USER'S MANUAL

INTELLIGENT MOTOR CONTROLLERS

VME58 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 http://www.OMSmotion.com

COPYRIGHT NOTICE

© 1993,1996, 1997 Oregon Micro Systems, Inc., A Pro-Dex Company ALL RIGHTS RESERVED

This document is copyrighted by Oregon Micro Systems, Inc. You may not reproduce, transmit, transcribe, store in a retrieval system, or translate into any language in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, any part of this publication without the express written permission of Oregon Micro Systems, Inc.

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. MD-DOS and Windows 3.1 are registered trademarks of Microsoft Corporation. QEMM and Manifest are registered trademarks of Quarterdeck Office Systems.

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.

3301-2300000
Revision C

TABLE OF CONTENTS

This page intentionally left blank

1. GENERAL DESCRIPTION

1.1. INTRODUCTION

The Oregon Micro Systems, Inc. (OMS) VME58 family of intelligent motion controls can manage up to 8 axes of servo or stepper motors in one VME 6U compatible I/O slot. Incremental encoder feedback is provided on all servo axes and is available as an option on stepper axes. The VME58 functions as a motion coprocessor within the host computer. It utilizes a 68332 microprocessor with DSP type instructions 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 VME58 calculates the optimum velocity profile to reach the desired destination in the minimum time while conforming to the programmed velocity and acceleration parameters. A PID digital filter is provided for all servo axes to allow the system gain and phase shift to be adjusted for system stability.

Commands may be sent to the VME58 by simple I/O commands using virtually any language on the host computer which has the ability to do I/O. It is easily programmed by writing ASCII character strings to a dual port RAM on the controller. Programming examples are shown in Section 5 of this manual.

1.2. FUNCTIONAL DESCRIPTION

The VME58, 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 velocity profile is calculated for each axis providing independent but synchronized (if desired) profiles. The VME58 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. Several moves may be chained together to provide a more complex pattern. The VME58 is able to store up to 256 input and 256 output characters in the dual port RAM plus 200 commands and parameters in separate command queues for each axis, allowing several moves to be made without host intervention.

1.3. VELOCITY PROFILES

The VME58 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

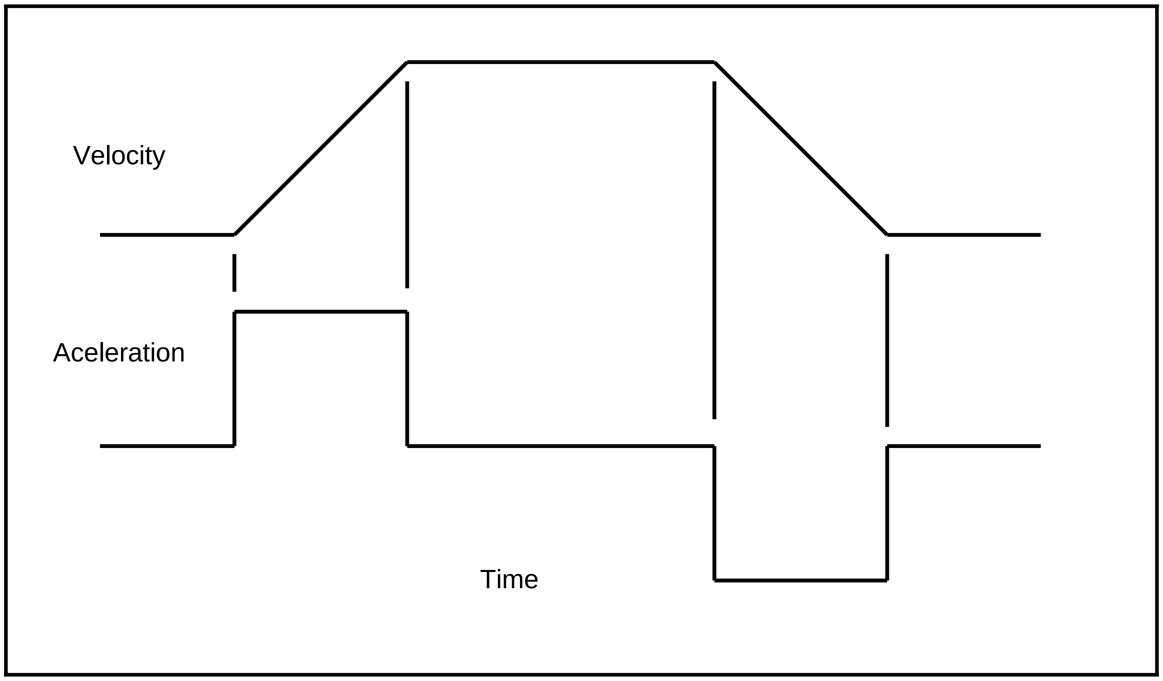

Figure 1-1 TYPICAL VELOCITY PROFILE

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 ensures 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

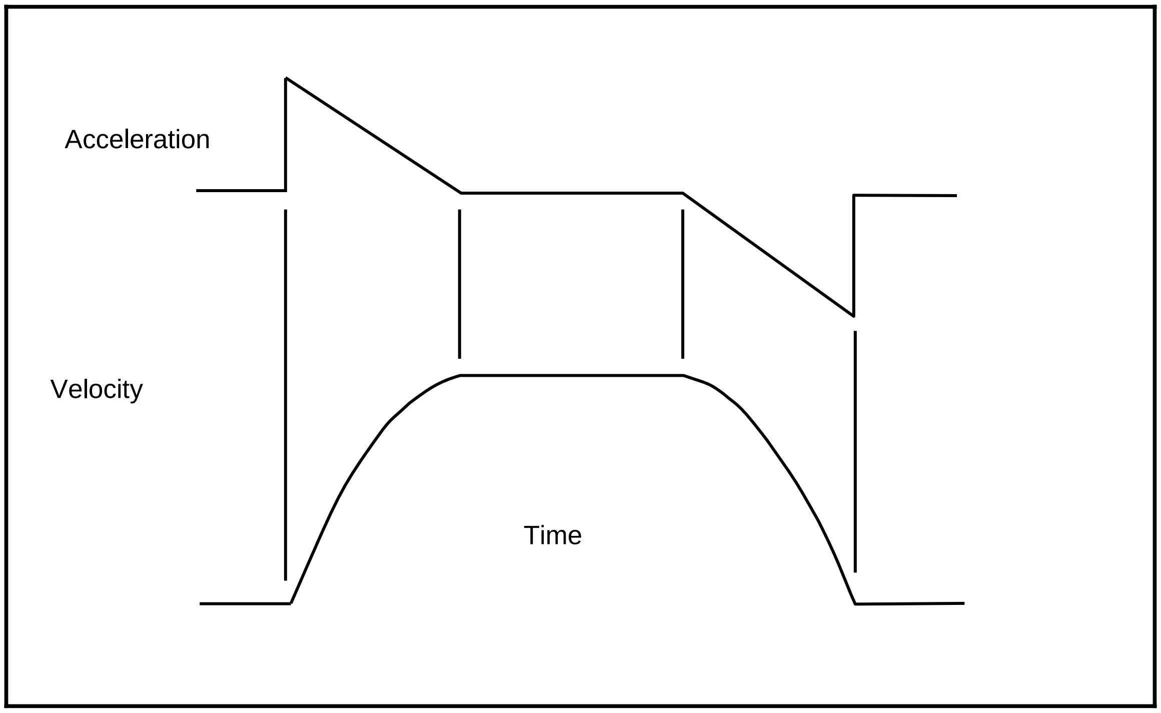

Figure 1-2 PARABOLIC VELOCITY PROFILE

parabolic ramp mode may result in reduced move time at high speeds, since a larger acceleration may be used.

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:

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

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:

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

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).

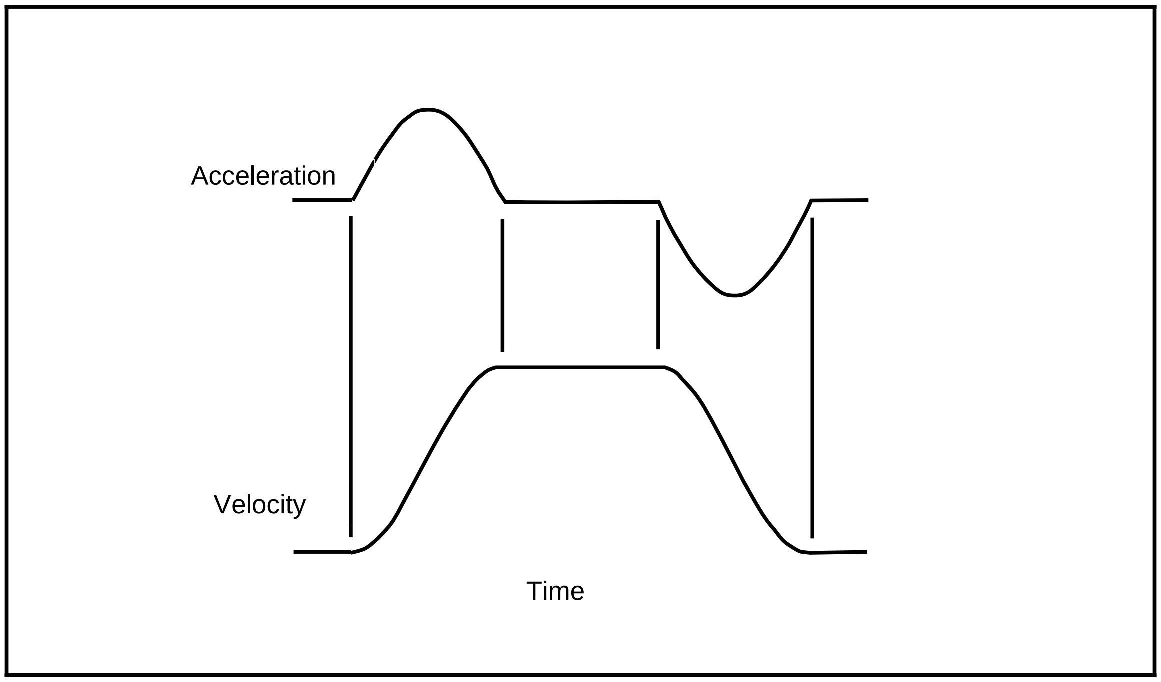

Figure 1-3 COSINE VELOCITY PROFILE

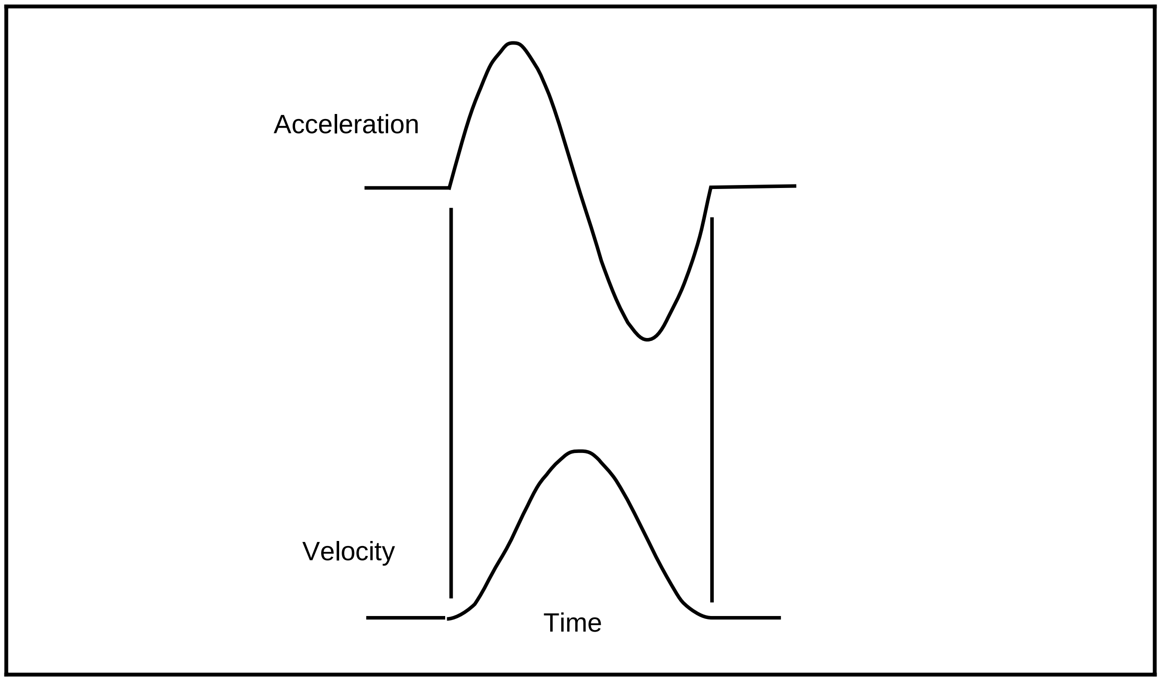

Figure 1-4 SHORT MOVE COSINE VELOCITY PROFILE

This page intentionally left blank

2. GETTING STARTED

2.1. INTRODUCTION

The VME58 board requires one full width slot in the VME card cage. In most cases the jumpers on the VME58 board will not have to be changed assuming there are no address or interrupt conflicts with existing boards. The factory default settings for the board have it using a block of 4K contiguous address from F000 to FFFF in the short address space. If these do not conflict with any previously installed hardware in your computer you will not need to change any jumpers on the VME58 board.

2.2. JUMPERS

There are twelve blocks of square pin jumpers on the VME58 board. These can be thought of as 4 logical groups: board address selection jumpers J61; interrupt selection jumpers J71 and J73; limit polarity jumper, J15; and I/O jumpers J16 & J26. See Figure 3-1 for the locations of the jumpers. Recommended jumpers are Amp part number 531220-2 or equivalent.

2.3. ADDRESS SELECTION

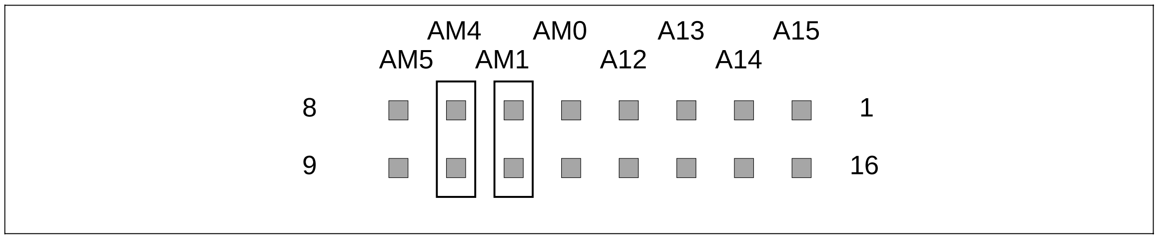

Figure 2-1 J61 ADDRESS SELECT 1 (default setting)

The VME58 uses a 4K byte block of short address memory. The starting address for this block is selected by square pin jumpers on the board. A jumper across a pair of pins indicates that the bit is a 0, without the jumper the bit is a 1. To decide on jumper settings, choose your starting address. Address lines A0 through A11 are decoded by the VME58 board and for base address selection are assumed to be 0. To set the board for the factory default address of F000, jumpers for address lines A15, A14, A13 and A12 on J61 selecting a 1 for those lines.

The VME58 allows selection from several address modifier values. Lines AM0, AM1, AM4 and AM5 are user selectable by square pin jumpers on J61. AM2 is always selected as low and AM3 is always selected as high. The hex code for the default address modifier setting is 29 hex. This allows Short Non-Privileged Access to the VME58 board.

To select this default value a jumper must be placed on lines AM1 and AM4 to decode them when they are low, no jumper should be on AM0 and AM5 so they will be decoded when high.

2.4. INTERRUPT SELECTION

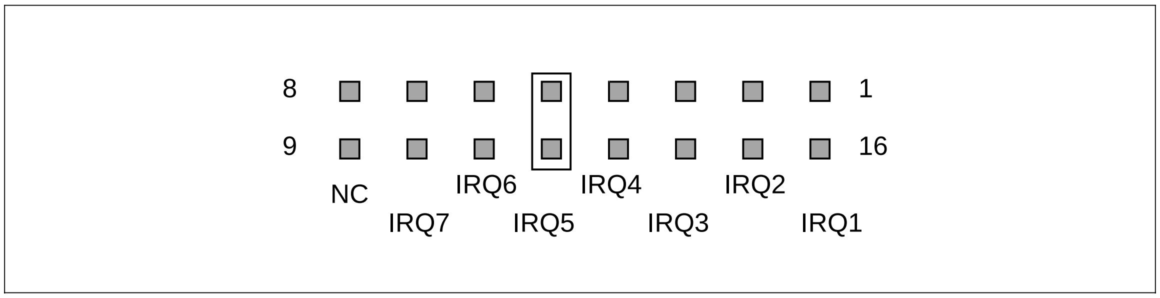

Figure 2-2 J71 INTERRUPT SELECT 1 (default setting)

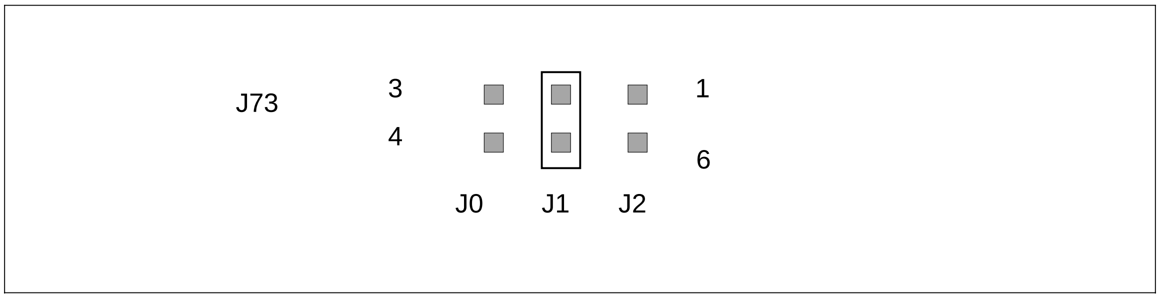

Figure 2-3 J73 INTERRUPT SELECT 2 (default setting)

Two sets of jumpers are used to select which VME bus interrupt signal is used by the VME58 board, J71 and J73. J71 selects to which interrupt request line the VME58 board is wired and J73 selects on-board logic to properly decode the interrupt acknowledge.

