USER'S MANUAL

INTELLIGENT MOTOR CONTROLLERS

PC68 FAMILY

OREGON MICRO SYSTEMS, INC.

TWIN OAKS BUSINESS CENTER
1800 NW 169th PLACE, SUITE C100
BEAVERTON, OR 97006
PHONE 503-629-8081
FAX 503-629-0688
EMAIL sales@OMSmotion.com
WEB SITE www.OMSmotion.com

COPYRIGHT NOTICE

TRADEMARKS

IBM, IBM PC, IBM PC/XT, IBM PC/AT, IBM PS/2 and IBM PC DOS are registered trademarks of International Business Machines Corporation.

DISCLAIMER

Oregon Micro Systems, Inc. makes no representations or warranties regarding the contents of this document. We reserve the right to revise this document, or make changes to the specifications of the product described within it at any time without notice and without obligation to notify any person of such revision or change.

3301-1000000
Revision G

TABLE OF CONTENTS

5. EXPANSION BOARDS

6. COMMAND STRUCTURE

7. HOST SOFTWARE

8. SERVICE

A. LIMITED WARRANTY

B. TECHNICAL INFORMATION / RETURN FOR REPAIR PROCEDURES

C. SPECIFICATIONS

INDEX

This page intentionally left blank

1. GENERAL DESCRIPTION

1.1. INTRODUCTION

The OMS PC/104 family of products is built around the base controller, the PC68, and can expand to satisfy most any motion control application. The PC68 intelligent motion controller is a 100% PC/104 (IEEE P996.1 specification) compliant controller. It is capable of fully supporting the PC/104 ISA bus, as well as the RS-232 interface. The PC68 controller can manage four axes of stepper, servo or a combination of both types of motion and can be expanded with a daughter board to support eight axes. It can manage coordinated or independent motion of each or all of the axes simultaneously. With high level functionality, such as circular and linear interpolation, multi-tasking, custom profiling, etc., the PC68 can satisfy most any motion control application.

In a PC/104 computer stack of boards the PC68 functions as a motion coprocessor. Through the RS-232 interface the PC68 operates as a stand-alone controller. It utilizes a 32-bit microprocessor and patented, proprietary technology to control the trajectory profile, acceleration, velocity, deceleration and direction of selected axes. In response to commands from the host computer, the PC68 controller will calculate the optimum velocity profile to reach the desired destination in the minimum time while conforming to the programmed acceleration and velocity parameters.

The stepper control of the PC68 produces a 50% duty cycle square wave step pulse at velocities of 0 to 1,044,000 pulses per second and an acceleration of 0 to 8,000,000 pulses per second, per second. The servo control utilizes a 16-bit DAC and outputs either +/- 10V or 0 to +10V. The encoder feedback control can be used as feedback for the servo PID, position maintenance for the stepper axes or as strictly a position feedback of any axis. The encoder input supports either differential or single ended quadrature TTL signals at a rate of up to 4MHz and counts at a 4 times resolution. This means a 1000 line encoder will produce 4000 counts per revolution in the PC68 controller.

The PC68 is commanded using virtually any programming language to pass simple ASCII command strings to the PC68 through either the PC/104 I/O address or RS-232 port. For a typical motion requirement of 1,000,000 counts at 400,000 counts/sec and an acceleration of 500,000 counts/sec/sec the following string would be sent from the host computer to the PC68:

VL400000; AC500000; MR1000000; GO

For additional command programming examples see Section 6.

The OMS PC/104 family of products consist of the PC68 and a series of expansion boards that essentially are daughter boards to the PC68 that pass the PC/104 ISA bus signals straight through. OMS has defined a proprietary board to board bus that joins the PC68 with the expansion boards. Each module provides defined functionality, such as four additional axes of control, user definable I/O, etc. The expansion boards provide the

feature that you pay for what you need, rather than paying for additional features you don't need. The list of expansion boards is lengthy and OMS will gladly entertain the possibilities for custom modules for large OEM applications.

1.2. SYSTEM OVERVIEW

The PC68 is only 3.55" x 3.775" and is intended to be embedded into a stack of PC/104 boards where it behaves as a slave expansion board to the ISA bus of the PC/104 CPU, or used as a stand-alone controller where it communicates as a slave device through an RS-232 port. The PC68 utilizes Flash Memory where programs (command sequences) can be stored permanently as macros.

The PC68 utilizes the Motorola 68332 32-bit microcontroller and FPGA technology for extensive logic integration and flexibility. The firmware, which resides in Flash Memory, can be upgraded through either of the communication interfaces, RS-232 or PC/104, without having to remove the controller from the system, if you have a 32 Bit Operating System such as Windows or Windows NT. All signals to and from the PC68 are buffered through TTL devices and are found on the shielded SCSI-3 type connector J5. The IO68 is a connection board that is used to route the signals of the PC68 to individual screw terminal blocks. Both the PC68 and IO68 utilize a resettable fuse on the +5V on the SCSI connectors for protection.

As a stand-alone controller the PC68 obtains its power from connector J3 and the RS-232 communication port connects to J4. The PC68 supports baud rates of 300 to 38.4K baud. With the ability to store macros into Flash Memory the PC68 could essentially be programmed once and then be embedded into a machine where it could run independently.

When the PC68 is configured into a PC/104 stack it supports four I/O address registers that provide near real-time information. The data communication is performed by sending and receiving strings of data through the data port register. The status register provides that handshaking information for writing to the data register as well as some status information including error conditions, motion complete, etc. The PC68 can generate an interrupt to the PC/104 host and the conditions that cause interrupt can be individually selected by writing to the control register.

More details on each of the communication interfaces as well as the functionality of the controller are included in the following chapters.

2. GETTING STARTED

2.1. PREPARE FOR INSTALLATION

The installation of the PC68 board is flexible and dependent on the type of communication interface being used, either the RS-232 or PC/104 bus.

Minimum Requirements:

For RS-232:

1. Controller must be ≥ 0.1" from the top of the surface it's mounted on.
2. +5V power must be supplied. This can be done through the on-board 6 pin connector.

For Servo Models:

+/-12V power supplies are also required. This can be done through the on-board 6 pin connector.

For PC/104:

1. Install PC68 in a PC/104 system that complies with the PC/104 bus specification.
2. Set the PC68 to use IRQ and I/O Address Settings that do not conflict with other devices in the system.
3. The power for +5, +12 and -12 volts is provided through the PC104 Bus connector.

For All PC68 Models:

Allow for proper ventilation, including forced air around the PCB.

2.2. TO PREPARE FOR THE INSTALLATION INTO A PC/104 STACK

Before installing the PC68 into a PC/104 stack you must first:

- Set the I/O Address Range (default is 300 [hex]); discussed in section 2.3
- Set the Interrupt Request (IRQ) (default is IRQ5); discussed in section 2.4

Figure 2-1 illustrates the locations of the switches on the PC68 for the default address and IRQ setting.

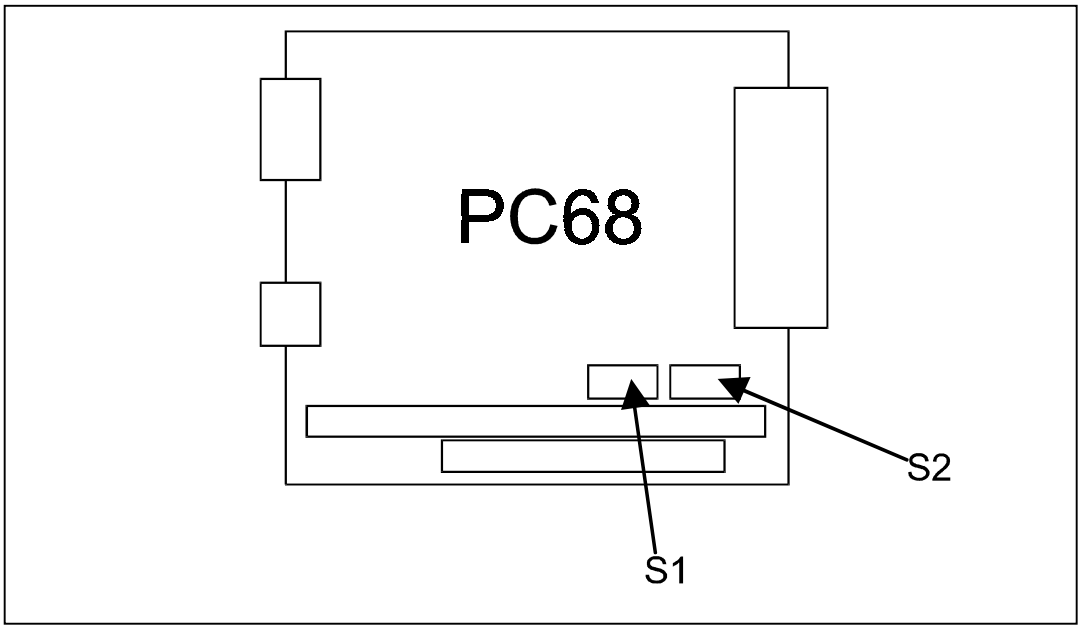

FIGURE 2-1 DEFAULT ADDRESS AND IRQ SETTING

2.3. **SET THE I/O ADDRESS RANGE**

The PC68's I/O address range is selected with the switch (S2) located on the top of the board. The PC68's I/O address ranges from 200 to 3FF (hex). The default address is 300 - 303 (hex).

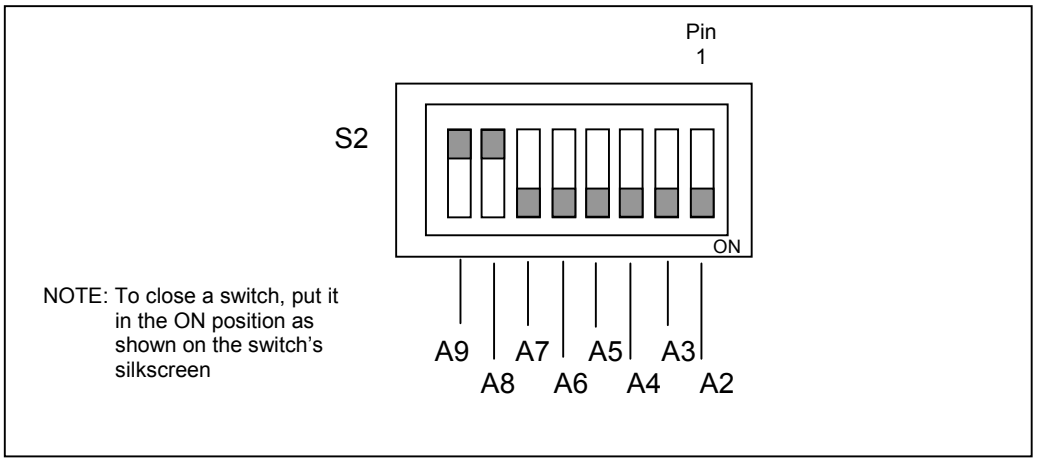

FIGURE 2-2 ADDRESS SELECTED SWITCH (Default setting)

The A0 and A1 address signals of the host are decoded internally by the PC68 and are treated as 0 for base address calculations. A closed switch sets that bit in the address low and an open switch sets the bit high. For example, an open on A9, A5 and A4 with all other switches closed selects base address 230.

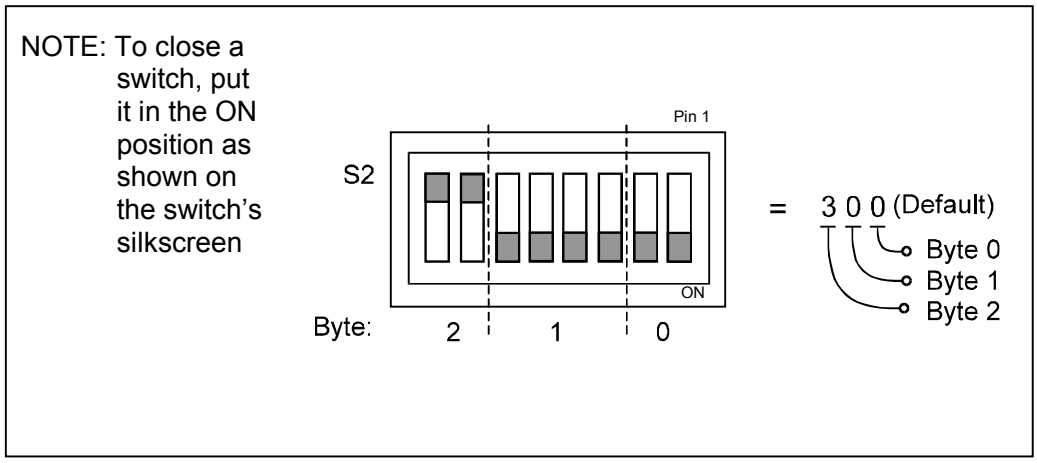

FIGURE 2-3 PC68 I/O ADDRESS

Byte 2		Byte 1		Byte 0	
MSB LSB		MSB LSB		MSB LSB	
0 0 = 0	(Not Recommended!)	0 0 0 0 = 0	(Default)	0 0 = 0	(Default)
0 1 = 1		0 0 0 1 = 1		0 1 = 4	
1 0 = 2		0 0 1 0 = 2		1 0 = 8	
1 1 = 3	(Default)	0 0 1 1 = 3		1 1 = C	
		0 1 0 0 = 4			
		0 1 0 1 = 5			
		0 1 1 0 = 6			
		0 1 1 1 = 7			
		1 0 0 0 = 8			
		1 0 0 1 = 9			
		1 0 1 0 = A			
		1 0 1 1 = B			
		0 1 0 1 = C			
		1 1 0 1 = D			
		1 1 1 0 = E			
		1 1 1 1 = F			

FIGURE 2-4 I/O ADDRESS SWITCH CONFIGURATION

2.4. SET THE INTERRUPT REQUEST (IRQ)

The interrupt level that the PC68 board uses for communication is selected with switch S1. The default setting is IRQ5. Close the specific switch to select the desired IRQ. All open IRQ switches disable the interrupt capability. There should never be more than one IRQ switch closed at the same time on S1.

The switch on the far left of S1 is used to determine PC/104 or RS-232 communication mode. When communicating through the PC/104 bus, this switch should be open (off). When communicating to the PC68 via RS-232, this switch should be closed (on).

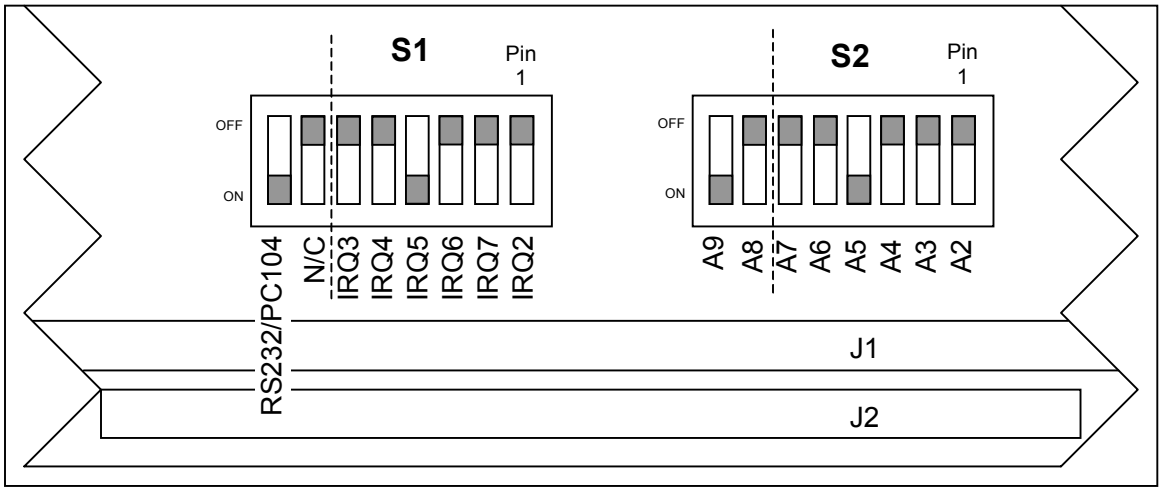

FIGURE 2-5 INTERRUPT SWITCH (default setting for a PC68-42)

2.5. INSTALL THE PC68 INTO A PC/104 SYSTEM

Reference installation instructions provided in text files (i.e. README.TXT) that came with the support software disk.

Turn off power to the PC/104 computer, and disconnect its power cord from the wall socket.

Align the PC68 on to the PC/104 J1/J2 connector, ensuring that the board is lined up correctly and push the boards together.

Make sure that the four standoffs are aligned with the four holes on the PC68.

Double check the board to ensure it is properly seated in the connector.

Either screw another set of standoffs in to the standoffs on the lower board or use screws to secure the PC68.

Replace the power cord, and turn on the computer.

Allow the computer to boot up.

For ease in troubleshooting, do not connect the PC68 to other parts of the system until communication is established with the host.

You can test that communications are working by running PC68BCOM.EXE from a command prompt. Otherwise, you can run Terminal or Executor from the OMS-EZ™ Suite to test communications to your PC68. Using one of these programs, send a "WY" to the controller. If you get a response that identifies your controller model and its firmware version, then you have a communication link. If you need assistance, contact Oregon Micro Systems.

2.6. CONNECTING THE PC68 FOR RS-232 COMMUNICATION

The first requirement for communication through the RS-232 interface is to insure that the PC68 is securely and safely mounted where damage is unlikely. This includes the exposure to possible static discharge, moisture, debris, etc. If the specific PC68 model is equipped with the PC/104 J1/J2 connectors then special mounting efforts may be required to protect the extended pins on the bottom of the PC68.

CAUTION:

> The PC68 is a static sensitive device and standard ESD (Electro Static Discharge) techniques are required when handling and installing the PC68.

The RS-232 communication port is a DTE com device so that straight connection can be used for communication; RxD to TxD, TxD to RxD. Two handshake signals are supported, CTS and DTR, that can also be connected straight through. Baud rates of 300, 600, 1200, 2400, 4800, 9600, 19200 and 38400 are supported. The 9600-baud rate is the factory default. Any terminal device that supports these signals and baud rates, be it a computer, dumb-terminal, etc., can be used to communicate to the PC68. Most any terminal software can be used to communicate interactively to the PC68, although the explanation in this manual is limited to the program supplied by OMS.

Select an unused COM-Port, COM1, COM2, etc., on the computer or terminal to be used. Connect a straight-through 9-pin RS-232 cable between the host terminal and the PC68. To prevent motors, switches or other devices from unexpected activation do not connect the cable to output connector J5 at this time.

Connect a +5VDC, 1 Amp power source to the power connector at J3. (See Table 2-1 for the connector pin-out)

NOTE: +/- 12VDC is required only for servo operation.

TABLE 2-1 CONNECTOR J3

	PIN	PIN	
+5V DC	5	6	Digital Ground
-12V DC	3	4	No Connect
+12V DC	1	2	Analog Ground

Reference the support software disk to find a serial communications utility that will work with the PC68.

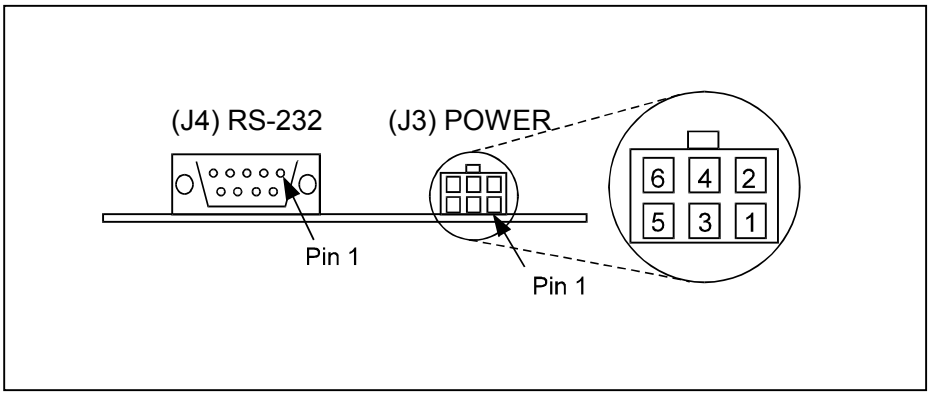

FIGURE 2-6 CONNECTOR CONFIGURATION

2.7. **CONNECT** AND CHECKOUT THE SERVO SYSTEM

Servo systems tend not to respond gracefully to connection errors. You can reduce the chance of making connection errors by following a step-by-step procedure:

Caution

The servo motor may jump or spin at a very high velocity during connection and configuration. The motor should be restrained via fastening to the physical system or by some other means before beginning this procedure. Keep hands and clothing clear of the motor and any mechanical assemblies while performing this procedure.

2.7.1. CONNECT AND CONFIGURE THE MOTOR/AMPLIFIER

1. Connect and configure your amplifier per the manufacturer's instructions (for "Torque" or "Open-Loop" mode).

2. With the motor and amplifier power turned off, connect the PC68 to the amplifier. Do not connect the encoder yet.

3. Balance your motor:

   a. Using a voltage meter, verify that the command signal from the PC68 is less than (500mV. If it is not, send the command “DZ0;” to the PC68 and recheck the voltage. If the voltage is still too high, contact Oregon Micro Systems’ Technical Support department for guidance.

   b. Turn on power to the amplifier and then to the motor.

   c. Adjust the balance setting of your amplifier (if equipped) until the motor stops moving.

   d. If the motor continues to revolve or your amplifier has no balance adjustment:

      i) Send the command “DZ100;” to the PC68.

      ii) If the motor spins faster, reduce the command parameter and resend the command, e.g. “DZ50;”.

      iii) If the motor spins slower but does not stop, increase the command parameter and resend the command, e.g. “DZ150;”.

      iv) Continue adjusting and resending the DZ command until the motor comes to rest. Write down the final DZ value for later reference as your “zero” setting.

4. Maximize your system’s usage of the PC68’s DAC:

   a. Connect the servo encoder to the PC68.

   b. Set the signal/command gain of your amplifier to it’s minimum setting.

   c. Send the “DZ3277;” command to the PC68 and observe the velocity of the motor. The output of PC68 will be near 1VDC.

   d. If the motor does not move at all, your amplifier does not work well at a low velocity. In this case, adjust the signal/command gain of the amplifier to approximately 20% of maximum or until the motor begins to move.

   e. Using a frequency meter, measure the pulse rate of Phase A of the encoder. The frequency measured is ¼ of the actual pulse rate.

   f. Adjust the signal/command gain of the amplifier until the pulse rate of Phase A (4 is approximately 10% of your desired peak operational velocity. If the pulse rate is already greater than 10% of peak, your amplifier is not designed for low velocity motion and you will likely have some difficulty tuning your motors.

   g. Send the “DZ-3277;” command to the PC68 and recheck the velocity. You may need to readjust your amplifier. If so, do not reduce the signal/command gain – only increase the setting as needed. Increasing the gain will not impair the forward peak velocity but reduction will.

   h. Send the DZ command with the “zero” value noted at the end of step 3d (iv) to the PC68. Send the same value using the KO command, e.g. “KO-175;”.

5. Verify the direction of your servo encoder:
   a. Send the “LP0;DZ2000;” command to the PC68.
   b. Send the “RE” command to the PC68 and observe the response.
   c. If the response is positive, no further action need be taken; go to step 6.
   d. If the response is negative, your encoder must be reversed.
      i) If your encoder produces a differential signal, swap Phase B with Phase B-not and repeat from step (a.) above.
      ii) If your encoder produces a single-ended (or TTL) signal, swap Phase A with Phase B and repeat from step (a.) above.
   e. If the RE response is still negative, contact OMS Technical Support for assistance.
6. Repeat from step 1 for the other servo axes.
7. Remember to set DZ and KO for each axis at every power-up unless you store the values in flash (see Section 6 COMMAND STRUCTURE)

NOTE: Most encoder problems are caused by lack of power or incorrect connections. If the encoder position changes by only 1 count, this is an indication that one of the phases is not connected.

Do not proceed until you perform all the steps in this procedure, ensure that the outputs of the PC68 are as described, and ensure that the encoder is operating correctly

2.7.2. TUNE THE SYSTEM

2.7.2.1. Introduction

Tuning a servo system is the process of balancing the conflicting requirements to achieve optimum performance of a real world system.

The first of these requirements is that of accuracy. In a closed loop system, an error signal is derived, then amplified, then supplied to the motor to correct any error. Clearly, if a system is to compensate for infinitely small errors, the gain of the amplifier needs to be infinite. Real world amplifiers do not possess infinite gain; therefore, there is some minimal error which cannot be corrected. In order to have the greatest possible accuracy, the gain needs to be as high as possible. Unfortunately, other real world considerations limit the maximum gain of the system.

The second of the requirements is that of stability. The system must not be unstable, e.g. oscillate. The degree to which a system is stable affects its performance. The effects can be seen when looking at the system’s response to a step change at the input. The step response falls into one of three categories: under damped, critically damped, over damped. Over damped systems are slow to reach their final value. Critically damped systems reach final value quickly, without overshoot. Under damped

systems reach final value quickly, but have various degrees of “ringing” that decay to zero.

The third requirement is that of bandwidth. The system should respond to the highest input frequency possible. The motor/load combination is the predominant feature of the open loop bandwidth. In the closed loop situation, the amplifier attempts to compensate for the limited response characteristics of the motor load. Increasing gain extends the closed loop bandwidth at the expense of stability.

2.7.2.2. Manual Tuning

In most all motion control applications the optimum tuning of the servo system is achieved through a manual tuning process. Auto-tuning algorithms typically can only get the system parameters close and require manual steps to fine tune the parameters. An empirical trial and error approach will be discussed first.

NOTE: You may want to use the OMS software to help during this process. It is capable of capturing the actual data and plotting it in reference to an ideal motion profile.

There are some system parameters that need to be defined to before attempting to tune a motor. The encoder resolution, counts per revolution, is one element to be determined. Another is the systems maximum velocity. Note that a motor should never exceed 90% of the motor’s top rpm. If the system requirement is for a velocity higher than 90% of the motors top rpm, then another motor with higher rpm capability is to be used.

The system’s maximum acceleration is determined a couple of different ways. The best method is to determine the system time constant, which includes “hitting” or “bumping” the motor under system load and measure the time from 0 rpm to maximum rpm and divide this value by 5. The maximum acceleration is either 2.5 times this value, or is based on the system requirements for handling the load as defined in the operating specifications of the system. This value is always lower than the calculated value and if this acceleration value is not high enough then a different motor/amplifier with more power or band-width should be utilized.

The PC68 can control either current mode or voltage mode amplifiers. The servo update rate of the PC68 is 488μs, for four axes. High following errors can be compensated for using the feedforward coefficients explained later in this section. There are some general formulas that have been developed to determine acceptable following error for both current and velocity mode systems:

Current mode following error for KP = (3°/360°) × (counts per revolution)

Voltage mode following error for KP = (90°/360°) × (counts per revolution)

It is obvious that the voltage mode allows for much greater following errors than the current mode. This value is the following error when the motor is at peak velocity and will be used when determining the proportional gain (KP).

The following error for the integral term (KI) or long-term gain value will follow the following guidelines:

Current Mode following error for KI = 0 counts

Voltage Mode following error for KI = 80° of 360° (expressed in motor counts)

1. While still in open-loop mode, hold off (HF), use the DZ command to zero the motor. This variable is used to provide a constant output that will compensate for any torque offset from the load. So, when the system should be stationary, the necessary voltage will be sent to the amplifier to cause the motor to maintain position. With the correct DZ value, the motor should successfully maintain a zero position.

   KO is the offset coefficient used while in closed-loop mode, hold on (HN). KO is essentially the same as DZ, but used for closed-loop operation. Once you have determined the correct value for DZ, this same value should be used for the KO variable before beginning to tune the PID filter.

   The values for DZ and KO range from –32640 to 32640.

2. Set the known values for velocity, acceleration and the move distance for a trapezoidal profile with at least a 20% flat spot at peak velocity. Formula:

   Profile distance = ((peak velocity)^2/(2×acceleration))×2.4

Example: ((50,000)^2/(2×500,000))×2.4 = 6,000

3. Execute the move by sending the move commands to the PC68.

Example: MR6000 GO

4. Adjust the KP term while repeating step 2 until the following error at the flat spot of the profile is acceptable. If the motor becomes unstable prior to obtaining the optimum KP term than increase the KD term until the motor stabilizes.

Example:
```
LP0 KP3 HN MR6000 GO
LP0 KP10 HN MR6000 GO
LP0 KP25 HN MR6000 GO
LP0 KD100 HN
LP0 KP35 HN MR6000 GO
LP0 KD125 HN
```

The values in the above example are totally arbitrary and may vary drastically with different systems. The LP0 command is used to set the position error to 0.

The values for KP range from 0 to 4096.

5. Once the KP term has been obtained, then continue executing the motion while raising the KI term until the long-term following error is acceptable. This error can be measured at the two knees of the motion profile. By increasing the KI term, the response time of your system will increase. The motion profile should have a steeper slope as KI increases.

   However, as KI increases the system can also become unstable. When the instability becomes unacceptable, increase the KD parameter. This will increase the dampening on the system's motion profile (therefore reducing oscillation, or "ringing"). Continue adjusting the KI and KD terms until the proper response time is obtained.

   The values for KI range from 0 to 4096.

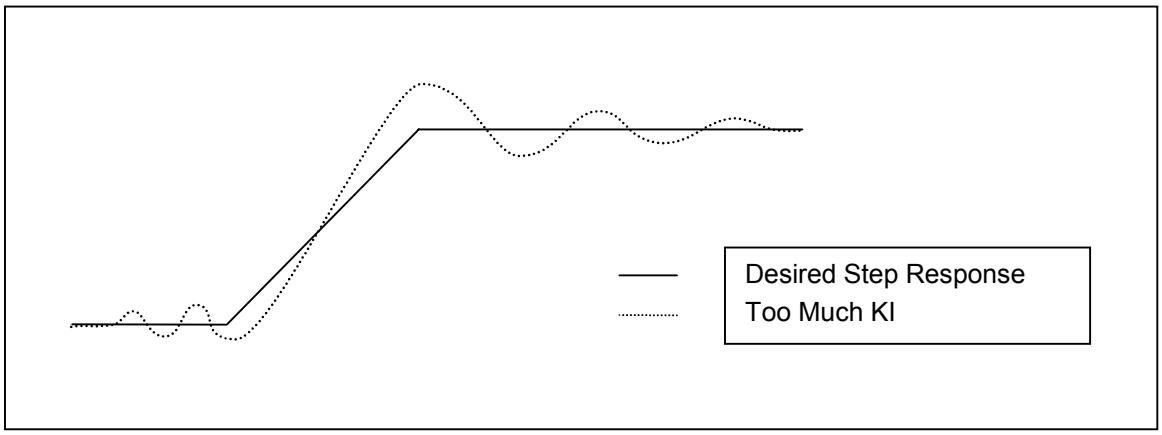

FIGURE 2-7

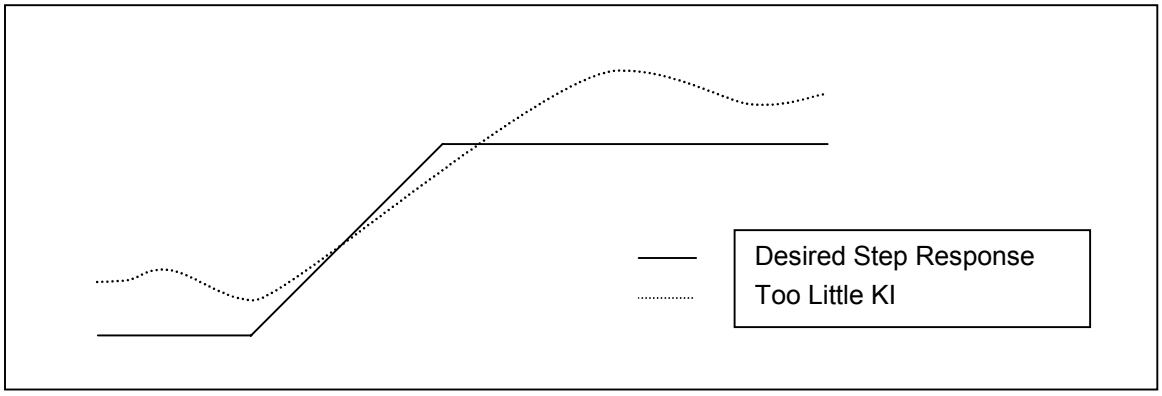

FIGURE 2-8

6. If you are getting too much "ringing" in the motion profile, then increase KD to help dampen the system's response. If, instead, the system is over-damped and is reaching the final velocity too slowly, then reduce the KD parameter. Optimally, the system's motion profile should show

the motor reaching the desired velocity as quickly as possible without overshoot and oscillation ("ringing").

The values for KD range from 0 to 4096.

