Performance Motion Devices, Inc.
12 Waltham St.
Lexington, MA, 02421

TEL: (781) 674-9860
FAX: (781) 674-9861
support@pmdcorp.com

Step by step guide to phase initialization on the Navigator and Pilot motion processors

History

Date	Version	Author	Notes
2001-05-18	1.0.0	JJ	First version
2001-05-23	1.0.1	JJ	Updated with input from engineering
2001-05-29	1.0.2	JJ	Added info on PWM active state

Introduction

This document describes in detail the setup procedure for initializing commutation on the brushless Navigator and Pilot families of motion processors. You should read this entire document before attempting the initialization procedure for the first time. It is assumed that you have read the Sinusoidal Commutation section of the *User's Guide* prior to reading this document.

When controlling a brushless motor, an initial phase angle must be determined before the motor can be rotated. The PMD brushless motion processors provide several methods for phase initialization. This document describes the Algorithmic and Hall-based methods.

Phase initialization must be performed whenever the motion processor is reset. This document will help you determine the correct sequence of commands to execute. This process is performed once after a hardware or software reset for every brushless axis that is controlled by the motion processor.

All commands are presented in a generic format that indicates a command name and optional parameters. You should use the commands in a format that is appropriate for the software you are using. For a detailed explanation of command format, refer to the *Programmer's Reference*. In this document all numbers prefixed with "0x" are presented in hexadecimal notation.

Getting started

Before attempting to initialize using any of the presented methods, input and output must be checked.

For information on pin and signal connections, refer to your motion processors *Technical Specifications* manual or the appropriate *Developer's Kit* manual.

Checking input

You should have an encoder connected to your motion processor, preferably one that provides an index signal once per rotation. Your motion processors' *Technical Specifications* manual has details on the required timing of the index signal and the way in which it is gated with the channel A and channel B signals. The index signal is used for correcting any commutation error caused by either lost encoder counts or a non-integer phase count value. This is discussed later.

While rotating the motor/encoder, execute the command GetActualPosition. The returned value should increment when rotating in one direction, and decrement when the rotating in the opposite direction. On the Navigator family you can also use the command GetSignalStatus to monitor the state of the encoder A, B and Index signals. The Pilot family does not support this feature.

You can also test the Index signal by issuing the command GetCaptureValue. When the encoder is rotated, GetCaptureValue should return a value once per rotation that is incrementing or decrementing by a value that is equal to the number of encoder counts per full rotation.

<u>Troubleshooting</u>

- GetActualPosition oscillates between 0 and 1
    - One of the encoder A or B signals is not connected correctly. Check these connections as well as GND and +5V supply to your encoder.
    - The encoder A and B signals must be 90 degrees out of phase for quadrature decoding to operate correctly. Verify this using an oscilloscope.
- Index capture does not work
    - The index signal is gated with the A and B encoder signals. This requires that all signals must be 'active' for a capture to occur. The index signal is expected to be active LOW by default. If it is not, the command SetSignalSense can be used to change its active state to be HIGH. **NOTE**: Care must be taken when inverting an A or B channel signal since this may violate the signal state required for an index capture.

Checking output

Even before the phasing is correctly initialized, the motion processor can generate motor output signals that can be checked. **The amplifier output signals should not be connected to the motor during this test**.

The following sequence of commands will generate amplifier output when the motor/encoder is rotated manually.

SetOutputMode m	// m is 0 for DAC, 1 for PWM S/M and 2 for PWM 5050
SetNumberPhases x	// x is 2 or 3 depending on type of motor
SetPhaseCounts 1024	// # of encoder counts per electrical cycle
SetMotorMode 0	// places axis in open loop mode
SetMotorCommand 16384	// This value specifies the magnitude of the sinusoidal waveform motor command. A value from 0 to 32,767 represents 0->100% of full-scale output. 16384 represents 50%.

The encoder should be connected to the motion processor and functioning correctly. Connect the output signals from either the motion processor or amplifier to an oscilloscope. Rotate the motor/encoder and the output signals from the motion processor/amplifier will fluctuate in a sinusoidal fashion. This confirms that your output is operational.

