USER'S MANUAL

INTELLIGENT MOTOR CONTROLLERS

PC48 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.

3301-0400000
Revision D

TABLE OF CONTENTS

This page intentionally left blank

1. GENERAL DESCRIPTION

1.1. INTRODUCTION

The PC48 family of intelligent motion controls can manage as many as 8 axes of motion in one I/O slot of a PC/AT or compatible computer. They meet the IBM I/O channel specifications and can be plugged directly into the backplane of these machines. The PC48-4E can, for example, simultaneously control four axes of step motors while monitoring their actual position with the built in incremental encoder interface. It can manage coordinated or independent motion on each of the four axes simultaneously.

The PC48 functions as a motion coprocessor within the PC/AT or compatible computer. It utilizes a 68332 microprocessor and patented proprietary technology to control direction of motion, acceleration, deceleration and velocity of an associated motor. In response to commands from the host computer, the PC48 controller will calculate the optimum velocity profile to reach the desired destination in the minimum time while conforming to the programmed velocity and acceleration parameters.

The PC48 family of controllers use microstepping techniques for increased position resolution and decreased low speed resonance. When combined with the appropriate driver and step motor, the PC48 can divide the normal step angle into 250 discrete steps of 0.0072 degrees each or 50,000 steps per revolution. The Oregon Micro Systems drivers are capable of microstepping and can be controlled by the PC48 family of controllers.

Commands may be sent to the PC48 by simple I/O commands using virtually any language on the PC which has the ability to do I/O. It is easily programmed using ASCII character strings. For a typical motion requirement of 1,000,000 pulses at 400,000 pulses/sec and an acceleration of 500,000 pulses/sec^2 the following string would be sent from the host computer to the PC48:

VL400000 AC500000 MR1000000 GO

For additional programming examples see Section 6

1.2. FUNCTIONAL DESCRIPTION

The PC48, in response to commands from the host computer, provides controlled acceleration to a predefined peak speed followed by a constant velocity and controlled deceleration to a stop. This is achieved by calculating the optimum velocity 2048 times each second, providing a very smooth acceleration curve. This calculation is used to control a variable frequency pulse train which is derived from a crystal oscillator providing very accurate pulse rates.

The 68332 microprocessor calculates this velocity profile for each of the axes providing independent but synchronized (if desired) profiles for each axis. The PC48 can perform a smooth coordinated move on up to eight axes using linear, parabolic or cosine velocity profiles. It can manage as many as eight independent or coordinated processes.

The PC48 will calculate the optimum velocity profile to generate the desired move, while conforming to the acceleration and velocity data input by the host computer. This move will consist of a smooth acceleration, followed by a constant velocity section and a smooth deceleration to the desired distance. A graph of a typical linear velocity is shown in Figure 1-1.

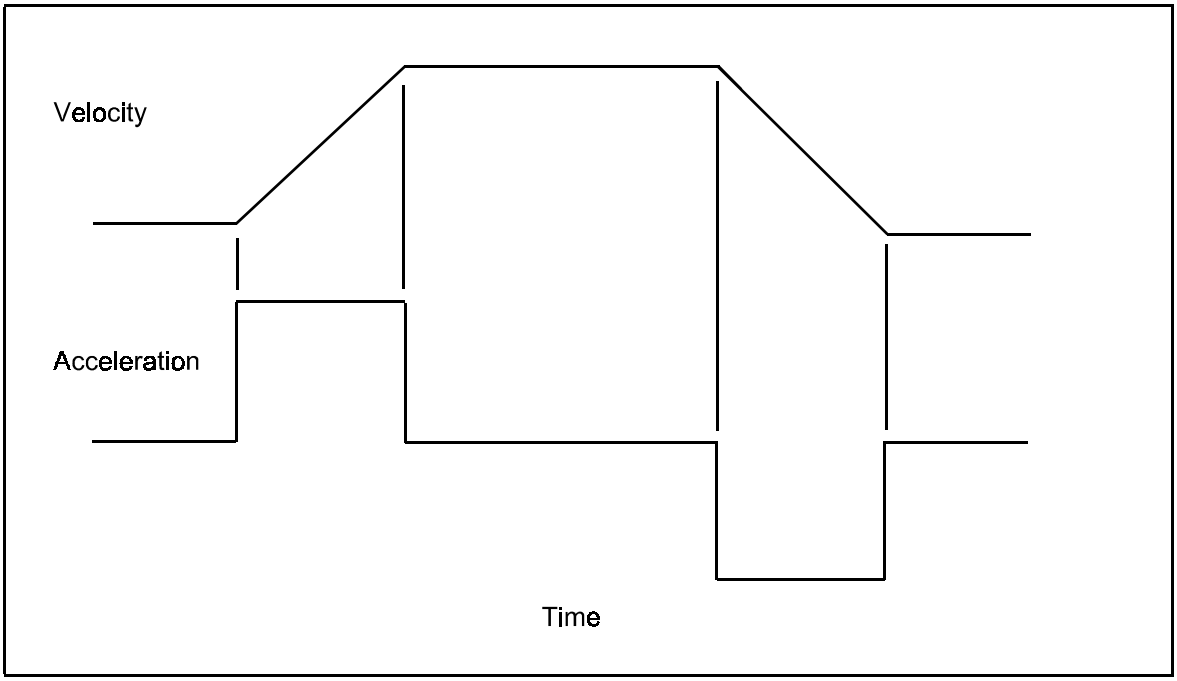


FIGURE 1-1 TYPICAL VELOCITY PROFILE

If the move parameters do not allow the motor to accelerate to the desired velocity in the desired distance, the PC48 will automatically generate an optimum triangular velocity profile. It can also be commanded to accelerate to a velocity and hold that velocity until told to stop or change to a new velocity. It will then smoothly decelerate to a stop or accelerate/decelerate to the new velocity.

Several moves of this type may be chained together to provide a more complex pattern. The PC48 is able to store up to 124 characters in an input character buffer, plus a separate 200 command and parameter queue for each axis, allowing several moves to be made without host intervention. A loop counter is provided to repeat desired sections of a complex move pattern. Loops may be nested up to four levels deep on all axes.

1.3. VELOCITY PROFILES

The PC48 offers three options for ramping the device to speed. The traditional constant acceleration or linear velocity ramp (see Figure 1-1) is the default at power up or reset. The half sinusoid acceleration or half cosine velocity ramp (see Figure 1-3) is selected by the CN command. Since the acceleration is zero at the velocity inflection points, this offers very smooth operation. It is used in sensitive applications such as wafer handling on a vacuum chuck. The third option is a reverse ramp of acceleration or parabolic velocity curve (see Figure 1-2), which can be selected by the PN command. This ramp is commonly used to compensate for loss of motor torque at high speeds, i.e. since the acceleration is reduced at higher speeds the required forces are reduced proportionally. The parabola may be truncated to allow the user to select, under program control, the reduction in acceleration (force) appropriate for the application.

LINEAR RAMPS. The OMS controls generate a linear velocity ramp in real time, i.e. while the stage is in motion. There is no table building prior to the move and thus minimal latency. The controls will accelerate to the specified velocity and hold that speed until just enough move distance is left, then decelerate to a stop. If the move distance is too short to reach speed, a triangular velocity ramp will automatically be generated. The acceleration is a constant A_m and the velocity is then:

$$v = A_m t$$

A useful relationship is the distance required to accelerate at acceleration A_m to peak velocity V_p is:

$$s = \frac{V_p^2}{2A_m}$$

or the acceleration A_m required to accelerate to peak velocity V_p in distances s is:

$$A_m = \frac{V_p^2}{2s}$$

PARABOLIC RAMPS. The parabolic ramp is generated in a similar fashion except the acceleration is reduced as the stage accelerates to speed thus reducing the velocity slope, as shown in Figure 1-2.

The acceleration follows the equation:

$$a = A_0 - A_0 \frac{t}{T_2}$$

and the velocity is then:

$$v = A_0 t - \frac{A_0 t^2}{2T_2}$$

and the distance traveled in the ramp is:

$$s = \frac{A_0 t^2}{2} - \frac{A_0 t^3}{6T_2}$$

where A_0 is the initial acceleration, t is time during the ramp and T_2 is total ramp time if the acceleration had reached zero. The parameter supplied with the PN command is 10 times the ratio $\frac{t}{T_2}$ which can take on values from 3 to 10, allowing the final acceleration to range from 70% to 10% respectively of the programmed or initial value. When a move is specified, the controls will fit the resulting velocity curve to the desired acceleration profile. This insures that the desired acceleration is always reached at the programmed velocity, as long as the move is long enough for the stage to reach the programmed speed. If the move is too short to reach the programmed speed the curve is truncated, causing the shape of the velocity curve to remain the same up to the velocity reached by the specific move. This is consistent with the desired result of compensating for loss of motor torque. Since the motor has not reached the programmed speed, less compensation is needed. The parabolic ramp mode may result in reduced move time at high speeds, since a larger acceleration may be used.

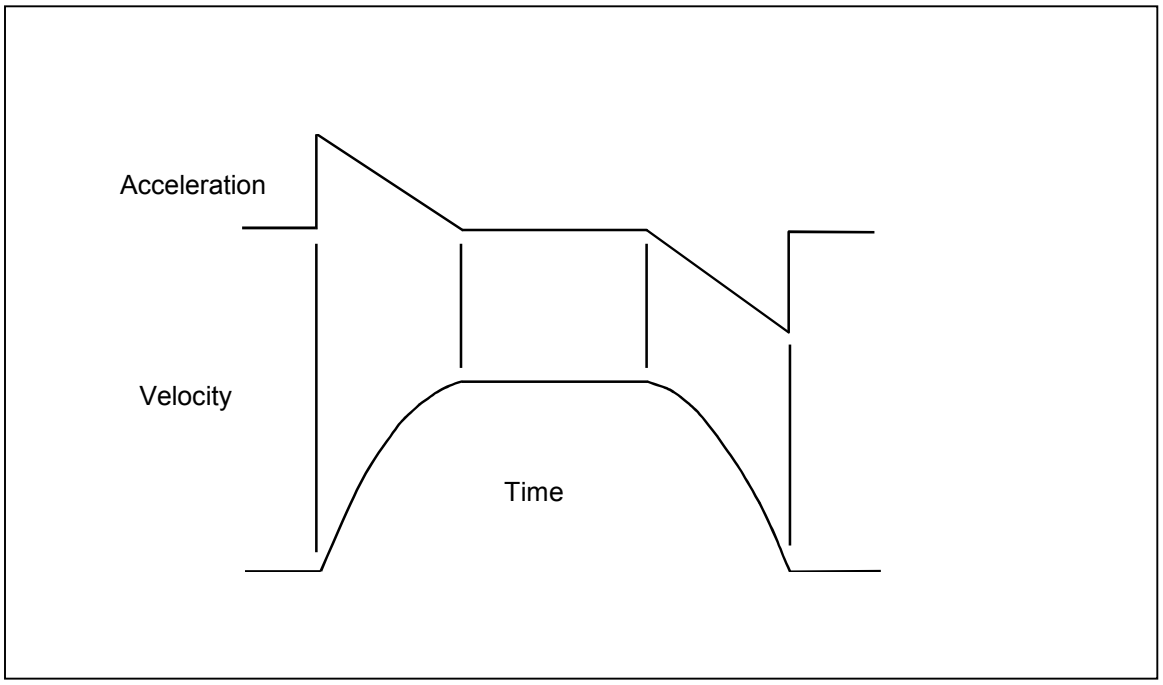


FIGURE 1-2 PARABOLIC VELOCITY PROFILE

COSINE RAMPS. The cosine ramps are generated in a similar fashion to the parabolic ramps, except the acceleration is:

$$a = A_m \sin\frac{2A_m}{V_p} t$$

and the velocity is then:

$$v = \frac{V_p}{2}\left(1 - \cos\frac{2A_m}{V_p}t\right)$$

and the distance traveled in the ramp is:

$$s = \frac{V_p}{2}t - \frac{V_p{}^2}{4A_m}\sin\frac{2A_m}{V_p}t$$

where V_p is the peak velocity, A_m is the peak acceleration. The distance needed to ramp up is then:

$$S_1 = \frac{\pi V_p^2}{4A_m}$$

and the time required to ramp up is:

$$T = \frac{\pi V_p}{2A_m} = \sqrt{\frac{\pi S_1}{A_m}}$$

and the peak velocity is:

$$V_p = \sqrt{\frac{4A_m S_1}{\pi}}$$

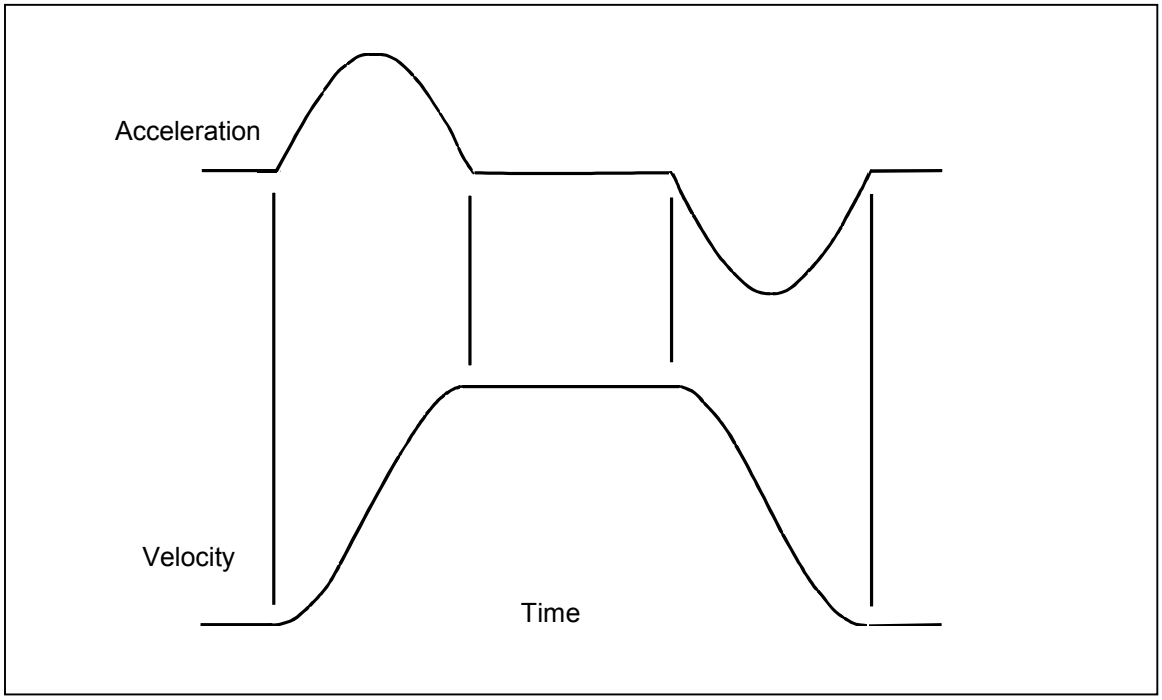


FIGURE 1-3 COSINE VELOCITY PROFILE

The cosine ramp requires $\frac{\pi}{2}$ times longer than a linear ramp to reach the same velocity when using the same peak acceleration.

Since the purpose of the cosine ramp is smooth operation, it is desirable to adjust the velocity parameters such that the desired profile is achieved even when the stage does not reach the programmed speed as opposed to truncating the curve as the parabolic modes do. The OMS controls look ahead to determine if the stage will be able to reach speed in the programmed move. If not, the acceleration curve will be adjusted such that the peak acceleration will be the programmed acceleration and the acceleration curve will be 360 degrees of a sine wave (see Figure 1-4).

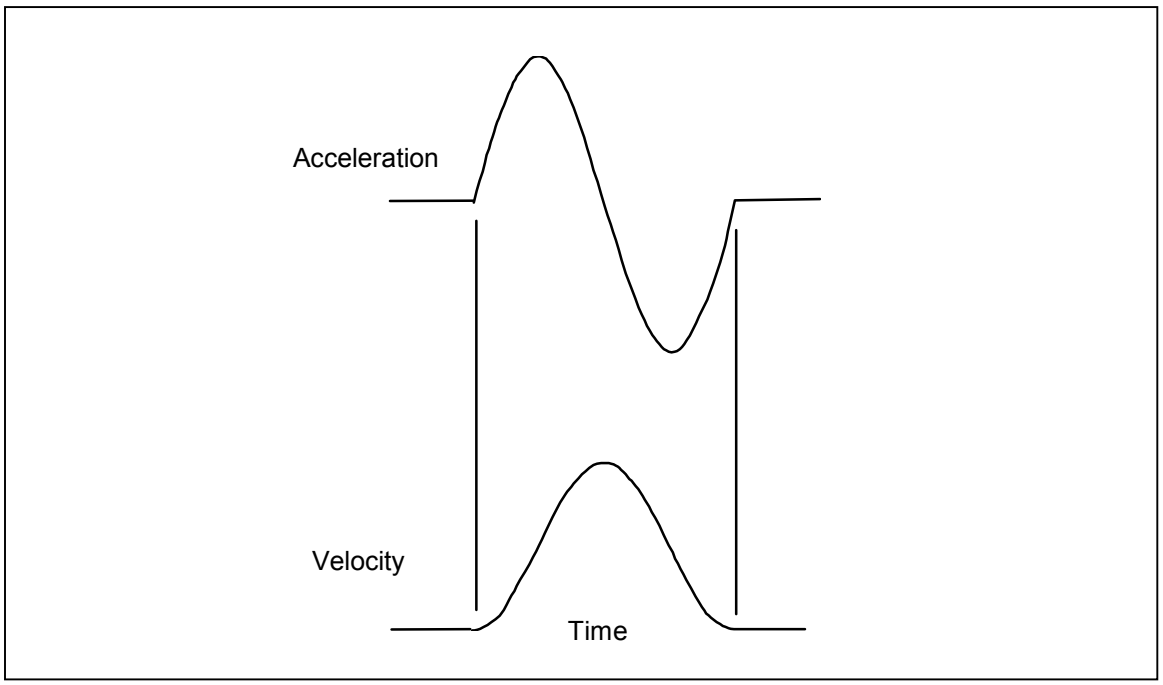


FIGURE 1-4 SHORT MOVE COSINE VELOCITY PROFILE

2. GETTING STARTED

2.1. INTRODUCTION

The PC48 board requires one half-length slot in the PC/AT. In most cases the jumpers on the PC48 board will not have to be changed, assuming there are no conflicts with interrupt and I/O address lines. The factory default settings for the board have it using I/O addresses 300 to 303 (hex) and the IRQ5 interrupt line. If these do not conflict with any previously installed hardware in your computer, you will not need to change any jumpers on the PC48 board.

2.2. JUMPERS

There are six blocks of square pin jumpers on the PC48 board J11, J14, J16, J44, J86 and J96. See Figure 3-1 for the location of the jumpers.

J44, which is half-way up the board above the edge connector, determines whether the limit inputs to an individual axis are active low or active high. With the jumper in place, the associated axis will stop moving if the limit line, for the direction the axis is moving, is grounded. With the jumper removed, the axis will stop if the limit line is at +5VDC. These lines are internally pulled-up with a 2.2K ohm resistor to +5VDC, so that only a switch is required to control the limit lines.

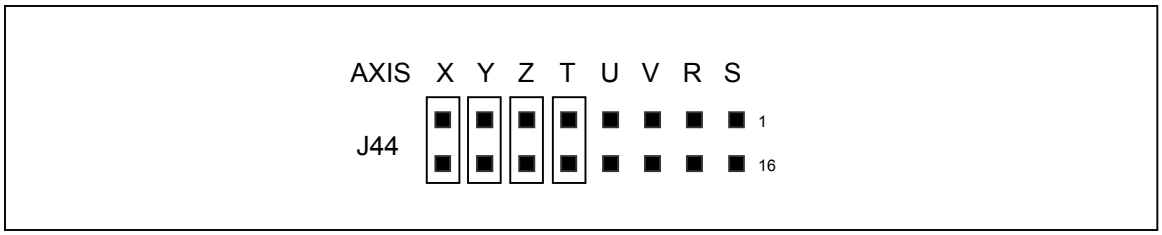


FIGURE 2-1 LIMIT SENSE JUMPERS FOR A FOUR AXIS BOARD

J16, just above the edge connector at the bottom of the board, determines the I/O address range of the board. A jumper across a pair of pins sets that bit in the address low. With the jumper removed the bit is high. The computers I/O address ranges from 200 to 3FF (hex). The A0 and A1 address lines are decoded internally by the PC48 board and are assumed to be 0 for base address calculations. The jumpers set the base address lines A2 through A9. The factory default address is 300 hex. This requires jumpers across A2 and A3, making all four of the lowest bits a 0 and the least significant hex digit of the address a 0. Jumpers across A4, A5, A6 and A7 make the middle four bits 0, making the middle hex digit of the address a 0. Removing the jumpers from A8 and A9 makes each of these lines a 1, making the most significant hexadecimal digit a 3.

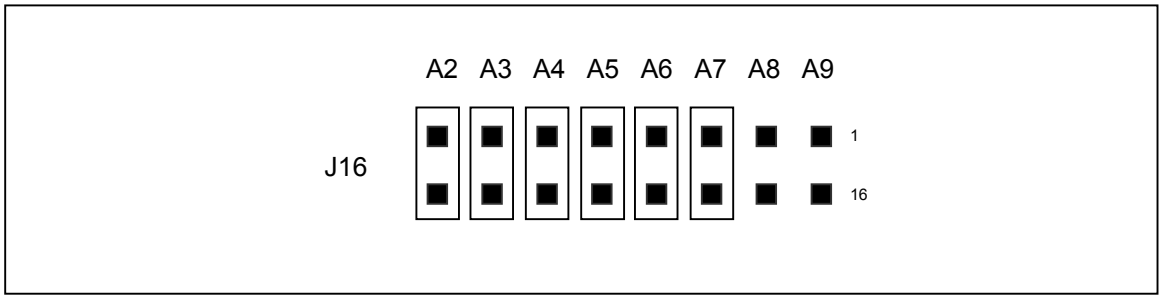


FIGURE 2-2 ADDRESS SELECT JUMPERS (DEFAULT SETTING)

J14, just above the edge connector, determines which interrupt level the PC48 board will use when it communicates. IRQ5 is the factory default setting.

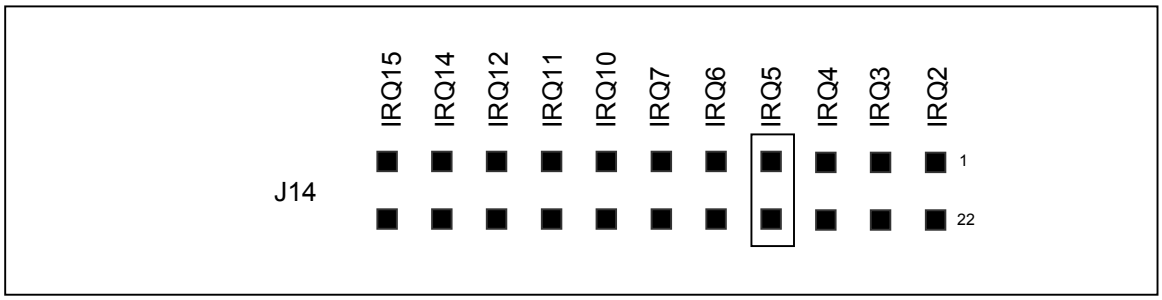


FIGURE 2-3 INTERRUPT JUMPERS (DEFAULT SETTING)

J11, located at the bottom left of the board, selects the configuration of I/O bits 0 through 13 as inputs or outputs. No jumpers on pins 1&8 and 2&7 of J11 selects I/O bits 0-7 as inputs, Jumpers on pins 3&6 and 4&5 select I/O bits 8 through 13 as outputs. Figure 2-4 shows the default settings.

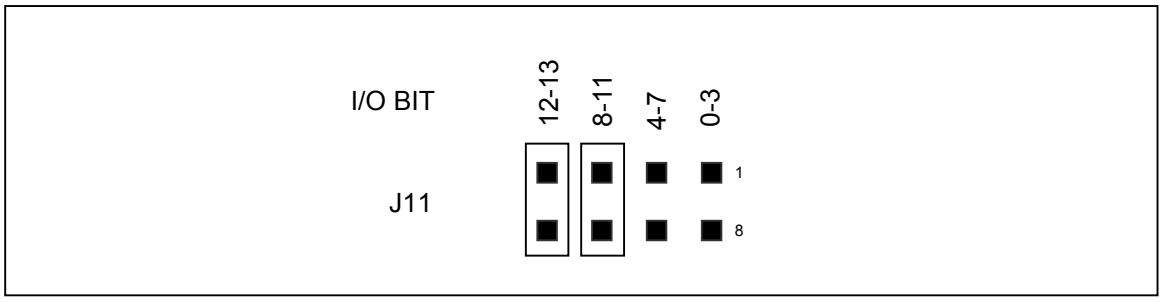


FIGURE 2-4 I/O CONFIGURATION JUMPERS (default setting)

J96, in the upper right corner of the board, and J86, just below J96, are for biasing unused encoder inputs. When a single ended encoder is used, the unused inputs can be biased using wire-wrap wire or jumpers. J86 is for the X and Y axes and J96 is for the Z and T axes. See Figure 3-1 for jumper locations.

NOTE: Jumpers J20, J21 and J80 are intended for factory use *ONLY* and are not to be reconfigured by the user.

2.3. HARDWARE INSTALLATION

1. Follow the instructions included in your controller's software support. (Usually these can be found in a file called README.TXT.)
2. Turn off power to your computer and disconnect its power cord.
3. Remove the computer's cover.
4. Choose an empty expansion slot in the motherboard and remove its associated metal cover from the back of the computer. Be sure to save the screw.
5. Check the PC48 board's jumpers for proper configuration. Note: It is a good practice to operate the PC48 at the default jumper settings for the first time and for trouble-shooting your system, then proceed with the customizing the jumpers to your specific application.
6. Slide the PC48 board down into the edge connector, insuring the board is lined up correctly in the card guides and in the connector.
7. Double check the board to insure it is properly seated in the connector.
8. Use the screw removed from the cover to attach the metal bracket of the PC48 board to the chassis of the computer.
9. Replace the cover of the computer.
10. Replace the power cord and turn the computer on. (Do not connect the PC48 to other parts of system until communication is established with the host for ease in trouble-shooting.)
11. Allow the computer to boot up.
12. Install the OMS support software disk in the PC after boot up is complete. Type: PC4XCOMM at the command prompt with the drive letter corresponding to the location of the support software disk. This will load the program that will allow the user to enter PC48 control commands at the keyboard or download a file of commands. The responses from the PC48 board will be displayed on the computer's screen.
13. Press <Enter> when the program asks for a file name.