To jumper the board to a given interrupt, place a square pin jumper across the appropriate pin pair on J71, then using the binary equivalent of that interrupt number, place jumpers on J73, J0, J1 or J2 pin pairs, where the desired bit should be a 0.

For example, the factory default interrupt line is IRQ5. To jumper the board for this, place a square pin jumper across pins 5 and 12 of J71. Next, figure the binary equivalent of 5, which is 101. This means that (pin pair 1 and 6) of J73 should have no jumper since bit 2 of 101 is a 1, pin pair 2 and 5 should have a jumper since bit 1 is a 0, and pin pair 3 and 4 should have no jumper since bit 0 is a 1.

2.5. LIMIT POLARITY SELECTION

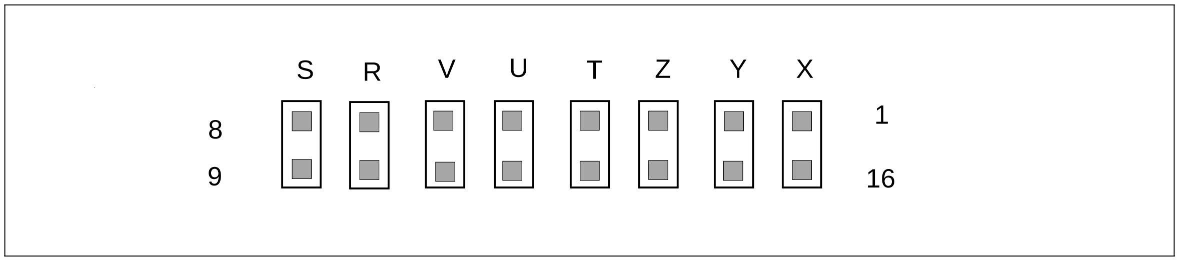

Figure 2-4 J15 LIMIT POLARITY

J15 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 switched to ground voltage level. With the jumper removed, the axis will stop if the limit line is switched to +5VDC. These inputs are internally pulled-up with a 2.2K Ohm resistor to +5VDC so the opening and closing of a switch to ground can control the limit input signals.

Factory defaults for J15 set the limit inputs to active low. This requires jumpers on all eight pairs of pins.

2.6. MOTOR CONTROL CONNECTORS

The motor control connector (J29) on the front panel of the board consists of a 100 pin connector. All motor control signals and I/O are available at J29. The motor control connector (P2) on the VME58 board consists of three rows of pins labeled A, B and C. Each row has 32 pins for a total of 96 pins. The A and C rows contain the motor control lines. I/O signals are not available at P2.

The motor control lines can be considered as 8 logical sets of 9 pins. Each set is used for an individual axis. The 9 pins of an axis set are: Axes Output , Auxiliary Output, Direction Output, Negative Limit Switch Input, Home Switch Input, Positive Limit Switch Input, encoder phase A, encoder phase B and index. Digital ground, analog ground and +5VDC are provided for each pair of axes. The auxiliary outputs and the analog ground are not available on the P2 connector.

See Section 4 for a detailed description of the connector.

2.7. USER I/O CONFIGURATION

J16 connects I/O bits 0 through 7 to pull-up resistors when they are configured as inputs. The jumpers should be removed when I/O bits 0 through 7 are configured as outputs. J26 is near to J16 and selects the configuration of the I/O bits as inputs or outputs. A jumper on pins 1 and 6 of J26 selects I/O bits 0 through 3 as inputs. U18 must be a 7402 logic gate. A jumper on pins 2 and 5 of J26 selects I/O bits 4 through 7 as inputs. U17 must be a 7402 gate when these pins are configured as inputs. No jumper on pins 3 and 4 of J26 selects I/O bits 8 through 11 as outputs and U27 must be 7408 gate. I/O bits 12 and 13

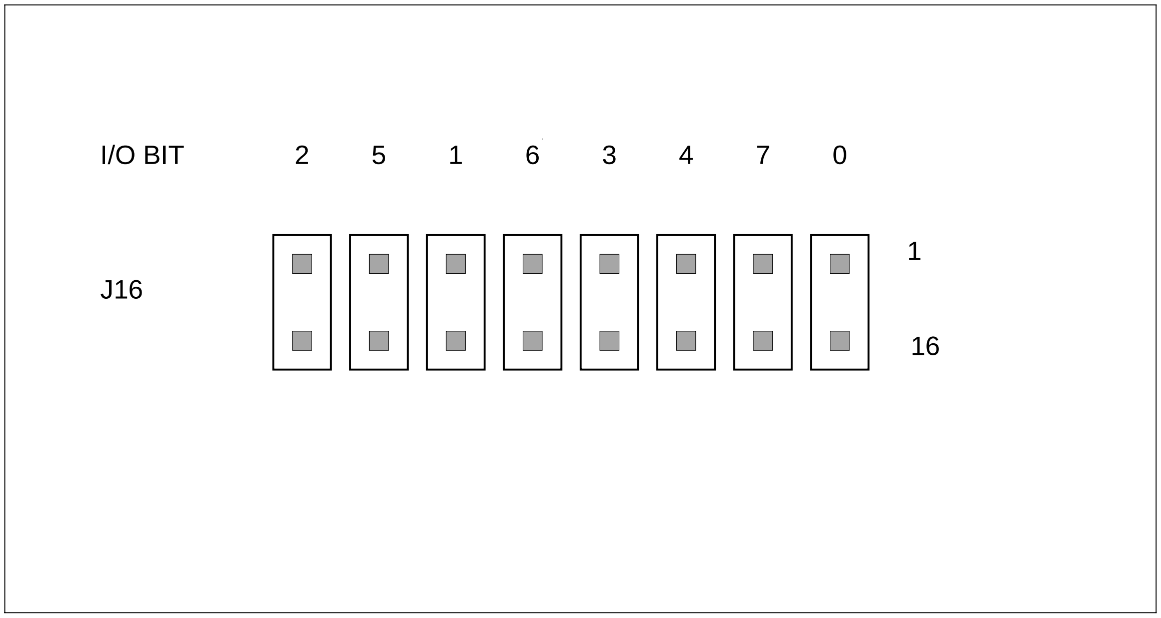

Figure 2-5 USER I/O PULL-UP JUMPERS (default setting)

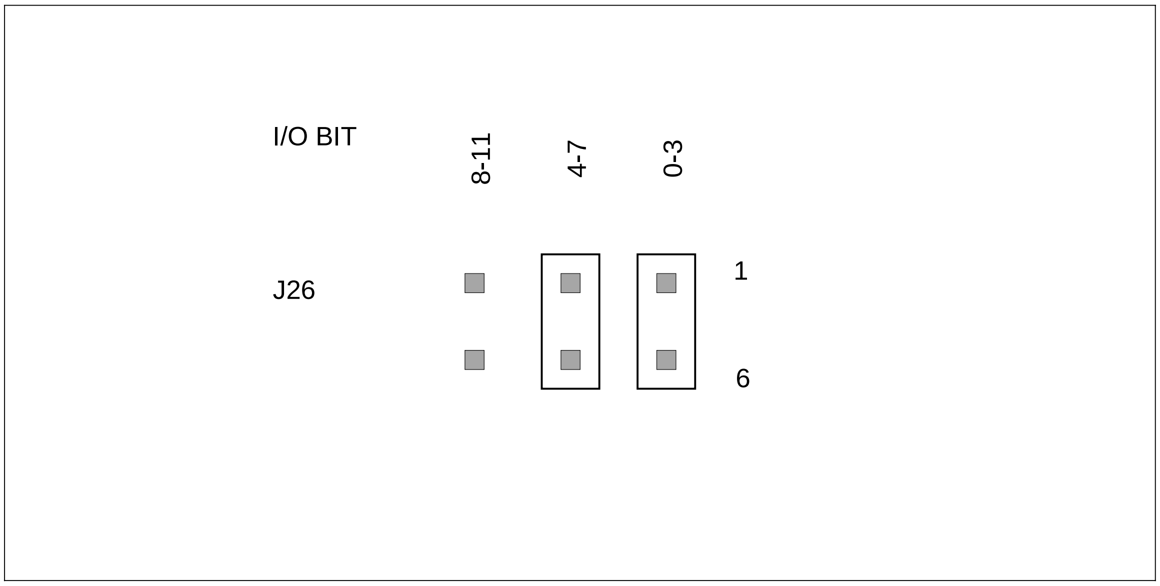

Figure 2-6 I/O CONFIGURATION JUMPERS (default setting)

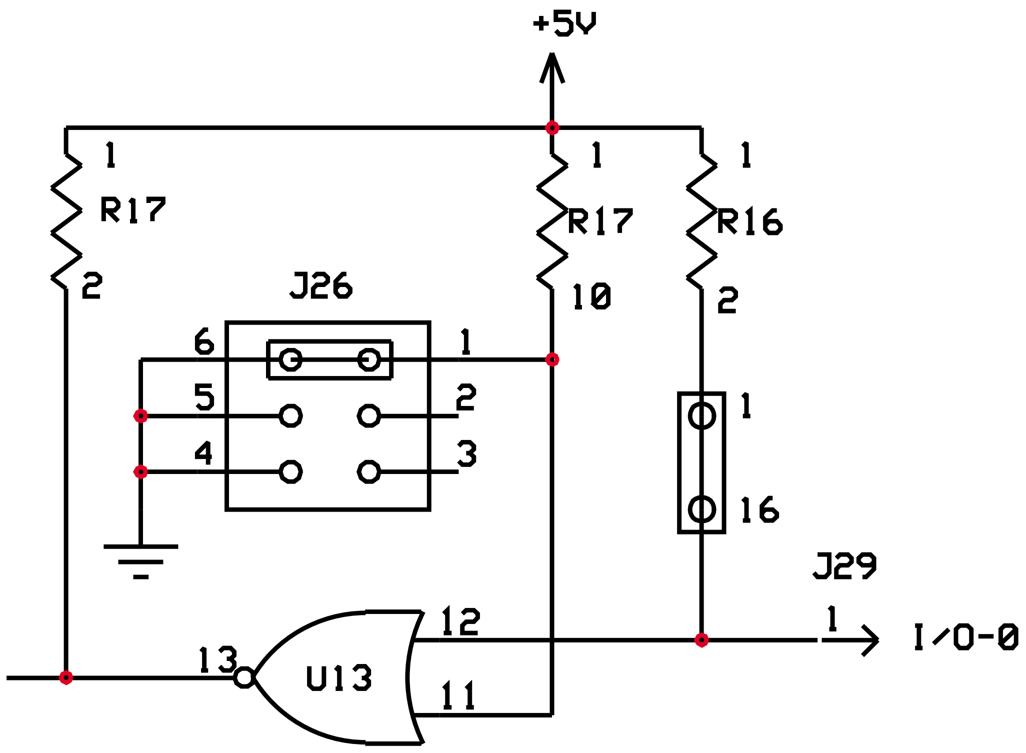

Figure 2-7 USER I/O INPUT CONFIGURATION

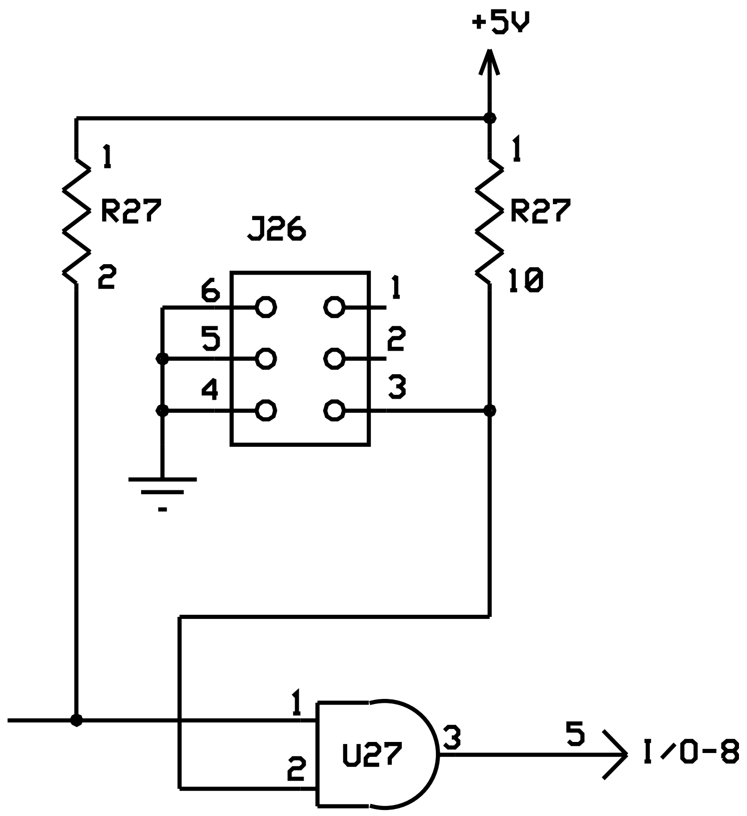

Figure 2-8 USER I/O OUTPUT CONFIGURATION

are fixed as outputs therefor U28 will always be a 7408. Figure 2-7 shows the required configuration for an input bit, while Figure 2-8 shows the configuration for an output bit. The default configuration is I/O bits 0-7 set as inputs and I/O bits 8-13 set as outputs.

2.8. OUTPUT SIGNAL OPTIONS

There are three hardware options available for the output signal to the power driver, i.e. analog output, pulse width modulated (PWM) output and step pulse output. Jumper block J58 selects the digital output (DIG) type either PWM or step pulse output for the X, Y, Z and T axes. J21 selects the digital output selected by J58 or the analog output for the X. Y, Z and T axes. J52 selects the digital output for the U, V, R and S axes while J41 selects the digital or analog output for these same axes. Note that the step output is only available on axes that have been factory configured for step motor use, while the PWM and analog