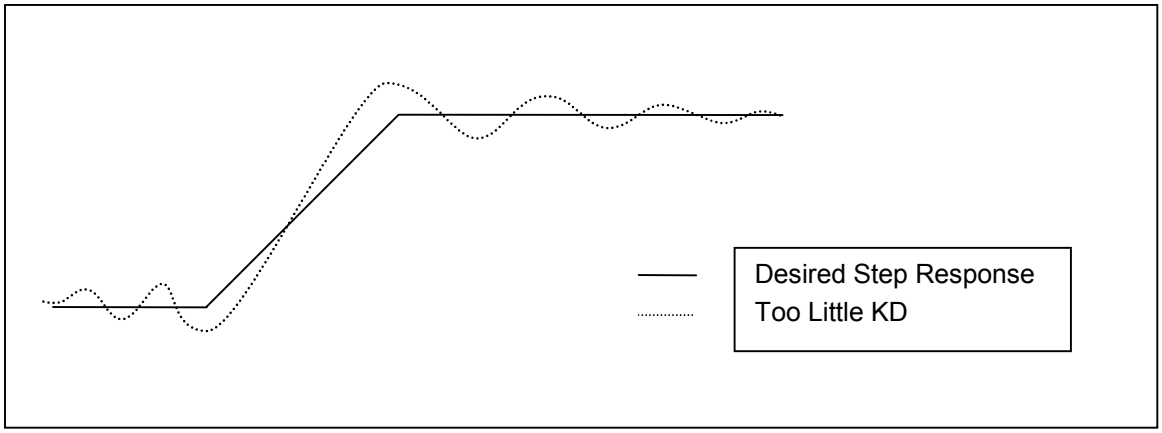

FIGURE 2-9

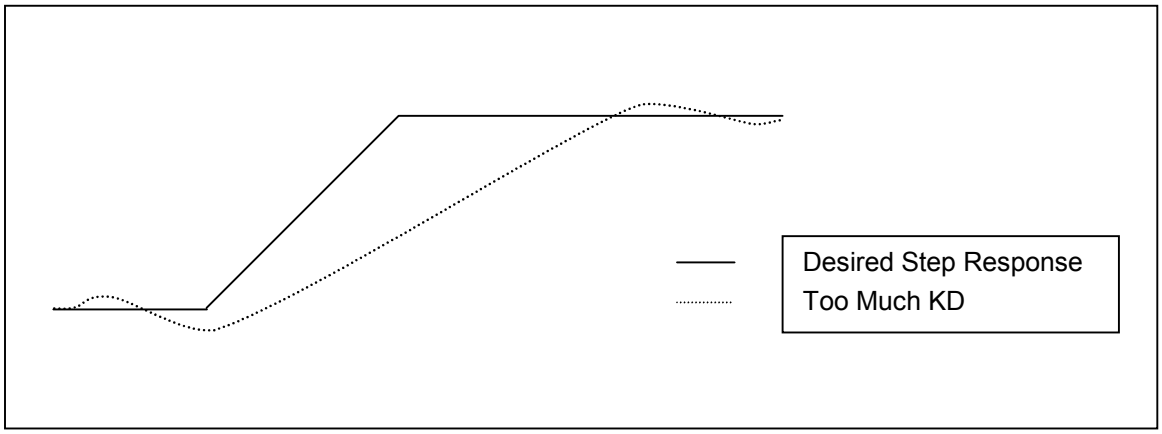

FIGURE 2-10

7. KP, KI, and KD are the primary parameters of concern when tuning a servo system. Once the optimum values for these variables have been determined, then you can adjust some of the secondary parameters that will help fine tune your system's performance. These other variables are described in the subsequent steps.

8. The KV variable is used when tuning velocity controlled servos (voltage mode servo amplifiers). This is the velocity feedforward coefficient. KV determines how closely the system follows the desired constant velocity portion of the motion profile. By increasing this term, the following error of the system's response can be minimized. However, too large of a value may result in unstable behavior after command velocity changes.

The values for KV range from 0 to 4096.

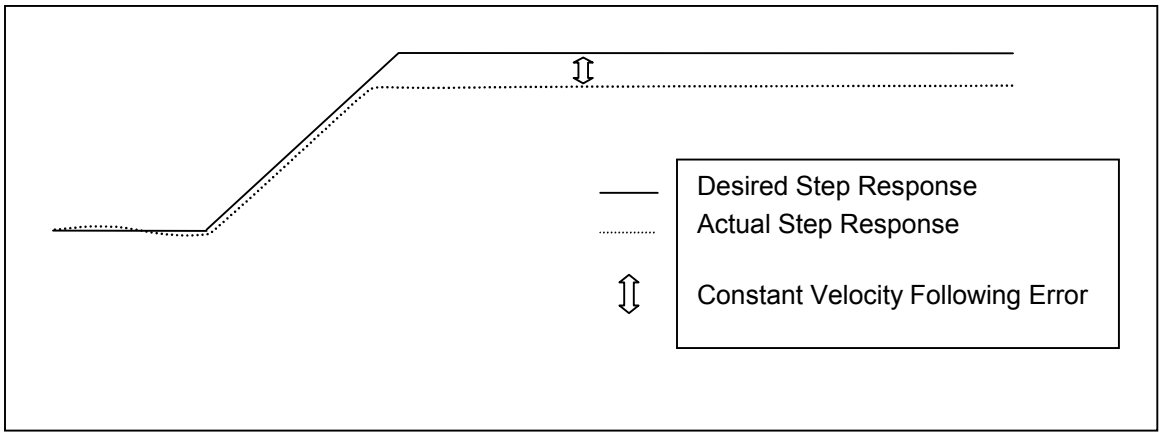

FIGURE 2-11

9. The KA variable is used when tuning torque controlled servos (current mode servo amplifiers). This is the acceleration feedforward coefficient. Systems with high inertial loads may require additional torque during acceleration or deceleration to achieve optimum performance. KA determines how closely the system follows the desired acceleration and deceleration portions of the motion profile. Increasing this term reduces the following error occurring during acceleration and deceleration of the system. Although, if KA is too large, instability may occur.

   The values for KA range from 0 to 4096.

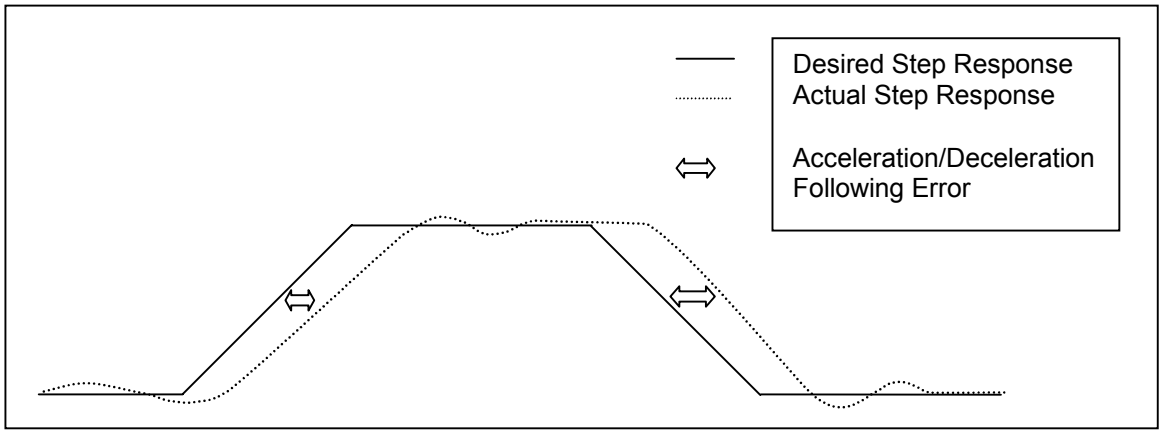

FIGURE 2-12

The block diagram below describes the feedback loop that is taking place in the servo system:

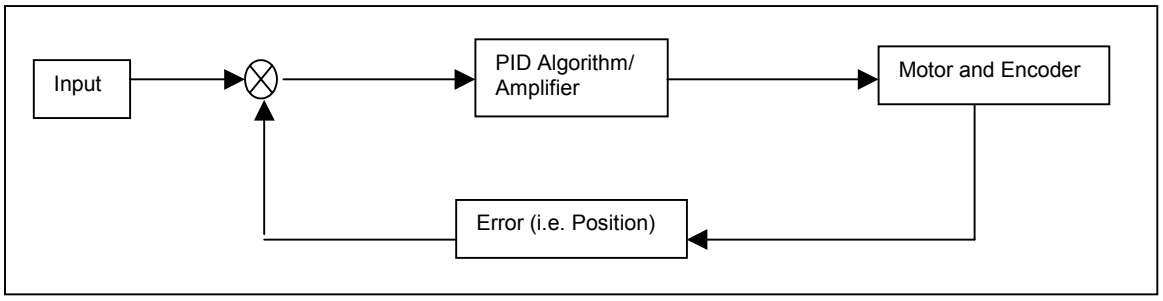

FIGURE 2-13 FEEDBACK LOOP

10. You may want to save the values for KP, KI, KD, etc., for future reference. These values can be saved in the board's Flash Memory, so they can be accessed easily on reset or power-up. The command AP will store your current parameter assignments, such as KP, KI, KD, etc., into flash memory. These saved parameters will then be used as the power up default set of values. Refer to page 6-24 for more detailed information regarding how to use the commands to save and load parameter sets from flash memory.

11. Once the KP term has been obtained then continue executing the motion while raising the KI term until the long-term following error is acceptable. This error is to be measured at the two knees of the profile. If the motor becomes unstable before obtaining the optimum following error than increase the KD term until the motor becomes stable.

12. To verify that your motor is tuned properly after you have completed the first 11 steps perform the following test to test the holding torque: Send lp0;HN commands and check the shaft of the motor to make sure it is stiff. If there is play in the motor shaft when you turn it then you may have to re-adjust your PID filter.

13. Once you are satisfied with the static holding torque you could check for position error. Send the command "ac100000;vl5000;mr64000;go". With an 8000 line encoder this move would be equivalent to 8 revolutions of the motor. After the move is complete check the position error by sending the RE and RP commands for the specific axis you are moving. Compare the difference in the two responses. If they are the same then you are on the right track, if the error was greater than 32768 than the controller will disable the PID so that you don't have a runaway motor and major changes to the PID parameters may be required. For minor differences in the encoder and the position reading you can fine-tune your PID filter according from the earlier steps.

2.8. SETTING THE USER DEFAULT CONFIGURATION

There are several parameters that can be defined by the user as default. These parameter values supersede the factory default values and are stored in Flash Memory for power-up configuration. Most of these parameters consist of axis specific values, i.e. velocity, acceleration, limit switch logic sense, etc. The configuration of the User I/O must be configured once the communication interface has been established.

2.8.1. SETTING THE USER I/O DEFAULTS

The factory default of these signals sets I/O bits 0 through 3 as inputs and 4 through 7 as outputs. If these signals are to be used in a configuration other than the factory defaults then they must be reconfigured before any hardware is connected. If an input device, such as a position sensor or limit switch, is attached to a User I/O signal configured as an output then the logic gate on the PC68 may be destroyed and additional damage to the PC68 could occur. The "IO#" command is used to change the User I/O from input to output and visa-versa. User I/O bits are configured in blocks of four. The syntax for the IO command uses a 0 to set the corresponding four I/O bits to inputs and a 1 to set them to outputs.

Example: IO0,1;

This will set I/O bits 0 – 3 as inputs and 4 – 7 as outputs (factory default).

Example: IO1,1;

This will set all bits (0 – 7) to outputs.

Refer to page 6-48 for more on the IO command.

The "RB" Report Bits command is used to check the configuration of the I/O bits. This command should be executed to ensure that the configuration of the I/O bits is as required by the system. This should be performed PRIOR to connecting hardware to the User I/O signals of the PC68. Refer to page 6-50 for more on the RB command.

The eight User I/O bit all have a 2.2k Ohm pull-up resistor connected to them. By default the active state of the inputs is a low true. When the I/O bits are configured as inputs it is only necessary to have a switch closure to ground to activate the input. The "BX" command is used to check the status of the input bits. The use of a 100-Ohm pull-down resistor can be used to change the state of the inputs for testing. Refer to page 6-49 for more on the BX command.

The "BH" Bit High and "BL" Bit Low commands are used to toggle the logic state of the User I/O bits. Before connecting the User I/O signals to exterior hardware, you should be certain that the bits are configured correctly. An LED connected through a 100 Ohm resistor to +5V can be used to test the I/O bits when toggling them. Refer to page 6-47 for more on the BH and BL commands.

2.8.2. OTHER USER DEFINABLE DEFAULT PARAMETERS

The PC68 comes from the factory with default values for all parameters. For instance, the default value for the velocity of all axes is 100,000 counts per second. (A count is equivalent to a step pulse or one count of an encoder.) In a typical application, when the system is powered up, the main host computer would initialize all of the peripherals, such as the PC68, sending to each of the axes the peak velocity. When the User Definable Default Parameter value is defined for the velocity then the initialization of the system can skip initializing the velocities of the defined axes. This feature can greatly simplify the software and initialization process.

Once the values for all of the associated parameters are defined, i.e. velocity, acceleration, PID values, etc. then the "AP" Archive Parameters command is executed to place the values into Flash Memory. From this point forward these defined values will be used after reset or power-up. The individual parameters can be over-written at anytime by using the associated command, i.e. VL#, AC#, etc. To restore the factory defaults the command "RF" Restore Factory defaults is executed. To restore the User Defined Default Parameters the command "RD" Restore Defaults is executed. Refer to Section 6 COMMAND STRUCTURE for more information on these commands.

The following is a list of parameters that can be defined as part of the User Definable Power-Up Default Parameters.

- Baud Rate for serial communication
- I/O bit configuration
- Overtravel limit (soft limit or hard limit)
- Overtravel limit (enabled or disabled)
- Overtravel limit polarity (active high or active low)
- Software based overtravel for each axis
- Direction Bit polarity
- Acceleration value for each axis
- Trajectory profile for each axis (linear, parabolic, S-curve, custom)
- Velocity Peak
- Velocity Base
- User Unit values for each axis
- Auxiliary output settle time for each axis
- Automatic auxiliary control axis by axis
- Encoder Ratio for each axis
- Encoder Slip tolerance for each axis
- Position Maintenance Dead-Band, Hold Gain and Hold Velocity.
- Servo axis unipolar/bipolar output
- Servo PID values: KP, KD, KI, KO, KV, KA
- Servo zero value: DZ

NOTE: Use the AP command sparingly since it writes to Flash Memory. There is a finite number of times the Flash can be re-written (i.e. less that 10,000 times, typical.)

2.9. POWER SUPPLY REQUIREMENTS

The PC68 is designed to operate from the power supplied by to the PC/104 bus which must be capable of supplying +5V at 1 amp typical to the PC68. Servo models of the PC68 require +/-12V at 0.1 amp typical.

CAUTION

Under no circumstances should the PC68 be installed in or removed from the PC/104 bus with the power on.

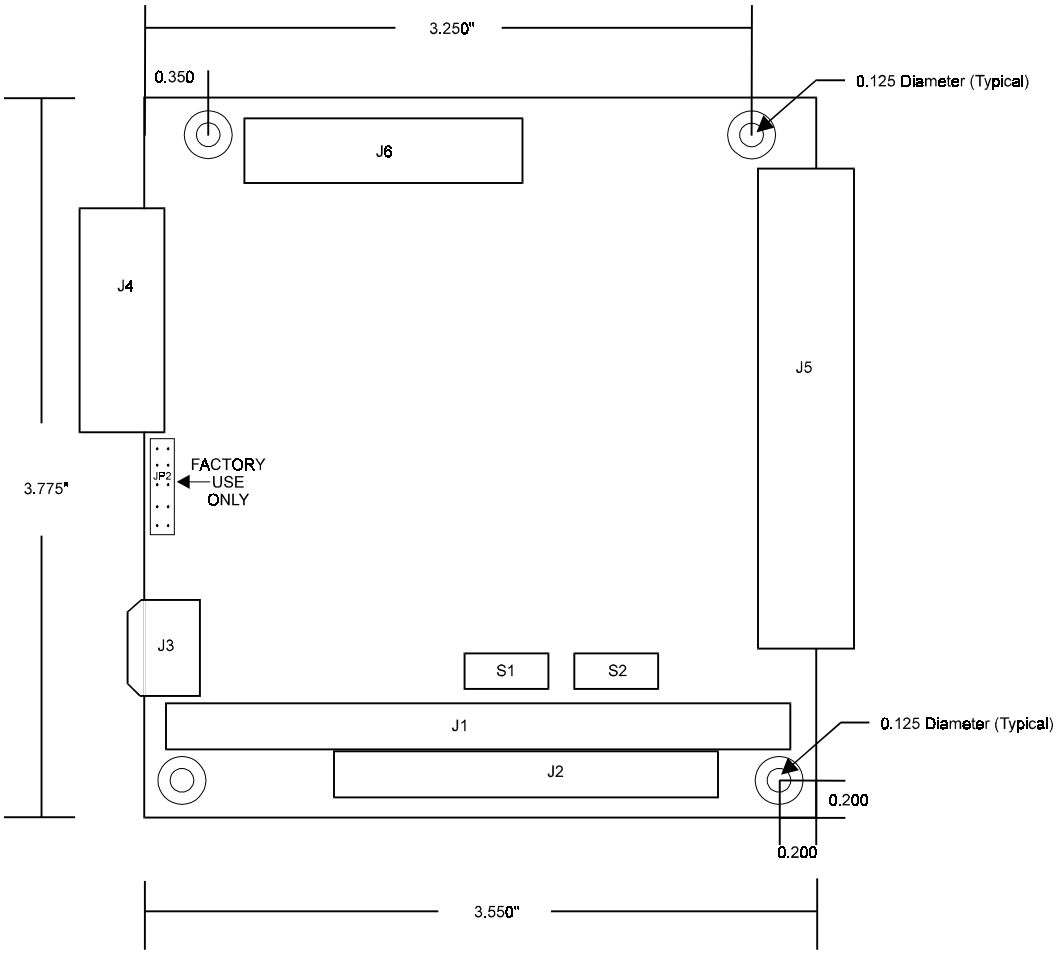

FIGURE 2-14 PC68 DIMENSIONAL LAYOUT

3. COMMUNICATION INTERFACE

3.1. INTRODUCTION

The PC68 is 100% compatible with PC/104 ISA (IEEE P996.1 specification) computers and can also communicate through an RS-232 port. The PC68 can be considered a motion coprocessor in the PC/104 computer where it can execute the motion process independent to the PC/104 CPU or continue to be interactive. The PC/104 interface of the PC68 has four consecutive 8 bit I/O ports available for control of interrupts, status of interrupt requests, data transfer as well as done flags. The PC68 can generate interrupts for several different conditions or can be polled.

The RS-232 port on the PC68 provides the ability for the PC68 to be a complete stand-alone controller without interaction from the host computer or it can be continually interactive through the RS-232 port. Baud rates of 300 to 38.4K are supported by the PC68.

3.2. PC/104 INTERFACE

The PC/104 interface to the controller consists of four consecutive I/O registers. Factory default address is 300 – 303H. The base address is selectable with switch S2 located on the top of the controller board.

> Note: Because the PC68 is a double-sided PCB, the top of the controller board is considered the side with the female half of the PC/104 connector and the I/O connector (J5). Where the bottom of the PCB is considered the side where the pins of the PC/104 connector are exposed.

The PC68 registers are all 8-bit I/O registers and are addressed using decode logic for addresses A0 – A9 of the PC/104 bus. (See Table 3-1 for I/O Register Description)The four registers, in order of the address sequence, are the Data register, Done Flag registers, Control register and the Status register. The PC68 supports interrupt levels 2 – 7 and can be set to any one of these levels with switch S1 located on the top of the controller board. If interrupts are not desired the PC68 can operate in a polled mode without generating any interrupts; however, it is preferred to use interrupt mode when communicating with the PC68.

3.3. PC/104 COMMUNICATION THEORY

The process for communicating to the PC68 through the PC/104 bus at its simplest form consists of the data register and the status register of the PC68. Reference Figure 3-1 for a flow chart of the communication sequence.

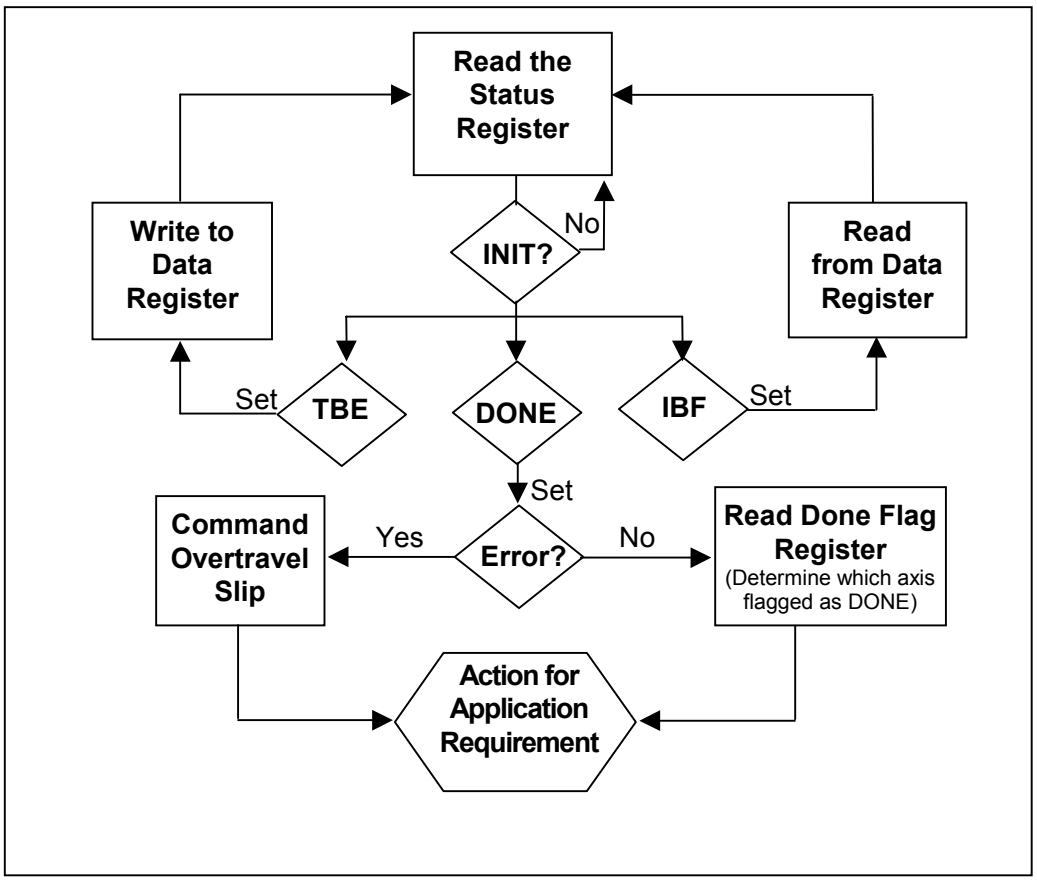

FIGURE 3-1 DATA COMMUNICATION FLOW CHART

The host CPU first reads the status register. The information provided in those 8 bits informs the host whether the PC68 is initialized and ready to receive a character and if there are any characters in the data register to be read by the host. The other information provided in the status registers is not a function of the communication at this point. It is required that the INIT bit of the status register is set low and that the TBE_S bit of the status register is set high before the host can write a character to the data register. If those two bit of the status register are acceptable then the host CPU writes a character to the data register. The TBE_S bit will go low until the PC68 reads the character from the data register. The host is either waiting for an interrupt generated by a high TBE_S or polling the status register so that the host may send another character. When the host detects an active TBE_S then it writes another character to the data register. The TBE_S bit goes low until the PC68 reads the character from the data register.

If the two characters written to the PC68 have requested a response from the PC68 then the IBF_S bit will go high. This can generate an interrupt to the host or the host can poll the status register to detect this condition. At this point the host would read the data register. This will cause the IBF_S bit to go low and then return high very fast, as long as there was more than one character to be read from the data register. The host would continue checking the status register between each read of the data register until all characters that the PC68 has sent to the data register are read.

Some rules of thumb:

1. Every time the host reads the status register it must check the INIT bit for a low before continuing to read or write anything to the controller.
2. Every time the host reads the status register and the IBF_S bit is high the host should read the data register until the IBF_S bit goes low.
3. The host should never write to the data register when the TBE_S bit is low.
4. While developing your system, it is a good idea to monitor all bits of the Status Register so you have a clear picture of what is happening on the controller.
5. If the controller has been reset (i.e. via the RS command), you should allow 3 seconds before reading the INIT bit for valid data.

3.4. PC/104 ADDRESS SELECTION

The PC68 is operated as an I/O mapped device and occupies a block of 4 contiguous I/O addresses. The factory default addresses are from base address 300 through 303 hex. Reference for the configuration of switch S2 for the address selection. A closed switch sets the associated address bit to low where an open switch sets the associated address bit high. See section 2.3 Set the I/O Address Range.

TABLE 3-1 I/O REGISTER DESCRIPTION

ADDRESS OFFSET	FACTORY DEFAULT	DESCRIPTION
0	300 hex	Data Register
1	301 hex	Done Flag Register
2	302 hex	Control Register
3	303 hex	Status Register

3.5. USING INTERRUPTS

Full interrupt capability is provided by the PC68 in accordance with the PC/104 specification, but not required. Interrupts provided are: input buffer full (IBF), transmit buffer empty (TBE), overtravel fault (limit) and operation complete (done). Interrupt levels 2 through 7 are selectable with switch S1. Polled operation is also supported with separate status bits for each of the above interrupt sources. See section 2.4 Set the Interrupt Request (IRQ).

3.6. DATA REGISTER

The data register is the data communication port between the PC68 and the PC/104 host CPU. All data is passed between the host processor and the PC68 through this port. The port is full duplex in both directions. This allows for faster processing of the data between the host and the PC68.

3.7. DONE FLAG REGISTER

The done flag register is a Host CPU read only register of the PC68. Each bit of this register identifies the done status of each of the axes of the PC68. It is an 8-bit register where the first 4-bits are used for the four axes of the PC68 and the other 4-bits are supported when an expansion board is used to expand the total number of axes to 8. When requested by command the PC68 will set the appropriate bits of the done flag register to indicate when a process is complete on the associated axes. The host can then read the register at any time to determine the done status of any axes. The register is cleared at the completion of the host read or by the execution of the RA or RI commands. The detail definition of the register is shown in Table 3-2.

TABLE 3-2 DONE FLAG REGISTER DESCRIPTION

DONE STATUS REGISTER DESCRIPTION	
BIT	**DESCRIPTION**
0	Done Status of X Axis
1	Done Status of Y Axis
2	Done Status of Z Axis
3	Done Status of T Axis
4*	Done Status of U Axis
5*	Done Status of V Axis
6*	Done Status of R Axis
7*	Done Status of S Axis

* Valid when an expansion board is used to make a controller with more than four axes.

3.8. INTERRUPT CONTROL REGISTER

This is a host read/write register that provides different interrupt sources from the PC68 to be individually enabled or disabled. This may be performed at any time by a write to the associated bit of the register. The register may be read back at anytime to verify or determine the state of the interrupts. Reference Table 3-3 for a detailed definition of the control register.

TABLE 3-3 CONTROL REGISTER DESCRIPTION

BIT	NAME	CONTROL DESCRIPTION
7	IRQ_E	Interrupt enable bit. This bit must be on to enable any interrupts.
6	TBE_E	Transmit buffer empty interrupt enable bit. This bit should be checked before writing to the data register to avoid sending a character when the interrupt has been disabled.
5	IBF_E	Input buffer full interrupt enable bit.
4	DON_E	Done or error status interrupt enable bit.
3	Unused	
2	Unused	
1	Unused	
0	Unused	

3.9. STATUS REGISTER

This register is a host read only register that provides status information to the host CPU. This status is independent of the enable status of the interrupt, allowing the board to operate in a polled mode if desired. In an interrupt mode the host would read this register upon receiving an interrupt. The information provided in this register can define the source of the interrupt. The functionality of this and the other registers is consistent with other OMS products. Future expansions to these registers may be performed to enhance the controllers capabilities. See Table 3-4 for details on the status register.

TABLE 3-4 STATUS REGISTER DESCRIPTION

BIT	NAME	STATUS DESCRIPTION
7	IRQ_S	Interrupt request status.
6	TBE_S	Transmit buffer empty status. This high true bit indicates a character may be written to the transmit buffer.
5	IBF_S	Input buffer full status. This high true bit indicates a character is available in the input buffer to be read by the host.
4	DON_S	Done or error status. This high true bit indicates the command is complete; i.e. an ID command has been detected. If bits 0 through 3 are all false it indicates a command completion; i.e. an ID command has been executed. The error bits indicate one or more errors have been detected.
3	OVRT	Overtravel. An overtravel switch was true, indicating attempted travel out of bounds.
2	ENC_S	Encoder error. This bit flags a slip error on models with the encoder option if the interrupt on slip (IS) command has been issued.
1	INIT	Init flag. This bit indicates the PC68 is being reset or the 68332 microprocessor has not completed initialization. Host initialization routines should check this bit for a zero before proceeding.
0	CMD_S	Command error. An unrecognizable command has been detected or LS and LE commands are not in matched pairs.

In order to resolve the source of a done or error interrupt, the DON_S bit (bit 4) of the status register should be read first. This bit in the status register is automatically reset upon the termination of the read cycle. If the DON_S flag is true the error bits should be checked to determine if the interrupt was caused by an error condition. If no error condition is present, the done flag register should be read to determine which axis or axes are done. The error bits of the status register and the done flag register are automatically cleared when these registers are read.

The transmit buffer empty (TBE_S) bit of the status register is reset by a host write to the data register and the input buffer full (IBF_S) bit of the status register is reset by a host read of the status register. If the PC68 is responding to, for instance, an axis query and there are several characters to be read from the data register by the host the IBF_S bit of the status register may never appear to reset, though it does very fast. Also, the PC68

can read characters from the data register very fast which may make the TBE_S bit of the status register appear to never reset, though it does very fast.

The error bits, command error (CMD_S), encoder slip (ENC_S) and overtravel (OVRT) of the status register are automatically cleared when the Status Register is read. This will reset these bits in the status register without altering the state of the done flag register.

The INIT bit only goes high when the board is initializing and cannot communicate. It will go low and remain low when initialization is complete.

3.10. RS-232 INTERFACE

The PC68 is compatible with the EIA standard RS-232 specifications for serial communication at baud rates of 300, 600, 1200, 2400, 4800, 9600, 19200 and 38400. The factory default baud rate is 9600. The PC68 is configured as a data terminal device so that the signals of the host terminal can connect straight through, there is no need for a null-modem, and a standard RS-232 cable can be used. The RS-232 interface to the PC68 consists of transmit data (TxD), receive data (RxD) and handshake signals clear to send (CTS) and data terminal ready (DTR).

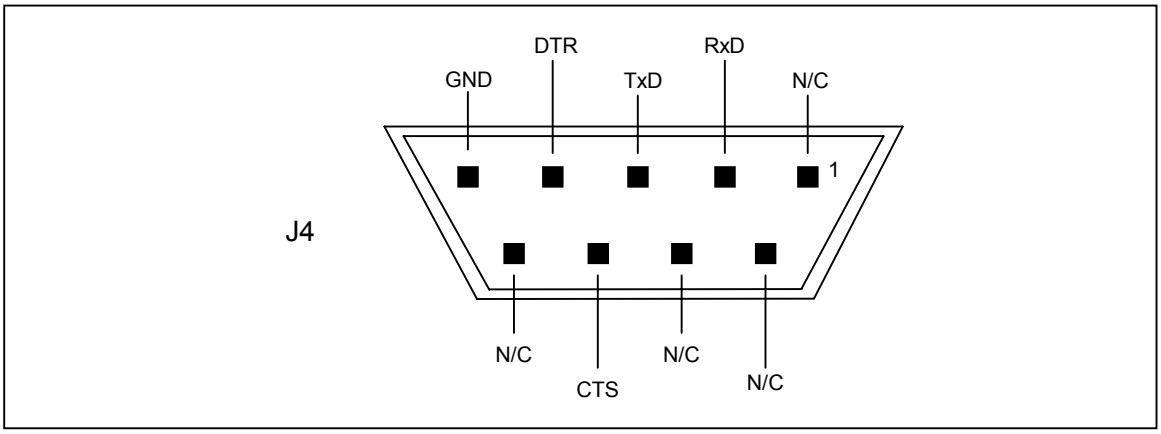

FIGURE 3-2 RS-232 CONNECTOR PINOUT

Due to the lack of registers when operated in RS-232 mode the PC68 returns status information to the host via special characters inserted into the response stream. If a “Done” or “Error” condition occurs, the PC68 will send one or more of these characters to the host. These characters are:

#	Command Error
$	Motor Slip (with encoders only)
@	Overtravel Limit
!	Done

These characters are not axis-specific. It is necessary for the host to query the PC68 for that information if it is required.