14. The program will state that you are in the interactive mode. Enter the PC48 commands EN WY on the keyboard. EN turns echoing on which causes the typed commands to be echoed from the controller to the display. WY asks the PC48 board "who are you". The board responds with its model type and firmware version number (i.e. PC48 ver 3.00-4).

15. If a message similar to this is displayed, the board was correctly installed and you are communicating with the PC48.

16. If no message was displayed, double check the boards installation. If you are still having a problem contact Oregon Micro Systems for assistance.

2.4. SOFTWARE INSTALLATION

The support software disk contains several versions of a program which allow the user to interact with an Oregon Micro Systems PC48 motion control board. The user may type PC48 commands on the computer's keyboard and they are passed to the PC48 board. The PC48 board's responses are displayed on the computer's screen. The user may, when prompted, optionally give the program the name of an ASCII text file that contains PC48 commands. The program will send the contents of the file to the PC48 board.

Example code is included on the disk, allowing PC48 users to use the routines in application programs using OMS motion controls. No license is required.

See section 7. HOST SOFTWARE of this manual for more information on the software provided by OMS.

2.5. MOTOR CONTROL CONNECTOR

The motor control connector (J69) on the PC48 board consists of two rows of square pins, each row has 40 pins for a total of 80 pins. Fourteen of the pins are general purpose I/O lines which can be used for any function the user wishes to define. They can also be used to synchronize motor movement with external events or vice versa.

The other 64 pins can be considered as 8 logical sets of 8 pins. Each set is used to control an individual axis. If your board is not an 8 axis board or a 4 axis board with encoder feedback, some of the sets of pins will not be used.

The 8 pins of an axis set are: Step Output, +5VDC Output, Auxiliary Output, Direction Output, Negative Limit Switch Input, Ground, Home Switch Input and Positive Limit Switch Input. See Section 4. DRIVER INTERFACE for a detailed description of the connector.

Boards with the Encoder Option use two or four sets of 8 pins as encoder input for the X, Y, Z and T axes. The 8 pins of an encoder pin set are: Index +, +5VDC, Phase A -, Phase A +, Index -, Ground, Phase B -, Phase B +. See Section 5. ENCODER OPTION for a detailed description of the connector.

3. I/O CHANNEL INTERFACE

3.1. I/O CHANNEL

The PC system bus has available several slots for interfacing I/O cards such as the PC48. The AT has an additional connector to make available the 16 bit data bus and additional address lines. Since the PC48 is an 8 bit I/O device, it uses only the low order 8 bits of the data bus and the first 10 address lines in this family of computers. It can thus be used in any computer which meets the IBM I/O channel specification. It may not fit physically in some XT computers due to conflict of the AT style connector with parts on the mother-board. The PC48 is 1/2 of a full-length board. Table 3-1 and Table 3-2 show the I/O channel signals which are also the descriptions of the pins on the edge connector of the PC48 board. The front, or component side of the board, is the A side of the connector on the PC48 while the back, or solder side, is the B side.

3.1.1. PC/AT DATA BUS

The data bus is an 8 bit, bi-directional, 3-state bus. Direction of data is controlled by the PC/AT host computer. The data bus uses high-level active logic. All data and commands are passed through this data bus.

3.1.2. PC/AT ADDRESS BUS

The address bus is a 20 bit high-level active bus. This bus is always driven by the PC/AT host computer. The address bus provides the 20 address lines for decoding by either memory or I/O. MEMW, MEMR, IOW and IOR control lines distinguish between memory and I/O transfers and determine the direction of transfer. Since the PC48 is an I/O device it uses only A0 through A9 and only IOW and IOR to determine the direction of transfer.

3.1.3. MEMORY AND I/O CONTROL

These lines provide the signals for fundamental memory and I/O operations.

3.1.4. INTERRUPT REQUEST

The PC/AT supports interrupts to the system bus at levels 2 (IRQ2) through 7 (IRQ7), 10 through 12, 14 and 15. These are active high input only signals to the system bus. They go directly to the system interrupt controller which generates the vector during an interrupt acknowledge cycle. The PC48 supports all 11 interrupt levels which can be selected by a jumper at J14. Refer to Section 3.3 for details of configuration.

TABLE 3-1 CONTROL LINE DESCRIPTION

SIGNAL	DESCRIPTION
MEMW*	This is a low level active signal used to write data from the system bus into memory and is thus not used by the PC48
MEMR*	This is a low level active signal used to read data from memory to the system bus and is thus not used by the PC48
IOW*	This is a low level active signal used to write data into I/O. It is driven by the system bus and indicates the address bus contains a valid I/O address. It is used by the PC48 as an address qualifier for write operations.
IOR*	This is a low level active signal used to read data from an I/O device onto the system bus. It is driven by the system bus and indicates the address bus contains a valid I/O address. It is used by the PC48 as an address qualifier for read operations.
ALE	This output only signal driven by the system bus is decoded by the PC48 to indicate a valid address on the system bus.

3.1.5. CLOCK AND OSC LINES

These are provided on the system bus. The PC48 has its own on board clock and does not use either of these signals.

3.1.6. RESET DRV

This line is a reset driver which is provided on the bus. The PC48 has an on board reset timer and uses the RESET DRV signal to initiate this timer at power up or during a system reset. This bus signal is not active during a warm boot such as the CTL-ALT-DEL or CTL-BREAK and thus the PC48 is not reset under these conditions. It can, however, be reset on computers equipped with a reset push button.

3.1.7. I/O CH RDY

This line is used in conjunction with memory wait state generation. It is used by the PC48 since wait states are not required.

3.1.8. I/O CH CK

This signal is used in conjunction with data parity checking. It is used since parity is not checked by the PC48.

TABLE 3-2 I/O CHANNEL CONNECTOR PIN LIST

PIN	DESCRIPTION	PIN	DESCRIPTION
B1	Ground	A1	IO CH CK*
B2	Reset Drive	A2	D7
B3	+5VDC	A3	D6
B4	IRQ2	A4	D5
B5	-5VDC	A5	D4
B6	DRQ2	A6	D3
B7	-12VDC	A7	D2
B8	Unused	A8	D1
B9	+12VDC	A9	D0
B10	Ground	A10	IO CH RDY
B11	MEMW*	A11	AEN
B12	MEMR*	A12	A19
B13	IOW*	A13	A18
B14	IOR*	A14	A17
B15	DACK3*	A15	A16
B16	DRQ3	A16	A15
B17	DACK1*	A17	A14
B18	DRQ1	A18	A13
B19	*REFRESH	A19	A12
B20	CLK	A20	A11
B21	IRQ7	A21	A10
B22	IRQ6	A22	A9
B23	IRQ5	A23	A8
B24	IRQ4	A24	A7
B25	IRQ3	A25	A6
B26	DACK2*	A26	A5
B27	T/C	A27	A4
B28	ALE	A28	A3
B29	+5VDC	A29	A2
B30	OSC	A30	A1
B31	Ground	A31	A0

* Indicates low true signal

3.2. ADDRESS SELECTION

The PC48 is operated as an I/O mapped device. Each board occupies a block of 4 contiguous I/O addresses. The factory default addresses are from base address 300 through 303 hex. Refer to Figure 3-1 for configuration of jumpers[1]. The actual address is chosen by jumpers on J16. Connecting a jumper selects a 0 for that address bit, while no jumper selects a 1. The address bits are selected by the jumpers as shown in Figure 3-1.

[1] The jumpers are wire-wrap posts on 0.1 inch centers which can be shorted with a shorting plug or by wire-wrapping. For additional shorting plugs see Molex Corp. part number 90050-0007.

TABLE 3-3 ISA / E-ISA BUS PIN IDENTIFICATION (J13)

PIN	DESCRIPTION	PIN	DESCRIPTION
D01	MEMCS16*	C01	SBHE*
D02	IOCS16*	C02	LA23
D03	IRQ10	C03	LA22
D04	IRQ11	C04	LA21
D05	IRQ12	C05	LA20
D06	IRQ15	C06	LA19
D07	IRQ14	C07	LA18
D08	DACK0*	C08	LA17
D09	DRQ0	C09	MEMR*
D10	DACK5*	C10	MEMW*
D11	DRQ5	C11	SD8
D12	DACK6*	C12	SD9
D13	DRQ6	C13	SD10
D14	DACK7*	C14	SD11
D15	DRQ7	C15	SD12
D16	+5VDC	C16	SD13
D17	MASTER16*	C17	SD14
D18	GROUND	C18	SD15

* Indicates low true signal

TABLE 3-4 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

TABLE 3-5 INTERRUPT LEVEL SELECTION JUMPERS

FUNCTION	JUMPER	PIN	PIN	FUNCTION
Local IRQ	J14	1	22	IRQ2
Local IRQ	J14	2	21	IRQ3
Local IRQ	J14	3	20	IRQ4
Local IRQ	J14	4	19	IRQ5
Local IRQ	J14	5	18	IRQ6
Local IRQ	J14	6	17	IRQ7
Local IRQ	J14	7	16	IRQ10
Local IRQ	J14	8	15	IRQ11
Local IRQ	J14	9	14	IRQ12
Local IRQ	J14	10	13	IRQ14
Local IRQ	J14	11	12	IRQ15

3.3. USING INTERRUPTS

Full interrupt capability is provided by the PC48 in accordance with the I/O channel specification. Interrupts for input buffer full (IBF), transmit buffer empty (TBE), overtravel fault and operation complete are provided. Interrupt levels 2-7, 10-12, 14&15 are jumper selectable at J14. Polled operation is also supported with separate status bits for each of the above sources. Although, the preferred method of operation is to run in interrupt-mode, Table 3-5 shows the detail of the level select jumpers on J14.

3.4. I/O REGISTERS

The PC48 occupies 4 contiguous addresses in PC /AT I/O space. The registers associated with each address are described in the following paragraphs.

3.4.1. DATA REGISTER

The data register is the data communication port between the PC48 and the PC/AT or compatible host. All data is passed between the host processor and the PC48 68332 processor through this port. This port is full duplex and double buffered in both directions allowing data to be written to the port before the previous byte has been read and processed. This allows for faster processing of the data between the host and the PC48.

3.4.2. DONE FLAG REGISTER

The done flag register is a read only register from the PC48. The status bit indicating the done status of each axis is written by the 68332 on the PC48. The host can then read it at any time to determine the status of the motion process. This register is cleared automatically after it is read. The bit definition is shown in Table 3-6.

TABLE 3-6 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

3.4.3. INTERRUPT CONTROL REGISTER

This read/write register allows different interrupt sources from the PC48 to be individually enabled or disabled. This may be performed at any time by a write to this register. The register may be read back to verify or determine the state of the interrupts. (See Table 3-7)

NOTE: There is no provision for using DMA Mode in the PC48 Family.

TABLE 3-7 I/O 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	

3.4.4. STATUS REGISTER

The status register is a 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.

NOTE: Operation with polled mode is NOT recommended.

In order to resolve the source of a done or error interrupt, the DON_S bit (bit 4) should be read first. This bit in the status register is automatically reset upon reading the status register. If the DON_S flag is true, the error bits should be read to determine if the interrupt was caused by an error condition. If no error condition is present, the done flag register can be read to determine which axes are done. The TBE_S bit is reset by writing to the data register and the IBF_S bit is reset by reading the data register.

The error bits, CMD_S, ENC_S and OVRT are automatically reset after reading the 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. (See Table 3-8)

TABLE 3-8 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.
4	DON_S	Done or error status. This high true bit indicates the command is complete, i.e. an ID command has been executed or an error 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 used.
1	INIT	Init flag. This bit indicates the PC48 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 patched pairs.

3.5. POWER SUPPLY REQUIREMENTS

The PC48 is designed to operate from the power supplied in the PC/AT backplane. The host computer must be capable of supplying 1.14 amps typical for operation of the PC48 board.

CAUTION:

Under no circumstances should the card be installed in the computer with the power on.

