LACCEL**, **LDECEL**, and **LV**.

Returns: `NONE`

Examples: Send: `XLGR 500.0`

The above example commands the X axis to move 500 µms. Although the relative move value exceeds the **LSIZE**, (Low Speed Loop Size) the motion will occur in the Low Speed Loop.

Recommendations: NONE

Related commands: **LACCEL**, **LDECEL**, **LV**, **LSIZE**

LSIZE - Define Low Speed Loop Threshold

Syntax: `[n]LSIZE <threshold>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Low Speed Loop threshold is to be defined. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command defines the Low Speed Loop threshold for the specified axis in µms or arc-sec. This parameter is used for defining the threshold point where motions equal to or less than the **LSIZE** value will be executed in the Low Speed Loop. The PM500 has two “Speed Loops” which improves step and settle responses by restricting small moves to take place at low speed and vice versa for large moves. This enables you to execute moves of various sizes without redefining acceleration, deceleration and velocity for each move. The **LSIZE** is the demarcation between these regimes.

Range: Device dependent; any valid value in µms or arc-sec.

WARNING

Do NOT set LSIZE less than system resolution or ZERO. Since position holding is performed within the low speed loop, doing so will partially disable the axis ability to hold and repeat position.

Returns: `NONE`

Examples:

Send:	`XLSIZE 50.0`
Send:	`XLSIZE?`
Receive:	`XD+000050.`

The above example defines the Low Speed Loop threshold to 50 µms for the X axis. The **LSIZE** setting can be queried via **LSIZE?**

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings. Keep in mind that **LACCEL**, **LDECEL** and **LV** define the parameters for low speed moves. **ACCEL**, **DECEL**, and **V** will have ***NO*** effect on moves smaller than the defined **LSIZE**.

Related commands: **LDECEL**, **LACCEL**, **LV**

LSIZE? - Low Speed Loop Threshold Query

Syntax: `[n]LSIZE?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Low Speed Loop threshold is to be queried. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the Low Speed Loop threshold for the specified axis in µms or arc-sec. The **LSIZE** parameter is used for defining the threshold point where motions equal to or less than the **LSIZE** value will be executed in the Low Speed Loop. The PM500 has two "Speed Loops" which improve step and settle responses by restricting small moves to take place at low speed and vice versa for large moves. This enables you to execute moves of various sizes without redefining acceleration, deceleration and velocity for each move. The **LSIZE** is the demarcation between these regimes.

Range: Device dependent; any valid value in µms or arc-sec.

Returns: Current **LSIZE** value in µms or arc-sec.

Examples:

Send:	`XLSIZE?`
Receive:	`XD+000050.`

The above example queries the Low Speed Loop threshold for the X axis.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings. Keep in mind that **LACCEL**, **LDECEL**, and **LV** define the parameters for low speed moves. **ACCEL**, **DECEL**, and **V** will have *NO* effect on moves smaller than the defined **LSIZE**.

Related commands: **LSIZE**, **LDECEL**, **LACCEL**, **LV**

LV - Define Low Speed Loop Velocity

Syntax: `[n]LSIZE <velocity>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Low Speed Loop velocity is to be defined. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier [n] and the command.

Function: This command defines the Low Speed Loop velocity for the specified axis in mm/sec^2 or k arc-sec/sec^2. The **LV** parameter is used for defining the velocity for moves executed in the Low Speed Loop. **LSIZE** defines the threshold for the Low Speed Loop. The PM500 has two "Speed Loops" which improves step and settle responses by restricting small moves to take place at low speed and vice versa for large moves. This enables you to execute moves of various sizes without redefining acceleration, deceleration and velocity for each move. **LSIZE** is the demarcation between these regimes. **LV** has *NO* effect on motions executed in the High Speed Loop.

Range: Device dependent; any valid value in mm/sec^2 or k arc-sec/sec^2

Returns: `NONE`

Examples: Send: `XLV 20.0`

The above example defines the Low Speed Loop velocity at 20.0 mm/sec^2 for the X axis.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings. Keep in mind that **LACCEL**, **LDECEL**, and **LV** define the parameters for low speed moves. **ACCEL**, **DECEL**, and **V** will have *NO* effect on moves smaller than the defined **LSIZE**.

Related commands: **LSIZE**, **LDECEL**, **LACCEL**

LV? - Low Speed Loop Velocity Query

Syntax: `[n]LV?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Low Speed Loop velocity is to be queried. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the Low Speed Loop velocity for the specified axis in mm/sec^2 or k arc-sec/sec^2. The **LV** parameter is used for defining the velocity for moves executed in the Low Speed Loop. **LSIZE** defines the threshold for the Low Speed Loop. The PM500 has two "Speed Loops" which improve step and settle responses by restricting small moves to take place at low speed and vice versa for large moves. This enables you to execute moves of various sizes without redefining acceleration, deceleration and velocity for each move. **LSIZE** is the demarcation between these regimes. **LV** has ***NO*** effect on motions executed in the High Speed Loop.

Range: Device dependent; any valid value in mm/sec^2 or k arc-sec/sec^2

Returns: Current **LV** value in mm/sec^2 or k arc-sec/sec^2

Examples:

Send: `XLV?`
Receive: `XD+000020.0`

The above example queries the Low Speed Loop velocity for the X axis.

Recommendations: **LV** cannot be queried on-the-fly—the return will be the current set **LV** value. Keep in mind that **LACCEL**, **LDECEL**, and **LV** define the parameters for low speed moves. **ACCEL**, **DECEL**, and **V** will have ***NO*** effect on moves smaller than the defined **LSIZE**.

Related commands: **LSIZE**, **LDECEL**, **LACCEL**

M - Motor Off

Syntax: `[n]M`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose motor is to be turned off. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

No parameter is allowed with this command.

Function: This command turns the motor off for the axis specified. This command will terminate motion for the specified axis. The location where the "**M**" command was issued becomes the last commanded position. The stage can be moved manually via the knob opposite the motor side of the leadscrew. The **M** command does not disable the encoder, thus, the controller keeps track of stage position, which can be queried via the "**R**" command with the motor off.

The motor can be turned on in the following ways:

T; (Transfer position) energizes motor and servos to current position.

Any command which causes motion

NOTE

Any motion commands such as G, GP, and GR will move relative to the location where the M command was initiated, i.e. the last commanded position.

Returns: `NONE`

Examples: Send: `XM`

The above example turns the X axis motor off.

Recommendations: This command should be used when manual manipulation of the stage is required. Should manual motion cause the stage to reach a limit the motor will re-energize to prevent motion beyond limit.

Related commands: **T**, **R**

M? - Motor Status Query

Syntax: `[n]M?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose motor status is to be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

No parameter is allowed with this command.

Function: This command queries the motor status, whether ON or OFF, for the axis specified.

Returns: 0 = Motor OFF
1 = Motor ON

Examples: Send: `XM?`
Receive: `XD+00001.0`

The above example queries the X axis motor status, which is on.

Recommendations: The motor can be turned on using the **T** transfer position command.

Related commands: **M**, **T**, **R**

MPACC - Define Acceleration for Preset Motion

Syntax: `[n]MPACC <acceleration>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose preset motion acceleration is to be set. If the axis identifier is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier and the command.

NOTE

This command is only for systems operating under SCUM 1 mode.

Function: This command defines the acceleration for the specified axis in mm/sec^2 or k arc-sec/sec^2 for Preset Motion. This parameter is used for controlling the ramp up profile of a preset move. Adjusting this parameter allows very soft or harsh moves to be programmed. Actual acceleration is somewhat dependent upon system load; no absolute calibration is implied in this parameter. This parameter has no effect on motions executed outside of the use of the preset motion function.

Returns: `NONE`

Range: Device dependent

Examples:

Send:	`XMPACC 500`
Send:	`XMPACC?`
Receive:	`XD+000500.0`

The above example sets the preset motion acceleration for the X axis at 500mm/sec^2.

User set values can be stored in non-volatile memory using the [n]SAVEAX command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

NOTE

The setting of motion parameters for preset motion must be enabled via bit 15 on the SMPL System Motion Preset Logic register, otherwise this setting will be ignored.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. For large or heavy loads the MPACC should be decreased to avoid long settling times or vibration caused by sudden acceleration. See the System Settings section in this manual for factory settings.

Related Commands: **MPDEC**, **MPVEL**, **SMPL**, **SMRP**, **SMP**

***MPDEC** - Define Deceleration for Preset Motion*

Syntax: `[n]MPDEC <acceleration>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose preset motion deceleration is to be set. If the axis identifier is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier and the command.

NOTE

This command is only for systems operating under SCUM 1 mode.

Function: This command defines the deceleration for the specified axis in mm/sec^2 or k arc-sec/sec^2 for Preset Motion. This parameter is used for controlling the ramp down profile of a preset move. Adjusting this parameter allows very soft or harsh moves to be programmed. Actual deceleration is somewhat dependent upon system load; no absolute calibration is implied in this parameter. This parameter has no effect on motions executed outside of the use of the preset motion function.

Returns: NONE

Range: Device dependent

Examples:

Send: `XMPDEC 500`
Send: `XMPDEC?`
Receive: `XD+000500.0`

The above example sets the preset motion deceleration for the X axis at 500mm/sec^2.

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

NOTE

The setting of motion parameters for preset motion must be enabled via bit 15 on the SMPL System Motion Preset Logic register, otherwise this setting will be ignored.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. For large or heavy loads the **MPACC** should be decreased to avoid long settling times or vibration caused by sudden acceleration. See the System Settings section in this manual for factory settings.

Related Commands: **MPACC**, **MPVEL**, **SMPL**, **SMRP**, **SMP**

MPVEL - Define Velocity for Preset Motion

Syntax: `[n]MPVEL <velocity>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose preset motion velocity is to be set. If the axis identifier is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier and the command.

NOTE

This command is only for systems operating under SCUM 1 mode.

Function: This command defines the velocity for the specified axis in µm/s or arc-sec/sec for Preset Motion. This parameter is used for controlling the top speed of a preset move. Adjusting this parameter allows very soft or harsh moves to be programmed. Actual velocity is somewhat dependent upon system load; no absolute calibration is implied in this parameter. This parameter has no effect on motions executed outside of the use of the preset motion function.

Returns: NONE

Range: Device dependent

Examples:

Send:	`XMPVEL 100`
Send:	`XMPVEL?`
Receive:	`XD+000100.0`

The above example sets the preset motion velocity for the X axis at 100µm/s.

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

NOTE

The setting of motion parameters for preset motion must be enabled via bit 15 on the SMPL System Motion Preset Logic register, otherwise this setting will be ignored.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. For large or heavy loads the MPACC should be decreased to avoid long settling times or vibration caused by sudden acceleration. *See* the System Settings section in this manual for factory settings.

Related Commands: **MPACC**, **MPDEC**, **SMPL**, **SMRP**, **SMP**

MR - Move Relative

Syntax: `[n]MR <r move>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose motion is to be commanded. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command defines the relative distance to move for the specified axis in µms or arc-sec. This command adds the relative position to the actual axis position.

Range: Any valid value in µms or arc-sec

Returns: `NONE`

Examples: Send: `XMR 100.0`

The above example commands the X axis to move 100 µms relative to its current position in the positive direction.

Send : `XMR -100.`

The above example commands the X axis to move 100 µms relative to its current position in the negative direction.

Recommendations: NONE

Related commands: **G**, **LGR**

MR? - Move Relative Query

Syntax: `[n]MR?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose last **MR** command value is to be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the relative distance commanded to move for the specified axis, via the last **MR** command.

Returns: The last relative move value via **MR** in µms or arc-sec.

Examples:

Send: `XMR?`
Receive: `XD+000050.0`

The above example queries the last **MR** value for the X axis.

Recommendations: NONE

Related commands: **G**, **LGR**

NSLIM - Define Negative Soft Limit

Syntax: `[n]NSLIM <limit val>`

Parameters: [n] defines the axis identifier, `n = X, Y, Z, A, B, C` whose Soft Limit is to be defined. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command defines the negative soft limits for the specified axis in µms or arc-sec. The limit is with respect to the fiducial position, and any subsequent commands such as **C**, **OC**, **OFFSET** will adjust this value accordingly. This parameter is used for preventing motion of the stage beyond a certain user-definable point. Adjusting this parameter allows the stage to come to a stop at the programmed soft limit value.

Default: Minimum negative number allowed in position register.

Returns: `NONE`

Examples: Send: `XNSLIM -12500.0`

The above example sets the soft limits of the X axis at 12500.0 µms relative to the fiducial point.

User set values can be stored in non-volatile memory using the `[n]SAVELM` command. User set **SAVELM** values are restored after system power-up. *See* **SAVELM** command for details.

Recommendations: Fiducial must search **F** command to initialize the coordinate system for the motor device before soft limits can be defined.

Related commands: **PSLIM**, **DEFLM**, **INITLM**, **SAVELM**

NSLIM? - Negative Soft Limit Query

Syntax: `[n]NSLIM?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Soft Limit is to be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the negative soft limits for the specified axis as defined via **NSLIM**. The value returned is with respect to the fiducial position. The **NSLIM** parameter is used for preventing motion of the stage beyond a certain, user definable point. Adjusting **NSLIM** allows the stage to come to a stop at the programmed soft limit value.

Returns: Current Negative Soft Limit as set via **NSLIM** in µms

Examples:

Send: `XNSLIM -12500.0`
Receive: `XD-012500.0`

The above example sets then queries the soft limits of the X axis.

Recommendations: Fiducial must search **F** command to initialize the coordinate system for the motor device, before soft limits can be defined.

Related commands: **PSLIM**, **DEFLM**, **INITLM**, **SAVELM**

NULL - Define Null Window Size

Syntax: `[n]NULL <value>`

Parameters: [n] defines the axis identifier, n = X, Y, Z, A, B, C whose null window is to be modified. If the axis identifier [n] is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier [n] and the command.

Function: This command defines the Null window for the specified axis in µms or arc-sec. This parameter is used to modify the done response for motion commands. Adjusting this parameter allows the system to signal completion before it actually reaches position. **NULL** does not affect system precision.

Range: Any valid value in µms or arc-sec. A value of 0 will set **NULL** equal to system resolution. Setting the NULL value too large may cause long motion completion times.

Default: Equal to system resolution

Returns: NONE

Examples: Send: YNULL 0.5

The above example sets the **NULL** for the Y axis at +/– 0.5 µms.

User set values can be stored in non-volatile memory using the [n]SAVEAX command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings. When setting **NULL** to a larger value be aware that motion complete signaling will occur before the system actually reaches position.

Related commands: **NULL?**

NULL? - Null Window Size Query

Syntax: `[n]NULL?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose null window size is to be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the Null window for the specified axis as set via **NULL**. The **NULL** parameter is used to modify the done response for motion commands. Adjusting this parameter allows the system to signal completion before it actually reaches position. **NULL** does not affect system precision.

Range: Any valid value in µms or arc-sec. A value of 0 will set **NULL** equal to system resolution.

Default: Equal to system resolution

Returns: Current **NULL** window size in µms or arc-sec

Examples:

Send: `YNULL?`
Receive: `YD+000000.1`

The above example queries the **NULL** for the Y axis.

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings. When setting **NULL** to a larger value be aware that motion complete signaling will occur before the system actually reaches position.

Related commands: **NONE**

OC - *Offset Clear*

Syntax: `[n]OC`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Offset is to be cleared. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command clears the offset for the specified axis as defined via the **OFFSET** command. **OFFSET** generates a home position by reading the current position and transferring it to the Offset register. The original home position (generated by power up, or a **C** or **F** command) is left undisturbed, and may be found by simply clearing the offset register (**OFFSET 0**) and sending axis home (**H**).

Returns: `NONE`

Examples: Send: `YOC`

The above example clears the offset of the Y axis. An **OFFSET** query would show the **OFFSET** equal to the current position of the stage.

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: NONE

Related commands: **FFSET**

OFFSET - Define Position Offset

Syntax: `[n]OFFSET <O-distance>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose offset is to be defined. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command allows you to specify coordinate offsets relative to home (position = 0). These offsets allow the absolute moves and home commands to be based around the offsets instead of the actual home position. Executing a **C** (clear) or **F** (fiducial) command will clear the offset.

Returns: `NONE`

Examples: Send: `XOFFSET 500.0`

The above example sets an offset home 500 µms relative to the home (position = 0).

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: NONE

Related commands: **OC**

OFFSET? - Position Offset Query

Syntax: `[n]OFFSET?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose offset is to be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the offsets relative to home (position = 0) as defined by the **OFFSET** command. **OFFSET** allows absolute moves and home commands to be based around the offsets instead of the actual home position. Executing a **C** (clear) or **F** (fiducial) command will clear the offset.

Returns: Current **OFFSET** value in µms or arc-sec

Examples: Send: `XOFFSET?`

The above example queries the current offset relative to the home (position = 0).

Recommendations: NONE

Related commands: **OC**

S/OMA - Power On/Off PM500-K6

Syntax: (S)*OMA <option>

Parameters: No axis identifier is allowed with this command. A space is allowable between the command and the parameter but is not required.
**"S" system specifier required under SCUM1 mode.*

Function: his command allows the PM500-K6 Keypad to be turned Off or On, depending upon the option specified.

Options: 0 = Turns PM500-K6 Off
1 = Turns PM500-K6 On

Default: = PM500-K6 On.

Returns: `NONE`

Examples: Send: `SOMA 1`

The above example turns the PM500-K6 ON.

User set values can be stored in non-volatile memory using the `[n]SAVEOM` command. User set **SAVEOM** values are restored after system power-up. *See* **SAVEOM** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMA? - *Power On/Off PM500-K6 Query*

Syntax: `(S)*OMA?`

Parameters: No axis identifier is allowed with this command. A space is allowable between the command and the parameter but is not required.
**"S" system specifier required under SCUM1 mode.*

Function: This command queries the status, Off or On of the PM500-K6 Keypad.

Returns: 0 = PM500-K6 OFF
1 = PM500-K6 ON

Examples: Send: `SOMA?`
Receive: `1`

The above example queries the power state of the PM500-K6.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

N/OMAA - Enable PM500-K6 Axis

Syntax: [n]*OMAA<option>

Parameters: [n] defines the axis identifier, n = X, Y, Z, A, B, C whose **axis to be defined**. If the axis identifier [n] is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier [n] and the command.

Function: **OMAA** allows you to lock-out critical axes of motion.

This command enables or disables the PM500-K6 depending upon the option specified.

Options: 0 = Disable Axis key
1 = Enable Axis key

Default: 1 = Axis key Enabled

Returns: NONE

Examples: Send: XOMAA

The above example disables the PM500-K6 Setup menu for X axis.

User set values can be stored in non-volatile memory using the [n]SAVEOM command. User set **SAVEOM** values are restored after system power-up. *See* **SAVEOM** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

N/OMAA? - PM500-K6 Active Axis Enabled Query

Syntax: `[n]*OMAA?`

Parameters: No axis identifier is allowed with this command.

Function: This command queries the status enabled or disabled of the PM500-K6 depending upon the option returned.

Returns: 0 = Axis key Disabled
1 = Axis key Enabled

Examples: Send: `XOMAA?`

The above example queries the PM500-K6.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMDA - Enable PM500-K6 Display

Syntax: `(S)*OMDA <option>`

Parameters: No axis identifier is allowed with this command. A space is allowable between the command and the parameter but is not required.
**"S" system specifier required under SCUM1 mode.*

Function: This command enables or disables the PM500-K6 Keypad display, depending upon the option specified.

Options: 0 = Turns PM500-K6 display OFF
1 = Turns PM500-K6 display ON

Default: = PM500-K6 display ON

Returns: `NONE`

Examples: Send: `SOMDA 1`

The above example turns the PM500-K6 display On.