If you are using PWM output, check the 'active' interpretation that your amplifier expects. Refer to the *User's Guide* for details on the format and encoding of this signal.

Calculating Phase Counts (Sinusoidal commutation)

To perform sinusoidal commutation it is necessary to specify the number of encoder counts per electrical cycle. To determine this value the number of electrical cycles of the motor and the number of encoder counts per motor revolution must be known. Knowing these two quantities, the number of encoder counts per electrical cycle is given by the following equation:

Counts_per_cycle = Counts_per_rot/Electrical_cycles

where:

Counts_per_rot is the number of encoder counts per motor rotation
Electrical_cycles is the number of motor electrical cycles per rotation

The number of electrical cycles can normally be determined by examining the motor manufacturer's specification. The number of electrical cycles is usually half the number of poles. Care should be taken not to confuse poles with pole pairs. If pole pairs are specified, then take this to mean the number of electrical cycles.
The command used to set the number of encoder counts per electrical cycle is SetPhaseCounts. To read back this value use the command GetPhaseCounts.

Index Pulse Referencing

To enhance long term commutation reliability the PMD motion processors provides the ability to utilize an index pulse input from the motor encoder as a reference point during commutation. By using an index pulse during the phase calculations any long-term loss of encoder counts, which might otherwise affect the accuracy of the commutation, are automatically eliminated. To utilize index pulse referencing the motor encoder chosen must provide an index pulse signal to the chipset once per rotation. This index pulse is connected to the chipset using the Index signal. For information on pin and signal connections, refer to your motion processors *Technical Specifications* manual or the appropriate *Developer's Kit* manual.

Index pulse referencing is recommended for all rotary brushless motors with quadrature encoders. For linear brushless motors it is generally not used, although it can be used as long as the index pulses are arranged so that each pulse occurs at the same phase angle within the commutation cycle.

When using an index pulse the number of encoder counts per electrical cycle is not required to be an exact integer. In the case that this value is not an integer, the nearest integer should be chosen. Conversely, if index pulses are not being used then the number of counts per electrical cycle must be an exact integer, with no remainder. For example if a 6-pole brushless motor is to be used with an encoder without an index pulse than an encoder with 1200 counts per rotation would be an appropriate choice, but an encoder with 1024 would not because 1024 cannot be divided by 3 evenly.

The command SetPhaseCorrectionMode is used to enable/disable index pulse phase correction.

Which initialization method should I use?

If the motor has hall signals and they are 120 degrees apart, then hall-based initialization is the best method to use. This is because it requires no initial motor rotation for phasing to be determined. However, as a first step it can often be easiest to use the algorithmic method just to verify your connections and to establish motor rotation. Algorithmic can also be used to obtain a reference point for verifying the hall initialization. This is discussed later.

If you do not have hall sensors, algorithmic initialization should be used. Other methods are also available. These are discussed in the *User's Guide* sinusoidal commutation section.

Algorithmic Initialization

In the algorithmic initialization mode no additional motor sensors beyond the position encoder are required. To determine the phasing the chipset performs a sequence that briefly stimulates the motor windings, and sets the initial phasing using the observed motor response. From the resulting motion the chipset can automatically determine the correct motor phasing.

Depending on the size and speed of the motor, the time between the start of motor phasing and the motor coming to a complete rest (settling time) will vary. To accommodate these differences the amount of time to wait for the motor to settle is programmable using the command SetPhaseInitializeTime. To read back this value use the command GetPhaseInitializeTime.

To minimize the impact on the system mechanics this method utilizes a motor command value set by the host processor to determine the overall amount of power to "inject" into the motor during phase initialization. Typically, the amount of power to inject should be in the range of 5 - 25 % of full-scale output, but in any case should be at least 3 times the breakaway starting friction. For best results the initialization motor command value should be determined experimentally. The command used to set the motor output level is SetMotorCommand. To read back this value use the command GetMotorCommand.