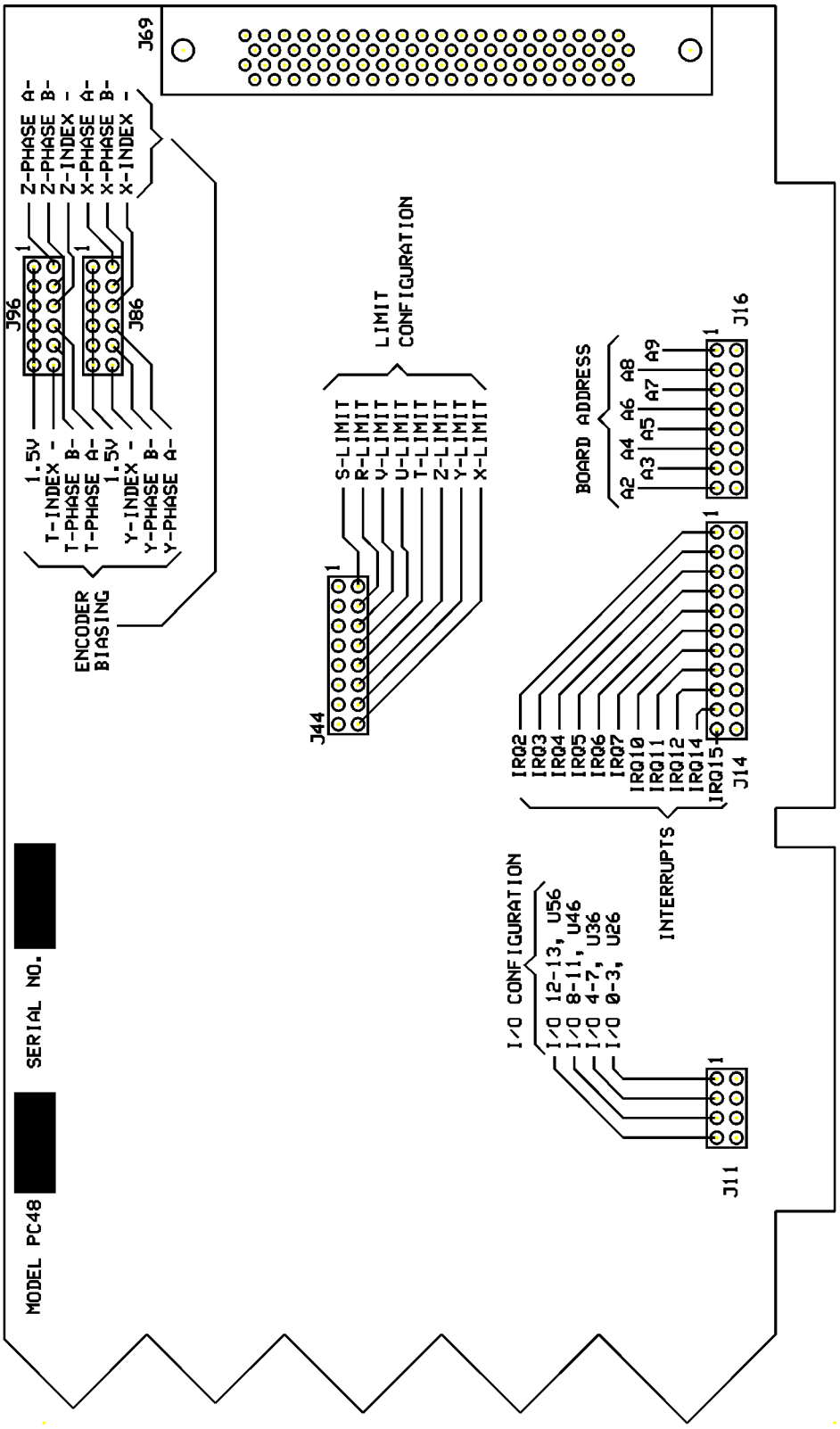


FIGURE 3-1 LOCATION OF OPTION JUMPERS

4. DRIVER INTERFACE

4.1. OUTPUT CONNECTIONS

Table 4-1 lists the input and output interface signals available at output connector J69 on each PC48 board. The connector uses 0.025 inch square posts on 0.05 by 0.10 inch centers. The connector pin assignment is shown as viewed looking into the connector on the board. The mating connector is an AMP, Inc. part number 749111-7 with a 749196-1 hood and strain relief.

Refer to Section 5 for pin assignments on PC48 boards with the encoder option.

A separate 4-conductor shielded cable should be used for each axis for connections to its associated driver module and must be limited to 50 feet. A connection to the PC/AT +5VDC power is provided for each axis to supply power to the emitter diode within an opto-isolated motor driver module. This allows the use of such drivers without the need for an external power supply.

CAUTION:

This power supply connection must not be connected to any other supply or used for any other purpose or damage may result to the host computer or PC48 or both.

A ground connection is provided for each axis for convenience in connecting up the system. The PC48 is supplied with 7406 open collector TTL drivers as standard. Each device handles the step, direction and auxiliary output for two axes. U37 on the board handles the X and Y axes, U27 handles the Z and T axes, U67 is for the U and V axes while U85 handles the R and S axes. The cable shields should be connected to the appropriate ground pins, as shown in Table 4-1, and left open at the driver end when used with opto-isolated loads to avoid ground loops and ensure isolation.

The pin numbering on J69 is as follows:

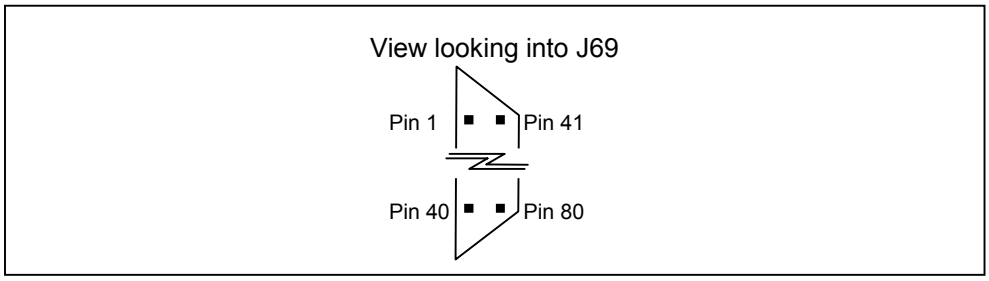


FIGURE 4-1 J69 PIN ORIENTATION

TABLE 4-1 J69 INPUT AND OUTPUT PIN ASSIGNMENTS

FUNCTION	PIN	PIN	FUNCTION
User I/O 0	1	41	+5VDC
User I/O 2	2	42	User I/O 1
User I/O 4	3	43	User I/O 3
User I/O 6	4	44	User I/O 5
User I/O 8	5	45	User I/O 7
User I/O 10	6	46	User I/O 9
User I/O 12	7	47	User I/O 11
User I/O 13	8	48	Ground
U Direction Output	9	49	+5VDC
U Auxiliary Output	10	50	U Step Output
U Positive Limit Switch	11	51	Ground
U Home Switch	12	52	U Negative Limit Switch
V Direction Output	13	53	+5VDC
V Auxiliary Output	14	54	V Step Output
V Positive Limit Switch	15	55	Ground
V Home Switch	16	56	V Negative Limit Switch
X Direction Output	17	57	+5VDC
X Auxiliary Output	18	58	X Step Output
X Positive Limit Switch	19	59	Ground
X Home Switch	20	60	X Negative Limit Switch
Y Direction Output	21	61	+5VDC
Y Auxiliary Output	22	62	Y Step Output
Y Positive Limit Switch	23	63	Ground
Y Home Switch	24	64	Y Negative Limit Switch
Z Direction Output	25	65	+5VDC
Z Auxiliary Output	26	66	Z Step Output
Z Positive Limit Switch	27	67	Ground
Z Home Switch	28	68	Z Negative Limit Switch
T Direction Output	29	69	+5VDC
T Auxiliary Output	30	70	T Step Output
T Positive Limit Switch	31	71	Ground
T Home Switch	32	72	T Negative Limit Switch
R Direction Output	33	73	+5VDC
R Auxiliary Output	34	74	R Step Output
R Positive Limit Switch	35	75	Ground
R Home Switch	36	76	R Negative Limit Switch
S Direction Output	37	77	+5VDC
S Auxiliary Output	38	78	S Step Output
S Positive Limit Switch	39	79	Ground
S Home Switch	40	80	S Negative Limit Switch

4.2. MULTI-AXIS SYNCHRONIZATION

Each PC48 has provision for synchronizing several PC48 boards to drive larger systems. Systems requiring more than 8 axes and thus more than one PC48 can be synchronized by connecting an auxiliary or general purpose output on one board to the general purpose input on the other board. The boards can signal each other at the appropriate place in the command stream without interrupting the host computer. Synchronization can be accomplished with other devices as well.

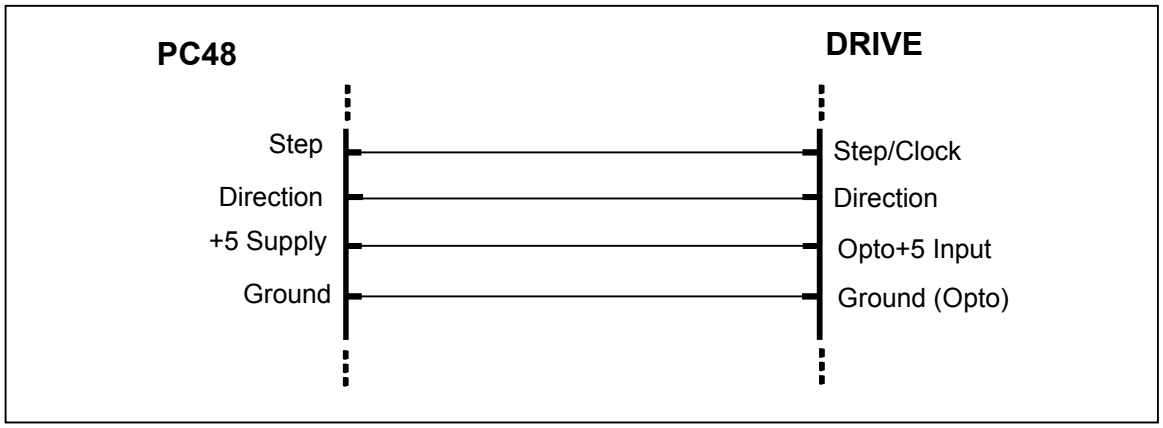


FIGURE 4-2 CONNECTION TO STEP DRIVES

4.3. LIMIT AND HOME LINES

The limit and home lines can be activated using mechanical switches using contact closures or other suitable active switches, such as a hall effect switch or opto-isolator, that connect the line to ground. The limit switch closure will stop the associated pulse stream if the motor travels beyond its allowable limits and trips the switch. 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 HM command, will cause the motor to stop when the switch closes. On finding the home position the internal position counters will be initialized. The sense of the home switches may be changed to true when open, if desired, by use of the HH command. The limit switches may be changed to true when open, if desired, by removing the jumper on J44. Figure 4-2 CONNECTION TO STEP DRIVES shows a typical connection between a PC48 board and a motor using an OMS motor driver.

4.4. FUSED PROTECTION

The external +5VDC supply available at the connector J69 of the PC48 is protected by a semiconductor type fuse. This supply is intended to be utilized with accessories used in conjunction with the PC48 such as the IO38 module, motor driver modules, etc., and is specified to supply a maximum current of 1 amp for these purposes. If an over current situation (such as an external short circuit) is detected by the fuse, the supply will shut down. It can be re-activated by powering the PC48 down, ensuring the over current situation has been removed, and restoring power to the PC48.

TABLE 4-2 IO38 DRIVER CONNECTIONS

FUNCTION	PINS		FUNCTION
Ground	1	6	Ground
Step Pulse	2	7	Positive Limit
Direction	3	8	Negative Limit
Auxiliary	4	9	Home Input
+5VDC	5		

TABLE 4-3 IO38 ENCODER CONNECTIONS

FUNCTION	PINS		FUNCTION
Ground	1	6	Ground
Index +	2	7	Phase B+
Phase A+	3	8	Index-
Phase A-	4	9	Phase B-
+5VDC	5		

TABLE 4-4 IO38 USER I/O CONNECTIONS

FUNCTION	PINS		FUNCTION
Ground	1	14	Ground
I/O Bit 0	2	15	I/O Bit 1
I/O Bit 2	3	16	I/O Bit 3
+5VDC	4	17	+5VDC
I/O Bit 4	5	18	I/O Bit 5
I/O Bit 6	6	19	I/O Bit 7
Ground	7	20	+5VDC
Ground	8	21	I/O Bit 9
I/O Bit 8	9	22	I/O Bit 11
I/O Bit 10	10	23	+5VDC
+5VDC	11	24	I/O Bit 13
I/O Bit 12	12	25	Ground
Ground	13		

TABLE 4-5 CABL38 COLOR CODES

WIRE COLOR	CONTACT	CONTACT	WIRE COLOR
White/Tan	41	1	Tan/White
White/Brown	42	2	Brown/White
White/Pink	43	3	Pink/White
White/Orange	44	4	Orange/White
White/Yellow	45	5	Yellow/White
White/Green	46	6	Green/White
White/Blue	47	7	Blue/White
White/Violet	48	8	Violet/White
White/Gray	49	9	Gray/White
Tan/Brown	50	10	Brown/Tan
Tan/Pink	51	11	Pink/Tan
Tan/Orange	52	12	Orange/Tan
Tan/Yellow	53	13	Yellow/Tan
Tan/Green	54	14	Green/Tan
Tan/Blue	55	15	Blue/Tan
Tan/Violet	56	16	Violet/Tan
Tan/Gray	57	17	Gray/Tan
Brown/Pink	58	18	Pink/Brown
Brown/Orange	59	19	Orange/Brown
Brown/Yellow	60	20	Yellow/Brown
Brown/Green	61	21	Green/Brown
Brown/Blue	62	22	Blue/Brown
Brown/Violet	63	23	Violet/Brown
Brown/Gray	64	24	Gray/Brown
Pink/Orange	65	25	Orange/Pink
Pink/Yellow	66	26	Yellow/Pink
Pink/Green	67	27	Green/Pink
Pink/Blue	68	28	Blue/Pink
Pink/Violet	69	29	Violet/Pink
Pink/Gray	70	30	Gray/Pink
Orange/Yellow	71	31	Yellow/Orange
Orange/Green	72	32	Green/Orange
Orange/Blue	73	33	Blue/Orange
Orange/Violet	74	34	Violet/Orange
Orange/Gray	75	35	Gray/Orange
Yellow/Green	76	36	Green/Yellow
Yellow/Blue	77	37	Blue/Yellow
Yellow/Violet	78	38	Violet/Yellow
Yellow/Gray	79	39	Gray/Yellow
Green/Blue	80	40	Blue/Green

4.5. IO38 ADAPTER MODULE

The optional IO38 is an adapter module designed to provide separate connectors for each axis and each encoder input when used with the PC48. It includes a 12 foot cable with mating connector to fit the PC48 I/O connections. Each driver module has its own 9 pin subminiature D connector, as does each encoder input, when used with the PC48 equipped with encoder feedback. The mating connector is an AMP, Inc. part number 747944-2 or equivalent. The user defined I/O uses a 25 pin subminiature D connector. This mating connector is an AMP, Inc. part number 747948-2 or equivalent. The previous Table 4-2, Table 4-3, Table 4-4 and Table 4-5 show the connections to the IO38.