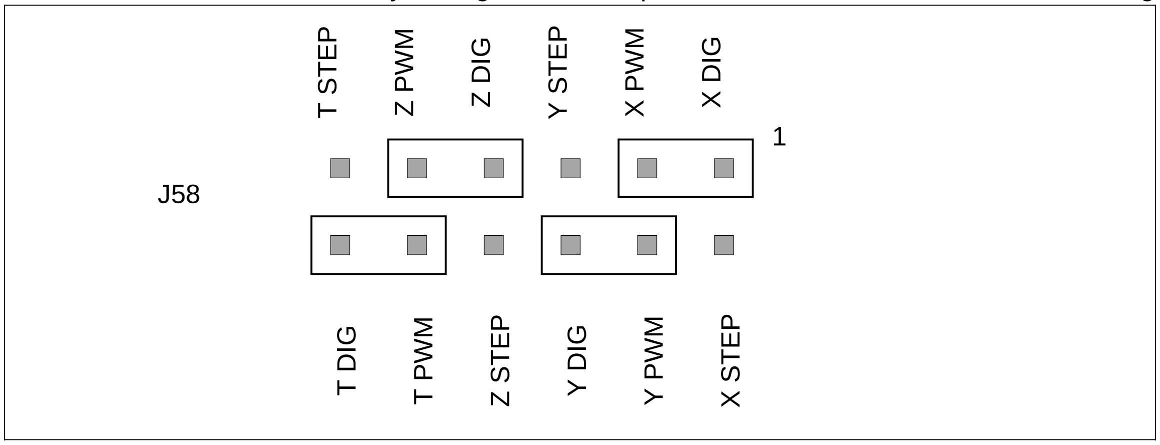

Figure 2-9 Digital Output Type Selection

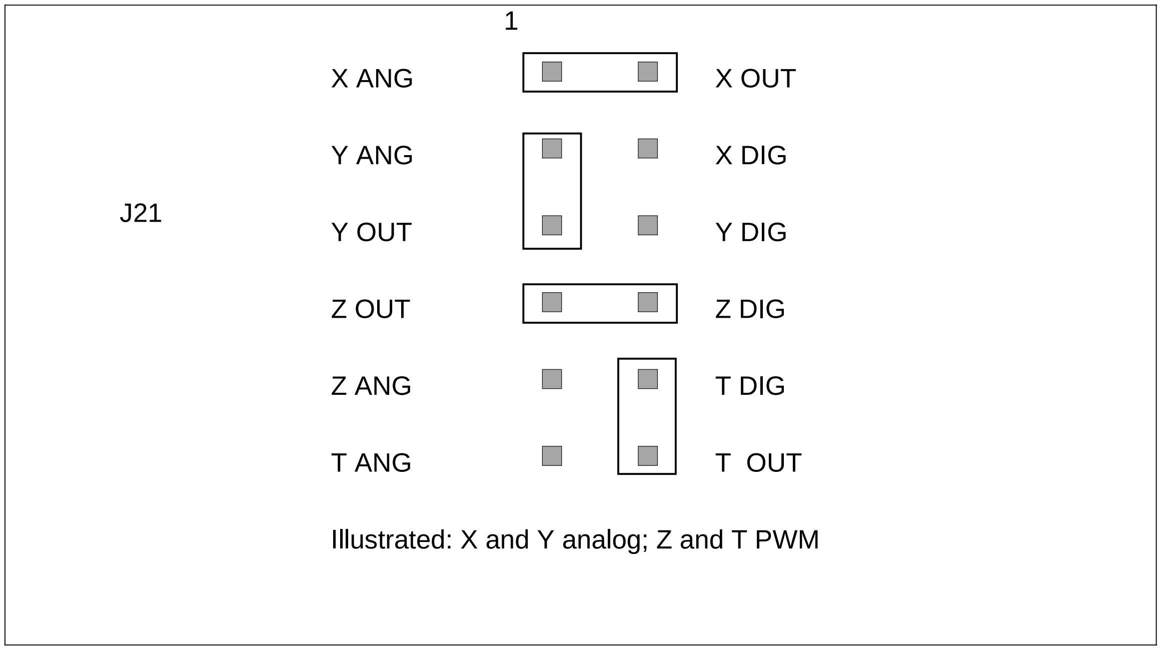

Figure 2-10 Digital or Analog Type Selection

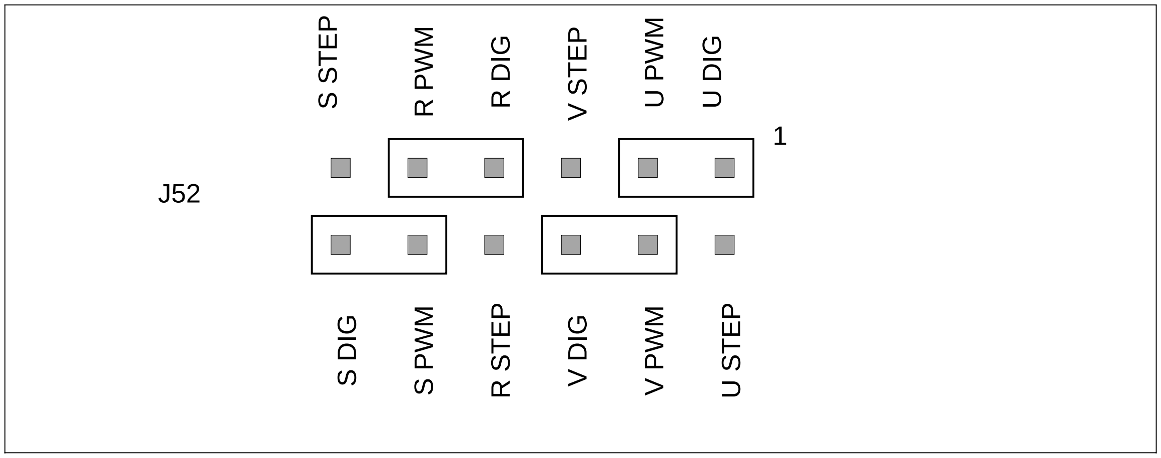

Figure 2-11 Digital Output Type Selection

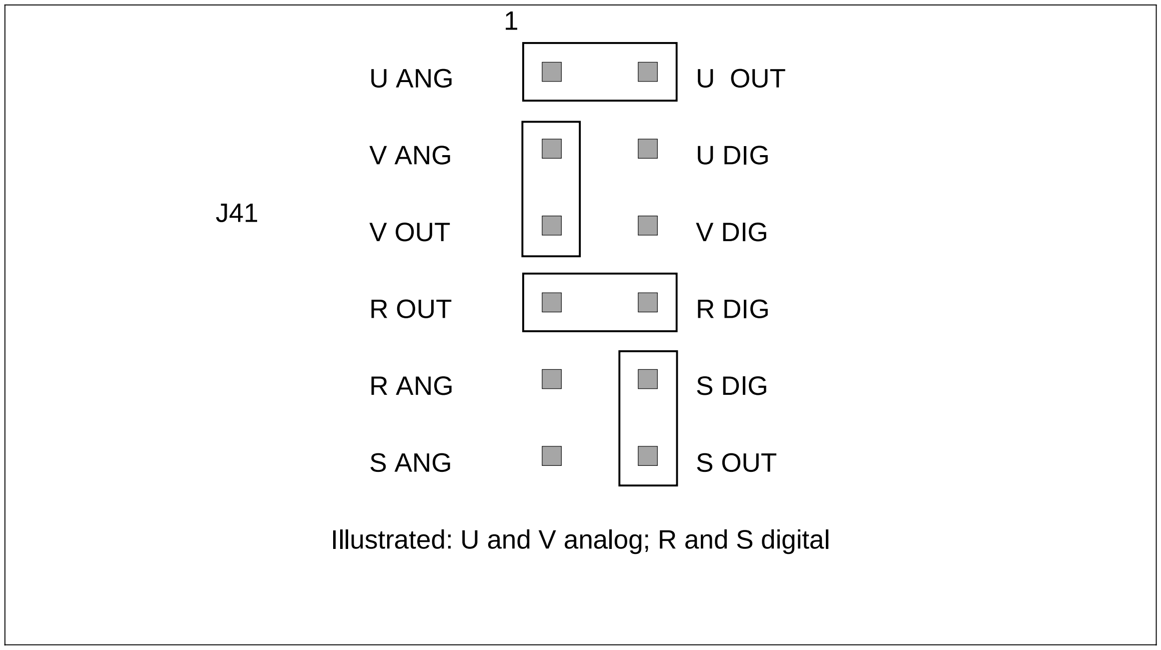

Figure 2-12 Digital or Analog Type Selection

output are available on axes that have been factory configured for servo motor use. The default for all servo axes is to be configured as analog outputs, unless ordered as a special product.

J62 allows the user to tie the analog ground to the computer digital ground. No jumper leaves the grounds isolated. The default is for this jumper to be installed connecting the analog to the digital ground. Removal of this jumper is rare.

2.9. HARDWARE INSTALLATION

1. Turn off power to your computer and disconnect its power cord.

2. Remove the computer’s cover.

CAUTION

To prevent possible damage to the VME58 and/or the system chassis, the VME58 should never be installed into a system utilizing metal or conductive guide rails. Though most VME chassis utilize plastic guide rails, some have metal guide rails or grounding strips in the guide rails. Revision D of the VME58 PCB design has a filter capacitor (C88) located at the edge of the board where +5VDC is exposed and may be shorted to conductive surfaces when installed into the VME chassis. Only nonconductive guide rails may be used with the VME58 controller family. Contact OMS technical support with any questions on this issue.

3. Choose an empty expansion slot in the VME rack and backplane and remove its associated metal cover from the back of the computer. Be sure to save the screw.
4. Check the VME58 board’s jumpers for proper configuration.
5. Slide the VME58 board into the rack and connector, ensuring the board is lined up correctly in the card guides and in the connector.
6. Double check the board to ensure it is properly seated in the connector.
7. Screw the front panel of the VME58 board into the card cage.
8. Replace the cover of the computer.
9. Replace the power cord and turn the computer on. (Do not connect the VME58 to other parts of the system until communication is established with the host for ease in trouble-shooting.)
10. Allow the computer to boot up.
11. To ensure that the VME58 is set up for the proper address, try to read and write a byte to the VME58’s interrupt vector register at short address FFF1 (default). See if the value written to it is the same one read back.
12. Using your favorite debug utility, output the following words to the short address space:

F804=0057	ASCII “W”
F806=0059	ASCII “Y”

13. Update the Output Put Index by adding 2 to the existing value at short address F800. This value is 0 after reset. This tells the VME58 that 2 new characters have been writen to the buffer. No buffer wrap around testing is

required for this installation check out, but is required for any real application. The VME58 will respond to the “WY” command by outputting its board name and firmware revision to the input buffer. To read the input buffer, first read the value of the Input Get Index at short address F802. This value is 0 after reset. The Input Get Index is an offset, in HEX, to the Input Buffer which has its base located at short address F004. If you read sequential words starting at address F004, you should get the hex equivalent to ASCII string <LF><CR>VME58 ver 2.11-8S<LF><CR>, which are:0A 0D 56 4D 45 35 38 20 76 65 72 20 32 2E 31 31 2D 38 53 0A 0D. A real application would then increment the Input Get Index by the number of words that have been read. This last action tells the VME58 this portion of the Input Buffer may now be reused, but is not required for installation checkout.

14. Connect the motor drivers to P2 or J29 (see Section 4).

15. Send motion control commands to the VME58 board and see that the motors move. If not, double check your wiring, ensuring that everything is properly connected.

2.10. PID FILTER CONTROL COMMANDS

OMS PC58 and VME58 series motion control boards with servo control axes use an enhanced PID filter algorithm to compute the output signal that drives the servo amplifier. The input to the filter algorithm is the position error , ER, which is computed from the difference between the command position and the encoder position. The formula for calculating the output is:

$$OUTPUT = KP*ER + KI*ES + KD*ED + KV*Vel + KA*Ac + KO$$

where:

ER = position error
ED = difference between current position error and the previous position error
ES = summation of position error over the interval set by KN
Vel = command velocity
Ac = command acceleration

*The coefficients are defined in the Command Structure section (see 5.14.).

2.10.1. SERVO SYSTEM CONNECTION AND CHECKOUT

Servo systems tend not to respond gracefully when connection errors are made. By following a step by step procedure, connection errors can be reduced. The basic elements of the connection and checkout procedure are:

1. Encoder connection and checkout
2. Motor/Amplifier connection and checkout
3. System tuning

<u>CAUTION:</u>

Do not connect the motor shaft to the mechanical system until all electrical connections have been verified, and the control system checked out as outlined below.

2.10.2. ENCODER CONNECTION AND CHECKOUT

Turn system power off!

b. Connect the encoder outputs as described in the Driver Interface section of the Users Manual. Power for the encoder is available from the OMS board.

c. Apply power to the computer system.

CAUTION:

Do not apply power to the servo amplifier or motor until this step is complete.

d. Send the command string LP0 RE. The board should respond with an encoder position of 0.

e. Manually turn the motor/encoder shaft 1 revolution. Send the command RE and verify that the encoder position changes with shaft movement in both directions and that 1 revolution corresponds to 4 times the number of encoder lines.

f. Repeat this process for all axes.

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

CAUTION:

Do not proceed until encoder connection is checked out and operating correctly.

2.10.3. MOTOR/AMPLIFIER CONNECTION AND CHECKOUT

CAUTION:

Do not connect the motor shaft to the mechanical system until all connections have been verified, and the control system checked out as outlined below.

This section applies to bipolar analog input servo amplifiers only.

a. Turn system power off!

b. Connect the axis output of the OMS board to the control input of the servoamplifier as described in the Driver Interface section of the OMS Users Manual and the servo amplifier users manual. Verify that the OMS board is configured for analog operation.

c. Turn on the computer system. Do not apply power to the servo amplifier/motor yet.

d. With a voltmeter, verify that there is 0 VDC between control ground and the analog control signal at the servo amplifier.

e. Apply power to the amplifier/motor. The motor should not turn! If it does move, turn off the amplifier/motor power and check the connections.

f. Select single axis mode for the axis under test(AX, etc.). Enter the command string KP.1 KD.3 KI0 LP0 HN. The motor should not turn!

g. This step verifies that the motor/encoder phasing is correct. Repeatedly enter the command string KO.1 RE incrementing the value of KO by .1 until the RE response is not zero. If the RE response is positive, motor/encoder phasing is correct. If the RE response is negative, turn the motor/amplifier and computer system power off and reverse encoder phase A and phase B. After reversing encoder phases, repeat the motor/encoder phasing test.

h. Send the command string LP0 KK1 VL10000 AC50000 HN. The motor should not move. Verify proper servo control operation by sending the command string MR20000 GO. Verify that the motor moves 20000 counts in the positive direction.

i. Exercise the motor using various move commands in both the positive and negative direction and verify correct motion with the RE command.

j. Turn the motor/amplifier power off and connect the motor the rest of the mechanical system. The system is now ready for tuning.

2.10.4. SERVO SYSTEM TUNING

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

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

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

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

An empirical trial and error approach will be discussed first. A good place to begin is with a low proportional gain value, KP. In most systems, very good performance can be achieved with the derivative gain, KD, set at 3 times KP. The use of the KK command sets both KP and KD at this ratio. The command string for the X axis would look like:

Enter:	AX	Go to X axis
	KK1	Set PID Parameters

Hold must be turned on for proper servo operation. Hold deadband should also be set to the accuracy required. Too low of a deadband value may cause continuous “seeking” by the servo system. Set the velocity profile and initial position to the desired values, set the deadband, and activate servo control with HN:

```
Enter:      AX VL50000 AC500000 LP0

            HN
```

Execute a step move with the MR command. Choose a distance that is appropriate for the mechanical system:

```
Enter:      MR20000 GO
```

The performance of the system may be evaluated by observation and measurement. If the response to the step move was not “well behaved”, reduce the gain until smooth response is achieved. Measure the position accuracy by looking at the actual position vs. the programmed position using the commands:

Enter:	RP	Returned Programmed Position
	RE	Returned Encoder Actual Position

Slowly increase the gain via the filter parameter commands until accuracy tolerances are met or signs of instability appear, then reduce it slightly until the system is stable.

Other system performance attributes may be determined by measuring the analog output signal to the motor drive. An oscilloscope with storage capability would be a good choice for this measurement. Although this signal represents system error and not position, it is useful for stability and saturation measurements. First observe the magnitude of the analog signal and verify that it is not pushing the motor driver into current limit or saturation. Either increasing the available motor driver current or reducing the acceleration in the velocity profile will correct saturation problems. The shape of the step response indicates the stability of the system. Gain can be increased to improve system “stiffness” but is limited by stability. As gain is increased, “ringing” will begin to appear on the step response. Gain should be kept below values that produce unacceptable “ringing” and overshoot. Additional performance may be achieved by fine tuning the filter parameters. In general, stability will be improved with higher values of KD at the expense of rise time. For velocity controlled servos (voltage mode servo amplifiers), KV may be added to reduce velocity following errors. For torque controlled servos, KA may be added to speed up the step response.

3. VME BUS INTERFACE

3.1. VME BUS

The VME bus specification allows for a number of different interface complexity options. The VME58 supports the D08(O) and D16 short address in either supervisory or non-privileged mode as specified by the VME specification C.1. Refer to the VME Bus specifications for signal descriptions. The base address is jumper selectable to allow positioning the board on 4K boundaries in the short address space. VME interrupts are supported at any level. Table 3-1 describes the VME bus interface.

The VME bus P2 connector row B is used to decode additional address bits for 32 bit data transfers and extended addressing in the VME bus specification.

3.1.1. DATA BUS

The data bus is a 32 bit, bidirectional, 3-state bus. Direction of data is controlled by the VME bus master. The data bus uses high-level active logic. The VME58 supports both 8 bit odd address and 16 bit transfers.

3.1.2. ADDRESS BUS

The address bus is a 32-bit high-level active bus. This bus is always driven by the VME bus master. The address bus provides the 32 address lines for decoding memory. I/O is memory mapped on the VME bus. The direction of transfer is determined by the state of the write* line which is driven by the current bus master.

3.1.3. CONTROL LINES

The control lines provide the signals for fundamental memory (or I/O operations). They control the size and direction of transfers.

3.1.4. SYSRESET

The Sysreset is a reset driver which is provided on the bus. The VME58 has an on-board reset timer and thus uses this signal only to initiate this timer on power up or during a system reset.

