USER'S MANUAL

INTELLIGENT MOTOR CONTROLLERS

VME44 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

Part Number: E4MNVME44A
Revision A

TABLE OF CONTENTS

1. GENERAL DESCRIPTION

1.1. INTRODUCTION

The VME44 intelligent motor controller can manage 4 axes of motion and monitor each axes' associated incremental encoder in one slot of a VME bus computer. It meets the VME C.1 specification for D08(O) slaves and can be plugged directly into the backplane of these machines. It can manage coordinated or independent motion on each of the four axes simultaneously. The controller can also generate constant velocity profiles with circular interpolation for cutting and machining type operations.

The VME44 functions as a loosely coupled coprocessor within the VME computer. It utilizes a 68000 microprocessor and patented proprietary technology to control direction of motion, acceleration, deceleration and velocity of an associated motor. In response to commands from the host computer, the VME44 controller will calculate the optimum velocity profile to reach the desired destination in the minimum time while conforming to the programmed velocity and acceleration parameters. A block diagram of the VME44 is shown in Section 7.

The VME44 controller uses 'microstepping' techniques for increased position resolution and decreased low speed resonance. When combined with the appropriate driver and step motor, the VME44 can divide the normal step angle into 250 discrete steps of 0.0072 degrees each or 50,000 steps per revolution. The Oregon Micro Systems drivers, models MD10A and MH10 are 10 microstep per step motor drivers which can be driven by the VME44.

Commands may be sent to the VME44 by simple I/O commands using virtually any language on the host VME computer. These controllers are easily programmed with ASCII character strings. For a typical motion requirement of 1,000,000 pulses at 400,000 pulses/sec and an acceleration of 500,000 pulses/sec^2 the following string would be input from the host computer to the VME44:

```
VL400000 AC500000 MR1000000 GO
```

For additional programming examples see Section 5.

1.2. FUNCTIONAL DESCRIPTION

The VME44, in response to commands from the host computer, provides controlled acceleration to a predefined peak speed followed by a constant velocity and controlled deceleration to a stop. This is achieved by calculating the optimum velocity 1024 times each second, providing a very smooth acceleration curve. This calculation is used to control a variable frequency pulse train which is derived from a crystal oscillator providing very accurate pulse rates.

The 68000 microprocessor calculates this velocity profile for each of the axes providing independent but synchronized (if desired) profiles for each axis. The VME44 can perform a smooth coordinated move using linear, parabolic or cosine velocity profiles. It can manage as many as four independent or coordinated processes.

The VME44 will calculate the optimum velocity profile to generate the desired move, while conforming to the acceleration and velocity data input by the host computer. This move will consist of a smooth acceleration, followed by a constant velocity section and a smooth deceleration to the desired distance. A graph of a typical linear velocity is shown in Figure 1-1.

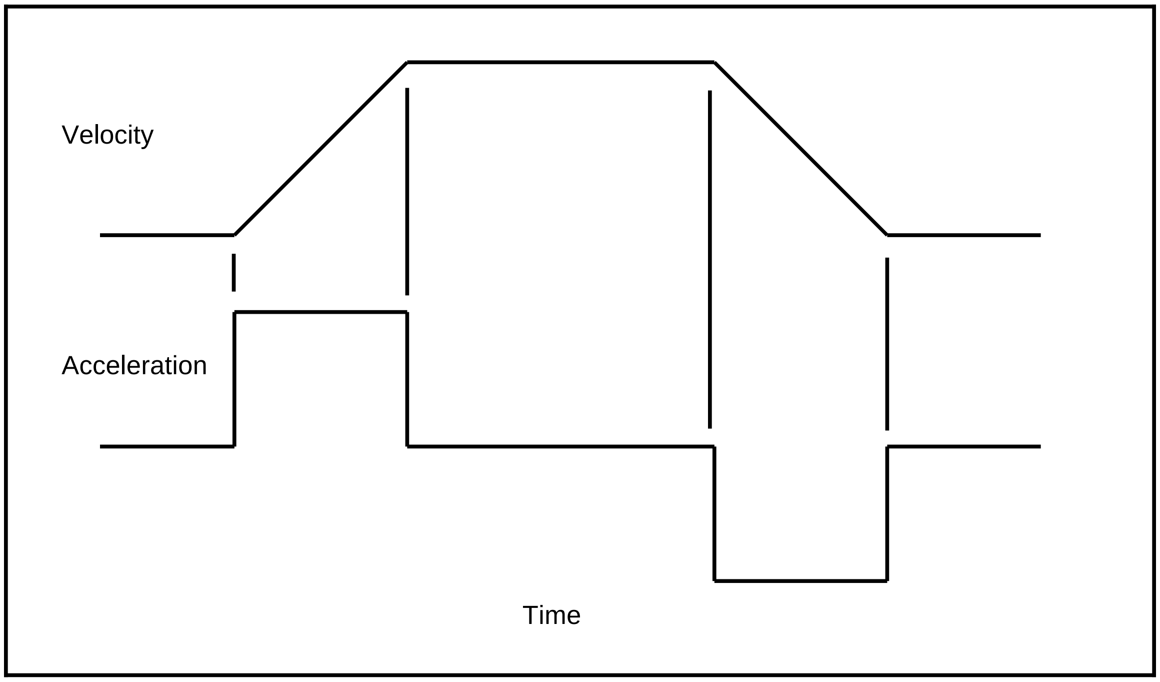

Figure 1-1 TYPICAL VELOCITY PROFILE

If the move parameters do not allow the motor to accelerate to the desired velocity in the desired distance, the VME44 will automatically generate an optimum triangular velocity profile. It can also be commanded to accelerate to a velocity and hold that velocity until told to stop or change to a new velocity. It will then smoothly decelerate to a stop or accelerate/decelerate to the new velocity.

Several moves of this type may be chained together to provide a more complex pattern. The VME44 is able to store up to 124 characters in an input character buffer, plus 200 commands and parameters in separate command queues for each axis, allowing several moves to be made without host intervention. A loop counter is provided to repeat desired sections of a complex move pattern. Loops may be nested up to four levels deep on all axes.

The VME44 has 12 bits of general purpose I/O in addition to the limit and auxiliary I/O associated with each axis. The I/O is user definable and is driven by quad TTL logic gates. The default configuration of the I/O is 8 inputs and 4 outputs that can be used to synchronize a VME44 with another VME44 or another external device. The I/O is generic in that it can be used for almost any purpose.

1.3. VELOCITY PROFILES

The VME44 offers three options for ramping the device to speed. The traditional constant acceleration or linear velocity ramp (see Figure 1-1) is the default at power up or reset. The half sinusoid acceleration or half cosine velocity ramp (see Figure 1-3) is selected by the CN command. Since the acceleration is zero at the velocity inflection points, this offers very smooth operation. It is used in sensitive applications such as wafer handling on a vacuum chuck. The third option is a reverse ramp of acceleration or parabolic velocity curve (see Figure 1-2), which can be selected by the PN command. This ramp is commonly used to compensate for loss of motor torque at high speeds, i.e. since the acceleration is reduced at higher speeds the required forces are reduced proportionally. The parabola may be truncated to allow the user to select, under program control, the reduction in acceleration (force) appropriate for the application.

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

$$v = A_m t$$

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

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

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

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

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

The acceleration follows the equation:

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

and the velocity is then:

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

and the distance traveled in the ramp is:

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

where A_0 is the initial acceleration, t is time during the ramp and T_2 is total ramp time if the acceleration had reached zero. The parameter supplied with the PN command is 10 times the ratio $\frac{t}{T_2}$ which can take on values from 3 to 10, allowing the final acceleration to range from 70% to 10% respectively of the programmed or initial value. When a move is specified, the controls will fit the resulting velocity curve to the desired acceleration profile. This 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 parabolic ramp mode may result in reduced move time at high speeds, since a larger acceleration may be used.