This page intentionally left blank

4. CONTROL SIGNAL INTERFACE

4.1. INTRODUCTION

The PC68 is available in several configurations to manage combinations of servo and step motor systems. A single 68 pin shielded SCSI-3 connector incorporates all the control signals of the PC68. The mating connector is an AMP, Inc. part number 749621-7 with a 749195-2 hood and strain relief. As a convenience in system integration, connection to the PC68 for +5VDC power, digital ground and analog ground is provided at the output connector (J5).

4.2. LIMIT AND HOME INPUTS

To facilitate system implementation, limit and home inputs are provided for each axis. Limits may be activated by mechanical switches using contact closures or other suitable active switches, such as a hall effect switch or opto-isolator that connects to ground.

If the motor travels beyond its allowable limits and trips the switch, the limit switch closure removes the excitation from the affected axis. You can select the limit switch active signal state with the LH and LL command on an axis by axis basis. The behavior of the limit functionality can be controlled with the System Control Commands (see page 6-14).

The home switch provides a means to synchronize the motor controller with the load at some home, or reference, position. The home switch, when used with the software commands HM or HR, causes the motor to decelerate to a stop when the switch closes. On finding the home position, the position counters are initialized to the parameter supplied with the command. You can change the sense of the home switches to TRUE when open by use of the HH command (described on page 6-15).

4.3. CONTROL OUTPUT

The PC68 is configured at the factory for control of servo motors, stepper motors or a combination of both. The servo output may be either unipolar analog (0/+10 volt) or bipolar analog (-10/+10 volt). (see the UN and BI commands in Section 6 Command Structure (on page 6-121 and on page 6-120).

Step pulse output is TTL open-collector which will wire directly into most driver inputs but may require a pull-up resistor to +5 VDC to operate some other drives. See wiring diagrams that follow.

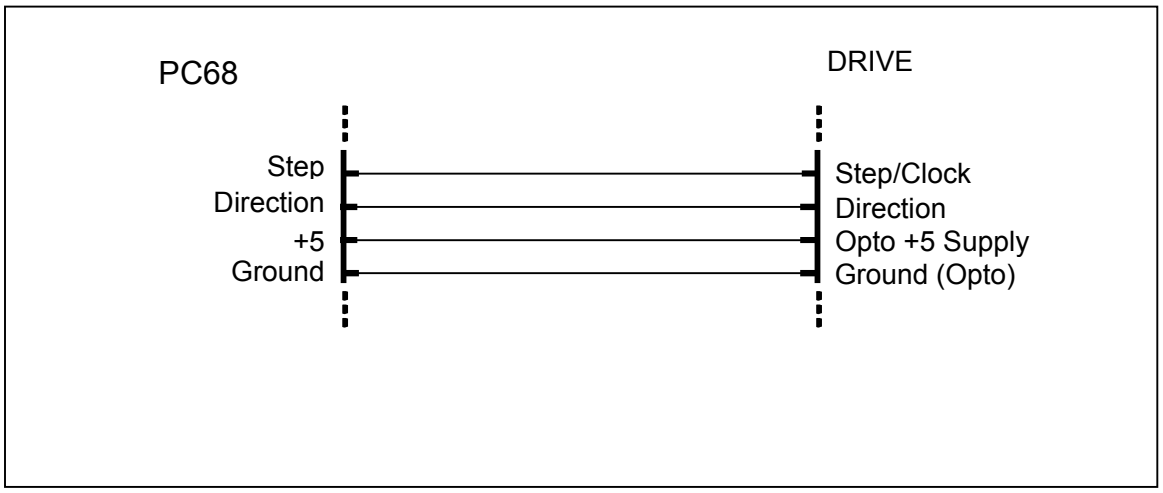

FIGURE 4-1 CONNECTION TO STEP DRIVES WITH INTERNAL PULL-UP RESISTORS AND OPTO-ISOLATION

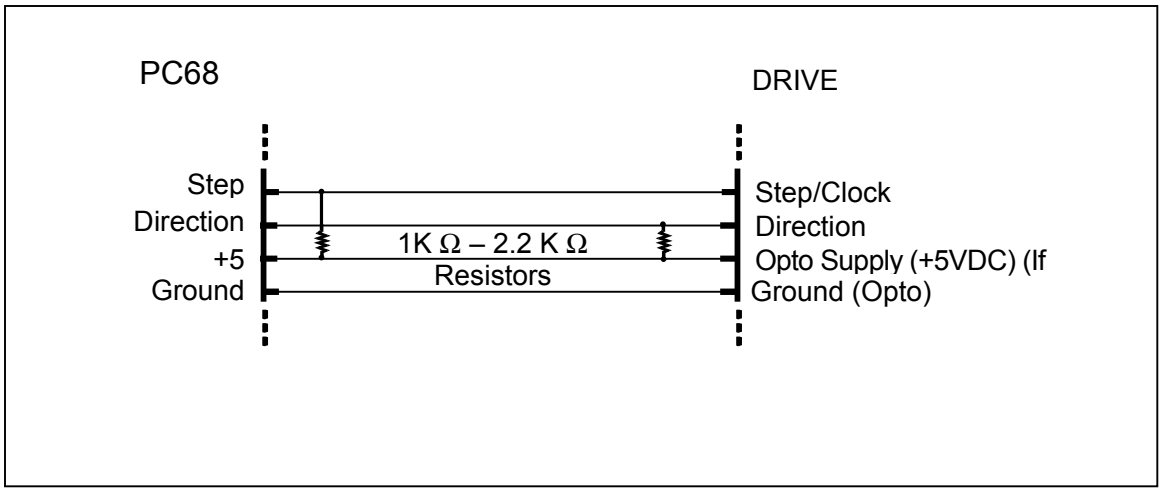

FIGURE 4-2 CONNECTION TO STEP DRIVES WITHOUT PULL-UP RESISTORS

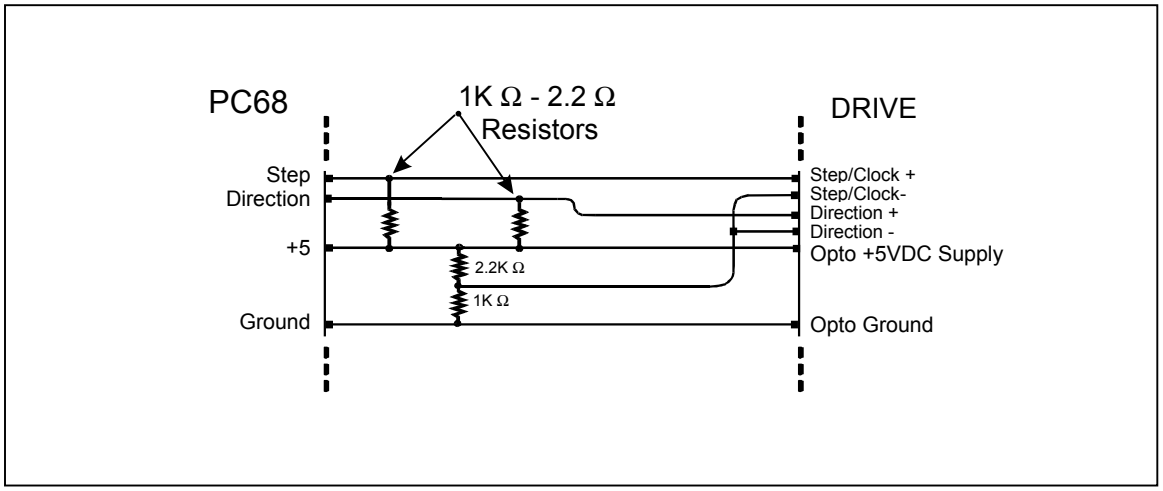

FIGURE 4-3 CONNECTION TO STEP DRIVES WITH DIFFERENTIAL INPUTS

TABLE 4-1 OUTPUT CONNECTOR PIN LIST (J5)

SIGNAL CONNECTOR (J5)			
Pin#	**Description**	**Pin#**	**Description**
1	Digital Ground	35	+5VDC
2	I/O-1	36	I/O-0
3	I/O-3	37	I/O-2
4	I/O-5	38	I/O-4
5	I/O-7	39	I/O-6
6	Digital Ground	40	+5VDC
7	X Index +	41	X Servo
8	X Index -	42	X Step
9	X Phase A +	43	X Auxiliary
10	X Phase A -	44	X Direction
11	X Phase B +	45	X Limit +
12	X Phase B -	46	X Limit -
13	Y Servo	47	X Home
14	Y Index +	48	Y Step
15	Y Index -	49	Y Auxiliary
16	Y Phase A +	50	Y Direction
17	Y Phase A -	51	Y Limit +
18	Y Phase B +	52	Y Limit -
19	Y Phase B -	53	Y Home
20	Analog Ground	54	+5VDC
21	Z Index +	55	Z Servo
22	Z Index -	56	Z Step
23	Z Phase A +	57	Z Auxiliary
24	Z Phase A -	58	Z Direction
25	Z Phase B +	59	Z Limit +
26	Z Phase B -	60	Z Limit -
27	T Servo	61	Z Home
28	T Index +	62	T Step
29	T Index -	63	T Auxiliary
30	T Phase A +	64	T Direction
31	T Phase A -	65	T Limit +
32	T Phase B +	66	T Limit -
33	T Phase B -	67	T Home
34	Digital Ground	68	+5VDC

4.4. IO68 ADAPTER MODULE

The optional IO68 is an adapter module designed to provide easy connection for each signal of the PC68. It incorporates a three row terminal block and some on board filters for the limit inputs. A cable is available with the mating connector to fit the PC68 connector (J5). The +5VDC on the IO68 is supplied by the PC68 and is protected by a resetable fuse on the PC68.

This supply voltage is intended to be utilized with accessories used in conjunction with the PC68 such as sensors, motor driver modules, etc., and is specified to supply a maximum current of 0.5 amps for these purposes.

If the fuse detects an over current situation (such as an external short circuit), the supply will shut down. It can be re-activated by powering down the PC68, ensuring the over current situation has been removed, and by powering the PC68 up again after 3 seconds.

As the fuse is a semiconductor device, it never has to be replaced and requires no maintenance.

4.4.1. EXPLANATION OF ADDITIONAL CIRCUITRY ON THE IO68-M

The IO68-M board is to be used in conjunction with a PC68 or an expansion board that provides more than four axes of control. The IO68-M board contains low-pass filtering circuitry for the positive and negative limit signals. The default values for the RC constants are 100Ω and 1.0μF. With these RC constants used the average cutoff frequency is 21.5kHz in a worse case scenario where all other signals are left unterminated.

In addition to the above mentioned circuitry, there are also 2.2k pull-up resistors that have been added to the step output signals for each axis. These resistors have been added to the IO68-M to help reduce noise that may occur on the step output signals.

These resistors and capacitors are packaged as through hole devices. Should your particular application require different RC constants, these devices can be removed from the board and replaced with the appropriate components. When changing these components, use appropriate methods to desolder and solder the components to avoid causing damage to the board (i.e. lifting pads from the board).

There are two switches on the IO68-M (S43 and S45) that are used in regards to encoder signals. If your system uses differential encoders, then these switches should be in the OFF position (except where indicated in Table 4-2.) The switches are set to the OFF position as default from factory. If, however, your system employs single-ended encoders, then the negative encoder signals need to be fed a biasing voltage. Since a biasing voltage has been made available on the IO68-M, simply change the position of the switches to ON to tap into this voltage. Be sure to change the position of these switches when power to the board is OFF (the green LED should be off).

TABLE 4-2

IO68-M ENCODER BIAS SWITCH (S43)

Switch Number	Signal Description
1	Y Phase B-
2	Y Phase A-
3	Y Index-
4	X Phase B-
5	X Phase A-
6	X Index-
7	(Leave ON)
8	(Leave ON)

IO68-M ENCODER BIAS SWITCH (S45)

Switch Number	Signal Description
1	T Phase B-
2	T Phase A-
3	T Index-
4	Z Phase B-
5	Z Phase A-
6	Z Index-
7	(Leave OFF)
8	(Leave OFF)

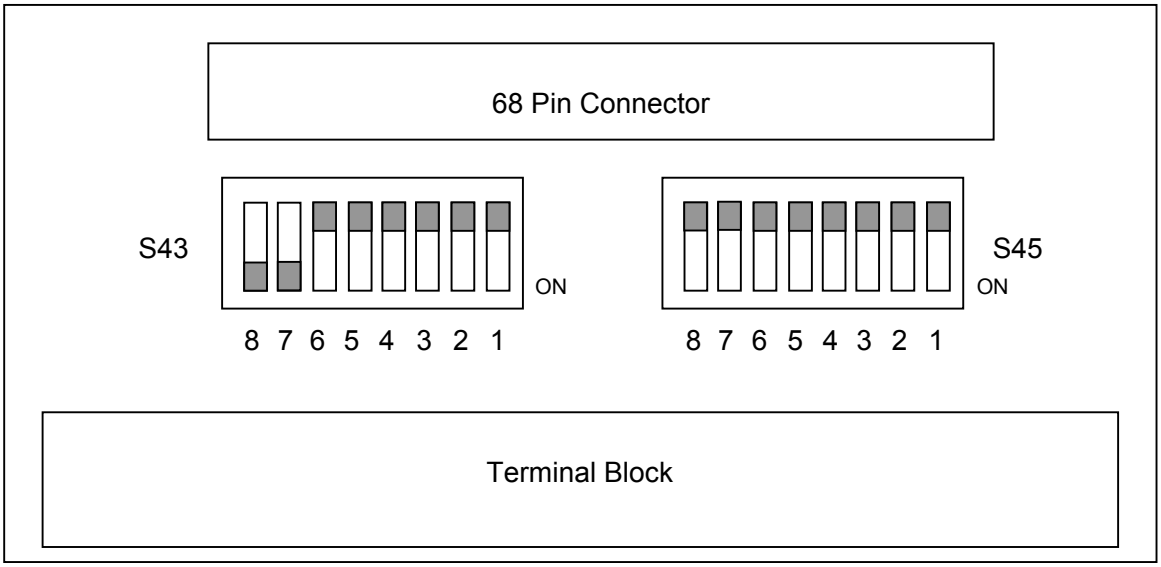

FIGURE 4-4 IO68-M DEFAULT SWITCH SETTING

If only one of the axes is using a single-ended encoder, while the others are using a differential encoder, you need only turn on the switch positions that relate to the axis with the single-ended encoder.

Besides biasing the negative encoder signals, these switches also configure the IO68-M. Placed in the position shown in Figure 4-4, the IO68-M is configured to work with the PC68 or the expansion board that adds more axes. For the IO68-M, S43, pins 7 and 8 are to be ON, while S45, pins 7 and 8 are to be OFF.

All other signals on the IO68-M are straight through connections, with no additional circuitry added. Should you need filtering circuitry on any of these other signals, it would have to be added external to the IO68-M. Contact Oregon Micro Systems technical support for further instruction.

Table 4-3 shows the pin definitions on the IO68-M terminal block connector.

TABLE 4-3 IO68-M TERMINAL BLOCK PIN-OUT

Row 1	Description	Row 2	Description	Row 3	Description
1	X Step	24	X Direction	47	X Auxiliary
2	X Phase A+	25	X Phase B+	48	X Index +
3	X Phase A-	26	X Phase B-	49	X Index -
4	X Limit +	27	X Limit-	50	X Home
5	+5VDC	28	X Servo	51	Digital Ground
6	Y Step	29	Y Direction	52	Y Auxiliary
7	Y Phase A+	30	Y Phase B+	53	Y Index +
8	Y Phase A-	31	Y Phase B-	54	Y Index -
9	Y Limit +	32	Y Limit-	55	Y Home
10	+5VDC	33	Y Servo	56	Digital Ground
11	I/O-0	34	I/O-3	57	I/O-5
12	I/O-1	35	No Connect	58	I/O-6
13	I/O-2	36	I/O-4	59	I/O-7
14	+5VDC	37	Z Servo	60	Analog Ground
15	Z Step	38	Z Direction	61	Z Auxiliary
16	Z Phase A+	39	Z Phase B+	62	Z Index +
17	Z Phase A-	40	Z Phase B-	63	Z Index -
18	Z Limit +	41	Z Limit-	64	Z Home
19	+5VDC	42	T Servo	65	T Auxiliary
20	T Step	43	T Direction	66	T Index +
21	T Phase A+	44	T Phase B+	67	T Index -
22	T Phase A-	45	T Phase B-	68	T Home
23	T Limit +	46	T Limit-	69	Digital Ground

NOTE: The pin out is the same for a IO68-M when used with an expansion board to add more axes, just exchange X, Y, Z and T with U, V, R and S respectively. Refer to Table 5-3.

4.4.2. EXPLANATION OF THE IO68-I

When the IO68 is configured as an IO68-I, the additional low-pass filter circuitry and added pull-up resistors are removed from the board. All signals are routed as straight through connections.

Also, S43 and S45 are configured so the IO68-I will work with the Px6IO board. For the IO68-I, S43, pins 7 and 8 are to be OFF, while S45, pins 7 and 8 are to be ON. Table 4-4 shows the pin definitions for these switches.

TABLE 4-4 SWITCH DEFINITION FOR IO68-I

S43

Switch Number	Signal Description
1	IO20
2	Field VCC
3	IO35
4	IO18
5	IO1
6	Field GND
7	(Leave OFF)
8	(Leave OFF)

S45

Switch Number	Signal Description
1	IO29
2	IO15
3	IO45
4	Field VCC
5	IO12
6	IO43
7	(Leave ON)
8	(Leave ON)

The factory default switch setting for the IO68-I is shown below:

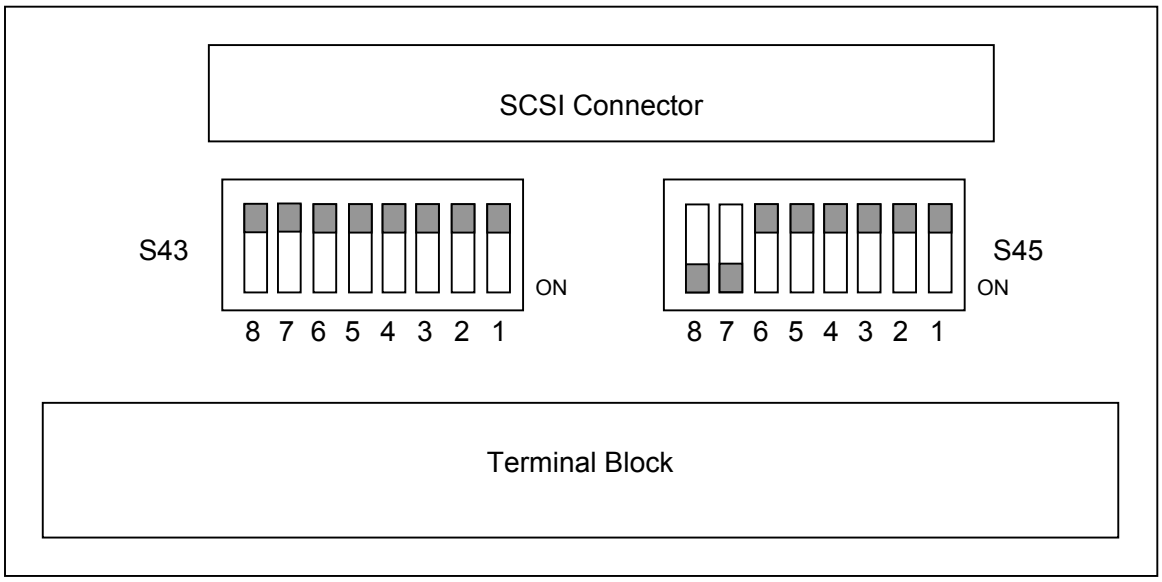

FIGURE 4-5 IO68-I DEFAULT SWITCH SETTING

Table 4-5 IO68-I Terminal Block pin-out shows the pin definitions for the IO68-I terminal block connector.

TABLE 4-5 IO68-I TERMINAL BLOCK PIN-OUT

Row 1	Description	Row 2	Description	Row 3	Description
1	GND	24	GND	47	+5VDC
2	IO0	25	IO17	48	+5VDC
3	IO1	26	IO18	49	FieldGND
4	+5VDC	27	IO19	50	IO31
5	IO2	28	+5VDC	51	IO32
6	IO3	29	GND	52	IO33
7	IO4	30	FieldGND	53	IO34
8	+5VDC	31	IO20	54	IO35
9	+5VDC	32	GND	55	+5VDC
10	IO5	33	IO21	56	IO36
11	IO6	34	IO22	57	IO37
12	IO7	35	No Connect	58	IO38
13	IO8	36	IO23	59	IO39
14	IO9	37	IO24	60	IO40
15	IO10	38	IO25	61	IO41
16	IO11	39	IO26	62	IO42
17	IO12	40	+5VDC	63	IO43
18	GND	41	+5VDC	64	GND
19	IO13	42	Field GND	65	GND
20	+5VDC	43	IO27	66	IO44
21	IO14	44	IO28	67	IO45
22	IO15	45	IO29	68	IO46
23	IO16	46	IO30	69	IO47

4.5. ENCODER FEEDBACK

Incremental encoder feedback is provided for all servo axes and is optional for the stepper axes. The encoder option accepts quadrature pulse inputs from high resolution encoders at rates up to 4 MHz (after quadrature detection). The encoder monitors actual position through the encoder pulse train. It then continuously calculates the position error on the servo axes and through the PID filter adjusts the output based on that error. The stepper axes can monitor the error and correct the position after the move is finished. The encoder input can also be used as an independent feedback source. All modes are capable of slip or stall detection and encoder tracking with electronic gearing. These options are selectable by the user through software commands.

4.6. ENCODER SELECTION AND COMPATIBILITY

The PC68 is compatible with virtually any incremental encoder which provides quadrature outputs. Times four quadrature detection is used to increase resolution. This means that an encoder rated for 1024 counts (or lines) per revolution will result in 4096 counts. The inputs are compatible with encoders, which have single ended or differential TTL outputs. The PC68 inputs have built in hysteresis to minimize effects of noise pickup. The PC68

has differential line receivers to accommodate encoders with differential line driver outputs.

4.7. HOME PROCEDURES

When single ended encoders are used the unused negative inputs, i.e. Phase A-, Phase B-, etc. must be biased at or near +1.5V. The IO68 provides convenient switches for this. (See Table 4-2 on 4-5)

Two modes are provided to synchronize the physical hardware with the PC68 controller, i.e. put the controlled motor in the home position.

HS mode (factory default):

The home switch input is a TTL level input signal that can be used to physically home a mechanical stage. This signal can be either a logic HIGH or logic LOW true by using the HH and HL commands. The HM or HR commands are used after reducing the velocity to no more than 2048 (1024 for more than 4 servo axes controllers) pulses per second. This limit on velocity is necessary to avoid ambiguity of the home position if more than one pulse occurs per sample interval. When this functionality is used the axis position counter will be reset to a select value when the switch is activated. At this point the PC68 can either ramp the axis to a stop or stop the axis immediately. The control of the direction of travel, the logic active state and the response to the active switch are controlled through commands.

HE mode:

PC68 home inputs can be used with encoders which provide one home pulse for the complete travel of the stage. The index input uses internal logic to establish the home position when used with the HE command mode. This position consists of the logical AND of the encoder index pulse, the home enable external input (LOW true only) and a single quadrant from the encoder logic. The home enable pulse must be true for less than one revolution of the encoder thus allowing only one home for the complete travel of the stage. This input is not inverted by the HH and HL commands. The home logic expressed in Boolean terms is:

$$\text{home} = \text{phaseA} * \overline{\text{phaseB}} * \text{index} * \overline{\text{home switch}}$$

It is necessary that the above quadrant occur within the index pulse as provided by the encoder for this logic to function properly. It may be necessary with some encoders to shift the phase of this quadrant by inverting one or both of the phases. Inverting one phase or swapping phase A for phase B will also reverse the direction. The encoder counter (read by an RE command) must increase for positive moves or the system will oscillate due to positive feedback.

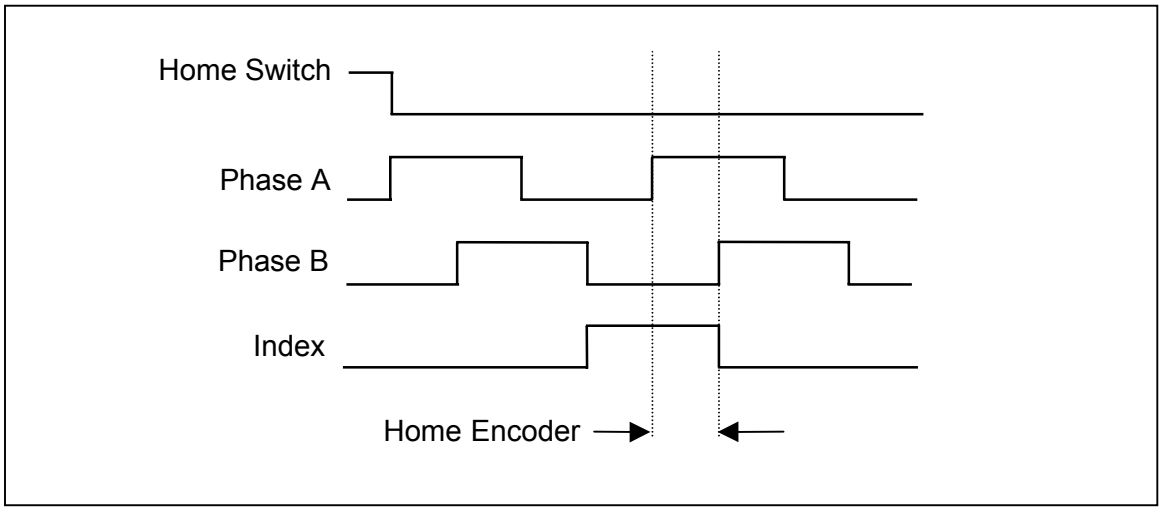

FIGURE 4-6 ENCODER HOMING STATE DETECTION

5. EXPANSION BOARDS

5.1. Px6IO I/O EXPANSION BOARDS

The Px6IO daughter board is meant to stack onto the PC68 board. The PC/104 stack through connectors on the Px6IO will mate to the PC68. Also, the 60-pin surface mount, board-to-board connector on the Px6IO board will mate to the 60-pin connector on the PC68. The physical size of the Px6IO I/O board meet the PC104 IEEE standard.

This board adds an additional 48 optically isolated, general purpose I/O bits. From the factory, the board is configured for 24 outputs and 24 inputs. Configurations other than this may also be available; contact Oregon Micro Systems for additional information.

The Px6IO board will draw +5V and digital Ground from the PC68 board via the PC/104 stack through connectors. A jumper at JP1 connects, Field Ground and Digital Ground together. Field VCC and +5V are connected together by a jumper at location JP2.

NOTE: 1. Should the user require isolated power and ground signals to go to the opto couplers contact OMS for instructions.
2. The DIP switch at S4 is set at the factory and is not user selectable.

5.1.1. HARDWARE

The I/O bit connections at the 68-Pin connector, J4, are expecting TTL signals (+5V High and 0V Low). Small current limiting resistors have been included on the Px6IO for the input bits to the Px6IO going to the optocouplers. Additional resistance can be added externally at the wiring connections if required. The current going to the input of the optocouplers should be 20mA, maximum.

Output bits are routed through a 220Ω resistor from the optocouplers to the 68-Pin connector. Output current from the optocouplers is rated at 60mA, maximum. Should you need to limit the current coming from the Px6IO, add additional resistance externally to the wiring connections of the board.

The pin definitions for the 68-Pin connector at J4 are described below.

TABLE 5-1

Px6IO Connector Pin-Out			
Pin Number	**Description**	**Pin Number**	**Description**
1	IO32	35	IO2
2	IO7	36	IO6
3	IO22	37	IO8
4	IO37	38	IO23
5	IO39	39	IO38
6	IO36	40	IO5
7	Field VCC	41	+5V
8	Field Ground	42	Digital Ground
9	IO0	43	+5V
10	IO1	44	Digital Ground
11	IO17	45	+5V
12	IO18	46	IO19
13	IO21	47	IO31
14	IO34	48	IO3
15	IO35	49	IO33
16	IO4	50	Digital Ground
17	Field VCC	51	+5V
18	Field Ground	52	Digital Ground
19	IO20	53	+5V
20	IO40	54	IO9
21	IO42	55	IO24
22	IO43	56	IO10
23	IO11	57	IO41
24	IO12	58	IO25
25	IO26	59	Digital Ground
26	Field VCC	60	+5V
27	Field Ground	61	Digital Ground
28	IO44	62	+5V
29	IO45	63	Digital Ground
30	IO14	64	IO27
31	IO15	65	IO16
32	IO28	66	IO30
33	IO29	67	IO46
34	IO47	68	IO13

5.1.2. CONNECTING THE Px6IO BOARD TO THE PC68 STACK

Should you need to attach the Px6IO board to the PC68 stack, be sure you begin by shutting OFF power to the PC68 controller and peripherals. Then, align the pins of the PC/104 connectors to the receptacle of the mating PC/104 connectors. While doing this, align the pins of the 60-pin connector on the Px6IO board with the 60-pin connector of the board directly beneath it. With these connectors aligned, carefully press them together until all the connectors are fully seated.

Now add the necessary hardware for the four standoffs in each corner of the Px6IO. Once the Px6IO has been successfully added to the stack, power can now be applied to the boards.

5.1.3. SOFTWARE

See the command section of the manual for further information regarding the I/O bit commands on the Px6IO board.

5.1.4. CONNECTING THE Px6IO TO AN IO68-I

Refer to Table 4-5 IO68-I Terminal Block pin-out on page 4-8 for pin out.

5.2. AXIS EXPANSION BOARD

The axis expansion board has been designed to stack onto a PC68 or any of the PC68's other daughter boards. It is equipped with PC/104 stackable connectors that will mate to the PC68 or its daughter boards. In addition to the PC/104 connectors is a 60-pin board-to-board type connector that is used to mate to the other boards in the stack.

The axis expansion board was designed to add four additional axes of control to the PC68 board. These four axes can be used as stepper, stepper with encoder feedback, or servo, in any of a number of configurations. Contact OMS for more information regarding the various configurations of this board.

In addition to the added four axes of control, there are an additional eight bits of general purpose I/O on the axis board. These are fixed at four inputs and four outputs. Along with the general purpose I/O, there is an Auxiliary bit that is available for each axis and can be treated as an I/O bit. So, the axis expansion board is capable of offering up to 12 additional bits of I/O.

The board receives the +5V, +12V, -12V, and Digital Ground that it uses from the PC/104 connectors. Two jumpers on the board are used to connect Digital and Analog Ground together. These two signals are tied together on the board from factory. If it is desired to use separate Digital and Analog Ground signals, these jumpers would have to be removed, and the separate Analog Ground signal could be fed to the board via the 68-pin connector or the 6-pin connector at J3. Contact OMS for instruction on how to implement these modifications.

5.2.1. HARDWARE

The step output of the axis expansion board produces a 50% duty cycle square wave step pulse. The velocity ranges from 0 to 1,044,000 steps/second. The acceleration ranges from 0 to 8,000,000 steps/second/second. The servo control uses a 16-bit DAC and outputs either a +/-10V or 0 to +10V range. The encoder feedback control can be used as feedback for the servo PID, position maintenance for the stepper axes or as just a position feedback for any axes. The encoder

input supports either differential or single ended quadrature TTL signals at a rate of up to 4MHz and counts at a 4 times resolution. This means a 1000 line encoder will produce 4000 counts per revolution in the axis expansion board.

Step output, auxiliary, and direction signals for the axes are open-collector TTL level signals. All of the general purpose I/O bits have a 2.2k pull up resistor on the axis expansion board. They output TTL totem pole output signals. As inputs, they expect TTL level signals. The home and limit switches are TTL level inputs with 2.2k pull up resistors on the board.