Table 3-1 VME BUS P1 PIN LIST

PIN	ROW C	ROW B	ROW A
1	D08	BBSY*	D00
2	D09	BCLR*	D01
3	D10	ACFAIL*	D02
4	D11	BG0IN*	D03
5	D12	BG0OUT*	D04
6	D13	BG1IN*	D05
7	D14	BG1OUT*	D06
8	D15	BG2IN*	D07
9	GROUND	BG2OUT*	GROUND
10	SYSFAIL*	BG3IN*	SYSCLK
11	BERR*	BG3OUT*	GROUND
12	SYSRESET*	BR0*	DS1*
13	LWORD*	BR1*	DS0*
14	AM5	BR2*	WRITE*
15	A23	BR3*	GROUND
16	A22	AM0	DTACK*
17	A21	AM1	GROUND
18	A20	AM2	AS*
19	S19	AM3	GROUND
20	A18	GROUND	IACK*
21	A17	SERCLK	IACKIN*
22	A16	SERDAT	IACKOUT*
23	A15	GROUND	AM4
24	A14	IRQ7*	A07
25	A13	IRQ6*	A06
26	A12	IRQ5*	A05
27	A11	IRQ4*	A04
28	A10	IRQ3*	A03
29	A09	IRQ2*	A02
30	A08	IRQ1*	A01
31	+12VDC	+5VDC STDBY	-12VDC
32	+5VDC	+5VDC	+5VDC

* low-level active indicator

3.1.5. INTERFACE THEORY OF OPERATION

The VME58 occupies a 4k block of memory designed to exist in the short address space. J61 is a jumper block used to select the address modifiers for the short address space and the 4k block within that address space.

[Note that only address bits below A16 are decoded. If the VME58 is mapped into an address modifier space other than the short address space the VME58 will alias at the higher address within that block of memory.]

After power is stable and configuration is complete the VME58 writes a one to the INIT bit of the status register. The host would read the status register at offset address 0FE3 and check this bit before sending any other information to the VME58. The next thing the host should do is send a WY (Who are You) to the VME58 through the data port to confirm proper configuration and that the communication link is established. This procedure is explained in Section 3.6. The VME58 will respond to the WY by placing an identification string of ASCII characters into the Input Buffer starting at offset 0004 and updating the Input Put Index at offset 0000. The string should look something like this:

<LF>< CR> VME58 ver2.20-4S4 <LF>< CR>

<LF> = Line Feed = hex 0A
<CR> = Carriage Return = hex 0D

There are nine 8 bit registers outside the Dual Port memory that are identified in Table 3-2 and explained in Section 3.4.

3.2. BOARD ADDRESS SELECTION

The VME58 occupies a block of 4K contiguous addresses. The factory default address is F000 through FFFF hex in the short address space. Refer to Figure 3-1 for configuration of jumpers.[1]

The actual address is chosen by jumpers on J61. Connecting a jumper selects a binary 0 for that address bit, while no jumper selects a binary 1. The factory default base address of F000 (hex) is shown in Section 2.

3.3. USING INTERRUPTS

Full interrupt capability is provided in accordance with the VME specification. Interrupts for input buffer full, transmit buffer empty, overtravel fault and operation complete are provided. Interrupt levels 1 through 7 are jumper selectable. Polled operation is also supported with separate status bits for each of the above sources. Table 3-3 shows the detail of the level select jumpers.

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 use Molex Corp. part number 90050-0007 or Amp 531220-2.

Table 3-2 I/O REGISTER DESCRIPTION

ADDRESS OFFSET	DESCRIPTION	FACTORY DEFAULT	FUNCTION
0FE1	Control Register	FFE1	Read/Write
0FE3	Status Register	FFE3	Read
0FE5	User Definable I/O (0-7)	FFE5	Read
0FE7	Slip Flags	FFE7	Read
0FE9	Done Flags	FFE9	Read
0FEB	User Definable I/O (8-13)	FFEB	Read
0FED	Limit Switch Status	FFED	Read
0FEF	Home Switch Status	FFEF	Read
0FF1	Interrupt Vector	FFF1	Read/Write

3.4. VME58 REGISTERS

The VME58 occupies 4K contiguous addresses in memory space. The registers associated with each address are described in the following sections (see also Tables 3-2 and 3-12).

Table 3-3 INTERRUPT LEVEL SELECTION JUMPERS

LEVEL	J71 PINS
IRQ1	1-16
IRQ2	2-15
IRQ3	3-14
IRQ4	4-13
IRQ5	5-12
IRQ6	6-11
IRQ7	7-10

3.4.1. CONTROL REGISTER

The control register is a read/write register which allows control of the host interrupt requests such as the done interrupt and VME58 programmable interrupt request. These interrupts may be enabled by writing a 1 to the appropriate bit or disabled by writing a 0 at any time. Interrupt levels 1 through 7 are supported and may be selected by J71. See Section 2.2. for more information. Bit 1 is used to request an update to the registers in the dual-port memory starting at address offset 1024(hex 400) through 1952(hex 7A0). Bit 5 is used as part of the mailbox functionality. The host would write a 1 to this bit to interrupt the VME58 to have it read its mailbox. Currently the only functionality supported by the mailbox is the Kill command. The host would write a 1 to bit 5 as an emergency stop situation. Reference section 3.7. Bit 7 is the enable for the general interrupt signal to the VME bus. The other bits selectively enable different interrupt conditions. This register is described in Table 3-4.

Table 3-4 CONTROL REGISTER DESCRIPTION

BIT	CONTROL DESCRIPTION
0	Data Area Update Request
1	Unused
2	Encoder Slip Interrupt Enable
3	Limit Register Interrupt Enable
4	Done Register Interrupt Enable
5	Interupt Request to the VME58
6	I/O bits 0 and 1 interrupt enable
7	Interrupt Request Enable

3.4.2. STATUS REGISTER

The status register is a read only register and indicates the status of the interrupt requests such as done or limit. These bits may be used as status indicators in a polled software environment or will interrupt the host processor if the interrupts have been enabled in the control register. A 1 at bit 0 will indicate a command error has occurred. This will generate an interrupt to the host if bit 7 of the control register is enabled. A read by the host to this register will reset this bit when the read is complete. When the VME58 is in a reset or configuration mode there will be a 0 at bit 1. Once the VME58 has completed configuration this bit becomes a 1. When encoder slip functionality is used (the ES# and IS commands) bit 2 will be set with a 1 when a slip on any axis has occurred. If bit 2 of the control register is set when a slip occurs an interrupt will be generated to the VME.

Bit 3 indicates the status of the logical OR of all the limit switches, unless the LF (limits off) command is used. Bit 3 of the control register will enable this bit to generate an interrupt to the VME. Bit 4 is the Done status and is a representation of the logical OR of all the bits in the Done Flag register. It will generate an interrupt to the VME if bit 4 in the control register is set and is reset by the completion of the host read. Bit 5 of both the status register and the control register are exact mirrors of each other and used in the mailbox functionality. The functionality of the mailbox is explained in Section 3.7. Bit 6 currently is not utilized and is reserved for future development. Bit 7 is the status of the interrupt signal to the VME and is the logical OR of bits 0,2,3,4, and bits 0 or 1 of the I/O register when the appropriate bit in the control register is set. When an interrupt occurs the host would read the status register to determine what caused the interrupt. If bit 6 of the control register is set and an interrupt occurs the status register may indicate that there is no error (no bits set), then the I/O register at offset 0FE5 should be read for the status of bits 0 and 1. (See Table 3-6)

Table 3-5 STATUS REGISTER DESCRIPTION

BIT	CONTROL DESCRIPTION
0	Command Error
1	Initialized (power up complete)
2	Encoder Slip
3	Overtravel Encountered
4	Done
5	Direct Interrupt Request Status to the VME58
6	Reserved/Unused
7	Interrupt Request Status

3.4.3. USER DEFINABLE I/O REGISTER (0-7)

The first 8 user I/O bits' logic states are reflected in this read-only register. It may be read at any time to determine the current states of bits 0 through 7, regardless of whether the bits are configured as inputs or outputs. (The RB command will return the I/O bit's configuration) The state of the bits may be changed with the BH and BL commands. See Section 5 for more information. Bits 0 and 1 are latched by an active low state and are cleared by an execution of the BX command or by reading the I/O register at address offset 0FE5. All other user inputs are unlatched. (See Table 3-6.) When bit 6 of the control register is enabled I/O bits 0 and 1 will generate an interrupt to the host when configured as inputs and triggered with a high to low signal. The interrupt is reset by the BX command or by reading the I/O register at offset 0FE5.

Table 3-6 USER I/O REGISTER DESCRIPTION (0-7)

BIT	DESCRIPTION
0	Bit 0
1	Bit 1
2	Bit 2
3	Bit 3
4	Bit 4
5	Bit 5
6	Bit 6
7	Bit 7

3.4.4. SLIP FLAG REGISTER

The slip flag register is a read only register from the VME58. The status bit indicating the slip status of each axis is defined in Table 3-7. These bits are written by the CPU on the VME58 when a slip is detected. A one indicates a slip. The host can then read it at any time to determine its status. All bits in the register are reset when the register is read. Refer to Section 5.16. for slip detection commands.

Table 3-7 SLIP FLAG REGISTER DESCRIPTION

BIT	DESCRIPTION
0	Status of X axis
1	Status of Y axis
2	Status of Z axis
3	Status of T axis
4	Status of U axis
5	Status of V axis
6	Status of R axis
7	Status of S axis

3.4.5. DONE FLAG REGISTER

The done flag register is a read only register from the VME58. The status bit indicating the done status of each axis is written by the CPU on the VME58 when a done command is executed (refer to Section 5.11.). The host can then read it at any time to determine the status of the motion process. It is cleared by reading the register. A one indicates a done. The bits are defined in Table 3-8.

Table 3-8 DONE FLAG 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

3.4.6. USER DEFINABLE I/O REGISTER (8-13)

The state of the user definable I/O may be read at any time. This is a read only register. The RB command will return the input and output configuration. The bits defined as outputs may be changed by sending the BH and BL commands through the command queues. Bits 12 and 13 are fixed as outputs. See Section 5 for more information. (See Table 3-9)

Table 3-9 USER I/O REGISTER DESCRIPTION (8-13)

BIT	DESCRIPTION
0	Bit 8
1	Bit 9
2	Bit 10
3	Bit 11
4	Bit 12
5	Bit 13
6	Unused
7	Unused

3.4.7. LIMIT STATUS REGISTER

The limit status register is a read only register which allows the host to determine whether one or more axes are currently in a limit condition. The individual axis limit bits are defined in Table 3-10.

Table 3-10 LIMIT STATUS REGISTER DESCRIPTION

BIT	DESCRIPTION
0	X axis limit status
1	Y axis limit status
2	Z axis limit status
3	T axis limit status
4	U axis limit status
5	V axis limit status
6	R axis limit status
7	S axis limit status

3.4.8. HOME SWITCH STATUS REGISTER

The home switch status register is a read only register which allows the host to determine the state of the home switches at any time. These bits are latched and are reset by a read of the register. (See Table 3-11)

Table 3-11 HOME SWITCH STATUS REGISTER DESCRIPTION

BIT	DESCRIPTION
0	X axis home status
1	Y axis home status
2	Z axis home status
3	T axis home status
4	U axis home status
5	V axis home status
6	R axis home status
7	S axis home status

3.4.9. INTERRUPT VECTOR REGISTER

The interrupt vector register is a byte wide read/write register. The host may write the desired vector into this register at anytime. It will be returned during an interrupt acknowledge when this VME58 board is the highest priority board in the system at the selected priority level.

3.5. POWER SUPPLY REQUIREMENTS

The VME58 is designed to operate from the power supplied in the VME bus backplane. The host computer or expansion box must be capable of supplying 1.75 amps typical for operation of the VME58 board.

CAUTION:

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

3.6. COMMUNICATION CHANNEL

The VME58 uses a 4K byte dual port RAM for communication with the host computer. A 256 character input buffer and a 256 character output buffer are available. The input and output buffers are circular queues. The host computer maintains its output pointer to the output buffer which is also in the dual port RAM and the input pointer to the input buffer. It can compare the input to the output pointer to determine if there is input data available and space for sending output data.

Data relevant to the motion process is also available in the dual port RAM to allow the host to monitor the process. Data such as position, velocity and acceleration are available in real time. Table 3-12 shows the offset assignments.

For communication purposes, the dual port RAM supports word accesses only. Thus, any character written to the output buffer is extended to a word; i.e. 2 bytes. Thus the 256 character buffer actually occupies 512 bytes (256 words) of memory in the dual port RAM.

3.6.1. THEORY OF OPERATION

To send a command to the VME58 the host would first read the Output Put Index and then the Output Get Index and compare the two to see if there is space available for more characters and the Output Put index will indicate the offset to the Output Buffer where to start sending the characters. The characters are written to consecutive word addresses and then the Output Put Index is updated by the host by writing the sum of the number of characters added to the number that is currently in that index location.

If a response is expected from the VME58, such as a WY command, the host would then read the Input Put Index and the Input Get Index and compare the results to see if there are characters to be read. The difference between the two indexes is the number of characters to be read. The host would read the characters and then write to the Input Get Index the number of characters it has read.

The axis information, referred to as data area, starting at address offset 1024 can be read at anytime. The host must request an update of this information by writing a 1 to bit 0 of the control register. Then the host must read the control register and wait until that bit is reset to 0. The VME58 will reset the bit in the control register once it has completed updating the dual-port memory. Typically this happens at a very fast rate and provides near real-time information.

Table 3-12 DUAL PORT RAM MEMORY OFFSET ASSIGNMENTS

ADDRESS OFFSET		DESCRIPTION	FUNCTION
HEX	DECIMAL		
F000	0-1	Input Put Index	Read
F002	2-3	Output Get Index	Read
F004-F203	4-515	Input Buffer	Read
F204-F3FF	516-1023	Reserved	Read
F400	1024	X Encoder Position	Read
F404	1028	X Command Position	Read
F408	1032	X Command Velocity	Read
F40C	1036	X Acceleration	Read
F410	1040	X Maximum Velocity	Read
F414	1044	X Base Velocity	Read
F418	1048	X Proportional Gain	Read
F41C	1052	X Derivative Gain	Read
F420	1056	X Integral Gain	Read
F424	1060	X Accel. Feed Forward	Read
F428	1064	X Velocity Feed Forward	Read
F42C	1068	X Offset	Read
F430-F47F	1072-1151	Reserved	Read
F480	1152	Y Encoder Position	Read
F484	1156	Y Command Position	Read
F488	1160	Y Command Velocity	Read
F48C	1164	Y Acceleration	Read
F490	1168	Y Maximum Velocity	Read
F494	1172	Y Base Velocity	Read
F498	1176	Y Proportional Gain	Read
F49C	1180	Y Derivative Gain	Read
F4A0	1184	Y Integral Gain	Read
F4A4	1188	Y Accel. Feed Forward	Read
F4A8	1192	Y Velocity Feed Forward	Read
F4AC	1196	Y Offset	Read
F4B0-F4FF	1200-1279	Reserved	Read
F500	1280	Z Encoder Position	Read
F504	1284	Z Command Position	Read
F508	1288	Z Command Velocity	Read
F50C	1292	Z Acceleration	Read
F510	1296	Z Maximum Velocity	Read
F514	1300	Z Base Velocity	Read
F518	1304	Z Proportional Gain	Read
F51C	1308	Z Derivative Gain	Read
F520	1312	Z Integral Gain	Read
F524	1316	Z Accel. Feed Forward	Read
F528	1320	Z Velocity Feed Forward	Read
F52C	1324	Z Offset	Read
F530-F57F	1328-1407	Reserved	Read
F580	1408	T Encoder Position	Read
F584	1412	T Command Position	Read
F588	1416	T Command Velocity	Read
F58C	1420	T Acceleration	Read
F590	1424	T Maximum Velocity	Read
F594	1428	T Base Velocity	Read
F598	1432	T Proportional Gain	Read
F59C	1436	T Derivative Gain	Read
F5A0	1440	T Integral Gain	Read
F5A4	1444	T Accel. Feed Forward	Read
F5A8	1448	T Velocity Feed Forward	Read
F5AC	1452	T Offset	Read
F5B0-F5FF	1456-1535	Reserved	Read

ADDRESS OFFSET HEX	DECIMAL	DESCRIPTION	FUNCTION
F600	1536	U Encoder Position	Read
F604	1544	U Command Velocity	Read
F608	1548	U Acceleration	Read
F60C	1552	U Maximum Velocity	Read
F610	1556	U Base Velocity	Read
F614	1560	U Proportional Gain	Read
F618	1564	U Derivative Gain	Read
F61C	1568	U Integral Gain	Read
F620	1572	U Accel. Feed Forward	Read
F624	1576	U Velocity Feed Forward	Read
F628	1590	U Offset	Read
F62C-F67F	1584-1663	Reserved	Read
F680	1664	V Encoder Position	Read
F684	1668	V Command Position	Read
F688	1672	V Command Velocity	Read
F68C	1676	V Acceleration	Read
F690	1680	V Maximum Velocity	Read
F694	1684	V Base Velocity	Read
F698	1688	V Proportional Gain	Read
F69C	1692	V Derivative Gain	Read
F6A0	1696	V Integral Gain	Read
F6A4	1700	V Accel. Feed Forward	Read
F6A8	1704	V Velocity Feed Forward	Read
F6AC	1708	V Offset	Read
F6B0-F6FF	1712-1791	Reserved	Read
F700	1792	R Encoder Position	Read
F704	1796	R Command Position	Read
F708	1800	R Command Velocity	Read
F70C	1804	R Acceleration	Read
F710	1808	R Maximum Velocity	Read
F714	1812	R Base Velocity	Read
F718	1816	R Proportional Gain	Read
F71C	1820	R Derivative Gain	Read
F720	1824	R Integral Gain	Read
F724	1828	R Accel. Feed Forward	Read
F728	1832	R Velocity Feed Forward	Read
F72C	1836	R Offset	Read
F730-F77F	1840-1919	Reserved	Read
F780	1920	S Encoder Position	Read
F784	1924	S Command Position	Read
F788	1928	S Command Velocity	Read
F78C	1932	S Acceleration	Read
F790	1936	S Maximum Velocity	Read
F794	1940	S Base Velocity	Read
F798	1944	S Proportional Gain	Read
F79C	1948	S Derivative Gain	Read
F7A0	1952	S Integral Gain	Read
F7A4	1956	S Accel. Feed Forward	Read
F7A8	1960	S Velocity Feed Forward	Read
F7AC	1964	S Offset	Read
F7B0-F7FF	1968-2047	Reserved	Read
F800	2048-2049	Output Put Index	Read/Write
F802	2050-2051	Input Get Index	Read/Write
F804-FA03	2052-2563	Output Buffer	Read/Write
FA04-FF87	2564-2975	Reserved	Read/Write
FF88	3976	Mail Box	Read/Write
FF8C-FFDF	3980-4063	Reserved	Read/Write
FFE0-FFFF	4064-4095	Registers	

3.7. MAILBOX FEATURE

A new Mailbox feature has been added to the “58 Series” of OMS motion control products. The mailbox allows a second mode of communications between the Host and OMS Controller. This second communications mode is useful in the event that the Output Buffer has become full preventing further communications. Only one Mailbox command is currently implemented. The command and its associated code is described in Table 3-13.