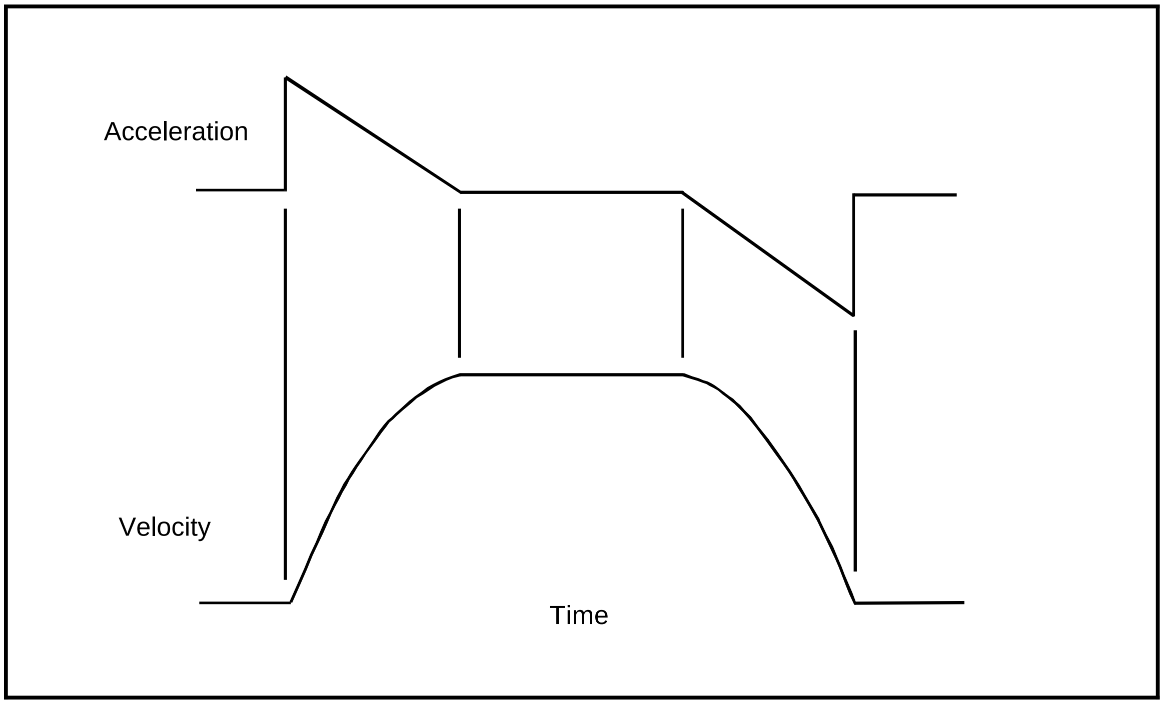

Figure 1-2 PARABOLIC VELOCITY PROFILE

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

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

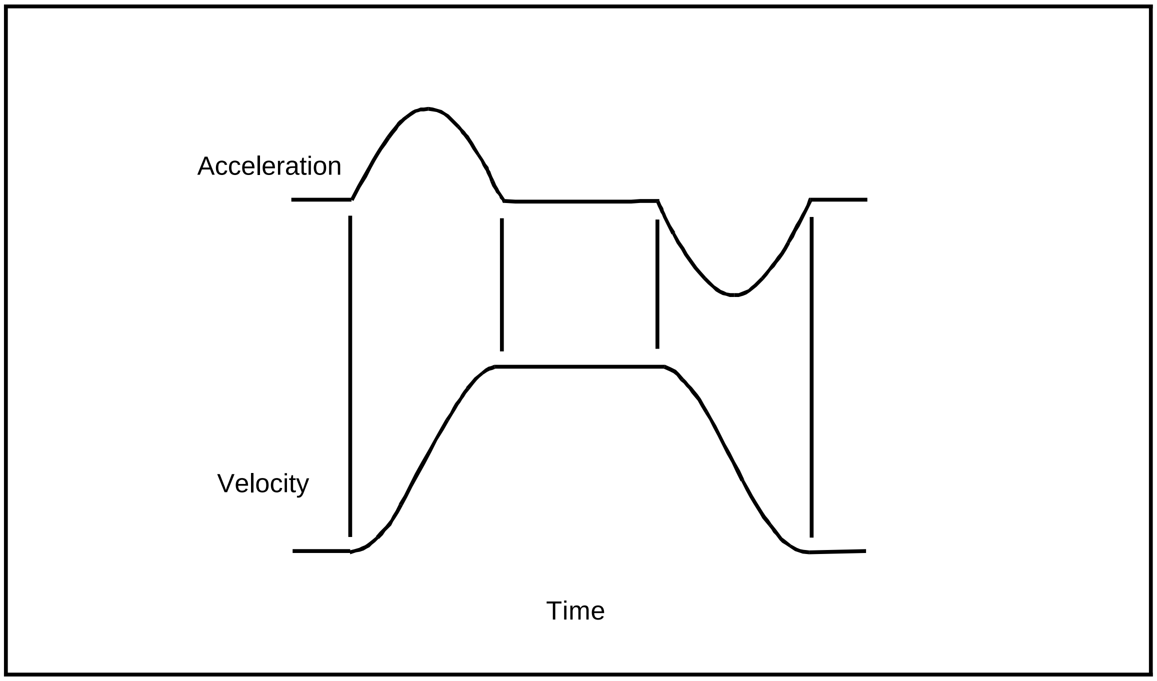

Figure 1-3 COSINE VELOCITY PROFILE

and the velocity is then:

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

and the distance traveled in the ramp is:

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

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

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

and the time required to ramp up is:

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

and the peak velocity is:

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

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

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

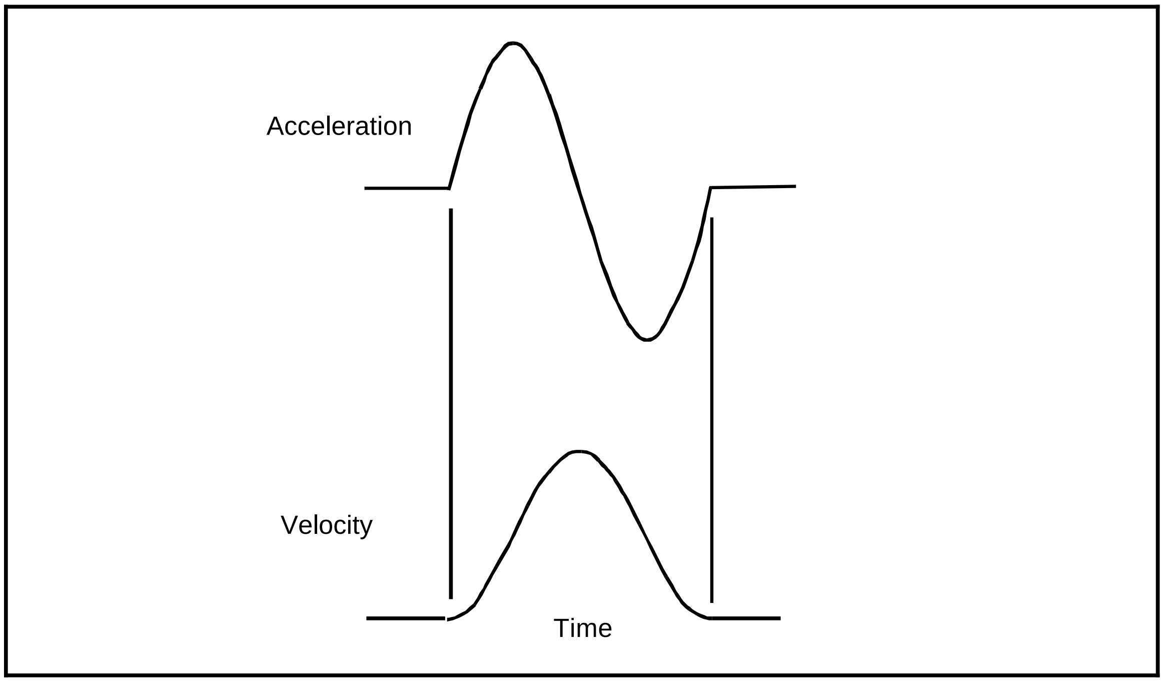

Figure 1-4 SHORT MOVE COSINE VELOCITY PROFILE

2. GETTING STARTED

2.1. INTRODUCTION

The VME44 board requires one full width slot in the VME card cage. In most cases the jumpers on the VME44 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 16 contiguous address from FF80 to FF8F 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 VME44 board.

2.2. JUMPERS

There are nine blocks of square pin jumpers on the VME44 board. These can be thought of as 5 logical groups: board address selection jumpers, J25 and J26; interrupt selection jumpers, J15 and J35; encoder bias jumpers, J11 and J14; limit polarity jumper, J24; and I/O control and pull-up jumpers, J32 and J34. See Figure 3-1 for the locations of the jumpers. Recommended jumpers are Molex part number 90059-0007 or equivalent.

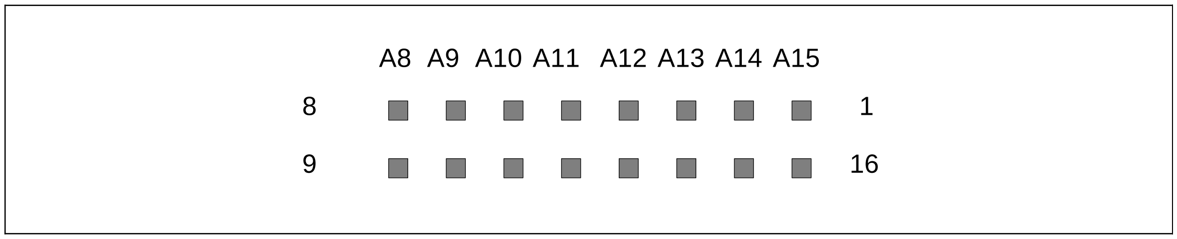

Figure 2-1 J25 ADDRESS SELECT 1 (default setting)

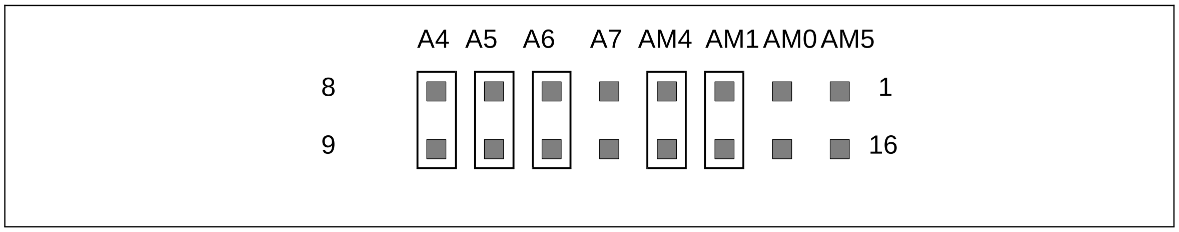

Figure 2-2 J26 ADDRESS SELECT 2 (default setting)

2.3. ADDRESS SELECTION

The VME44 uses a 16 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 A3 are decoded by the VME44 board and for base address selection are assumed to be 0. To set the board for the factory default address of FF80, jumpers for address lines A15, A14, A13, A12, A11, A10, A9 and A8 on J25 and A7 on J26 should be absent, selecting a 1 for those lines. Jumpers should be placed on A6, A5 and A4 on J26, selecting a 0 for those lines.

The VME44 allows selection from several address modifier values. Lines AM0, AM1, AM4 and AM5 are user selectable by square pin jumpers on J26. 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-Priveleged Access to the VME44 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

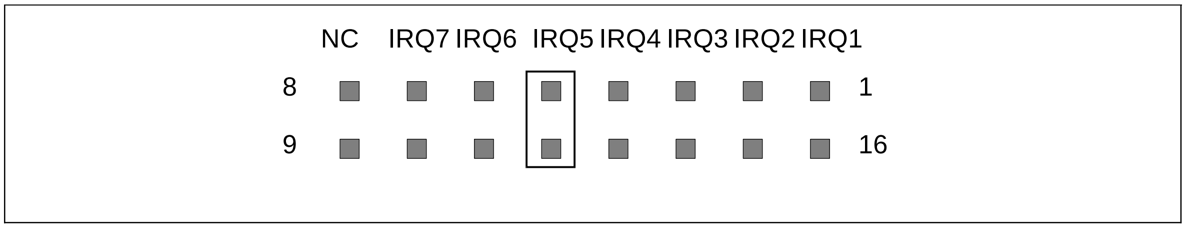

Figure 2-3 J15 INTERRUPT SELECT 1 (default setting)

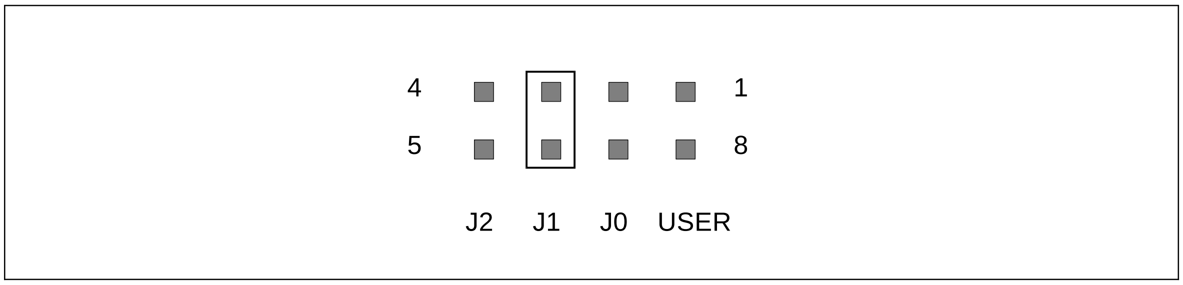

Figure 2-4 J35 INTERRUPT SELECT 2 (default setting)

Two sets of jumpers are used to select which VME bus interrupt signal is used by the VME44 board, J15 and J35. J15 selects to which interrupt request line the VME44 board is wired and J35 selects on-board logic to properly generate the interrupt.