To execute the initialization procedure, the command InitializePhase is used. Upon executing this command, the phasing procedure will immediately be executed.

1. Perform Initialization

Shown below is a typical sequence of commands to perform the initialization.

```
SetOutputMode m              // m is 0 for DAC, 1 for PWM S/M and 2 for PWM 5050
SetNumberPhases x            // x is 2 or 3 depending on type of motor
SetPhaseCounts yyyy          // yyyy is # of encoder counts per electrical cycle
SetPhaseInitializeMode 0     // set phase initialize mode to 'algorithmic'
SetMotorMode 0               // places axis in open loop mode, required for algorithmic initialization.
SetPhaseInitializeTime zzzz  // zzzz is # of motion processor cycles to initialize for
SetMotorCommand wwww         // wwww is motor command.
InitializePhase
```

During algorithmic phase initialization the motor may move suddenly in either direction. Proper safety precautions should be taken to prevent damage from this movement. In addition, to provide accurate results motor movement must be unobstructed in both directions and must not experience excessive starting friction.

As stated previously, the value used for the motor command should be between and 5-25% of full-scale output. Start at a value 1000 and work up slowly until you see motion occur. You can start with a value of 3,000 – 5,000 for the SetPhaseInitializeTime. The exact time required is determined experimentally and is the time it takes for the motor to “settle” between excitation cycles.

Once the above sequence has been issued, the motor will perform 2 short movements. The amount of rotation of each move is dependant upon the electrical characteristics of the motor. Before attempting any further controlled motion, the host software should issue the command GetActivityStatus and check for the state of bit 0. When bit 0 returns 1, the chipset has completed the initialization. Refer to the *User’s Guide* and *Programmer’s Reference* for more information on this command.

2. Check commutation

After phase initialization has been completed it is useful to check the smoothness of the motor rotation in open loop mode to verify that the motor phasing initialization and commutation is correct. To do this use the following command sequence:

```
SetMotorMode 0              // set axis for open loop operation
SetMotorCommand xxxx        // xxxx is the motor command from 0 to +/-32,767 to output
Update
```

The 'xxxx' value represents the fraction of the value 32,767 of total power that will be applied to the motor. For example a value of 1,000 sends roughly 3 % (1000/32767) of the total power to the motor.

When the motor mode is set to off (SetMotorMode 0) **the motor is not under servo control. Be aware that the motor may spin rapidly after a motor command value is applied. Use small values and increase slowly.**

After this command sequence the motor should smoothly spin in one direction or the other. The motor command is a signed number and the sign controls the rotation direction. When a positive motor command is given the motor should rotate in the positive (increasing encoder counts) direction. Verify this using the command GetActualPosition. The motor should rotate at approximately the same speed in both directions when given the same motor command magnitude. For example, SetMotorCommand –2000 and SetMotorCommand +2000 should result in motion at approximately the same speed in both directions. If the motor spins roughly, in the wrong direction, or if it moves a short distance and then abruptly stops there may be a problem with the commutation. See the troubleshooting notes below and re-test.

Once the motor is spinning smoothly in both directions under open loop control, re-enable closed-loop servo control by executing the command SetMotorMode On.

Troubleshooting

When performing algorithmic initialization there are 2 potential causes of problems. Depending on the nature of the problem, either or both of the following changes will be required.

1) Encoder counting direction is incorrect. When the motor windings are energized for positive direction rotation, the motion processor expects an incrementing encoder count. If this is not the case initialization will not succeed. The motor can lock or behave in an uncontrolled fashion. To solve this problem the counting direction must be reversed. This can be achieved by using the command SetSignalSense to invert one of the encoder channels. If you intend to use the automatic phase correction feature (SetPhaseCorrectionMode), care must be taken as inverting an encoder signal may prevent index captures from occurring. The direction can also be reversed by swapping the connection of A+/A- signals for a differential encoder or by swapping the connection of the A and B signals for a single ended encoder.
2) The motor windings are out of sequence. If the motor winding connections are not in the correct sequence there will be no motion during the initialization process. In a 2 phase motor, swapping the phase A and B connections results in the motor rotating in the opposite direction. It is identical to reversing the encoder counting direction. In a 3 phase motor, reversing either A and B, or B and C will correct an out of order connection sequence.