User set values can be stored in non-volatile memory using the `[n]SAVEOM` command. User set **SAVEOM** values are restored after system power-up. *See* **SAVEOM** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMDA? - PM500-K6 Display Status Query

`Syntax:` `(S)*OMDA?`

Parameters: No axis identifier is allowed with this command. A space is allowable between the command and the parameter but is not required.
**"S" system specifier required under SCUM1 mode.*

Function: This command queries the status, Off or On of the PM500-K6 Keypad display.

Returns: 0 = PM500-K6 display OFF
1 = PM500-K6 display ON

Examples: Send: `SOMDA?`
Receive: 1

The above example queries the display status of the PM500-K6.

Recommendations: `NONE`

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

N/OMDEL - *Define PM500-K6 Scan Key Delay*

Syntax: `[n] *OMDEL <delay>`

Parameters: [n] defines the axis identifier, n = X, Y, Z, A, B, C whose **axis to be defined**. If the axis identifier [n] is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier [n] and the command.

Function: This command enables you to define the time the axis scan keys must be depressed before scan motion is initiated for the axis.

Range: 0–32768 milliseconds

Default: 500 milliseconds

Returns: `NONE`

Examples: Send: `XOMDEL 1000`

The above example sets the X axis scan key delay to 1000 milliseconds.

User set values can be stored in non-volatile memory using the `[n]SAVEOM` command. User set **SAVEOM** values are restored after system power-up. *See* **SAVEOM** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with OM: Operator Module prefix, **INITOM**

N/OMDEL? - *PM500-K6 Scan Key Delay Query*

Syntax: `[n]*OMDEL?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose **axis to be queried**. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the current axis scan key delay as set by **OMDEL**.

Returns: Currently set **OMDEL** in milliseconds

Examples:

Send: `XOMDEL?`
Receive: `XD+001000.00`

The above example queries scan key delay for the PM500-K6, which is set to 1000 msec (1sec).

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

N/OMDIR - *Define PM500-K6 Axis Key Direction*

Syntax: [n] *OMDIR <direction>

Parameters: [n] defines the axis identifier, n = X, Y, Z, A, B, C whose **axis to be defined**. If the axis identifier [n] is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier [n] and the command.

Function: This command sets the direction of the PM500-K6 X axis keys, forward or reverse depending upon the option specified.

Options: +1 = Forward Motion
–1 = Reverse Motion

Default: +1 = Forward Motion

Returns: NONE

Examples: Send: XOMDIR -1

The above example sets the PM500-K6 X axis key direction in reverse, opposite of the default.

User set values can be stored in non-volatile memory using the [n]SAVEOM command. User set **SAVEOM** values are restored after system power-up. *See* **SAVEOM** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

N/OMDIR? - PM500-K6 Axis Key Direction Query

Syntax: `[n] *OMDIR?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose **axis to be queried**. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the axis keys direction of the PM500-K6 as set by **OMDIR**.

Returns: +1 = Forward Motion
–1 = Reverse Motion

Examples: Send: `XOMDIR?`
Receive: `XD-000001.0`

The above example queries the X axis key direction for the PM500-K6.

Recommendations: NONE

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMECE - Enable PM500-K6 System Error Clearing

Syntax: `(S)*OMECE <option>`

Parameters: No axis identifier is allowed with this command. A space is allowable between the command and the parameter but is not required.
**"S" system specifier required under SCUM1 mode.*

Function: This command enables or disables the PM500-K6 interface from clearing system errors depending upon the option specified.

Options: 0 = Disable PM500-K6 system error clearing
1 = Enable PM500-K6 system error clearing

Default: 1 = PM500-K6 system error clearing enabled

Returns: `NONE`

Examples: Send: `SOMECE 0`

The above example disables the system error clearing capability of the PM500-K6 interface.

User set values can be stored in non-volatile memory using the `[n]SAVEOM` command. User set **SAVEOM** values are restored after system power-up. *See* **SAVEOM** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMECE? - PM500-K6 System Error Clearing Status Query

Syntax: (S)*OMECE?

Parameters: No axis identifier is allowed with this command. A space is allowable between the command and the parameter but is not required.
**"S" system specifier required under SCUM1 mode.*

Function: This command queries the system error clearing ability, enabled or disabled of the PM500-K6 interface.

Returns: 0 = Disable PM500-K6 system error clearing
1 = Enable PM500-K6 system error clearing

Examples: Send: SOMECE 0

The above example disables the system error clearing capability of the PM500-K6 interface.

User set values can be stored in non-volatile memory using the [n]SAVEOM command. User set **SAVEOM** values are restored after system power-up. *See* **SAVEOM** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMHA - PM500-K6 Active Query

Syntax: `(S)*OMHA`

Parameters: No axis identifier is allowed with this command. No parameter is allowed with this command, otherwise a command error will occur.
**"S" system specifier required under SCUM1 mode.*

Function: This command queries whether the PM500-C recognizes the presence of the PM500-K6.

Returns: 0 = PM500-K6 hardware inactive
1 = PM500-K6 hardware active

Examples: Send: `OMHA`
Receive: `1`

The above example queries the active presence of the PM500-K6 interface.

Recommendations: NONE

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMMA - Enable PM500-K6 Menu System

Syntax: `(S)*OMMA <option>`

Parameters: No axis identifier is allowed with this command. A space is allowable between the command and the parameter but is not required.
**"S" system specifier required under SCUM1 mode.*

Function: This command enables or disables the PM500-K6 menu system depending upon the option specified.

Options: 0 = Disable PM500-K6 menu system
1 = Enable PM500-K6 menu system

Default: 1 = PM500-K6 menu system enabled

Returns: `NONE`

Examples: Send: `SOMMA 0`

The above example disables the menu system of the PM500-K6 interface.

User set values can be stored in non-volatile memory using the `[n]SAVEOM` command. User set **SAVEOM** values are restored after system power-up. *See* **SAVEOM** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMMA? - PM500-K6 Menu System Enable Query

Syntax: `(S)*OMMA?`

Parameters: No axis identifier is allowed with this command. A space is allowable between the command and the parameter but is not required.
**"S" system specifier required under SCUM1 mode.*

Function: This command queries the menu status, enabled or disabled of the PM500-K6.

Returns: 0 = Disable PM500-K6 menu system
1 = Enable PM500-K6 menu system

Examples: Send: `SOMMA?`
Receive: `0`

The above example queries the menu system of the PM500-K6 interface.

Recommendations: NONE

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

N/OMMULT - Define PM500-K6 Speed Key Multiplier

Syntax: `[n]*OMMULT <multi>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose **axis to be defined**. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command sets the multiplier value of the speed key for the PM500-K6. When an axis key is depressed, depressing the speed multiplier key increases the current axis slew rate to the current set value of **OMMULT**.

Range: 0–32768

Default: 10

Examples: Send: `XOMMULT 20`

The above example sets the speed multiplier to 20x.

User set values can be stored in non-volatile memory using the `[n]SAVEOM` command. User set **SAVEOM** values are restored after system power-up. *See* **SAVEOM** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

N/OMMULT? - *PM500-K6 Speed Key Multiplier Query*

Syntax: [n]*OMMULT?

Parameters: [n] defines the axis identifier, n = X, Y, Z, A, B, C whose **axis to be defined**. If the axis identifier [n] is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier [n] and the command.

Function: This command queries the current value for the speed multiplier key as set by **OMMULT**.

Returns: 0–32768

Examples:

Send: XOMMULT?
Receive: XD$+000020.0

The above example queries the currently set multiplier value for the PM500-K6 speed multiplier key.

Recommendations: NONE

Related commands: All commands with **OM**: Operator Module prefix, **INITOM.**

S/OMNAD - *Define PM500-K6 Number of Axes Displayed*

Syntax: `(S)*OMNAD <option>`

Parameters: No axis identifier is allowed with this command. A space is allowable between the command and the parameter but is not required.
**"S" system specifier required under SCUM1 mode.*

Function: This command defines the number of axes displayed on the PM500-K6 keypad. The maximum number of axes the PM500-K6 can display is 3.

Options:
1 = 1 Axis displayed
2 = 2 Axis displayed
3 = 3 Axis displayed

Default: 3 or the maximum number of axes installed, whichever is less.

Returns: NONE

Examples: Send: `OMNAD 2`

The above example defines the PM500-K6 interface to display 2 axes.

User set values can be stored in non-volatile memory using the `[n]SAVEOM` command. User set **SAVEOM** values are restored after system power-up. *See* **SAVEOM** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMNAD? - PM500-K6 Number of Axes Displayed Query

Syntax: `(S)*OMNAD?`

Parameters: No axis identifier is allowed with this command.
**“S” system specifier required under SCUM1 mode.*

Function: This command queries the number of axes displayed on the PM500-K6 keypad. The maximum number of axes the PM500-K6 can display is 3.

Returns: 1= 1 Axis displayed
2 = 2 Axis displayed
3 = 3 Axis displayed

Examples: Send: `SOMNAD?`

The above example queries the number of displayed axes on the PM500-K6 interface.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMPDA - *Define PM500-K6 Position of Displayed Axes*

Syntax: `(S)*OMPDA <d-location><axis>`

Parameters: No axis identifier is allowed with this command. A space is allowable between the command and the parameter but is not required.
**"S" system specifier required under SCUM1 mode.*

Function: This command defines the position of and axes displayed on the PM500-K6 keypad. The maximum number of axes the PM500-K6 can display is 3.

Return Options: `<d-location>` The PM500-K6 display positions are numbered starting from the left:

0 = Display position 1
1 = Display position 2
2 = Display position 3

`<axis>`	0 = X	3 = A
	1 = Y	4 = B
	2 = Z	5 = C

Default: 0; 0 = X Axis: left display position
1; 1 = Y Axis: center display position
2; 2 = Z Axis: right display position

Returns: NONE

Examples: Send: `SOMPDA 0,3`

The above example sets axis A to occupy the left display position of the PM500-K6.

User set values can be stored in non-volatile memory using the `[n]SAVEOM` command. User set **SAVEOM** values are restored after system power-up. *See* **SAVEOM** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMPDA? - Define PM500-K6 Position of Display Axes Query

Syntax: `(S)*OMPDA <d-location>?`

Parameters: No axis identifier is allowed with this command. A space is allowable between the command and the parameter but is not required.
**"S" system specifier required under SCUM1 mode.*

Function: This command queries the axes assigned to the display position specified on the PM500-K6 keypad. The maximum number of axes the PM500-K6 can display is 3.

Options: `<d-location>` The PM500-K6 display positions are numbered starting from the left:

0 = Display position 1
1 = Display position 2
2 = Display position 3

`<axis>`	0 = X	3 = A
	1 = Y	4 = B
	2 = Z	5 = C

Examples:

Send: `SOMPDA 2?`
Receive: `4`

The above example queries the axis currently assigned to display position 2, center of the PM500-K6. The return identifies that axis B is currently assigned to display position 2.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMRAE - Enable PM500-K6 Axis Control during Remote Operation

Syntax: (S)*OMRAE<option>

Parameters: No axis identifier is allowed with this command. A space is allowable between the command and the parameter but is not required.
**"S" system specifier required under SCUM1 mode.*

Function: This command enables or disables the PM500-K6 from commencing motion when system motion is currently under way for another axis via RS-232 or IEEE control.

Options: 0 = Disable PM500-K6 Control during motion
1 = Enable PM500-K6 Control during motion

Default: 0 = PM500-K6 Control during motion disabled

Returns: NONE

Examples: Send: SOMRAE 1

The above example enables the PM500-K6 control during motion.

User set values can be stored in non-volatile memory using the [n]SAVEOM command. User set **SAVEOM** values are restored after system power-up. *See* **SAVEOM** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMRAE - *PM500-K6 Axis Control During Remote Operation Query*

Syntax: `(S)*OMRAE?`

Parameters: No axis identifier is allowed with this command.
**"S" system specifier required under SCUM1 mode.*

Function: This command queries the status enabled or disabled of the PM500-K6 depending upon the option returned.

Returns: 0 = PM500-K6 Control during motion disabled
1 = PM500-K6 Control during motion enabled

Examples:

Send: `SOMRAE?`
Receive: `XD$+000001.0`

The above example queries the status of axis control of the PM500-K6 when motion is under way via RS-232 or IEEE.

Recommendations: NONE

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMSA - *Enable PM500-K6 Setup Menu*

Syntax: `(S)*OMSA <option>`

Parameters: No axis identifier is allowed with this command. A space is allowable between the command and the parameter but is not required.
"S" system specifier required under SCUM1 mode.

Function: This command enables or disables the PM500-K6 Setup Menu system depending upon the option specified.

Options: 0 = Disable Menu System
1 = Enable Menu System

Default: 1 = Menu System Enabled

Returns: `NONE`

Examples: Send: `SOMSA 0`

The above example disables the PM500-K6 Setup menu.

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMSA? - PM500-K6 Setup Menu Enabled Query

Syntax: (S)*OMSA?

Parameters: No axis identifier is allowed with this command.
"S" system specifier required under SCUM1 mode.

Function: This command queries the status, enabled or disabled, of the PM500-K6 Setup Menu system depending upon the option returned.

Returns: 0 = Disable Menu System
1 = Enable Menu System

Default: 1 = Menu System Enabled

Returns: NONE

Examples: Send: SOMSA 0?

The above example queries enable status the PM500-K6 Setup menu.

User set values can be stored in non-volatile memory using the [n]SAVEAX command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMSCAN - Define PM500-K6 Key Scan Rate

Syntax: `(S)*OMSCAN <rate><axis>`

Parameters: No axis identifier is allowed with this command. A space is allowable between the command and the parameter but is not required.
**"S" system specifier required under SCUM1 mode.*

Function: This command defines the scan rate for the PM500-K6 axis keys.

Options:

`<rate>` 0 = Low Speed
1 = Medium Speed
2 = High Speed

`<axis>` 0 = X 3 = A
1 = Y 4 = B
2 = Z 5 = C

Default: 0 = Low speed for all axis keys

Returns: `NONE`

Examples: Send: `SOMSCAN 1,0`

The above example sets the X axis key for medium speed.

User set values can be stored in non-volatile memory using the `[n]SAVEOM` command. User set **SAVEOM** values are restored after system power-up. *See* **SAVEOM** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMSCAN? - PM500-K6 Key Scan Rate Query

Syntax: `(S)*OMSCAN <axis>?`

Parameters: A space is allowable between the command and the parameter but is not required.
**"S" system specifier required under SCUM1 mode.*

Function: This command queries the current scan rate of the PM500-K6 as set by **OMSCAN**, for the axis specified.

Returns:

`<axis>`

0 = X	3 = A
1 = Y	4 = B
2 = Z	5 = C

`<rate>`
0 = Low Speed
1 = Medium Speed
2 = High Speed

Examples:

Send: `SOMSCAN 2?`

Receive: `XZ$+000001.0`

The above example queries the currently set scan rate for the Z axis.

Recommendations: NONE

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMSPD - Define PM500-K6 Speed Level

Syntax: `(S)*OMSPD<option>`

Parameters: No axis identifier is allowed with this command. A space is allowable between the command and the parameter but is not required.
**"S" system specifier required under SCUM1 mode.*

Function: This command defines the speed level for PM500-K6 depending upon the option specified.

Options: 0 = Low Speed1 = Medium Speed
2 = High Speed

Default: 1 = Medium speed

Returns: `NONE`

Examples: Send: `SOMSPD 2`

The above example defines the PM500-K6 Speed level at high speed.

User set values can be stored in non-volatile memory using the `[n]SAVEOM` command. User set **SAVEOM** values are restored after system power-up. *See* **SAVEOM** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMSPD? - PM500-K6 Speed Level Query

Syntax: `(S)*OMSPD?`

Parameters: No axis identifier is allowed with this command.
**"S" system specifier required under SCUM1 mode.*

Function: This command queries the currently set speed level of the PM500-K6 as set by **OMSPD**.

Returns: 0 = Low Speed
1 = Medium Speed
2 = High Speed

Examples: Send: `SOMSPD?`
Receive: `2`

The above example queries the currently set speed level of the PM500-K6.

Recommendations: NONE

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMSPDA - *Enable PM500-K6 Speed Select Keys*

Syntax: `(S)*OMSPDA <option>`

Parameters: No axis identifier is allowed with this command. A space is allowable between the command and the parameter but is not required.
**"S" system specifier required under SCUM1 mode.*

Function: This command enables or disables the speed select keys of the PM500-K6 depending upon the option specified.

Options: 0 = Disable Speed Select Keys
1 = Enable Speed Select Keys

Default: 1 = Speed Select Keys Enabled

Returns: `NONE`

Examples: Send: `SOMSPDA 0`

The above example disables the PM500-K6 Speed Select Keys.

User set values can be stored in non-volatile memory using the `[n]SAVEOM` command. User set **SAVEOM** values are restored after system power-up. *See* **SAVEOM** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

S/OMSPDA? - PM500-K6 Speed Select Keys Enable Query

Syntax: `(S)*OMSPDA?`

Parameters: No axis identifier is allowed with this command.
**"S" system specifier required under SCUM1 mode.*

Function: This command queries the status, enabled or disabled, of the PM500-K6 depending upon the option returned.

Returns: 0 = Speed Select Keys Disabled
1 = Speed Select Keys Enabled

Examples: Send: `SOMSPDA?`

The above example queries status of the PM500-K6 Speed Select Keys.

Recommendations: NONE

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

N/OMSTEP - Define PM500-K6 Speed Key Step Size

Syntax: `[n] *OMSTEP <s-key><s-size>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose **axis to be defined**. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command sets the step size for each of the speed keys on the PM500-K6.

Options:

`<s-key>`
0 = Low Speed Key
1 = Medium Speed Key
2 = High Speed Key

`<s-size>` Any valid step size in µms or arc-sec

Default:
0 = 5.0 µms
1 = 10.0 µms
2 = 25.0 µms

Returns: `NONE`

Examples: Send: `OMSTEP 1,50.0`

The above example sets the medium speed key step size to 50 µms.

User set values can be stored in non-volatile memory using the [n]SAVEOM command. User set **SAVEOM** values are restored after system power-up. *See* **SAVEOM** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

N/OMSTEP? - PM500-K6 Speed Key Step Size Query

Syntax: `(S)*OMSTEP <s-key>?`

Parameters: A space is allowable between the command and the parameter but is not required.
**"S" system specifier required under SCUM1 mode.*

Function: This command queries the step size for the specified speed key on the PM500-K6 as defined via **OMSTEP**.

Options: `<s-key>` 0 = Low Speed Key
1 = Medium Speed Key
2 = High Speed Key

Returns: Any valid step size in μms or arc-sec

Examples:
Send: `OMSTEP 2?`
Receive: `XD$+00075.0`

The above example queries the step size for the High speed key.

Recommendations: NONE

Related commands: All commands with **OM**: Operator Module prefix, **INITOM**

PM - Preventive Maintenance

Syntax: `[n]PM`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose axis will commence the preventive maintenance routine. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command begins a preventive maintenance process for the axis specified. The **PM** command cycles the stage between its limits. This does several important things: re-distributes lubricant along the bearings and leadscrew, centers the bearing cages, and auto-zeroes the servo drive electronics. The use of this command is indicated to correct axis stalling due to over-current limits caused by roller cage migration.

The **PM** command first establishes an absolute position reference by finding the fiducial position. The fiducial reference will then be the first fiducial found out of limit. For linear stages, the axis will then try to reach each limit. The axis will install **PMACC** (Preventive Maintenance Acceleration) and **PMVEL** (Preventive Maintenance Velocity) as the motion parameters. If you wish soft limits to be enabled during this process, PMSLE1 (Preventive Maintenance Soft Limit Enable) must first be set and the fiducial position sought via the F0 command. If the system fails to reach limit due to a current sense error, the system assumes that crossed-roller cage migration has reduced travel and will begin a software routine designed to increase travel by progressively moving the cages.