The pin out for the 68-pin connector on the axis expansion board is shown below:

TABLE 5-2

Expansion Board Connector Pin-Out			
Pin Number	**Description**	**Pin Number**	**Description**
1	Digital Ground	35	+5VDC
2	I/O Bit 9	36	I/O Bit 8
3	I/O Bit 11	37	I/O Bit 10
4	I/O Bit 13	38	I/O Bit 12
5	I/O Bit 15	39	I/O Bit 14
6	Digital Ground	40	+5VDC
7	U Index +	41	U Servo
8	U Index -	42	U Step
9	U Phase A +	43	U Auxiliary
10	U Phase A -	44	U Direction
11	U Phase B +	45	U Limit +
12	U Phase B -	46	U Limit -
13	VANG	47	U Home
14	V Index +	48	V Step
15	V Index -	49	V Auxiliary
16	V Phase A +	50	V Direction
17	V Phase A -	51	V Limit +
18	V Phase B +	52	V Limit -
19	V Phase B -	53	V Home
20	Analog Ground	54	+5VDC
21	R Index +	55	R Servo
22	R Index -	56	R Step
23	R Phase A +	57	R Auxiliary
24	R Phase A -	58	R Direction
25	R Phase B +	59	R Limit +
26	R Phase B -	60	R Limit In
27	SANG	61	R Home
28	S Index +	62	S Step
29	S Index -	63	S Auxiliary
30	S Phase A +	64	S Direction
31	S Phase A -	65	S Limit +
32	S Phase B +	66	S Limit -
33	S Phase B -	67	S Home
34	Digital Ground	68	+5VDC

5.2.2. CONNECTING THE AXIS EXPANSION BOARD TO AN IO68-M

An IO68-M and a SCSI-3 68-pin cable should be used to route the signals from the 68-pin connector on the axis expansion board to the screw terminal block connector on the IO68-M. When using the IO68-M with the board, the pin descriptions for axis U, V, R and S of the terminal block connector on the IO68-M are as shown below:

TABLE 5-3

IO68-M TERMINAL BLOCK PIN-OUT WHEN CONNECTED TO AXIS EXPANSION BOARD					
Row 1	**Description**	**Row 2**	**Description**	**Row 3**	**Description**
1	U Step	24	U Direction	47	U Auxiliary
2	U Phase A+	25	U Phase B+	48	U Index +
3	U Phase A-	26	U Phase B-	49	U Index -
4	U Limit +	27	U Limit -	50	U Home
5	+5V	28	UANG	51	Digital Ground
6	V Step	29	V Direction	52	V Auxiliary
7	V Phase A+	30	V Phase B+	53	V Index +
8	V Phase A-	31	V Phase B-	54	V Index -
9	V Limit +	32	V Limit -	55	V Home
10	+5V	33	VANG	56	Digital Ground
11	I/O-8	34	I/O-11	57	I/O-13
12	I/O-9	35	No Connect	58	I/O-14
13	I/O-10	36	I/O-12	59	I/O-15
14	+5V	37	RANG	60	Analog Ground
15	R Step	38	R Direction	61	R Auxiliary
16	R Phase A+	39	R Phase B+	62	R Index +
17	R Phase A-	40	R Phase B-	63	R Index -
18	R Limit +	41	R Limit -	64	R Home
19	+5V	42	SANG	65	S Auxiliary
20	S Step	43	S Direction	66	S Index +
21	S Phase A+	44	S Phase B+	67	S Index -
22	S Phase A-	45	S Phase B-	68	S Home
23	S Limit +	46	S Limit -	69	Digital Ground

5.2.3. SOFTWARE

See the command section of the manual for information regarding the commands for the axis expansion board.

This page intentionally left blank

6. COMMAND STRUCTURE

6.1. INTRODUCTION

An extensive command structure is built into the PC68 family of intelligent motor controls. It includes a 200 command and parameter buffer for each axis and a command loop counter which allows multiple executions of almost any command string.

Commands that are defined for use on the X, Y, Z, and T axes, are used with the PC68 controller. Commands referencing the U, V, R, and S axes apply to the axis expansion board that connects to the PC68. The commands that are used on I/O bits other than 0 to 15, require the Px6IO I/O board to be connected to the PC68 stack of boards.

The following commands in this section are included in the PC68 family of controllers. Most all of the commands are two ASCII characters and may be in upper or lower case. Some of the commands expect a numerical operand to follow. These commands are identified with a "#" after the command. The operand must be terminated by a space, carriage return or semi-colon to indicate the end of the number. No terminator is required on the other commands, but may be included to improve readability. Semi-colons are the preferred termination character because they are visible in your code. The operand must immediately follow the command with no space or separation character. The "#" indicates a signed integer input parameter or a signed fixed point number of the format ##.# when user units are enabled. With user units enabled distances, velocity and acceleration parameters may be input in inches, revolutions, etc.

Synchronized moves may be made by entering the AA command. This command performs a context switch which allows entering the commands in the format MRx#,y#,z#,t#,u#,v#,r#,s#. Numbers are entered for each axis which is to be commanded to move. An axis may be skipped by entering a comma with no parameter. The command may be prematurely terminated with a ";", i.e. a move requiring only the X and Y axes would use the command MRx#,y#; followed by the GO command. Each axis programmed to move will start together upon executing the GO command. The PC68 can be switched back to the unsynchronized mode by entering the desired single axis command such as AX.

The AM command is provided for complex applications where the host manages multiple motion processes by a multitasking operating system. This mode shares the same instructions as the AA mode, but allows starting a task while some other task involving one or many axes is active. For example, the X and Y axes could be doing linear interpolation while the Z axis is making an unrelated move simultaneously.

Constant velocity contouring provides another mode wherein the move parameters are predefined by entering AA then CD#,#;. The PC68 will then calculate the move profile in advance and move at constant velocity in the prescribed pattern. It can do linear interpolation on as many as 8 axes between the predefined points or it can do circular interpolation mixed with linear on two axes.

6.2. COMMAND QUEUES

The input characters are placed in a character buffer on input then removed and interpreted. The commands are then placed in separate command queues for each axis. As they are executed the space is reclaimed allowing the host to pass commands ahead of the moves actually being processed. Most of the commands are placed in the appropriate command queue for execution, while others are executed immediately allowing return of status information in a timely way rather than when encountered in the command stream. This information is provided in a table for each command which shows the queue requirements, if any, and indicates immediate in those cases where the command is not queued. The queue requirements shown in the tables are typical. Depending on the circumstances in which the command is issued, the actual queue requirement may vary slightly. The single axis cases are indicated by the mode reference indicating the appropriate axis. The synchronized mode is indicated by the mode identifier AA or AM. The contouring case is indicated by AA/CD for multiple axes in contour definition mode. The RQ command may be used to determine the actual space available at any time. The queues operate independently allowing each axis to perform separate processes concurrently. The synchronized modes (AA) insert special wait opcodes which allow the axes to be synchronized in this mode. When the commands are nested within loops, the queue space is not reclaimed until after the loop has been executed the programmed number of times. For loops larger than the queue space, the loop may never be completed since it cannot reclaim the queue space and cannot accept the loop terminator. The RQ command may be used to examine the remaining queue space. A Control-D may clear this condition if the input character queue is not also filled since it bypasses the command interrupter.

Some commands are valid only for stepper axes, others for stepper axes with encoder feedback, and still others for servo axes. Most are valid for all three types or some combination of types. A set of symbols to the right of each command identifies which motor types with which each command may be used. The symbols' meanings are as follows:

 Stepper motor with or without an encoder

 Stepper motor with an encoder

 Servo motor

If a command is usable with one of these motor types, the symbol will appear in black. If the command is not usable with a motor type, that motor symbol will be displayed in gray:

 This command is not usable with servo motors

 Indicates an example.

The following commands are available in firmware revision 6.00 and above.

6.3. COMMAND SUMMARY

The following commands are included in the PC68 family of motor controllers. The '#' indicates a signed integer input parameter or a signed fixed point number of the format ##.# when user units are enabled. With User Units enabled, distances, velocity and acceleration parameters may be input in inches, revolutions, etc.

COMMANDS IN CHAPTER 6		
COMMAND	**SECTION PAGE NUMBER**	**COMMAND DESCRIPTION**
AA	6-9	Any following commands are for the AA (All Axes) mode
?AC	6-58	Report AC command
AC#	6-57	Acceleration, set acceleration/deceleration register
?AD	6-54	Report default auxiliary bit state
ADH	6-51	Set auxiliary default to high
ADL	6-52	Set auxiliary default to low
AF	6-43, 6-142	Auxiliary off
AM	6-9	Axes multitasking mode
AN	6-42, 6-142	Auxiliary on
AP	6-24	Make the current parameter set the power up default values
AR *	6-13	Any following commands are for the R axis.
AS *	6-13	Any following commands are for the S axis.
AT	6-11	Any following commands are for the T axis
AU *	6-12	Any following commands are for the U axis.
AV *	6-12	Any following commands are for the V axis.
AX	6-10	Any following commands are for the X axis (default on reset)
AY	6-10	Any following commands are for the Y axis
AZ	6-11	Any following commands are for the Z axis
?BD#	6-55	Report the direction of a general purpose I/O
BH#	6-47, 6-144	Set selected I/O bit high (off)
BI	6-120	Bipolar, set the analog torque outputs to bipolar
BL#	6-47, 6-144	Set selected I/O bit low (on)
?BS#	6-56	Report the state of the specified I/O bit
BS	6-53	Set all bits of the general purpose output port to the state specified by the hex argument
BW	6-97	Wait for input to go low
BX	6-49, 6-102	Return bit status in hex format
CA	6-94	Clear done flag of currently addressed axis
CB	6-160	Clear all macro links to input bits
CD#,#;	6-145	Define a contour
CE	6-146	End contour definition, ramp to a stop
CK	6-146	End contour definition, immediately stop step pulses

COMMANDS IN CHAPTER 6		
COMMAND	**SECTION PAGE NUMBER**	**COMMAND DESCRIPTION**
CN	6-21	Cosine on, enable cosine velocity profiles
CR#,#,#	6-147	Circular interpolation, move in a circle
CV#	6-147	Contouring velocity, definition
CW	6-86	Clear while flag, i.e. terminate WH/WG loop
CX	6-148	Contour execute
?DA	6-32	Print a custom ramp
DAB	6-27	Define custom ramp breakpoint
DAE	6-28	End custom ramp definition
DAR	6-29	Begin custom ramp definition
?DB	6-33	Report direction bit logic
DBI	6-30	Invert direction bit
DBN	6-31	Normalize direction bit
DC	6-68	Set the deceleration rate that will be used by the GU command
?DE	6-34	Report an acceleration ramp definition table entry
?DS	6-35	Report the size of a custom acceleration ramp table
?DZ	6-110	Report DAC open-loop offset
DZ#	6-110	Offset coefficient, used in open-loop mode
EA	6-137	Encoder status, return encoder status of currently addressed axis
EF	6-14	Echo off, turn off echo to host (default at power up)
EN	6-14	Echo on, turn on echo to host
?ER	6-123	Report motor:encoder ratio
ER#,#	6-122	Encoder ratio, set encoder count to motor count ratio
?ES	6-128	Report encoder slip tolerance
ES#	6-132	Encoder slip tolerance, set tolerance before slip or stall is flagged
ET	6-135	Encoder tracking, set encoder tracking mode
FL	6-78	Flush an axis command queue
FP#	6-141	Force position, flush queue and attempt to stop at specified position
GD	6-70	Go and reset done flags
GO	6-69	Go command, start execution of motion
GS	6-75	Go and use the home switch to monitor for motor slip
GU	6-74	Go and use the AC values to accelerate and the DC values to decelerate
?HD	6-129	Report position maintenance deadband
HD#	6-125	Hold deadband, specify deadband tolerance for position hold
HE	6-136	Encoder home mode, set home on encoder logic
HF	6-118, 6-125, 6-133, 6-135	Hold off, disable position hold, slip detection and tracking modes
?HG	6-130	Report position maintenance gain

COMMANDS IN CHAPTER 6		
COMMAND	**SECTION PAGE NUMBER**	**COMMAND DESCRIPTION**
HG#	6-124	Hold gain, specify position hold gain parameter
HH	6-15	Home high, home switches are active high
HL	6-15	Home low, home switches are active low
HM#	6-87	Home, find home and initialize the position counter
HN	6-118, 6-127	Hold on, enable position correction after move
HR#	6-88	Home reverse, find home in reverse direction and initialize position counter
HS	6-136	Home switch, enable home switch mode
?HV	6-131	Report position maintenance velocity
HV#	6-124	Hold velocity, specify maximum position hold correction velocity
IC	6-93	Interrupt clear, clear done interrupt status and error flags
ID	6-90	Interrupt host when done and set done flag
II	6-91	Interrupt independent
IN#	6-91	Interrupt when nearly done
IO#	6-48	Designates I/O bits as inputs or outputs
IP	6-92, 6-128	Interrupt when in position
IS	6-132	Interrupt slip, interrupts host on slip or stall detection
IX	6-98	Interrupt when done. Sends Hex character via RS-232
JF#	6-72	Jog at fractional rates
JG#	6-71	Jog command, run motor at specified velocity until a new velocity command is sent or it is stopped by a stop or kill command
?KA	6-111	Report acceleration feedforward
KA#	6-111	Acceleration Feedforward coefficient, used in tuning servo systems
?KD	6-112	Report PID derivative gain
KD#	6-112	Derivative Gain coefficient, used for PID filter
?KI	6-113	Report PID integral gain
KI#	6-113	Integral Gain coefficient, used for PID filter
KL	6-77	Kill, flush queue and terminate pulse generation immediately on all axes without decelerating
KM	6-88	Home and kill pulse generation
?KO	6-115	Report PID closed-loop offset
KO#	6-114	Offset coefficient, used in closed-loop mode
?KP	6-116	Report PID proportional gain
KP#	6-116	Proportional Gain coefficient, used for PID filter
KR	6-89	Home in reverse and kill pulse generation
KS	6-79	Kill selected axes
?KV	6-117	Report velocity feedforward
KV#	6-117	Velocity Feedforward coefficient, used in tuning servo systems

COMMANDS IN CHAPTER 6		
COMMAND	**SECTION PAGE NUMBER**	**COMMAND DESCRIPTION**
LA	6-22	Linear ramp selection per axis
LE	6-83	Loop end, terminate most recent LS command
LF	6-16	Disable limit switches for selected axis
LH	6-17	Limit high, limit switch is active high
LL	6-17	Limit low, limit switch is active low
LN	6-16	Enable limit switches for selected axis
LP#	6-63	Load position, load position counter with parameter
?LS	6-18	Report limit active state
LS#	6-82	Loop start, set loop counter, from 1 to 32000 loops; (may be nested to 4 levels)
MA#	6-64	Move absolute, move to absolute position
MD#	6-150	Define a temporary macro
ML#,#;	6-66	Move linear, move specified distance relative from current position
MM	6-139	Move minus, set minus direction for MV type move
MO	6-67	Move one pulse in current direction
MP	6-139	Move plus, set positive direction for MV type move
MR#	6-65	Move relative, move specified distance from current position
MT#,#;	6-66, 6-148	Move to, move to specified position
MV#,#	6-140	Move velocity, move to first parameter (absolute position) at second parameter velocity without stopping at end of move
MX#	6-151	Execute a macro command string
?PA	6-44	Report power automatic state
PA#	6-44, 6-143	Power automatic, turn power on before each move and off after the move
PC#	6-161	Reports the configuration of a Px6IO input/output port
PE	6-100, 6-138	Report encoder positions of all encoder and servo axes
PF	6-22	Parabolic off, disable parabolic ramps, i.e. linear ramps will be generated
PI#	6-162	Reports the bit states of a Px6IO port as a hex number
?PM	6-126	Report PID state
PM#	6-151	Print a macro command string
PN#	6-21	Parabolic on, enable parabolic ramps
PO#,#	6-162	Sets the bits of a Px6IO output port to the states specified by a hex number
PP	6-100	Report motor positions of all axes
PR#	6-23	Parabolic ramp selection per axis
PS	6-159	Reports macro link to specified PC68 input bit
PT#,#	6-152	Preserve a temporary macro by copying it to non-volatile memory
PX	6-160	Reports macro link to specified Px6IO

COMMANDS IN CHAPTER 6		
COMMAND	**SECTION PAGE NUMBER**	**COMMAND DESCRIPTION**
QA	6-104	Query status of switches and flags for addressed axis without affecting flags
QI	6-105	Query status of switches and flags on all axes without affecting flags
RA	6-103	Return status of switches and flags and reset flags
RB	6-50	Return programmed direction of I/O bits in hex format
RC	6-105	Return current acceleration or deceleration of the current axis
RD	6-25	Restore the current parameter set to the power up default values
RE	6-138	Request encoder position, return current encoder position
RF	6-25	Restore the current parameter set to the factory default values
RI	6-104	Return status of switches and flags for all axes and reset flags
RL	6-133	Return slip status of each axis
RM#	6-67	Return remainder of position divided by parameter in position counter
RP	6-99	Request position, returns current position
RQ	6-101, 6-149	Request queue status, return number of queue entries available
RS	6-24	Software reset of PC68
?RT	6-26	Report ramp type
RU	6-106	Return current position in user units
RV	6-106	Return current velocity at which the axis is moving
SA	6-76	Stop all, flush queue and stops all axes with deceleration
?SB	6-166	Report current baud rate setting
SB#	6-166	Sets the controller's serial communications baud rate
SC	6-23	Cosine ramp selection per axis
SD	6-77	Stop all axes and clear any done flags
?SE	6-46	Report settling time
SE#	6-45	Set settling time before power is reduced in PA mode
SF	6-19	Soft limit off, restore normal overtravel operation
SI	6-81	Stop selected motors
SK	6-156	Links KILL function to specified PC68 input bit
?SL	6-20	Report soft limit status
SL	6-19	Soft limit mode, allow pulse train to ramp down on overtravel
SM	6-153	Enables/Disables stand-alone mode
?SO	6-121	Report analog output mode
SO	6-80	Stop at a designated position using a specified ramp down distance

COMMANDS IN CHAPTER 6		
COMMAND	**SECTION PAGE NUMBER**	**COMMAND DESCRIPTION**
SP#	6-141	Stop at position, stop at specified position if possible after all commands have been executed
SR	6-36	Selects custom ramp
ST	6-76	Stop, flush queue and decelerate to stop
?SV	6-39	Report servo voltage invert selection
SVI	6-37	Invert servo voltage
SVN	6-38	Normalize servo voltage
SW#	6-96	Sync wait, wait for the input bit to be released by other controllers
SX	6-154	Links macro to specified PC68 input bit
TF	6-134	Turn encoder slip kill off
?TL	6-41	Report software overtravel ranges
TL	6-40	Set software travel limits
TM#	6-72	Jog at the current velocity for the specified number of milliseconds
TN	6-134	Turn encoder slip kill on
UF	6-109	User units off, turn off user unit translation
UN	6-121	Unipolar, set the analog torque outputs to unipolar
?UU	6-108	Report axis user units' axis assignment
UU#	6-107	User units, multiply acceleration, velocity and distance parameters by specified parameter
?VB	6-62	Report axis base velocity
VB	6-61	Base velocity, set base velocity
?VL	6-60	Report axis velocity limit
VL#	6-59	Set maximum velocity to be used in profile
VS#,#,#	6-73	Velocity stream, slave velocity mode for profiling
WA	6-95	Wait until all moves on all axes are finished
WD	6-84	While end, WS loop terminator
WG	6-86	Terminate WH loop
WH	6-85	While, execute all commands until WG loop terminator, until flag cleared by CW command
WQ	6-95	Wait until current axis queue is empty
WS#	6-84	While sync, execute while sync is true
WT#	6-97	Wait, wait for specified number of milliseconds
WY	6-99	Who are you , returns model and software revision
XC#	6-163	Reports the configuration of a Px6IO input/output bit
XH#	6-164	Sets a Px6IO output bit to 1
XI#	6-164	Reports the bit state of a Px6IO bit
XK	6-158	Links KILL function to specified Px6IO
XL#	6-165	Sets a Px6IO output bit to a zero
XX	6-157	Links macro to specified Px6IO input bit

* These commands are valid when an expansion board is used with the PC68 to get more than four axes of control.

6.4. AXIS SPECIFICATION COMMANDS

The following commands set the context to direct the commands which follow to the appropriate axis. They remain in effect until superseded by another command of the same type, specifying a different axis.

AA AXES ALL

The AA command will perform a context switch to coordinated moves.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	2	2
AM	2	2	2
AA/CD	Not Valid		

Example: Perform an absolute move using the X and Y axes.

Enter:
```
AA MA12000,14000; GO
```

AM AXES MULTITASKING

The AM mode allows several tasks to be managed simultaneously. For instance, a task may be performing coordination motion on 2 axes, while a second task is performing unrelated but simultaneous motion on another axis.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Perform a coordinated move on the X and Y axes, while moving the T axis as a separate move.

Enter:
```
AM MR2000,3000; GO MA,,,10000; GO
```

AX AXIS X

The AX command sets the context to direct all the following commands to the X axis. This is the default mode at power up or reset.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Make the X axis step at a rate of 5,000 steps/second.

Enter: `AX JG5000;`

AY AXIS Y

The AY command sets the context to direct all the following commands to the Y axis.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Examine the status of the Y axis.

Enter: `AY RA`

AZ AXIS Z

The AZ command sets the context to direct all the following commands to the Z axis.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Move the Z axis 2,000 steps at a rate of 500 steps/second.

Enter:
```
AZ VL500 MR2000 GO
```

AT AXIS T

The AT command sets the context to direct all the following commands to the T axis.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Move the T axis to absolute position -2468.

Enter:
```
AT MA-2468; GO
```

AU AXIS U

The AU command sets the context to direct all the following commands to the U axis.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Set the U axis position register to -56789.

Enter: `AU LP-56789`

AV AXIS V

The AV command sets the context to direct all the following commands to the V axis.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Set the auxiliary line low on the V axis.

Enter: `AV AF`

AR AXIS R

The AR command sets the context to direct all the following commands to the R axis.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Set the auxiliary line low on the R axis.

Enter: `AR AF`

AS AXIS S

The AS command sets the context to direct all the following commands to the S axis.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Set the auxiliary line low on the R axis.

Enter: `AS AF`

6.5. SYSTEM CONTROL COMMANDS

These commands allow control of various system parameters and operating modes to allow the user to optimize the response of the system for his/her application needs.

EN ECHO ON

The EN command enables echoing. All commands and parameters will be echoed to the host. This mode is useful for debugging command strings from a terminal. This mode also outputs an English readable error message to the host which may be echoed to the terminal or computer to aid in debugging.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Enable echoing by the PC68 so that commands are echoed and the error message is returned to the host as a readable ASCII string. This command would probably be the first command executed after turning on the system when this mode is desired.

Enter: `EN`

EF ECHO OFF

The EF command disables echoing from the PC68 motion system. This is the default mode at power up or reset.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Stop echoing to the host.

Enter: `EF`

HH HOME HIGH

The HH command sets the sense of the home switch on the current axis to active high. This allows the use of a normally closed switch.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	1	1	1
AM	1	1	1
AA/CD	Not Valid		

Example: (see HL command below)

HL HOME LOW

The HL command sets the sense of the home switch on the current axis to active low. This is the default mode at power up or reset.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	1	1	1
AM	1	1	1
AA/CD	Not Valid		

Example: A faster home sequence may be used in applications which have a long distance to travel to reach home. The stage is moved through home at high speed with the home switch set for active high then reversed at low speed to meet the 2048 steps per second requirement of the home command.

Enter:

```
AX VL20000 HH HM0
VL1000 HL HR0
```

or

```
AA HH ,,1,,1;
```

LF LIMITS OFF

The LF command turns off the limit switches for the addressed axis. This allows the stage to move beyond the limit switch and should be used with caution.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	1	1	1
AM	1	1	1
AA/CD	Not Valid		

Example: Set up a board to ignore the Y axis limit switches.

Enter:
```
AY LF
```
or
```
AA LF,1,,1;
```

LN LIMITS ON

The LN command restores the operation of the limit switches for the addressed axis. This is the default mode at power up or reset.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	1	1	1
AM	1	1	1
AA/CD	Not valid	Not Valid	Not Valid

Example: Set up the Y and T axes to stop immediately when a limit switch is encountered.

Enter:
```
AA LN, 1,,1;
```
or
```
AX LN;
```

LH LIMITS HIGH

This limits high command sets the overtravel limit switches so they are active high. For instance, the overtravel occurs when the input level is high. See the AP Command on page 6-24 to preserve the LH settings as the Power up/Reset modes.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	1	1	1
AM	1	1	1
AA/CD	Not valid	Not Valid	Not Valid

Example: Select the limit switch high true condition for the X axis.

Enter: `AX LH`

Example: Select a high true limit condition for the Z and U axes.

Enter: `AA LH,,1,,1;`

LL LIMITS LOW

The LL command specifies that overtravel occurs when the limit input signal is low (active low). This is the factory default mode. See the AP Command on page 6-24 to preserve the LL settings as the Power up/Reset mode.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	1	1	1
AM	1	1	1
AA/CD	Not valid	Not Valid	Not Valid

Example: Select the limit switch low true condition for the X axis.

Enter: `AX LL`

Example: Select a low true limit condition for the Z and U axes.

Enter: `AA LL,,1,,1;`

?LS REPORT LIMIT ACTIVE STATE

The ?LS command reports the active state of the limits for the current axis. The LL and LH commands are used to set this value and are the possible responses of this command.

Example: Find out whether the Y axis limits are active high or active low.

Enter: `AY ?LS`

Response: `<LF><CR>LL<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?LS	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: LH, LL

SL SOFT LIMIT

The SL command changes the operation of the limit inputs causing the output pulse train to ramp down instead of terminating immediately. The output queue is not flushed except for the current move. This mode is effective for point to point and JG moves only. See the AP Command on page 6-24 to preserve the SL settings as the Power up/Reset value.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	1	1	1
AM	1	1	1
AA/CD	Not Valid		

Example: Set up a board to allow the X axis to ramp to a stop when a limit is encountered.

Enter: `AX SL`

Example: Set up a board I/O to allow the Z and U axes to ramp to a stop when a limit is encountered.

Enter: `AA SL,,1,,1;`

SF SOFT LIMIT OFF

The SF command restores the normal operation of the limit switches. This is the factory default mode. See the AP Command on page 6-24 to preserve the restore the SF selection as the Power up/Reset value.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	1	1	1
AM	1	1	1
AA/CD	Not Valid		

Example: Set up a board to make the X axis stop immediately when a limit is encountered.

Enter: `AX SF`

Example: Set up a board to make the Z and U axes to stop immediately when a limit is encountered.

Enter: `AA SF,,1,,1`

?SL REPORT SOFT LIMIT STATUS

The ?SL command reports whether soft limits are currently enabled for the active axis. Possible responses are SL and SF, the same commands used to enable and disable soft limits.

Example: Find out whether soft limits are enabled for axis Z

Enter: `AZ ?SL`

Response: `<LF><CR>SL<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?SL	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: SL, SF

CN COSINE ON

The CN command enables cosine velocity ramps, i.e. half sinusoid acceleration profiles for all axes. The cosine is not truncated in moves that do not reach full speed. This command should not be given while an axis is in motion or the results may not be predictable. **This command affects all axes, even if issued in the single axis mode.** The PF command is used to return to linear motion profiles. See the AP Command on page 6-24 to preserve the CN setting as the Power up/Reset ramp.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	29	29	29
AA	29	29	29
AM	29	29	29
AA/CD	Not Valid		

Example: Set the board to be in cosine mode.

Enter: `CN`

PN# PARABOLIC ON

The PN command sets all axes to truncated parabolic ramps. This acceleration profile starts at 100% of the programmed acceleration and decreases in steps of 10% of the initial acceleration down to as low as 10%. The parameter supplied selects the number of steps. It must be in the range of 3 to 10 corresponding to 70% and 10% acceleration at the peak respectively. A parameter out of this range or no parameter supplied defaults to 70% or 3 steps. Note that the parameter is the number of steps, not the acceleration values. The larger number is a lower acceleration at the peak. This command should not be given while an axis is in motion or the results may not be predictable. This command affects all axes, even if issued in the single axis mode. PF is the factory default setting. The PF command is used to return to linear motion profiles. See the AP Command on page 6-24 to preserve the PN setting as the Power up/Reset ramp.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	15	29	Not valid
AA	15	29	Not Valid
AM	15	29	Not Valid
AA/CD	Not Valid		

Example: Set the board to be in the smoothest parabolic acceleration ramp.

Enter: `PN10;`

PF PARABOLIC OFF

The PF command restores all axes to linear acceleration and deceleration ramps. This command should not be given while an axis is in motion or the results may not be predictable. This command turns off the PN and CN modes. This command affects all axes, even if issued in the single axis mode. PF is the factory default setting. See the AP Command on page 6-24 to restore the PF setting as the Power up/Reset mode.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	11	11	11
AA	11	11	11
AM	11	11	11
AA/CD	Not valid	Not Valid	Not Valid

Example: Turn off cosine or parabolic ramps, returning to linear.

Enter: `PF`

LA LINEAR RAMP PER AXIS

The LA command specifies that the linear acceleration ramp is to be used by the selected axes. This is the factory default for all axes. See the AP Command on page 6-24 to preserve the LA settings as the Power up/Reset values.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	11	11	Not valid
AA	11	11	Not Valid
AM	11	11	Not Valid
AA/CD	Not Valid		

Example: Select a linear ramp for the X axis.

Enter: `AX LA`

Example: Select the linear ramp for the Y and T axes.

Enter: `AA LA,1,,1;`

SC COSINE RAMP PER AXIS

The SC command specifies that the standard cosine acceleration ramp is to be used by the selected axis/axes.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	29	29	Not valid
AA	29	29	Not Valid
AM	29	29	Not Valid
AA/CD	Not Valid		

Example: Select the cosine ramp for the X axis.

Enter: `AX SC`

Example: Select the cosine ramp for the Y and T axes.

Enter: `AA SC,1,,1;`

PR# PARABOLIC RAMP PER AXIS

PR selects that a parabolic acceleration ramp is to be used by the specified axes.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	15	29	Not valid
AA	15	29	Not Valid
AM	15	29	Not Valid
AA/CD	Not Valid		

Example: Select a 10 step parabolic ramp for the T axis.

Enter: `AT PR10`

Example: Select a 10 step parabolic ramp for the Y axis and a 8 step parabolic ramp for the T axis.

Enter: `AA PR,10,,8;`

RS RESET

The RS command is a software reset which causes the local PC68 microprocessor to reset. All previously entered data and commands are lost. All internal parameters are initialized to defaults. All interrupts are disabled. This command is intended for catastrophic failure recovery only. The KL command should be used to reset queues or return the system to a known state. Monitor the INIT flag in the status register for completion of the initialization process; see Table 3-4. The "Initializing" in process bit goes high during the initialization process.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Clear everything in the board and stop all movement. Reset all hardware registers.

Enter: `RS`

AP ASSIGN CURRENT PARAMETERS AS POWER UP DEFAULT VALUES

The AP command sets the current parameter set to be the power up default set of values. This is done by writing the current parameter set to flash memory. The following list of parameters will have their values saved to flash memory when the AP command is used: SB, IO, SF/SL, LN/LF, LH/LL, AC, LA/SC/PR/SC, VL, VB, UU, SE, PA0/PA1, ER, ES, HD, HV, HG, BI/UN, KP, KD, KI, KO, KV, KA, and DZ.

NOTE: This command should not be issued when an axis is in motion. It also should be used sparingly because the flash memory has a limited number of write cycles. (i.e. Less than 10,000 times, typical)

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Save the current parameter set to be the power up default set of values.

Enter: `AP`

RD RESTORE DEFAULT VALUES

RD assigns the current parameter set to be the default values that are in flash memory.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Assign the current parameter set to be the default values.

Enter: `RD`

RF RESTORE FACTORY DEFAULT VALUES

RF assigns the current parameter set to be the factory default values.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Assign the current parameter set to be the factory default values.

Enter: `RF`

?RT REPORT RAMP TYPE

The ?RT command reports the current acceleration ramp assigned to the active axis. Possible responses are:

LA	Default linear ramp
PRn	Parabolic where n specifies number of segments
SC	Cosine ramp
SRn	Custom ramp where n specifies the table number

Example: Make sure custom ramp #3 was assigned to the Y axis

Enter: `AY ?RT`

Response: `<LF><CR>SR3<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?RT	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: PF, LA, PN, PR, CN, SC, SR

DAB DEFINE CUSTOM RAMP BREAKPOINT

The DAB command sets a breakpoint in a custom ramp table. This is the only command that should be used after DAR and before DAE. Each custom ramp may contain up to 25 breakpoints, each defined by a DAB command.

The DAB command takes two parameters; the first specifies the acceleration level that should be used to achieve the second parameter, velocity level. Both levels are expressed in terms of percentage in decimal format; i.e. 1.00 is 100%. At no time should a DAB command be entered in which the velocity parameter is less than the velocity parameter of the prior DAB. The PC68 will not flag this as a command error but the results of such a ramp will be unpredictable. Each DAB command sent should be equal to or greater than the DAB command that preceded it. It is the user's responsibility to make sure this command is used properly.

RANGE:
0.00000 <= Parameter 1 <= 1.00000
0.00000 <= Parameter 2 <= 1.00000

Example: See the DAR command on page 6-29 for a complete example of a custom profile.

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
DAB#,#;	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: DAR, DAE, ?DA, ?DS, ?DE, SR

DAE END CUSTOM RAMP DEFINITION

The DAE command terminates a custom ramp table definition initiated by the DAR command.

Example: See the DAR command on page 6-29 for a complete custom ramp table definition.

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
DAE	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: DAR, DAB, ?DA, ?DS, ?DE, SR

DAR BEGIN CUSTOM RAMP DEFINITION

The DAR command starts the definition of a custom ramp table. A parameter supplied with this command, from 1 to 8, specifies which ramp table to create. If a ramp table by that number has already been defined, it will be overwritten.

Once the DAR command has been issued, only the DAB and DAE commands will be valid. A series of ramp table breakpoints may be entered using the DAB command which define the profile breakpoints for this ramp table. Up to 25 breakpoints may be defined but a smaller number may be used. A ramp table containing no breakpoints is invalid and will result unpredictably if used.

RANGE: 1 <= DAR <= 8

Example: Create a ramp table definition resembling a jerk-limited linear profile.