After attempting the above changes you must re-run the initialization procedure and re-check the commutation. When operation appears correct you can move onto the next step.

3. Set Filter Parameters

For motion to occur, some amount of feedback gain must be specified. Initially use just a proportional gain (Kp) with a very low value between 1 and 25. Later you can add integral or derivative gains as well as feedforward gains if desired. The following sequence shows how to set the P, I, and D terms of the filter and how to 'update' them, making them active.

```
SetKp xxxx                  // xxxx is the desired proportional gain
SetKd yyyy                  // yyyy is the desired derivative gain
SetKi zzzz                  // zzzz is the desired integral gain
SetIntegrationLimit aaaa    // aaaa is the desired integration limit
Update                      // make thee values active.
```

It is not necessary to specify all 3 gains. Just Kp, followed by an Update can be specified initially. Kd and Ki are optional.

When exercising the motor use extreme caution. It is the responsibility of the user to observe safety precautions at all times.

At this stage, if a force is applied to the motor shaft it should respond by attempting motion in the opposite direction, thereby attempting to maintain it's initial position. If this is not the case go back and re-check commutation, making certain that the positive encoder counting direction and the motor positive rotation direction are the same.

4. Make a Trajectory Move

To test that the motor is being driven properly, set up and execute a small trapezoidal move. Specify a small distance of (for example) 5,000 counts, and a low velocity and acceleration of (for example) 10,000, and 10 respectively. With a cycle time of 153 μsec, these values correspond to roughly 997 counts/sec, and 6516 counts/sec^2, respectively.

Whatever profile values you use, be sure that they are safe for your system.

Here is the command sequence to use:

```
SetProfileMode 0            // Sets current profile mode to trapezoidal
SetPosition 5000            // 5000 is the desired destination position
SetVelocity 10000           // 10000 is the desired maximum velocity
SetAcceleration 10          // 10 is the desired acceleration
SetDeceleration 10          // 10 is the desired deceleration
Update                      // execute the move
```

After entering this sequence of commands you should see the axis smoothly move for about 6 seconds if the suggested values are used and the cycle time of the motion processor is 153 μsec.

If you do not see the axis moving, or if the axis jumps rapidly in one direction or the other, there may be a problem with the board or software settings. Re-check and review the setup procedures, as well as the parameter settings.

5. Initialization complete

When the initialization sequence is repeated, the results should be consistent. If they are not the initialization may not be correct. Re-check and contact PMD if problems persist.

If you do not have hall sensor signals, initialization is complete. Record the steps followed and incorporate them into your host processor initialization sequence.