To jumper the board to a given interrupt, place a square pin jumper across the appropriate pin pair on J15, then using the binary equivalent of that interrupt number, place jumpers on J35, 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 J15. Next, figure the binary equivalent of 5, which is 101. This means that J2 (pin pair 3 and 6) of J35 should have no jumper since bit 2 of 101 is a 1, J1 (pin pair 4 and 5) should have a jumper since bit 1 is a 0, and J0 (pin pair 2 and 7) should have no jumper since bit 0 is a 1.

2.5. LIMIT POLARITY SELECTION

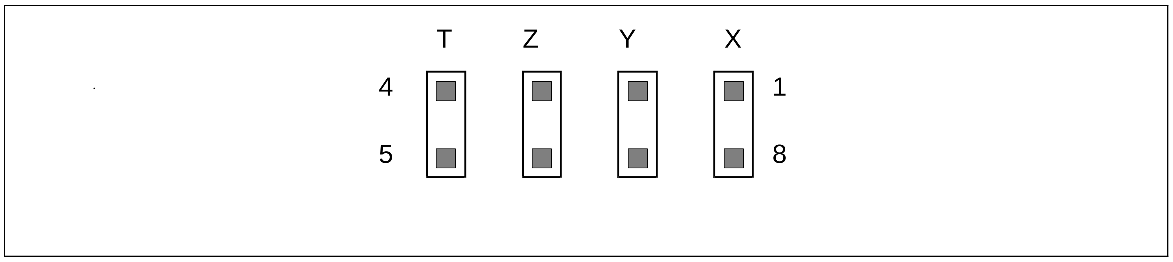

Figure 2-5 J24 LIMIT POLARITY

J24 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 +5VDC. These lines 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 J24 set the limit inputs to active low. This requires jumpers on all four pairs of pins.

2.6. ENCODER BIAS SELECTION

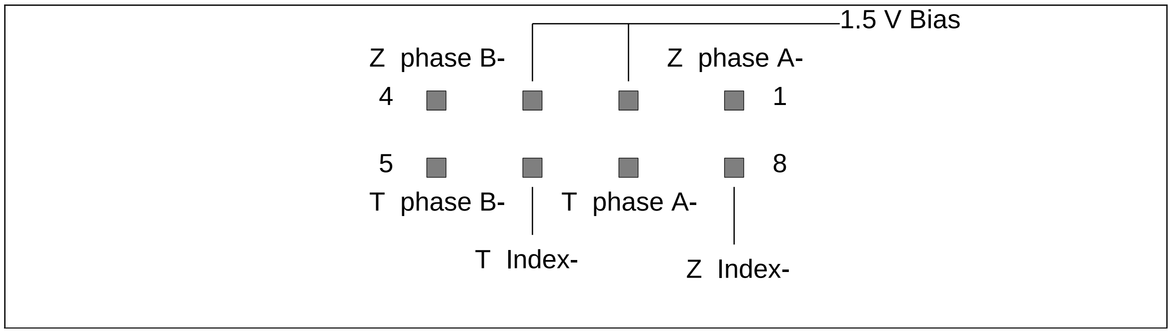

Figure 2-6 J11 T&Z AXES ENCODER BIAS (default setting)

The VME44 uses MC3486 type differential line receiver ICs to sense signals from the encoders. If the encoder used does not have differential outputs, the minus side of the encoder inputs to the VME44 must be biased at +1.5VDC to ensure proper signal reception. This is accomplished by adding wire wrap jumpers to the appropriate pins on J11 or J14. J11 supplies the bias for the Z and T axes and J14 supplies the bias for the X and Y axes. See Figures 2-6 and 2-7 for exact pin numbers. The factory default settings for these two jumper blocks are to have no jumpers on them.

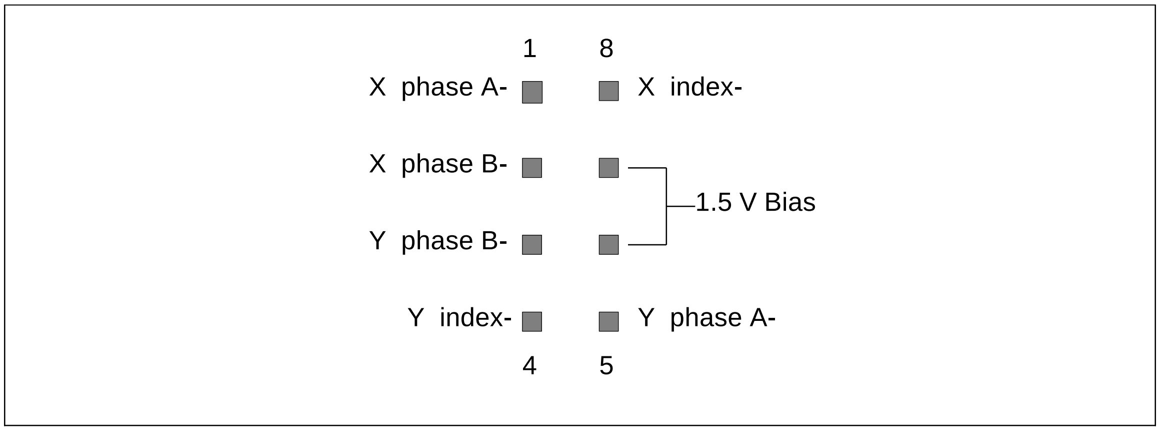

Figure 2-7 J14 X&Y AXES ENCODER BIAS (default setting)

2.7. I/O AND PULLUP SELECT

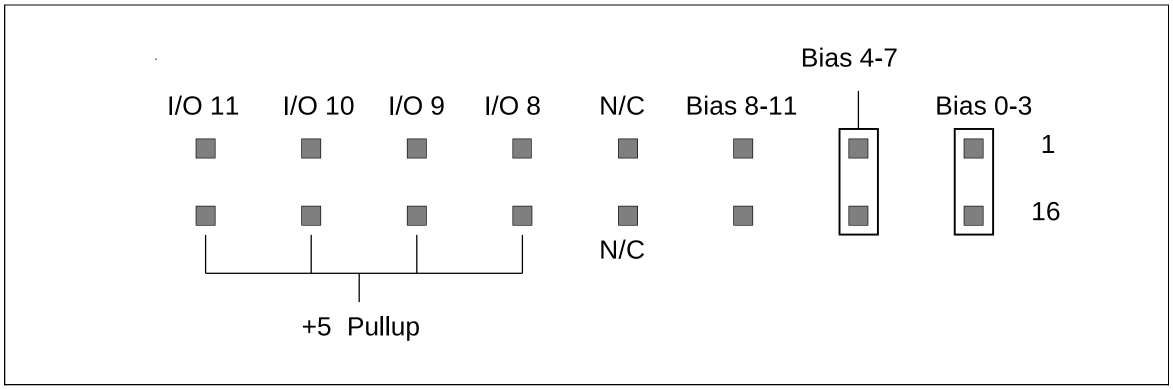

Figure 2-8 J32 I/O AND PULLUP SELECT (default setting)

The VME44 board has 12 user definable I/O lines available. The user can configure these, in four bit groups, as either input or output. This requires changing ICs on the board and changing the jumper settings on J32 and J34. The factory defaults have I/O8 through I/O11 as outputs and I/O0 through I/O7 as inputs. This requires jumpers on pin pairs 1-16 and 2-15 to enable this configuration of "ins" and "outs".

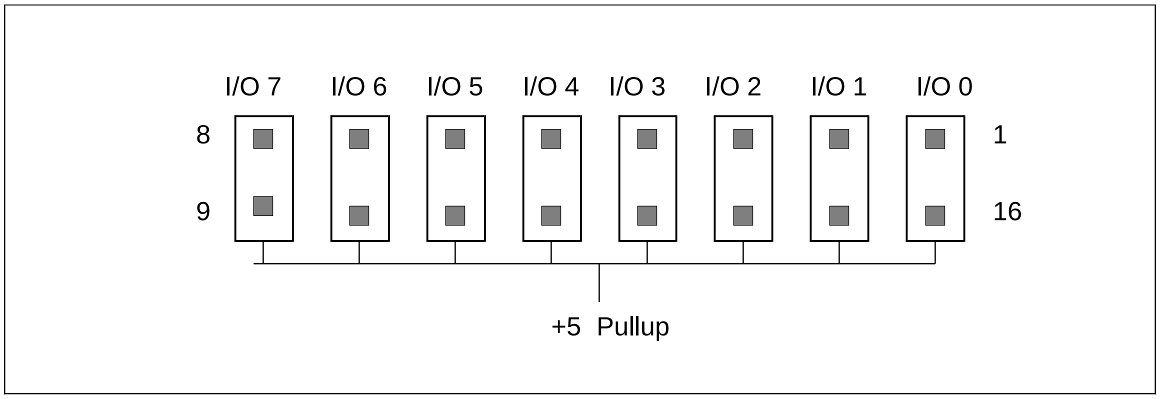

Figure 2-9 J34 PULLUP SELECT

All inputs have 2.2k Ohm resisters pulling them to +5VDC. This is done by installing square pin jumpers on all 8 pin pairs on J34.

The user I/O lines use some of the row B pins of the P2 connector. These lines are also used by controllers that use 32 bit addressing modes. If your controller uses row B of P2, you will have to disable the user I/O on the VME44 board. This is done by removing the pull-up jumpers from J32 and J34 and also removing the I/O output driver ICs U31, U33, and J34.

2.8. MOTOR CONTROL CONNECTOR

The motor control connector (P2) on the VME44 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 and encoder inputs. Row B has the user I/O lines on it (pins 14 through 26).

The motor control lines can be considered as 4 logical sets of 8 pins. Each set is used for an individual axis. The 8 pins of an axis set are: Step Output, +5VDC Output, Auxiliary Output, Direction Output, Negative Limit Switch Input, Ground, Home Switch Input and Positive Limit Switch Input.