In order to center the cages, the following procedure is utilized. The axis will retreat from the stall location a distance of PMMOVE (Preventive Maintenance Move). The system will then scan the axis into limit. If the axis reaches limit, the axis proceeds on to the next step. If the axis stalls on a current sense error again, the axis compares the location of the last two stall positions to verify that the cages have moved. If this difference is less than **PMNULL** (Preventive Maintenance Null), the axis assumes that the system has stalled against a foreign object and posts a current sense error. If not, the axis restarts this procedure.

Once the axis has successfully centered the cages, the electronic drive circuitry is recalibrated. Programmed values for the aforementioned parameters may be stored in non-volatile memory for recovery on system power-up using the **SAVEAX** command.

Returns: `NONE`

Examples: Send: `XPM`

The above example commences preventive maintenance for the X axis.

Recommendations: The **PM** Preventive Maintenance feature should be used periodically to maintain specified performance and prolong the operating life of linear translation stages.

Related commands: All commands with the **PM** prefix.

PMACC - Define Preventive Maintenance Acceleration

Syntax: `[n]PMACC <accel>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Preventive Maintenance Acceleration is to be specified. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command defines the acceleration used during the execution of the **PM**, Preventive Maintenance command in mm/sec^2 or k arc-sec/sec^2.

Range: Any valid value in mm/sec^2 or k arc-sec/sec^2

Default: System dependent

Returns: `NONE`

Examples: Send: `XPMACC 50`

The above example sets the X axis **PM** acceleration to 50 mm/sec^2.

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings. **PMACC** should be reduced in situations with large or heavy loads.

Related commands: **PM**, **PMOVE**, **PMNULL**, **PMSDIR**, **PMSLE**, **PMVEL**

PMACC? - Preventive Maintenance Acceleration Query

Syntax: `[n]PMACC?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Preventive Maintenance Acceleration is to be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the acceleration used during the execution of the **PM**, Preventive Maintenance command as defined by the **PMACC** command. The return value is in mm/sec^2 or k arc-sec/sec^2.

Range: Current **PMACC** value in mm/sec^2 or k arc-sec/sec^2

Default: System dependent

Returns: `NONE`

Examples:

Send: `XPMACC?`
Receive: `XD$+000050.0`

The above example queries the X axis **PM** acceleration value.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings. **PMACC** should be reduced in situations with large or heavy loads.

Related commands: **PM**, **PMOVE**, **PMNULL**, **PMSDIR**, **PMSLE**, **PMVEL**

PMOVE - Define Preventive Maintenance Move Size

Syntax: `[n]PMOVE <distance>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Preventive Maintenance move size is to be defined. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command defines the move size that will be used during the execution of the **PM**, Preventive Maintenance command in µms or arc-sec

Range: Any valid value in µms or arc-sec

Default: System dependent

Returns: `NONE`

Examples: Send: `XPMOVE 12500.0`

The above example sets the **PMOVE** size to 12500 µms.

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **PM**, **PMACC**, **PMNULL**, **PMSDIR**, **PMSLE**, **PMVEL**

PMOVE? - Preventive Maintenance Move Size Query

Syntax: `[n]PMOVE?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Preventive Maintenance move size is to be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the move size that will be used during the execution of the **PM**, Preventive Maintenance as set via the **PMOVE** command. The return value is in µms or arc-sec.

Returns: Current **PMOVE** value in µms or arc-sec

Examples:

Send: `XPMOVE?`
Receive: `XD$+00012500.0`

The above example queries the **PMOVE** size of the X axis.

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **PM**, **PMACC**, **PMNULL**, **PMSDIR**, **PMSLE**, **PMVEL**

PMNULL - Define Preventive Maintenance NULL

Syntax: `[n]PMNULL <n-size>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Preventive Maintenance **NULL** size is to be defined. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command sets the minimum cage motion allowable for the system to not determine that a foreign object caused the current sense during the execution of the **PM**, Preventive Maintenance command.

Returns: `NONE`

Default: System dependent

Examples: Send: `XPMNULL 300`

The above example sets the cage migration **NULL** value to 300 µms for the X axis.

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **PM**, **PMACC**, **PMOVE**, **PMSDIR**, **PMSLE**, **PMVEL**

PMNULL? - Preventive Maintenance NULL Size Query

Syntax: `[n]PMNULL?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Preventive Maintenance **NULL** size is to be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the minimum cage motion allowable for the system to not determine that a foreign object caused the current sense as defined by the **PMNULL** command. This value was used during the execution of the **PM**, Preventive Maintenance command.

Returns: Current **NULL** size in µms or arc-sec

Examples:

Send: `XPMNULL?`
Receive: `XD$+000050.0`

The above example queries the the **PMNULL** value for the X axis.

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **PM**, **PMACC**, **PMOVE**, **PMSDIR**, **PMSLE**, **PMVEL**

PMSDIR - Define Preventive Maintenance Scan Direction

Syntax: `[n]PMSDIR <option>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose **PM** scan direction is to be defined. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command defines the scan direction that will be used for the execution of the **PM**, Preventive Maintenance command. **PMSDIR** allows you to program which limit the system will scan to. **PMSDIR** is used in systems without center Seeking fiducials.

Options: –1 = Motor away
+1 = Toward motor

Default: +1

Examples: Send: `XPMSDIR -1`

The above example sets the **PM** scan direction away from the motor.

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **PM**, **PMACC**, **PMOVE**, **PMNULL**, **PMSLE**, **PMVEL**

PMSDIR? - Define Preventive Maintenance Scan Direction

Syntax: [n]PMSDIR?

Parameters: [n] defines the axis identifier, n = X, Y, Z, A, B, C whose **PM** scan direction is to be queried. If the axis identifier [n] is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier [n] and the command.

Function: This command queries the scan direction that will be used for the execution of the **PM**, Preventive Maintenance command as set via the **PMSDIR**. **PMSDIR** allows you to program which limit the system will scan to. **PMSDIR** is used in systems without center Seeking fiducials.

Returns: –1 = Motor away
+1 = Toward motor

Examples:

Send: XPMSDIR?
Receive: XD$+000001.0

The above example queries the **PM** scan direction for the X axis.

User set values can be stored in non-volatile memory using the [n]SAVEAX command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **PM**, **PMACC**, **PMOVE**, **PMNULL**, **PMSLE**, **PMVEL**

PMSLE - *Enable Preventive Maintenance Soft Limit*

Syntax: [n]PMSLE <option>

Parameters: [n] defines the axis identifier, n = X, Y, Z, A, B, C whose soft limits restrictions are to be enabled. If the axis identifier [n] is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier [n] and the command.

Function: This command enables the soft limits restriction during execution of the **PM**, Preventive Maintenance command. The soft limits are defined by the **NSLIM** and **PSLIM** command.

Options: 1 = Soft Limit restrictions enabled
0 = Soft Limit restrictions disabled

Default: 1 = Soft Limit restrictions enabled

Examples: Send: XPMSLE 1

The above example enables soft limit restrictions during the execution of the **PM** command for the X axis.

User set values can be stored in non-volatile memory using the [n]SAVEAX command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **PM**, **PMACC**, **PMOVE**, **PMSDIR**, **PMVEL**

PMSLE? - Preventive Maintenance Soft Limit Enable Query

Syntax: `[n]PMSLE?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose soft limits restrictions are to be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the soft limits restriction during execution of the **PM**, Preventive Maintenance command. The soft limits are defined by the **NSLIM** and **PSLIM** command.

Returns: 1 = Soft Limit restrictions enabled
0 = Soft Limit restrictions disabled

Examples:

Send: `XPMSLE?`
Receive: `XD$+000001.0`

The above example query soft limit restrictions during the execution of the **PM** command for the X axis.

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **PM**, **PMACC**, **PMOVE**, **PMSDIR**, **PMVEL**

PMVEL - Define Preventive Maintenance Velocity

Syntax: `[n]PMVEL <velocity>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose **PM** velocity is to be defined. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command sets the velocity that will be used during the execution of the **PM**; Preventive Maintenance command.

Range: Any valid velocity in mm/sec^2 or k arc-sec/sec^2

Default: System dependent

Examples: Send: `XPMVEL 250`

The above example sets the **PM** velocity to 250 mm/sec^2 for the X axis.

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **PM**, **PMACC**, **PMOVE**, **PMNULL**, **PMSDIR**, **PMSLE**

PMVEL? - Preventive Maintenance Velocity Query

Syntax: [n]PMVEL?

Parameters: [n] defines the axis identifier, n = X, Y, Z, A, B, C… whose **PM** velocity is to be queried. If the axis identifier [n] is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier [n] and the command.

Function: This command queries the velocity that will be used during the execution of the **PM**; Preventive Maintenance command as set via **PMVEL**.

Returns: Current **PM** velocity in mm/sec^2 or k arc-sec/sec^2

Examples:

Send: XPMVEL?
Receive: XD$+0000250.0

The above example queries the **PM** velocity for the X axis.

User set values can be stored in non-volatile memory using the [n]SAVEAX command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **PM**, **PMACC**, **PMOVE**, **PMNULL**, **PMS**, **PMSLE**

PSLIM - Define Positive Soft Limit

Syntax: `[n]PSLIM <limit val>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Soft Limit is to be defined. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command defines the positive soft limits for the specified axis in µms or arc-sec. The value is with respect to the fiducial position, any subsequent commands such as **C**, **OC**, **OFFSET**, and **CC** will adjust this value accordingly. This parameter is used for preventing motion of the stage beyond a certain, user definable point. Adjusting this parameter allows the stage to come to a stop at the programmed soft limit value.

Default: Maximum positive value allowed in position register.

Returns: `NONE`

Examples: Send: `XPSLIM 7500.0`

The above example sets the soft limits of the X axis at 7500.0 µms relative to the fiducial point.

User set values can be stored in non-volatile memory using the `[n]SAVELM` command. User set **SAVELM** values are restored after system power-up. *See* **SAVELM** command for details.

Recommendations: A Fiducial must search "**F**" command must be entered to initialize the coordinate system for the motion device, before soft limits can be defined

Related commands: **NSLIM**, **DEFLM**, **INITLM**, **SAVELM**

PSLIM?- Positive Soft Limit Query

Syntax: `[n]PSLIM?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose Soft Limit is to be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the positive soft limits for the specified axis as defined via **PSLIM**. The value returned is with respect to the fiducial position. The **PSLIM** parameter is used for preventing motion of the stage beyond a certain, user definable point. Adjusting **PSLIM** allows the stage to come to a stop at the programmed soft limit value.

Returns: Current Negative Soft Limit as set via **NSLIM** in µms

Examples:

Send: `XPSLIM 7500.0`
Receive: `XD-007500.0`

The above example sets then queries the soft limits of the X axis.

Recommendations: NONE

Related commands: **NSLIM**, **DEFLM**, **INITLM**, **SAVELM**

R - Read Axis Position

Syntax: `[n]R`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose position is to be read. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the current position of the specified axis. The response will include a **STAT** response (*See* **STAT?** command) if enabled by the **ENAINT** command. The PM500 position encoder system is independent of motor status. Thus, position updates and read are possible even with the motor off (**M** command).

Returns: Current position in mms or arc-sec

Examples:

Send: `XR`
Receive: `XD+000754.0`

The above example queries the current position of the X axis, the return also includes a status character depending upon the setting of **ENAINT**, in this example; “**D**” which indicates the X axis is done and servoing to position.

Recommendations: Position can be read on the fly.

Related commands: NONE

RPOS - Read Offset Position

Syntax: `[n]RPOS`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose offset position is to be read. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command reads and returns with the distance between the current reference frame and the fiducial reference frame. This command would be used when an **OFFSET** has been defined and you wish to know the distance between the current position and the true 0.0 fiducial position.

Returns: Current offset distance in mms or arc-sec.

Examples:

Send: `YRPOS?`
Receive: `YD-000345.1`

The above example queries the current positions distance from the fiducial reference frame.

Recommendations: This command is useful in conjunction with the **OFFSET** command.

Related commands: **OFFSET**, **OC**, **F**

S - Scan Axis

Syntax: `[n]S <velocity>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose scan is to be commanded. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This commands the specified axis to scan at the defined velocity in mm/sec^2 or k arc-sec/sec^2 in the High speed loop. This will cause the stage to scan until a limit is reached.

Default: `NONE`

Returns: `NONE`

Examples: Send: `BSCAN 100`

The above example commences the B axis to scan at 100 mm/sec^2.

Recommendations: NONE

Related commands: NONE

SAVEAX - Save Axis Parameter to Non-Volatile Memory

Syntax: `[n]SAVEAX`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose axis parameters is to be stored. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command stores user set parameters in non-volatile memory for a given axis. Once you have configured the axis to the configuration desired, issuing the **SAVEAX** command will save the current condition, and restore it every time the system power is applied. The following commands are saved by **SAVEAX**:

ACCEL	**LACCEL**
DECEL	**LDECEL**
V	**LV**
CSIZE	**LSIZE**
HSLOPE	**NULL**
AZ value	

SAVEAX values can be cleared via storing new **SAVEAX** parameters or the **DEFEE** (Default System to System Settings) Command.

Returns: `NONE`

Examples: Send: `XSAVEAX`

The above example stores users settings for the X axis.

Recommendations: User set values should be saved when favorable adjustment has been achieved.

Related commands: **DEFLT**

SAVEAX? - Query Axis Parameters in Non-Volatile Memory

Syntax: `[n]SAVEAX?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose storage status of axis parameters is to be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries if the users set parameters are stored in non-volatile memory via **SAVEAX**. The following commands are saved by **SAVEAX**:

ACCEL	**LACCEL**
DECEL	**LDECEL**
V	**LV**
CSIZE	**LSIZE**
HSLOPE	**NULL**
AZ value	

SAVEAX values can be cleared via storing new **SAVEAX** parameters, **RESET** or the **DEFEE** (Default System to System Settings) Command.

Returns: 0 = Axis Parameters Unsaved
1 = Axis Parameters Saved

Examples:

Send: `XSAVEAX?`
Receive: `XD+000001.0`

The above example queries if the users settings for the X axis are currently stored.

Recommendations: User set values should be saved when favorable adjustment has been achieved.

Related commands: **DEFLT**

SAVECS - Save Current Sense Parameters to Non-Volatile Memory

Syntax: `[n]SAVECS`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose current sense parameters are to be stored. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command stores users Current Sense parameters in non-volatile memory for the specified axis. Once you have configured the axis, issuing the **SAVECS** command will save the current condition, and restore it every time the system power is applied. The following commands are saved by **SAVECS**.

CSUR
CSMOVE
CSTIME

SAVECS values can be cleared via storing new **SAVECS** parameters, **INITCS** or the **DEFEE** (Default System to System Settings) Command.

Returns: `NONE`

Examples: Send: `YSAVECS`

The above example stores your set Current Sense parameters to Non-Volatile memory for the Y axis.

Recommendations: User set values should be saved when favorable adjustment has been achieved.

Related commands: **DEFCS**, **INITCS**

SAVECS? - Query Status of Current Sense Parameters to Non-Volatile Memory

Syntax: `[n]SAVECS?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose current sense parameters status is to be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the status of the users Current Sense parameters in non-volatile memory for the specified axis. The following commands are saved by **SAVECS**.

CSUR
CSMOVE
CSTIME

SAVECS values can be cleared via storing new **SAVECS** parameters or the **DEFEE** (Default System to System Settings) Command.

Returns: 0 = Axis Current Sense Parameters Unsaved
1 = Axis Current Sense Parameters Saved

Examples:

Send: `YSAVECS?`
Receive: `YD+000001.0`

The above example queries the status of the user set Current Sense parameters in Non-Volatile memory for the Y axis.

Recommendations: User set values should be saved when favorable adjustment has been achieved.

Related commands: **DEFCS**, **INITCS**

SAVELM - Save Soft Limit Parameters in Non-Volatile Memory

Syntax: `[n]SAVELM`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose soft limit parameters are to be stored. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command stores users Soft Limit parameters in non-volatile memory for the specified axis. Once you have configured the axis, issuing the **SAVELM** command will save the current condition, and restore it every time the system power is applied. The following commands are saved by **SAVELM**.

NSLIM
PSLIM

SAVELM values can be cleared via storing new **SAVELM** parameters or the **DEFEE** (Default System to System Settings) Command.

Returns: `NONE`

Examples: Send: `ZSAVELM`

The above example stores the Soft Limit parameters for the Z axis.

Recommendations: User set values should be saved when favorable adjustment has been achieved.

Related commands: **DEFCS**, **INITCS**, **RPOS**

SAVELM? - *Query Status of Soft Limit Parameters in Non-Volatile Memory*

Syntax: `[n]SAVELM?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose soft limit parameters is to be stored. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the status of the users Soft Limit parameters in non-volatile memory for the specified axis. The following commands are saved by **SAVELM**:

NSLIM
PSLIM

SAVELM values can be cleared via storing new **SAVELM** parameters, **DEFLM** or **DEFEE** (Default System to System Settings) Command.

Returns: **0 = Axis Current Sense Parameters Unsaved**
1 = Axis Current Sense Parameters Saved

Examples:

Send: `YSAVEML?`
Receive: `YD+000001.0`

The above example queries the status of your set Soft Limit parameters in Non-Volatile memory for the Y axis.

Recommendations: User set values should be saved when favorable adjustment has been achieved.

Related commands: **DEFLM**, **INITLM**, **RPOS**

SSAVEOM - Save PM500-K6 User Parameters to Non-Volatile Memory

Syntax: SSAVEOM

Parameters: No axis identifier is allowed with this command otherwise a command error will result.

Function: This command stores users PM500-K6 parameters in non-volatile memory. Once you have configured the PM500-K6, issuing the **SAVEOM** command will save the current condition, and restore it every time the system power is applied. The following commands are saved by **SAVEOM**:

OMDA	**OMPDA**
OMECE	**OMSA**
OMHA	**OMSPD**
OMMA	**MSPDA**
OMNAD	**OMA**
OMDEL	**OMDIR**
OMMULT	**OMRAE**
OMSCAN	**OMSTEP**

SAVEOM values can be cleared via storing new **SAVEOM** parameters, **DEFOM**, or **DEFEE** (Default System to System Settings) Command.

Returns: `NONE`

Examples: Send: `SAVEOM`

The above example saves the PM500-K6 user settings to non-volatile memory.

Recommendations: User set values should be saved when favorable adjustment has been achieved.

Related commands: **DEFOM**, **INITOM**

SAVEOM? - *Query Status of PM500-K6 User Parameters to Non-Volatile Memory*

Syntax: `SSAVEOM?`

Parameters: No axis identifier is allowed with this command otherwise a command error will result.

Function: This command queries the status of your PM500-K6 parameters in non-volatile memory. The following commands are saved by **SAVEOM**:

OMDA	**OMPDA**
OMECE	**OMSA**
OMHA	**OMSPD**
OMMA	**OMSPDA**
OMNAD	**OMA**
OMDEL	**OMDIR**
OMMULT	**OMRAE**
OMSCAN	**OMSTEP**

SAVEOM values can be cleared via storing new **SAVEOM** parameters, **DEFOM**, or **DEFEE** (Default System to System Settings) Command.

Returns: 0 = PM500-K6 Parameters Unsaved
1 = PM500-K6 Parameters Saved

Examples: Send: `SSAVEOM?`
Receive: `YD+000001.0`

The above example queries the status of the PM500-K6 user settings in non-volatile memory.

Recommendations: User set values should be saved when favorable adjustment has been achieved.

Related commands: **DEFOM**, **INITOM**

SCUM - System Communications Mode

Syntax: `SCUM <mode>`

Parameters: Defines the system communications mode. A space is allowable between the command and the mode specifier but is not required. If no mode specifier is added the command will be ignored.