Table 3-13 MAILBOX FEATURE

CODE	COMMAND	FUNCTION
0001 H	Kill	Terminate motion, flush command queue

The Mailbox resides in the Dual Port RAM at offset 0xF88 and is defined as a long word (4 bytes). To use the Mailbox, the Host first places the desired code in the mailbox. The Host then sets bit 5 of the Control Register to interrupt the OMS Controller and signal that a Mailbox message is available. The OMS Controller then reads and interprets the code, clears the mailbox and resets bit 5 of the Control and Status Registers.

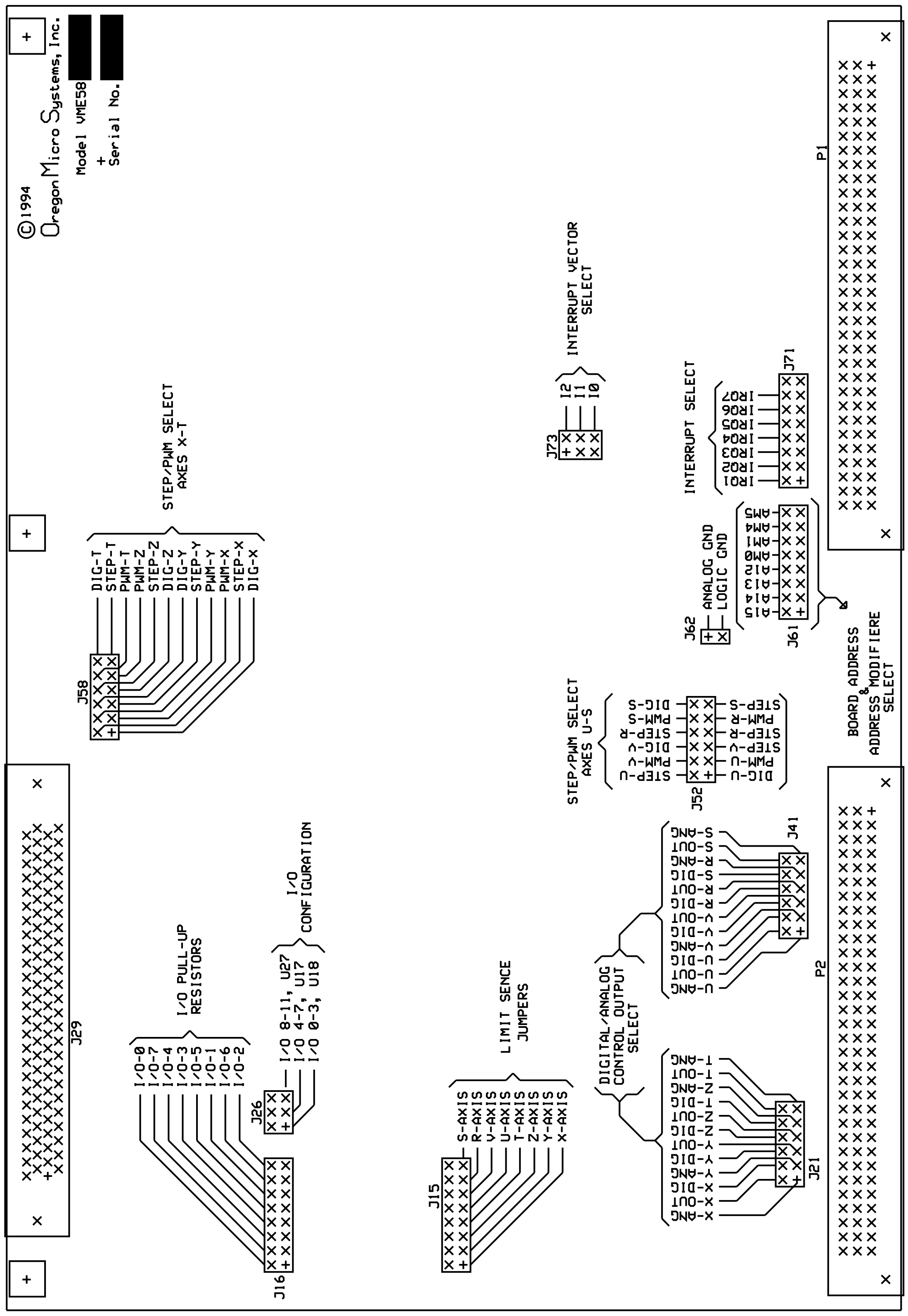

Figure 3-1 LOCATION OF OPTION JUMPERS

This page intentionally left blank

4. DRIVER INTERFACE

4.1. INTRODUCTION

The VME58 is available in several configurations to manage combinations of servo and step motor systems. The front panel connector uses 0.025 inch square posts on 0.05 by 0.10 inch centers. The mating connector is an AMP, Inc. part number 749621-9 with a 749081-1 hood and strain relief. A connection to the host computer for +5VDC power, digital ground and analog ground is provided for each axis pair for convenience in system integration. Motor control signals are also available on the P2 connector of the VME bus.

4.2. LIMIT AND HOME INPUTS

Limit and home inputs are provided for each axis to facilitate system implementation. They may be activated by 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 remove the excitation from the affected axis if the motor travels beyond its allowable limits and trips the switch. The limit switches may be changed to true when open, if desired, by removing the jumper on J15. See Section 2.2. for further information.

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 or HR command will cause the motor to decelerate to a stop when the switch closes. On finding the home position, the position counters will be initialized to the parameter supplied with the command. The sense of the home switches may be changed to true when open, if desired, by use of the HH command.

4.3. DRIVER OUTPUT

The VME58 is configured at the factory for a combination of servo drive, stepper and digital servo drive capability. The servo drive output may be either unipolar analog (0–10 volt) or bipolar analog (-10 to +10 volt), bipolar PWM or unipolar PWM which are user selectable. See Section 2.2. for more information on the jumper selection of these parameters (see also the UN and BI commands in Section 5). Tables 4-1 through 4-4 show the output connections for the standard output configurations.

Table 4-1 VME58-8S OUTPUT CONNECTOR PIN LIST (J29)

FUNCTION	PINS		FUNCTION
User I/O 0	1	51	+5VDC
User I/O 2	2	52	User I/O 1
User I/O 4	3	53	User I/O 3
User I/O 6	4	54	User I/O 5
User I/O 8	5	55	User I/O 7
User I/O 10	6	56	User I/O 9
User I/O 12	7	57	User I/O 11
User I/O 13	8	58	Ground
Analog Ground	9	59	+5VDC
X Phase A	10	60	Ground
X Phase B	11	61	X Index
X Direction	12	62	X Axis Output
X Auxiliary Output	13	63	X Positive Limit
X Home	14	64	X Negative Limit
Y Phase A	15	65	Y Index
Y Phase B	16	66	Y Axis Output
Y Direction	17	67	Y Positive Limit
Y Auxiliary Output	18	68	Y Negative Limit
Y Home	19	69	+5VDC
Analog Ground	20	70	Ground
Z Phase A	21	71	Z Index
Z Phase B	22	72	Z Axis Output
Z Direction	23	73	Z Positive Limit
Z Auxiliary Output	24	74	Z Negative Limit
Z Home	25	75	T Index
T Phase A	26	76	T Axis Output
T Phase B	27	77	T Auxiliary Output
T Direction	28	78	T Positive Limit
T Home	29	79	T Negative Limit
Analog Ground	30	80	+5VDC
U Phase A	31	81	Ground
U Phase B	32	82	U Index
U Direction	33	83	U Axis Output
U Auxiliary Output	34	84	U Positive Limit
U Home	35	85	U Negative Limit
V Phase A	36	86	V Index
V Phase B	37	87	V Axis Output
V Direction	38	88	V Positive Limit
V Auxiliary Output	39	89	V Negative Limit
V Home	40	90	+5VDC
Analog Ground	41	91	Ground
R Phase A	42	92	R Index
R Phase B	43	93	R Axis Output
R Direction	44	94	R Positive Limit
R Auxiliary Output	45	95	R Negative Limit
R Home	46	96	S Index
S Phase A	47	97	S Axis Output
S Phase B	48	98	S Auxiliary Output
S Direction	49	99	S Positive Limit
S Home	50	100	S Negative Limit

Table 4-2 VME58-4S4 OUTPUT CONNECTOR PIN LIST (J29)

FUNCTION	PINS		FUNCTION
User I/O 0	1	51	+5VDC
User I/O 2	2	52	User I/O 1
User I/O 4	3	53	User I/O 3
User I/O 6	4	54	User I/O 5
User I/O 8	5	55	User I/O 7
User I/O 10	6	56	User I/O 9
User I/O 12	7	57	User I/O 11
User I/O 13	8	58	Ground
Analog Ground	9	59	+5VDC
X Phase A	10	60	Ground
X Phase B	11	61	X Index
X Direction	12	62	X Axis Servo Output
X Auxiliary Output	13	63	X Positive Limit
X Home	14	64	X Negative Limit
Y Phase A	15	65	Y Index
Y Phase B	16	66	Y Axis Servo Output
Y Direction	17	67	Y Positive Limit
Y Auxiliary Output	18	68	Y Negative Limit
Y Home	19	69	+5VDC
Analog Ground	20	70	Ground
Z Phase A	21	71	Z Index
Z Phase B	22	72	Z Axis Servo Output
Z Direction	23	73	Z Positive Limit
Z Auxiliary Output	24	74	Z Negative Limit
Z Home	25	75	T Index
T Phase A	26	76	T Axis Servo Output
T Phase B	27	77	T Auxiliary Output
T Direction	28	78	T Positive Limit
T Home	29	79	T Negative Limit
Analog Ground	30	80	+5VDC
	31	81	Ground
	32	82	
U Direction	33	83	U Axis Step Output
U Auxiliary Output	34	84	U Positive Limit
U Home	35	85	U Negative Limit
	36	86	
	37	87	V Axis Step Output
V Direction	38	88	V Positive Limit
V Auxiliary Output	39	89	V Negative Limit
V Home	40	90	+5VDC
Analog Ground	41	91	Ground
	42	92	
	43	93	R Axis Step Output
R Direction	44	94	R Positive Limit
R Auxiliary Output	45	95	R Negative Limit
R Home	46	96	
	47	97	S Axis Step Output
	48	98	S Auxiliary Output
S Direction	49	99	S Positive Limit
S Home	50	100	S Negative Limit

Table 4-3 VME58-8 OUTPUT CONNECTOR PIN LIST (J29)

FUNCTION	PINS		FUNCTION
User I/O 0	1	51	+5VDC
User I/O 2	2	52	User I/O 1
User I/O 4	3	53	User I/O 3
User I/O 6	4	54	User I/O 5
User I/O 8	5	55	User I/O 7
User I/O 10	6	56	User I/O 9
User I/O 12	7	57	User I/O 11
User I/O 13	8	58	Ground
Analog Ground	9	59	+5VDC
	10	60	Ground
	11	61	
X Direction	12	62	X Axis Step Output
X Auxiliary Output	13	63	X Positive Limit
X Home	14	64	X Negative Limit
	15	65	
	16	66	Y Axis Step Output
Y Direction	17	67	Y Positive Limit
Y Auxiliary Output	18	68	Y Negative Limit
Y Home	19	69	+5VDC
Analog Ground	20	70	Ground
	21	71	
	22	72	Z Axis Step Output
Z Direction	23	73	Z Positive Limit
Z Auxiliary Output	24	74	Z Negative Limit
Z Home	25	75	
	26	76	T Axis Step Output
	27	77	T Auxiliary Output
T Direction	28	78	T Positive Limit
T Home	29	79	T Negative Limit
Analog Ground	30	80	+5VDC
	31	81	Ground
	32	82	
U Direction	33	83	U Axis Step Output
U Auxiliary Output	34	84	U Positive Limit
U Home	35	85	U Negative Limit
	36	86	
	37	87	V Axis Step Output
V Direction	38	88	V Positive Limit
V Auxiliary Output	39	89	V Negative Limit
V Home	40	90	+5VDC
Analog Ground	41	91	Ground
	42	92	
	43	93	R Axis Step Output
R Direction	44	94	R Positive Limit
R Auxiliary Output	45	95	R Negative Limit
R Home	46	96	
	47	97	S Axis Step Output
	48	98	S Auxiliary Output
S Direction	49	99	S Positive Limit
S Home	50	100	S Negative Limit

Table 4-4 VME58-4E OUTPUT CONNECTOR PIN LIST (J29)

FUNCTION	PINS		FUNCTION
User I/O 0	1	51	+5VDC
User I/O 2	2	52	User I/O 1
User I/O 4	3	53	User I/O 3
User I/O 6	4	54	User I/O 5
User I/O 8	5	55	User I/O 7
User I/O 10	6	56	User I/O 9
User I/O 12	7	57	User I/O 11
User I/O 13	8	58	Ground
Analog Ground	9	59	+5VDC
	10	60	Ground
	11	61	
X Direction	12	62	X Axis Step Output
X Auxiliary Output	13	63	X Positive Limit
X Home	14	64	X Negative Limit
	15	65	
	16	66	Y Axis Step Output
Y Direction	17	67	Y Positive Limit
Y Auxiliary Output	18	68	Y Negative Limit
Y Home	19	69	+5VDC
Analog Ground	20	70	Ground
	21	71	
	22	72	Z Axis Step Output
Z Direction	23	73	Z Positive Limit
Z Auxiliary Output	24	74	Z Negative Limit
Z Home	25	75	
	26	76	T Axis Step Output
	27	77	T Auxiliary Output
T Direction	28	78	T Positive Limit
T Home	29	79	T Negative Limit
Analog Ground	30	80	+5VDC
X Phase A	31	81	Ground
X Phase B	32	82	X Index
	33	83	
	34	84	
	35	85	
Y Phase A	36	86	Y Index
Y Phase B	37	87	
	38	88	
	39	89	
	40	90	+5VDC
Analog Ground	41	91	Ground
Z Phase A	42	92	Z Index
Z Phase B	43	93	
	44	94	
	45	95	
	46	96	T Index
T Phase A	47	97	
T Phase B	48	98	
	49	99	
	50	100	

Table 4-5 P2 CONNECTOR PIN ASSIGNMENTS

ROW C FUNCTION	PIN	ROW A FUNCTION
X Index	1	X Phase B
X Phase A	2	X Axis Output
X Dir	3	X Pos. Limit
X Home	4	X Neg. Limit
Y Index	5	Y Phase B
Y Phase A	6	Y Axis Output
Y Dir	7	Y Pos. Limit
Y Home	8	Y Neg. Limit
Z Index	9	Z Phase B
Z Phase A	10	Z Axis Output
Z Dir	11	Z Pos. Limit
Z Home	12	Z Neg. Limit
T Index	13	T Phase B
T Phase A	14	T Axis Output
T Dir	15	T Pos. Limit
T Home	16	T Neg. Limit
U Index	17	U Phase B
U Phase A	18	U Axis Output
U Dir	19	U Pos. Limit
U Home	20	U Neg. LImit
V Index	21	V Phase B
V Phase A	22	V Axis Output
V Dir	23	V Pos. Limit
V Home	24	V Neg. Limit
R Index	25	R Phase B
R Phase A	26	R Axis Output
R Dir	27	R Pos. Limit
R Home	28	R Neg. Limit
S Index	29	S Phase B
S Phase A	30	S Axis Output
S Dir	31	S Pos. Limit
S Home	32	S Neg. Limit

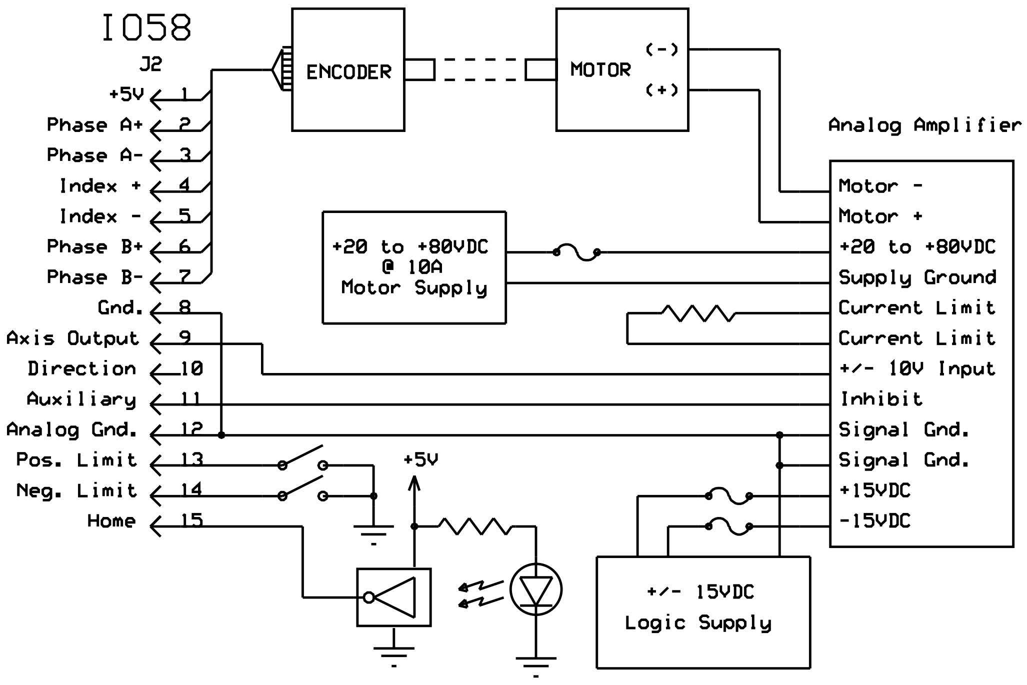

Figure 4-1 ANALOG SERVO CONFIGURATION

Figure 4-1 shows the system connections for a typical analog servo configuration.