Enter:

```
DAR3;              * Store as table #3
DAB.1,.05;         * Ramp at 10% of AC until 5% of
VL
DAB.3,.1;          * Ramp at 30% of AC until 10% of
VL
DAB.9,.9;          * Ramp at 90% of AC until 90% of
VL
DAB.3,.95;         * Ramp at 30% of AC until 95% of
VL
DAB.1,1;           * Ramp at 10% of AC until 100% of
VL
DAE                * End table definition
```

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
DAR#;	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: DAB, DAE, ?DA, ?DS, ?DE, SR

DBI INVERT DIRECTION BIT

The DBI command inverts the logic of the direction control output of the addressed axis or axes. By default, the direction output of an axis is a TTL low when traveling in the positive direction and high when traveling negative. After using the DBI command, the direction bit will be high when traveling positive and low when traveling negative. This is useful for inverting the logical direction of a motor when the encoder counts opposite the motor direction. This command can be canceled using the DBN command. To make this the default at powerup or reset, use the AP command.

Example: Set the direction outputs for axes Z and T to output high when traveling positive and low when traveling negative. Leave X and Y as they are.

Enter:
```
AZ DBI AT DBI
      or
AA DBI,,1,1;
```

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
DBI	AX – AT	1	1
DBIb,b,b,b;	AA-AM	1	1
-	AA/CD	Not Valid	

Related commands: DBN, ?DB, UN, BI, SVI, SVN

DBN NORMALIZE DIRECTION BIT

The DBN command normalizes the logic of the direction control output of the addressed axis or axes, returning their output logic to default; i.e. TTL low when traveling in the positive direction and high when traveling negative. This command negates the effect of the DBI command. To make this the default at powerup or reset when DBI has already been made the default, use the AP command.

Example: Set the direction outputs for axes Z and T to default output logic; i.e. output low when traveling positive and high when traveling negative. Leave X and Y as they are.

Enter:
```
AZ DBN AT DBN
      or
AA DBN,,1,1;
```

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
DBN	AX – AT	1	1
DBNb,b,b,b;	AA-AM	1	1
-	AA/CD	Not Valid	

Related commands: DBI, ?DB, UN, BI, SVI, SVN

?DA PRINT A CUSTOM RAMP

This command will print out a previously defined custom ramp table. If an undefined table number is specified, only linefeed + carriage return pairs will be returned.

RANGE: 1 <= ?DA <= 8

Example: Print out custom ramp table #2

Enter: `?DA2;`

Response:
```
<LF><CR>DAR2<LF><CR>
DAB0.10000,0.20000<LF><CR>
DAB0.90000,0.80000<LF><CR>
DAB0.10000,1.00000<LF><CR>
DAE<LF><CR>
```

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?DA#;	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: DAR, DAB, DAE, ?DS, ?DE

?DB REPORT DIRECTION BIT LOGIC

The DB command returns the command, either DBI or DBN, that would set the direction bit to its current logical operation. If the direction bit is low when moving positive, this command will return the DBN command. If the direction bit has been inverted, this command will return DBI.

Example: Report whether the direction bit for the T axis is low or high when making positive moves

Enter: `AT ?DB`

Response: `<LF><CR>DBI<LF><CR>` (The DBI result indicates the T axis direction bit is high for positive moves)

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?DB	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: DBI, DBN, ?SV

?DE REPORT A CUSTOM RAMP TABLE ENTRY

The ?DE command will return a specific entry from a specific custom ramp table. The first parameter specifies the table to examine and the second parameter specifies the entry to return from the table.

RANGE:
1 <= Parameter1 <= 8
1 <= Parameter2 <= 25

Example: We can't remember what the 23rd breakpoint in table 4 was set to. Use the ?DE command to find out.

Enter: `?DE4,23;`

Response: `<LF><CR> <LF><CR>` (there is no 23rd entry in table 4)

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?DE#,#;	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: DAR, DAB, DAE, ?DA, ?DS

?DS REPORT THE SIZE OF A CUSTOM RAMP TABLE

The ?DS command returns the size of the specified custom ramp table as the number of breakpoint entries in the table.

RANGE: 1 <= ?DS <= 8

Example: The 3rd custom ramp should be 17 breakpoints long. Make sure this is true.

Enter: `?DS3;`

Response: `<LF><CR>17<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?DS#;	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: DAR, DAB, DAE, ?DA, ?DE

SR SELECT CUSTOM RAMP

The SR command selects a previously defined custom ramp profile for use with a specific axis. This command will override previous ramp type selection for the given axis such as PN and CN.

RANGE: 0 <= SR <= 9

Example: Select custom ramp number 4 for use with axis Y and ramp number 8 for axis T.

Enter: `SR,4,,8;`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
SR#;	AX – AT	12	60*
SR#,#,#,#;	AA-AM	12	60
-	AA/CD	Not Valid	

* (10+(2 for each segment in the ramp))

Related commands: ?RT, PF, LA, PN, PR, CN, SC, DAR, DAB, DAE

SVI INVERT SERVO VOLTAGE

The SVI command inverts the voltage output for the current axis. After receiving this command, the PC68 will produce a negative voltage for positive motion and a positive voltage for negative motion. To cancel this command, issue an SVN command. To make inverted servo outputs the default at powerup or reset, use the AP command.

Example: The Y axis encoder is counting opposite the expected direction. Setup the Y axis to produce a negative voltage when moving positive instead of a positive voltage to correct the problem.

Enter: `AY SVI`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
SVI	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: SVN, ?SV, UN, BI, DBI, DBN

SVN NORMALIZE SERVO VOLTAGE

The SVN command normalizes the voltage output for the current axis, negating the effects of the SVI command. After receiving this command, the PC68 will produce a positive voltage for positive motion and a negative voltage for negative motion, the default behavior. To make this the default behavior (if it has been changed via SVI/AP), use the AP command. (SVN is the factory default setting.)

Example: The Y axis encoder was rewired and now counts in the correct direction. Return the Y axis servo output to normal; i.e. output positive voltage for positive motion.

Enter: `AY SVN`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
SVN	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: SVI, ?SV, UN, BI, DBI, DBN

?SV REPORT SERVO VOLTAGE INVERSION STATE

The ?SV command reports the current logical direction for the current servo axis. The state is set with the SVI and SVN commands.

Example: Report whether servo voltage is positive for positive moves on axis X

Enter: `AX ?SV`

Response: `<LF><CR>svn<LF><CR>` (voltage is normal; i.e. positive for positive moves)

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?SV	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: SVI, SVN, ?DB

TL SET SOFTWARE TRAVEL LIMITS

The TL command sets logical limits on the range of travel for an axis. Two parameters must be supplied; one for the upper travel limit and the other for the lower travel limit, both as absolute positions. If the axis reaches either of these logical limits, the PC68 will flag a limit condition just as it would using the physical limit switch inputs. Move Relative (MR) type motion is subject to software travel limits because the PC68 checks an internal absolute position register.

Set both parameters equal to disable software travel limits.

RANGE:
-33,554,000 <= Parameter 1 <= 33,554,000 (Upper Position Limit)
-33,554,000 <= Parameter 2 <= 33,554,000 (Lower Position Limit)

Example: Set logical position limits for the X axis of +/-1,000,000.

Enter: AX TL1000000,-1000000;

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
TL#,#;	AX – AT	3	3
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ?TL, LL, LH, LN, LF, SL, SF

?TL REPORT SOFTWARE TRAVEL LIMITS

The ?TL command reports the software travel limits for the current axis assigned via the TL command. The first value returned is the upper (or "positive") limit and the second value is the lower (or "negative") limit. These are not physical limits but rather positional limits that an axis should not exceed. If one of these limits is exceeded, the PC68 will set the current axis' limit flag and notify the host computer of the condition as though the axis encountered a hard limit.

Example: Find out what the software limits of the Y axis are currently set to.

Enter: `?TL`

Response: `<LF><CR>TL101000,-1000<LF><CR>`

Example: Find out what the software limits of the T axis are currently set to.

Enter: `AT ?TL`

Response: `<LF><CR> TL0,0;<LF><CR>` (software limits for axis T are currently disabled)

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?TL	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: TL

6.6. USER I/O COMMANDS

The following commands are for accessing the bit I/O functions of the board. See also the SW (page 6-96) and WS (page 6-84)commands.

AN AUXILIARY ON

The AN command turns on the selected auxiliary output ports. That is, it allows the open collector line to be pulled high by an external pull up resistor. The AN command may be used to change power level on driver modules so equipped, trigger another board's input or as a user specified output. This is the default mode for the auxiliary line at power up or reset.

A parameter must be supplied for the desired axes when used in the AA mode so that the other axes are not affected. The parameter only serves as a place holder to show which axes should be affected, the value given does not affect the active state of the auxiliary line. No parameter is required in the single axis mode.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	1	1	1
AM	1	1	1
AA/CD	2	2	2

Example: Turn on the Y axis auxiliary output in the single axis mode.

Enter:
```
AY AN
```

Example: Turn on the X and Z axes auxiliary outputs when in the AA command mode. The Y axis is unchanged in this example.

Enter:
```
AA AN1,,1;
```

AF AUXILIARY OFF

The AF command turns off the selected auxiliary outputs. That is, it causes the open collector line to be driven low. The AF command may be used to change power level on driver modules so equipped or as a user specified output. Same parameter rules apply as the AN command.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	1	1	1
AM	1	1	1
AA/CD	2	2	2

Example: Turn off the Y axis auxiliary output in the single axis mode.

Enter: `AY AF`

Example: Turn off the X and Z axes auxiliary outputs when in the AA command mode. The Y axis is unchanged in this example.

Enter: `AA AF1,,1;`

PA# POWER AUTOMATIC

The PA command will turn on or off the auxiliary outputs at the beginning of each GO or GD command execution and complement the outputs after the move is executed. The auxiliary will be turned on, i.e. pulled high, upon the execution of the GO or GD and off at the end of that move, if the parameter is zero or not specified in the single axis mode. If the parameter is non-zero, the sense is reversed, i.e. the auxiliary output is turned off (driven low) upon the execution of the GO or GD command and on at the end of the move.

The SE command can be used to apply a settling time at the end of each move before complementing the auxiliary bit. This is useful for systems that need to retain torque for some specific amount of time before allowing the motor drive to reduce current output.

This mode need only be set once and can be turned off by using the AN or AF command. Axes can be selectively affected in the AA mode by following the syntax as described for the AN command. The values of the included parameters set the state of the auxiliary line during the move. The following queue requirements apply to each GO or GD command in the command stream in the AA and single axis modes. This mode is off by factory default. See the AP Command on page 6-24 to preserve the PA settings as the Power up/Reset values.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	1	1	1
AM	1	1	1
AA/CD	Not Valid		

Example: Turn on the Y axis auxiliary output at the beginning of a move and turn the T axis output off at the beginning of a move, while in the AA command mode. (Note the reversed logic; i.e. 0 = on, 1 = off. “On” pulls the signal line to ground. “Off” lets it rise to 5 volts or its pull-up reference voltage.)

Enter:
```
AA PA,0,,1;
```

Related commands: AN, AF

SE# SETTLING TIME

The SE command allows specification of a settling time, in milliseconds, to be used before the power is reduced, when using the PA mode. The parameter may be any value to 1000 milliseconds. Specification of a parameter of zero turns off the mode. This command is available in single axis mode only. The use of this command requires 3 queue slots with the execution of each GO or GD command. The factory default settling time is zero. See the AP Command on page 6-24 to preserve the SE settings as the Power up/Reset values.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Turn on the Z axis auxiliary output upon execution of a move and have it remain on for 500 milliseconds after the move is complete.

Enter:

```
AZ PA SE500;
```

?SE REPORT SETTLING TIME

The ?SE command reports the settling time setting (SE) used with power automatic mode (PA) for the current axis.

Example: Report the current settling time for axis X

Enter: `AX ?SE`

Response: `<LF><CR>SE250<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?SE	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: SE, ?PA

BL# BIT LOW

The BL command sets the selected general purpose output on (i.e. logic low).

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	3	3
AM	2	3	3
AA/CD	2	2	2

Example: Turn on output bits 10 and 12 after a move. Note that this is only valid for bits which have been configured as outputs and an axis expansion board us used. See the RB command in this section.

Enter:
```
AX MA1000 GO BL10; BL12;
```

BH# BIT HIGH

The BH command sets the selected general purpose output off (i.e. logic high). The state of general purpose outputs is off at power up or reset. Valid bits depend on which bits are programmed as outputs. Factory default output bits are 8 through 13.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	3	3
AM	2	3	3
AA/CD	2	2	2

Example: Set general purpose bits 8 and 11 to high.

Enter:
```
BH8; BH11;
```

IO# SET I/O BIT DIRECTION

The IO command defines the direction of the general purpose I/O bits 0 to 7 as inputs or outputs. The bit direction selection is made in groups of four bits. A null argument skips that bit group. A zero value sets that bit group to be inputs. A one value configures that bit group to be outputs. The factory default configuration is: Bits 0-3 are input and bits 4-7 are output. Addition of the axis expansion board adds input bits 8-11 and output bits 12-15 to the system I/O space.

See the AP Command on page 6-24 to preserve the IO setting as the Power up/Reset setting.

NOTE: Bits 8-15 are not configurable.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Set bits 0-3 as output and bits 4-7 as inputs.

Enter:
```
IO1,0;
```

BX BIT REQUEST IN HEX

The BX command returns the state of the general purpose I/O bits in a six digit hex format, surrounded by line feed and carriage return pairs. The two left hex digits are unused and are always set to 0. A one in any binary position signals that bit as being low.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: User output bits 4 and 7 were previously turned on (i.e. low, ground). Input bits 0 and 3 are on (i.e. low, ground). Check their status with the BX command.

Enter: `BX`

Response: `<LF><CR>99<LF><CR>`

NOTE: When the axis expansion board is used with the PC68, this response expands to four characters. When a Px6IO is added to these, it expands to 16 characters

RB REQUEST BIT DIRECTION

The RB command returns the direction of the general purpose I/O lines as they are currently defined, in hex format surrounded by line feed and carriage return pairs. Output bits return a 1 while input bits return a 0. The two left hex digits are unused and are always set to 0.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Factory default settings have bits 0 through 3 as inputs and 4 through 7 are outputs on the PC68. Verify this with the RB command.

Enter: `RB`

Response: `<LF><CR>F0<LF><CR>`

NOTE: When the expansion board is used with the PC68, this response expands to four characters. When a Px6IO is added to these, it expands to 16 characters.

ADH SET AUXILIARY DEFAULT TO HIGH

The ADH command sets the default powerup or reset state of the auxiliary line for the current axis to high. This change is stored in nonvolatile memory automatically and need not be stored via the AP command. Since this command writes to non-volatile memory it should be used only when necessary and not in repeatedly called functions.

Example: Set the powerup state of the Z axis auxiliary line to high

Enter: `AZ ADH`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
ADH	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ADL, ?AD

ADL SET AUXILIARY DEFAULT TO LOW

The ADL command sets the default powerup or reset state of the auxiliary line for the current axis to low. This change is stored in nonvolatile memory automatically and need not be stored via the AP command. Since this command writes to non-volatile memory it should be used only when necessary and not in repeatedly called functions.

Example: Set the powerup state of the Y axis auxiliary line to low

Enter: `AY ADL`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
ADL	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ADH, ?AD

BS BIT SET

Set all of the output bits to a known state at the same time. This command will affect all output bits, setting their states to the specified mask nearly simultaneously. The mask must be in ASCII hex format where the least significant bit (bit 0) is on the right. To set a line low, the corresponding bit in the hex mask must be a 1. A zero (0) in any bit position will set the corresponding bit high.

Example: Set output 0 high, 1 - 4 low, and 5 - 7 high, (1E = (hex) 00011110)

Enter: `BS1E;`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
BS#;	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: AN, AF, BH, BL, BX

?AD REPORT DEFAULT AUXILIARY BIT STATE

This command will reply with the current powerup default state of the current axis' auxiliary bit as set by ADH or ADL. The response will be either ADH or ADL based on the current powerup state of the auxiliary bit.

Example: Report the powerup state of the Y axis auxiliary bit

Enter: `AY ?AD`

Response: `<LF><CR>ADL<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?AD	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ADL, ADH

?BD# REPORT BIT DIRECTION

The ?BD command reports if a general purpose I/O bit is an input or output.

Example: Find out whether I/O bit 2 is configured as an input or an output.

Enter: `?BD2;`

Response: If the bit is an input the response will be: `<LF><CR>io0<LF><CR>`
An output bit response is: `<LF><CR>io1<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?BD	AX – AS	Immediate	
-	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: BH, BL, BX

?BS# REPORT BIT STATE

The ?BS command reports the state of the specified general purpose I/O bit.

Example: Determine the state of I/O bit 6.

Enter: `?BS6;`

Response: If the bit is set to a TTL high the response will be:
`<LF><CR>1<LF><CR>`
If the bit is set to a TTL low the response will be:
`<LF><CR>0<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?BS	AX – AS	Immediate	
-	AA-AM	Immediate	
-	AA/CD	Not Valid	

Related commands: BH, BL, BX

6.7. MOVE SPECIFICATION COMMANDS

These commands allow specification of move parameters. They allow move parameters to be tailored to the user's system requirements.

AC# ACCELERATION

The AC command sets the acceleration/deceleration register to the operand which follows the command. The parameter must be greater than zero (zero is not valid) and less than 8,000,000, where 2,000,000 is the factory default, and the unit is in steps per second per second. All the following move commands for the axis being programmed will accelerate or decelerate at this rate until another AC command is entered. See the AP Command on page 6-24 to preserve the AC settings as the Power up/Reset values.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	6	15	30
AA	6	15	30
AM	6	15	30
AA/CD	Not Valid		

Example: In the single axis mode, set the Y axis acceleration to 200,000 counts per second per second.

Enter:
```
AY AC200000
```

Example: In the AA mode, set the acceleration of the X axis to 200,000 and the Z axis to 50,000 and leave the other axes with their previous values.

Enter:
```
AA AC200000,,50000;
```

?AC REPORT AC COMMAND

This command will reply with the current acceleration value for the current axis in an AC command format (see below).

Example: Report the current AC value for this axis.

Enter: `?AC`

Response: `<LF><CR>AC200000<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?AC	AX – AT	Immediate	
	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: AC

VL# VELOCITY

The VL command sets the maximum velocity register of the axis being programmed to the operand which follows the command. The operand must be greater than zero and less than or equal to 1,044,000 steps per second, where the factory default is 200,000 steps per second. See the AP Command on page 6-24 to preserve the VL settings as the Power up/Reset values.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	11	26
AA	2	11	26
AM	2	11	26
AA/CD	Not Valid		

Example: In the single axis mode, set the X axis velocity to 10,000 counts per second per second.

Enter:
```
AX VL10000
```

Example: In the AA mode, set the peak velocity of the X axis to 5,000 and the T axis to 50,000 and leave the other axes with their previous values.

Enter:
```
AA VL5000,,,50000;
```

?VL REPORT PEAK VELOCITY SETTING

The ?VL command returns the peak velocity setting for the current axis as set by the VL command.

Example: Make sure our "AXVL50000;" command worked.

Enter: `?VL`

Response: `<LF><CR>vl50000<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?VL	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: VL, ?VB

VB# VELOCITY BASE

The VB command allows the velocity ramp to start at the specified velocity. This allows faster acceleration and the ability to pass through resonance quickly in some applications. The velocity jumps instantly to the specified velocity, then ramps as usual. The deceleration is the same in reverse. This mode is active only for linear ramps. It is ignored for cosine and parabolic ramps but not flagged as a command error. The parameter must be greater than zero and less than the programmed velocity, where the factory default is zero steps per second. This command is not valid with the JG command. See the AP Command on page 6-24 to preserve the VB settings as the Power up/Reset values.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	2	2
AM	2	2	2
AA/CD	Not Valid		

Example: In the single axis mode, set the Y axis velocity base to 200.

Enter:
```
AY VB200
```

Example: In the AA mode, set the X and Y axes velocity bases to 200.

Enter:
```
AA VB200,200;
```

?VB REPORT BASE VELOCITY SETTING

The ?VB command returns the base (starting) velocity setting for the current axis as set by the VB command.

Example: The acceleration ramp should start at 0pps. Make sure we didn't leave it at some other value.

Enter: `?VB`

Response: `<LF><CR>vb1500<LF><CR>` (Oops! We forgot to set it back to zero)

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?VB	AX – AT	Immediate	
	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: VB, ?VL

LP# LOAD POSITION

The LP command will immediately load the number supplied as a parameter in the absolute position registers of the axis. In models with the encoder option, the parameter will be loaded into the encoder position register and the parameter times the encoder ratio will be loaded into the position counter. If no parameter is supplied, the value of zero is used. This command turns off the position hold and interrupt on slip modes when used in a PC68 with the encoder option.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	4	4
AA	2	4	4
AM	2	4	4
AA/CD	Not Valid		

Example: The following would load the X axis position register with 1000, and the Z axis position register with 2000.

Enter: `AA LP1000,,2000;`

Example: The following would load the Y axis position register with 20,000 and the encoder position register with 30,000 counts, in encoder models.

Enter: `AY ER3,2 LP30000`

MA# MOVE ABSOLUTE

The MA command will set up the axis to move to the absolute position supplied as a parameter. The default value of zero is used if no parameter is supplied in the single axis mode. In the AA mode, an axis may remain stationary by entering a comma but omitting the parameter. The move is actually initiated by a GO or GD command.

In the AA mode, each axis will use its predefined acceleration and velocity values to move to the new absolute position. Each axis may, or may not, get to the destination at the same time, because each axis utilizes individual velocities and accelerations.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	31	61
AA	4	33	65
AM	4	33	65
AA/CD	Not Valid		

Example: In the single axis mode, move the X axis to absolute position 100,000 counts with the previously entered acceleration and velocity parameters.

Enter:
```
AX MA100000 GO
```

Example: In the AA mode, move the Y axis to absolute position 10,000 counts and the T axis to absolute position 1,000 counts. The other axes will remain in their current positions.

Enter:
```
AA MA,10000,,1000; GO
```

MR# MOVE RELATIVE

The MR command will set up the axis to move relative from the current position at the time the move is executed. In the AA mode, an axis may remain stationary by entering a comma but omitting the parameter. The move is actually initiated by a GO or GD command.

In the AA mode, each axis will use its predefined acceleration and velocity values to move to the new absolute position. Each axis may, or may not, get to the destination at the same time, because each axis utilizes individual velocities and accelerations.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	31	61
AA	4	33	65
AM	4	33	65
AA/CD	Not Valid		

Example: In the single axis mode, move the X axis 2468 steps in the negative direction.

Enter:
```
AX MR-2468 GO
```

Example: In the AA mode, move the X axis 12345 steps in the positive direction and the Y axis 6789 steps in the positive direction. Both axes will start at the same time.

Enter:
```
AA MR12345,6789; GO
```

ML#,#; MOVE LINEAR

The ML command uses linear interpolation to perform a straight line relative move to the new location. Input parameters are relative distance for each axis in the move. The ML command should be followed by a GO or GD to start the axes together. The velocity and acceleration parameters are scaled to allow the axes to move and finish together. All axes are scaled to the axis with the longest move time.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Not Valid		
AA	4	28	60
AM	4	28	60
AA/CD	Not Valid		

Example: In the AA mode, move the Y, Z and T axes 10000, 100 and 1000 counts respectively with each starting and finishing together. The other axes remain in their previous positions.

Enter:
```
AA ML,10000,100,1000; GO
```

MT#,#; MOVE TO

The MT command uses linear interpolation to move to the specified absolute position. The syntax is similar to the ML command. This command is invalid while in the CN mode, if loops are being used. The command will become valid again after executing an ST or KL command. The MT command is not valid in loops (LS-LE, WH-WG) at anytime. When used in the contour definition mode, only the axes being used in the contour must be provided for in the MT syntax. A GO or GD command initiates the move.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Not Valid		
AA	4	28	60
AM	4	28	60
AA/CD	4 + number of axes		

Example: In the AA mode, move the X, Y and T axes to absolute positions 1000, 10000 and 100 counts respectively, with each starting and finishing together. The unused axes remain in their previous positions.

Enter:
```
AA MT1000,10000,,100; GO
```

MO MOVE ONE PULSE

The MO command will output one step pulse in the current direction (do not use the GO command). The direction may be reversed by use of the MM or MP command. This command generates the output pulse in one sample interval and thus eliminates the latency of generating a ramp with an MR1 GO command sequence. This command is not available in models with an encoder option.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	Not Valid		
AM	Not valid		
AA/CD	Not Valid		

Example: Move the Z axis one pulse in the negative direction.

Enter:
```
AZ MM MO
```

RM# REMAINDER

The RM command will divide the position counter by the parameter supplied and replace the position counter with the resulting remainder. The parameter must be greater than zero and less than 65,000. This command is used in applications where the controller is managing the motion of a continuously rotating object. It allows the position counter to keep track of the absolute position without regard to the number of revolutions it may have rotated. This command has no effect on the encoder position register on boards with the encoder feedback option.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	Not valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: An RM2000 command with a position counter of -4050 will return a position of 1950 since it is within 50 counts of rolling over at -4000 (i.e. the axis is 1950 counts from the starting point.)

DC DECELERATION

The DC command sets a deceleration rate overriding the AC parameter when the GU command is used to initiate a move. Only the GU command will use the DC value. The deceleration rate defaults to 200,000 and will take on whatever value is entered via the AC command. Therefore, the DC command must be reentered after using AC if a different deceleration rate is desired.

RANGE: 1 <= DC <= 8000000

Example: Send the Y axis on a 100,000 count move that accelerates at 100,000 counts per second per second up to 50,000 counts per second and decelerates at 20,000 counts per second per second.

Enter:
```
AY AC100000; DC20000; VL50000; MR100000; GU
```

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
DC#;	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: AC, RC, VL, VB, GU

6.8. MOVE EXECUTION COMMANDS

These commands allow execution of the moves which have been previously specified.

GO **GO** 

The GO command will initiate the move which has been previously programmed with such commands as MA, MR, MT, and ML. No operand is required with the GO command.

To find the total queue requirements for a specific application, find the appropriate value in Table A. If the board is an encoder version, add the value found in Table B to the value from Table A, to determine total queue usage.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	6	9	9
AA	7	10	10
AM	7	10	10
AA/CD	Not Valid		

Example: In the single axis mode, move the X axis to absolute position 12345.

Enter: `AX MA12345 GO`

Example: In the AA mode, move the X axis 2468 steps in the positive direction and the Y axis 2468 steps in the negative direction.

Enter: `AA MR2468,-2468; GO`

GD GO AND RESET DONE

The GD command may be substituted for a GO command. It will reset the done flags, then initiate the move which has been previously programmed with such commands as MA, MR, MT, and ML; just as the GO command does. In the single axis mode, only the done flag for the selected axis will be reset.

In the AA mode, all the done flags will be reset. In the AM mode, the axes involved in the move will be reset. This allows the host to reset the interrupts on the axis involved in the next move, without affecting other axes which may be still active. Note that this command is probably only useful in applications where commands are queued in advance, since the interrupt may be reset before the host has the opportunity to service it, if the GD command is waiting in the queue.

To find the total queue requirements for a specific application, find the appropriate value in Table A. If the board is an encoder version, add the value found in Table B to the value from Table A, to determine total queue usage.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	7	10	10
AA	8	11	11
AM	8	11	11
AA/CD	Not Valid		

Example: In the single axis mode, move the Y axis 12345 steps in the negative direction and set the done flag when the move is completed. Then move it 12345 steps in the positive direction, clear the previous done flag and set the done flag, again, when the move is completed.

Enter:
```
AY MR-12345 GO ID MR12345 GD ID
```

Example: In the AA mode, perform a linear absolute move with the X and Y axes to the position 10000,20000 and set the done flag when the move is completed. Then perform a linear relative move on both axes, moving the X axis 10000 steps in the negative direction and the Y axis 20000 steps in the negative direction.

Enter:
```
AA MT10000,20000; GO ID ML-10000,-20000; GD ID
```

JG# JOG

The JG command is a velocity mode and will step the axis at the velocity supplied as a parameter. The JG command will accelerate to the programmed velocity and run until altered by an ST, SA, KL or another JG command. The jog velocity may be changed by following the command with another JG command of a different velocity. The axis must be stopped before reversing directions. This command modifies the move velocity parameter (VL) for the affected axis. The JG command does not require a GO or GD command to start the motion.

Ramp will be at currently defined acceleration (AC).

Value Range: 0 to 1,044,000

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	5	15	30
AA	5	15	30
AM	5	15	30
AA/CD	Not Valid		

Example: Jog the motor at 100,000 steps per second then change to 35,000 steps per second when the second JG is entered, then stop by decelerating to a stop.

Enter:
```
JG100000 JG35000 ST
```

NOTE: Output events waiting for completion of JG will begin when JG is up to its requested velocity.

Example: Jog the motor at 5000 steps per second in the negative direction.

Enter:
```
JG-5000;
```

JF# JOG FRACTIONAL VELOCITIES

The JF command will jog the axis at the velocity specified, like the JG command. The parameter may include a fractional part allowing better resolution at low speeds. The velocity set by this command will remain the default velocity until altered by a VL, JG or another JF command.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	5	6	6
AA	5	6	6
AM	5	6	6
AA/CD	Not Valid		

Example: Jog the Y axis at 2 2/3 steps per second.

Enter:
```
AY JF2.667
```

TM# TIMED JOG

The TM command performs a jog at the current velocity limits defined for the axis/axes. It then jogs the axis/axes for the specified number of milliseconds. In multi axis mode, all axes begin moving at the same time and stop when their respective jog times have elapsed.

Time range: 0 to 200000 msec.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	7	8	8
AA	7	8	8
AM	7	8	8
AA/CD	Not Valid		

Example: Jog the X axis for 1000 milliseconds.

Enter:
```
AX TM1000;
```

Example: Jog the X axis for 1000 milliseconds and the Z axis for 2000 milliseconds.

Enter:
```
AA TM1000,,2000;
```

VS#,#,# VELOCITY STREAMING

The VS command will generate a pulse train without acceleration or deceleration at the rates specified. The parameters are time in 1/2048 second sample intervals, X velocity, and Y velocity. (The time parameter for more than 4 axes servo models is in 1/1024 second sample intervals.) This is a slave mode and cannot be mixed or queued with other commands. You must be in the AX mode, since the VS command and all parameters are inserted in the X axis command queue. The VS command does not require a GO command to start the motion.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	5	5	5
AA	Not Valid		
AM	Not valid		
AA/CD	Not Valid		

Example: Create a stair step ramp on the X and Y axes, with the X axis moving in the negative direction and the Y axis in the positive direction. Make each step last 1 second long and increase velocity by 1,000 steps/second, until a velocity of 3,000 steps/second is reached, then step down to 0 steps/second. (Assume this is an 8 axes stepper controller.)

Enter:
```
AX VS2048,-1000,1000; VS2048,-2000,2000; VS2048,-3000,3000;
VS2048,-2000,2000; VS2048,-1000,1000; VS1,0,0;
```

GU GO ASYMMETRICAL

The GU command initiates a previously defined move using the AC value for acceleration and the DC value for deceleration. This command may be used with only one axis at a time; i.e. it is *not* valid with the ML and MT commands.

If this command is issued without having defined a move, the results are undefined. Issuing a GU command to execute an already-executed move also has undefined results. Only one GU command should be issued per defined move.

Example: Move the Y axis to position 1,500 using the current acceleration and velocity and a deceleration of 5,000 counts per second per second.

Enter:
```
AY DC5000; MA1500; GU
```

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
GU	AX – AT	3*	11*
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

* AX-AT

- If the controller has an encoder or servo axis add 3
- If PA mode is active add 2
- If an auxiliary output bit settle time has been specified add 3

Related commands: MA, MR, GO, GD, GS

GS GO AND MONITOR SLIP TRIGGER

The GS command works exactly like the GO command except that the home switch will be monitored during the motion. If the home switch becomes active the slip flag will be set for the axis. The host application can read the slip flag and see that the home switch was encountered during the move. This is useful in applications that register slip conditions by means other than encoder position verification; in fact, this command is *not* valid in controls with encoder feedback which includes servo motors.

If this command is issued without having defined a move, the results are undefined. Issuing a GD command to execute an already-executed move also has undefined results. Only one GD command should be issued per defined move.

Example: Move the X axis 50,000 counts in the positive direction. If the motor slips it will close a switch wired to the home input of the X axis. Monitor this switch during the move and set the slip flag for axis X if the switch becomes active.

Enter:
```
AX MR50000; GS
```

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
GS	AX – AT	5*	15*
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

* AX-AT

- If PA mode is active add 2
- If an auxiliary output bit settle time has been specified add 3

Related commands: MA, MR, GO, GD, GU

6.9. MOVE TERMINATION COMMANDS

The following commands allow termination of move sequences in process.