Table 4-5 is a color code for the PC48 and IO38 cable.

5. ENCODER OPTION

5.1. INTRODUCTION

The encoder feedback option is intended primarily for applications where desired positional accuracy exceeds the accuracy of the mechanical drive components, such as lead screws, or position feedback is required to detect motor slip or stall.

The encoder option accepts quadrature pulse outputs from high resolution optical encoders. Up to 50,000 pulse per revolution encoders may be used while the indexers are generating pulses at their maximum rate. This allows position feedback information to match the resolution of the microstepping motor drive. The X and Y axes may be configured for encoder feedback on a two axis PC48 equipped with option E or the X, Y, Z and T axes on four axis boards with option E.

5.2. MODES OF OPERATION

The PC48-E can monitor the actual position through the encoder pulse train. It can then correct for position errors due to system backlash or mechanical tolerances or report slip or stall of the motor to the host. A tracking mode is also provided which allows one axis to track the activity of another axis or positioning device. These options are selectable through software commands.

5.3. ENCODER SELECTION AND COMPATIBILITY

The PC48 with option E is compatible with virtually any incremental encoder which provides quadrature outputs. Times four quadrature detection is used to increase resolution. The inputs are compatible with encoders which have single ended TTL outputs as well as differential line drivers. Provisions are also provided for an index pulse (differential or single ended) and an index enable for systems requiring more than one revolution of travel and thus multiple index pulses from the encoder. A biasing network is provided on the board for termination of unused encoder inputs.

The encoder count/motor count ratio can be specified for position maintenance and encoder tracking mode. This ratio is handled internally in floating point format and can be virtually any ratio. Slip detection requires that the encoder resolution (after the 4X quadrature detection) match the motor resolution.

5.4. ENCODER INTERFACE

The encoder connections are as shown in Table 5-1 and Table 5-2.

TABLE 5-1 J69 ENCODER INPUT AND OUTPUT PIN ASSIGNMENT

FUNCTION	PIN	PIN	FUNCTION
User I/O 0	1	41	+5VDC
User I/O 2	2	42	User I/O 1
User I/O 4	3	43	User I/O 3
User I/O 6	4	44	User I/O 5
User I/O 8	5	45	User I/O 7
User I/O 10	6	46	User I/O 9
User I/O 12	7	47	User I/O 11
User I/O 13	8	48	Ground
X Phase A+	9	49	+5VDC
X Phase A-	10	50	X Index +
X Phase B+	11	51	Ground
X Phase B-	12	52	X Index -
Y Phase A+	13	53	+5VDC
Y Phase A-	14	54	Y Index +
Y Phase B+	15	55	Ground
Y Phase B-	16	56	Y Index -
X Direction Output	17	57	+5VDC
X Auxiliary Output	18	58	X Step Output
X Positive Limit Switch	19	59	Ground
X Home Switch	20	60	X Negative Limit Switch
Y Direction Output	21	61	+5VDC
Y Auxiliary Output	22	62	Y Step Output
Y Positive Limit Switch	23	63	Ground
Y Home Switch	24	64	Y Negative Limit Switch
Z Direction Output	25	65	+5VDC
Z Auxiliary Output	26	66	Z Step Output
Z Positive Limit Switch	27	67	Ground
Z Home Switch	28	68	Z Negative Limit Switch
T Direction Output	29	69	+5VDC
T Auxiliary Output	30	70	T Step Output
T Positive Limit Switch	31	71	Ground
T Home Switch	32	72	T Negative Limit Switch
Z Phase A+	33	73	+5VDC
Z Phase A-	34	74	Z Index +
Z Phase B+	35	75	Ground
Z Phase B-	36	76	Z Index -
T Phase A+	37	77	+5VDC
T Phase A-	38	78	T Index +
T Phase B+	39	79	Ground
T Phase B-	40	80	T Index -

TABLE 5-2 PC48-6E PIN ASSIGNMENT

FUNCTION	PIN	PIN	FUNCTION
User I/O 0	1	41	+5VDC
User I/O 2	2	42	User I/O 1
User I/O 4	3	43	User I/O 3
User I/O 6	4	44	User I/O 5
User I/O 8	5	45	User I/O 7
User I/O 10	6	46	User I/O 9
User I/O 12	7	47	User I/O 11
User I/O 13	8	48	Ground
X Phase A+	9	49	+5VDC
X Phase A-	10	50	X Index +
X Phase B+	11	51	Ground
X Phase B-	12	52	X Index -
Y Phase A+	13	53	+5VDC
Y Phase A-	14	54	Y Index +
Y Phase B+	15	55	Ground
Y Phase B-	16	56	Y Index -
X Direction Output	17	57	+5VDC
X Auxiliary Output	18	58	X Step Output
X Positive Limit Switch	19	59	Ground
X Home Switch	20	60	X Negative Limit Switch
Y Direction Output	21	61	+5VDC
Y Auxiliary Output	22	62	Y Step Output
Y Positive Limit Switch	23	63	Ground
Y Home Switch	24	64	Y Negative Limit Switch
Z Direction Output	25	65	+5VDC
Z Auxiliary Output	26	66	Z Step Output
Z Positive Limit Switch	27	67	Ground
Z Home Switch	28	68	Z Negative Limit Switch
T Direction Output	29	69	+5VDC
T Auxiliary Output	30	70	T Step Output
T Positive Limit Switch	31	71	Ground
T Home Switch	32	72	T Negative Limit Switch
U Direction Output	33	73	+5VDC
U Auxiliary Output	34	74	U Step Output
U Positive Limit Switch	35	75	Ground
U Home Switch	36	76	U Negative Limit Switch
V Direction Output	37	77	+5VDC
V Auxiliary Output	38	78	V Step Output
V Positive Limit Switch	39	79	Ground
V Home Switch	40	80	V Negative Limit Switch

If single ended encoders are used, the unused line receiver inputs must be biased in the middle of the voltage swing of the active output. J86 and J96 are provided with a built in bias supply. The appropriate unused inputs should be connected to the +1.5VDC supply as needed. Reference Figure 3-1 for the pin definition of J86 and J96.

Please note that the U and V axes limit select jumpers (J47) and done flag register bit assignments on the PC48-6E version controllers are as defined in the manual for the R and S axes. Table 5-2 shows the pin assignments for the model PC48-6E.

5.5. HOME PROCEDURES

Two logical inputs are provided to synchronize the physical hardware with the PC48 controller, i.e. put the controlled motor in the home position.

The PC48 home inputs can be used with encoders which provide one home pulse for the complete travel of the 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 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.

The index input on J69 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. The home logic expressed in Boolean terms is:

$$home = phase\_A * /phase\_B * index * /home\_switch$$

Note that 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.

6. COMMAND STRUCTURE

6.1. INTRODUCTION

An extensive command structure is built into the PC48 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 any command string.

The following commands in this section are included in the PC48 family of controllers. All 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. Semi-colons are the recommended terminating character because of their visibility in the command stream. No terminator is required on the other commands, but may be included to improve readability. 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 PC48 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 PC48 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 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 that 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.

The following commands are available in firmware revision 2.11 and above.

6.3. COMMAND SUMMARY

The following commands are included in the PC48 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	6-6	Any following commands are for the AA (All Axes) mode
AC#	6-22	Acceleration, set acceleration/deceleration register
AF	6-18, 6-70	Auxiliary off
AM	6-6	Axes multitasking mode
AN	6-17	Auxiliary on
AR	6-10	Any following commands are for the R axis
AS	6-10	Any following commands are for the S axis
AT	6-8	Any following commands are for the T axis
AU	6-9	Any following commands are for the U axis
AV	6-9	Any following commands are for the V axis
AX	6-7	Any following commands are for the X axis (default on reset)
AY	6-7	Any following commands are for the Y axis
AZ	6-8	Any following commands are for the Z axis
BH#	6-20, 6-71	Set selected I/O bit high (off)
BL#	6-20, 6-71	Set selected I/O bit low (on)
BX	6-21	Return bit status in hex format
CA	6-46	Clear done flag of currently addressed axis

COMMANDS IN CHAPTER 6		
COMMAND	**SECTION PAGE NUMBER**	**COMMAND DESCRIPTION**
CD#,#;	6-72	Define a contour
CE	6-73	End contour definition, ramp to a stop
CK	6-73	End contour definition, immediately stop step pulses
CN	6-15	Cosine on, enable cosine velocity profiles
CR#,#,#	6-74	Circular interpolation, move in a circle
CV#	6-74	Contouring velocity, definition
CW	6-40	Clear while flag, i.e. terminate WH/WG loop
CX	6-75	Contour execute
EA	6-65	Encoder status, return encoder status of currently addressed axis
EF	6-11	Echo off, turn off echo to host (default at power up)
EN	6-11	Echo on, turn on echo to host
ER#,#	6-57	Encoder ratio, set encoder count to motor count ratio
ES#	6-61	Encoder slip tolerance, set tolerance before slip or stall is flagged
ET	6-63	Encoder tracking, set encoder tracking mode
FP#	6-69	Force position, flush queue and attempt to stop at specified position
GD	6-31	Go and reset done flags
GO	6-30	Go command, start execution of motion
HD#	6-58	Hold deadband, specify deadband tolerance for position hold
HE	6-64	Encoder home mode, set home on encoder logic
HF	6-59, 6-62, 6-63	Hold off, disable position hold, slip detection and tracking modes
HG#	6-58	Hold gain, specify position hold gain parameter
HH	6-12	Home high, home switches are active high
HL	6-12	Home low, home switches are active low
HM#	6-41	Home, find home and initialize the position counter
HN	6-59	Hold on, enable position correction after move
HR#	6-42	Home reverse, find home in reverse direction and initialize position counter
HS	6-64	Home switch, enable home switch mode
HV#	6-57	Hold velocity, specify maximum position hold correction velocity
IC	6-46	Interrupt clear, clear done interrupt status and error flags
ID	6-44	Interrupt host when done and set done flag
II	6-44	Interrupt independent
IN#	6-45	Interrupt when nearly done
IP	6-45, 6-60	Interrupt when in position
IS	6-61	Interrupt slip, interrupts host on slip or stall detection
JF#	6-32	Jog the current axis at fractional rates
JG#	6-32	Jog command, run motor at specified velocity until a new

COMMANDS IN CHAPTER 6		
COMMAND	**SECTION PAGE NUMBER**	**COMMAND DESCRIPTION**
		velocity command is sent or it is stopped by a stop or kill command
KL	6-35	Kill, flush queue and terminate pulse generation immediately on all axes without decelerating
KM	6-42	Home and kill pulse generation
KR	6-43	Home in reverse and kill pulse generation
LE	6-37	Loop end, terminate most recent LS command
LF	6-13	Disable limit switches for selected axis
LN	6-13	Enable limit switches for selected axis
LP#	6-24	Load position, load position counter with parameter
LS#	6-36	Loop start, set loop counter, from 1 to 32000 loops; (may be nested to 4 levels)
MA#	6-25	Move absolute, move to absolute position
ML#,#;	6-27	Move linear, move specified distance relative from current position
MM	6-67	Move minus, set minus direction for MV type move
MO	6-28	Move one pulse in current direction
MP	6-67	Move plus, set positive direction for MV type move
MR#	6-26	Move relative, move specified distance from current position
MT#,#;	6-28, 6-75	Move to, move to specified position
MV#,#	6-68	Move velocity, move to first parameter (absolute position) at second parameter velocity without stopping at end of move
PA#	6-19	Power automatic, turn power on before each move and off after the move
PF	6-16	Parabolic off, disable parabolic ramps, i.e. linear ramps will be generated
PN#	6-15	Parabolic on, enable parabolic ramps
QA	6-53	Query status of switches and flags for addressed axis without affecting flags
QI	6-54	Query status of switches and flags on all axes without affecting flags
RA	6-52	Return status of switches and flags and reset flags
RB	6-21	Return programmed direction of I/O bits in hex format
RC	6-54	Return current acceleration or deceleration of the current axis
RE	6-66	Request encoder position, return current encoder position
RI	6-53	Return status of switches and flags for all axes and reset flags
RL	6-62	Return slip status of each axis
RM#	6-29	Return remainder of position divided by parameter in position counter
RP	6-50	Request position, returns current position
RQ	6-51, 6-76	Request queue status, return number of queue entries available