The <mode> must be a numeric value within the specifiec range in a string format.

Function: The **S**ystem **C**omm**u**nications **M**ode (**SCUM**) command specifies whether the controller should operate in a manner compatible with the original 5-axis PM500 controller or whether the new, system-level motion and configuration commands and synchronization features (available only in units with SBC PROMs 11.14 and later) should be enabled. This parameter is used for changing the communication protocol to or from the standard or high-speed pipelining command mode. Adjusting this parameter changes the way the system behaves programmatically and responses to commands.

NOTE

6 axis controllers only operate in SCUM1 mode. 5 axis controllers can be configured to operate with either mode

Parameters: **0** = Standard Communications Mode (System default 5 axis controllers)
1 = Pipeling Mode (System default 6 axis controllers)

*See the Command Summary for details on the functions of each of the above mentioned modes.

Examples:

Send: `SCUM 1`
Receive: No response

The above example sets the system to pipeline command mode.

NOTE

When the SCUM 1 mode is invoked. Command structures and protocols change. All system level control commands must have an “S” prefix, otherwise a command error will result. Axis commands do not change, but acknowledgments and completion signaling must conform SCUM 1 to communications protocol.

Example:

Mode	*Command*	*Comments*
SCUM 0	RSTART	Restart command under normal COM mode.
SCUM 1	SRSTART	Restart command under SCUM 1 mode
However axis specific commands are unchanged under SCUM 1 mode.		
SCUM 0	XGR100	X axis Go Relative under standard COM mode.
SCUM 1	XGR100	X axis Go Relative under SCUM 1mode.

Recommendations: The factory default setting **SCUM 0** of this value will provide excellent performance under a broad range of applications. **SCUM 1** mode is only recommended for particular demanding applications. 5 axis controllers can operate in either **SCUM1** or 2 mode. 6 axis controllers only operate in **SCUM1** mode.

Related commands: All commands with the “S” prefix and all axis motion commands.

SETUP - Service Command - Servo Adjustment

Syntax: `[n]SETUP <option>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose axis is to be servo tuned. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

NOTE

This command is for use in conjunction with axis board tuning only.

Function: This command is for use during servo adjustment of the High resolution Axis boards *only*. Servo is not required in the normal maintenance for the PM500. Only in the event where the Troubleshooting section or Newport Technical support advises should servo adjustment be attempted.

Returns: `NONE`

Range:

SETUP0	Low speed slew
SETUP1	High speed slew
SETUP9	Scan limit-to-limit
SETUP10	Limits fiducial search
SETUP12	Random motions

Examples: *See* Servo Tuning Section.

Recommendations: Servo tuning is not required in the normal use and service of the PM500. Consult your troubleshooting guide on the advice of a Newport Technical ***BEFORE*** attempting servo adjustment.

Related commands: NONE

CAUTION

Setup command motions usurp the current limit and safety algorithms of the stage. Only use SETUP commands in conjunction with servu adjustment.

SIGN - Define Default Direction for Motion Coordinate System

Syntax: `[n]SIGN <option>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose sign is to be changed. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command defines the default direction for positive moves and velocity commands for the specified axis. The setting of this command will be automatically stored in non-volatile memory and will be present at power-up.

Options: –1 = Away from motor
+1 = Towards motor

Default: Positive direction moves towards motor.

Examples: Send: `XSIGN +1`

The above example sets the sign for the X axis to + 1, thus any positive move command will cause motion towards the stage motor.

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. See **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: NONE

SMP - System Move to Preset Absolute Positionc (For use in SCUM1 mode only)

Syntax: SMP

Parameters: This is a system level command. no axis identifier or parameter is allowed, or else a command error will be posted.

Function: This command triggers a preset absolute motion as defined by the DP command. This is a system level trigger for any preset absolute move loaded in any of the axes by the DP command. The execution of this command will cause all axes properly instructed by the DP command to move to their assigned absolute position. This command is the SCUM 1 version of the GP command. The GP "Go Preset" command only functions under SCUM 0 mode. Use the SMP command when the system is under SCUM1 mode.

Returns: `None`

Examples:

Send:	`XDP 150.00`
Send:	`SMP`
Receive:	`Nothing`

The above example preset the X axis for an absolute move to 150.00 the SMP command triggers the X absolute move to the preset position.

Recommendations: The method of presetting motions for axes and using the single trigger SMP provides better axis synchronization and faster program execution time due to the reduced instructions required to initialize moves for multiple axes.

Related Commands: **DP**, **DRP**, **SMRP**

SMPL - System Move Preset Logic (For use in SCUM1 mode only)

Syntax: `SMPL <$><hexadecimal value>`

Parameters: This is a system level command. No axis identifier is allowed, otherwise the command will not be executed and a command error will be posted. The parameter must be in a hexadecimal format and thus requires the value to be prepended with a dollar sign `($)`. A space is allowable between the command and the parameter but is not required.

NOTE

This command is for is only for systems operating under SCUM1 mode.

Function: This command performs the masking of the preset motion register. The preset motion register allows the exclusion or selective inclusion of axes when system level commands are initiated. The user can also configure this register to allow specific motion parameters to be defined for each axis, separate from the normal settings for that axis.

The **SMPL** Register:

Bit	Meaning	Comments
0	X axis	0: Disabled; this axis excluded from preset motions 1: Enabled; this axis included in preset motions
1	Y axis	0: Disabled; this axis excluded from preset motions 1: SMPL <$><hexadecimal value>
2	Z axis	0: Disabled; this axis excluded from preset motions 1: Enabled; this axis included in preset motions
3	A axis	0: Disabled; this axis excluded from preset motions 1: Enabled; this axis included in preset motions
4	B axis	0: Disabled; this axis excluded from preset motions 1: Enabled; this axis included in preset motions
5	C axis	0: Disabled; this axis excluded from preset motions 1: Enabled; this axis included in preset motions
12	NA	*Reserved =Ø*
13	NA	*Reserved =Ø*
14	NA	*Reserved =Ø*

15	Specify motion parameters preset	0: Disabled; each axis will use parameters as set by: **ACCEL**, **DECEL**, **V** 1: Enabled; User can specify parameters especially for move using: **PMACC**, **PMDEC**, **PMVEL**

Default: $FFF

Returns: NONE

Examples:

Send: `SMPL $FFF`
Receive: `Nothing`

The above example enables all axes for preset motion and disables bit 15; the specification of motion parameters.

Recommendations: The method of presetting motions for axes and using the single trigger SMRP command provides better axis synchronization and faster program execution time due to the reduced instructions required to initialize moves for multiple axes.

NOTE

The preset distance of Ø, will be loaded into the command register for all axes enabled via this register, but not specifically designated with a move distance. With relative moves, a relative distance command of Ø will cause no motion. However, when an absolute preset move is initiated all axes not directly designated with an absolute move distance will be commanded to move to Ø, which in some cases *will cause uncommanded axes to move*.

Related commands: **SMRP**, **DRP**, **MPACC**, **PMDEC**, **PMVEL**

SMRP - System Move to Relative Preset (For use in SCUM1 mode only)

Syntax: SMRP

Parameters: This is a system level command. No axis identifier or parameter is allowed, otherwise the command will not be executed and a command error will be posted.

NOTE

This command is for is only for systems operating under SCUM1 mode.

Function: This command triggers the preset relative motion as defined by the **DRP** command. This is the system-level trigger for any preset relative move loaded in any of the axes by the **DRP** command. The execution of this command will cause all axes properly instructed by the **DRP** command to move their assigned preset relative distances. The move will take place in the high speed loop if the distance of the move is larger than the setting of **LSIZE**. This command is the **SCUM 1** version of **GRP** command. The GP Go Relative Preset command only functions under **SCUM 0** mode. Use the **SMRP** command when the system is under **SCUM 1** mode.

Returns: NONE

Examples:

Send:	`XDRP 100.0`
Send:	`SMRP`
Receive:	`Nothing`

The above example presets a relative move for the X axis of 100.0 microns. The **SMRP** command triggers the system to execute the preset motion as defined by the **DRP** command. The **DRP** command must be issued for each axis to be moved. Different values can be defined for each axis.

Recommendations: The method of presetting motions for axes and using the single trigger **SMRP** command provides better axis synchronization and faster program execution time due to the reduced instructions required to initialize moves for multiple axes.

Related commands: **DRP**

SRQCTL - Define SRQ Serial Poll Mask

Syntax: (S)*SRQCTL <$><Hexadecimal bitmap value>

Parameters: This is a system level command. The system specifier "S" is required under SCUM1 else the command will not be executed and a command error will be posted. The parameter must be in a hexadecimal format and thus requires the value to be pre-pended with a dollar sign ($). A space is allowable between the command and the parameter but is not required.
**"S" system specifier required under SCUM1 mode.*

Function: This command allows the user to modify what events trigger a GPIB (SRQ) Service Request. This command is used for masking the Serial Poll Register. Adjusting this parameter allows the user to customize a command what will trigger an SRQ event. This is divided into to 2 areas:

What axis will be observed for events *(i.e., only installed axes)*
What system event will trigger an SRQ *(i.e., motion complete, message available)*

The PM500's Serial Poll register format is different under SCUM0 or 1 mode.

There are 2 Serial Poll Registers available-depending if System Level Communications (SCUM) Mode is enabled or disabled. Under SCUM1 you may select a new Serial Poll format that supports up to 12 motion axes and has a MAV bit sent via the **SENAINT** command.

Parameters: Serial Poll Register-

S Poll Bit	***Meaning***	***Comments***
0	Axis bitmap	Not maskable
1	Axis bitmap	Not maskable
2	Axis bitmap	Not maskable
3	Axis bitmap	Not maskable
4	MAV	Message available. Read message buffer
5	FAULT*	See bits 0-3 for specified axis in fault condition
6	SRQ	Not maskable
7	DONE/ACK*	Signals done when a specific axis is specified Signal command acknowledge when bits 0-3 are ON: system.

**When the unit requests service with both bits 5 (FAULT) and 7 (ACK) off, completion is indicated for the specified axes.*

Returns: Current **SRQCTL** mask setting

Range: In Hexadecimal format $0 - $FF

Default: $0

NOTE

You must use the proper form of this command compatible with the systems mode i.e., SSRQCTL$EF for SCUM1 mode. You may query the current system mode using the SCUM? query.

Examples:

Send: `SRQCTL$FF` **(SCUM 0)**
Receive: `Nothing`

The above example sets all axis events enabled, fault and done bits enabled. If your system did not have an A or B axis, bits 3 and 4 could be set to 0, disabling events from axes not present in the system.

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations:

SCUM 0 `SRQCTL$FF`
SCUM 1 `SSRQCTL$EF` (No MAV bit)

The SRQ system is disabled at power-on. Thus the Serial Poll register must be set at the start of your program or whenever the system is first powered-on or restarted.

The PM500 “stacks” it’s service requests and their corresponding serial poll status bytes. To clear the service request system, repetitively poll the controller until the return byte is zero. The MAU bit (16) can only be cleared by performing a GPIB read.

Related Commands: **ENAINT**, **COMOPT**

S/RSTART - Restart Entaire System

Syntax: `(S)*RSTART`

Parameters: No parameter or axis identifier is allowed with this command under **SCUM 0**. System specifier required under **SCUM 1** or otherwise a command error will result.
**"S" system specifier required under SCUM1 mode.*

Function: This command performs a system reset. This performs system reset. This Command has no response. All positions are lost, all defined parameters will be reset to their default values stored in non-volatile memory. You should expect a power on condition after issuing this command. If a sign on prompt is enabled in RS232 mode, a sign on will be issued. Commands which are reset to their condition stored in non-volatile memory:

ACCEL	**LACCEL**
DECEL	**LDECEL**
CSIZE	**LSIZE**
FV	**LV**
V	**NULL**
AZ values	

Returns: `NONE`

Examples: Send: `RSTART` (under SCUM0)
`SRSTART` (under SCUM1)

The above example restarts the entire system.

Recommendations: NONE

Related commands: **DEFLT**, **DEFEE**

STAT? - Read Axis Status

Syntax: `[n]STAT?`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose status is to be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command. The `"?"` is optional, the system will always return a response from the **STAT** command.

Function: This command queries the status of the specified axis. This command may be used at any time to determine the status of an axis. The response will be an axis specifier followed by any of the returns listed. This command allows you to access and diagnose system errors. Upon receiving an error response, you should always query the error status by executing the appropriate Error Status (**ESTAT**) command. Should **ESTAT** return with "Execute Error" you to proceed to Execute Error (**EESTAT**) command for a definition of the error.

Returns: `B` = Busy; axis currently in motion or actively servoing to position.
`D` = Done; has completed last command
`L` = Limit; axis is in limit
`E` = Error; an error has occurred
`M` = Motor OFF**Default:** `NONE`

Examples:

Send: `XSTAT?`
Receive: `XD`

The above example requests the status of the X axis which reports `"Done".`

Recommendations: This command is most commonly used during RS-232 communication which doesn't possess the sophisticated error/response handling of GPIB.

Related commands: **ESTAT, EESTAT**

S/VN - Report System Firmware Revision

Syntax: `(S)*VN`

Parameters: This is a system query, no axis identifier is allowed otherwise a command error will result. No parameter is allowed with this command.
**"S" system specifier required under SCUM1 mode.*

The "**?**" is optional, the system will always return a response from the **SVN** command.

Function: This command queries the firmware revision number from the system controller. The return will be in hexadecimal format. System firmware revision version is required when upgrading a system or installing additional axes.

Returns: Current system firmware revision

Examples:

Send: `SVN?`
Receive: `XD$11206120`

The above example queries the firmware revision for the system controller. Note that the first to character/s will be the last commanded axis and its status character, if enabled.

NOTE

The VN? is recognized by the system in either SCUM0 or SCUM 1 mode.

Recommendations: When inquiring to the factory about upgrades or adding additional axes to your system, be prepared to provide the **VN** information for your system.

Related commands: NONE.

S/WTRIG - Activate Wait Trigger Function

Syntax: `(S)*WTRIG`

Parameters: No axis identifier or parameter is allowed with is command. This command is supported by GPIB/IEEE interfaces only.
**"S" system specifier required under SCUM1 mode.*

Function: Wait for group execute trigger command. This command causes the commands which follow **WTRIG** to hold off execution until the hardware group execute trigger is received. This command functions for GPIB IEEE only. This function is useful when synchronization is required.

The trigger for **WTRIG** is a function of your computers GPIB board/controller. Refer to the trigger function in its documentation for how to implement its trigger function.

After the trigger function has executed and motion is under way, normal service requests and **STAT** responses may be used to determine the completion of each axis move.

Caution: Do not use multiple read commands inside this construct. The second read will destroy the first reads response.

Limits: No more than 80 characters are allowed after **WTRIG**, or else a buffer overflow will occur.

Returns: `NONE`

Example: Using National Instruments AT-GPIB board and IBIC communications software.

Send: `WTRIG;XGR1200;YGR500`
Send: `IBTRG` National's board trigger function

Motion should execute per the commands specified.

NOTE

After WTRIG all following commands, whether concatenated or not, will be stored for execution until the trigger function is executed. This includes reads and STAT queries.

Recommendations: This function can be used to synchronize motion and other instruments via the GPIB interface.

Related commands: None. The trigger for this PM500 function is a feature of your computer's GPIB card.

T - Transfer Position

Syntax: `[n]T`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose position is to be transferred from the position register to the command register. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command transfers the current position from the position register to the command register, causing the motor to servo to the current position. If issued during motion the axis would servo where the "**T**" command was issued. If during a motor off "**M**" condition the axis will servo at the current position, even if it was arrived at manually.

This command is especially useful to stop motion during a scan via **S**, **LS**, or **ULS** or return to motor to the "ON" condition after the "**M**" (motor OFF) has been issued.

Returns: `NONE`

Default: `NONE`

Examples:

Send: `ZM` – *(Turn Z motor OFF.)*
Move stage using manual knob.
Send: `ZT`

The above example turns the Z motor OFF allowing manual movement. The "**T**" command reads the current stage position and servos to that position.

Recommendations: The "**T**" command should be used to stop scans or the turns motor ON after the "**M**" motor OFF command.

Related commands: **M**, **S**, **LS**, **ULS**

ULS - *Ultra-Low-Speed Scan*

Syntax: `[n]ULS <s-velocity>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B,` C which is to be commanded to commence an ultra-low-speed scan. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command initiates an ultra-low-speed scan (sub- µm/sec) of the specified axis for the defined velocity. The scan will be executed in the Ultra-Low Speed loop.

Options: Velocity in: µm/sec for linear
arc-sec/sec for rotary

Range: The following are maximum values for **ULS**
1000 mms/sec for PM500-A1 actuators
4000 mms/sec for PM500 Stages
100 arc-sec/sec for Rotary stages

Default: `NONE`

Examples: Send: `XULS 100`

The above example commences a ultra-low-speed scan of the X axis at 100 mm/sec.

Recommendations: To stop a **ULS** scan use the "**T**" (Transfer Position) or "**M**" (Motor Off) commands. The direction of the scan can be defined by the sign or the velocity parameter +1-.

Related commands: **ULV**, **T**

ULS? - Ultra-Low-Speed Scan Query

Syntax: [n]ULS?

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` which ultra-low-speed scan is to be queried. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. No space is allowed between the axis identifier `[n]` and the command.

Function: This command queries the velocity of the ultra-low-speed scan defined by **ULS** for the specified axis. The return will be the defined **ULS** velocity and not the actual velocity. **ULS?** cannot report velocity on-the-fly.

Returns: Velocity in: µm/sec for linear
arc-sec/sec for rotary

Examples: Send: `XULS?`
Receive: `XD+000050.0`

The above example queries the ultra-low-speed scan velocity setting of **ULS** for the X axis.

Recommendations: To stop a **ULS** scan use the “**T**” (Transfer Position) or “**M**” (Motor Off) commands.

Related commands: **ULV**, **T**

ULV - *Define Ultra-Low-Speed Scan Loop Velocity*

Syntax: `[n]ULV <s-velocity>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose ultra-low-speed scan loop velocity is to be defined. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command defines the Ultra-Low-Speed Loop threshold for the specified axis in µm/sec or arc-sec/sec. This parameter is used for defining the threshold point where velocities equal to or less than the ULV value will be executed in the Ultra-Low Speed Loop. The PM500 has two Scan Loops which improve scan velocity smoothness by restricting slow scans (sub µm/sec) to take place at Ultra-Low speed Loop and vice versa for faster scans. This enables you to execute scans of various velocities without redefining velocity for each scan. The ULV is the demarcation between these regimes.

Options: Velocity in: µm/sec for linear
arc-sec/sec for rotary

Range: The following are maximums values for **ULV**
1000 mm/sec for PM500-A1 actuators
4000 mm/sec for PM500 Stages
100 arc-sec/sec for Rotary stages

Default: `NONE`

Examples: Send: `XULV 500`

The above example defines an ultra-low-speed scan velocity threshold to 500 mm/sec. for the X axis.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

Related commands: **ULS**, **T**

V - Define Velocity for High Speed Loop

Syntax: `[n]V <velocity>`

Parameters: `[n]` defines the axis identifier, `n = X, Y, Z, A, B, C` whose High Speed loop velocity is to be modified. If the axis identifier `[n]` is omitted the command will not be executed and a command error will be posted. A space is allowable between the command and the parameter but is not required. No space is allowed between the axis identifier `[n]` and the command.

Function: This command defines the maximum High Speed loop velocity for the specified axis in mm/sec^2 or k arc-sec/sec^2. This parameter only affects moves greater than **LSIZE**. Actual velocity is somewhat dependent upon system load; no absolute calibration is implied in this parameter.