ST STOP

The ST command flushes the queue for the current axis only, in the single axis mode, and causes the axis to decelerate to a stop at the rate previously specified in an AC command. This command is used to stop the motor in a controlled manner from the jog mode or an unfinished GO or GD command. This command is executed immediately. All status and position information is retained. When executed in the AA mode, the ST command is equivalent to the SA command.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	10	28	58
AA	10	28	58
AM	Not Valid		
AA/CD	Not Valid		

Example: Move the Y axis for a while at 1200 steps/second, then ramp to a stop.

Enter: `AY JG1200` (wait awhile) `ST`

SA STOP ALL

The SA command flushes all queues and causes all axes to decelerate to a stop at the rate previously specified in an AC command. All status and position information is retained.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	10	28	58
AA	10	28	58
AM	10	28	58
AA/CD	Not Valid		

Example: Send all axes on a move, then ramp them to a stop, before they finish.

Enter: `AA VL100,100,100,100,100,100,100,100;`
`MR1000,2000,3000,4000,5000,6000,7000,8000; GO` (wait awhile)
`SA`

SD STOP AND RESET DONE

The SD command may be substituted for the SA command. It will reset the done flags, then proceed to stop all axes. This allows the host to be interrupted when all axes have stopped by using the ID command after the SD. The SA ID combination may flag the completion early if one of the axes is already done from a previously executed ID.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	10	28	58
AA	10	28	58
AM	10	28	58
AA/CD	Not Valid		

Example: Flag a done when all axes have stopped.

Enter: `AA SD ID`

KL KILL

The KL command will flush the command queue and terminate pulse generation of all axes immediately. It is intended for emergency termination of any program and to reset the input queues to a known state. The motor may not stop immediately even though no more pulses are delivered due to inertia of the motor rotor and load. Therefore, the position counter may not accurately reflect the true position of the motor following this command. The homing sequence should be used to reestablish the position counters. A Control-D (ASCII 4) will perform the same functions as the KL command. It bypasses the command interpreter and may work when the character buffer is full and the KL command cannot get through the interpreter. A Control-D should be used instead of KL, when the board appears hung-up. This can occur when its input queue is inadvertently filled, by entering a loop sequence that was so long you could not enter the LE command.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	2	2
AM	2	2	2
AA/CD	Not Valid		

Example: Stop all previously defined movement and flush the queue of a partially entered incorrect move command (you wanted a negative move not a positive one), before GO is entered.

Enter: `AX MR5000` (oops!) `KL MR-5000 GO`

FL FLUSH

The FL command will flush an individual axis' command queue. This command is similar in operation to the KL and ST commands except that current motion will remain unaffected by the FL command. All unexecuted commands remaining in the current axis queue will be flushed upon receipt of this command.

Example: Several motion commands have been sent to the X axis but a situation arose and now those commands must be cleared out. The currently executing motion must be allowed to complete to avoid damage to the product.

Enter: `AX FL`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
FL	AX – AT	Immediate	
	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: KL, KS, RS, ST, SA

KS KILL SELECTED AXES

This command performs the same operation as the KL (kill) command except that individual axes can be killed without affecting others. KS will flush only the selected axes' command queues rather than the entire board. Refer to the KL command for more details.

Example: The Y axis has hit a limit switch and is now executing commands that were waiting in the queue. This axis must be reset but the other axes must be allowed to continue operation.

Enter:
```
AY KS
```
or
```
AA KS,1;
```

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
KS	AX – AT	1*	5*
KSb,b,b,b;	AA-AM	1*	5*
-	AA/CD	Not Valid	

* AX-AT
- If PA mode is active add 1
- If an auxiliary output bit settle time has been specified add 3

* AA-AM
- If PA mode is active add 1
- If an auxiliary output bit settle time has been specified add 3

Related commands: KL, RS, ST, SA, SD, SI, SO, FL

SO STOP AT POSITION BY RAMPING FROM DISTANCE

The SO command instructs the PC68 to continue moving until reaching a specified distance (parameter 2) from a specified stop point (parameter 1). The axis will then ramp to a stop within the specified distance. This allows the user to control the point at which deceleration begins, the rate of deceleration, and the stop point, all with a single command.

RANGE:
-33,554,541 <= Parameter 1 (Stop Position)<= 33,554,541
-33,554,541 <= Parameter 2 (Distance from Stop Position to Start Decelerating)<= 33,554,541

Example: The X axis is jogging at 10,000 steps per second. We want the axis to stop at position 50,000 but it must not start ramping until reaching position 46,000.

Enter: `SO50000,4000;`

Related commands: ST, SA, SD, SI, KL, KS

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
SO#,#;	AX – AT	5*	10*
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

*AX-AT:
- If the controller has an encoder or servo axis, add 1
- If PA mode is active, add 1
- If an AUX bit settling time has been specified, add 3

SI STOP INDIVIDUAL

This command can be used to stop only certain axes. In a single axis mode, the SI command behaves identically to ST. In a multiaxis mode, however, SI can be used to stop any number of axes and can be used in place of SA. Like SA, SI will ramp those axes to be stopped using the rate previously specified via the AC command. This command is useful for stopping a specific axis when the current axis mode is unknown and for stopping several axes without affecting current motion on other axes.

Each parameter represents an axis from X through T. Any non-zero value in a parameter will cause the corresponding axis to be stopped.

Example: Start a motion on all four axes. When input bit 1 becomes true, stop axes Y and T without affecting X and Z.

Enter:
```
AM MR15000,30000,20000,40000; GO
SW1; SI,1,,1;
```

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
SI	AX – AT	9*	61*
SIb,b,b,b;	AA-AM	9*	61*
-	AA/CD	Not Valid	

* AX-AT/AA-AM
- PN0 & PR0 ramp selections add 4
- All other PN & PR ramp selections add 18
- CN & SC ramp selections add 18
- SR (Custom) ramp selections add 2*(number of steps in ramp-1)
- If PA mode is active add 1
- If an auxiliary output bit settle time has been specified add 3

Related commands: ST, SA, SD, SO, KL, KS

6.10. LOOP CONTROL COMMANDS

These commands allow move sequences to be repeated within loops. Loops can be nested up to four levels deep on each axis.

LS# LOOP START

The LS command sets the loop counter for the axis being programmed in the single axis mode and all axes in the AA mode. The command expects a loop counter operand following the command. The commands up to the LE loop terminator will be executed the number of times specified by the operand. Loops may be nested up to four levels deep on each axis. The parameter must be less than 32,000.

The first loop of commands will occur immediately as they are entered. The remaining loops will be executed after the loop terminator LE has been entered.

Because of the excess processing overhead involved, the MA command cannot be used in the loop mode, while the board is in the cosine (CN) velocity profile mode, and the MT command cannot be used in the loop mode at any time.

The axis mode (e.g. AX, AY, AA) must be the same when entering and exiting the loop, otherwise the matching loop termination command will not be found by the board's command processor.

If you want one axis to wait for another in the loop, you must be in the AA mode throughout the loop. If you are in the single axis mode in the loop, each axis' commands will go into their separate queues and execute independently of each other.

Another important thing to note is that the command queue size is 200. Each queued command takes one or more slots. If, when entering a looping sequence of commands, all 200 queue slots are filled, before the LE loop terminator is entered, the board will hang. This is because there is no space for the LE command, or any other commands. To clear this hang up, send the board a Control-D (same as KL, but shorter) to kill all moves and flush all queues. When programming a loop of more than four or five moves, the queue size should be examined with the RQ command to see if it is nearing zero.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	2	2
AM	2	2	2
AA/CD	Not Valid		

Example: Execute a 100,000 count relative move on the Z axis 5 times.

Enter: `AZ LS5 MR100000 GO LE`

NOTE: The first move will occur immediately after entering the GO command. The remaining 4 moves will be executed after the loop terminator LE has been entered.

Example: Execute a 100,000 count move relative on the X axis together with a 100 count move on the T axis, followed by a move absolute to 100 counts on the X axis and 200 counts on the T axis, four times.

Enter:
```
AA LS4 MR100000,,,100; GO MA100,,,200; GO LE
```

LE LOOP END

The LE command terminates the most recent LS command. The axis will loop back and repeat the commands within the loop the number of times specified in the LS command. The loop will start repeating as soon as this command is terminated.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	2	2
AM	2	2	2
AA/CD	Not Valid		

Example: (see LS command page 6-82)

WS# WHILE SYNC

The WS command will execute the commands between the WS and WD commands as a loop while the specified general purpose input line is true, i.e. low. When the line goes high it will exit the loop and execute the commands which follow. The test is at the bottom of the loop, i.e. it will always be executed at least once.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	2	2
AM	2	2	2
AA/CD	Not Valid		

Example: Execute a continuous loop, moving the X axis 10,000 counts and then move the Y axis -1000 counts, until an external device terminates the loop.

Enter: AA WS1 MR10000; GO MR, -1000; GO WD

WD WHILE END

The WD command serves as the loop terminator for the WS command.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	2	2
AM	2	2	2
AA/CD	Not Valid		

Example: (see WS command above)

WH WHILE

The WH command will execute all commands between it and the terminating WG command as a loop until terminated by a CW command. This allows repeated execution of a command sequence which can be terminated by the host. These commands may not be nested but may be executed sequentially.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	3	3	3
AA	3	3	3
AM	3	3	3
AA/CD	Not Valid		

Example: You have a 3 axis platform that you use to drill holes in the center of a ¼ inch thick sheet of metal. The sheet is 6 inch square. The driver / motor / lead - screw pitch provide 10000 steps per inch. The operator must manually insert and remove the square from the platform. The X and Y axis move a drill into the desired position. The Z axis lifts and lowers the drill. The operator presses a switch which tells the motion controller that the square is in place and ready to be drilled. The operator will continuously remove and replace the squares until ready to take a break. The following is a description of how to set up an OMS board to perform this task.

Procedure: Connect a normally closed switch between user I/O line 0 and ground. This will be the “Ready to Drill” switch.

Enter:

```
AX UU10000          *set up user units so we can reference move to inches
AY UU10000            *10000 steps = 1 inch
AZ UU10000
AX VL.1; AC10;      *set up X axis homing velocity and acceleration
AY VL.1; AC10;      *set up Y axis homing velocity and acceleration
AZ VL.1; AC10;      *set up Z axis homing velocity and acceleration
AX HR AY HR AZ HR   *send each axis to home
AA VL3,3,.5;        *set normal move velocity for X, Y and Z axes
WH                        *start of loop to drill squares indefinitely
                    *(operator removes/replaces square into platform)
      SW0           *wait until operator presses switch
      MA3,3; GO     *move to center of square
      MA,,.5; GO    *move the drill through the square (a 1/2 inch move
                      on the Z axis drills through the square)
      MA,,0; GO     *lift the drill
      MA0,0; GO     *move the platform to home position
WG                  *loop back to starting WH command
(CW)                *operator wants a break so he/she sends CW from
                      keyboard and presses switch once more (since
                      loop will most likely be waiting for the switch at
                      this point)
                    *the loop ends and the following commands
                      execute
MA0,0,0; GO         *move to home position
```

WG WHILE FLAG

The WG command serves as the terminator for the WH command.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	3	3	3
AM	3	3	3
AA/CD	Not Valid		

Example: (see WH command page 6-85)

CW CLEAR WHILE

The CW command breaks the WH command upon execution of the remaining commands in the loop, i.e. the current execution of the loop is finished. The WH loop is always executed at least one time since the test for the flag is at the bottom.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: (see WH command page 6-85)

6.11. HOME AND INITIALIZATION CONTROL COMMANDS

These commands allow the initialization of the physical stage with the controller.

HM# **HOME**

The HM command will cause the current axis to step in the positive direction at the predefined velocity, until the home input line goes true. The position counter will be initialized to the position supplied as a parameter. The velocity should be less than 2048 counts per second to maintain accuracy of the home position loaded. (For models with more than 4 servo axes, velocity should be less than 1024 counts per second.) The axis will not stop at home, but will initialize the position counter when the home switch becomes true and decelerates to a stop. The axis may be commanded to go home by following this command with a move absolute to the same position as specified in the HM command. The parameter defaults to zero if none is supplied.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	6	8	8
AA	14	34	64
AM	14	34	64
AA/CD	Not Valid		

Example: Find the physical home position of the X axis of the stage. (NOTE: The velocity should be less than 2048 pulses per second (1024 pulses per second for models with more than 4 servo axes.) to minimize position error for this command.) The motor runs until the home switch input is activated and then initializes the position counter to the parameter supplied. Since the motor decelerates to a stop after reaching home, it is necessary to do an MA# to the same position as specified in the home command if it is desired to physically position the device at home. The following commands will find home, initialize it to 1000 counts, then return to home. In many cases it will not be necessary to return home, only find the position and synchronize the controller to it.

Enter:
```
AX VL1000 HM1000 MA1000 GO
```

HR# HOME REVERSE

The HR command will cause the current axis to step in the negative direction at the predefined velocity, until the home input line goes true. It behaves exactly like the HM command, except it travels in the reverse direction. The parameter defaults to zero if none is supplied.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	6	8	8
AA	14	34	64
AM	14	34	64
AA/CD	Not Valid		

Example: In a long stage it may be awkward to travel the full distance to home at less than 2048 pulses per second. The following will get close to home at higher speed, then refine the position at lower speed in the reverse direction.

Enter:
```
AX VL100000 HH HM VL1000 HL HR
```

KM HOME AND KILL

The KM command will find home and stop generating pulses immediately, i.e. no deceleration ramp will be generated. The position counter is not cleared or reset. Due to motor and platform inertia, the load and board may lose position synchronization.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	4	4	4
AA	Not valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: Move the Y axis in a positive direction to the home sensor and stop movement as quickly as possible.

Enter:
```
AY KM
```

KR HOME REVERSE AND KILL

The KR command will find home in reverse and stop generating pulses immediately, i.e. no deceleration ramp will be generated. The position counter is not affected. Due to motor and platform inertia, the load and board may lose position synchronization.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	4	4	4
AA	Not Valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: Move the Y axis in a negative direction to the home sensor and stop movement as quickly as possible.

Enter:
```
AY KR
```

6.12. MOVE SYNCHRONIZATION COMMANDS

These commands allow the synchronization of moves with external events or multiple axis sequences.

ID INTERRUPT DONE

The ID command will set the done flag and interrupt the host if the interrupt has been enabled. This allows the PC68 to signal the host when a string of commands has been completed. In the AA mode, the done flag register bits will be set as each axis encounters the ID in its command stream, but the done flag in the status register will not be set until all axes have executed the ID command. In the AM mode, only the axes active in the most recent move will set their done flags.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	1	1	1
AM	1	1	1
AA/CD	1	1	1

Example: Interrupt the host CPU after the execution of Move Absolute is finished. When the move is finished the ID command will be encountered in the command queue and will set the done flags.

Enter:
```
AX MA100000 GO ID
```

NOTE: The explanation above applies to PC68 boards communicating via the PC/104 bus. When communicating through the RS-232 port, the PC68 will generate a '!' when a DONE is encountered.

II INTERRUPT INDEPENDENT

The II command allows the control to interrupt the host when each axis finishes a move. Only those axes which have been supplied a parameter in the most recent move command will cause interrupts.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Not valid		
AA	1	1	1
AM	1	1	1
AA/CD	Not Valid		

Example: The following command sequence would cause interrupts when the Y and T axes finish. If they do not complete at the same time, two interrupts would be generated.

Enter:
```
MR,1000,,10000; GO II
```

IN# INTERRUPT NEARLY DONE

The IN command allows the control to interrupt the host when the axis or combination of axes is nearly complete. When used in an application involving probing a part after a move, the probes could start accelerating down while the stage is finishing its move, improving the overall system throughput. This command is valid in all modes. The IN command must be entered before the GO or GD command since it is executed before the move is complete. The test is only performed during deceleration. If the IN parameter is greater than the ramp down distance, the interrupt will be generated when the control starts decelerating.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	3	3	3
AA	3	3	3
AM	3	3	3
AA/CD	Not Valid		

Example: The following sequence would interrupt the host when the X axis is complete and the Z axis is within 10,000 counts of being complete. The Y axis completion would be ignored in this example.

Enter:
```
AA
IN0,,10000;
MR100000,100000;GO
MR,,50000;GO
```

IP INTERRUPT WHEN IN POSITION

The IP command operates like the ID command, except the interrupt is deferred until the stage is within the specified deadband. The GD command should be used in place of the GO command to reset the done flags before the next move. If the position hold HN is not enabled for an axis, the command will behave like an ID command for that axis. This command is available only in models with the encoder option.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	1	1	1
AM	1	1	1
AA/CD	Not Valid		

Example: Send DONE when axis is within deadband.

Enter:
```
AX HV1000 HG100 HD10 HN
MR1000 GO IP
```
(DONE will occur after move is complete and in position.)

IC INTERRUPT CLEAR

The IC or the ASCII character Control-Y (hex 19) command is used to clear the done and error flags in the status register and the done flag register. This command will be executed immediately and will usually be placed in the done and error handler interrupt service routine to clear the interrupt and the associated flags. The Control-Y version of this command is preferred to minimize the latency in its execution. The flags may be polled by an RA or RI command which will also reset the flags.

This command is **not** recommended unless used to clear flags after they have been read via the QA or QI commands. These flags are otherwise automatically cleared when read via the RA or RI commands, or by directly accessing the registers on the board.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Clear the flags after an X axis move relative of 5000 steps was flagged as done when an ID executes.

Enter: `AX MR5000 GO ID` (done flag set) `IC`

CA CLEAR AXIS DONE FLAG

The CA command operates like the IC command, except it clears the done flag of the addressed axis only.

This command is **not** recommended unless used to clear flags after they have been read via the QA or QI commands. These flags are otherwise automatically cleared when read via the RA or RI commands, or by directly accessing the registers on the board.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: After a multi-axis move, clear the Z axis done status only.

Enter:
```
AA MR1000, 2000, 3000, 4000; GO ID
AZ CA
```

Example: After a multi-axis move, clear the Y and Z axis done status only.

Enter:
```
AA MR1000, 2000, 3000, 4000; GO ID
CA,1,1;
```

WA WAIT FOR AXES

The WA command, only valid in the AA mode, allows a command to wait until all moves on all axes are finished before it executes.

Some commands which can affect a non-moving axis, such as AN, AF and PA, may execute before a previous move on other axes has finished, especially while in the looping (LS-LE, WH-WG) mode. By preceding these commands with a WA, they will not execute until all previously defined moves have finished.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Not Valid		
AA	2	2	2
AM	2	2	2
AA/CD	Not Valid		

Example: The Z axis auxiliary line controls a laser beam that you only want on while the Z axis moves in a positive direction. The X and Y axes position the laser. You want to repeat the action 10 times.

Enter:
```
AA VL1000,1000,1000; AC10000,10000.10000;
LS10 MR1000,1000; GO WA AN,,1; MR,,500; GO AF,,1;
MR,,-500 GO LE
```

WQ WAIT FOR QUEUE TO EMPTY

The WQ command is a special command that stops the board from processing any new command until the queue for the current axis mode is empty, i.e. all previous moves have finished. This command is not valid in looping (LS-LE, WH-WG) mode.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Move the Y axis 1,000 steps and wait until the move is complete before asking for the position.

Enter:
```
AY MR1000 GO WQ RP
```

SW# SYNC WAIT

The SW command allows synchronization of multi-axis moves or other tasks on one or more PC68 boards by using one of the general purpose input lines. This command causes the axes to wait until the general purpose input line has been released (allowed to go high) before proceeding with the next command. The SW command can be used to cause an axis to wait until the others are finished. Wire OR the auxiliary lines from several axes together and connect them to a general purpose input line. Use the SW command on that line. All commands after that will wait until all axes release their auxiliary lines.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	2	2
AM	2	2	2
AA/CD	Not Valid		

Example: The following command sequence will cause the X axis move to wait until the Y axis has finished its move and turned off its auxiliary output which has been wired to the general purpose input 0 line.

Enter:
```
AY AN MR2000 GO AF
AX SW0 MR10000 GO
```

The SW command provides a way to synchronize moves on two or more boards. The following example shows one way to do this.

Example: You have 3 eight axis boards, for a total of 24 axes to move together. Call board 1 the “master” and boards 2 and 3 the “slaves”. Wire board 1’s X axis auxiliary line to the two slave boards’ general purpose input 0 line. Send to the master the command “AX PA0”, setting the master’s X axis auxiliary line low until its move starts. This also sets the slaves’ general purpose input 0 line low. Enter the “SW0” command to the two slaves, followed by the move and GO commands. On the master, enter the move command, followed by the GO command. When the master’s move starts, the PA command will set the auxiliary line high releasing the wait on the slave boards. All three boards will start their moves.

Procedure: Wire board 1’s X axis auxiliary line to board 2’s and board 3’s general purpose input 0 line.

Enter:
(Board 1) `AX PA0;`
(Board 2) `AA SW0; MR200,200,200,200,200,200,200,200; GO`
(Board 3) `AA SW0; MR300,300,300,300,300,300,300,300; GO`
(Board 1) `AA MR100,100,100,100,100,100,100,100; GO`

WT# WAIT

The WT command will wait for the specified number of milliseconds before proceeding with the next command in the queue. In the AA mode, all axes will wait. Immediate commands will not "wait". The parameter must be between 1 and 32,000.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	3	3	3
AA	3	3	3
AM	3	3	3
AA/CD	Not Valid		

Example: You want to produce pulses on the X axis at 5,000 steps/second for 2 seconds, then 10,000 pulses/second for 3 seconds, then stop.

Enter: `AX JG5000 WT2000 JG10000 WT3000 JG0`

BW WAIT FOR INPUT TO GO LOW

The BW command is just like the SW command except that it waits for the input line to reach a TTL low rather than a TTL high. Refer to the SW command for more detail.

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
BW#;	AX – AT	2	2
BW#;	AA-AM	4	4
-	AA/CD	Not Valid	

Example: See the examples for the SW command

Related commands: WT, WA, WQ, SW

IX INTERRUPT WHEN AXES DONE

The IX command is a special form of the ID command. It is intended for use by the serial communications version of the PC68. Each axis which has been supplied a position in the most recent move command will send a special done character to the host as they complete their move.

AXIS DONE INDICATOR CHARACTERS	
AXIS	DONE CHARACTER IN HEX
X	80
Y	81
Z	82
T	83
U	84
V	85
R	86
S	87

Example: The following command sequence would cause two characters to be sent to the host. An 81 hex character will be sent when the Y axis finishes and an 83 hex character when the T axis finishes.

Enter: AM MR,1000,,10000; GO IX

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
-	AX – AT	Not Valid	
IX	AA-AM	1	1
-	AA/CD	Not Valid	

Related commands: ID, IP, IN

6.13. SYSTEM STATUS REQUEST COMMANDS

These commands allow the host to request the status of various move parameters, including the status of limit and home switches.

WY WHO ARE YOU

The WY command returns the model type, firmware revision number, and number of controlled axes of the board being addressed, surrounded by line feeds and carriage returns.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: You want to examine the board information.

Enter: `WY`

Response: `<LF><CR>PC68 ver 3.00-25 S/N001234<LF><CR>`

RP REQUEST POSITION

The RP command returns the current position of the currently addressed axis in the single axis mode or all positions separated by commas in the AA or AM modes. The position will be returned to the host via the data port in ASCII format. This command is not queued, i.e. the current position will be returned immediately even if the axis is in motion. The response is surrounded by line feeds and carriage returns.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: The current position on the Y axis is 12345. Use the RP command to verify the position.

Enter: `AY RP`

Response: `<LF><CR>12345<LF><CR>`

PE REPORT ENCODER POSITIONS

PE reports the encoder positions of all encoder and/or servo axes.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Report the encoder positions of a PC68-45 (four axis servo board).

Enter: `PE`

Response: `<LF><CR>0,50,156,0<LF><CR>`

PP REPORT MOTOR POSITIONS

The PP command reports the motor positions of all axes.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Report the motor positions of a four axis controller.

Enter: `PP`

Response: `<LF><CR>0,0,0,125<LF><CR>`

RQ REQUEST QUEUE STATUS

The RQ command returns the number of entries available in the queue of the currently addressed axis, in the single axis mode, or all axes separated by commas, in the AA or AM modes. The ASCII string is surrounded by line feeds and carriage returns. The maximum available in each command queue is 200. The response is at a fixed length of 3 characters. For example, if the current free queue space is 67, the response from the board to the RQ command is `<LF><CR>067<LF><CR>.`

When issuing an RQ command, while defining a contour, the available space in the contouring queue will be returned. The maximum available is 1016. The response is fixed in length at 4 characters.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Immediate		

Example: See the size of the command queue for the T axis.

Enter: `AT RQ`

Response: `<LF><CR>200<LF><CR>`

BX BIT REQUEST IN HEX

The BX command returns the state of the general purpose I/O bits in a six digit hex format, surrounded by line feed and carriage return pairs. The two left hex digits are unused and are always set to 0. A one in any binary position signals that bit as being low.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: User output bits 10 and 12 were previously turned on (i.e. low, ground). Input bits 0 and 3 are on (i.e. low, ground). Check their status with the BX command.

Enter: `BX`

Response: `<LF><CR>001409<LF><CR>`

RA REQUEST AXIS STATUS

The RA command returns the state of the limit and home switches, and the done and direction flags for the currently addressed axis. The limit flag in the hardware status register will be reset by the RA command, providing another axis is not in limit. The done flag register will also be reset by this command. The status is returned in the following format:

CHARACTER MEANING		
CHAR	**SENT**	**DESCRIPTION**
1	LF	Line feed
2	CR	Carriage return
3	CR	Carriage return
4	P	Moving in positive direction
	M	Moving in negative direction
5	D	Done (ID, II or IN command has been executed, set to N by this command or IC command)
	N	No ID executed yet
6	L	Axis in overtravel. Char 4 tells which direction. Set to N when limit switch is not active.
	N	Not in overtravel in this direction
7	H	Home switch active. Set to N when home switch is not active.
	N	Home switch not active
8	LF	Line feed
9	CR	Carriage return
10	CR	Carriage return

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Not valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: The Y axis just encountered a limit, verify its status.

Enter: `AY RA`

Response: `<LF><CR><CR>PNLN<LF><CR><CR>`

RI REQUEST INTERRUPT STATUS

The RI command is an AA mode command that returns the same status information on all axes as the RA command in the single axis mode. The 4 character fields for each axis are separated by commas and the string has one line feed and two carriage returns on each end. The done flag is reset by this command.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Not valid		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Check the status of a 4 axis board.

Enter: `AA RI`

Response: `<LF><CR><CR>MDNN,MDNN,MDNN,MDNN<LF><CR><CR>`

QA QUERY AXIS

The QA command returns the status of the single addressed axis like the RA command, except flags are not affected.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Not valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: Check the status of the X axis.

Enter: `AX QA`

Response: `<LF><CR><CR>PNNH<LF><CR><CR>`

QI QUERY INTERRUPT STATUS

The QI command returns the same information for all axes when in the AA mode, as the QA command does in the single axis mode. The 4 character fields for each axis are separated by commas and the string has one line feed and two carriage returns on each end.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Not Valid		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Check the status of a four axis board.

Enter: `AA QI`

Response: `<LF><CR><CR>PNNN,MNNN,PDNN,MNLN<LF><CR><CR>`

RC REQUEST ACCELERATION

The RC command will return the current acceleration or deceleration of the current axis. This may differ from the programmed acceleration if a cosine (CN) or parabolic (PN) ramp is being generated. When the stage is stopped, the parameter returned will be the acceleration at the beginning of a ramp. When the stage is running at programmed speed, i.e. not accelerating, the parameter returned will be the acceleration at the end of the ramp. While a contour is executing, the value computed to generate the appropriate lead in will be returned. The response to the RC command is surrounded by line feed and carriage return pairs.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Display current acceleration values for all axes on a four axis board.

Enter: `AA RC`

Response: `<LF><CR>2000000,2000000,2000000,2000000<LF><CR>`

RV REQUEST VELOCITY

The RV command will return the current velocity at which the axis is moving. This may differ from the programmed velocity if the axis is ramping up to speed or stopping. The response is surrounded by line feed and carriage return pairs. If the JF command is executing, the command only reports the integer part of the velocity.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Jog the Y axis at 12345 steps per second.
Display the current velocity.

Enter:
```
AY JG 12345
RV
```

Response: `<LF><CR>12345<LF><CR>`

RU REPORT POSITION IN USER UNITS

The RU command returns the current position in user units (see UU command on page 6-107). The format of response is a floating point number with five characters to the right of the decimal point. This response is surrounded by line feed and carriage return pairs.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: One revolution of a motor is 2000 steps. Define user units so moves can be referenced in revolutions. Move the Z axis 3 1/2 revolutions. Use RU to display the position when the move is complete.

Enter:
```
AZ UU2000; LP0;
MR3.5; GO
(Wait until move is complete.)
RU
```

Response: `<LF><CR>3.50000<LF><CR>`

6.14. USER UNIT COMMANDS

The following commands allow specification of move parameters in user defined units. The OMS controls will automatically convert all move parameters to these units once they have been initialized.

UU# USER UNITS

The UU command converts all move velocities, distances, etc. to user specified units by multiplying by the specified parameter. The UF command is used to terminate this mode. Factory default is with this command off. See the AP Command on page 6-24 to preserve the UU settings as the Power up/Reset values.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: The motor, driver and gear ratio you are using requires 10,000 steps to move one inch. Set up the X, Y and Z axes so you can enter move information in inches.

Enter:
```
AX UU10000 AY UU10000 AZ UU10000
        or
AA UU 10000,10000,10000;
```

?UU REPORT AXIS USER UNITS' AXIS ASSIGNMENT

This command returns the current user units' multiplier as set via the UU command.

Example: Make sure the UU512 command we sent earlier is still current. The command will return the UU value with six digits to the right of the decimal point. If the UU value exceeds six digits for the fractional value, the value will be rounded off to the sixth decimal place.

Enter: `?UU`

Response: `<CR><LF>uu512.000000<CR><LF>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?UU	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

If user units are turned off ?UU returns:

`<CR><LF>UF<CR><LF>`

Related commands: UU, UF

UF USER OFF

The UF command turns off user units, it is the factory default setting for all axes. This command is equivalent to and preferred over UU1 since it turns off the mode thus minimizing unnecessary overhead. See the AP Command on page 6-24 to make the UF settings as the Power up/Reset values.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Turn off user unit conversion on the X, Y and Z axes.

Enter:
```
AX UF AY UF AZ UF
      or
AA UF 1,1,1;
```

6.15. PID FILTER CONTROL COMMANDS

The following commands are valid only for servo axes and should never be executed while the specific axis is in motion.

DZ# DEFINE ZERO POSITION IN OPEN-LOOP MODE

The DZ command defines the offset coefficient needed to produce a zero or stationary position by the servo motor. This command is used in the open-loop mode, hold off (HF). The factory default value is zero. See the AP Command on page 6-24 to preserve the DZ settings as the Power up/Reset values.

Value range: -32640 to +32640

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	Not valid
AA	Not Valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: Define the offset coefficient to be 250 for the X axis.

Enter: `AX DZ250`

?DZ REPORT DAC OPEN-LOOP OFFSET

The ?DZ command reports the current setting of the DZ command in DZ command format.

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?DZ	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Example: The closed-loop offset needs to be set the same as the open-loop offset. We've already set the open-loop offset but forgot what value we used. Send the ?DZ command to find out.

Enter: `?DZ`

Response: `<LF><CR>DZ28<LF><CR>`

Related commands: DZ, ?KO

KA# ACCELERATION FEEDFORWARD

KA is the acceleration feedforward coefficient used in the PID filter calculations. Refer to Section 2 for more information regarding this parameter. The factory default value is zero. See the AP Command on page 6-24 to preserve the KA settings as the Power up/Reset values.

Value range: 0 to 4096

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	Not Valid		
AM	Not valid		
AA/CD	Not Valid		

Example: Define KA to be 2 on the T axis.

Enter: `AT KA2`

?KA REPORT ACCELERATION FEED-FORWARD

The ?KA command reports the current setting of the acceleration feed-forward constant (KA) for the current servo axis.

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?KA	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Example: Find out what the current KA value is for servo axis Y

Enter: `AY ?KA`

Response: `<LF><CR>KA10.50000<LF><CR>`

Related commands: KA, ?KV

KD# DERIVATIVE GAIN COEFFICIENT

KD is the derivative gain coefficient used in the PID filter calculations. See Section 2 for more information regarding this parameter. The factory default value is 20.0. See the AP Command on page 6-24 to preserve the KD settings as the Power up/Reset values.

Value range: 0 to 4096

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	Not valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: Set KD to 56 on the Z axis.

Enter: `AZ KD56`

?KD REPORT PID DERIVATIVE GAIN

The ?KD command reports the current setting of the derivative gain constant (KD) in the PID of the current servo axis.

Example: Forgot to write down the KD setting which is working well. Report the setting so it can be recorded.

Enter: `?KD`

Response: `<LF><CR>KD5.12500<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?KD	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: KD, ?KP, ?KI

KI# INTEGRAL GAIN COEFFICIENT

KI is the integral gain coefficient used in the PID filter calculations. See Section 2 for more information regarding this parameter. The factory default value is 0.04. See the AP Command on page 6-24 to preserve the KI settings as the Power up/Reset values.