COMMANDS IN CHAPTER 6		
COMMAND	**SECTION PAGE NUMBER**	**COMMAND DESCRIPTION**
RS	6-16	Software reset of PC48
RU	6-55	Return current position in user units
RV	6-55	Return current velocity at which the axis is moving
SA	6-34	Stop all, flush queue and stops all axes with deceleration
SD	6-35	Stop all axes and clear any done flags
SE#	6-19	Set settling time before power is reduced in PA mode
SF	6-14	Soft limit off, restore normal overtravel operation
SL	6-14	Soft limit mode, allow pulse train to ramp down on overtravel
SP#	6-69	Stop at position, stop at specified position if possible after all commands have been executed
ST	6-34	Stop, flush queue and decelerate to stop
SW#	6-48	Sync wait, wait for the input bit to be released by other controllers
UF	6-56	User units off, turn off user unit translation
UU#	6-56	User units, multiply acceleration, velocity and distance parameters by specified parameter
VB#	6-24	Base velocity, set base velocity
VL#	6-23	Set maximum velocity to be used in profile
VS#,#,#	6-33	Velocity stream, slave velocity mode for profiling
WA	6-47	Wait until all moves on all axes are finished
WD	6-38	While end, WS loop terminator
WG	6-40	Terminate WH loop
WH	6-39	While, execute all commands until WG loop terminator, until flag cleared by CW command
WQ	6-47	Wait until current axis queue is empty
WS#	6-38	While sync, execute while sync is true
WT#	6-49	Wait, wait for specified number of milliseconds
WY	6-50	Who are you , returns model and software revision

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	
AX - AS	Immediate
AA,AM	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	
AX - AS	Immediate
AA,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	
AX - AS	Immediate
AA,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	
AX - AS	Immediate
AA,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	
AX - AS	Immediate
AA,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	
AX - AS	Immediate
AA,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	
AX - AS	Immediate
AA,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	
AX - AS	Immediate
AA,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	
AX - AS	Immediate
AA,AM	Immediate
AA/CD	Not valid

Example: Examine the queue size of the R axis.

Enter: AR RQ

AS AXIS S

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

QUEUE REQUIREMENTS	
MODE	
AX - AS	Immediate
AA,AM	Immediate
AA/CD	Not valid

Example: Stop all movement on the S axis only.

Enter: AS ST

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	
AX - AS	Immediate
AA,AM	Not valid
AA/CD	Not valid

Example: Enable echoing by the PC48 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 PC48 motion system. This is the default mode at power up or reset.

QUEUE REQUIREMENTS	
MODE	
AX - AS	Immediate
AA,AM	Not valid
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	
AX - AS	1
AA,AM	Not valid
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	
AX - AS	1
AA,AM	Not valid
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 1024 steps per second requirement of the home command.

Enter: AX VL20000 HH HM0
VL1000 HL HR0

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	
AX - AS	1
AA,AM	Not valid
AA/CD	Not valid

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

Enter: AY LF

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	
AX - AS	1
AA,AM	Not valid
AA/CD	Not valid

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

Enter: AT LN

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 moves only. This command is valid in the single axis mode only, but affects all axes simultaneously.

QUEUE REQUIREMENTS	
MODE	
AX - AS	1
AA,AM	Not valid
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

SF SOFT LIMIT OFF

The SF command restores the normal operation of the limit switches. This is the default mode at power up or reset.

QUEUE REQUIREMENTS	
MODE	
AX - AS	1
AA,AM	Not valid
AA/CD	Not valid

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

Enter: AX 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. See Section 1 for an explanation of velocity profiles. 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.**

Because of the excess processing overhead involved, absolute moves, such as MA and MT, cannot be used within loops (LS-LE, WH-WG) while the board is in the cosine (CN) velocity profile mode. Relative moves, such as MR and ML, will work properly within loops, when in the cosine mode.

QUEUE REQUIREMENTS	
MODE	
AX - AS	1
AA,AM	Not valid
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. See Section 1 for an explanation of velocity profiles. 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.**

QUEUE REQUIREMENTS	
MODE	
AX - AS	Immediate
AA - AM	Immediate
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 is the default mode at power up or reset. See Section 1 for an explanation of velocity profiles. 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**. This is the default mode at power up or reset.

QUEUE REQUIREMENTS	
MODE	
AX - AS	Immediate
AA - AM	Immediate
AA/CD	Not valid

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

Enter: PF

RS RESET

The RS command is a software reset which causes the local PC48 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 3-8. The “Initializing” in process bit goes high during the initialization process.

QUEUE REQUIREMENTS	
MODE	
AX - AS	Immediate
AA - AM	Immediate
AA/CD	Not valid

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

Enter: RS

6.6. USER I/O COMMANDS

The following commands are for accessing the bit I/O functions of the board. See also the SW and WS 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	
AX - AS	1
AA - AM	1
AA/CD	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	
AX - AS	1
AA - AM	1
AA/CD	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.

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.

QUEUE REQUIREMENTS	
MODE	
AX - AS	2
AA - AM	2
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.

Enter: AA PA,0,,1;

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.

QUEUE REQUIREMENTS	
MODE	
AX - AS	3
AA - AM	3
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;

BL# BIT LOW

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

QUEUE REQUIREMENTS	
MODE	
AX - AS	2
AA - AM	2
AA/CD	Not valid

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. 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	
AX - AS	2
AA - AM	2
AA/CD	Not valid

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

Enter: BH8; BH11;

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	
AX - AS	Immediate
AA - 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>

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	
AX - AS	1
AA,AM	Not valid
AA/CD	Not valid

Example: Factory default settings have bits 0 through 7 as inputs and 8 through 13 are outputs. Verify this with the RB command.

Enter: RB

Response: <LF><CR>003F00<LF><CR>

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. All the following move commands for the axis being programmed will accelerate or decelerate at this rate until another AC command is entered. All acceleration registers default to 1,000,000 steps per second per second upon power-up or reset.

The acceleration register may be automatically modified by the PC48 if an ML or MT instruction is sent in the AA or AM modes. The user must then redefine them with an AC command, when returning to the single axis mode, or when using move commands in the AA or AM modes which do not do interpolation, such as the MA or MR commands.

QUEUE REQUIREMENTS			
MODE	**LINEAR**	**PARABOLIC**	**COSINE**
AX - AS	4	15	15
AA,AM	4	15	15
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;

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. The velocity defaults to 100,000 at power up or reset. This is a write only register and controls the maximum velocity used in relative and absolute position moves except as modified by the linear interpolation instructions.

If the velocity register is modified by an ML or MT instruction in the AA or AM modes, the user must redefine the velocity with a VL command when returning to the single axis mode or using a move command which does not use interpolation in the AA or AM modes.

QUEUE REQUIREMENTS			
MODE	**LINEAR**	**PARABOLIC**	**COSINE**
AX - AS	2	13	13
AA,AM	2	13	13
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;

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. This command is not valid with the JG command. The base velocity defaults to zero at power up or reset.

QUEUE REQUIREMENTS			
MODE	**LINEAR**	**PARABOLIC**	**COSINE**
AX - AS	2	2	2
AA,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;

LP# LOAD POSITION

The LP command will immediately load the position supplied as a parameter in the absolute position register 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 PC48 with the encoder option. The range is -33,554,431<=LP<=33,554,431.

QUEUE REQUIREMENTS		
MODE	**NO ENCODER**	**ENCODER**
AX - AS	2	4
AA,AM	Not valid	
AA/CD	Not valid	

Example: The following would load the X axis position register with 1000.

Enter: AX LP1000

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.

Because of the excess processing overhead involved, the MA command cannot be used within loops (LS-LE, WH-WG) while the board is in the cosine (CN) velocity profile mode.

The linear move commands (ML and MT) and the constant velocity mode may alter predefined acceleration and velocity values. These values should be redefined if you go from a linear move to a non-linear move, such as an MA or MR type, in both single axis or all axes modes.

QUEUE REQUIREMENTS			
MODE	**LINEAR**	**PARABOLIC**	**COSINE**
AX - AS	2	2	2
AA,AM	2	2	2
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.

The linear move commands (ML and MT) and the constant velocity mode may alter predefined acceleration and velocity values. These values should be redefined if you go from a linear move to a non-linear move, such as an MA or MR type, in both single axis or all axes modes.

QUEUE REQUIREMENTS			
MODE	**LINEAR**	**PARABOLIC**	**COSINE**
AX - AS	2	2	2
AA,AM	2	2	2
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. Velocity and acceleration parameters of each axis may be automatically adjusted by the PC48 controller to perform the linear move. If linear and single axis moves are mixed, it will be necessary to reset the velocity and acceleration parameters for the single axis move following a linear move.

The parameters may have been modified by the PC48 depending on the relative distances of the linear 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	**LINEAR**	**PARABOLIC**	**COSINE**
AX - AS	Not valid		
AA,AM	6	30	30
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.

The MT command may alter predefined acceleration and velocity values. These values should be redefined if you go from a linear move to a non-linear move, such as an MA or MR type, in both single axis or all axes modes.

QUEUE REQUIREMENTS			
MODE	**LINEAR**	**PARABOLIC**	**COSINE**
AX - AS	Not valid		
AA,AM	6	30	30
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	
AX - AS	1
AA,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	
AX - AS	1
AA,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.

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.

TABLE A		
QUEUE REQUIREMENTS		
MODE	**NO ENCODER**	**ENCODER**
AX - AS	4	7
AA,AM	5	8
AA/CD	Not valid	

TABLE B			
ADDITIONAL QUEUE REQUIREMENTS			
MODE	**LINEAR**	**PARABOLIC**	**COSINE**
AX - AS	0	0	0
AA,AM	6	15	15
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.

TABLE A
QUEUE REQUIREMENTS

MODE	NO ENCODER	ENCODER
AX - AS	5	8
AA,AM	6	9
AA/CD	Not valid	

TABLE B
ADDITIONAL QUEUE REQUIREMENTS

MODE	LINEAR	PARABOLIC	COSINE
AX - AS	0	0	0
AA,AM	6	15	15
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).

QUEUE REQUIREMENTS			
MODE	**LINEAR**	**PARABOLIC**	**COSINE**
AX - AS	2	Linear ramp	
AA,AM	Not valid		
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.

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	**NO ENCODER**	**ENCODER**
AX - AS	2	3
AA,AM	Not valid	
AA/CD	Not valid	

Example: Jog the Y axis at 2 $^{2}/_{3}$ steps per second.

Enter: AY JF2.667

VS#,#,# VELOCITY STREAMING

The VS command will generate a pulse train without acceleration or deceleration at the rates specified. The parameters are time in 1/1024 second sample intervals, X velocity, and Y velocity. 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	
AX	5
AY - AS	Not valid
AA,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.

Enter: AX VS1024,-1000,1000; VS1024,-2000,2000; VS1024,-3000,3000; VS1024,-2000,2000; VS1024,-1000,1000; VS1,0,0;

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	
AX - AS	Flush + 2
AA,AM	Flush + 2
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	
AX - AS	Flush + 2
AA,AM	Flush + 2
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	
AX - AS	Flush + 2
AA,AM	Flush + 2
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	
AX - AS	Flush + 2
AA,AM	Flush + 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

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	
AX - AS	2
AA,AM	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	
AX - AS	2
AA,AM	2
AA/CD	Not valid

Example: (see LS command page 6-36)

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	
AX - AS	2
AA	2
AA/CD, AM	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	
AX - AS	2
AA,AM	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	
AX - AS	2
AA,AM	2
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	
AX - AS	2
AA,AM	2
AA/CD	Not valid

Example: (see WH command on page 6-39)

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	
AX - AS	1
AA,AM	1
AA/CD	Not valid

Example: (see WH command on page 6-39)

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. 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	**NO ENCODER**	**ENCODER**
AX - AS	4	6
AA,AM	Not valid	
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 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	**NO ENCODER**	**ENCODER**
AX - AS	4	6
AA,AM	Not valid	
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	**NO ENCODER**	**ENCODER**
AX - AS	2	2
AA,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	**NO ENCODER**	**ENCODER**
AX - AS	2	2
AA,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 PC48 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	
AX - AS	1
AA,AM	1
AA/CD	Not valid

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

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	
AX - AS	1
AA,AM	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	
AX - AS	2
AA,AM	2
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	
AX - AT	1
AU - AS	Not valid
AA, AM	Not valid
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, otherwise the axis would always appear to be “done”. 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.