Range: Any valid velocity value in µm or arc-sec

Returns: `NONE`

Default: System Dependent

Examples: Send: `YV 200`

The above example sets the Y axis velocity to 200 arc-sec/sec

User set values can be stored in non-volatile memory using the `[n]SAVEAX` command. User set **SAVEAX** values are restored after system power-up. *See* **SAVEAX** command for details.

Recommendations: The factory default setting of this value will provide excellent performance under a broad range of loads and velocities. *See* the System Settings section in this manual for factory settings.

When the users wishes to increase move speeds, acceleration (**ACCEL**) and deceleration (**DECEL**) should also be adjusted with **V** for the best step and settle performance.

Related commands: **LSIZE**, **ACCEL**, **DECEL**

This page is intentionally left blank

17 Functional Description & Specifications

17.1 Outline of Section 17

- PM500 Controller
- Controller board
- Axis boards
- Motion devices

This section describes the basic functional elements of the controller and motion devices, and provides an explanation of their operation. It has been included to aid you in locating specific components, and to gain some understanding of basic components of the motion system.

17.2 Controller

The PM500 controller is modular in design. Each axis is powered by an axis board. A controller will accommodate up to six axis boards. A controller board coordinates axis functions and controls the external interface communication.

For aid in locating specific controller components, a top view of the PM500 is shown in Figure 41. The rear panel is shown in Figure 42.

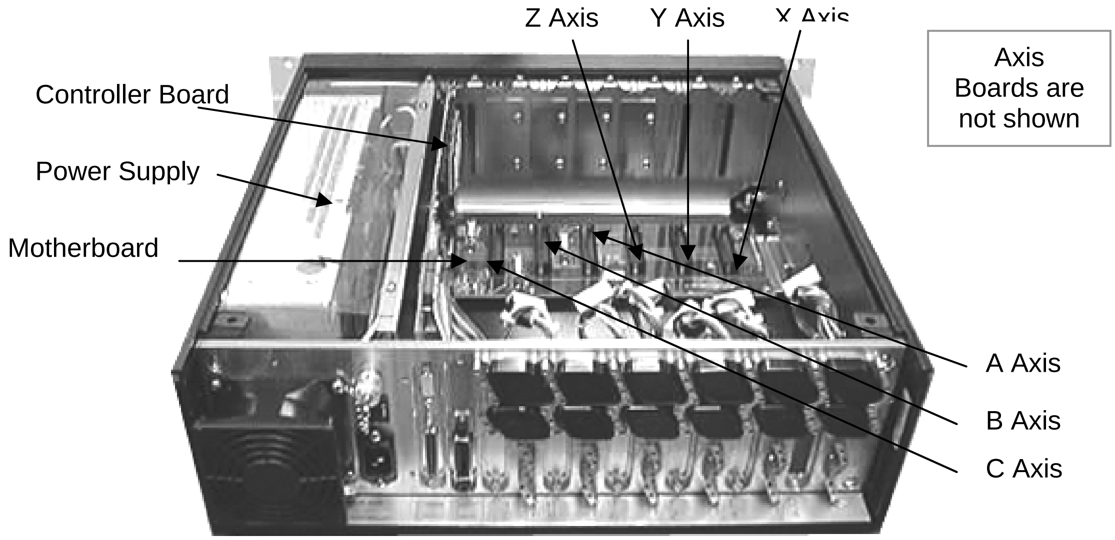


Figure 41: Top view of Model PM500 Controller.

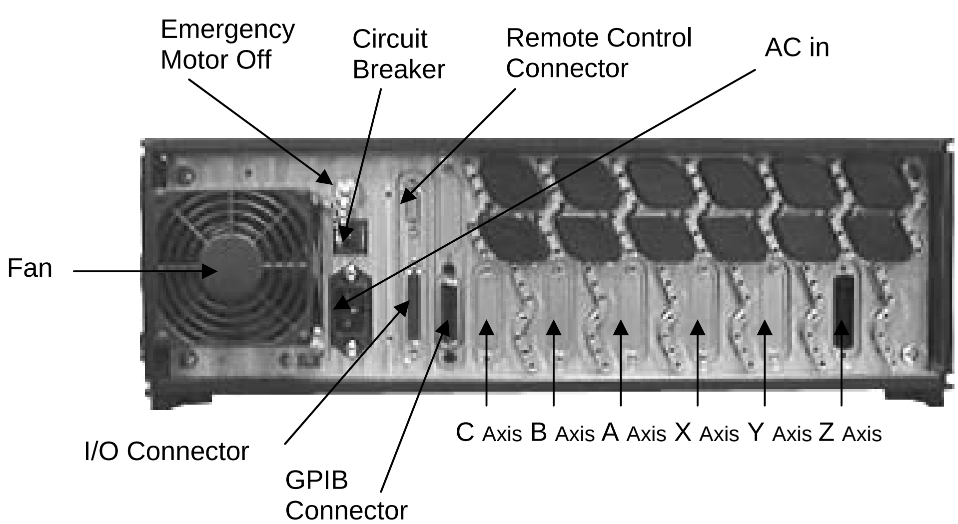


Figure 42: Rear view of Model PM500 Controller.

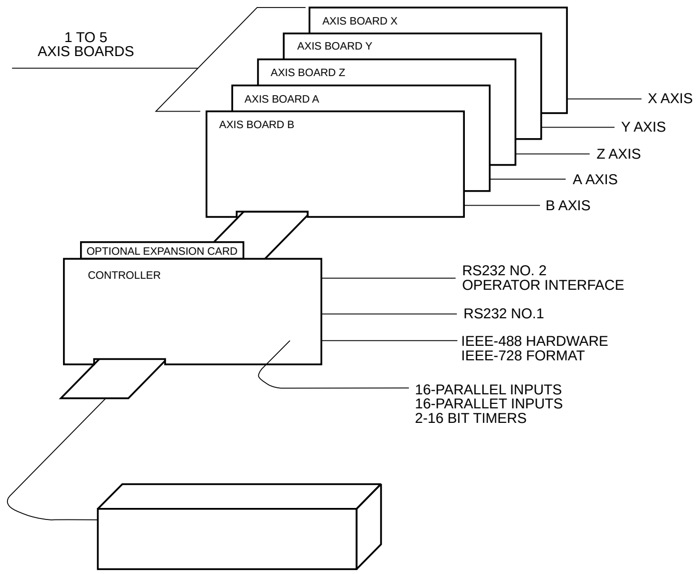


Figure 43: PM500 Controller functional elements.

Figure 43 depicts the basic functional elements of the PM500 controller. Each axis board is a microprocessor-controlled DC-servo positioning system optimized for precise point-to-point positioning control. The function of each axis board is to control the mechanical portion of each axis of motion.

The controller board interfaces and coordinates the axis board(s), and communicates with you via a high-level interface such as RS-232 or GPIB. Frequently used parameters are stored in the controller board's non-volatile memory. The controller board also allows direct input through an optional Remote Control Console, Model PM500-K. This device allows you to initiate moves and read out positions by means of keyswitches.

A switching regulator power supply provides all necessary power requirements. Over-power protection is provided by internal circuitry and fuses.

17.2.1 Power supply/motherboard

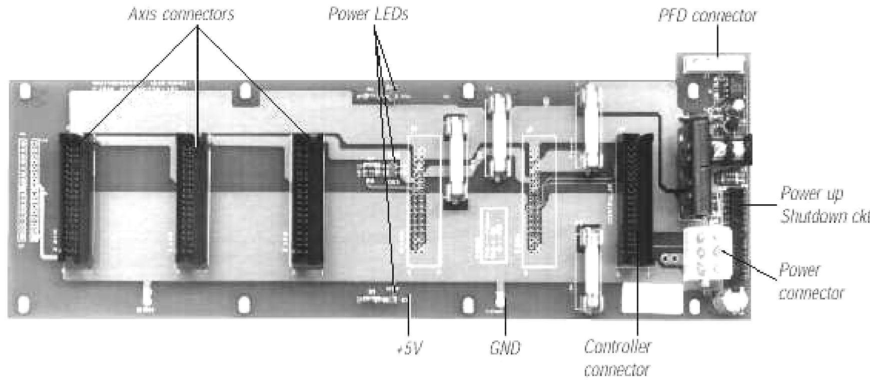


Figure 44: Motherboard.

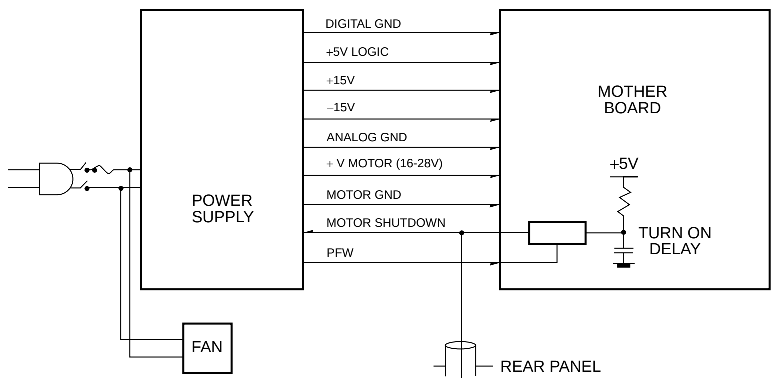


Figure 45: Power supply/Motherboard functional elements.

The power supply is a switching regulator with a maximum output capacity of 500 watts. It has the following features:

- 4 output voltages, +5V, +15V, –15V and VMOTOR(28V)
- Switchable line input capability of 90–132 or 180–264 VAC, 47–440 Hz
- Built-in line filtering, meets FCC and VDE 0871 Level A standard
- 3750 VAC Safety Isolation (4 mm spacing primary to ground, 8 mm spacing primary to secondary)
- Built-in overload protection on all outputs
- Built-in overvoltage protection on main output (+5V)
- The motherboard has fuses on each supply input.

All boards connected to the motherboard receive their power from it. For troubleshooting, each input voltage to the motherboard is indicated with a green LED and protected by a fuse. In addition, there is a +5V test point and a ground test point for servicing. Any loss of power can be isolated by removing the PC boards one by one until the power supply light comes on again.

A motor voltage shutdown circuit ensures that the motor power remains off until the logic supply is present and the microprocessors are running. This circuit ensures that the motor supply comes on only when the +5V is present, and goes off when the line voltage goes below a level that will cause the +5V to go down.

17.3 Controller Board

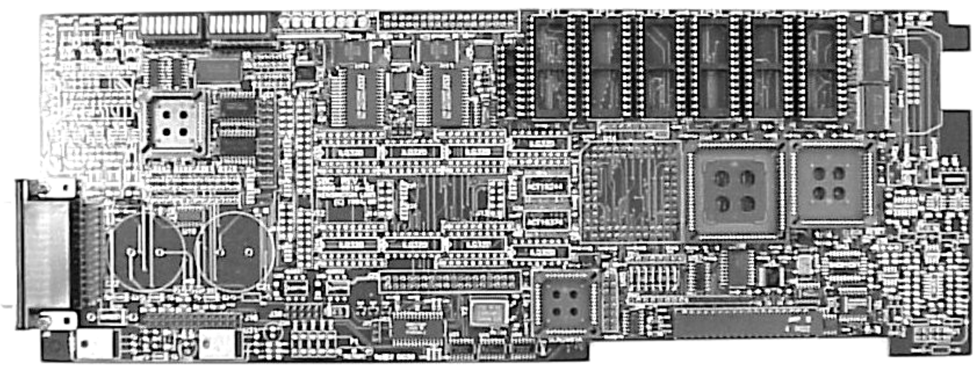


Figure 46: Controller board.

The controller board is a specialized, highly integrated, 16-bit microcomputer that provides the following functions for the system:

- General purpose I/O
- Non-volatile motion parameter storage
- High-level command conversion to axis-level command
- System multi-axis coordination
- Service diagnostics
- System reset

The types of I/O available for the system are:

- RS-232C
- GPIB (IEEE-488)
- PM500-K dedicated serial I/O

The CPU is a 16-bit NMOS microprocessor (80186). Timer functions, interrupt control, and memory control are provided internally. Main program storage is contained in ROM. Volatile variable storage is provided by RAM. Nonvolatile storage for axis and system parameters is provided by PROM. Status LEDs provide system diagnostic information. DIP switches provide system communication configuration control.

The RS-232 port and PM500-K port are handled via a dual UART chip. One port of this interface goes through J2. The other port goes through J1 to the PM500-K connector. JU6 connects pin 20 (the Data Terminal Ready) to the RS-232 interface.

GPIB is performed with a 9914A controller chip and buffered via a 75160 and 75161A. A ribbon cable to the rear panel is connected between J5 and standard IEEE-488 connector on the rear panel. (***Note:*** This product represents one standard bus load to the bus.)

The factory-set GPIB address is 1, or as recorded in the system settings (Appendix Z) provided with the controller.

A thermal overload switch (normally closed) input is connected via J8 pins 1 and 2. When these input pins are open, the controller causes the motor off status "**M**" to report when I/O commands are issued to the controller. The motor supplies are locally shut off. When the pins are closed, the next command that activates the motor will restore motor power. (***Note:*** Not all PM500 controllers have the thermal overload switch, but in all cases a jumper or a switch needs to be present on J8.)

A note about controller- and axis-board PROMs

New controller features are constantly being developed. In many cases, these new features can be added to existing PM500 systems via a straightforward PROM upgrade in the field at a nominal cost. Consult with Newport on PROM upgrades; be sure the entire system is upgraded as a unit. When adding new motion devices to an existing system, it is often necessary that the PROMs on the CPU and existing axis boards be upgraded to the same version level as the new device's PROMs. Consult with Newport whenever you add a motion device to an existing system.

17.4 Axis Boards

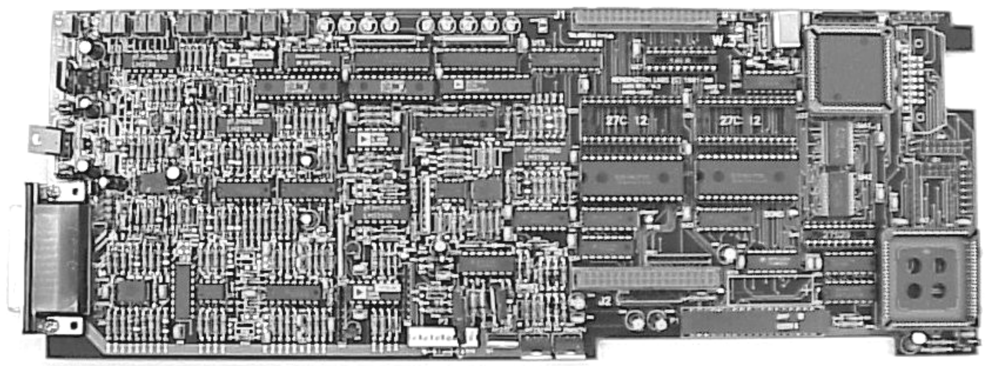


Figure 47: High Resolution Axis Board.

Each axis board contains feedback elements and driving power to control a motor/tach/position encoder system. Axis boards and cables are normally included with each motion device and need not be purchased separately. A block diagram is shown in Figure 48.

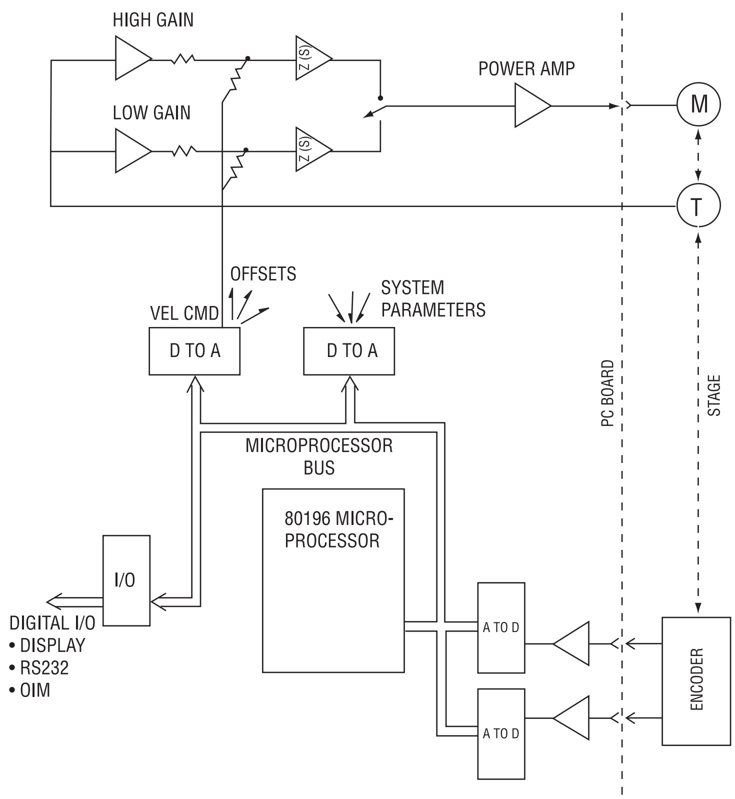


Figure 48: Axis board schematic.

NOTE

One axis board is required for each motion device. Since many PM500 motion devices are electrically matched to their specific axis card, it is recommended that you refer to the system setting sheet when connecting cables. Rotary stage and actuator cards are not interchangeable without readjustment. Additionally, linear axis boards can never be interchanged with rotary or actuator axis boards. Late-model (1989 and later) linear boards cannot be interchanged with earlier boards.

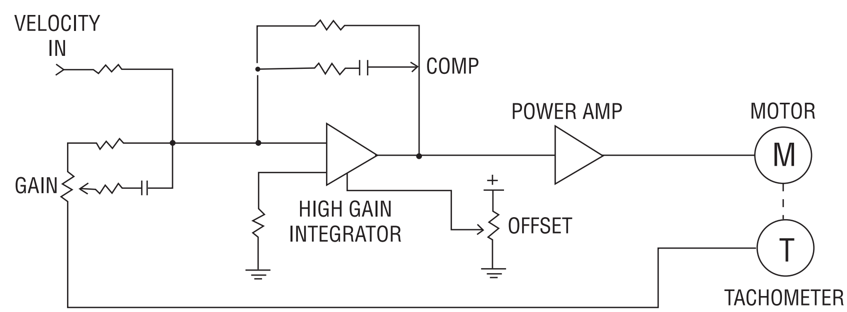


Figure 49: Schematic of axis board velocity loop.

The axis board consists of two nested, closed-loop feedback systems (a velocity loop inside a position loop).

The velocity loop consists of a closed system between the tachometer and the motor. The velocity loop input voltage produces a controlled velocity using the tachometer for feedback. A block diagram of this subsystem is shown in Figure 49.

For a given velocity command voltage on the axis board, the motor will turn fast enough to cause the tachometer to produce a scaled equivalent voltage into the summing amp. The TACH GAIN pot scales the speed versus the velocity command function. Phase differences occur between the motor and tachometer as a function of frequency. The COMP pot provides adjustment of the gain at higher frequencies. Adjustable compensation permits optimum matching of the controls to the motion system.

Two velocity loops accommodate a wide velocity range. The low speed loop has an extra amplifier to raise the tach output for precise velocity control at low velocities. The LOW TACH AMP BAL pot provides for the initial balancing of this high gain amplifier. The axis microprocessor does the final balancing of both loops when an Auto-Zero (**AZ**) command is issued to the axis.

Position information from the encoder system is used to close the position loop around the velocity loop, *see* Figure 50. The distance from the present position to the commanded position is computed and the appropriate velocity command is issued to move the stage to the commanded position.

The axis parameters such as acceleration and maximum velocity are utilized in generating the velocity command. Each point-to-point move is done in only one velocity loop as determined by the parameters and the distance to be moved. The low speed loop is used while the stage is stationary.

The power amplifier provides power to drive the motor (typically 20 to 100 watts.) The amplifier limits the current to the motor. The PM500 current limit is programmable, with a maximum of approximately 3.5 amps.