Collecting information helpful for Hall-based initialization

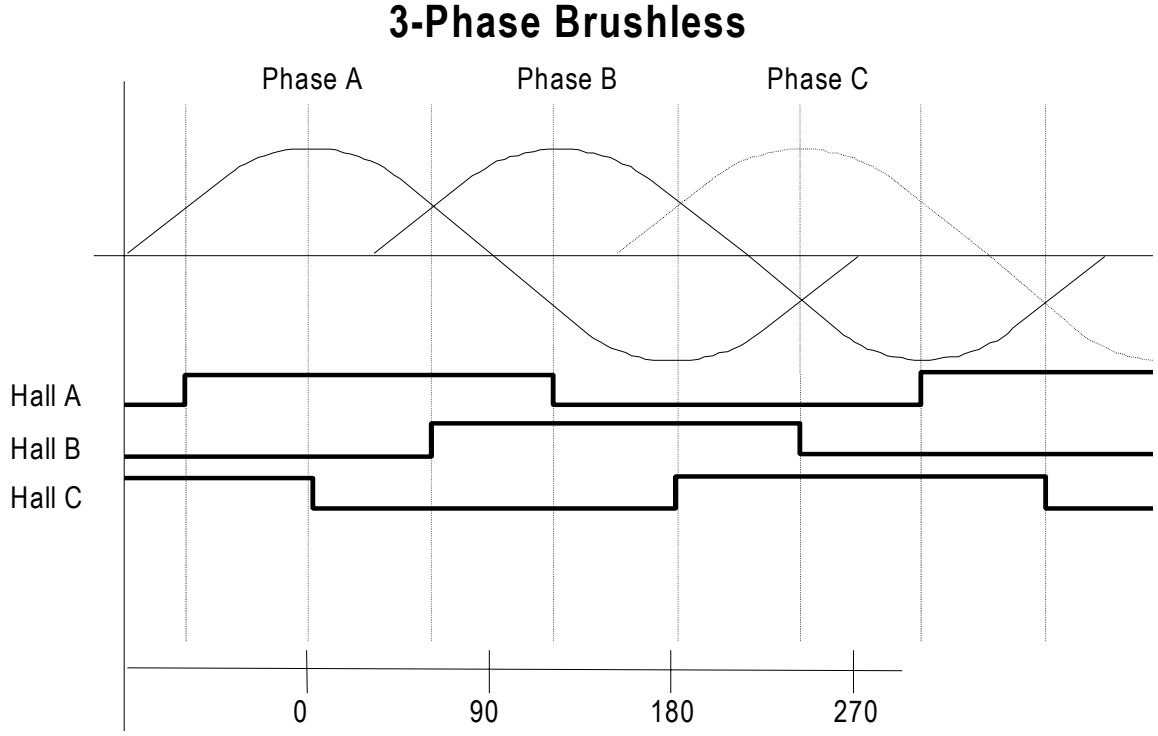
Assuming that you have an index signal present, complete the initialization procedures and program motion that results in at least full rotation of the motor in either direction. After this execute the command GetPhaseOffset and record the result. Repeat this 5 times. The value returned by GetPhaseOffset should be within the same range for successive trials. If it is not refer to the above trouble-shooting sections. This value will be used later to verify a successful hall-based initialization since this reference data should be approximately the same with both initialization methods. It is the location of the start of the first electrical cycle of phase A with respect to the index signal.

Hall-based Initialization

To set the chipset for Hall-based initialization use the command SetPhaseInitializeMode and specify Hall-based as the parameter. In this mode 3 Hall-Sensor signals are used to initially determine the motor phasing, and sinusoidal commutation begins automatically after the motor has moved through one full rotation.

The Hall-Sensor signals are fed back to the chipset through the signals Hall1A-C (axis #1) and Hall2A-C (axis #2), etc. Care should be taken to connect these sensors properly. To read the current status of the hall sensors use the command GetSignalStatus.

The following diagram shows the relationship between the state of the three Hall sensor inputs for each axis and the commutated motor outputs. This graph shows the expected Hall sensor states and winding excitation for forward motion (increasing position).



Unlike the algorithmic method, using Hall-based phase initialization no special motor setup procedures is required. Initialization is performed using the command InitializePhase, and occurs immediately, without any motor motion.

To accommodate varying types of Hall sensors, or sensors that contain inverter circuitry, the signal level/logic interpretation of the Hall sensor input signals can be set through the host. The command SetSignalSense accepts a bit-programmed word that controls whether the incoming Hall signals are interpreted as active high or active low. To read back this Hall interpretation value use the command GetSignalSense. For details on the programming of this control word see the *Programmer's Reference* manual.

Hall-based initialization should only be used with a 3-phase commutation waveform, and with Hall sensors located 120 degrees apart. Hall-sensors located 60 degrees apart should not be used.

It is imperative that the hall signals are as noise free as possible and provide a strong ground and +5V signal to the motion processor inputs.

1. Verify the hall sensor connection

The following can be performed without power being applied to the motor/amplifier. Rotation of the motor can be done manually.