The encoder inputs use two sets of 8 pins on both ends of the connector as encoder inputs for the X, Y, Z and T axes. The 8 pins of an encoder pin set are: Index +, +5VDC, Phase A-, Phase A +, Index -, Ground, Phase B -, Phase B +.

See Section 4 for more information on the motor control connector.

2.9. HARDWARE INSTALLATION

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

2. Remove the computer’s cover.

3. Choose an empty expansion slot in the VME rack and backplane and remove its associated metal cover if one exists.

4. Slide the VME44 board into the rack and connector, insuring the board is lined up correctly in the card guides and in the connector.

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

6. Screw the front panel of the VME44 board into the card cage.

7. Replace the cover of the computer.

8. Replace the power cord and turn the computer on.

9. Allow the computer to boot up.

10. To ensure that the VME44 is set up for the proper address, try to read and write a byte to the VME44's interrupt vector register at short address FF87 (default). See if the value written to it is the same one read back.

11. Using your favorite debug utility, output the following two bytes to the short address FF81 (the VME44 data register default address): 0x57 0x59. These are the hexidecimal equivalents to W and Y. The VME44 board will respond by outputting its board name and firmware version to the data register. If you read the input from the data register (FF81) one byte at a time, you should get the hex equivalent to the ASCII characters:

    ```
    VME44 ver 1.76-4E
    ```

    Which are: 0A 0D 56 4D 45 34 34 20 76 65 72 20 31 2E 34 35 2D 34 45 0A 0D. If you do, the board is working correctly. If not, recheck the board's jumpers to ensure that they are correct. If the board still fails to work contact Oregon Micro Systems, Inc. for further assistance.

12. Connect the motor drivers and encoders to P2 (see Section 4).

13. Send motion control commands to the VME44 board and see that the motors move. If not, double check your wiring, ensuring that everything is properly connected.

3. VME BUS INTERFACE

3.1. VME BUS

The VME bus specification allows for a number of different interface complexity options. The VME44 supports the D08(O) short address in either supervisory or non-privileged mode as specified by the VME specification C.1. The base address is jumper selectable to allow positioning the board anywhere 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 specifications. The VME44 does not support either, but uses some of these pins as optional I/O. If a extended addressing is used, the user I/O must be disabled by removing the input and output chips from their sockets.

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 VME44 supports only 8 bit odd address 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 VME44 has an on board reset timer and thus uses this signal only to initiate this timer on power up or during a system reset.

3.2. BOARD ADDRESS SELECTION

The VME44 occupies a block of 16 contiguous addresses. The factory default address is FF80 through FF8F hex in the short address space. Refer to Figure 3-1 for configuration of jumpers.[1]

Table 3-1 VME BUS P1 PIN LIST

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

* low-level active indicator

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

The actual address is chosen by jumpers on J25 and J26. Connecting a jumper selects a binary 0 for that address bit, while no jumper selects a binary 1. The factory default base address of FF80 (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 3-2 shows the detail of the level select jumpers.

Table 3-2 INTERRUPT LEVEL SELECTION JUMPERS

LEVEL	J15 PINS	J35 PINS		
IRQ1	1 - 16		3-6	4-5
IRQ2	2 - 15	2-7		4-5
IRQ3	3 - 14			4-5
IRQ4	4 - 13	2-7	3-6	
IRQ5	5 - 12		3-6	
IRQ6	6 - 11	2-7		
IRQ7	7 - 10			

3.4. VME44 REGISTERS

The VME44 occupies 16 contiguous addresses in I/O space but is responsive to only the odd addresses in this space. The registers associated with each address are described in the following sections.

Table 3-3 VME44 REGISTER DESCRIPTION

ADDRESS OFFSET	FACTORY DEFAULT	DESCRIPTION
1	FF81 Hex	Data Register
3	FF83 Hex	Done Flag Register
5	FF85 Hex	Control Register
7	FF87 Hex	Status Register
9	FF89 Hex	Interrupt Vector
11	FF8B Hex	Unused
13	FF8D Hex	Unused
15	FF8F Hex	Unused

3.4.1. DATA REGISTER

The data register is the data communication port between the VME44 and the VME bus master. All data is passed between the two processors through this port. This port is double buffered in both directions allowing data to be written to the port before the previous byte has been read and processed. This allows for faster processing of the data between the two microprocessors.

3.4.2. DONE FLAG REGISTER

The done flag register is a read only register from the VME44. The status bit indicating the done status of each axis is written by the 68000 on the VME44. The host can then read it at any time to determine the status of the motion process. It is cleared by reading the register. The bit definition is shown in Table 3-4:

Table 3-4 DONE REGISTER STATUS BITS

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	Unused
5	Unused
6	Unused
7	Unused

3.4.3. INTERRUPT CONTROL REGISTER

The interrupt control register allows different interrupt sources from the VME44 to be individually enabled or disabled. A write to this register will change the enabled interrupts at any time. The register may be read back to verify or determine the state of the interrupts at any time.

Table 3-5 CONTROL REGISTER BIT DEFINITION

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

3.4.4. STATUS REGISTER

The status register is a read only register that provides status information to the host CPU. This status is independent of the enable status of the interrupt, allowing the board to operate in a polled mode, if desired.

In order to resolve the source of a done or error interrupt, the DON_S bit (bit 4) should be read first. This bit in the status register is automatically reset upon reading the status register. If the DON_S flag is true, the error bits should be read to determine if the interrupt was caused by an error condition. If no error condition is present, the done flag register can be read to determine which axes are done. An “IC” command (or Control-Y is preferred) should then be given from within the interrupt service routine or poll routine to reset the done flag register and error bits. The TBE_S bit is reset by writing to the data register and the IBF_S bit is reset by reading the data register. Both bits may remain true after a single read or write since the data port is double buffered and may still remain ready for data.

The error bits, CMD_S, ENC_S and OVRT may also be cleared by sending a Control-X (ASCII value 0X18). This will reset these bits in the status register without altering the state of the done flags. The INIT bit only goes high when the board is initializing and cannot communicate. It will go low and remain low when initialization is complete.

Table 3-6 STATUS REGISTER BIT DEFINITION

BIT	NAME	DESCRIPTION
7	IRQ_S	Interrupt request status. Bit 7 is the logical OR of bits 4 through 6. It is high when an interrupt is being requested or indicates a request for service in a polled mode.
6	TBE_S	Transmit buffer empty status. This high true bit indicates a character may be written to the transmit buffer.
5	IBF_S	Input buffer full status. This high true bit indicates a character is available in the input buffer.
4	DON_S	Done or error status. This high true bit indicates the command is complete (i.e. an ID command has been executed) or an error has been detected. If bits 0-3 are all false it indicates a command completion. Error bits indicate one or more errors have been detected.
3	OVRT	Overtravel. An overtravel switch was true indicating attempted travel out of bounds.
2	ENC_S	Encoder request. This bit indicates that the encoder option is requesting service.
1	INIT	High during reset.
0	CMD_S	Command error. An unrecognizable command has been detected or LS and LE commands are not in matched pairs.

3.4.5. 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 any time. It will be returned during an interrupt acknowledge when this VME44 board is the highest priority board in the system at the selected priority level.

3.5. POWER SUPPLY REQUIREMENTS

The VME44 is designed to operate from the power supplied in the VME bus backplane. The host computer or expansion box must be capable of supplying 2.43 amps typical for operation of the VME44 board.

CAUTION:

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

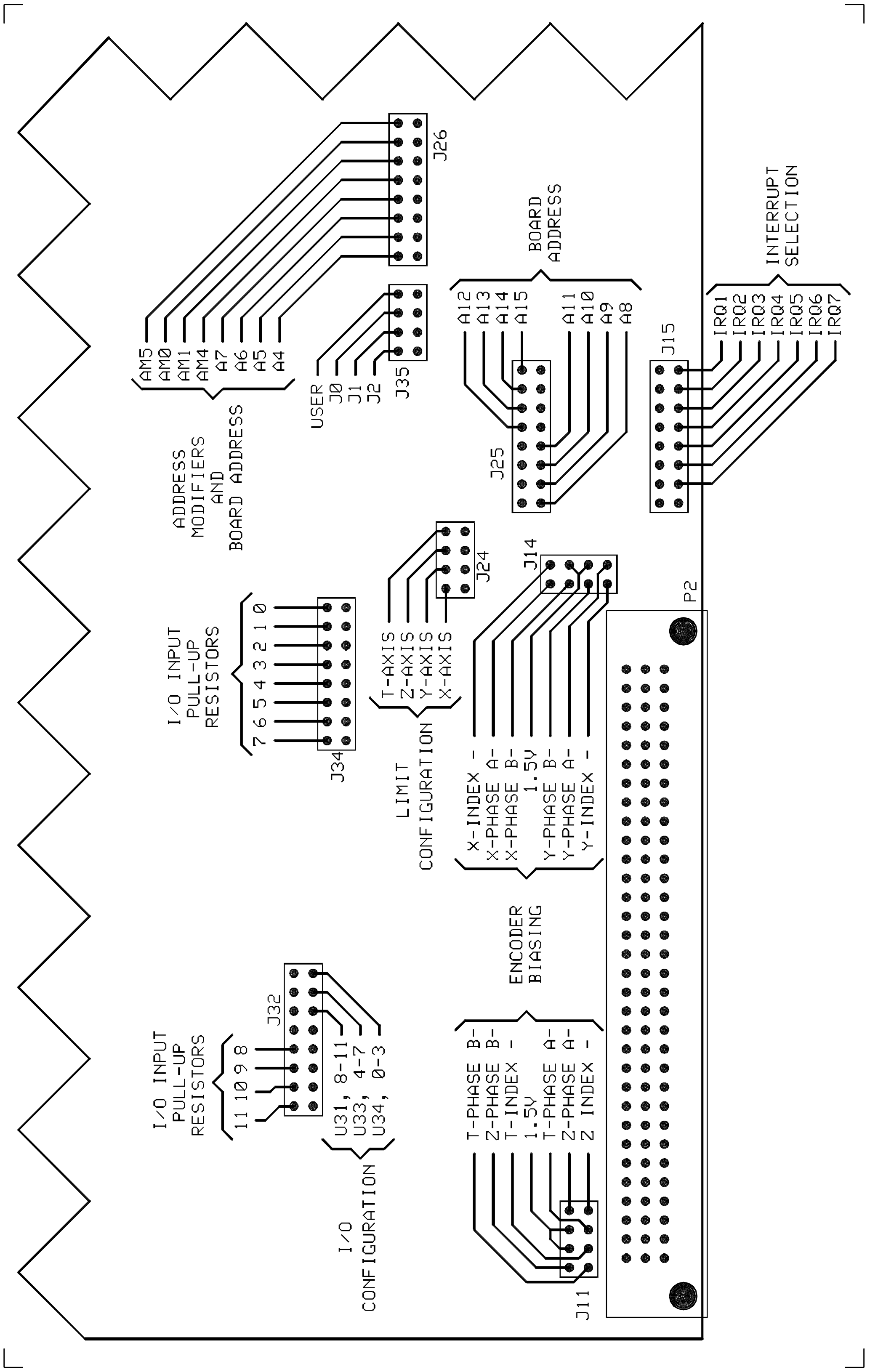

Figure 3-1 VME44 JUMPER LOCATION

4. DRIVER and ENCODER INTERFACE

4.1. OUTPUT CONNECTION

Table 4-1 lists the input and output interface signals available at output connector P2 on each VME44 board. The table is shown as seen looking at the end of the connector.