Value range: 0 to 4096

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	Not valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: Define KI to be 3.42 on the X axis.

Enter: `AX KI3.42`

?KI REPORT PID INTEGRAL GAIN

The ?KI command reports the current setting of the integral gain constant (KI) in the PID of the current servo axis.

Example: Report the setting of the KI command on the Z axis

Enter: `AZ ?KI`

Response: `<LF><CR>KI0.01000<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?KP	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: KI, ?KP, ?KD

KO# OFFSET COEFFICIENT IN CLOSED-LOOP MODE

The KO command defines the offset coefficient to cause the motor to remain stationary and compensate for additional torque on the motor from loading. This command is used in the closed-loop mode, hold on (HN). The factory default value is zero. See the AP Command on page 6-24 to preserve the KO settings as the Power up/Reset values.

The factory default value is zero. Full-scale, the KO command has a range of +/- 32,667 which corresponds directly to the 16-bit range of the DAC less a few counts as a buffer zone. Each increment/decrement of the KO value will result in an approximate change in the output voltage of 0.0003 volts. See the AP Command on page 6-24 to preserve the KO settings as the Power up/Reset values.

This command affects the offset only when in closed-loop mode. The DZ command is used to set the offset when the loop is open. Typically, the value found satisfactory with the KO command will also be used with the DZ command. Since this is not always the case, KO and DZ are available to set differing offsets.

RANGE: -32,667 <= KO <= 32,667

Example: Define the offset coefficient to be –2000 (~ -610mV) on the Y axis.

Enter: `AY KO-2000;`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
KO#;	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: ?KO, DZ, HN, HF

?KO REPORT PID CLOSED- LOOP OFFSET

The ?KO command reports the closed-loop voltage offset (KO) setting for the current servo axis. See the ?DZ command for reporting open-loop offset.

Example: The open-loop offset is 218. Make sure the closed-loop offset is the same.

Enter: `?KO`

Response: `<LF><CR>KO218<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?KO	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: KO, ?DZ

KP# PROPORTIONAL GAIN COEFFICIENT

KP is the proportional gain coefficient used in the PID filter calculations. See Section 2 for detailed information regarding this parameter. The factory default value is 10.0. See the AP Command on page 6-24 to preserve the KP settings as the Power up/Reset values.

Value range: 0 to 4096

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	Not valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: Define KP to be 45.6 on the Z axis.

Enter: `AZ KP45.6`

?KP REPORT PID PROPORTIONAL GAIN

The ?KP command reports the current setting of the proportional gain constant (KP) in the PID of the current servo axis.

Example: Find out what the X axis proportional gain is set to.

Enter: `AX ?KP`

Response: `<LF><CR>KP10.00000<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?KP	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: KP, ?KI, ?KD

KV# VELOCITY FEEDFORWARD

KV is the velocity feedforward coefficient used in the PID filter calculations. See Section 2 for a more detailed explanation of this parameter. The factory default value is zero. See the AP Command on page 6-24 to preserve the KV settings as the Power up/Reset values.

Value range: 0 to 4096

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	Not valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: Set KV to 35.3 on the Y axis.

Enter: `AY KV35.3`

?KV REPORT VELOCITY FEEDFORWARD

The ?KV command reports the current velocity feed-forward setting (KV) of the current servo axis.

Example: Make sure the velocity feed-forward setting of axis T is zero

Enter: `AT ?KV`

Response: `<LF><CR>KV0.00000<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?KV	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: KV, ?KA

HF HOLD OFF

The HF command disables position hold, stall detection and tracking modes.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	2	2
AM	2	2	2
AA/CD	Not Valid		

Example: Turn off encoder hold mode on the X axis.

Enter:
```
AX HF
```

HN HOLD ON

The HN command enables position correction after a move and activates the HV, HG and HD commands. Hold and slip detection are disabled if an LP, HM, HR, SA, ST or KL command is entered or if a limit is encountered.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	3	3
AA	1	3	3
AM	1	3	1
AA/CD	Not Valid		

Example: The following commands could be used to set up the position correction mode. This sequence sets up a move velocity of 100,000 steps per second and an acceleration of 500,000 steps per second per second. The position correction velocity is set for 50,000 steps per second, a deadband of 10 steps and correction gain of 2,000. The correction is then enabled. A 200,000 step move is performed, then that position is maintained within the 10 step deadband until commanded to a new position.

Enter:
```
AX VL100000 AC500000
HV50000 HD10 HG2000 HN
MR200000 GO
```

?PM REPORT PID STATE

The ?PM command reports whether the PID for the current servo axis is enabled. The HN and HF commands are used to enable and disable the PID and are the possible responses from this command.

Example: A limit switch was hit by servo axis Y. See if the PID is still enabled for that axis.

Enter: `AY ?PM`

Response: `<LF><CR>HF<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?PM	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: HN, HF

BI BIPOLAR

The BI command sets the analog and PWM torque outputs to bipolar. When bipolar is selected, a zero torque reference will result in a 50% PWM duty cycle. A negative torque reference will result in less than 50% PWM duty cycle; a positive torque reference will result in greater than 50% PWM duty cycle. The analog output will range between +10VDC and -10VDC when bipolar is enabled. It is necessary to issue either the UN or the BI command to enable PWM operation for a particular axis. The BI command is valid only in the single axis mode. This is the default mode at power up or reset.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Set up servo axis X for bipolar operation.

Enter: `AX BI HN`

UN UNIPOLAR

The UN command sets the analog and PWM torque outputs to unipolar. When unipolar is selected, a zero torque reference will result in a low DC level (or minimum PWM duty cycle) and maximum torque reference will result in a high DC level (or maximum PWM duty cycle). The analog output will range between 0.0VDC and +10VDC when unipolar is enabled. The direction output signal will define the sign of the output. It is necessary to issue either the UN or the BI command to enable PWM operation for a particular axis. This command is valid only in the single axis mode.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Not valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: Set up servo axis X for unipolar operation.

Enter:
```
AX UN HG5,200,100 HN
```

?SO REPORT ANALOG OUTPUT MODE

The ?SO command reports whether the analog output type for the current servo axis is bipolar or unipolar. The possible responses are BI and UN, the same commands used to set one mode or the other.

Example: The Y axis should be setup with unipolar outputs. Use ?SO to make sure.

Enter:
```
AY ?SO
```

Response:
```
<LF><CR>UN<LF><CR>
```

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?SO	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: UN, BI

6.16. POSITION MAINTENANCE COMMANDS

ER#,# ENCODER RATIO

The ER command allows specification of encoder ratio by entering encoder counts, followed by stepper motor counts, for position maintenance mode. **This command is not designed for use with servo motors.** These counts must be integers unless user units are enabled. The ratio of encoder counts to motor counts must be equal to one, i.e. encoder counts must match motor counts when slip detection is enabled. All distance, velocity and acceleration parameters are input in encoder counts when this mode is enabled. The correct number of motor counts are generated, while the user need only be concerned with encoder counts. This mode can be combined with user units, allowing units such as inches or revolutions to be specified in encoder counts. All parameters are then input in the user units which have been defined. The factory default setting is 1.1. See the AP Command on page 6-24 to preserve the ER settings as the Power up/Reset values.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	Not valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: You have an encoder connected, through a series of gears, to a stepper motor. When the motor steps 25,000 times, the encoder produces 10,000 counts. Set up an encoder ratio so the hold mode will work correctly.

Enter: `ER10000,25000`

?ER REPORT MOTOR:ENCODER RATIO

The ?ER command reports the motor-to-encoder ratio as set with the ER command.

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?ER	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Example: Find out what the last ER command sent was.

Enter: `?ER`

Response: `<LF><CR>er2.000000<LF><CR>`
(The encoder produces 1 count for every 2 steps of the motor.)

Related commands: ER

HV# HOLD VELOCITY

The HV command specifies maximum position hold correction velocity. This is the peak velocity which will be used while making position corrections. The factory default setting is zero. See the AP Command on page 6-24 to preserve the HV settings as the Power up/Reset values. **This command is not designed for use with servo motors.**

Value range: 0 to 1,044,000

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	2	2
AM	2	2	Immediate
AA/CD	Not Valid		

Example: (see HN command page 6-127)

HG# HOLD GAIN

The HG command allows the user to specify position hold gain parameter. This gain parameter is multiplied by the position error in determining the velocity during correction. This command is designed to work with stepper motor applications using encoders and is not designed for use with servo motors. The parameter must be between 1 and 32,000. The parameter should be set experimentally by increasing it until the system is unstable then reducing it slightly below the threshold of stability. The factory default value is 1. See the AP Command on page 6-24 to preserve the HG settings as the Power up/Reset values.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: (see HN command page 6-127)

HD# HOLD DEADBAND

The HD command specifies deadband counts for position hold. If the stage is within this limit, it is considered in position and no further correction will be made. This parameter interacts with the HG command, i.e. a larger deadband will allow a larger gain parameter in many applications. This command is designed to work with stepper motor applications using encoders and is not designed for use with servo motors. The factory default value is zero. See the AP Command on page 6-24 to preserve the HD settings as the Power up/Reset values.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	2	2
AM	2	2	2
AA/CD	Not Valid		

Example: (see HN command page 6-127)

HF HOLD OFF

The HF command disables position hold, stall detection and tracking modes as well as the servo PID. This is the default mode at power up or reset.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	2	2
AM	2	2	2
AA/CD	Not Valid		

Example: Turn off encoder hold mode on the X axis.

Enter: `AX HF`

?PM REPORT PID STATE

The ?PM command reports whether the PID for the current servo axis is enabled. The HN and HF commands are used to enable and disable the PID and are the possible responses from this command.

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?PM	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Example: A limit switch was hit by servo axis Y. See if the PID is still enabled for that axis.

Enter: `AY ?PM`

Response: `<LF><CR>HF<LF><CR>`

Related commands: HN, HF

HN HOLD ON

The HN command enables position correction after a move and activates the HV, HG and HD commands as well as all of the PID filter commands. On stepper motor axes Hold and slip detection are disabled if an LP, HM, HR, SA, ST or KL command is entered or if a limit is encountered.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	3	3
AA	1	3	3
AM	1	3	1
AA/CD	Not Valid		

Example: The following commands could be used to set up the position correction mode. This sequence sets up a move velocity of 100,000 steps per second and an acceleration of 500,000 steps per second per second. The position correction velocity is set for 50,000 steps per second, a deadband of 10 steps and correction gain of 2,000. The correction is then enabled. A 200,000 step move is performed, then that position is maintained within the 10 step deadband until commanded to a new position.

Enter:
```
AX VL100000 AC500000
HV50000 HD10 HG2000 HN
MR200000 GO
```

IP INTERRUPT WHEN IN POSITION

The IP command operates like the ID command, except the interrupt is deferred until the stage is within the specified deadband. The GD command should be used in place of the GO command to reset the done flags before the next move. If the position hold HN is not enabled for an axis, the command will behave like an ID command for that axis.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	1	1	1
AM	1	1	1
AA/CD	Not Valid		

Example: Send DONE when axis is within deadband.

Enter:
```
AX HV1000 HG100 HD10 HN
MR1000 GO IP
```
(DONE will occur after move is complete and in position.)

?ES REPORT ENCODER SLIP TOLERANCE

The ?ES command reports the current setting of the ES command.

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?ES	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Example: Report the current deadband for encoder slip detection tolerance

Enter:
```
?ES
```

Response:
```
<LF><CR>ES15<LF><CR>
```

Related commands: ES

?HD REPORT POSITION MAINTENANCE DEADBAND

The ?HD command reports the current setting of the HD command. This command will only work with stepper axes with encoders.

Example: Find out what HD was last set to.

Enter: `?HD`

Response: `<LF><CR>HD5<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?HD	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: HD, ?HG, ?HV

?HG REPORT POSITION MAINTENANCE GAIN

The ?HG command reports the current setting of the HG command for the current axis. This command works only with stepper + encoder axes.

Example: Position corrections seem slow. Check the setting of HG to be sure it is correct.

Enter: `?HG`

Response: `<LF><CR>HG100<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?HG	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: HG, ?HV, ?HD

?HV REPORT POSITION MAINTENANCE VELOCITY

The ?HV command reports the current setting of the HV command for the current axis. This command works only with stepper + encoder axes.

Example: Check the peak correction velocity for the T axis

Enter: `AT ?HV`

Response: `<LF><CR>HV20000<LF><CR>`

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
?HV	AX – AT	Immediate	
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: HV, ?HG, ?HD

6.17. SLIP AND STALL DETECTION COMMANDS

ES# ENCODER SLIP TOLERANCE

The ES command parameter specifies tolerance before slip or stall is flagged in the status register and by the RL command. The mode must be turned on with an IS command and off with an HF command. The factory default value is 1. This command is intended to be used with stepper motors and not servo motors. See the AP Command on page 6-24 to preserve the ES settings as the Power up/Reset values.

Value Range: 1 to 65535

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	Not valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: Your application can tolerate being up to 5 steps from the desired position before the controlling program should be notified of a slip condition.

Enter: `ES5 IS`

IS INTERRUPT ON SLIP

The IS command enables the PC68 to interrupt the host on slip or stall detection, if the appropriate bit has been set in the interrupt control register. Hold and slip detection are disabled if an LP, HM, HR, SA, ST or KL command is entered or if a limit is encountered. If a slip occurs, slip detection must be re-enabled. The factory default value is 1. This command is intended to be used with stepper motors and not servo motors.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	Not valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: (see ES command above)

RL RETURN SLIP STATUS

The RL command returns the slip detection status of each axis. An S is returned if slip has occurred for that axis, or else an N is returned. The results are bounded by an LF CR pair, as in other status commands. The number of characters returned corresponds to the number of axes available on the board. This command is intended to be used with stepper motors with encoders and not with servo motors.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: On a four axis board, see if any axis has slipped.

Enter: `RL`

Response: `<LF><CR>NNSN<LF><CR>` (The Z axis has slipped.)

HF HOLD OFF

The HF command disables position hold, stall detection and tracking modes.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	2	2
AM	2	2	2
AA/CD	Not Valid		

Example: Disable slip detection on the X axis.

Enter: `AX HF`

TF TURN OFF SLIP KILL MODE

The TF command disables slip kill mode (enabled with TN.)

Example: Slip kill mode is enabled but a move needs to be performed where slip is likely and not important for this move. Disable slip kill mode.

Enter:
```
TF
```

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
TF	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: TN, ES, IS, RL

TN TURN ON SLIP KILL MODE

The TN command enables slip kill mode. In this mode, if the motor slips beyond the deadband set by the ES command, the PC68 will kill motion on the axis that slipped as though a KL command had been issued to the axis. This mode can be disabled (default) with the TF command.

Example: X axis is sent on a move. Its encoder cable was not connected to the controller (oops!). The controller issues a KL (Kill) command to the X axis after receiving the slip interrupt. The slip interrupt is generated once the difference between the motor position counts and encoder counts exceed 20.

Enter:
```
AX ES20; TN; IS; LP0;
MA30; GO
```

QUEUE REQUIREMENTS			
FORMAT	**MODE**	**Min**	**Max**
TN	AX – AT	2	2
-	AA-AM	Not Valid	
-	AA/CD	Not Valid	

Related commands: TF, ES, IS, RL

6.18. ENCODER TRACKING COMMANDS

ET ENCODER TRACKING

The ET command turns on the encoder tracking mode. The axis will track its encoder input, thus allowing one axis to follow the activity of another or a thumbwheel for manual positioning or the movement of another device that produces a signal compatible to the encoder inputs. No acceleration or deceleration ramps are generated. The axis will duplicate the encoder input. The ER command allows the user to scale the motor's movements relative to the encoder. This command is intended to be used with stepper motors with encoders and not with servo motors.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	Not valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: Set up the X axis so it will follow its encoder input.

Enter: `AX ET`

HF HOLD OFF

The HF command disables position hold, stall detection and tracking modes.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	2	2
AM	2	2	2
AA/CD	Not Valid		

Example: Turn off encoder tracking on X axis.

Enter: `AX HF`

6.19. ENCODER HOME CONTROL COMMANDS

HE HOME ENCODER

The HE command enables encoder index mode when an HM or HR command is executed. Home is defined as the logical AND of the encoder index, the external home enable and the encoder quadrant where channel A is positive and channel B is negative. The external enable is low true, i.e. the HH and HL commands are not valid in this mode. The home logic expressed in Boolean terms is:

*home = phase_A * / phase_B * index * / home_switch*

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Set up the Y axis so it will use the encoder signals to recognize the home position.

Enter: `AY HE`

HS HOME SWITCH

The HS command enables PC68 home switch mode to determine where home is when an HM or HR command is executed (default at power up or reset). This mode can also be used with encoders which contain internal home logic by connecting their output to the PC68 home input for the appropriate axis. The active level of this input may be controlled by the HH and HL commands.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Set up the Y axis so it will ignore the encoder signals and only use the home input to recognize the home position.

Enter: `AY HS`

6.20. ENCODER STATUS REQUEST COMMANDS

EA ENCODER STATUS

The EA command returns encoder status of the currently addressed axis in the following format:

EA COMMAND RESPONSE DESCRIPTION		
CHAR	**SENT**	**DESCRIPTION**
1	LF	Line feed
2	CR	Carriage return
3	CR	Carriage return
4	E	Slip detection enabled
	D	Slip detection disabled
5	E	Position maintenance enabled
	D	Position maintenance disabled
6	S	Slip or stall detected (reset by execution of EA command)
	N	No slip or stall detected
7	P	Position Maintenance within deadband
	N	Position not within deadband
8	H	Axis is home
	N	Axis is not home
9	N	Unused/reserved
10	LF	Line feed
11	CR	Carriage return
12	CR	Carriage return

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Examine the status of the Y axis encoder.

Enter: `AY EA`

Response: `<LF><CR><CR>EENPNN<LF><CR><CR>`

RE REQUEST ENCODER POSITION

The RE command returns current encoder position of the currently addressed axis in encoder counts. The ASCII string is surrounded by line feed and carriage return pairs.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Examine the current encoder position of the Y axis.

Enter: `AY RE`

Response: `<LF><CR>12345<LF><CR>`

PE REPORT ENCODER POSITIONS

PE reports the encoder positions of all encoder and/or servo axes.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Report the encoder positions of a PC68-45 (four axis servo board).

Enter: `PE`

Response: `<LF><CR>0,50,156,0<LF><CR>`

6.21. VELOCITY STAIRCASE COMMANDS

The following commands describe the velocity staircase mode. This mode is useful in applications requiring a change in velocity at a prescribed position without stopping.

MP MOVE POSITIVE

The MP command sets the direction logic to move in the positive direction.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	Not valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: (see MV command page 6-140)

MM MOVE MINUS

The MM command sets the direction logic to move in the negative direction.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	Not valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: Set the direction line to move in the minus direction on the Y axis.

Enter: `AY MM`

MV#,# MOVE VELOCITY

The MV command causes the motor to run to the new absolute position (parameter 1) at the new velocity (parameter 2). When the destination is reached control will be passed to the next command which should be another MV command or an SP command. If the command is not received in time the controller will continue to move at the specified velocity. Note that this is a slave mode and it is the responsibility of the user to provide the commands in time. They may be queued ahead of time. If a new MV command is sent after the controller has already passed the destination specified in the command, the controller will continue to move at the old velocity. Any number of steps can be specified in this manner with both acceleration and deceleration. The controller will not reverse direction if the position has already passed, but will behave as explained above. Thus the direction of the move must be specified before starting the move with the MP or MM commands. All destinations must be in absolute position, no position relative moves are allowed due to the nature of these commands. Cosine and parabolic acceleration will not apply.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	6	6	6
AA	Not valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: Generate a velocity staircase with the breakpoints given in absolute position. Default acceleration (AC) of 200,000

Enter:
```
MP
MV10000,30000
MV20000,50000
MV30000,10000
SP35000
```

The move as shown in Figure 6-1

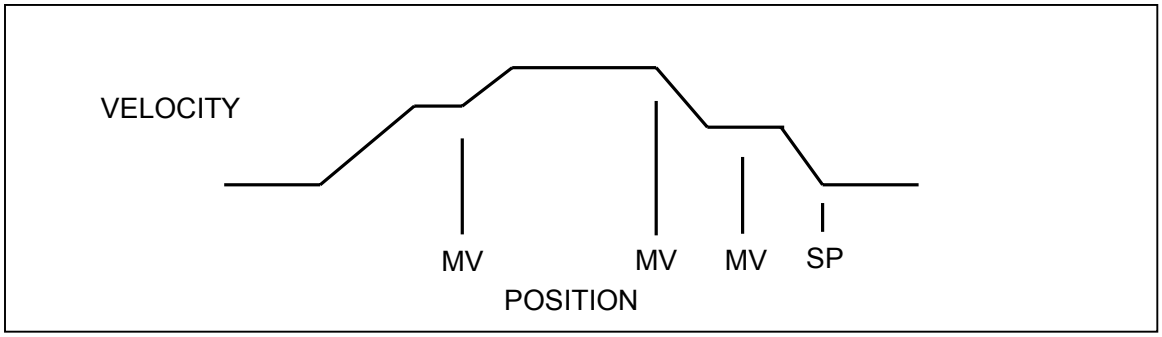

FIGURE 6-1 VELOCITY STAIRCASE PROFILE

SP# STOP AT POSITION

The SP command will cause the axis to stop at the specified position. The controller will attempt to stop at the specified destination. If there is insufficient distance to stop at the previously specified deceleration when the command is received, the controller will stop as soon as possible at that deceleration. This command is not compatible with the JG command.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	4	5	5
AA	Not valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: (see MV command page 6-140)

FP# FORCE POSITION

The FP command will flush the command queue and attempt to stop at the specified position. The axis will overshoot if there is insufficient distance left to stop at the programmed acceleration. This command should not be given to a servo axis while it is in motion. The results may be unpredictable.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	4	4	4
AA	Not valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: Force axis to stop at 25,000.

Enter: `FP25000`

6.22. CONSTANT VELOCITY CONTOURING

The PC68 will attempt to generate any profile which it is asked to do. It is the responsibility of the host to be sure the acceleration required when generating a circle or any other change in direction is possible within the mechanical constraints of the system. All corners must be defined by arcs and tangents to those arcs, else the change in direction will be instantaneous and generate very large accelerations. The arc radius must be chosen so that the acceleration constraints of the system are met.

AF#,# AUXILIARY OFF

The AF command may be used within a contour definition allowing control of other devices at any instruction within the contour. The AA mode syntax is used. Any auxiliary can be exercised with this command. All axes must be specified or specifically skipped, rather than those axes defined within the contour, as the other commands in this section.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	1	1	1
AM	1	1	1
AA/CD	2	2	2

Example: (see CD command page 6-145)

AN#,# AUXILIARY ON

The AN command may be used with a contour by using the AA mode syntax as above. Any auxiliary can be exercised with this command. All axes must be specified or specifically skipped, rather than those axes defined within the contour, as the other commands in this section.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	1	1	1
AM	1	1	1
AA/CD	2	2	2

Example: (see CD command page 6-145)

PA# POWER AUTOMATIC

The PA command will turn on or off the auxiliary outputs at the beginning of each GO or GD command execution and complement the outputs after the move is executed. The auxiliary will be turned on, i.e. pulled high, upon the execution of the GO or GD and off at the end of that move, if the parameter is zero or not specified in the single axis mode. If the parameter is non-zero, the sense is reversed, i.e. the auxiliary output is turned off (driven low) upon the execution of the GO or GD command and on at the end of the move.

The SE command can be used to apply a settling time at the end of each move before complementing the auxiliary bit. This is useful for systems that need to retain torque for some specific amount of time before allowing the motor drive to reduce current output.

This mode need only be set once and can be turned off by using the AN or AF command. Axes can be selectively affected in the AA mode by following the syntax as described for the AN command. The values of the included parameters set the state of the auxiliary line during the move. The following queue requirements apply to each GO or GD command in the command stream in the AA and single axis modes. This mode is off by factory default. See the AP Command on page 6-24 to preserve the PA settings as the Power up/Reset values.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	1	1	1
AA	1	1	1
AM	1	1	1
AA/CD	Not Valid		

Example: Turn on the Y axis auxiliary output at the beginning of a move and turn the T axis output off at the beginning of a move, while in the AA command mode. (Note the reversed logic; i.e. 0 = on, 1 = off. “On” pulls the signal line to ground. “Off” lets it rise to 5 volts or its pull-up reference voltage.)

Enter:

```
AA PA,0,,1;
```

Related commands: AN, AF

BL# BIT LOW

The BL command sets the selected general purpose output on (i.e. logic low).

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	3	3
AM	2	3	3
AA/CD	2	2	2

Example: (see the following BH command)

BH# BIT HIGH

The BH command sets the selected general purpose output off (i.e. logic high). The state of general purpose outputs is off at power up or reset.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	3	3
AM	2	3	3
AA/CD	2	2	2

Example: Set bit 10 high at the start of a contour and low at the end.

Enter:

```
AA CV2000
CD0,0;
BH10
CR0,10000,6.2831853;
BL10
CE
CK
```

CD#,#; CONTOUR DEFINE

The CD command enters contour definition mode. It allows entry of commands for contouring mode. Commands are queued for execution by the CX command. The parameters define the axes for which the contour is defined and the starting position of the contour in absolute units. The contour may be defined on up to 8 axes if circular interpolation is not used, or 2 axes with circular mixed with linear interpolation. Attempting to do circular interpolation in a contour which is being defined for more than 2 axes will be flagged as a command error. This command is executed in the AA mode. The contouring axes must be at positions which allow them to reach the specified contouring velocity by the specified position when the contour is executed. If the actual position of the stage is equal to the starting position as defined by the CD command, the stage will jump to the contouring velocity with no ramp up. This could cause the stage to stall if it is not able to accelerate at this high rate. It is recommended that some ramp up distance be allowed. There is also some ramp down distance as the stage slows from the constant velocity value to a stop. This distance is adjustable using the AC command. It can almost be eliminated using the CK command. The CX command cannot be placed within a loop or while construct.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Not valid		
AA	Immediate		
AM	Immediate		
AA/CD	5 + number of axes in the contour		

Example: The following demonstrates cutting a hole with a 10,000 count radius using constant velocity contouring and circular interpolation. The contouring velocity is set to 1000 pulses per second. A contour is then defined beginning at coordinates 0,0 on the X and Y axes. The auxiliary output of the Z axis is turned on, which could turn on the cutting torch or laser starting the cut at the center of the circle. A half circle is cut from the center to the outside of the hole, positioning the cutting tool at the start of the desired hole. The hole is then cut, the torch turned off, the stage stopped and the definition is complete. The stage is then positioned and the hole cut with the CX command. The AN and AF commands must have commas for all axes since they can all be addressed from within the contour definition.

Enter:

```
AA
CV1000 CD0,0;
AN,,0; CR0,5000,3.1415926;
CR0,0,6.2831853;
AF,0;MT 10,000,-1000;
CE
MT-1000,0; GO CX
```

CE CONTOUR END

The CE command marks the end of the contour sequence. It will terminate the CD mode, ramp to a stop and exit to the AA command mode when executed. The end of the contour should contain at least a short linear segment just prior to the CE command to initialize the parameters for the deceleration of the stage.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Not valid		
AA	Not Valid		
AM	Not Valid		
AA/CD	2	2	2

Example: (see CD command on page 6-145)

CK CONTOUR END AND KILL

The CK command will end the contour sequence, like the CE command, except there is no ramp down, i.e. the pulses will stop abruptly. This command should be used with caution to prevent the stage from missing steps or loosing its correct position. It is used in place of the CE command.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Not Valid		
AA	Not valid		
AM	Not Valid		
AA/CD	2	2	2

Example: Same scenario as CD command, but we want to end the contour with the minimum ramp down.

Enter:

```
AA
CV1000 CD0,0;
AN,,0; CR0,5000,3.1415926;
CR0,0,6.2831853;
AF,0; MT 10,000,-1000;
CK
MT-1000,0; GO CX
```

CR#,#,# CIRCULAR INTERPOLATION

The CR command defines a move in a circular pattern from the entry position. The first two parameters are the center of the circle in absolute units and the third parameter is the distance to move in radians. Positive radians equal counter clockwise movement. Negative radians equal clockwise movement. The distance parameter should be supplied to seven significant digits if a full circle is to be generated.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Not valid		
AA	Not Valid		
AM	Not Valid		
AA/CD	8	8	8

Example: (see CD command on page 6-145)

CV# CONTOUR VELOCITY

The CV command allows specification of contouring velocity. It is executed from the AA mode before a contour definition. A contour defined by a CD command cannot be executed if followed by a CV command. Changing this parameter will make any previously defined contours invalid. The contour velocity defaults to 1000 at power up or reset. Use WQ between contour definitions to avoid having a CV associated with a second contour definition affect a prior contour still in motion. A CV cannot be issued between a CD and CE command.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Not valid		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: (see CD command on page 6-145)

CX CONTOUR EXECUTE

The CX command will execute the previously entered contour sequence. The stage must be positioned such that it can accelerate to speed by the absolute position specified by the CD command it is executing and must be traveling in the proper direction. Once a contour is defined it may be executed at any time by executing a CX command until it is replaced by another contour definition. The CX command cannot be placed within a loop or while construct.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Not valid		
AA	18	38	68
AM	18	38	68
AA/CD	Immediate		

Example: (see CD command on page 6-145)

MT#,# MOVE TO

The MT command causes the axes defined by the CD command to move to the specified absolute position using linear interpolation. Only the axes being used in a contour must be specified in the contouring mode.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Not valid		
AA	4	28	60
AM	4	28	60
AA/CD	4 + number of axes		

Example: Make a hexagon in CV mode using the X and Y axes.

Enter:

```
AA CV5000;
CD10000,0;
MT20000,0
MT25000,10000
MT20000,20000
BL9
MT10000,20000
MT5000,10000
BH9
MT10000,0
CK
CX
```

RQ REQUEST QUEUE STATUS

The RQ command returns the number of entries available in the contouring queue.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Immediate		

Example: Examine contour queue size.

Enter: `AA CD0,0; RQ`

Response: `<LF><CR>1016<LF><CR>`

6.23. MACRO CONTROL COMMANDS

Macros are typically used as a shortcut to save some keystrokes. They can be used to save common parameter settings that may need to be recalled. They can also be used to store common command sequences that may be used for a particular process.

Once macros are defined, an entire command sequence can be sent to the controller through just executing the MX command and the appropriate macro number. As a result, sending a stream of frequently used commands to the controller is done simply through the use of one command. Detailed information regarding the macro commands is shown below.

MD# TEMPORARY MACRO DEFINE

MD is used to begin defining a temporary macro. A macro can contain up to 250 characters. Macros 0 through 4 are temporary and they will be erased when the controller is reset or power is turned off. Macros 5 through 24 are stored in non-volatile memory and will be preserved when the controller is reset or powered off.

Enter the macro number immediately after the MD command. The macro number must be between 0 and 4. Next enter the command string, which is made up of up to 250 ASCII characters. After entering the command string for the macro, enter a control Z to end the macro definition.

Be careful not to exceed 250 ASCII characters or the size of the axis queue when working with macros.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Define macro 2 to set velocities to 20000 on all axes of a two axis board.

Enter:
```
MD2
AA VL20000,20000;
^Z
```

MX# MACRO EXECUTE

The MX command will execute the specified macro and its command string. The macro number that is entered as the argument of the command must be between 0 and 24.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Execute macro number 6.

Enter: `MX6`

PM# PRINT MACRO

The PM command will print the specified macro's command string. The macro number entered as the argument for this command must be between 0 and 24.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Print the command string contained in macro 19.

Enter: `PM19`

PT#,# PRESERVE A TEMPORARY MACRO

Use PT to save a temporary macro permanently by copying it to non-volatile memory. The temporary macro number, which is entered as an argument for this command, must be between 0 and 4. The non-volatile macro number, which is also entered as an argument for this command, must be between 5 and 24.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Copy temporary macro 3 to non-volatile macro 19.

Enter: `PT3,19`

6.24. STAND-ALONE COMMANDS

The Stand-alone mode allows a PC68 Motion Controller to run in a completely independent operation mode. This mode has several commands that can establish links to macros. When set up properly in this mode the PC68 can scan for a predefined I/O Input bit, until it reaches the specified state (i.e. goes high.) Upon sensing that this condition has been met, it will execute the permanent Macro (5-24) from Nonvolatile flash memory that had been previously associated with this I/O bit and its condition.

A common application the stand-alone mode is to incorporate the KILL (KL) function. Reference the SK command. This will allow the user to stop motion of the device.

All of these selections are temporary. They can be made permanent by executing the AP command, which assigns the current parameter values as the Power Up defaults.

Note: The AP command should be used sparingly as it causes a write to the on board Flash Memory and there is a finite amount of times that it can be re-written to (i.e. less than 10,000 times, typical).

APPLICATION OVERVIEW:
The setup of the stand-alone mode is performed through the communication interface by the use of the commands. The user would define the required motion and processes and store them in a macro (see section 6.23. Macro Control Commands on page 6-150). Then, with the use of the commands below, the execution of the specific macros would be defined. Once all of the setup is completed the controller would be put in the stand-alone mode (SM) and the execution of the macros controlled by the defined input bits.