QUEUE REQUIREMENTS	
MODE	
AX - AS	Immediate
AA,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.

QUEUE REQUIREMENTS	
MODE	
AX - AS	Immediate
AA,AM	Not valid
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

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	
AX - AS	Immediate
AA,AM	Not valid
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	
AX - AS	Immediate
AA,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 PC48 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	
AX - AS	Immediate
AA,AM	Immediate
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	
AX - AS	3
AA,AM	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

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	
AX - AS	Immediate
AA,AM	Immediate
AA/CD	Not valid

Example: You want to examine the board information.

Enter: WY

Response: <LF><CR>PC48 ver 3.00-2<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	
AX - AS	Immediate
AA,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>

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	
AX - AS	Immediate
AA,AM	Immediate
AA/CD	Not valid

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	
AX - AS	Immediate
AA,AM	Immediate
AA/CD	Immediate

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	
AX – AS	Immediate
AA,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	
AX – AS	Not valid
AA,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	
AX – AS	Immediate
AA,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	
AX – AS	Not valid
AA,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	
AX – AS	Immediate
AA,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	
AX – AS	Immediate
AA,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 6-56). 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	
AX – AS	Immediate
AA,AM	Not valid
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. This command must be given in the single axis mode but will remain effective in the AA or AM mode. The PC48 defaults to user units off at power up or reset.

QUEUE REQUIREMENTS	
MODE	
AX – AS	Immediate
AA,AM	Not valid
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

UF USER OFF

The UF command turns off user units. This command is equivalent to and preferred over UU1 since it turns off the mode thus minimizing unnecessary overhead.

QUEUE REQUIREMENTS	
MODE	
AX – AS	Immediate
AA,AM	Not valid
AA/CD	Not valid

Example: Turn off user unit conversion on the X, Y and Z axes.

Enter: AX UF AY UF AZ UF

6.15. POSITION MAINTENANCE COMMANDS

ER#,# ENCODER RATIO

The ER command allows specification of encoder ratio by entering encoder counts, followed by motor counts, for position maintenance mode. 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 ratio defaults to 1 at power up or reset.

QUEUE REQUIREMENTS		
MODE	**NO ENCODER**	**ENCODER**
AX - AT	Not valid	1
AU – AS	Not valid	
AA,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

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.

QUEUE REQUIREMENTS		
MODE	**NO ENCODER**	**ENCODER**
AX - AT	Not valid	2
AU – AS	Not valid	
AA,AM	Not valid	
AA/CD	Not valid	

Example: (see HN command on page 6-59)

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. 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.

QUEUE REQUIREMENTS		
MODE	**NO ENCODER**	**ENCODER**
AX - AT	Not valid	2
AU – AS	Not valid	
AA,AM	Not valid	
AA/CD	Not valid	

Example: (see HN command on page 6-59)

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. A parameter of zero is allowed.

QUEUE REQUIREMENTS		
MODE	**NO ENCODER**	**ENCODER**
AX - AT	Not valid	1
AU – AS	Not valid	
AA,AM	Not valid	
AA/CD	Not valid	

Example: (see HN command on page 6-59)

HF HOLD OFF

The HF command disables position hold, stall detection and tracking modes. This is the default mode at power up or reset.

QUEUE REQUIREMENTS		
MODE	**NO ENCODER**	**ENCODER**
AX - AT	Not valid	2
AU – AS	Not valid	
AA,AM	Not valid	
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	**NO ENCODER**	**ENCODER**
AX - AT	Not valid	2
AU – AS	Not valid	
AA,AM	Not valid	
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	
AX - AT	1
AU – AS	Not valid
AA,AM	Not valid
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.)

6.16. 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.

QUEUE REQUIREMENTS		
MODE	**NO ENCODER**	**ENCODER**
AX - AT	Not valid	2
AU – AS	Not valid	
AA,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 PC48 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.

QUEUE REQUIREMENTS		
MODE	**NO ENCODER**	**ENCODER**
AX - AT	Not valid	1
AU – AS	Not valid	
AA,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.

QUEUE REQUIREMENTS		
MODE	**NO ENCODER**	**ENCODER**
AX - AT	Not valid	Immediate
AU – AS	Not valid	
AA,AM	Immediate	
AA/CD	Not valid	

Example: On a four axes 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	**NO ENCODER**	**ENCODER**
AX - AT	Not valid	2
AU – AS	Not valid	
AA,AM	Not valid	
AA/CD	Not valid	

Example: Disable slip detection on the X axis.

Enter: AX HF

6.17. 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.

QUEUE REQUIREMENTS		
MODE	**NO ENCODER**	**ENCODER**
AX - AT	Not valid	2
AU – AS	Not valid	
AA,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	**NO ENCODER**	**ENCODER**
AX - AT	Not valid	2
AU – AS	Not valid	
AA,AM	Not valid	
AA/CD	Not valid	

Example: Turn off encoder tracking on X axis.

Enter: AX HF

6.18. 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	**NO ENCODER**	**ENCODER**
AX - AT	Not valid	Immediate
AU – AS	Not valid	
AA,AM	Not valid	
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 PC48 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 PC48 home input for the appropriate axis. The active level of this input may be controlled by the HH and HL commands.

QUEUE REQUIREMENTS		
MODE	**NO ENCODER**	**ENCODER**
AX - AT	Not valid	Immediate
AU – AS	Not valid	
AA,AM	Not valid	
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

6.19. 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	**NO ENCODER**	**ENCODER**
AX – AT	Not valid	Immediate
AU – AS	Not valid	
AA,AM	Not valid	
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	**NO ENCODER**	**ENCODER**
AX – AT	Not valid	Immediate
AU – AS	Not valid	
AA,AM	Not valid	
AA/CD	Not valid	

Example: Examine the current encoder position of the Y axis.

Enter: AY RE

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

6.20. 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	
AX – AS	Not valid
AA,AM	Immediate
AA/CD	Not valid

Example: (see MV command on page 6-68)

MM MOVE MINUS

The MM command sets the direction logic to move in the negative direction.

QUEUE REQUIREMENTS	
MODE	
AX – AS	Not valid
AA,AM	Immediate
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	**NO ENCODER**	**ENCODER**
AX – AS	4	5
AA,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.

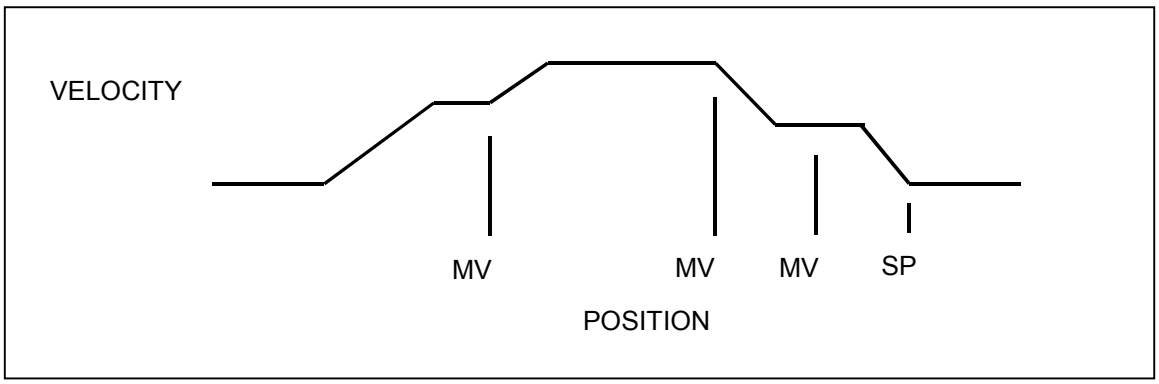


FIGURE 6-1 VELOCITY STAIRCASE PROFILE

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	**NO ENCODER**	**ENCODER**
AX – AS	3	4
AA,AM	Not valid	
AA/CD	Not valid	

Example: (see MV command on page 6-68)

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.

QUEUE REQUIREMENTS		
MODE	**NO ENCODER**	**ENCODER**
AX – AS	Flush + 4	Flush + 4
AA,AM	Not valid	
AA/CD	Not valid	

Example: Force axis to stop at 25,000.

Enter: FP25000

6.21. CONSTANT VELOCITY CONTOURING

The PC48 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	
AX – AS	1
AA,AM	1
AA/CD	2

Example: (see CD command on page 6-72)

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	
AX – AS	1
AA,AM	1
AA/CD	2

Example: (see CD command page 6-72)

BL# BIT LOW

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

QUEUE REQUIREMENTS	
MODE	
AX – AS	1
AA,AM	1
AA/CD	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	
AX – AS	1
AA,AM	1
AA/CD	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
```

Note: See Section 2.2 JUMPERS on page 2-1

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. The distance required may be calculated from the equations in Section 1. 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	
AX – AS	Not valid
AA,AM	0
AA/CD	Not valid

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	
AX – AS	Not valid
AA,AM	Not valid
AA/CD	1

Example: (see CD command on page 6-72)

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	
AX – AS	Not valid
AA,AM	Not valid
AA/CD	1

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	
AX – AS	Not valid
AA,AM	Not valid
AA/CD	8

Example: (see CD command on page 6-72)

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	
AX – AS	Not valid
AA,AM	Immediate
AA/CD	Not valid

Example: (see CD command on page 6-72)

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	
AX – AS	Not valid
AA,AM	6
AA/CD	Not valid

Example: (see CD command on page 6-72)

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	**LINEAR**	**PARABOLIC**	**COSINE**
AX – AS	Not valid		
AA,AM	6	30	30
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
MT20000,2 0000
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	
AX – AS	Not valid
AA,AM	6
AA/CD	Not valid

Example: Examine contour queue size.

Enter: AA CD0,0; RQ

Response: <LF><CR>1016<LF><CR>

7. HOST SOFTWARE

7.1. INTRODUCTION

The support software disk for Oregon Micro Systems PC family boards is supplied with the initial purchase of an OMS PC family board. This disk contains several programs which allow the user to interact with the OMS PC family of motion control boards so they can become familiar with their instruction sets. The disk contains a Programs section and a section with driver and DLL files for particular operating systems.

After a PC48 board is installed as described in Section 2 (Getting Started), insert the support disk into the PC. Run the communication program (PC4XCOMM) from the disk. You can now type OMS commands on the keyboard and they will be sent to the board. Their responses will be shown on the screen. Type EN WY and you will enable the board's echo and the board will display its board type and firmware version number.

7.2. PROGRAM FILES

This section contains example code for different software languages. Portions of the programs on the demo disk may be adapted by OMS motion controller users for use in application programs using OMS motion controls. No license is required.

All C language programs in this directory were compiled using Microsoft C 6.00A and were verified to compile correctly using Borland C++ 3.0.

The programs use the factory default I/O address of 300 HEX. The "C" versions use the factory default IRQ5 to communicate with the controller.

7.3. DRIVER SUPPORT SECTION

Each driver support section is organized and identified by the operating system its files are intended for. For example, driver support for WIN98 may be labeled OMSPC4xWIN9x. Note that the communication programs can be found here.

7.3.1. DEVICE DRIVER INSTALLATION

Reference the appropriate README file associated with the driver you are trying to install for installation instructions (i.e. README.TXT).

7.3.2. COMMAND LINE OPTIONS

There are several command line options to configure the device driver to the user's needs. They are as follows:

A:XXX where XXX is the hexadecimal I/O ADDRESS to which the Oregon Micro System board is jumpered. The driver will accept any address up to 3FF hex. It is up to the user to insure that there are no conflicts between the OMS board and any other board in the computer. The default is 300 hex.

Example: device=omsdrive.sys A:310 (driver expects the board to be set to I/O address 310 hex)

B:XXXX where XXXX is a hexadecimal number from 0010 to 7fff (16 to 32767 decimal). This value defines the "output to board" BUFFER size. This value determines how many characters you can send to the device buffer at one time. The default value is 0100 hex (256 decimal). When specifying a value, all 4 digits must be given, i.e. leading zeros must be included.

Example: device=omsdrive.sys B:0800 (buffer size set to 800 hex)

E:XX where XX is the hexadecimal value of the ASCII character you want to use to show the END OF STRING from the board (like a response to an RP command). If the value is less than 10, the leading 0 must be included.

Normally an OMS board surrounds each response it returns with an LF-CR pair. The response to an RP command when talking straight to the board is "<LF > <CR > 0 <LF > <CR >". This program strips off the LF and CR's and allows the user to define a character to show the end of the string. The default value is an LF. The response from an RP command using the driver would be "0 <LF >"

Example: device=omsdrive.sys E:0D (changes end of string char to a<CR >)

I:X where X is a number from 2-7,10-12, 14 or 15. This option sets the IRQ number to which the driver expects the board to be jumpered. It is up to the user to insure that there are no IRQ conflicts with any other board in the computer. The default value is 5.

Example: device=omsdrive.sys I:7 (device driver expects the board to be jumpered for IRQ7)

N:XXXXXXXX where XXXXXXXX is the new NAME you want to use for the device driver. The name must be padded on its right with spaces to take up 8 characters. It is a good idea to use a ; as the 9th character to insure that all 8 characters are processed by the driver correctly. The name becomes the file name you will use to communicate with the board. The default name is BOARD1.

Example: device=omsdrive.sys N:MYNAME ; (the new file (device) name you will use when talking to the driver is MYNAME)

S: This SOUND option causes the device driver to produce a "POCK" sound using the computer's speaker whenever a DONE, LIMIT, SLIP, or COMMAND ERROR interrupt occurs. You may want to enable this to help debug some commands that are not acting the way you expect or to let you know when each move is done.

Example: device=omsdrive.sys S:

All characters given as command line options are converted to upper case. The options can be given in any order and in any combination.

7.3.3. BOARD STATUS MESSAGE HANDLING

When a DONE, LIMIT, SLIP or COMMAND ERROR interrupt occur, the device driver processes the interrupt and inserts certain characters into the data stream from the board.

When a command error occurs it places a # symbol followed by the "E:" specified end character described above. A limit interrupt causes an @ character followed by the end character to be placed in the data stream, and an encoder slip interrupt will place a $ and the end character into the stream. A DONE interrupt will place an ! into the data stream followed by the byte read from the done flag register of the board, followed by the end character. If the board is returning a message when an interrupt occurs, the interrupt symbol will be delayed until after the end of the message string.

7.3.4. OTHER DEVICE DRIVER FEATURES

There are three special characters that are recognized by the device driver. These characters affect the way commands are sent to the board. They are the ! (WAIT FOR DONE), @ (WAIT FOR LIMIT) and * (COMMENT) characters.

When the driver encounters the ! (WAIT FOR DONE) symbol as it processes the characters in its ‘input to board’ buffer, it stops sending characters to the board until the board produces a DONE interrupt. This means that previous to sending the ! symbol to the board an ID command must be sent. When the DONE interrupt occurs after the driver processes the ! symbol, the DONE condition is cleared with a Control-Y. Characters are then allowed to be passed to the board.

The @ character acts in a similar manner except the driver holds off sending characters to the board until a limit interrupt occurs. When the board receives a limit signal from the platform, it flushes its command queue and any pending commands stored in the board's queue are lost. By using the 'wait for limit' character you can hold off sending commands to the controller until a limit is hit. This feature is useful when sending the platform to its home position.

The * character allows the user to place comments into a command file. All characters following the * up to the next control character (<LF>, <CR>,

<TAB>, et cetera) are ignored by the device driver and are not passed to the controller.

The following example shows a command file using the special characters listed above.