A separate 4-conductor shielded cable should be used for each axis for the connections to its associated driver module and must be limited to 50 feet. A connection to the VME bus +5VDC power is provided for each axis to supply power to the emitter diode within an opto-isolated motor driver module, such as the MH10. This allows the use of such drivers without the need for an external power supply.

CAUTION:

This power supply connection must not be connected to any other supply or used for any other purpose or damage may result to the host computer or VME44 or both.

A ground connection is provided for each axis for convenience in connecting up the system. The VME44 is supplied with 7406 open collector TTL drivers as standard. These parts are in sockets and can be replaced with 7404 totem pole drivers for driver modules which do not have opto-isolated inputs. Each device handles the step, direction and auxiliary output for two axes. The cable shields should be connected to the appropriate ground pins, as shown in Table 4-1, and left open at the driver end when used with opto-isolated loads to avoid ground loops and ensure isolation.

4.2. MULTI-AXIS SYNCHRONIZATION

Each VME44 has provision for synchronizing several VME44 boards to drive larger systems. Systems requiring more than 4 axes and thus more than one VME44 can be synchronized by connecting an auxiliary or I/O output on one board to the I/O input on the other board. The boards can signal each other at the appropriate place in the command stream without interrupting the host computer. Synchronization can be accomplished with other devices as well.

4.3. LIMIT AND HOME LINES

The limit and home lines can be activated using mechanical switches using contact closures or other suitable active switches, such as a hall effect switch or opto-isolator, that

Table 4-1 VME BUS P2 PIN ASSIGNMENTS

PIN	ROW C FUNCTION	ROW B FUNCTION	ROW A FUNCTION
1	X Index + Input		+5VDC
2	X Phase A- Input		X Phase A+ Input
3	X Index - Input		Ground
4	X Phase B- Input		X Phase B+ Input
5	Y Index + Input		+5VDC
6	Y Phase A- Input		Y Phase A+ Input
7	Y Index - Input		Ground
8	Y Phase B- Input		Y Phase B+ Input
9	X Step Output		+5VDC
10	X Auxiliary Output		X Direction Output
11	X Negative Input		Ground
12	X Home Input		X Positive Limit Input
13	Y Step Output	+5VDC	+5VDC
14	Y Auxiliary Output	I/O-0	Y Direction Output
15	Y Negative Limit Input	I/O-1	Ground
16	Y Home Input	I/O-2	Y Positive Limit Input
17	Z Step Output	I/O-3	+5VDC
18	Z Auxiliary Output	I/O-4	Z Direction Output
19	Z Negative Limit Input	I/O-5	Ground
20	Z Home Input	I/O-6	Z Positive Limit Input
21	T Step Output	I/O-7	+5VDC
22	T Auxiliary Output	Ground	T Direction Output
23	T Negative Limit Input	I/O-8	Ground
24	T Home Input	I/O-9	T Positive Limit Input
25	Z Index + Input	I/O-10	+5VDC
26	Z Phase A - Input	I/O-11	Z Phase A + Input
27	Z Index - Input		Ground
28	Z Phase B - Input		Z Phase B + Input
29	T Index + Input		+5VDC
30	T Phase A - Input		T Phase A + Input
31	T Index - Input		Ground
32	T Phase B - Input		T Phase B + Input

connect the line to ground. The limit switch closure will stop the associated pulse stream if the motor travels beyond its allowable limits and trips the switch. The home switch provides a means to synchronize the motor controller with the load at some home or reference position. The home switch, when used with the software HM command, will cause the motor to stop when the switch closes. On finding the home position the internal position counters will be initialized. The sense of the home switches may be changed to true when open, if desired, by use of the HH command. The limit switches may be changed to true when open, if desired, by removing the jumper on J24. Figure 4-1 shows a typical connection between a VME44 board and a motor using an OMS MH10 motor driver.

4.4. I/O CONFIGURATION

The user definable I/O is driven by TTL quad logic gates. A 7400 NAND gate or a 7438 open collector NAND gate drives four outputs while a 74LS02 NOR gate buffers four inputs. These ICs are installed in sockets for convenience and are located above the connector P2. They are U31, U33 and U34. To change I/O bits 0 through 3 from inputs to outputs U34 must be changed from a 74LS02 to a 7400, J34 pins 1 through 4 must have the jumpers removed to remove the pull-up resistors and J32 pin 1 must be removed to flag these pins as outputs to the VME44. To change from outputs to inputs the process is reversed. The default configuration of the I/O has bits 0 through 7 as inputs and 8 through 11 as outputs.

J32 pins 1 through 3 flag the VME44 which bits of the I/O are configured as inputs and outputs. Pin 4 of J32 is unused and should be ignored. Pins 5 through 8 on J32 are used to select pull-up resistors on I/O bits 8 through 11. J34 pins 1 through 8 select pull-ups for I/O bits 0 through 1 respectively. A jumper in place selects a pull-up resistor for the corresponding I/O bit.

The user I/O lines use some of the middle lines of the P2 connector. These lines are also used by controllers that use 32 bit addressing modes. If your controller uses row B of P2 you will have to disable the user I/O on the VME44 board. This is done by removing the

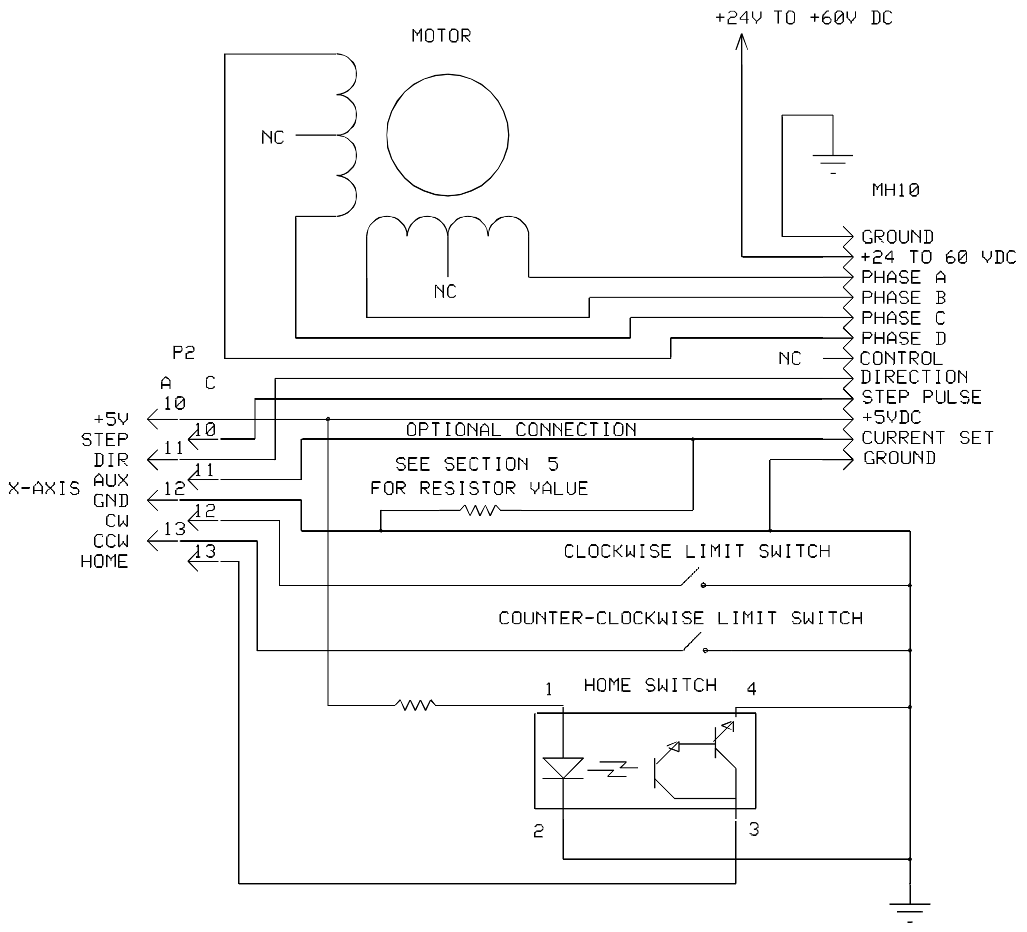

Figure 4-1 VME44/MH10 INTERCONNECTIONS

pull-up jumpers from J32 and J34 and also removing the I/O output driver ICs U31, U33, and U34.

4.5. ENCODER INTRODUCTION

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

The encoder accepts quadrature pulse outputs from high resolution optical encoders. Up to 50,000 pulse per revolution encoders may by used while the indexers are generating pulses at their maximum rate. This allows position feedback information to match the resolution to the microstepping motor drive.

4.6. MODES OF OPERATION

The VME44 can monitor the actual position through the encoder pulse train. It can then correct for position errors due to system backlash or mechanical tolerances or report slip or stall of the motor to the host. A tracking mode is also provided which allows one axis to track the activity of another axis or positioning device. These options are selectable by the user through software commands.

4.7. ENCODER SELECTION AND COMPATIBILITY

The VME44 is compatible with virtually any incremental encoder which provides quadrature outputs. Times four quadrature detection is used to increase resolution. The inputs are compatible with encoders which have single ended TTL outputs as well as differential line drivers. Provisions are also provided for an index pulse (differential or single ended) and an index enable for systems requiring more than one revolution of travel and thus multiple index pulses from the encoder. A biasing network is provided on the board for termination of unused encoder inputs.

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

4.8. ENCODER INTERFACE

The encoder connections are as shown in Table 4-1.