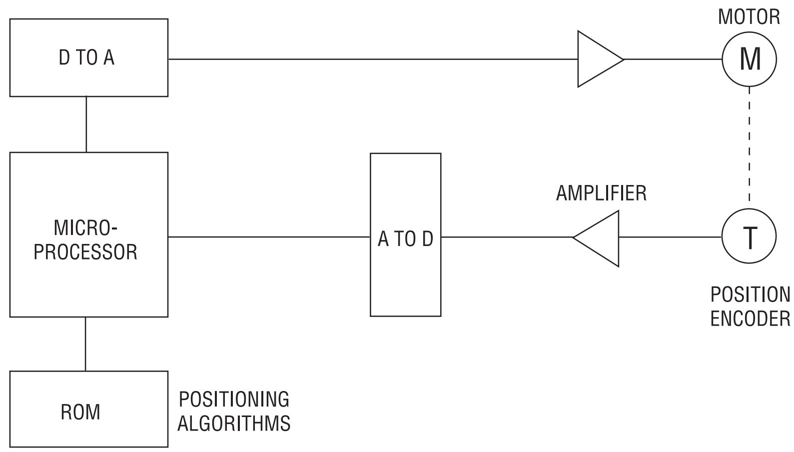


Figure 50: Schematic of axis board position loop.

During a point-to-point move, a proprietary algorithm generates velocity commands to accelerate, run at a constant velocity and then decelerate to a stop. As the stage approaches the endpoint, the microprocessor smoothly switches from velocity control mode to position control mode for holding the endpoint position. The dynamics of compensation and gain for the position control loop are all contained in the PROM software algorithms and commands.

The axis board communicates with the controller board to allow you to access its commands and registers. This is performed by the following software protocol:

- Controller board receives command from user to read position
- Controller board sends Read Position command to axis board
- Axis board returns position to controller board in binary
- Controller board returns ASCII position to the user

The relationship between the axis specifier (or axis name) and the physical axis location is fixed. Example: If the axis board in the X axis location is moved to the Y axis location, commands issued to the same board must have the Y axis specifier. By convention, linear axis boards are usually located in slots X, Y, or Z; rotary axis boards are located in slots A or B. Since PM500 motion devices are electrically matched to their specific axis card, it is recommended that you refer to the system settings (Section A.12) when connecting cables to keep the motion devices connected to the axis cards to which they are matched. Axis cards for recent (1989 or later) linear stages are interchangeable since the stages incorporate normalizing electronics. Rotary stage and actuator cards are not interchangeable without readjustment. Additionally, linear axis boards can never be interchanged with rotary or actuator axis boards. Late-model (1989 and later) linear boards cannot be interchanged with earlier boards.

Controller specifications

Power:	Switchable 90-132 VAC/180-264 VAC 47–440Hz 300 VA max. Supply meets FCC and VDE level A requirements; includes AC line filter. Forced-air cooling.
Axis Connection:	25-pin D subminiature female (up to six axes/cabinet) connection containing quadrature sinusoidal inputs from encoder, optical Limit switch inputs, motor/tach connections, encoder power.
Interfaces:	RS-232 25-pin D subminiature, IEEE-488-Blue Ribbon standard (adheres strictly to IEEE-488.1 standard.)

17.5 Motion Devices

Newport offers a wide variety of precision linear and rotary stages for use with PM500 Controllers. Each stage incorporates a DC motor, DC tachometer, and a position encoder.

Position and direction of motion information is developed by the read head, amplified, and standardized by electronics within the PM500 motion device. Two analog signals, called sine and cosine, are developed in phase quadrature from phototransistors which receive light from LEDs through the glass scale and reticle, *see* Figure 51. The light is modulated as a function of position along the scale as a result of interaction between the lines on the scale and similar phased sets of lines on the reticle. One cycle of these signals occurs when the stage has moved one cycle on the encoder. A typical PM500 linear stage has a 25-line-per-mm glass scale, thus one cycle occurs for each 40 microns of relative motion between the scale and the reticle. The microprocessor keeps track of position by monitoring these signals.

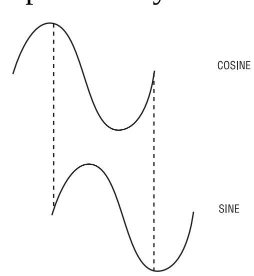


Figure 51: Axis board sine/cosine signals.

Power to the read head LEDs is provided through a current-limiting resistor from +5 volts. Power for the amplifier board in the stage comes from + and –15 volts. Reference voltages for the read head are generated at the stage (+VREF and –VREF in the figure). The axis board has a nominal encoder signal gain of 2. The MAG and BAL pots are used to standardize the axis board. The encoder signals should be + or –4.1 volts around ground at TP2 and TP3.

Figure 52 is a schematic version of what's contained in the read head and the receiving electronics for one channel (either sine or cosine).

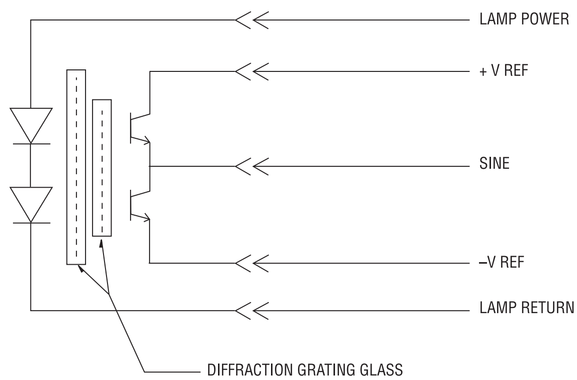


Figure 52: Read head.

There are similar circuits for the fiducial home and limits. The glass scale has a separate track which may be used to relocate (after a system power down) a home position of the stage. The fiducial track is opaque for one half of the stage travel and clear for the other half of travel. The fiducial signal from the read head is approximately 4.5 volts (input to R1). The "FØ" command will send the axis from the present location to the center of travel (transition from opaque to clear.)

The read head has two limit detectors which are used to limit stage travel, *see* Figure 53. The LEDs illuminate phototransistors through a clear track at the top of the scale. The limit signal is received by a comparator and sent to the microprocessor and power amplifier disable circuitry. Moveable tabs are used on the linear stage encoders to block the light to signal a limit condition. The tab is relieved to provide some overtravel without loss of position information. Actuator limits are fixed.

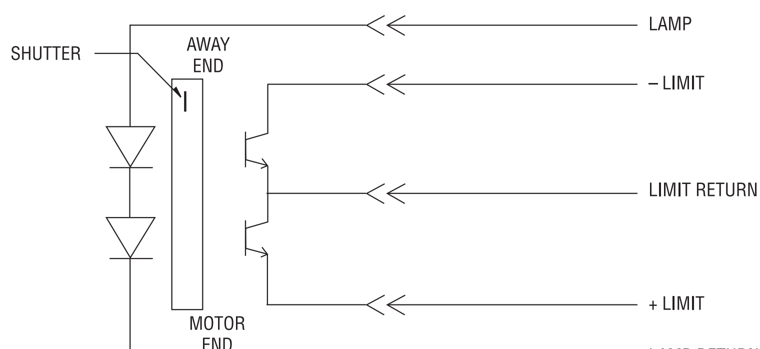


Figure 53: Motion device limit circuit.

17.5.1 Motion device specifications

Linear stages and actuators

Resolution	0.1 µm, 0.05µm or 0.025 µm
Minimum incremental motion	0.1 µm, 0.05µm or 0.025 µm
Bi-directional repeatability	0.1 µm, 0.05µm or 0.025 µm
Actuator travel	25 mm; fits dozens of Newport components
Stage travel ranges	2, 4, 6, or 8 inches standard; up to 12 inches available
Straightness and flatness of travel	PM500-8, <1.25 µm PM500-4, <0.75 µm; PM500-6, <1 µm;
X-Y stage orthogonality	<5 arc-sec
Material and finish	7075-T6 aluminum alloy with sealed Type III black anodized finish
Options	Side-mounted motor, left-hand drive

Rotary stages

Resolution	1 arc-sec (optional 0.5 arc-sec)
Minimum incremental motion	1 arc-sec (optional 0.5 arc-sec)
Bi-directional repeatability	1 arc-sec (optional 0.5 arc-sec)
Radial runout	1.25 µm
Axial runout	1.25 µm
Axis wobble	3 arc-sec
Material and finish	7075-T6 aluminum alloy with sealed Type III black anodized finish

18 Maintenance

	WARNING **115/230 volts AC and 28 volts DC are used in this instrument. Maintenance and servicing should be performed by qualified service personnel only. Some of the maintenance and servicing operations described here are performed with power supplied to the instrument while protective covers are removed. Exercise extreme care when performing these operations. Line voltage is always present on some terminals including the power input connector, fuse holder, power switch and other points. Energy available at many points may result in personal injury or death when contacted.**

18.1 Outline of Section 18

To assure the intended performance of the PM500, specific routines and scheduled procedures are necessary. This section covers these requirements:

- Cleaning
- Lubrication
- Scheduled calibration
- Adjustments

Please see the Service section (Section 19) for information on other procedures and information needed to support use of the product such as:

- Adding axes
- Changing axis assignments
- Troubleshooting
- Adjustments not routinely required
- Axis board tuning
- Upgrading system firmware
- Preparing motion devices for cleanroom use

The PM500 motion system was designed for high-duty-cycle production use. Most of the mechanics require no maintenance and are self-lubricating. Servo-adjustment or other board level adjustments are never required under normal use or wear.

If the system begins to behave improperly, and the restoration of software settings does not return the system to its high performance level, this is usually an indication of a failure or other problem. Follow the troubleshooting section guide for remedies. Contact Newport before attempting any servo adjustments.

18.2 Auto-Zero Self Calibration

For ultra-repeatable sub-micron positioning, the controller has an automatic capability to adjust servo-loops to compensate for minor variation due to aging, environmental differences, vibration, and handling. This automatic compensation capability is called Auto-Zero.

For optimum performance—particularly in very low-speed operations—we recommend performing the Auto-Zero routine after a warm-up period of one hour. Auto-Zeroing should also be done when sub-optimum performance is observed; it can quickly fix the problem in most cases.

When the Auto-Zero command is issued to an axis, the controller halts all motion and enters a monitoring routine which rebalances (re-zeroes) the servo-loop circuitry. The process is analogous to measuring and compensating for the dark current of a photo detector.

Multiple axes may be Auto-Zeroed simultaneously. The process takes less than 45 seconds. Do not touch or disturb the motion devices during the Auto-Zero process. If you're using a computer, the results of the Auto-Zero process may be stored in non-volatile memory by executing the `<axis> SAVEAX` command for each axis and will be reused every time the system is restarted.

Auto-Zero may be invoked from the PM500-K Remote Control Console or via the computer interfaces. *See* Section 14 - Command Reference and PM500-K sections in this manual for more information on issuing the Auto-Zero command.

18.3 Stages and Actuators- Maintenance

The Auto-Zero (**AZ**) command should be issued daily after warm-up to keep the servo system properly fine-tuned and providing specified performance.

Periodic execution of the **PM** (preventive maintenance) command is all that is recommended to keep your PM500 motion devices in mechanical top condition. This feature should be used periodically to maintain specified performance and to prolong the operating life of the linear translation stages. The procedure involves issuing the **PM** command to cycle the stage between its limits.

The **PM** feature does several important things:

- Re-distributes lubricant along the bearings and leadscrew
- Centers the bearing cages
- Auto-Zeroes the servo drive electronics
- Corrects axis stalling due to over-current limits caused by roller cage migration.

The preventive maintenance command (**PM**) includes a full set of auxiliary commands which allow you to custom-configure the preventative maintenance operation. The operating parameters can each be stored in non-volatile memory. The preventive maintenance configuration commands are found in the Command Reference section (Section 14) and are identified by the “**PM**” prefix.

The following environmental considerations should be made.

- The device should be kept free of dirt and contamination which might impair performance.
- A clean environment will help maintain proper performance.
- For your personal safety, turn off the PM500 Controller when servicing motion devices.

18.4 Motors- Maintenance

PM500 Motors are lubricated and sealed for life. No maintenance or lubrication is required. Occasional end-to-end rapid travel is sufficient to keep the motor brushes clean. Absolutely no adjustments should be made to the motor under ***any*** circumstances.

18.5 Leadscrew- Maintenance

Do not oil the leadscrew. Most applications require no periodic maintenance of the leadscrew.

Each PM500 device is shipped with the proper lubricants applied to the leadscrew and bearings. Leadscrews are self lubricating. Occasional end-to-end rapid travel is sufficient to keep the grease evenly distributed. Absolutely no adjustments should be made to the leadscrew under ***any*** circumstances.

18.6 Crossed-Roller Bearings- Maintenance

No periodic lubrication is recommended for the crossed-roller bearings.

Newport will supply special oil should re-lubrication be advised by Newport technical representatives. Do not use other oils.

Paraffin and other components of commonly available oils leave deposits which adversely affect the quality of motion. Absolutely no adjustments should be made to the crossed-roller bearings under ***any*** circumstances.

18.7 Encoders- Maintenance

Most applications require no periodic maintenance of the encoder. The encoder may need cleaning in very dirty environments.

- Use a lint-free swab or wipe dampened with alcohol or acetone to gently wipe the glass scale. Avoid using excessive solvent—it may leave a film.
- Wipe scale with a soft lens tissue to remove dust or finger-prints.

Absolutely no adjustments should be made to the encoder assembly under ***any*** circumstances.

18.8 Cables- Maintenance

Do not operate the system without all cable screws tightened. Both screws on each end of the stage cable must be securely fastened for reliable operation.

- Avoid sharp bends in the cable.
- Cables should be replaced if worn, cut, or severely crushed.
- Extending the cables is not recommended. Contact Newport for special cable length requirements.
- Absolutely no modification of axis cables should be attempted.

18.9 Controller- Maintenance

The controller is dependent upon the fan for adequate cooling. You should determine the inspection period, considering environmental conditions.

- To maintain proper cooling: always keep the controller cover in place.
- Turn off the controller to inspect and/or clean the air inlet.

No other controller periodic maintenance is necessary.

18.10 Maintenance Schedule

The following is the recommended maintenance schedule for your PM500 System in relation to usage. Refer to the appropriate Maintenance or Command Reference section for details on the processes recommended below.

	Normal Use ***R&D and scanning small loads***	**High-Duty-Cycle Use** ***Production and manufacturing***
Weekly	Issue **AZ** (Auto-Zero) to all axes after warm-up, followed by **SAVEAX** command	Issue **AZ** (Auto-Zero) to all axes after warm-up, followed by **SAVEAX** command
Monthly		Issue **PM** (preventive maintenance) command
Biannually	Issue **PM** (preventive maintenance) command	*Check:* • Encoders for contamination • Controller ventilation • Cables for wear, looseness
Yearly	*Check:* • Encoders for contamination • Controller ventilation • Cables for wear, looseness	Inspect stage bearings. Consult Newport for proper lubricant and recommended procedure

This page is intentionally left blank

19 Service

19.1 Outline of Section 19

The following topics are covered in this section:

- Changing axis assignments
- Adding axes
- Upgrading system firmware
- Preparing motion devices for cleanroom use
- Troubleshooting procedures and correcting action

For information on cleaning, lubrication, scheduled calibrations and adjustments, *see* the Maintenance section (Section 18).

WARNING

115/230 volts AC and 28 volts DC are used in this instrument. Maintenance and servicing should be performed by qualified service personnel only. Some of the maintenance and servicing operations described here are performed with power supplied to the instrument while protective covers are removed. Exercise extreme care when performing these operations. Line voltage is always present on some terminals including the power input connector, fuse holder, power switch and other points. Energy available at many points may result in personal injury or death when contacted.

19.2 Removing and Installing Axis Boards

This section outlines procedures that allow you to remove and reinstall axis boards to:

- Change axis assignments
- Install additional axes
- Upgrade axis board firmware

The modular design of the PM500 makes it easy for qualified individuals to do field upgrading in certain instances.

WARNING

Hazardous voltages exist within the PM500. Internal upgrades and servicing should be performed by qualified individuals only.

The PM500 contains static-sensitive devices. Exercise appropriate cautions when handling PM500 boards, cables and other internal components.

Do not install anything into your PM500 Controller except items provided by Newport specifically for installation in the PM500.

What you'll need

- Phillips screwdriver
- 3/16-inch nut driver
- Computer interfaced to PM500 via RS-232 or GPIB

Also check that you have the following:

- Original owner's manual for the system*
- Verify that the information on the system setting sheet (Appendix Z) is accurate for your particular system.*

* If you don't have this documentation or information, record the (i) axis card serial number, (ii) matching stage serial numbers, and (iii) axis port assignments as currently installed so that you can replace the system axis cards correctly.

19.2.1 Axis card removal

1. Turn off and unplug the controller.
2. Detach all axis and control cables.
3. Place the controller in a static-safe area.
4. Remove the four Phillips-head screws on the controller cover.
5. To remove the cover panel, tip rear of cover up, and pull towards controller rear.

WARNING

Touch the controller chassis before coming in contact with internal components to protect them from static discharge.

6. Remove the axis board retaining bar located along the inside front of the controller. *See* Figure 54.

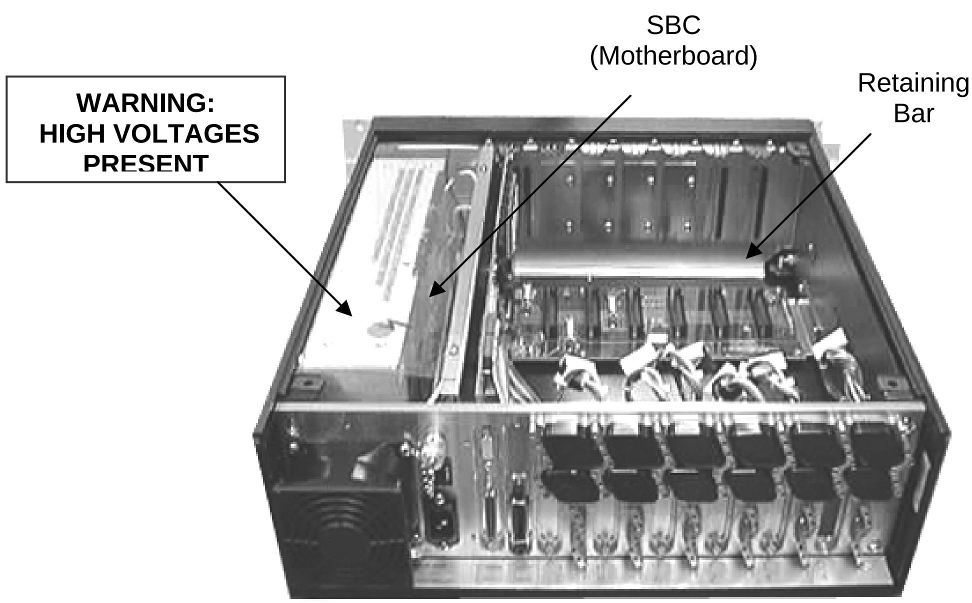


Figure 54: The retaining bar removal from the controller.

7. Remove the cable nuts for the axis to be removed. Cable nuts are located directly above and below the axis connector. *See* Figure 55.

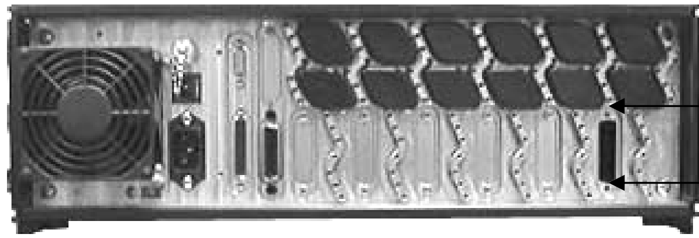


Figure 55: Cable nuts removal.