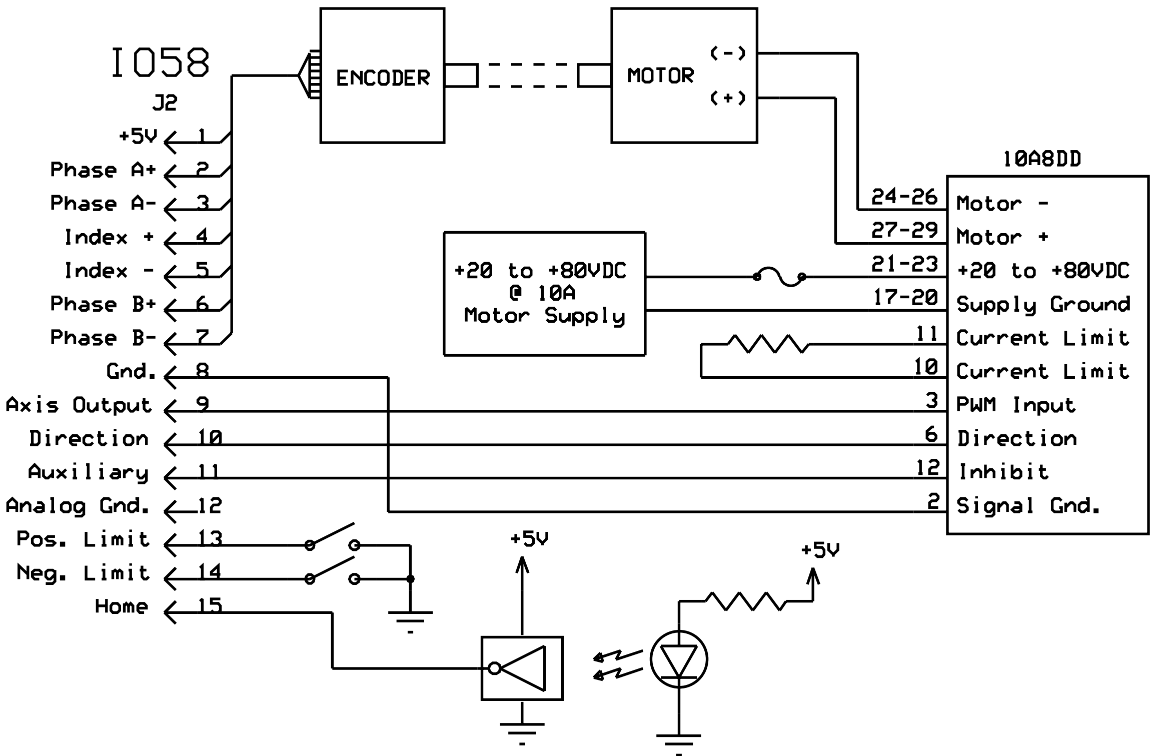

Figure 4-2 PWM SERVO CONFIGURATION

Figure 4-2 shows the system connections for a typical PWM servo configuration.

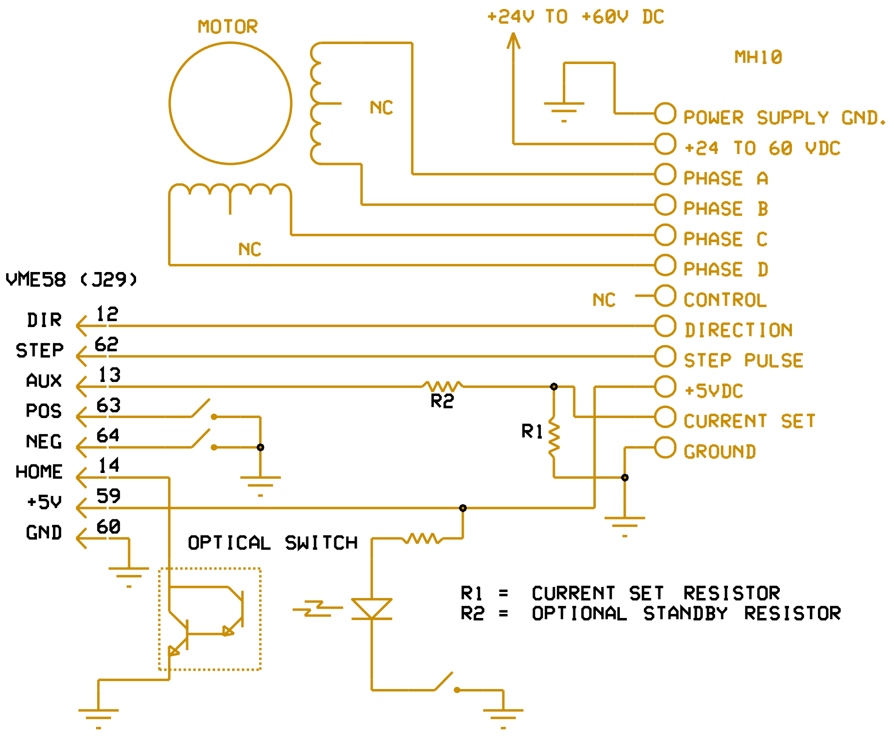

Figure 4-3 VME58/MH10 INTERCONNECTIONS

Figure 4-3 shows the system connections for a typical stepper configuration.

4.4. IO58 ADAPTER MODULE

The optional IO58 is an adapter module designed to provide separate connectors for each axis. It includes a 3 meter cable with the mating connector to fit the VME58 I/O connections. Each axis has its own 15 pin subminiature D connector.* Tables 4-6 and 4-7 show

Table 4-6 IO58 INDIVIDUAL CONNECTOR PER AXIS

FUNCTION	PINS		FUNCTION
+5VDC	1	9	Output
Phase A+ *	2	10	Direction
Phase A- *	3	11	Auxiliary
Index + *	4	12	Analog Ground
Index - *	5	13	Positive Limit
Phase B+ *	6	14	Negative Limit
Phase B- *	7	15	Home
Ground	8		

* **NOTE:** Encoder inputs for step motor configurations are not on the same connectors as their respective motor outputs. They are offset by 4 axes, i.e. Axis X motor signals are found on J2 and encoder signals for axis X should connect at J6. Servo axis feedback signals belong on their respective motor's output connector, i.e. all axis X signals on J2.

Table 4-7 IO58 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
+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		

the connections to the IO58. Note: The encoder inputs are on separate connectors on -E models. See the VME58 connector pin lists.

4.4.1. FUSED PROTECTION

The external +5VDC supply available at the connectors (J29 and P2) of the VME58 is protected by a semiconductor type fuse. This supply is intended to be utilized with accessories used in conjunction with the VME58 such as the IO58 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 VME58 down, ensuring the over current situation has been removed, and by powering the VME58 back up again.

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

4.5. ENCODER FEEDBACK

Incremental encoder feedback is provided for all servo axes and is optional for the stepper axes. The encoder option accepts quadrature pulse inputs from high resolution encoders at rates up to 2 MHz (after quadrature detection). The VME58 monitors actual position through the encoder pulse train. It then continuously calculates the position error on the servo axes and adjusts the output based on that error. The stepper axes will monitor the error and correct the position after the move is finished. All modes are capable of slip or stall detection and encoder tracking with electronic gearing. These options are selectable by the user through software commands.

4.6. ENCODER SELECTION AND COMPATIBILITY

The VME58 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. The VME58 inputs have built in hysteresis to minimize effects of noise pickup. The IO58 has additional line receivers to accommodate encoders with differential line driver outputs. These line receivers may be disabled for systems with single ended encoders when used with the IO58 (see Figure 4-5).

4.7. HOME PROCEDURES

Two logical inputs are provided to synchronize the physical hardware with the VME58 controller, i.e. put the controlled motor in the home position. (See Figures 4-4 and 4-5.)

The VME58 home inputs can be used with switches 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 index input uses internal logic to establish the home position when used with the HE command mode. This position consists of the logical AND of the encoder index pulse, the home switch external input (low true only) and a single quadrant from the encoder logic. The home switch 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 HM and HR commands should be used only after reducing the velocity to no more than 2048 pulses per second (1024 pulses per second for 8 axes boards). This limit on velocity is necessary to avoid ambiguity of the home position if more than one pulse occurs per sample interval. This input is not inverted by the HH and HL commands. 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.