If single ended encoders are used, the unused line receiver inputs must be biased in the middle of the voltage swing of the active output. J11 and J14 are provided with a built in bias supply. The appropriate unused inputs should be connected to the +1.5VDC supply as needed.

4.9. HOME PROCEDURES

Two logical inputs are provided to synchronize the physical hardware with the VME44, i.e. put the controlled motor in the home position.

The VME44 home inputs can be used with encoders which provide one home pulse for the complete travel of the stage. This signal can be either a logic high or logic low true by using the HH and HL commands. The HM or HR commands are used after reducing the velocity to no more than 1024 pulses per second. This limit on velocity is necessary to avoid ambiguity of the home position if more than one pulse occurs per sample interval.

The index input on VME bus P2 uses internal logic to establish the home position when used with the HE command mode. This position consists of the logical AND of the encoder index pulse, the home enable external input (low true only) and a single quadrant from the encoder logic. The home enable pulse must be true for less than one revolution of the encoder thus allowing only one home for the complete travel of the stage. The home logic expressed in boolean terms is:

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

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

5. COMMAND STRUCTURE

5.1. INTRODUCTION

An extensive command structure is built into the VME44 intelligent motor control. It includes a 200 command and parameter buffer for each axis and a command loop counter which allows multiple executions of any command string.

The following commands in this section are included in the VME44 controller. 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#;. 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 VME44 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 VME44 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 4 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. A Control-D may clear this condition if the input character queue is not also filled since it bypasses the command interrupter.

The following commands are available in firmware revision 1.76 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 - AT	Immediate
AA,AM	2
AA/CD	Not valid

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

Enter: AA MR12000,14000; 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 REQUIRE-MENTS	
MODE	
AX - AT	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 - AT	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 - AT	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 - AT	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 - AT	Immediate
AA,AM	Immediate
AA/CD	Not valid

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

Enter: AT MA-2468; GO

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 - AT	Immediate
AA,AM	Not valid
AA/CD	Not valid

Example: Enable echoing by the VME44 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 VME44 motion system. This is the default mode at power up or reset.

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

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 - AT	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 - AT	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 steps. It must be in the range of 3 to 10 corresponding to 70% and 10% acceleration at the peak respectively. A parameter out of this range or no parameter supplied defaults to 70% or 3 steps. Note that the parameter is the number of steps, not the acceleration values. The larger number is a lower acceleration at the peak. See Section 1 for an explanation of velocity profiles. This command should not be given while an axis is in motion or the results may not be predictable. This command affects all axes, even if issued in the single axis mode.

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

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

Enter: PF

RS RESET

The RS command is a software reset which causes the local VME44 microprocessor to reset. All previously entered data and commands are lost. All internal parameters are initialized to defaults. All interrupts are disabled. This command is intended for catastrophic failure recovery only. The KL command should be used to reset queues or return the system to a known state.

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

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

Enter: RS

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

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

QUEUE REQUIREMENTS	
MODE	
AX - AT	1
AA,AM	1
AA/CD	2

Example:	Turn on the Y axis auxiliary output in the single axis mode.
Enter	AY AN
Example:	Turn on the X and Z axes auxiliary outputs when in the AA command mode. The Y axis is unchanged in this example.
Enter:	AA AN1,,1;

AF AUXILIARY OFF

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

QUEUE REQUIREMENTS	
MODE	
AX - AT	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 - AT	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 - AT	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 - AT	2
AA,AM	2
AA/CD	Valid

Example: Turn on output bit 10 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

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 11.

QUEUE REQUIREMENTS	
MODE	
AX - AT	2
AA,AM	2
AA/CD	Valid

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

Enter: BH8; BH11;

BX BIT REQUEST IN HEX

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

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

Example: User output bits 10 and 11 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>000C09<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 feeds and carriage returns. Output bits return a 1 while input bits return a 0. The three left hex digits are unused and are always set to 0.

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

Example: Factory default settings have bits 0 through 7 as inputs and 8 through 11 are outputs. Verify this with the RB command.

Enter: RB

Response: <LF><CR>000F00<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 8,000,000. All the following move commands for the axis being programmed will accelerate or decelerate at this rate until another AC command is entered. All acceleration registers default to 2,000,000 steps per second per second upon power-up or reset.

The acceleration register may be automatically modified by the VME44 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 - AT	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 522,000 steps per second. The velocity defaults to 200,000 at power up or reset. This is a write only register and controls the maximum velocity used in relative and absolute position moves except as modified by the linear interpolation instructions.

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

QUEUE REQUIREMENTS			
MODE	**LINEAR**	**PARABOLIC**	**COSINE**
AX - AT	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 - AT	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. 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 the no parameter is supplied, the value of zero is used. This command turns off the position hold and interrupt on slip modes.

QUEUE REQUIREMENTS	
MODE	
AX - AT	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 20000 and the encoder position register with 30000 counts.

Enter: AY ER3,2 LP30000

MA# MOVE ABSOLUTE

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

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

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

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

QUEUE REQUIREMENTS			
MODE	**LINEAR**	**PARABOLIC**	**COSINE**
AX - AT	2	2	2
AA,AM	2	2	2
AA/CD	Not valid		

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

Enter: AX MA100000 GO

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

Enter: AA MA,10000,,1000; GO

MR# MOVE RELATIVE

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

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

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

QUEUE REQUIREMENTS			
MODE	**LINEAR**	**PARABOLIC**	**COSINE**
AX - AT	2	2	2
AA,AM	2	2	2
AA/CD	Not valid		

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

Enter: AX MR-2468 GO

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

Enter: AA MR12345,6789; GO

ML#,#; MOVE LINEAR

The ML command uses linear interpolation to perform a straight line relative move to the new location. Input parameters are relative distance for each axis in the move. Velocity and acceleration parameters of each axis may be automatically adjusted by the VME44 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 VME44 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 - AT	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 axis remains in its previous position.

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

MT#,#; MOVE TO

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

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

QUEUE REQUIREMENTS			
MODE	**LINEAR**	**PARABOLIC**	**COSINE**
AX - AT	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 axis remains in its previous position.

Enter: AA MT1000,10000,,100; GO

MO MOVE ONE PULSE

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

QUEUE REQUIREMENTS	
MODE	
AX-AV	1
AA, AM	Not Valid
AA/CD	Not Valid

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

Enter: AZ MM MO

RM# REMAINDER

The RM command will divide the position counter by the parameter supplied and replace the position counter with the resulting remainder. The parameter must be greater than zero and less then 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 in may have rotated. This command has no effect on the encoder position register.

QUEUE REQUIREMENTS	
MODE	
AX - AT	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. Add the value found in Table B to the value from Table A, to determine total queue usage.

TABLE A	
QUEUE REQUIREMENTS	
MODE	
AX - AT	7
AA,AM	8
AA/CD	Not valid

TABLE B			
ADDITIONAL ENCODER QUEUE REQUIREMENTS			
MODE	**LINEAR**	**PARABOLIC**	**COSINE**
AX - AT	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. Add the value found in Table B to the value from Table A, to determine total queue usage.

TABLE A	
QUEUE REQUIREMENTS	
MODE	
AX - AT	8
AA,AM	9
AA/CD	Not valid

TABLE B			
ADDITIONAL ENCODER QUEUE REQUIREMENTS			
MODE	**LINEAR**	**PARABOLIC**	**COSINE**
AX - AT	0	0	0
AA,AM	6	15	15
AA/CD	Not valid		

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

Enter: AY MR-12345 GO ID MR12345 GD ID

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

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

JG# JOG

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

QUEUE REQUIREMENTS			
MODE	**LINEAR**	**PARABOLIC**	**COSINE**
AX - AT	2	Linear ramp	
AA,AM	Not valid		
AA/CD	Not valid		

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

Enter: JG100000 JG35000 ST

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	
AX - AT	3
AA,AM	Not valid
AA/CD	Not valid

Example: Jog the Y axis at $2\frac{2}{3}$ steps per second.

Enter: AY JF2.667

VS#,#,# VELOCITY STREAMING

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

QUEUE REQUIREMENTS	
MODE	
AX	5
AY - AT	Not valid
AA,AM	Not valid
AA/CD	Not valid

Example:	Create a stair step ramp on the X and Y axes, with the X axis moving in the negative direction and the Y axis in the positive direction. Make each step last 1 second long and increase velocity by 1,000 steps/second, until a velocity of 3,000 steps/second is reached, then step down to 0 steps/second.
Enter:	AX VS1024,-1000,1000; VS1024,-2000,2000; VS1024,-3000,3000; VS1024,-2000,2000; VS1024,-1000,1000; VS1,0,0;

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 - AT	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 - AT	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 - AT	Flush +2
AA,AM	Flush +2
AA/CD	Not valid

Example: Flag a done when all axes have stopped.

Enter: AA SD ID

KL KILL

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

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

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

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

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.

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

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

Example: (see LS command on the previous page)

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 - AT	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 MR 10000; GO MR, -1000; GO WD

WD WHILE DONE

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

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

Example: (see WH command on the previous page)

CW CLEAR WHILE

The CW command breaks the WH command upon execution on 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 - AT	1
AA,AM	1
AA/CD	Not valid

Example: (see WH command on the previous page)

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	
AX - AT	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	
AX - AT	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 HH HM VL1000 HL HR

KM HOME AND KILL

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

QUEUE REQUIREMENTS	
MODE	
AX - AT	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	
AX - AT	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 VME44 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 - AT	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 - AT	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. 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 - AT	2
AA,AM	2
AA/CD	Not valid

Example: The following sequence would interrupt the host when the X axis is complete and 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.

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

Example: Send DONE when axis is within deadband.

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

IC INTERRUPT CLEAR

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

QUEUE REQUIREMENTS	
MODE	
AX - AT	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 - AT	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 command with a WA, they will not execute until all previously defined moves have finished.

QUEUE REQUIREMENTS	
MODE	
AX - AT	Not valid
AA	2
AA/CD	Not valid

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

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

WQ WAIT FOR QUEUE TO EMPTY

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

QUEUE REQUIREMENTS	
MODE	
AX - AT	Immediate
AA	Immediate
AA/CD	Not valid

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

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

SW# SYNC WAIT

The SW command allows synchronization of multi-axis moves or other tasks on one or more VME44 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 - AT	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 four axis boards, for a total of 12 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; GO
(Board 3) AA SW0; MR300,300,300,300; GO
(Board 1) AA MR100,100,100,100; 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 - AT	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 JG0

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.

WY WHO ARE YOU

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

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

Example: You want to examine the board information.