8. To remove the axis card, grasp the axis card near the center of the board firmly and pull upwards ***just until the axis card is disconnected from the controller bus.***

9. Unplug the wire harness (red connector) from the axis card and remove card from controller.

10. Repeat steps 6 through 8 for all remaining axis cards to be removed.

19.2.2 Axis card installation

1. Seat boards carefully and be certain to plug power transistor (red connector) on each axis card.

2. Replace axis board retaining bar.

3. Attach all stages to their assigned axis and secure BOTH screws on cable nuts. Verify the stage and axis card match from the system settings sheet (Appendix Z).

4. Attach power cord and computer interface to controller.

5. Leave controller cover off.

6. Observe the row of yellow LEDs on the controller board as the unit is powered on. *See* Figure 56.

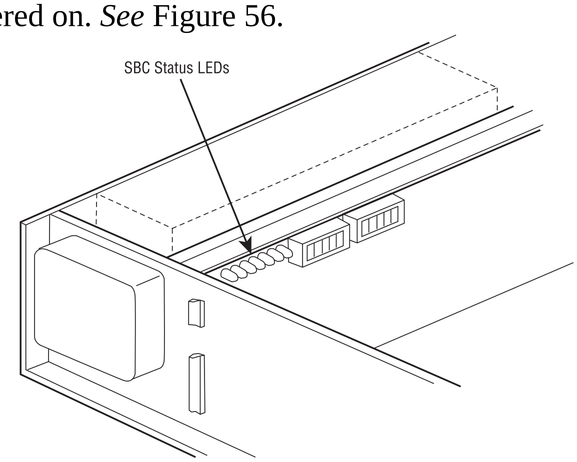


Figure 56: The controller board status LEDs.

7. Refer to the silver label in the inside of the top cover. Observe LEDs 2 and 3.

8. LED number 2 should be ON indicating that the controller board processor is functioning properly.

9. LEDs 3 through 8 indicate the status of the axis bus. The LEDs should only be ON for axes where there is NO axis card. If an LED is OFF for an axis with an axis board installed, the processor has not recognized the installed axis card. If an error occurs:

   - Check to be sure that axis board is seated properly.
   - Check the fuse status LEDs on the controller bus. If all these LEDs are not on, a fuse has blown.
   - If you've just installed a firmware upgrade, check that PROMs are installed correctly (correct location, orientation, etc.).

10. If the LEDs indicate no errors, perform the system initialization procedure (*See* Section 19.5).

WARNING

Do not attempt to command motion or operate the system until the System Initialization procedure is completed.

19.3 Removing and Installing the Controller Board

We recommend removing the controller card from the controller when removing or installing firmware only. No other routine maintenance procedures require controller board removal.

WARNING

Hazardous voltages are present inside the PM500. Internal upgrades and servicing should be performed by qualified individuals only.

The PM500 contains static-sensitive devices. Exercise appropriate cautions when handling PM500 boards, cables and other internal components.

Do not install anything into your PM500 controller except items provided by Newport specifically for installation in the PM500.

Controller board removal

1. Turn off and unplug the controller.
2. Detach all axis and control cables.
3. Place the controller in a static-safe area or on a static mat.
4. Remove the four Phillips-head screws from the top cover.
5. Remove the cover panel by tipping the rear of cover up and pulling toward the rear of the controller.

> **WARNING**
>
> **Touch the controller chassis before coming in contact with internal components.**

6. Remove the axis board retaining bar located along the inside front of the controller. *See* Figure 57.

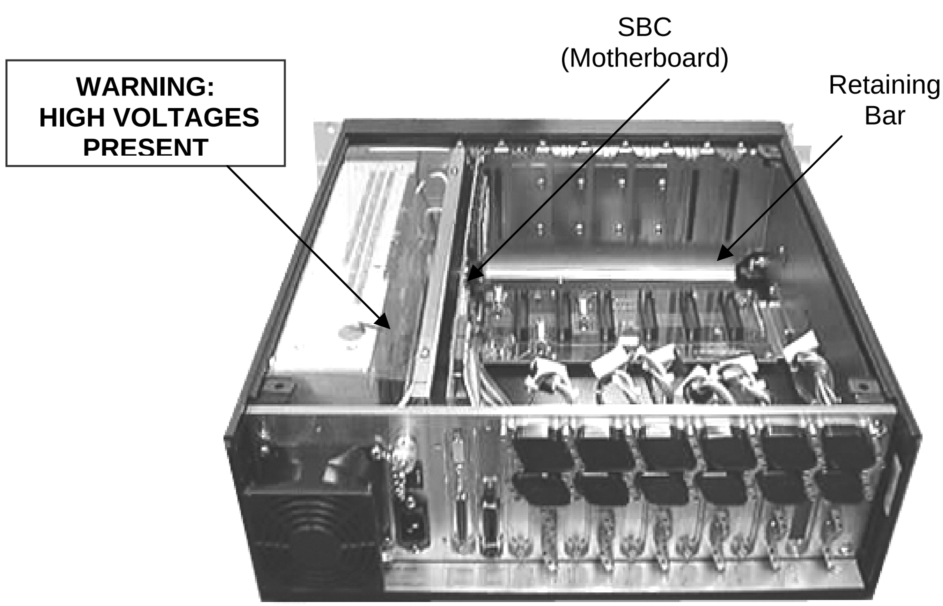


Figure 57: The retaining bar removal

7. Unplug the GPIB ribbon harness and the Operator Interface port harness from the controller board.
8. Using a 3/16-inch nut driver, remove the cable nuts from the RS-232 port.

9. Remove the card by grasping it firmly near the center and pulling upwards. Be certain that all connections have been unplugged from the controller card.

10. Place the card in a static-safe area or on a static mat.

11. Refer to the upgrading system firmware section (Section 19.4) for instructions on installing firmware. Return to this section for controller board installation.

19.3.1 Controller board installation

1. Seat controller board carefully and be certain to reconnect RS-232, GPIB and OIM connectors on the board.

2. Replace board retaining bar.

3. Attach all stages to their assigned axes and secure BOTH screws on cable nuts.

4. Attach line cord and computer interface to controller.

5. Leave top controller cover off.

6. Observe the row of yellow LEDs on the controller board as the unit is powered on. *See* Figure 58.

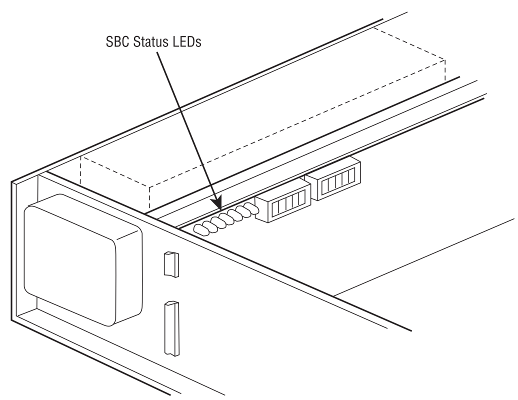


Figure 58: The controller board status

7. Refer to the silver label in the inside of the top cover. Observe LEDs 2 and 3.

8. LED number 2 should be ON indicating that the controller board processor is functioning properly.

9. LEDs 4 through 8 indicate the status of the axis bus. The LEDs should only be ON for axes where there is NO axis card. If an LED is OFF for an axis with an axis board installed, the processor has not recognized the installed axis card. If error occurs:

   - Check to be sure that axis board is seated properly.

- Check the fuse status LEDs on the controller bus. If all these LEDs are not on, a fuse has blown.
- If you have just installed a firmware upgrade, check that PROMs are installed correctly (correct location, orientation, etc.).

10. If the LEDs indicate no errors, perform the system initialization procedure (Section 19.5).

WARNING

Do not attempt to command motion or operate the system until the System Initialization procedure is completed.

19.4 Upgrading System Firmware

Firmware upgrade may be necessary to:

- Access the added features of new firmware revisions
- Upgrade the system when installing new motion devices
- Upgrade the system when installing optional hardware interfaces

There are two types of firmware in the PM500—system firmware, located on the controller card—and axis card firmware, located on each axis board. The firmware version is indicated on the PROM labels. It is not required that any axis card and controller board revisions agree, because axis card firmware will differ depending on the type of motion device. However, it is important that the revision level on any one card be the same.

WARNING

The PM500 contains static-sensitive devices. Exercise appropriate cautions when handling PM500 boards, cables and other internal components. Internal upgrades and servicing should be performed by qualified individuals only.

Do not install anything in your PM500 Controller except items provided by Newport specifically for installation in the PM500.

19.4.1 What you'll need

You will require the following to perform this procedure:

- Replacement firmware
- PROM puller or chip puller (recommended)
- Computer interfaced for the PM500 via RS-232 or GPIB
- Static-safe area or static mat with wrist-strap

19.4.2 For axis card firmware upgrades

Refer to "Removing and installing axis cards" (Section 19.2) and remove the appropriate axis card or cards from the system.

19.4.3 For controller card firmware upgrades

Refer to "Removing and installing the controller board" (Section 19.3) and remove the controller and all axis cards.

NOTE

The firmware and code it contains is the property of Newport Corporation. Please return the old firmware PROMs to Newport.

19.4.4 Firmware upgrade procedure

Check that you have the following:

- Original owners manual for the system.
- Verify that the information on the system setting sheet (Appendix Z) is accurate for your particular controller.

If you don't have this documentation or information, record the axis card serial numbers, matching stage serial numbers, and axis port assignments as currently installed in the controller after the controller cover has been removed.

1. Place boards on a static mat or static-safe area.
2. Verify that you have selected the correct firmware for the card you are about to upgrade.
3. Using a PROM or chip puller, remove the old firmware from one card.
4. When replacing the new firmware:
   - Be patient
   - Do not use excessive force
   - Make certain to support the axis card from behind when seating PROMs.

Check that you are installing the proper PROM the ***correct way*** in the ***proper socket***. Each PROM has a notch on one end indicating which way it should be placed in the socket. The index notches on the socket and the chip should be on the same side.

To ensure that the correct PROM goes in the right PCB socket, the socket and PROM identification numbers must match. The PROM identification number is at the top of the label and begins with the letter "U." The PCB socket is marked by a number on the PCB near the socket. See Figure 59.

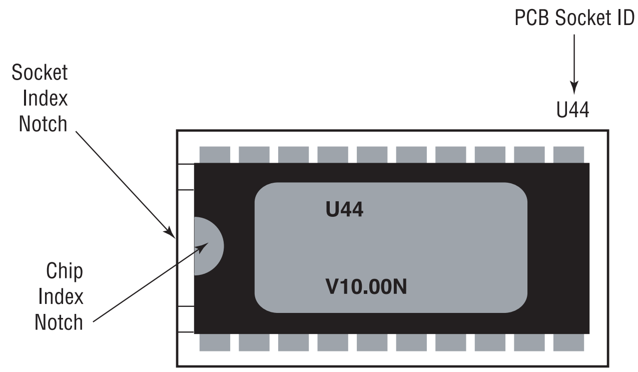


Figure 59: Proper PROM placement in socket and location on card.

5. Install firmware for each axis board and controller where required. Carefully check that all legs on the PROMs are inserted properly and not bent or folded under.

6. Replace boards via the instructions in the appropriate board installation procedure in this manual.

7. The system must be initialized before use. Proceed to the System Initialization (Section 19.5) to complete the firmware upgrade process.

⚠	**WARNING** **Do not attempt to command motion or operate the system until the System Initialization procedure is completed.**

19.5 System Initialization

This procedure should be done whenever:

- Axis card assignment has been changed
- Additional axis cards have been installed
- After firmware upgrade
- Axis interface options have been installed

The axis card initialization process requires that commands be sent to the controller through the RS-232 or IEEE interface. The PM500-K6 cannot be used for system initialization.

1. Via RS-232 or GPIB, issue the following commands individually:

```
SSCUM0
SDEFEE
```

Commands system to revert to default parameters stored in firmware. This will instruct the controller to identify and load parameters from all installed axis cards.*

```
RSTART
```

System restart. All controller status LEDs should blink momentarily. This restarts the controller using the newly load parameters.*

2. Issue `[n]AZ:` Auto-Zero for all installed axis.*

**Refer to the Command Reference section for details on these commands.*

CAUTION

Do not disturb stages for 40 seconds after commencing the Auto-Zero function.

3. Issue `[n]SETUP12`: for each installed axis. The stages should begin random motions. Check that motion is crisp. Allow stages to run for 5 minutes.

If you experience problems, refer to the Troubleshooting section of this manual (Section 19.6). If the problem persists, contact Newport ***before*** attempting any servo adjustment or other repairs.

19.6 Troubleshooting

19.6.1 LED diagnostics

With the use of LEDs on the printed circuit boards, an electrical failure can be visually isolated to any of the following major components:

- Power supply
- Controller board
- Axis board(s)

19.6.2 Motherboard/power supply

On the motherboard/power supply board, there are indicator LEDs corresponding to each of the power supply voltages:

- +5
- +15

- –15
- V Motor

If you have difficulty locating the LEDs, check Section 16.2 - Functional description and specification (Motherboard). Any LED that is out or dim indicates a problem either with a board or with the power supply. Refer to Section 16.3 - Functional description and specification (Controller board) to troubleshoot any problems indicated by the LEDs on the motherboard. A power supply change must be made at the factory or by a qualified Newport service technician.

NOTE

If all the lights are on, there is still the possibility of an incorrect voltage. Typically, if the +5 volt supply varies more than ±5%, it can cause failures on all boards.

19.6.3 Controller board

By checking the status of the LEDs on the controller board, it's possible to determine:

- If the selected interface is working
- If the processor is OK
- If any axis is not present or has failed

The LED status lights are located at the top edge of the controller board next to the DIP switches. If you have difficulty locating the status lights, *see* Section 16 - Functional description and specifications. Under normal operation, the LEDs DS-1 through DS-8 yield the following information:

DS1	Used for GPIB Interface. Shows that the PM500 Controller is addressed to talk or listen when ON.
DS2	When ON, shows the controller board is running its program and is OK. When DS2 is off, the controller board has not passed its self-test.
DS3	Flashes ON when a transmission transfers through the selected interface. If a communication is initiated and no indication is seen, there is a communication problem. Probably a disconnected or mis-wired cable, a bad switch setting, or improper use of available handshaking lines.
DS4	ON indicates board in X-axis slot is absent or its I/O has failed.
DS5	ON indicates board in Y-axis slot is absent or its I/O has failed.
DS6	ON indicates board in Z-axis slot is absent or its I/O has failed.
DS7	ON indicates board in A-axis slot is absent or its I/O has failed.

19.6.4 Axis boards

When observing an axis board under normal operation, it is easy to determine the following facts about the board and the stage it drives by observing the LEDs DS1-DS9:

- Whether the board is operating
- Whether the motor has been turned off
- Whether the stage is at limit
- Whether the stage cable has become disconnected
- Whether the stage is at null or off of null for some reason

The LEDs DS1-DS9 give the following information:

DS1	When ON, indicates that the motor amplifier is enabled (other sources may turn it off even though this light is still on). When OFF, indicates that the command "**M**" (motor off) has been sent or the "**AZ**" (Auto-Zero) command is being executed.
DS2-DS4	When ON, indicates an Auto-Zero is in progress.
DS5	When ON, indicates the stage is at limit towards the motor.
DS6	When ON, indicates a position somewhere between null and motor limit.
DS7	When ON, indicates a null (position) at the internal magnified resolution (is not on steadily in most systems because of noise).
DS8	When ON, indicates a position somewhere between null and the away limit.
DS9	When ON, indicates the stage is at limit away from the motor.
DS5+DS9	When ON, indicates a disconnected stage cable, or a failure of the axis board. Both limits in a normally operating system will not be on at the same time.
DS6, DS7, and DS8	Together, these three LEDs can tell you a lot about system performance. When all three lights are ON and flickering, the system is servoing around null, and the microprocessor is operational. This is a normal condition. If there's a bias in the flashing, (more toward one limit or the other), then the axis may have a thrust on it (such as from a dragging cable), an Auto-Zero may be needed, or the dynamics of the system may need to be adjusted via **HSLOPE**, **V**, **ACCEL**, or **DECEL** commands.

19.7 Troubleshooting Tables

The following tables will help you locate the faulty element in the PM500 system. Use the Basic Troubleshooting Table (Table 19.1) that follows to categorize the failure. Then use the other tables to further isolate the failure.

The goal of these tables is to troubleshoot an electrical problem to the board (or power supply) level. Board level repair, major failures or subtle performance problems should be referred to the factory for repair and re-verification.

It is important to note that all peripheral devices must be connected to the controller before power is applied.

Table 2: Basic Troubleshooting			
Step	***Symptom***	***Test***	***Action***
1	Observe a failure		
2	No power-on light, no fan or no axes move	Is there power and lights?	Refer to Table 19.2
3	Does not respond to the computer, terminal or PM500-K	Are the communication links working?	Refer to Table 19.3
4	Axis does not move or does not move as expected	Does axis respond (servo to position)?	Refer to Table 19.4
5	Axis overshoots and/or undershoots or does not meet other specifications	Is performance marginal?	Refer to Table 19.5
6	Axis does not respond to PM500-K input; no PM500-K sign-on at power-up	Does Remote Control Console (PM500-K) work?	Refer to Table 19.6
7		System still does not Work properly	Contact Newport customer service department

Table 3: Check Power and Lights			
Step	***Symptom***	***Test***	***Action***
1	Provide suitable power: 110 /220 volts, 250 VA	Check AC connection	Proceed to Step 2.
2	Dim front panel, Fan light; Fan barely operates	Check 110/220 switch switch on rear panel (if present)	Set switch to proper setting. Proceed to Step 3.
3	No front panel light, no fan	Check for tripped input CP	Disconnect power cord, reset CP. Proceed to Step 4.
4		Remove top cover	

Step	*Symptom*	*Test*	*Action*
5	Lights are bright	Check power supply lights on motherboard	Return to Table 19.1 [Basic]
	Turn off power	Lights are off or very dim	Check fuses. Proceed to Step 6.
	+15 or –15V light out, RS-232 quit working	Check analog supplies	Check fuses (don't drop scope while testing unit grounds ontoboards!)
	Blew a new fuse	Check analog supplies	Power off, test for short. Proceed to Step 7.
	+Vm light is out; no power to motors	Check motor supply fuse, motor xstr blown or shorted to ground shorted motor xstr	Check fuse, if bad remove AC power, check for short to ground through
	Locate troubled axis short to ground	Locate +Vm	Disconnect motor xstrs, 1 axis at a time, until no short, get factory help. Proceed to Step 7.
6	Fan is off	Check fan	Contact service department
	Fan is on		Proceed to Step 7.
7	Short is on board	Turn off the power, remove an axis or controller board	Replace fuse, same amp rating. Proceed to Step 8.
8		Turn power on	Proceed to Step 9.
9	Lights are bright	Check power supply	Contact service department, return board(s) for repair
	Lights are off or very dim.		Proceed to Step 10
10		Turn off the power, replace the board that was removed in Step 7.	Proceed to Step 11
11		Remove a different axis	Proceed to Step 12 or controller board

Step	Symptom	Test	Action
12		Repeat Steps 8 thru 11 until defective board is found	When all boards are checked. Proceed to Step 13
13	Power supply lights are still very dim or off.	Check power supply, lights	Contact service department

Table 4: Communication Link(s) Troubleshooting

Step	Symptom	Test	Action
1	Controller does not	Check interface cable respond to any connections commands or controller responds erratically to commands	Proceed to Step 2.
2		Remove top cover	Proceed to Step 3.
3	Switch settings correct	Check switch settings on controller board [RS-232 DIP settings]	Proceed to Step 4.
	Switch setting(s) wrong	Correct switch(s) settings.	Proceed to Step 4.
4		Turn power off	Proceed to Step 5.
5	DS1-8 should flash on for 0.5 sec, then off for 0.5 sec. Check that: DS1 is on or off; DS2 is always on; DS3 is off unless character is sent from computer or terminal; DS4-8 are off for each axis that is installed (i.e., DS4 off= X Axis installed, DS5 off= Y Axis installed, DS6 off= Z Axis installed, DS7 off= A Axis installed, DS8 off= B Axis installed)	While observing LEDs (DS1-8) on controller board, turn power on	Controller board functioning OK. Proceed to Step 6.
	DS1-8 do not follow the sequence above		Controller board problem. Proceed to Step 6.