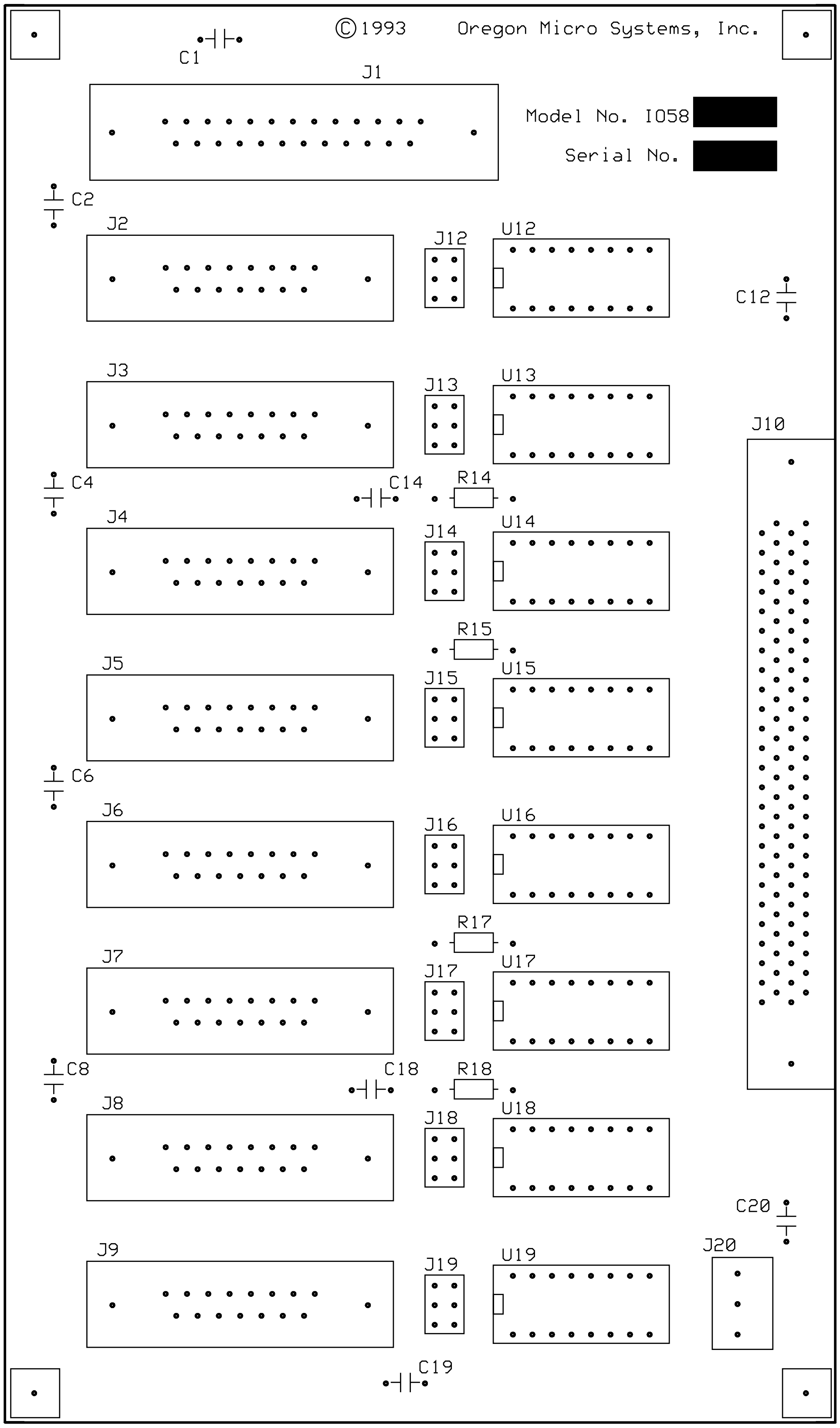

Figure 4-4 JUMPERS FOR DIFFERENTIAL INPUTS

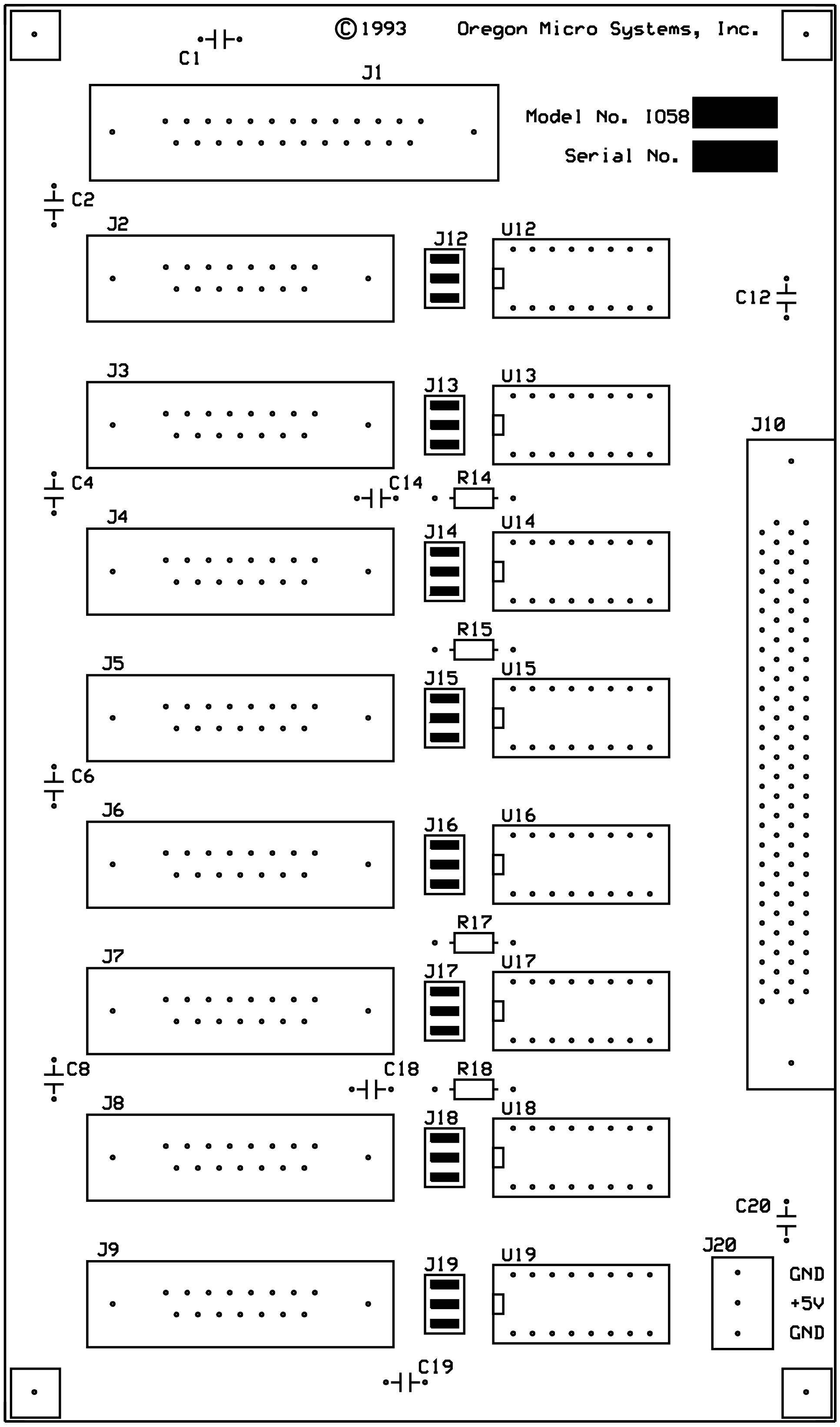

Figure 4-5 JUMPERS FOR SINGLE ENDED INPUTS

Table 4-8 CBL58 CABLE COLOR CODES

WIRE COLOR	CONTACT	CONTACT	WIRE COLOR
Pink/Yellow	51	1	White/Tan
Yellow/Pink	52	2	Tan/White
Pink/Green	53	3	White/Brown
Green/Pink	54	4	Brown/White
Pink/Blue	55	5	White/Pink
Blue/Pink	56	6	Pink/White
Pink/Violet	57	7	White/Orange
Violet/Pink	58	8	Orange/White
Pink/Gray	59	9	White/Yellow
Gray/Pink	60	10	Yellow/White
Orange/Yellow	61	11	White/Green
Yellow/Orange	62	12	Green/White
Orange/Green	63	13	White/Blue
Green/Orange	64	14	Blue/White
Orange/Blue	65	15	White/Violet
Blue/Orange	66	16	Violet/White
Orange/Violet	67	17	White/Gray
Violet/Orange	68	18	Gray/White
Orange/Gray	69	19	Tan/Brown
Gray/Orange	70	20	Brown/Tan
Yellow/Green	71	21	Tan/Pink
Green/Yellow	72	22	Pink/Tan
Yellow/Blue	73	23	Tan/Orange
Blue/Yellow	74	24	Orange/Tan
Yellow/Violet	75	25	Tan/Yellow
Violet/Yellow	76	26	Yellow/Tan
Yellow/Gray	77	27	Tan/Green
Gray/Yellow	78	28	Green/Tan
Green/Blue	79	29	Tan/Blue
Blue/Green	80	30	Blue/Tan
Green/Violet	81	31	Tan/Violet
Violet/Green	82	32	Violet/Tan
Green/Gray	83	33	Tan/Gray
Gray/Green	84	34	Gray/Tan
Blue/Violet	85	35	Brown/Pink
Violet/Blue	86	36	Pink/Brown
Blue/Gray	87	37	Brown/Orange
Gray/Blue	88	38	Orange/Brown
Violet/Gray	89	39	Brown/Yellow
Gray/Violet	90	40	Yellow/Brown
White	91	41	Brown/Green
Tan	92	42	Green/Brown
Gray	93	43	Brown/Blue
Brown	94	44	Blue/Brown
Blue	95	45	Brown/Violet
Pink	96	46	Violet/Brown
Violet	97	47	Brown/Gray
Orange	98	48	Gray/Brown
Green	99	49	Pink/Orange
Yellow	100	50	Orange/Pink

This page intentionally left blank

5. COMMAND STRUCTURE

5.1. INTRODUCTION

An extensive command structure is built into the VME58 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. A separate 7160 command and parameter buffer is provided for the contour definition.

The following commands in this section are included in the VME58 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. 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 VME58 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 VME58 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.

5.2. 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 simultaneously. The synchronized modes (AA) insert special wait opcodes which allow the axes to be synchronized in this mode. When the commands are nested within loops, the queue space is not reclaimed until after the loop has been executed the programmed number of times. For loops larger than the queue space, the loop may never be completed since it cannot reclaim the queue space and cannot accept the loop terminator. The RQ command may be used to examine the remaining queue space.

The following commands are available in firmware revision 2.00 and above.

5.3. 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

5.4. 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 VME58 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 VME58 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 each 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 each 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	Immediate
AA,AM	Immediate
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 counts. 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 counts. Note that the parameter is the number of counts, 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

PI# PWM PERIOD

The PI command sets the period for the PWM output signal. The command argument is in microseconds. Valid command arguments are greater than 20 µs and less than 7812 µs. The default PWM period is 40 µs.

QUEUE REQUIREMENTS	
MODE	**SERVO**
AX - AS	3
AA,AM	Not valid
AA/CD	Not valid

Example: Set up servo axis X for unipolar PWM operation with a period of 2000 µs.

Enter: AX UN PI2000 HG5,200,100 HN

BI BIPOLAR

The BI command sets the analog and PWM torque outputs to bipolar. When bipolar is selected, a zero torque reference will result in a 50% PWM duty cycle. A negative torque reference will result in less than 50% PWM duty cycle; a positive torque reference will result in greater than 50% PWM duty cycle. The analog output will range between +10VDC and -10VDC when bipolar is enabled. It is necessary to issue either the UN or the BI command to enable PWM operation for a particular axis. The BI command is valid only in the single axis mode. This is the default mode at power up or reset.

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

Example: Set up servo axis X for bipolar operation.

Enter: AX BI HG5,200,100 HN

UN UNIPOLAR

The UN command sets the analog and PWM torque outputs to unipolar. When unipolar is selected, a zero torque reference will result in a low DC level (or minimum PWM duty cycle) and maximum torque reference will result in a high DC level (or maximum PWM duty cycle). The analog output will range between 0.0VDC and +10VDC when unipolar is enabled. The direction output signal will define the sign of the output. It is necessary to issue either the UN or the BI command to enable PWM operation for a particular axis. This command is valid only in the single axis mode.

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

Example: Set up servo axis X for unipolar operation.

Enter: AX UN HG5,200,100 HN

This page intentionally left blank.

5.5. 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. 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 for 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	2

Example: Turn on output bits 10 and 12 after a move. Note that this is only valid for bits which have been configured as outputs. 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	2

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 four digit hex format, surrounded by line feed and carriage return pairs. A one in any binary position signals that bit as being low. Input bits 0 and 1 are latched by an active low state, but are cleared after execution of this command. The SW or WS command will reset input bits 0 or 1 if the respective bit is designated in the command.

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>1409<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.

Note: I/O bits 12 and 13 are dedicated outputs and cannot be configured as inputs. Therefore, the hex digit third from the left will always be a 3.

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>3F00<LF><CR>

5.6. 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 and less than 1,000,000,000. 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 200,000 units per second per second upon power-up or reset.

The acceleration register may be automatically modified by the VME58 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 1,000,000 counts per second. The velocity defaults to 20,000 at power up or reset. This 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. For encoder axes, 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. For stepper axes, this command turns off the position hold and interrupt on slip modes when used with the encoder option. Position maintenance is not disabled by the LP command for servo axes.

QUEUE REQUIREMENTS		
MODE	**NO ENCODER**	**ENCODER AND SERVO**
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 an interpolated move to a non-interpolated 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 an interpolated move to a non-interpolated 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 counts in the negative direction.

Enter: AX MR-2468 GO

Example: In the AA mode, move the X axis 12345 counts 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 VME58 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 VME58 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, KL or PF command. The MT command is not valid in loops (LS-LE, WH-WG, WS-WD) 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, except in a contour.

The MT command may alter predefined acceleration and velocity values. These values should be redefined if you go from an interpolated move to a non-interpolated 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
```

Example: Define a contour to move the X and Z axes at a constant velocity between three points.

Enter:
```
CD100,,200;
MT1000,1500;
MT2000,3000;
MT3000,3300;
CE
```

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 both the position counter and the encoder position register 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.

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.

5.7. 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 has encoder axes, add the value found in Table B to the value from Table A, to determine total queue usage of those axes.

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

TABLE B			
ADDITIONAL ENCODER 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 has encoder axes, add the value found in Table B to the value from Table A, to determine total queue usage of those axes.

TABLE A
QUEUE REQUIREMENTS

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

TABLE B
ADDITIONAL ENCODER 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 after ramping up at the rate specified by the AC command. 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.

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. The application program may control the time delay between commands to affect the actual distance traveled. See also the WT (page 49) and WQ (page 47) commands.

Enter: JG100000;JG35000;ST

Example: Turn on user I/O bit 8 for .5 second (WT500) once the X axis has ramped to its programmed velocity. The WQ command following the JG command will pause the command parser until the X axis has ramped to its programmed velocity. Turn off bit 8 after .5 second and ramp to a stop.

Enter:
```
AX;
AC500000;
JG100000;
WQ;
BL8;
WT500;
BH8;
ST;
```

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 AND SERVO**
AX - AS	2	3
AA,AM	Not valid	
AA/CD	Not valid	

Example: Jog the Y axis at 2$^2/_3$ counts per second.

Enter: AY JF2.667

FL FLUSH QUEUE

The FL command will immediately flush the command queue. The output will continue to run at the velocity it was traveling when the FL command was issued until another command is executed. The FL command is valid only in the single axis mode.

QUEUE REQUIREMENTS			
MODE	**LINEAR**	**PARABOLIC**	**COSINE**
AX - AS	Unused	Immediate	
AA,AM	Not valid		
AA/CD	Not valid		

5.8. 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.

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

Example: Stop all movement and flush all the axes queues because an incorrect movement command was executed.

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

5.9. 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.

It is important to note 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. 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 near 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 5-36)

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,AM	2
AA/CD	Not valid

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

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

WD WHILE END

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

QUEUE REQUIREMENTS	
MODE	
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	3
AA,AM	3
AA/CD	Not valid

Example: You have a 3 axis platform that you use to drill holes in the center of a 1/4 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 exe-
                      cute
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, page 5-39)

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, page 5-39)

5.10. 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 1024 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 AND SERVO**
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 1024 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 AND SERVO**
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 1024 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 HM VL1000 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 AND SERVO**
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 AND SERVO**
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

5.11. 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 VME58 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 for encoder axes.

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

Example: Send DONE when stepper 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 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 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	Not valid
AA,AM	2
AA/CD	Not valid

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

Enter:
```
AA VL1000,1000,1000; AC10000,10000,10000;
LS10 MR1000,1000; GO WA AN,,1; MR,,500; GO AF,,1;
MR,,-500 GO LE
```

WQ WAIT FOR QUEUE TO EMPTY

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

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

Example: Move the Y axis 1,000 counts 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 VME58 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	1
AA,AM	1
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/seconds for 3 seconds, then stop.

Enter: AX JG5000 WT2000 JG10000 WT3000 WQ ST

5.12. 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.

The commands identified with an '*****' are for backward compatibility with previous generation OMS controllers and should not be used with new designs. Other preferred mechanisms such as reading directly from the dual port RAM are available for this data.

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 feed and carriage return pairs.

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>VME58 ver 2.00-4S4<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 feed and carriage return pairs.

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 feed and carriage return pairs. 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 7016. The response is fixed in length at 4 characters.

QUEUE REQUIREMENTS	
MODE	
AX - AS	Immediate
AA,AM	Immediate
AA/CD	Immediate

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

Enter: AT RQ

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

BX BIT REQUEST IN HEX

The BX command returns the state of the general purpose I/O bits in a four digit hex format, surrounded by line feed and carriage return pairs. A one in any binary position signals that bit as being low. Input bits 0 and 1 are latched by an active low state, but are cleared after execution of this command.

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>1409<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. The string has one line feed and two carriage returns on each end.

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 or UU 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 counts 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). 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 counts. Define user units so moves can be referenced in revolutions. Move the Z axis 3½ 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>`

5.13. 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 VME58 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 counts 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

5.14. PID FILTER CONTROL COMMANDS

The following commands are valid only for servo axes and should never be executed while the specific axis is in motion. (Reference Section 2-2.10.4.)

KP# PROPORTIONAL GAIN COEFFICIENT

This term tends to govern the overall accuracy performance of the servo loop. Low values produce very stable systems with low stiffness. High values produce accurate, stiff systems that may show signs of instability. The default value is 2.0, the minimum value is 0.1 and the maximum value is 1999.9. One digit to the right of the decimal place is allowed.

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

KI# INTEGRAL GAIN COEFFICIENT

This term is used to reduce position errors due to friction when the system is at rest. Excessive KI may result in low frequency oscillation or hunting. The default value is 4.0, the minimum value is 0.0 and the maximum value is 1999.9. One digit to the right of the decimal place is allowed.

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

KD# DERIVATIVE GAIN COEFFICIENT

This term is used to provide damping and stability. Low values tend to produce systems with fast response, but may lack stability as evidenced by ringing or oscillations. High values produce systems with excellent stability but slightly slower response. High values may also allow the use of higher Proportional Gain for greater accuracy without oscillations. The default value is 6.0, the minimum value is 0.0 and the maximum value is 1999.9. One digit to the right of the decimal place is allowed.

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

KV# VELOCITY FEEDFORWARD COEFFICIENT

This term is used in conjunction with velocity controlled servos(voltage mode servo amplifiers). This term is used to cancel position errors that are a result of high speed operation, since large errors are required to generate the high voltage required for high speed. Too large a value of KV may result in erratic operation after command velocity changes. The default value is 0.0, the minimum value is 0.0 and the maximum value is 1999.9. One digit to the right of the decimal place is allowed.

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

KA# ACCELERATION FEEDFORWARD COEFFICIENT

This term is used in conjunction with torque controlled servos(current mode servo amplifiers). Systems with high inertial loads may require additional torque during acceleration/deceleration to achieve optimum performance. The default value is 0.0, the minimum value is 0.0 and the maximum value is 1999.9. One digit to the right of the decimal place is allowed.

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

KO# OFFSET COEFFICIENT

This term provides a constant output to compensate for any torque offset in the load. The default value is 0.0, the minimum value is -127.9 and the maximum value is 127.9. One digit to the right of the decimal place is allowed.

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

KK# COMBINED COEFFICIENT

This term provides a short cut method for setting PID filter parameters. The coefficients set by this command are:

KP = KK#
KD = 3∗KK#
KI = 2∗KK#

The default value is 2.0, the minimum value is 0.1 and the maximum value is 1999.9.

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

KN# POSITION ERROR SUMMATION INTERVAL

This term sets the error summation interval. Factory default is KN = 8 yielding a summation of 256 update intervals. Units are in 2^{KN} update intervals. The default value is 8, the minimum value is 1 and the maximum value is 12.

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

PI# PWM PERIOD

The default PWM period defaults to 40 µs. Care must be taken when adjusting the PWM period. Motors with very low inductance may be adversely effected by a PWM period that is too long. Longer PWM periods may also produce undesirable audio noise in the motor/amplifier. Shorter PWM periods reduce the resolution of the PWM output. The default value is 40, the minimum value is 20 and the maximum value is 7812.

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

5.15. 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 - AS	Not valid	1
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 moves 25,000 counts, the encoder produces 10,000 counts. Set up an encoder ratio so the hold mode will work correctly.

Enter: ER10000,25000

HV# HOLD VELOCITY (STEPPERS ONLY)

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**	SERVO
AX - AS	Not valid	2	Not Valid
AA,AM	Not valid		
AA/CD	Not valid		

Example: (see HN command, page 5-65)

HG# HOLD FILTER PARAMETERS (STEPPERS ONLY)

The HG command requires one parameter when applied to a stepper axis and three parameters when applied to a servo axis.

When applied to a stepper axis with encoder feedback, 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	**STEPPER NO ENCODER**	**STEPPER ENCODER**	**SERVO**
AX - AS	Not valid	2	Not valid
AA,AM	Not valid		
AA/CD	Not valid		

Example: Set up the Y stepper axis with the encoder feedback option for position correction mode with a gain parameter of 2000.

Enter: AY HV2000 HD10 HG2000 HN

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. For servo axes this means that a zero torque output is generated. For stepper axes, 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 and is the default.

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

Example: (see HN command on the next page)

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 AND SERVO**
AX - AS	Not valid	2
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 the HG parameters for servo axes. The HN command enables position correction after a move and activates the HV, HG and HD commands for stepper axes with encoders. For stepper axes, hold and slip detection are disabled if an LP, HM, HR, SA, HF, ST or KL command is entered or if a limit is encountered. For servo axes, hold and slip detection are disabled if an HF command is entered.

QUEUE REQUIREMENTS		
MODE	**NO ENCODER**	**ENCODER AND SERVO**
AX - AS	Not valid	2
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. This command is available for encoder axes only.

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

Example: Send DONE when the axis is within deadband.

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

5.16. SLIP AND STALL DETECTION COMMANDS

ES# ENCODER SLIP TOLERANCE

The ES command parameter specifies tolerance before slip or stall is flagged in either the status register, or by the RL command response, or before the TN command activates. The mode must be turned on with an IS command and off with an HF command. The maximum value for the ES command parameter is 32,767.

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

Example: Your application can tolerate being up to 5 counts from the desired position before the controlling program should be notified of a slip condition.

Enter: ES5 IS

TN SLIP TOLERANCE KILL ON

The TN command enables additional action taken by the encoder slip tolerance system when the position error exceeds that specified by the ES command. When this mode is on the controller will flush the command queue, terminate motion and disable position maintenance for that particular axis when the slip tolerance is exceeded. This is the default mode at power up or reset for the servo axes. The IS command must be envolked before this command will have an effect.

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

Example: Enable slip tolerance kill on the X axis.

Enter: AX ES# IS TN

TF SLIP TOLERANCE KILL OFF

The TF command disables the TN command. The TN mode can not be disabled on a servo axes. This is the default mode at power up or reset for the stepper axes.

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

Example: Disable slip tolerance kill on the X axis.

Enter: AX TF

IS INTERRUPT ON SLIP

The IS command enables the VME58 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, i.e. the IS command must be re-initialized.

QUEUE REQUIREMENTS		
MODE	**NO ENCODER**	**ENCODER AND SERVO**
AX - AS	Not valid	1
AA,AM	Not valid	
AA/CD	Not valid	

Example: (see ES command, page 5-67)

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 surrounded by line feed and carriage return pairs, 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 AND SERVO**
AX - AS	Not valid	Immediate
AA,AM	Immediate	
AA/CD	Not valid	

Example: On a four axis board, see if any axis has slipped.

Enter: RL

Response: <LF><CR>NNSN<LF><CR> (The Z axis has slipped.)

HF HOLD OFF

The HF command disables position hold, stall detection and tracking modes.

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

Example: Disable slip detection on the X axis.

Enter: AX HF

5.17. 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. This command would normally be used only with stepper axes.

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

Example: Turn off encoder tracking on X axis.

Enter: AX HF

5.18. ENCODER HOME CONTROL COMMANDS

HE HOME ENCODER

The HE command enables encoder index mode when an HM, KM, KR 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. This command is available for encoder axes only. The home logic expressed in boolean terms is:

home=phase_A∗/phase_B∗index∗/home_switch

QUEUE REQUIREMENTS		
MODE	**NO ENCODER**	**ENCODER AND SERVO**
AX - AS	Not valid	Immediate
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 VME58 home switch mode to determine where home is when an HM, KM, KR 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 VME58 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 AND SERVO**
AX - AS	Not valid	Immediate
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

5.19. ENCODER STATUS REQUEST COMMANDS

The commands identified with an '*****' are for backward compatibility with previous generation OMS controls and should not be used with new designs. Other preferred mechanisms such as reading directly from the dual port RAM are available for this data.

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 AND SERVO**
AX - AS	Not valid	Immediate
AA,AM	Not valid	
AA/CD	Not valid	

Example: Examine the status of the Y axis encoder.

Enter: AY RE

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 AND SERVO**
AX - AS	Not valid	Immediate
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>

5.20. 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	1
AA,AM	Not valid
AA/CD	Not valid

Example: (see MV command on the next page)

MM MOVE MINUS

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

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

Example: Set the direction line to move in the minus direction on the Y axis.

Enter: AY MM

MV#,# MOVE VELOCITY

The MV command causes the motor to run to the new absolute position (parameter 1) at the new velocity (parameter 2). When the destination is reached control will be passed to the next command which should be another MV command or an SP command. If the command is not received in time the controller will continue to move at the specified velocity. Note that this is a slave mode and it is the responsibility of the user to provide the commands in time. They may be queued ahead of time. If a new MV command is sent after the controller has already passed the destination specified in the command, the controller will continue to move at the old velocity. Any number of counts 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.

QUEUE REQUIREMENTS		
MODE	**NO ENCODER**	**ENCODER AND SERVO**
AX - AS	4	5
AA,AM	Not valid	
AA/CD	Not valid	

Example: Generate a velocity staircase with the breakpoints given in absolute position.

Enter:

```
MP
MV10000,30000
MV20000,50000
MV30000,10000
SP35 000
```

The move as shown in Figure 5-1.

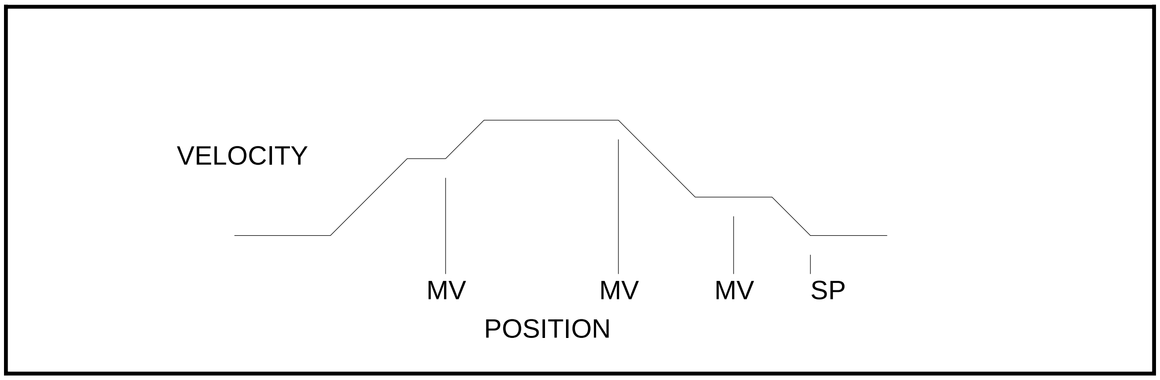

Figure 5-1 VELOCITY STAIRCASE PROFILE

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 AND SERVO**
AX - AS	3	4
AA,AM	Not valid	
AA/CD	Not valid	

Example: (see MV command on the previous page)

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 AND SERVO**
AX - AS	Flush + 4	Flush + 4
AA,AM	Not valid	
AA/CD	Not valid	

Example: Force axis to stop at 25,000.

Enter: FP25000

5.21. CONSTANT VELOCITY CONTOURING

The VME58 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, page 5-79)

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 5-79)

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: (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	2
AA,AM	2
AA/CD	Not valid

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
CX
```

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 or AM mode. A separate command queue for the contour definition is 7160 commands and parameters.

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.

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 counts per second. A contour is then defined beginning at coordinates 0,0 on the Z and T axes. The auxiliary output of the Y 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. Note that no commas are provided in the MT and CR commands for the inactive X and Y axes within the contour definition. 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 CD,,0,0;
AN,0; CR0,5000,3.1415926
CR0,0,6.2831853
AF,0; MT-10,10000
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 or AM 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 the previous page)

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 CD,,0,0;
AN,0;CR0,5000,3.1415926
CR0,0,6.2831853
AF,0; MT-10,10000
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. 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, page 5-79)

CV# CONTOUR VELOCITY

The CV command allows specification of contouring velocity. It is executed from the AA or AM 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.

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

Example: (see CD command, page 5-79)

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, page 5-79)

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. This means that no comma is required for a place holder for axes not defined in the contour.

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 Z axes.

Enter:
```
AA CV5000;
CD10000,,0;
MT20000,0
MT25000,10000
MT20000,20000
MT10000,20000
MT5000,10000
MT10000,0
CK
CX
```

RQ REQUEST QUEUE STATUS

The RQ command returns the number of entries available in the contouring queue. The response is surrounded by line feed and carriage return pairs.

QUEUE REQUIREMENTS	
MODE	
AX - AS	Immediate
AA,AM	Immediate
AA/CD	Immediate

Enter:	AA CD0,0; RQ
Response:	<LF><CR>7160<LF><CR>

5.22. COMMAND SUMMARY

The following commands are included in the VME58 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.

SUMMARY OF COMMANDS IN CHAPTER 5

COMMANDS	SECTION PAGE NUMBER	COMMAND DESCRIPTION
AA	2	Axes all, any following commands are for the AA (All Axes) mode
AC#	21	Acceleration, set acceleration/deceleration register
AF	17, 77	Auxiliary off, turn off selected auxiliary output ports
AM	3	Axes multitasking, enable multitasking mode
AN	16, 77	Auxiliary on, turn on selected auxiliary output ports
AR	6	Axis R, any following commands are for the R axis
AS	7	Axis S, any following commands are for the S axis
AT	5	Axis T, any following commands are for the T axis
AU	5	Axis U, any following commands are for the U axis
AV	6	Axis V, any following commands are for the V axis
AX	3	Axis X, any following commands are for the X axis (default on reset)
AY	4	Axis Y, any following commands are for the Y axis
AZ	4	Axis Z, any following commands are for the Z axis
BH#	19, 78	Bit high, set selected I/O bit high (off)
BI	14	Bipolar, set the analog and PWM torque outputs to bipolar
BL#	19, 78	Bit low, set selected I/O bit low (on)
BX	20, 52	Bit request in hex, return bit status in hex format
CA	46	Clear axis done flag, clear done flag of currently addressed axis
CD#,#;	79	Contour define, enter contour definition mode
CE	80	Contour end, end contour definition and ramp to a stop
CK	80	Contour end and kill, immediately stop step pulses (no ramp down)
CN	12	Cosine on, enable cosine velocity profiles
CR#,#,#	81	Circular interpolation, move in a circle
CV#	81	Contour velocity, allow specification of contouring velocity
CW	40	Clear while, terminate WH/WG loop
CX	82	Contour execute, execute previously entered contour sequence
EA	72	Encoder status, return encoder status of currently addressed axis
EF	8	Echo off, turn off echo to host (default at power up)
EN	8	Echo on, turn on echo to host
ER#,#	63	Encoder ratio, set encoder count to motor count ratio

SUMMARY OF COMMANDS IN CHAPTER 5		
COMMANDS	**SECTION PAGE NUMBER**	**COMMAND DESCRIPTION**
ES#	67	Encoder slip tolerance, set tolerance before slip or stall is flagged
ET	70	Encoder tracking, set encoder tracking mode
FL	33	Flush queue, immediately flush the command queue
FP#	76	Force position, flush queue and attempt to stop at specified position
GD	31	Go and reset done, reset done flags and then initiate previously programmed move
GO	30	Go, start execution of motion
HD#	64	Hold deadband, specify deadband tolerance for position hold
HE	71	Home encoder, set home on encoder logic
HF	65, 69, 70	Hold off, disable position hold, slip detection and tracking modes
HG#	64	Hold gain, specify position maintenance gain parameter
HH	9	Home high, home switches are active high
HL	9	Home low, home switches are active low
HM#	41	Home, find home and initialize the position counter
HN	65	Hold on, enable position correction after move
HR#	42	Home reverse, find home in reverse direction and initialize position counter
HS	71	Home switch, enable home switch mode
HV#	63	Hold velocity, specify maximum position hold correction velocity
IC	46	Interrupt clear, clear done interrupt status and error flags
ID	44	Interrupt done, interrupt host when done and set done flag
II	44	Interrupt independent, interrupt host when each axis finishes a move
IN#	45	Interrupt nearly done, interrupt host when axis or combination of axes nearly complete
IP	45, 66	Interrupt when in position, interrupt is deferred until stage is within specified deadband
IS	68	Interrupt on slip, interrupt host on slip or stall detection
JF#	33	Jog fractional velocities, jog the current axis at fractional rates
JG#	32, 60	Jog, run motor at specified velocity until a new velocity command is sent or it is stopped by a stop or kill command
KA#	60	Acceleration feed forward coefficient
KD#	59	Derivative gain coefficient
KI#	58	Integral gain coefficient
KK#	61	Combined coefficient
KL	35	Kill, flush queue and terminate pulse generation immediately on all axes without decelerating

SUMMARY OF COMMANDS IN CHAPTER 5

COMMANDS	SECTION PAGE NUMBER	COMMAND DESCRIPTION
KM	42	Home and kill, find home and stop pulse generation immediately
KN#	61	Position error summation interval
KO#	60	Offset coefficient
KP#	58	Proportional gain coefficient
KV#	59	Velocity feed forward coefficient
KR	43	Home reverse and kill, find home in reverse and stop pulse generation immediately
LE	37	Loop end, terminate most recent LS command
LF	10	Limits off, disable limit switches for selected axis
LN	10	Limits on, enable limit switches for selected axis
LP#	24	Load position, load position counter with parameter
LS#	36	Loop start, set loop counter, from 1 to 32000 loops; (may be nested to 4 levels)
MA#	25	Move absolute, move to absolute position
ML#,#;	27	Move linear, move specified distance relative from current position
MM	74	Move minus, set minus direction for MV type move
MO	29	Move one pulse, output one step pulse in the current direction
MP	74	Move positive, set positive direction for MV type move
MR#	26	Move relative, move specified distance from current position
MT#,#;	28, 82	Move to, move to specified position
MV#,#	75	Move velocity, move to first parameter (absolute position) at second parameter velocity without stopping at end of move
PA#	18	Power automatic, turn power on before each move and off after the move
PF	13	Parabolic off, disable parabolic ramps, i.e. linear ramps will be generated
PI#	13, 62	PWM Period, set the period for the PWM output signal
PN#	12	Parabolic on, enable parabolic ramps
QA	54	Query axis, query status of switches and flags for addressed axis without affecting flags
QI	55	Query interrupt status, query status of switches and flags on all axes without affecting flags
RA	53	Request axis status, return status of switches and flags and reset flags
RB	20	Request bit direction, return programmed direction of I/O bits in hex format
RC	55	Request acceleration, return current acceleration or deceleration of the current axis
RE	73	Request encoder position, return current encoder position
RI	54	Request interrupt status, return status of switches and flags for all axes and reset flags

SUMMARY OF COMMANDS IN CHAPTER 5

COMMANDS	SECTION PAGE NUMBER	COMMAND DESCRIPTION
RL	69	Return slip status, return slip detection status of each axis
RM#	29	Remainder, return remainder of position divided by parameter in position counter
RP	51	Request position, return current position
RQ	52, 83	Request queue status, return number of queue entries available
RS	15	Reset, software reset of VME58
RU	56	Report position in user units, return current position in user units
RV	56	Request velocity, return current velocity at which the axis is moving
SA	34	Stop all, flush queue and stop all axes with deceleration
SD	35	Stop and reset done, stop all axes and clear any done flags
SE#	18	Settling time, set settling time before power is reduced in PA mode
SF	11	Soft limit off, restore normal overtravel operation
SL	11	Soft limit, allow pulse train to ramp down on overtravel
SP#	76	Stop at position, stop at specified position if possible after all commands have been executed
ST	34	Stop, flush queue and decelerate to stop
SW#	48	Sync wait, wait for the input bit to be released by other controllers
TF	68	Slip tolerance kill off, disable the TN command
TN	67	Slip tolerance kill on, flush the command queue, terminate motion and disable position maintenance
UF	57	User off, turn off user unit translation
UN	14	Unipolar, set the analog and PWM torque outputs to unipolar
UU#	57	User units, multiply acceleration, velocity and distance parameters by specified parameter
VB#	23	Velocity base, set base velocity
VL#	22	Velocity, set maximum velocity to be used in profile
WA	47	Wait for axes, wait until all moves on all axes are finished
WD	38	While end, WS loop terminator
WG	40	While flag, terminate WH loop
WH	39	While, execute all commands until WG loop terminator, until flag cleared by CW command
WQ	47	Wait for queue to empty, wait until current axis queue is empty
WS#	38	While sync, execute while sync is true
WT#	49	Wait, wait for specified number of milliseconds
WY	50	Who are you, returns model and software revision

This page intentionally left blank

6. HOST SOFTWARE

6.1. INTRODUCTION

Host software support for the VME58 is available. Please call Oregon Micro Systems, Inc. at 503-629-8081 for assistance.

This page intentionally left blank

7. SERVICE

7.1. USER SERVICE

The VME58 family of controllers contain no user serviceable parts.

7.2. THEORY OF OPERATION

The 68332 microprocessor on the VME58 controllers maintains four concurrent processes. The highest priority process calculates the desired velocity information at regular intervals which are model dependent. The position error is computed and applied to a PID filter in the servo models to determine the torque command. On stepper axes a pulse train is developed to send to the driver module. Synchronization of each axis is also handed by the 68332.

The commands are written by the host computer into a dual port RAM on the VME58. This memory may be mapped by the host into any desired memory block. The command is then parsed and executed immediately or routed to separate command queues for each axis. The command queue contains a list of addresses to execute followed by optional parameters. A command from the host may be expanded into several commands to the appropriate axis. Position, velocity and other motion related variables are written to the dual port RAM. These variables may be then read by the host at any time providing feedback of the move status.

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:	http://www.omsmotion.com
3. Telephone:	8:00 a.m. - 5:00 p.m. Pacific Standard Time (503) 629-8081
4. Facsimile:	24 Hours (503) 629-0688
5. USPS:	Oregon Micro Systems Inc 1800 NW 169th Place Suite C100 Beaverton OR 97006

RETURN FOR REPAIR PROCEDURES

1. Call Oregon Micro Systems Customer Service at 503-629-8081.
2. Explain the problem and we may be able to solve it on the phone. If not, we will provide you with 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

This page intentionally left blank

INDEX