In addition to connecting the hall sensors to the motion processor, connect the three hall signals to an oscilloscope and arrange them as A, B and C as shown above. The command GetSignalStatus should be used to verify the correct hall sequence as outlined above. When the motor is being manually rotated in a positive direction (the value returned by GetActualPosition is increasing), the following sequence should be observed:

6Fx
4Fx
5Fx
57x
77x
67x

The ‘x’ indicates we don’t care about the value of bits 0-3. The above values are represented in hexadecimal notation. Since the limit and AxisIn signals are also encoded in the word returned by GetSignalStatus, the exact values returned to you may be different. For example if the limit signals and AxisIn are LOW, the sequence will look like:
68x
48x
58x
50x
70x
60x

Refer to the *Programmer’s Reference* description of GetSignalStatus for a detailed explanation of the bit encoding.

Troubleshooting

If the above sequence is not observed, you may need to change the hall wiring. It is wise to perform any changes to connections in conjunction with an oscilloscope so that the signal state can be observed electrically and numerically.

1) A hall signal is inverted. Use the command SetSignalSense to invert the hall sensor that is in a state opposite to what the motion processor expects.
2) One of the hall sensors is out of sequence. Because hall transitions are expected at known points within the electrical cycle, the most common problem is that these transitions are occurring out of phase. To correct this shift the connections until the correct sequence of states as outlined above is observed. If you reference to one signal, for example HallA, swapping the connections for signals B and C should solve this problem.
3) The hall sequence is reversed. If you observe that when the encoder is counting in an incrementing fashion the above sequence is reversed, you must reverse the hall sensor connections. Swap the hall sensor connections for HallA and HallC.

2. Perform Initialization

Once you have verified the hall sensor sequence for positive motor rotation, execute the following sequence of commands.

```
SetOutputMode m              // m is 0 for DAC, 1 for PWM S/M and 2 for PWM 5050
SetNumberPhases x            // where x is 2 or 3 depending on type of motor
SetPhaseCounts yyyy          // yyyy is # of encoder counts per electrical cycle
SetPhaseInitializeMode 1     // set phase initialize mode to 'Hall-based'
InitializePhase
```

Index signal available

At this point there is no requirement to power the motor or amp. The initialization can be checked without power.

Rotate the motor at least one turn. Issue the command GetPhaseOffset and record the result. If the value returned is 65535, the index signal is not working. Refer to the start of this document for more information.

If the value recorded above is approximately equal to the value derived during algorithmic phase initialization, than the initialization may be correct. Proceed to the next step.

If the value is not equal, then it is likely that the hall sensors are out of phase. Because hall transitions are expected at known points within the electrical cycle, the most common problem is that these transitions are occurring out of phase. To correct an out of phase problem, rotate the hall connections and repeat the initialization and test procedure. That is if the starting connection sequence is:-
A B C
try
C A B
and
B C A

By using the SetSignalSense command it is also possible to shift the hall states by 180 degrees. Issue the command SetSignalSense 0x380. 0x380 is a hexadecimal number. This inverts the state of all signals, shifting the transition points by 180 degrees.

A combination of both of the above changes may be required in order to obtain a correct phase offset reading.

Index signal not available or algorithmic initialization not performed

If the index signal is not available, initialization must be checked while there is power applied to the motor/amplifier.

Proceed to the next step.

3. Check commutation

After phase initialization has been completed it is useful to check the smoothness of the motor rotation in open loop mode to verify that the motor phasing initialization and commutation is correct. To do this use the following command sequence:

```
SetMotorMode 0              // set axis for open loop operation
SetMotorCommand xxxx        // xxxx is the motor command from 0 to +/-32,767 to output
Update
```

The 'xxxx' value represents the fraction of the value 32,767 of total power that will be applied to the motor. For example a value of 1,000 sends roughly 3 % (1000/32767) of the total power to the motor.