Step	Symptom	Test	Action
6	Bent leads	If ROMs have been changed, check that they are properly inserted	Correct problem. Proceed to Step 12
	Reversed ROM		OM has been damaged. Contact service department.
7	TP2= 4.45V 4.45V +0.05V	Check voltage on TP2 on controller board	Adjust R1 for TP2<> 4.45V at TP2
	No communic. check	Check motherboard lights	±15V lights
		Lights out [Power/lights]	*See* Table 19.2
		Lights all OK	Proceed to Step 8.
8	DS3 lights for 1 sec	Send character to controller board	Receives character. Proceed to Step 9.
	DS3 off		Not receiving
	character.		
	DS1 is off (GPIB only)		Proceed to Step 10 Check address switches and computer program. Proceed to Step 7.
9		Check ENAINT command [system commands].	*See* Section 15 Proceed to Step 8
10	RS-232 out	Check computer/ terminal interface settings and computer program	Recheck Step 7.
11		If problem still exists	Contact service
	department		

Table 5: Check Servo Action and Axis Response			
Step	***Symptom***	***Test***	***Action***
1	Servo action, limits and fiducials operate	Check all cables for tight connection	Tighten connector cables—*both screws*
	Servo action, limits and fiducials operate poorly or not at all		If problem still exists. Proceed to Step 2
2	Motor does not provide opposing torque when turned by hand	Is motor off?	Proceed to Step 3
	Motor provides oppo-sing torque, but does not return to original position		Proceed to Step 7
3	Rear heatsink in excess of 75	Is rear heatsink Cexcessively hot?	Check motor(s) for stall condition. If the heatsink is still hot, contact service department
	Rear heatsink temp less than 75 C		Proceed to Step 4.
4	J2 pin 9 <= .4V	Is optional motor off input floating or at logic high?	J2 pin 9 > .4V. Ground J2 pin 9. Proceed to Step 5.
5	Thermal switch: broken wires contacts > 10 ohms disconnected	Check shunt or thermal switch on J8 (controller board)	Shunt missing, replace shunt. Replace or repair thermal switch
6		Problem still exists department	Contact service
7	Axis will not move, or runs away; singing or unusual noise from motor	Make sure dynamic adjustments have been made	Adjust axis board. If problem still exists, proceed to Step 8.
8		Problem still exists	Contact service department

Table 6: Check Marginal Performance/Positioning

Step	*Symptom*	*Test*	*Action*
1	Stage oscillates when tapped lightly in direction of travel	Axis home	Adjust HSLOPE. Proceed to Step 2.
2	Undershoots moves Overshoots moves Excites mechanical system during move Accelerates too slowly and/or DECEL.	Make moves > LSIZE	Decrease HSLOPE. Decrease DECEL. Decrease ACCEL and/or DECEL Increase ACCEL Proceed to Step 3.
3	Small moves take too long	Make moves < LSIZE	Increase LACCEL and/or LDECEL. Decrease HSLOPE. Proceed to Step 4.
4	Does not repeat	No improvement, have moves properly	Perform axis board Dynamic Adjustments adjustment. been made?
	Stage “runs away”, moves are larger/ smaller than they should be, system oscillates when HSLOPE >= minimum resolution		Proceed to Step 5.
5	Axis reads an offset even after being cleared	Check DAC Offset (*see* Section 19.7) [axis adj.]	Proceed to Step 6.
6		If problem still exists	Contact service department

Table 7: Check Remote Console (PM500-K)

Step	Symptom	Test	Action
1	Cable disconnected	Check interface cable from PM500-K to PM500	Reconnect cable
	Cable in place		Proceed to Step 2.
2	Sign on display appears	Type RSTART, or turn	Proceed to Step 4. power OFF then ON again
	Sign on display does not appear		Check OPON, OPDEF commands, Refer to Section 16 [PM500K commands]. Proceed to Step 3.
3	Sign on display appears	Check sign on display	Proceed to Step 4.
	Sign on display does not appear		Contact service department.
4		Use PM500-K to make a move with each axis	Go to Step 5.
5		If problem still exists	Contact service department

This page is intentionally left blank

20 Appendix A
Cabling and Communications

20.1 Outline of Appendix A

The following topic is covered in this Appendix:

- Cabling and communications

20.2 Cabling and Communications

Axis cable pinouts and wire assignments

WARNING

Cable connections should be made with the power OFF to prevent possible injury or equipment damage. Connections should be securely fastened with both screws on the D-style connector.

Very short lengths of customer-supplied cable may be used in very electrically-quiet environments. Ribbon cable is not recommended due to lack of shielding. If you experience motion or performance problems, reinstall the factory supplied cable before beginning troubleshooting.

The connection between the PM500 and stages utilizes a 25-pin shielded cable. Figure 60 shows a pictorial view of the cable, and Figure 61 lists the connections and color codes of the wires. All stage-to-controller cabling should conform to Figure 61 to ensure proper performance and safety.

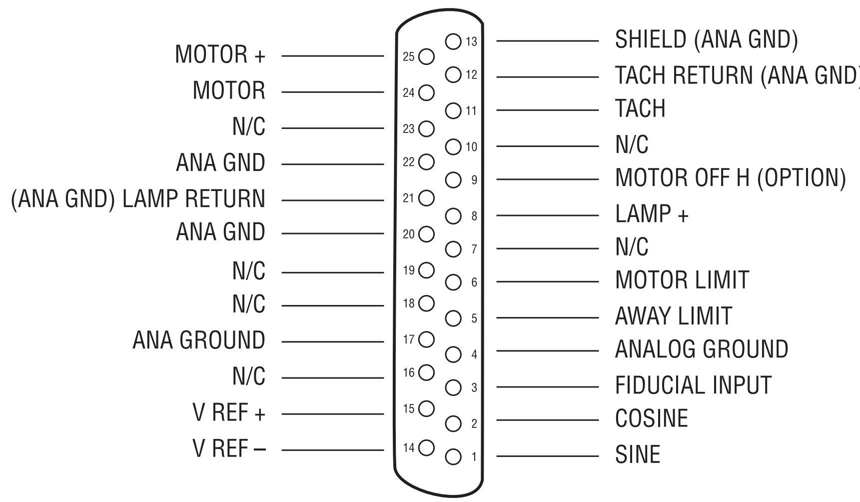


Figure 60: D-style 25-pin axis cable connector to rear panel.

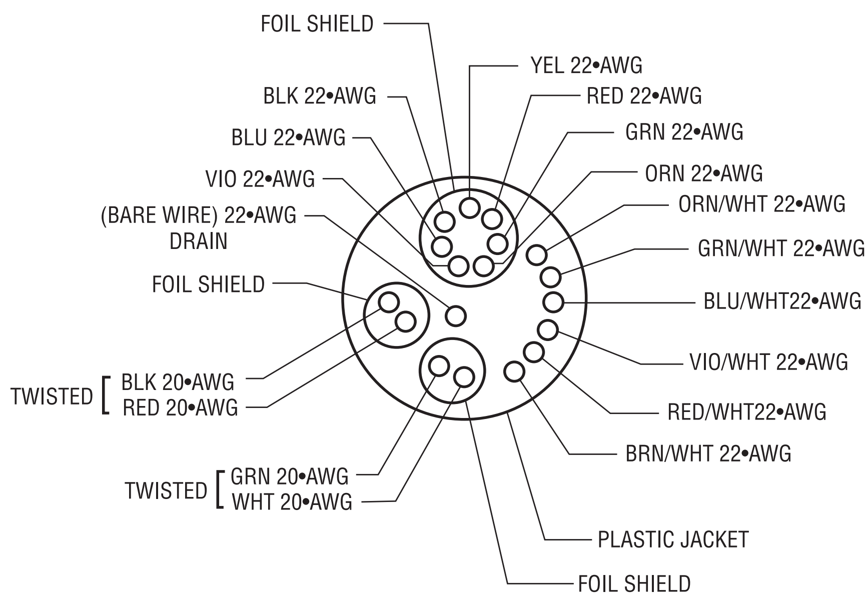


Figure 61: Axis cable.

21 Appendix B
Cleanroom Compatibility and Outgassing

21.1 Outline of Appendix B

The following topic is covered in this Appendix:

- Cleanroom compatibility and preparation
- Outgassing

21.2 Cleanroom Compatibility and Preparation

PM500 stages may be used in cleanrooms with no modifications down to Class 0.1 where *particulate* contamination is a concern. All that's required is a normal cleanroom-preparation procedure.

CAUTION

Disassembling the device voids your warranty.

Devices intended for cleanroom use should be new from the factory. Devices that have been used outside a cleanroom cannot be properly cleaned without disassembly—a procedure that can only be performed at the factory. Newport offers no onsite cleanroom-preparation services.

CAUTION

Device preparation for cleanroom is straight forward, provided that a few precautions are heeded:

- **Clean the exterior of the stage with alcohol, but don't flush.**
- **Do not use acetone, which may destroy the coating on the motor or dissolve the lubrication.**
- **Do not submerge the stage in any liquid.**
- **Do not flush, clean or re-lubricate the stage bearings.**

21.3 Cleanroom Compatibility and Preparation

The lubricants, grease, and motor coatings of PM500 devices will outgas to some degree. Where outgassing is a concern, take precautions to contain the outgassing contamination.

NOTE

The specific lubricant in PM500 stages is critical to their precision and longevity. Other lubricants degrade service life, repeatable sub-micron motion or both, Newport offers no alternative lubricants.

22 Appendix C
Vacuum Compatibilities

22.1 Outline of Appendix C

The following topic is covered in this Appendix:

- Vacuum compatibilities

22.2 Vacuum Compatabilities

At the time of this printing, no vacuum-compatible PM500 devices are available as standard products. For more information about special-purpose vacuum-compatible devices, contact Newport. Inquiries will be handled on a case-by-case basis.

This page is intentionally left blank

23 Appendix D
Memory Compatibilities

23.1 Outline of Appendix D

The following topic is covered in this Appendix:

- Memory capacities and capabilities of the PM500-5/C6 controller

23.2 Memory capacities & capabilities of the controller

This information is provided for those users where security concerns warrant the user to know if the PM500 memory can be used to store and/or transport information.

The PM500-C /C6 has three type of memory storage devices/components:

- EEPROM
- ROM - Read Only Memory
- RAM - Random Access Memory

These memory storage devices are present on both the SBC (motherboard) and Axis card. Please note that by nature ROM is read-only memory and thus is excluded from concerns of information storage.

23.2.1 SBC (Motherboard)

EEPROM: The motherboard has kilobytes of EEPROM memory in which only system parameters can be stored, the user can only store information that directly relates to system parameters.

ROM: The motherboard also has 4 Kilobytes of ROM memory where system software is stored and which cannot be rewritten while resident in the controller.

RAM: There is 32 kilobytes of RAM resident on the SBC which is used as program overhead space (scratchpad) for the storage of temporary information. Its contents are erased at power-off.

23.2.2 Axis Cards

RAM: Each axis card has 16 Kilobytes of RAM in which system software is stored from the SBC at power-up. It can only be accessed by the SBC. Its contents are erased at power-off.

There are extreme limitations regarding information that can be stored in the PM500 controller, in addition, the PM500 has no macro capabilities. The execution of the DEFEE command will erase all user set parameters and default the system to factory settings.

24 Appendix E
Motion Specifications

24.1 Motion Trajectory Specifications

The following are mechanical performance specifications that pertain to the bearing and raceways of the motion carriage during motion.

Runout

A motions system's accuracy specification, by convention, is along the desired line of travel and does not consider other positioning errors such as runout. Runout is defined as departures from a perfectly straight motion. By convention this is broken down into 2 orthogonal components:

Flatness of travel: deviations from a straight line perpendicular to the plane of travel.

Straightness of travel: deviations in the plane of travel.

These are both less than ±1.25µm over the full travel of a PM500 8" stage.

Angular Deviations

Runout measurements are generally made interferometrically with the interferometer's optic assembly mounted as close as possible to the stage's carriage. This is because the stage carriage also exhibits a small amount of twist (pitch, roll, yaw) as it translates. The effect of that twist is magnified as the effect of the lever-arm of the optics assembly increases. So the amount of motion error is directly related to the distance the sample is away from the motion carriage. This is known as "Abbe Error".

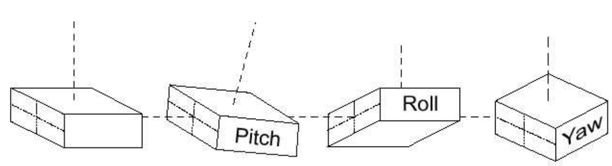


Figure 62: The trajectory error effects of pitch, roll, and yaw.

The average pitch, roll, and yaw for a PM500 stage is ±25µm per 25mm and is non-cumulative. For this reason we stress that the mounting surface for your stage be flat to 5 microns TIR. Warpage of the stage will not only also cause the bearings to bind, but will distort the bearing ways, worsening the overall positioning accuracy of your system.

24.2 Motion Performance Summary

There are many contributing factors to the overall positioning performance of a motion device. The complexity is increased when multiple motion devices are stacked to form a motion system. The following factors either separately or cumulatively contribute to error in the position of your desired sample:

- Absolute accuracy along its line of travel
- Indeterminacy - repeatability of motion
- Pitch, Roll, and Yaw
- Lever-arm of the sample on the stage
- Rigidity of the mounting surface
- Flatness of the mounting surface

A term which best describes the overall positioning accuracy for a motion system is the “Circle of Error” for a single linear stage or “Sphere of Error” for multiple axis stacks.

The mixing of low performance stages with the PM500 is not recommended as the motion performance of the low cost stage will most certainly be worse than the positioning performance of the PM500.

For applications were multiple motion combinations are required, we recommend the integrated PM500 stages such as the XYØ and XYZ Mini-stage. Integrated PM500 motion devices have better overall positioning performance than a stack of individual stages.

25 Appendix F
Packaging

25.1 Outline of Appendix F

The following topic is covered in this Appendix:

- Packaging your PM500 System

25.2 Packaging your PM500 System

The PM500 must be packed properly to assure that no damage, and resulting additional repairs, occurs during shipment. In the event of damage, repairs will be at your cost.

You should use the original packing materials to repack your system. If you do not have these materials, you may order shipping boxes from Newport for a nominal fee.

Unless otherwise specified by Newport Technical Support you should return the entire system— Motion device, controller, and cables—for evaluation and repair.

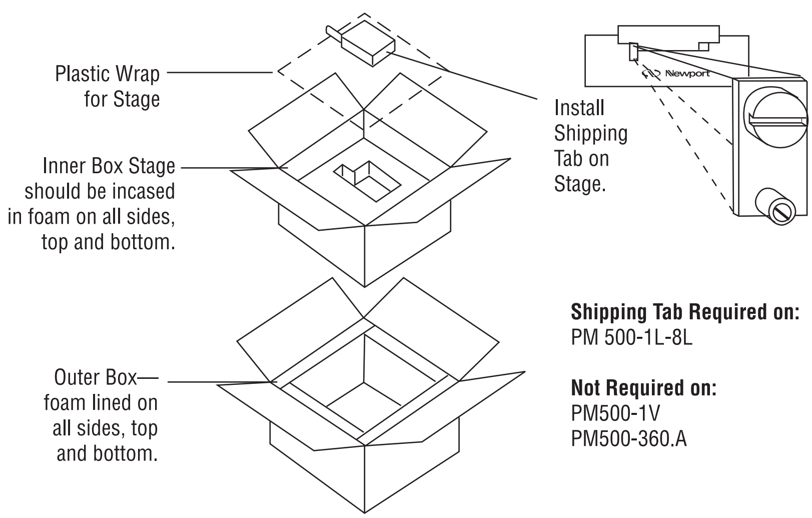


Figure 63: The typical factory packaging for PM500 devices

NOTE

The metal shipping brackets must be installed to prevent the possibility of damage and to preserve the warranty (if any).

CAUTION

Please request an empty box and/or stage locking tabs from Newport if you do not have them. Do not ship the stage without the locking tabs installed or without the proper shipping materials.

25.3 Packaging your PM500-C6 Controllers

PM500-C6 Controllers are single boxed encased in 2”foam on all sides. Axes Cables and line cords can be packed with the PM500-C6.

26 Appendix G
Warranty Return

26.1 Warranty and Service Policy

Newport Corporation warrants the PM500 family of products to be free from defects in materials and workmanship for a period of one year after delivery. Newport does not assume liability for installation, labor or incidental or consequential damages.

Returns

Obtain a Return Authorization Number from Newport before returning any product. Newport reserves the right to deduct an adequate service charge to cover inspection, testing, and handling from any credit.

Repairs

All repairs, warranty and non-warranty, are made on a FOB factory basis, and all transportation charges must be prepaid by the Buyer. Obtain a Return Authorization Number from Newport before returning any product to us for repair.

Weights and Dimensions

Published weights and dimensions are actual or careful estimates but are not guaranteed. Dimensions in catalogs are normally accurate but not to be used for construction. Upon request, details for construction purpose will be supplied.

This page is intentionally left blank

27 Appendix H

27.1 Service Form

Name ______________________________ **Return Authorization #** ________________
Company ____________________________ (Please obtain prior to return of item)
Address ____________________________ Date ________________________________
Country ____________________________ Phone Number ________________________
P.O. Number _________________________ FAX Number _________________________

Items(s) Being Returned:

Model # ___________________________ Serial # ______________________________
Description: ____
Reason for return of goods (please list any specific problems) __________________
____

Please complete the below, as appropriate
List all control settings and describe problem: ______________________________
____
____
____(Attach additional sheets as necessary.)

Show a block diagram of your system if appropriate.

Where is the measurement being performed?

(Factory, controlled laboratory, out-of-door, etc.) ___________________________
What power line voltage is used? ____________________ Variation? _______________
Frequency? ___________________ Ambient Temperature? _________________________
Variation? __________________ ºF. Rel. Humidity? ________ Other? _______________
Any additional information. (If special modifications have been made by the user, please describe below.)

____

This page is intentionally left blank

28 Appendix I

28.1 System Settings

PM500 factory settings

Customer	PM500-C6 S/N
Controller board S/N	Controller board ROM version
ENAINT	REVLVL
Set by	Setting date
GPIB address	

Hardware Identification						
	X	Y	Z	A	B	C
Stage model						
Stage S/N						
Stage travel						
Scale resolution						
Axis ROM rev						
Axis card S/N						
Maximum speed						
Encoder signal test period						
Limit to fid.						

Parameters						
AXIS	X	Y	Z	A	B	C
ACCEL						
DECEL						
FV						
HSLOPE						
LACCEL						
LDECEL						
LSIZE						
LV						
NULL						
RESFAC						
SIGN						
V						

PM500 user settings

Customer	PM500-C6 S/N
Controller board S/N	Controller board ROM version
ENAINT	REVLVL
Set by	Setting date
GPIB address	

Hardware Identification

	X	Y	Z	A	B	C
Stage model						
Stage S/N						
Stage travel						
Scale resolution						
Axis ROM rev						
Axis card S/N						
Maximum speed						
Encoder signal test period						
Limit to fid.						

Parameters

AXIS	X	Y	Z	A	B	C
ACCEL						
DECEL						
FV						
HSLOPE						
LACCEL						
LDECEL						
LSIZE						
LV						
NULL						
RESFAC						
SIGN						
V						

Index

R

S

T

U

V

W

Z