There are no queue requirements for these commands.

SM# **ENABLE / DISABLE STAND-ALONE MODE**

This command enables or disables the SM mode (Stand-Alone Mode).

SM mode value of 1 enables the **SM** mode

A **SM** mode value of 0 disables the **SM** mode.

Example: Enable Stand-Alone Mode

Enter: `SM1;`

SX#,#,#; DEFINE MACRO LINK ON PC68

This command creates the link to execute the previously defined permanent Macro when it senses the change in state of the defined standard input bit.

First Parameter
This specifies the standard input bit number. The factory default bits are 0,1,2,and 3, but it can be configured by the user to include bits 0 to 7.

(If an I/O expansion board is installed, bits 8-11 are also included.)

Second Parameter
Valid Bit States are 0 and 1

If the value of the selected bit state is ZERO, the selected macro will be executed if the selected bit changes from a TTL high to a TTL low.

If the value of <Bit State> is ONE, then the selected Macro will be executed when the selected bit changes from a TTL low to a TTL high.

NOTE: Each bit state can be linked with a macro. So, up to two macros can be assigned to an input bit. For example, macro 10 could be executed when I/O 0 goes low and macro 11 could be executed when I/O 0 goes high.

Third Parameter
Specifies the Macro number (5-24) to be executed when the conditions are met. A macro link can be deleted by specifying the bit number and the Bit State along with a Macro number zero. The macro assigned to a given input bit and Bit State can be changed by issuing the same SX command using a different macro number.

Example: When I/O bit 1 goes from low to high, macro 20 will be executed.

Enter: `SX1,1,20;`

Example: Upon Power-up, the user presses an “Activate Button-short to ground”, (linked to bit 1). This should cause the PC68 to position the X-axis at a position 1000 steps from zero and the Y-axis at 2000 steps from zero. At that point, it should execute Macro 10. The Stop button (short to ground) is linked to bit 2,

```
Enter: MD0;                |begin the definition of Macro #0
       AX;MA1000;GO        |move x-axis to position 1000
       AY;MA2000;GO        |move y-axis to position 2000
       <control Z>         |terminate the definition of Macro #0
       PT0,10;             |store macro #0 to non-volatile macro #10
       SX1,0,10;           |define I/O bit #1, active low will execute macro #10
       MD1;                |begin the definition of Macro #1
       KL                  |send a KILL to stop everything
       <control Z>         |terminate the definition of Macro #1
       PT1,9;              |store macro #1 to non-volatile macro #9
       SX2,0,9;            |define I/O bit #2, active low will execute macro #9
       SM1;                |enable the stand-alone mode
       AP;                 |set the current parameters to the power-up defaults
[this is the end of the command sequence]
```

SK#,#,#; DEFINE KILL LINK ON PC68

The SK command links the execution of the KILL function (KL) to the defined change of state of the assigned input bit.

First Parameter
This specifies the standard input bit number. The factory default bits are 0,1,2,and 3, but it can be configured by the user to include bits 0 to 7

(If an I/O expansion board is installed, bits 8-11 are also included.)

Second Parameter
Valid Bit States are 0 and 1

If the value of the selected bit state is ZERO, the selected macro will be executed if the selected bit changes from a TTL high to a TTL low.

If the value of <Bit State> is ONE, then the selected Macro will be executed when the selected bit changes from a TTL low to a TTL high.

NOTE: Each bit state can be linked with a macro. So, up to two macros can be assigned to an input bit. For example, macro 10 could be executed when I/O 0 goes low and macro 11 could be executed when I/O 0 goes high.

Third Parameter
If the value is ZERO the KILL function linkage for the specified Bit State is deleted.

If the value is ONE, the KILL function linkage for the specified Bit State will replace the current linkage

Example: The current macro linked to I/O bit 2 as it goes from high to low is to be replace with the KILL (KL) function.

Enter: `SK2,0,1;`

XX#, #, # DEFINE MACRO LINK ON Px6IO

The XX command links the execution of a permanent macro to a change of state of an input bit on the Expansion I/O board (Px6IO). The macro will be executed if the controller is in the Stand Alone (SM) mode and the selected bit changes to the specified state

First Parameter
This specifies the input bit number of the Px6IO. Valid numbers must coincide with the input bit numbers assigned to the Px6IO. If the bit is assigned by another Macro a “command error” is reported.

Second Parameter
Valid Bit States are 0 and 1

If the value of the selected bit state is ZERO, the selected macro will be executed if the selected bit changes from a TTL high to a TTL low.

If the Bit State is ONE, then the selected Macro will be executed when the selected bit changes from a TTL low to a TTL high.

NOTE: Each bit and state can be linked with a macro. So, up to two macros can be assigned to an input bit. For example, macro 15 could be executed when I/O 20 goes low, and macro 16 could be executed when I/O 20 goes high.

Third Parameter
Specifies the macro number (5-24) to be executed when the conditions are met. The macro assigned to an existing bit and state can be changed by entering the same XX command with a new macro number.

A Macro linkage can be deleted by specifying the bit number and the Bit State along with macro number Zero.

Example: When input bit 20 goes high macro 15 will execute.

Enter: `XX20,1,15;`

XK#,#,# DEFINE KILL LINK ON Px6IO

The XK function links the execution of the KILL (KL) function to the change in the sate of a Px6IO input bit in a similar manner as the XX function links the execution for a selected macro as above.

First Parameter
This specifies the input bit number of the Px6IO. Valid numbers must coincide with the input bit numbers assigned to the Px6IO. If the bit is assigned by another Macro a “command error” is reported.

Second Parameter
Valid Bit States are 0 and 1

If the value of the selected bit state is ZERO, the selected macro will be executed if the selected bit changes from a TTL high to a TTL low.

If the Bit State is ONE, then the selected Macro will be executed when the selected bit changes from a TTL low to a TTL high.

NOTE: Each bit and state can be linked with a macro. So, up to two macros can be assigned to an input bit. For example, macro 15 could be executed when I/O 20 goes low, and macro 16 could be executed when I/O 20 goes high.

Third Parameter
If the value is ZERO the KILL function linkage for the specified Bit State is deleted.

If the value is ONE, the KILL function linkage for the specified Bit State will replace the current linkage.

Example: When input bit 17 goes low, the KILL (KL) function linkage will be deleted.

Enter: `XK17,0,1;`

PS#,# REPORT MACRO LINK

This command reports the macro link or KILL (KL) function link to the specified input bit and Bit State.

See XX command on page 6-157

First Parameter
This specifies the standard input bit number. The factory default bits are 0,1,2,and 3, but it can be configured by the user to be the I/O bits 0 to 7

(If an expansion board is installed, bits 8-11 are also included.)

Second Parameter
Valid Bit states are 0 and 1

If the value of the selected bit state is ZERO, the selected macro will be executed if the selected bit changes from a TTL high to a TTL low.

If the Bit State is ONE, then the selected Macro will be executed when the selected bit changes from a TTL low to a TTL high.

NOTE: Each bit and state can be linked with a macro. So, up to two macros can be assigned to an input bit. For example, macro 15 could be executed when I/O 20 goes low, and macro 16 could be executed when I/O 20 goes high.

Output Formats:
If the bit is linked to the execution of a macro, the text response is the text of the SX command used to link the bit with the macro

Example: `<LF><CR>SX1,0,23;<LF><CR>`

If the bit is linked to the KL (KILL) function, the output will be the text response of the SK command used to link the bit with the kill function

Example: `<LF><CR>SK1,0,1;<LF><CR>`

If the bit is not linked the output will be: `<LF><CR><LF><CR>`

Example: This will report if there are any macro or KILL (KL) function links to input bit 2 when it goes from low to high.

Enter: `PS2,1;`

PX#, # REPORT KILL LINK

This command reports the macro link or KILL (KL) function link to the specified input bit and Bit State.

See XX command on page 6-157

First Parameter
Specifies the number of the input bit on the Px6IO. Valid numbers include any defined input bits on the Px6IO.

Second Parameter
Valid Bit States are 0 and1

If the value of the selected bit state is ZERO, the selected macro will be executed if the selected bit changes from a TTL high to a TTL low.

If the Bit State is ONE, then the selected Macro will be executed when the selected bit changes from a TTL low to a TTL high.

NOTE: Each bit and state can be linked with a macro. So, up to two macros can be assigned to an input bit. For example, macro 15 could be executed when I/O 20 goes low, and macro 16 could be executed when I/O 20 goes high.

Output Formats:
If the bit is linked to the execution of a macro, the text response is the text of the XX command used to link the bit with the Macro.

Example: `<LF><CR>XX1,0,23;<LF><CR>`

If the bit is linked to the KILL function (KL), the output will be the text response of the XK command used to link the bit with the kill function

Example: `<LF><CR>SK1,0,1;<LF><CR>`

If the bit is not linked the output will be: `<LF><CR><LF><CR>`

Example: This will report if there are any macro or KILL function links to input bit 21 when it goes from high to low

Enter: `PX21,0;`

CB CLEAR MACRO LINKS

This command clears all macro links of input bits to macro executions and/or the KILL (KL) function.

Example: Clear all previously defined macro links to input bits.

Enter: `CB;`

6.25. EXPANSION BOARD COMMANDS

Commands for the Px6IO.

6.25.1. Px6IO DIGITAL INPUT/OUTPUT MODULE

Port relative I/O commands:

PC# **REPORT I/O PORT CONFIGURATION**

The PC command reports the configuration of a Px6IO input/output port.

If Px6IO 0 board is installed valid port numbers include 0 through 5
If Px6IO 1 board is installed valid port numbers include 6 through 11
If Px6IO 2 board is installed valid port numbers include 12 through 17

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not valid		

Example: Report the configuration of port 1

Enter: `PC1;`

Result:

If port 1 is an input port	`<LF><CR>I<LF><CR>`
If port 1 is an output port	`<LF><CR>O<LF><CR>`
If port 1 is not installed	`<LF><CR>N<LF><CR>`

PI# REPORT I/O PORT BIT STATES

The PI command reports the bit states of a Px6IO port, as a hex number.

If Px6IO 0 board is installed valid port numbers include 0 through 5
If Px6IO 1 board is installed valid port numbers include 6 through 11
If Px6IO 2 board is installed valid port numbers include 12 through 17

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Report the bit states of port 1

Enter: `PI1;`

Result: If bit 0 of port 1 is high `<LF><CR>01<LF><CR>`
If port 1 is not installed `<LF><CR>NP<LF><CR>`

PO#,# SET THE BITS OF AN OUTPUT PORT

The PO command sets the bits of a Px6IO output port, to the states specified by a hex number.

If Px6IO 0 board is installed valid port numbers include 0 through 5
If Px6IO 1 board is installed valid port numbers include 6 through 11
If Px6IO 2 board is installed valid port numbers include 12 through 17

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	3	3	3
AA	3	4	4
AM	3	4	4
AA/CD	Not Valid		

Example: Set the bit states of port 1 so that bits 0 and 4 are 1 and all others are 0.

Enter: `PO1,11;`

6.25.2. BIT RELATIVE I/O COMMANDS

XC# REPORT I/O BIT CONFIGURATION

The XC command reports the configuration of a PPx6IO input/output bit.

If Px6IO 0 board is installed valid bit numbers include 0 through 47
If Px6IO 1 board is installed valid bit numbers include 48 through 95
If Px6IO 2 board is installed valid bit numbers include 96 through 143

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Report the configuration of bit 65.

Enter: `XC65;`

Response:

If bit 65 is an input bit	`<LF><CR>I<LF><CR>`
If bit 65 is an output bit	`<LF><CR>O<LF><CR>`
If bit 65 is not installed	`<LF><CR>N<LF><CR>`

XI# REPORT THE BIT STATE OF A PX6IO BIT

The XI command reports the bit state of a Px6IO bit.

If Px6IO 0 board is installed valid bit numbers include 0 through 47
If Px6IO 1 board is installed valid bit numbers include 48 through 95
If Px6IO 2 board is installed valid port numbers include 96 through 143

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Report the configuration of bit 65.

Enter: `XC65;`

Response:
If bit 65 is an input bit `<LF><CR>I<LF><CR>`
If bit 65 is an output bit `<LF><CR>O<LF><CR>`
If bit 65 is not installed `<LF><CR>N<LF><CR>`

XH# SET A PX6IO BIT STATE

The XH command sets a Px6IO output bit to a 1.

If Px6IO 0 board is installed valid bit numbers include 0 through 47
If Px6IO 1 board is installed valid bit numbers include 48 through 95
If Px6IO 2 board is installed valid port numbers include 96 through 143

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	3	3
AM	2	3	3
AA/CD	Not Valid		

Example: Set Px6IO bit 65 high.

Enter: `XH65;`

XL# SET A PX6IO BIT LOW

The XL command sets a Px6IO output bit to a zero.

If Px6IO 0 board is installed valid bit numbers include 0 through 47
If Px6IO 1 board is installed valid bit numbers include 48 through 95
If Px6IO 2 board is installed valid bit numbers include 96 through 143

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	2	2	2
AA	2	3	3
AM	2	3	3
AA/CD	Not Valid		

Example: Set Px6IO bit 65 high.

Enter: `XL65;`

6.26. SERIAL COMMUNICATION CONFIGURATION COMMANDS

SB# SET BAUD RATE

The SB command sets the controller's serial communications baud rate. Valid baud rates include 300, 600, 1200, 2400, 9600, 19200 and 38400. The factory default baud rate is 9600. See the AP Command on page 6-24 to preserve the SB your baud rate as the Power up/Reset rate.

NOTE: The PC68 uses CTS/DTR hardware hand shake for flow control.

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Immediate		
AM	Immediate		
AA/CD	Not Valid		

Example: Sets the serial communications baud rate to 19200.

Enter: `SB19200;`

?SB QUERY THE BAUD RATE

The ?SB command queries the board to determine the current baud rate setting

QUEUE REQUIREMENTS			
MODE	**Min (pf)**	**Max (pn/cn)**	**Custom ramp**
AX - AS	Immediate		
AA	Not Valid		
AM	Not Valid		
AA/CD	Not Valid		

Example: Query the board to determine its current baud rate setting.

Enter: `?SB`

Response: If the board is set for 9600 baud rate: `<LF><CR>9600<LF><CR>`

7. HOST SOFTWARE

7.1. INTRODUCTION TO PC68 SOFTWARE SUPPORT

A disk containing device drivers, application software, and demonstration code for Oregon Micro Systems PC family controller is supplied with the initial purchase of an OMS PC68 board. Reference the text files (i.e. README.TXT) on the disk for installation instructions and other information.

Some programs on the demo disk that include source code may be adapted for use in application programs that use OMS motion controls. No license is required.

7.2. COMMUNICATION METHODS

There are two possible ports that may be used for communication on a PC68. They are a RS-232 port and a PC/104 bus interface. (Note that not all models of the PC68 support the PC/104 bus interface.)

7.2.1. PC/104 MODE

7.2.1.1. Interrupt Driven Communication (Preferred Method)

This method provides the most efficient means of communication and status gathering with regard to CPU cycles spent. When the PC68 has a character to give to the host, it will generate an interrupt to which the host responds and collects the character. Interrupts will also be generated when the PC68 is ready to accept a character from the host. Further interrupts can be generated for status events such as encountering a limit switch or an invalid host command.

In interrupt-based communications, the host spends as little time as possible talking to the PC68 which frees up its time for other purposes. Instead of constantly polling the PC68 for status information, an interrupt can trigger a function that collects status just once.

Although this method can be more difficult to code and requires a programming language capable of compiling interrupt-driven code, it provides the greatest degree of efficiency. This is also the preferred method recommended by Oregon Micro Systems.

7.2.1.2. Polled-Mode Communication (Not Recommended)

Polled-mode communication can be easier to write code for than interrupt-driven communication. However, this method places a much higher load on the host CPU than interrupt-driven methods and leaves less time for other processes. Using polled mode also makes the code more temperamental should the timing specifications change in the Host PC that the code is running on, for example.

Instead of generating an interrupt, when the PC68 has a character to give to the host, a flag is set in the status register of the PC68. It is the responsibility of the host software to periodically check the status register and, if the flag is set, collect the character from the PC68. Similarly, if the host wishes to send a character to the PC68, the host must first check the status register for the appropriate flag. If the flag is present, the host can go ahead and send the character. If the flag is not present, the host must wait until it becomes set. Status information must be gathered in a similar fashion.

7.2.2. SERIAL (RS-232) COMMUNICATION

7.2.2.1. Interrupt Driven (Preferred Method)

Most serial port software and drivers use interrupts although it is possible to operate in polled-mode. Serial communication, when interrupt driven, is much faster than in polled-mode and, at 9600 baud and higher, are the only possible method; the host simply cannot keep up with the port at high rates and still pay attention to other tasks.

There are a couple of different options available for communication methods when using RS-232. The first and most difficult is to write the serial port driver directly into the code. This method, while the most efficient, requires the programmer to write an Interrupt Service Routine and deal with interrupt hooking and unhooking. Second, access the serial port as a character device, i.e. COM1: or via an operating system API, opening it as though it were a file. This method is far simpler and allows higher-rate communications but puts more overhead on the host software, slowing down response times for status events from the PC68.

7.2.2.2. POLLED-MODE (Not Recommended)

Polled-mode communication can be done in virtually any environment and programming language. However, polled-mode communication requires enough attention from the host CPU that high-rate character transfers are generally not possible and there is little time left over for performing other tasks. This is usually not an issue in non-multitasking environments but systems like Windows NT do not deal well with attention-hogs like polled-mode, high-speed serial communications.

In polled mode, the serial port does not use an interrupt. This means that the interrupt normally assigned to the serial port is available to other devices and that a large number of serial ports can be supported without running out of resources. This is an advantage in systems that require several serial devices.

For PC/104-capable PC68 controls, the best method by far is interrupt-driven bus communication. For RS-232 mode, standard interrupt-driver ports are best. As with any other system, non-multitasking environments allow the software to run with the full attention of the CPU which provides a better response time and causes fewer race conditions. If a multitasking system must be used, the system should be chosen with a heavy bias toward those that multitask most smoothly, spreading the CPU attention span evenly across all running tasks rather than toward those that allow the foreground task to take over.

There are several example applications on the support disk shipped with your PC68. Reference the source code and “README” files on the support disk.

This page intentionally left blank

8. SERVICE

8.1. USER SERVICE

The PC68 family of controllers contain no user serviceable parts.

8.2. THEORY OF OPERATION

The 68332 microprocessor on the PC68 controllers maintains four concurrent processes. The highest priority process calculates the desired pulse frequency 2048 times/second or 1024 times/second (depending on the model) with a proprietary algorithm (patent number 4,734,847). This frequency is used to generate the pulse trains. The velocity profile and synchronization of each axis is also handled by the 68332.

The commands from the PC/AT or compatible host computer are temporarily stored in a 124 character buffer until the 68332 microprocessor can parse them. The command is then executed immediately or routed to separate command queues for each axis. The command queue contains a list of addresses to execute followed by an optional parameter. A command from the host may be expanded into several commands to the appropriate axis. The GO command, for example, will expand into start, ramp up, constant velocity and ramp down commands. The LS command will save its parameter, i.e. the loop count, on a loop stack along with the address of the LS command to be used by the next LE command as a target for a jump command. The LE command will decrement the loop count and jump to the most recent LS command providing the loop count has not reached zero. If the loop count has reached zero and it is not nested inside another loop, the queue space will be flagged as available and the next instruction in the queue will be executed.

Interrupts to the PC/AT host are generated. Status of the interrupts and error flags may be read by the host. The PC68 address is compared to the I/O address selected by the host. When a match is detected the board decode logic is enabled.

This page intentionally left blank

APPENDIX A.
LIMITED WARRANTY

The Seller warrants that the articles furnished are free from defect in material and workmanship and perform to applicable, published Oregon Micro Systems, Inc. specifications for one year from date of shipment. This warranty is in lieu of any other warranty express or implied. In no event will Seller be liable for incidental or consequential damages as a result of an alleged breach of the warranty. The liability of Seller hereunder shall be limited to replacing or repairing, at its option, any defective units which are returned f.o.b. Seller's plant. Equipment or parts which have been subject to abuse, misuse, accident, alteration, neglect or unauthorized repair are not covered by warranty. Seller shall have the right of final determination as to the existence and cause of defect. As to items repaired or replaced, the warranty shall continue in effect for the remainder of the warranty period, or for 90 days following date of shipment by Seller of the repaired or replaced part whichever period is longer. No liability is assumed for expendable items such as lamps and fuses. No warranty is made with respect to custom equipment or products produced to Buyer's specifications except as specifically stated in writing by Seller and contained in the contract.

This page intentionally left blank

APPENDIX B.

TECHNICAL SUPPORT

Oregon Micro Systems , Inc. can be reached for technical support by any of the following methods:

1. Internet E-mail:	support@OMSmotion.com
2. World Wide Web:	www.OMSmotion.com
3. Telephone:	8:00 a.m. - 5:00 p.m. Pacific Standard Time (503) 629-8081 or (800) 707-8111
4. Facsimile:	24 Hours (503) 629-0688 or (877) 629-0688
5. USPS:	Oregon Micro Systems Inc 1800 NW 169th Place Suite C100 Beaverton OR 97006

RETURN FOR REPAIRS

1. Call Oregon Micro Systems Customer Service at 503-629-8081 or (800) 707-8111 or E-mail to sales@OMSmotion.com .
2. Explain the problem and we may be able to solve it on the phone. If not, we will give you a Return Materials Authorization (RMA) number.

   Mark the RMA number on the shipping label, packing slip and other paper work accompanying the return. We cannot accept returns without an RMA number.
3. Please be sure to enclose a packing slip with the RMA number, serial number of the equipment, reason for return, and the name and telephone number of the person we should contact if we have further questions.
4. Pack the equipment in a solid cardboard box secured with packing material.
5. Ship prepaid and insured to:

   OREGON MICRO SYSTEMS, INC.
   Twin Oaks Business Center
   1800 NW 169th Place, Suite C100
   Beaverton, OR 97006

This page intentionally left blank

APPENDIX C.
SPECIFICATIONS

Velocity
0 to 1,044,000 counts per second simultaneous on each axis

Acceleration
0 to 8,000,000 counts per second per second

Position range
67,000,000 counts (±33,500,000)

Accuracy
Position accuracy and repeatability ±0 counts for point to point moves
Velocity accuracy ±0.01% for step pulse output

Environmental
Operating temperature range: 0 to 50 degrees centigrade
Storage temperature range: -20 to 85 degrees centigrade
Humidity: 0 to 90% non-condensing

Power
+5VDC at 1 amp typical
+12VDC at 0.1 amp typical
-12VDC at 0.1 amp typical

Dimensions
3.550X3.775X0.5 inches high

Communication
PC/104: Meets all signal specifications for PC/104 ISA bus specifications (IEEE P996.1)
RS/232: Baud rates of 300 to 38.4K

Update Rate
488 μs for PC68 models with ≤4 axes
977 μs for PC68 models with >4 axes

PC/104 bus I/O address
The I/O address block utilizes 4 consecutive addresses and is user selectable. The factory default is 300-303 hex.

Limit switch inputs
TTL input levels with on board 2.2K pull up resistor, requires only external switch closure to ground or TTL level input signal. Input sense (low or high true) selectable by command input for each axis.

Home switch inputs
TTL input levels with on board 2.2K pull up resistor, requires only external switch closure to ground or TTL level input signal. Input sense (low or high true) selectable by command input for each axis.

User definable I/O
Up to12 bits of user definable I/O on PC68 models with 4 axes or less. 8 bits are user configurable as inputs or outputs. One auxiliary output per axis and these are fixed as outputs. Factory default is 4 inputs, 4 outputs and 1 auxiliary output per axis. (For PC68 models with more than 4 axes, up to twice this number of I/O is available. The default configuration of this I/O is the same as described above.)
TTL input levels with on board 2.2K pull up resistor, requires only external switch closure to ground or TTL level input signal.
The auxiliary outputs are TTL open collector outputs (7406, max 48mA). The other outputs are TTL totem pole outputs (74LS243, max 24mA).

Analog outputs
+/-10V and 0 to +10V

Step pulse output
Pulse width 50% duty cycle. Open collector TTL level signal (7406, max 48mA).

Direction output
Open collector TTL level signal (7406, max 48mA).

Encoder Feedback
Maximum 4 MHz after 4x quadrature detection
Differential TTL level signal MC26G32, max 150mA

PC/104 interrupt
Interrupts are user selectable, 2 through 7. The factory default is level 5.

OMS PC68 Intelligent Motion Controls						
MODEL	INTERFACE		SERVO AXES	STEPPER AXES		USER I/O
	PC/104	RS-232		CONTROL	FEEDBACK	
PC68-23		✓	2			10
PC68-25	✓	✓	2			10
PC68-33		✓	3			11
PC68-35	✓	✓	3			11
PC68-43		✓	4			12
PC68-45	✓	✓	4			12
PC68-47	✓	✓	2	2		12
PC68-21		✓		2		10
PC68-22	✓	✓		2		10
PC68-26	✓	✓			2	10
PC68-41		✓		4		12
PC68-42	✓	✓		4		12
PC68-1800		✓	8			24
PC68-2800	✓	✓	8			24
PC68-1602		✓	6	2		24
PC68-2602	✓	✓	6	2		24
PC68-1404		✓	4	4		24
PC68-2404	✓	✓	4	4		24
PC68-2206	✓	✓	2	6		24
PC68-1008		✓		8		24
PC68-2008	✓	✓		8		24

PC68 CONNECTOR (J5)			
Description	**Pin#**	**Pin#**	**Description**
Digital Ground	1	35	+5VDC
I/O-1	2	36	I/O-0
I/O-3	3	37	I/O-2
I/O-5	4	38	I/O-4
I/O-7	5	39	I/O-6
Digital Ground	6	40	+5VDC
X Index +	7	41	X Servo
X Index -	8	42	X Step
X Phase A +	9	43	X Auxiliary
X Phase A -	10	44	X Direction
X Phase B +	11	45	X Limit +
X Phase B -	12	46	X Limit -
Y Servo	13	47	X Home
Y Index +	14	48	Y Step
Y Index -	15	49	Y Auxiliary
Y Phase A +	16	50	Y Direction
Y Phase A -	17	51	Y Limit +
Y Phase B +	18	52	Y Limit -
Y Phase B -	19	53	Y Home
Analog Ground	20	54	+5VDC
Z Index +	21	55	Z Servo
Z Index -	22	56	Z Step
Z Phase A +	23	57	Z Auxiliary
Z Phase A -	24	58	Z Direction
Z Phase B +	25	59	Z Limit +
Z Phase B -	26	60	Z Limit -
T Servo	27	61	Z Home
T Index +	28	62	T Step
T Index -	29	63	T Auxiliary
T Phase A +	30	64	T Direction
T Phase A -	31	65	T Limit +
T Phase B +	32	66	T Limit -
T Phase B -	33	67	T Home
Digital Ground	34	68	+5VDC

PC68 AXIS EXPANSION BOARD			
Description	**Pin#**	**Pin#**	**Description**
Digital Ground	1	35	+5VDC
I/O-9	2	36	I/O-8
I/O-11	3	37	I/O-10
I/O-13	4	38	I/O-12
I/O-15	5	39	I/O-14
Digital Ground	6	40	+5VDC
U Index +	7	41	U Servo
U Index -	8	42	U Step
U Phase A +	9	43	U Auxiliary
U Phase A -	10	44	U Direction
U Phase B +	11	45	U Limit +
U Phase B -	12	46	U Limit -
V Servo	13	47	U Home
V Index +	14	48	V Step
V Index -	15	49	V Auxiliary
V Phase A +	16	50	V Direction
V Phase A -	17	51	V Limit +
V Phase B +	18	52	V Limit -
V Phase B -	19	53	V Home
Analog Ground	20	54	+5VDC
R Index +	21	55	R Servo
R Index -	22	56	R Step
R Phase A +	23	57	R Auxiliary
R Phase A -	24	58	R Direction
R Phase B +	25	59	R Limit +
R Phase B -	26	60	R Limit -
S Servo	27	61	R Home
S Index +	28	62	S Step
S Index -	29	63	S Auxiliary
S Phase A +	30	64	S Direction
S Phase A -	31	65	S Limit +
S Phase B +	32	66	S Limit -
S Phase B -	33	67	S Home
Digital Ground	34	68	+5VDC

RS-232 CONNECTOR (J4)			
Description	**Pin**	**Pin**	**Description**
Ground	5	9	No Connect
DTR	4	8	CTS
TxD	3	7	No Connect
RxD	2	6	No Connect
No Connect	1		

POWER SUPPLY CONNECTOR (J3)			
Description	**Pin**	**Pin**	**Description**
+5VDC	5	6	Digital Ground
-12VDC	3	4	No Connect
+12VDC	1	2	Analog Ground

Px6IO CONNECTOR			
Description	**Pin**	**Pin**	**Description**
IO32	1	35	IO2
IO7	2	36	IO6
IO22	3	37	IO8
IO37	4	38	IO23
IO39	5	39	IO38
IO36	6	40	IO5
Field VCC	7	41	+5V
Field Ground	8	42	Digital Ground
IO0	9	43	+5V
IO1	10	44	Digital Ground
IO17	11	45	+5V
IO18	12	46	IO19
IO21	13	47	IO31
IO34	14	48	IO3
IO35	15	49	IO33
IO4	16	50	Digital Ground
Field VCC	17	51	+5V
Field Ground	18	52	Digital Ground
IO20	19	53	+5V
IO40	20	54	IO9
IO42	21	55	IO24
IO43	22	56	IO10
IO11	23	57	IO41
IO12	24	58	IO25
IO26	25	59	Digital Ground
Field VCC	26	60	+5V
Field Ground	27	61	Digital Ground
IO44	28	62	+5V
IO45	29	63	Digital Ground
IO14	30	64	IO27
IO15	31	65	IO16
IO28	32	66	IO30
IO29	33	67	IO46
IO47	34	68	IO13

IO68-M TERMINAL BLOCK PIN-OUT					
Row 1	**Description**	**Row 2**	**Description**	**Row 3**	**Description**
1	X Step	24	X Direction	47	X Auxiliary
2	X Phase A+	25	X Phase B+	48	X Index +
3	X Phase A-	26	X Phase B-	49	X Index -
4	X Limit +	27	X Limit-	50	X Home
5	+5VDC	28	X Servo	51	Digital Ground
6	Y Step	29	Y Direction	52	Y Auxiliary
7	Y Phase A+	30	Y Phase B+	53	Y Index +
8	Y Phase A-	31	Y Phase B-	54	Y Index -
9	Y Limit +	32	Y Limit-	55	Y Home
10	+5VDC	33	Y Servo	56	Digital Ground
11	I/O-0	34	I/O-3	57	I/O-5
12	I/O-1	35	No Connect	58	I/O-6
13	I/O-2	36	I/O-4	59	I/O-7
14	+5VDC	37	Z Servo	60	Analog Ground
15	Z Step	38	Z Direction	61	Z Auxiliary
16	Z Phase A+	39	Z Phase B+	62	Z Index +
17	Z Phase A-	40	Z Phase B-	63	Z Index -
18	Z Limit +	41	Z Limit-	64	Z Home
19	+5VDC	42	T Servo	65	T Auxiliary
20	T Step	43	T Direction	66	T Index +
21	T Phase A+	44	T Phase B+	67	T Index -
22	T Phase A-	45	T Phase B-	68	T Home
23	T Limit +	46	T Limit-	69	Digital Ground

IO68-I TERMINAL BLOCK PIN-OUT					
Row 1	**Description**	**Row 2**	**Description**	**Row 3**	**Description**
1	GND	24	GND	47	+5VDC
2	I/O-0	25	I/O-17	48	+5VDC
3	I/O-1	26	I/O-18	49	FieldGND
4	+5VDC	27	I/O-19	50	I/O-31
5	I/O-2	28	+5VDC	51	I/O-32
6	I/O-3	29	GND	52	I/O-33
7	I/O-4	30	FieldGND	53	I/O-34
8	+5VDC	31	I/O-20	54	I/O-35
9	+5VDC	32	GND	55	+5VDC
10	I/O-5	33	I/O-21	56	I/O-36
11	I/O-6	34	I/O-22	57	I/O-37
12	I/O-7	35	No Connect	58	I/O-38
13	I/O-8	36	I/O-23	59	I/O-39
14	I/O-9	37	I/O-24	60	I/O-40
15	I/O-10	38	I/O-25	61	I/O-41
16	I/O-11	39	I/O-26	62	I/O-42
17	I/O-12	40	+5VDC	63	I/O-43
18	GND	41	+5VDC	64	GND
19	I/O-13	42	Field GND	65	GND
20	+5VDC	43	I/O-27	66	I/O-44
21	I/O-14	44	I/O-28	67	I/O-45
22	I/O-15	45	I/O-29	68	I/O-46
23	I/O-16	46	I/O-30	69	I/O-47

This page intentionally left blank

INDEX

!

?

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X