Enter: WY

Response: <LF><CR>VME44 ver 1.76-4E<LF><CR>

RP REQUEST POSITION

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

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

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

Enter: AY RP

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

RQ REQUEST QUEUE STATUS

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

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

QUEUE REQUIREMENTS	
MODE	
AX - AT	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 six digit hex format, surrounded by line feed and carriage return pairs. The three left hex digits are unused and are always set to 0. A one in any binary position signals that bit as being low.

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

Example: User output bits 10 and 11 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>000C09<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 - AT	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 - AT	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 addressed axis like the RA command except flags are not affected.

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

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

Enter: AA RC

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

RV REQUEST VELOCITY

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

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

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

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

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

RU REPORT POSITION IN USER UNITS

The RU command returns the current position in user units (see UU command). 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 - AT	Immediate
AA,AM	Not valid
AA/CD	Not valid

Example: One revolution of a motor is 2000 steps. Define user units so moves can be referenced in revolutions. Move the Z axis 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. These commands must be given in the single axis mode but will remain effective in the AA or AM modes. The VME44 defaults to user units off at power up or reset.

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

Example: The motor, driver and gear ratio you are using requires 10,000 steps to move one inch. Set up the X, Y and Z axes so you can enter move information in inches.

Enter: AX UU10000 AY UU1000 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 - AT	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. 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	
AX - AT	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 steps 25,000 times, the encoder produces 10,000 counts. Set up an encoder ratio so the hold mode will work correctly.

Enter: ER10000,25000

HV# HOLD VELOCITY

The HV command specifies maximum position hold correction velocity. This is the peak velocity which will be used while making position corrections.

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

Example: (see HN command on page 5-54)

HG# HOLD GAIN

The HG command allows the user to specify position hold gain parameter. This gain parameter is multiplied by the position error in determining the velocity during correction. The parameter must be between 1 and 32,000. The parameter should be set experimentally by increasing it until the system is unstable, then reducing it slightly below the threshold of stability.

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

Example: (see HN command on the next page)

HD# HOLD DEADBAND

The HD command specifies deadband counts for position hold. If the stage is within this limit, it is considered in position and no further correction will be made. This parameter interacts with the HG command, i.e. a larger deadband will allow a larger gain parameter in many applications. A parameter of zero is allowed.

QUEUE REQUIREMENTS	
MODE	
AX - AT	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.

QUEUE REQUIREMENTS	
MODE	
AX - AT	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 position correction after a move and activates the HV, HG and HD commands. Hold and slip detection are disabled if an LP, HM, HR, SA, ST or KL command is entered or if a limit is encountered.

QUEUE REQUIREMENTS	
MODE	
AX - AT	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.

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

Example: Send DONE when axis is within deadband.

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

5.15. SLIP AND STALL DETECTION COMMANDS

ES# ENCODER SLIP TOLERANCE

The ES command parameter specifies tolerance before slip or stall is flagged in the status register and by the RL command. The mode must be turned on with an IS command and off with an HF command.

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

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

Enter: ES5 IS

IS INTERRUPT ON SLIP

The IS command enables the VME44 to interrupt the host on slip or stall detection, if the appropriate bit has been set in the interrupt control register. Hold and slip detection are disabled if an LP, HM, HR, SA, ST or KL command is entered or if a limit is encountered. If a slip occurs, slip detection must be re-enabled.

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

Example: (see ES command above)

RL RETURN SLIP STATUS

The RL command returns the slip detection status of each axis. An S is returned if slip has occurred for that axis, or else an N is returned. The results are bounded by an LF CR pair, as in other status commands. The number of characters returned corresponds to the number of axes available on the board.

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

Example: Disable slip detection on the X axis.

Enter: AX HF

5.16. ENCODER TRACKING COMMANDS

ET ENCODER TRACKING

The ET command turns on the encoder tracking mode. The axis will track its encoder input, thus allowing one axis to follow the activity of another or a thumbwheel for manual positioning or the movement of another device that produces a signal compatible to the encoder inputs. No acceleration or deceleration ramps are generated. The axis will duplicate the encoder input. The ER command allows the user to scale the motor's movements relative to the encoder.

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

Example: Turn off encoder tracking on X axis.

Enter: AX HF

5.17. ENCODER HOME CONTROL COMMANDS

HE HOME ENCODER

The HE command enables encoder index mode when an HM or HR command is executed. Home is defined as the logical AND of the encoder index, the external home enable and the encoder quadrant where channel A is positive and channel B is negative. The external enable is low true, i.e. the HH and HL commands are not valid in this mode. The home logic expressed in boolean terms is:

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

QUEUE REQUIREMENTS	
MODE	
AX - AT	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 VME44 home switch mode to determine where home is when an HM or HR command is executed (default at power up or reset). This mode can also be used with encoders which contain internal home logic by connecting their output to the VME44 home input for the appropriate axis. The active level of this input may be controlled by the HH and HL commands.

QUEUE REQUIREMENTS	
MODE	
AX - AT	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

5.18. ENCODER STATUS REQUEST COMMANDS

EA ENCODER STATUS

The EA command returns encoder status of the currently addressed axis in the following format:

EA COMMAND RESPONSE DESCRIPTION		
CHAR	**SENT**	**DESCRIPTION**
1	LF	Line feed
2	CR	Carriage return
3	CR	Carriage return
4	E	Slip detection enabled
	D	Slip detection disabled
5	E	Position maintenance enabled
	D	Position maintenance disabled
6	S	Slip or stall detected
	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	
AX - AT	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	
AX - AT	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>

5.19. 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 - AT	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 - AT	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 steps can be specified in this manner with both acceleration and deceleration. The controller will not reverse direction if the position has already passed, but will behave as explained above. Thus the direction of the move must be specified before starting the move with the MP or MM commands. All destinations must be in absolute position, no position relative moves are allowed due to the nature of these commands.

QUEUE REQUIREMENTS	
MODE	
AX - AT	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
SP35000
```

The move as shown in Figure 5-1.

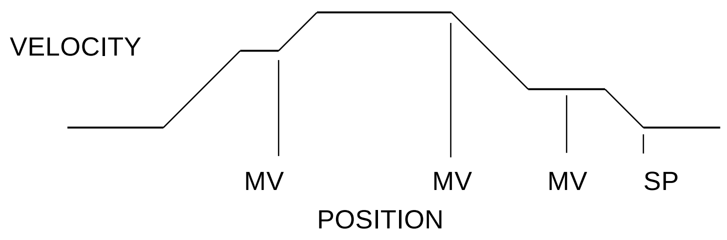

Figure 5-1 VELOCITY STAIRCASE PROFILE

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	
AX - AT	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	
AX - AT	Flush + 4
AA,AM	Not valid
AA/CD	Not valid

Example: Force axis to stop at 25,000.

Enter: FP25000

5.20. CONSTANT VELOCITY CONTOURING

The VME44 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 - AT	1
AA,AM	1
AA/CD	2

Example: (see CD command on page 5-67)

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 - AT	1
AA,AM	1
AA/CD	2

Example: (see CD command on page 5-67)

BL# BIT LOW

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

QUEUE REQUIREMENTS	
MODE	
AX - AT	2
AA,AM	2
AA/CD	Valid

Example: Turn on output bit 10 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
```

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-AT	2
AA,AM	2
AA/CD	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 4 axes if circular interpolation is not used, or 2 axes with circular mixed with linear interpolation. Attempting to do circular interpolation in a contour which is being defined for more than 2 axes will be flagged as a command error. This command is executed in the AA mode. The contouring axes must be at positions which allow them to reach the specified contouring velocity by the specified position when the contour is executed. If the actual position of the stage is equal to the starting position as defined by the CD command, the stage will jump to the contouring velocity with no ramp up. This could cause the stage to stall if it is not able to accelerate at this high rate. It is recommended that some ramp up distance be allowed. The distance required may be calculated from the equations in Section 1. There is also some ramp down distance as the stage slows from the constant velocity value to a stop. This distance is adjustable using the AC command. It can almost be eliminated using the CK command.

QUEUE REQUIREMENTS	
MODE	
AX - AT	Not valid
AA,AM	0
AA/CD	Not valid

Example: The following demonstrates cutting a hole with a 10,000 count radius using constant velocity contouring and circular interpolation. The contouring velocity is set to 1000 pulses per second. A contour is then defined beginning at coordinates 0,0 on the 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:

```
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 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 - AT	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 - AT	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 - AT	Not valid
AA,AM	Not valid
AA/CD	8

Example: (see CD command on page 5-67)

CV# CONTOUR VELOCITY

The CV command allows specification of contouring velocity. It is executed from the AA mode before a contour definition. A contour defined by a CD command cannot be executed if followed by a CV command. Changing this parameter will make any previously defined contours invalid. The contour velocity defaults to 1000 at power up or reset.

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

Example: (see CD command on page 5-67)

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 - AT	Not valid
AA,AM	6
AA/CD	Not valid

Example: (see CD command on page 5-67)

MT#,# MOVE TO

The MT command causes the axes defined by the CD command to move to the specified absolute position using linear interpolation. Only the axes being used in a contour must be specified in the contouring mode.

QUEUE REQUIREMENTS			
MODE	**LINEAR**	**PARABOLIC**	**COSINE**
AX - AT	Not valid		
AA,AM	6	30	30
AA/CD	4 + number of axes		

Example: Make a hexagon in CV mode using the X and Y axes.

Enter:

```
AA CV5000;
CD10000,0;
MT20000,0
MT25000,10000
MT20000,2 0000
BL9
MT10000,20000
MT5000,10000
BH9R MT10000,0;
CK
CX
```

RQ REQUEST QUEUE STATUS

The RQ command returns the number of entries available in the contouring queue.

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

Example: Examine contour queue size.

Enter: AA CD0,0; RQ

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

5.21. COMMAND SUMMARY

The following commands are included in the VME44 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		
COMMAND	**SECTION PAGE NUMBER**	**COMMAND DESCRIPTION**
AA	2	The following commands are for the AA mode.
AC#	17	Acceleration, set acceleration/deceleration register
AF	13, 65	Auxiliary off.
AM	3	Axes multitasking mode
AN	12, 65	Auxiliary on.
AT	5	The following commands are for the T axis
AX	3	The following commands are for the X axis (default on reset).
AY	4	The following commands are for the Y axis
AZ	4	The following commands are for the Z axis
BH#	15, 66	Set selected I/O bit high (off)
BL#	15, 66	Set selected I/O bit low (on)
BX	16, 46	Return bit status in hex format
CA	41	Clear done flag of currently addressed axis
CD#,#;	67	Define a contour
CE	68	End contour definition, movement ramps to stop at end of contour
CK	68	End contour definition, kill movement at end of contour
CN	10	Cosine on, enable cosine velocity profiles
CR#,#,#	69	Circular interpolation, move in a circle
CV#	69	Contouring velocity, definition
CW	35	Clear while flag, i.e. terminate WH/WG loop
CX	70	Contour execute
EA	60	Encoder status, return encoder status of currently addressed axis
EF	6	Echo off, turn off echo to host (default at power up)
EN	6	Echo on, turn on echo to host
ER#,#	52	Encoder ratio, set encoder count to motor count ratio
ES#	56	Encoder slip tolerance, set tolerance before slip or stall is flagged
ET	58	Encoder tracking, set encoder tracking mode
FP#	64	Force position, flush queue and attempt to stop at specified position
GD	26	Go and reset done flags
GO	25	Go command, start execution of motion

SUMMARY OF COMMANDS IN CHAPTER 5		
COMMAND	**SECTION PAGE NUMBER**	**COMMAND DESCRIPTION**
HD#	53	Hold deadband, specify deadband tolerance for position hold
HE	59	Encoder home mode, set home on encoder logic
HF	54, 57, 58	Hold off, disable position hold, slip detection and tracking modes
HG#	53	Hold gain, specify position hold gain parameter
HH	7	Home high, home switches are active high
HL	7	Home low, home switches are active low
HM#	36	Home, find home and initialize the position counter
HN	54	Hold on, enable position correction after move
HR#	37	Home reverse, find home in reverse direction and initialize position counter
HS	59	Home switch, enable home switch mode
HV#	52	Hold velocity, specify maximum position hold correction velocity
IC	41	Interrupt clear, clear done interrupt status and error flags
ID	39	Interrupt host when done and set done flag
II	39	Interrupt independent
IN#	40	Interrupt when nearly done
IP	40, 55	Interrupt when in position
IS	56	Interrupt slip, interrupt host on slip or stall detection
JF#	27	Jog the current axis at fractional rates
JG#	27	Jog command, motor will run at specified velocity until a new velocity command is sent or it is stopped by a ST or KL command
KL	30	Kill, flush queue and terminate pulse generation immediately on all axes without decelerating
KM	37	Home and kill pulse generation
KR	38	Home in reverse and kill pulse generation
LE	32	Loop end, terminate most recent LS command
LF	8	Disable limit switches for selected axis
LN	8	Enable limit switches for selected axis
LP#	19	Load position, load position counter with parameter
LS#	31	Loop start, set loop counter, from 1 to 32000 loops; (may be nested to 4 levels)
MA#	20	Move absolute, move to absolute position
ML#,#;	22	Move linear, move specified distance relative from current position
MM	62	Move minus, set minus direction for MV type move
MO	23	Move one pulse in current direction
MP	62	Move plus, set positive direction for MV type move
MR#	21	Move relative, move specified distance from current position

SUMMARY OF COMMANDS IN CHAPTER 5

COMMAND	SECTION PAGE NUMBER	COMMAND DESCRIPTION
MT#,#;	23, 70	Move to, move to specified position.
MV#,#	63	Move velocity, move to first parameter (absolute position) at second parameter velocity without stopping at end of move
PA#	14	Power automatic, turn power on before each move
PF	11	Parabolic off, disable parabolic ramps, i.e. linear ramps will be generated
PN#	10	Parabolic on, enable parabolic ramps
QA	48	Query status of switches and flags for addressed axis without affecting flags
QI	49	Query status of switches and flags on all axes without affecting flags
RA	47	Return status of switches and flags and reset flags
RB	16	Return programmed direction of I/O bits in hex format
RC	49	Return current acceleration or deceleration of the current axis
RE	61	Request encoder position, return current encoder position
RI	48	Return status of switches and flags for all axes and reset flags
RL	57	Return slip status of each axis
RM#	24	Return remainder of position divided by parameter in position counter
RP	45	Request position, return current position
RQ	46, 71	Request queue status, return number of queue entries available.
RS	11	Software reset of VME44
RU	50	Return current position in user units
RV	50	Return current velocity at which the axis is moving
SA	29	Stop all, flush queue and stop all axes with deceleration
SD	30	Stop all axes and clear any done flags
SE#	14	Set settling time before power is reduced in PA mode
SF	9	Soft limit off, restore normal overtravel operation
SL	9	Soft limit mode allow pulse train to ramp down on overtravel
SP#	64	Stop at position, stop at specified position if possible after all commands have been executed
ST	29	Stop, flush queue and decelerate to stop
SW#	43	Sync wait, wait for I/O line to be released by other controllers
UF	51	User units off, turn off user unit translation
UU#	51	User units, multiply acceleration, velocity and distance parameters by specified parameter

SUMMARY OF COMMANDS IN CHAPTER 5		
COMMAND	**SECTION PAGE NUMBER**	**COMMAND DESCRIPTION**
VB#	19	Base velocity, set base velocity
VL#	18	Set maximum velocity to be used in profile
VS#,#,#	28	Velocity stream, slave velocity mode for profiling
WA	42	Wait until all moves on all axes are finished
WD	33	While end, WS loop terminator
WH	34	While, execute all commands until WG loop terminator
WG	35	Terminate WH loop
WQ	42	Wait until current axis queue is empty
WS#	33	While sync, execute while I/O line is true
WT#	44	Wait, wait for specified number of milliseconds
WY	45	Who are you, return model and software revision

6. HOST SOFTWARE

6.1. HOST SOFTWARE EXAMPLE

The following is an example program for the VME host to access the VME44 motion control board. The program is written in 68000 assembler and runs under and uses system calls from VMEPROM. The program will take characters typed at the VME host console and pass them to the VME44 board and return all characters output by the VME44 to the host console. Invoke the VME44 program and enter the following at the keyboard:

```
EN WY
```

This will turn on echoing from the VME44 so you can see what you are typing and should respond with the current firmware revision number so that you know that communication is established with the VME44 board.

VME44 users are welcome to use any of the subroutines in this software for their application. No license is required.

6.2. VME44 DEMONSTRATION PROGRAM

The following is a source listing of the demonstration program which does not use interrupts:

```
MDATA   .equ $FFFF81
MDONE   .equ $FFFF83
MCNTL   .equ $FFFF85
MSTAT   .equ $FFFF87
MVECT   .equ $FFFF89
MIRQE   .equ $80
MTBE    .equ $40
MIBF    .equ $20
MDONE   .equ $10

start:
        move.b  #0,MCNTL

        clr.l   d0
loop:dc.w $a078                 ;char available from console
        beq  L10                ;no.
L1:     move.b  MSTAT,d1               ;get VME44 status
        and.b   #MTBE,d1               ;Transmit buffer empty?
        beq  L1                 ;no wait till it is
        move.b  d0,MDATA               ;yes send char
L10:
        move.b  MSTAT,d1               ;check VME44 status
        and.b   #MIBF,d1        ;input buffer full
        beq  loop       ;no go for more
        move.b  MDATA,d0        ;yes get char
        dc.w $a086      ;send to console
        bra  loop       ;and repeat forever

        .end start
```

6.3. VME44 DEMONSTRATION PROGRAM WITH INTERRUPTS

The following is a source listing of the demonstration program which uses interrupts:

```
MDATA   .equ $FFFF81
MDONE   .equ $FFFF83
MCNTL   .equ $FFFF85
MSTAT   .equ $FFFF87
MVECT   .equ $FFFF89
MIRQE   .equ $80
MTBE    .equ $40
MIBF    .equ $20
MDONE   .equ $10

VECADR .equ $100
IVECTOR .equ    $40

start:
        lea  serv-offs+2(PC),a0         ;get actual program counter
offs:
        move.l  #VECADR,a1      ;and vector address
        move.l  a0,(a1)         ;set up interrupt vector
        move.b  #IVECTOR,MVECT
        move.b  #MIBF,d0        ;turn on IBF interrupt
        or.b #MIRQE,d0  ;enable interrupts
        move.b  d0,MCNTL

        clr.l d0
loop:dc.w $a078         ;char available from console
        beq  L10        ;no.
L1:     move.b  MSTAT,d1        ;get VME44 status
        and.b   #MTBE,d1        ;Transmit buffer empty?
        beq  L1         ;no wait till it is
        move.b  d0,MDATA        ;yes send char
L10:
        bra  loop       ;and repeat forever

serv:move.b     MSTAT,d1        ;check flags
        and.b   #MIBF,d1        ;input buffer full
        beq  sexit
        move.b  MDATA,d0
        dc.w $a086      ;send to console
sexit:  rte

        .end start
```

7. SERVICE

7.1. USER SERVICE

The VME44 indexer contains no user serviceable parts. The following is provided as an aid to understanding the operation in order to facilitate software and system development.

7.2. VME44 THEORY OF OPERATION

The 68000 microprocessor on the VME44 indexer maintains four concurrent processes. The highest priority process calculates the desired pulse frequency 1024 times each second with a proprietary algorithm (patent number 4,734,847). This frequency is fed to U91 and U93 which generate the pulse trains. The velocity profile and synchronization of each axis is also handled by the 68000.

The commands from the VME bus master or compatible host computer are temporarily stored in a 124 character buffer until the 68000 microprocessor can parse them. The command is then executed immediately or routed to separate command queues for each axis. The command queue contains a list of addresses to execute followed by an optional parameter. A command from the host may be expanded into several commands to the appropriate axis. The GO command, for example, will expand into start, ramp up, constant velocity and ramp down commands. The LS command will save its parameter i.e. the loop count on a loop stack along with the address of the LS command to be used by the next LE command as a target for a jump command. The LE command will decrement the loop count and jump to the most recent LS command providing the loop count has not reached zero. If the loop count has reached zero and it is not nested inside another loop the queue space will be flagged as available and the next instruction in the queue will be executed.

Interrupts to the VME host are generated by U67. The enable status of each interrupt is stored by U68. The status of interrupts and error flags may be read by the host via U38. U16 and U17 are comparators to detect a match between the address jumper settings and the I/O request from the host.

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.

APPENDIX B.
TECHNICAL SUPPORT

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 give you a Return Materials Authorization (RMA) number.

   Mark the RMA number on the shipping label, packing slip and other paper work accompanying the return. We cannot accept returns without an RMA number.

3. Please be sure to enclose a packing slip with the RMA number, serial number of the equipment, reason for return, and the name and telephone number of the person we should contact if we have further questions.

4. Pack the equipment in a solid cardboard box secured with packing material.

5. Ship prepaid and insured to:

   OREGON MICRO SYSTEMS, INC.
   Twin Oaks Business Center
   1800 NW 169th Place, Suite C100
   Beaverton, OR 97006

APPENDIX C.
SPECIFICATIONS