```
ax pl                 *lift pen

ax jg-5000 @          *force x axis to go into minus limit and wait until it does

ay jg-5000 @          *force y axis to go into minus limit and wait until it does

ax vl1000 hm0 id !    *reduce the speed of the x axis and send it home
                      *wait until it gets the DONE after the home

ay vl1000 hm0 id !    *reduce the speed of the y axis and send it home
                      *wait until it gets the DONE after the home

aa vl10000,10000; ma30000,30000; go id !
                      *increase the speed and send the platform to its center
                      *wait until the move is done

ax lp0 ay lp0         *call the center 0,0
```

7.3.5. INTERFACING TO THE DEVICE DRIVER

Using the device driver under various programming languages is similar to reading and writing to a disk file. It is best to create a separate file number or handle for reading from and writing to the device. The device is opened as you would open a disk file.

The simplest way to send data to the device is to create the command string desired, then send it to the device driver as a whole string. DOS and the driver take care of the task of splitting the string into characters and sending them to the board.

Keep in mind that when a file (and this device driver) are opened, DOS creates a small buffer so it will be more efficient when accessing the disk. To make sure all commands are processed, this buffer should be flushed each time the device is written to, otherwise they will remain in the DOS buffer until it is full and not be executed at the desired time. An alternative is to eliminate the buffer if your language allows it.

The best way to receive data from the device driver is one character at a time. This allows the program to analyze the data to detect any of the status characters that the driver might produce. The characters can then be joined into a string which can be converted into a numeric value for processing.

The 'input from board' buffer is 256 bytes long. It is a 'circular' buffer i. e. when it fills up, it wraps around to the start. The device driver does not check to see if the buffer is getting full. If you do not read from it after the board has put 256 characters of responses in it, good data will be over written. To get accurate responses, the user should only send one response producing command to the board at a time before reading the response.

When sending response producing commands to the board (an RP for example), there is a certain amount of lag time before the full response is available from the device driver. The faster the computer, the more evident the lag time is. The lag time can cause an 'OUT OF DATA' DOS error if it is not handled correctly.

The process of sending the RP command, for example, is something like this:

1) Program tells DOS to send "RP" string to the device
2) DOS sends "RP" to device driver
3) Device driver puts characters R and P into the 'output to board' buffer
4) Device driver enables board's interrupt
5) Board sends interrupt to processor
6) Processor jumps to interrupt handling routine
7) Routine gets the R from the buffer and sends it to the board
8) Routine gets the P from the buffer and sends it to the board
9) Processor returns from interrupt and computer goes about its business
10) Board recognizes the RP command and processes it
11) Board reads binary position values stored in registers on the board
12) Board formats those values into ASCII character decimal numbers
13) Board sends interrupt to processor
14) Processor jumps to interrupt handling routine
15) Routine sees that board has a response to get
16) Routine gets a character from board and places it into the input buffer
17) Routine repeats step 16 until there are no more characters to get
18) Processor returns from interrupt and computer goes about its business

The steps to receive the response, are much simpler:

1) Program asks DOS for a character
2) DOS asks the device driver for the character
3) Device driver gets next character from 'input from board' buffer
4) Driver sends it to DOS
5) DOS returns it to the program
6) Program puts the characters together to form the position value string
7) Program converts the string into a numeric value

The amount of time it takes the board to perform steps 10 through 13 can vary slightly depending on how busy the board is. If the program asks to get the response to the command before Step 16, the device driver will say that its buffer is empty and DOS will send an error message telling the program that the device is out of data. If the response has been partially received, the device driver will keep hold of the program flow until the response is completed.

If your programming language normally gets a run-time error when a file is out of data then error detection should be disabled during the time you are reading from the device. The BASIC, C and Pascal program examples on this disk show examples of routines to read responses from the device driver without getting run-time errors.

Routines that get the response should be loops that will continue to attempt to get a character from the device until the 'end of string' character is received, ignoring any out of data errors.

7.3.6. MULTIPLE BOARDS IN ONE COMPUTER

By jumpering the boards correctly and using the command line options listed above, it is possible to put more than one board into a computer. If the serial and parallel ports of the computer are not using the other IRQ you could put up to FOUR boards into one computer, allowing up to 32 axes of interrupt driven motion control in one computer.

To allow 4 device drivers, the config.sys command lines could look like this:

```
device=omsdrive.sys     (using default I/O address of 300 hex and IRQ5)
device=omsdrive.sys n:board2  ; a:304 i:7 (using I/O address 304 hex and IRQ7)
device=omsdrive.sys n:board3  ; a:308 i:4 (using I/O address 308 hex and IRQ4)
device=omsdrive.sys n:board4  ; a:30c i:3 (using I/O address 30c hex and IRQ3)
```

The controlling program would have to direct which commands went to which boards. Commands for board number 1 would be directed to device board1. Commands for board number 2 would be directed to device board2. Commands for board number 3 would be directed to device board3. Commands for board number 4 would be directed to device board4.

8. SERVICE

8.1. USER SERVICE

The PC48 family of controllers contain no user serviceable parts.

8.2. THEORY OF OPERATION

The 68332 microprocessor on the PC48 controllers maintains four concurrent processes. The highest priority process calculates the desired pulse frequency 2048 times each second with a proprietary algorithm (patent number 4,734,847). This frequency is fed to U82 and &84 which 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 by U32. Status of the interrupts and error flags may be read by the host. U34 compares the PC48 address to the I/O address selected by the host and enables the board decode logic when a match is detected.

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 pulses per second simultaneous on each axis

Acceleration

0 to 8,000,000 pulses per second per second

Position range

134,000,000 pulses (±67,000,000)

Accuracy

Position accuracy and repeatability ±0 counts for point to point moves

Velocity accuracy ±0.01% of peak velocity in jog mode

Environmental

Operating temperature range 0 to 50 degrees centigrade

Storage temperature range -20 to 85 degrees centigrade

Humidity 0 to 90% non-condensing

Power

+5 volts at 1.14 amps typical (from the ISA/AT backplane)

Dimensions

8 13/16 x 4 13/16 x 1/2 inches high

ISA/EISA interface

Meets all IBM I/O channel signal specifications and definitions

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 on-board jumper 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 under software control for each axis.

User definable I/O

Up to 22 bits of TTL user definable I/O bits. Factory default includes 8 open collector auxiliary outputs (one per axis), 8 general purpose inputs with 2.2k Ohm pull-up resistors and 6 general purpose outputs. All 14 general purpose I/O may be configured as inputs or outputs by a jumper block.

Step pulse output

Pulse width 50% duty cycle. Open collector TTL signal.

Direction output

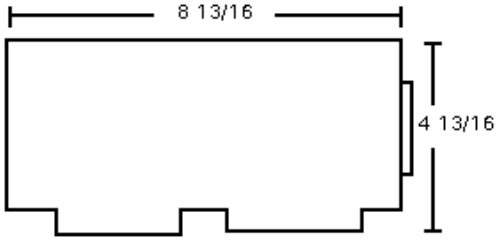
Same as step pulse output

ISA/AT Bus interrupt

The interrupt is user selectable from levels 2-7,10-11, 14 & 15. The interrupt is not required. The factory default is level 5.

ISA/AT Bus register interface

The I/O address block is user selectable. The factory default is 300 hex.



OMS PC48 FAMILY OF INTELLIGENT MOTION CONTROLS				
MODEL	STEPPER AXES		SERVO AXES	USER I/O
	Closed Loop (Encoder)	Open Loop		
PC48-2		2	N/A	16
PC48-4		4	N/A	18
PC48-6		6	N/A	20
PC48-8		8	N/A	22
PC48-2E	2		N/A	16
PC48-4E	4		N/A	18
PC48-6E	2	4	N/A	20

IO38 ADAPTER MODULE:

The IO38 is an adapter module for use with the PC48 to provide separate connectors to each driver axis and each encoder input. See the I/O Accessories table below for available model numbers.

IO38 SPECIFICATIONS:

Dimensions

7.5 x 3.0 x 0.78 inches tall

Power

No power required

CONNECTIONS TO IO38:

The following table defines the IO38 connections to the drivers. The mating connectors are 9 pin subminiature D, Amp part #747944-2 or equivalent.

CONNECTIONS TO DRIVER INDIVIDUAL CONNECTOR PER AXIS			
FUNCTION	**PINS**		**FUNCTION**
Ground	1	6	Ground
Step Pulse Output	2	7	Positive Limit Input
Direction Output	3	8	Negative Limit Input
Auxiliary Output	4	9	Home Input
+5 Volts	5		

The following table defines the IO38 connections to the encoder inputs on those models which include the encoder option. The mating connectors are 9 pin subminiature D, Amp part #747944-2 or equivalent.

CONNECTIONS TO ENCODER INDIVIDUAL CONNECTOR PER AXIS			
FUNCTION	**PINS**		**FUNCTION**
Ground	1	6	Ground
Index +	2	7	Phase B +
Phase A +	3	8	Index -
Phase A -	4	9	Phase B -
+5 Volts	5		

The following table defines the IO38 connections to the User Definable I/O. The mating connectors are 25 pin subminiature D, Amp part #747942-2 or equivalent.

IO38 CONNECTIONS TO USER DEFINABLE I/O			
FUNCTION	**PINS**		**FUNCTION**
Ground	1	14	Ground
I/O Bit 0	2	15	I/O Bit 1
I/O Bit 2	3	16	I/O Bit 3
+5 Volts	4	17	+5 Volts
I/O Bit 4	5	18	I/O Bit 5
I/O Bit 6	6	19	I/O Bit 7
Ground	7	20	+5 Volts
Ground	8	21	I/O Bit 9
I/O Bit 8	9	22	I/O Bit 11
I/O Bit 10	10	23	+5 Volts
+5 Volts	11	24	I/O Bit 13
I/O Bit 12	12	25	Ground
Ground	13		

CON38 CONNECTOR KIT:

The CON38 connector kit is a mating connector and hood for connecting the PC48 to driver modules, limit switches, etc. Cables are not included; however, they are available as an option. See the I/O Accessories table below.

CBL38 CABLE:

The CBL38 is an 80 wire shielded cable with a connector and hood on each end. It is available separately in various lengths or packaged with an IO38. See the I/O Accessories table below.

I/O Accessories	
IO38	I/O module-80pin for PC48 (without cable)
IO38-6	I/O module-80pin for PC48 supplied w/ 6 ft cable
IO38-12	I/O module-80pin for PC48 supplied w/ 12 ft cable
IO38-25	I/O module-80pin for PC48 supplied w/ 25 ft cable
CBL38-3	3 ft cable w/mating connector, 80pin(PC48)
CBL38-6	6 ft cable w/mating connector, 80pin(PC48)
CBL38-12	12 ft cable w/mating connector, 80pin(PC48)
CBL38-25	25 ft cable w/mating connector, 80pin(PC48)

INDEX

!

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W