When the motor mode is set to off (SetMotorMode 0) **the motor is not under servo control. Be aware that the motor may spin rapidly after a motor command value is applied. Use small values and increase slowly.**

After this command sequence the motor should smoothly spin in one direction or the other. The motor command is a signed number and the sign controls the rotation direction. When a positive motor command is given the motor should rotate in the positive (increasing encoder counts) direction. Verify this using the command GetActualPosition. If the motor spins roughly, in the wrong direction, or if it moves a short distance and then abruptly stops there may be a problem with the commutation. See the troubleshooting notes below and re-test. Once the motor is spinning smoothly in both directions under open loop control, re-enable closed-loop servo control by executing the command SetMotorMode On.

Troubleshooting

When performing hall-based initialization there are 2 potential causes of problems. Depending on the nature of the problem, either or both of the following changes will be required.

1) The hall sensors are out of phase. Because hall transitions are expected at known points within the electrical cycle, the most common problem is that these transitions are occurring out of phase. To correct an out of phase problem, rotate the hall connections and repeat the initialization and test procedure. That is, if the starting connection sequence is:-
   A B C
   try
   C A B
   and
   B C A
2) When the motor windings are energized for positive direction rotation, the motion processor expects an incrementing encoder count. If this is not the case initialization will not succeed. The motor can lock or behave in an uncontrolled fashion. To solve this problem the counting direction must be reversed. This can be achieved by using the command SetSignalSense to invert one of the encoder channels. If you intend to use the automatic phase correction feature (SetPhaseCorrectionMode), care must be taken as inverting an encoder signal may prevent index captures from occurring. The direction can also be reversed by swapping the connection of A+/A- signals for a

differential encoder or by swapping the connection of the A and B signals for a single ended encoder.

3) The motor windings are out of sequence. If the motor winding connections are not in the correct sequence there will be no motion during the initialization process. In a 3 phase motor, reversing either A and B, or B and C will correct an out of order connection sequence.

After attempting the above changes you must re-run the initialization procedure and re-check the commutation. When operation appears correct you can move onto the next step.

4. Set Filter Parameters

For motion to occur, some amount of feedback gain must be specified. Initially use just a proportional gain (Kp) with a very low value between 1 and 25. Later you can add integral or derivative gains as well as feedforward gains if desired. The following sequence shows how to set the P, I, and D terms of the filter and how to 'update' them, making them active.

```
SetKp xxxx                  // xxxx is the desired proportional gain
SetKd yyyy                  // yyyy is the desired derivative gain
SetKi zzzz                  // zzzz is the desired integral gain
SetIntegrationLimit aaaa    // aaaa is the desired integration limit
Update                      // make thee values active.
```

It is not necessary to specify all 3 gains. Just Kp, followed by an Update can be specified, just a Kd, etc.

When exercising the motor use extreme caution. It is the responsibility of the user to observe safety precautions at all times.

At this stage, if a force is applied to the motor it should respond by attempting motion in the opposite direction, thereby attempting to maintain it's current position. If this is not the case go back and re-check commutation, making certain that the positive encoder counting direction and the motor positive rotation direction are the same.

5. Make a Trajectory Move

To test that the motor is being driven properly, set up and execute a small trapezoidal move. Specify a small distance of (for example) 5,000 counts, and a low velocity and acceleration of (for example) 10,000, and 10 respectively. With a cycle time of 153 μsec, these values correspond to roughly 997 counts/sec, and 6516 counts/sec^2, respectively.

Whatever profile values you use, be sure that they are safe for your system.
Here is the command sequence to use:

```
SetProfileMode 0          // Sets current profile mode to trapezoidal
SetPosition 5000          // 5000 is the desired destination position
SetVelocity 10000         // 10000 is the desired maximum velocity
SetAcceleration 10        // 10 is the desired acceleration
SetDeceleration 10        // 10 is the desired deceleration
Update                    // execute the move
```

After entering this sequence of commands you should see the axis smoothly move for about 6 seconds if the suggested values are used and the cycle time of the motion processor is 153 μsec.

If you do not see the axis moving, or if the axis jumps rapidly in one direction or the other, there may be a problem with the board or software settings. Re-check and review the setup procedures, as well as the parameter settings.

When the initialization sequence is repeated, the results should be consistent. If they are not the initialization may not be correct. Re-check and contact PMD if problems persist.