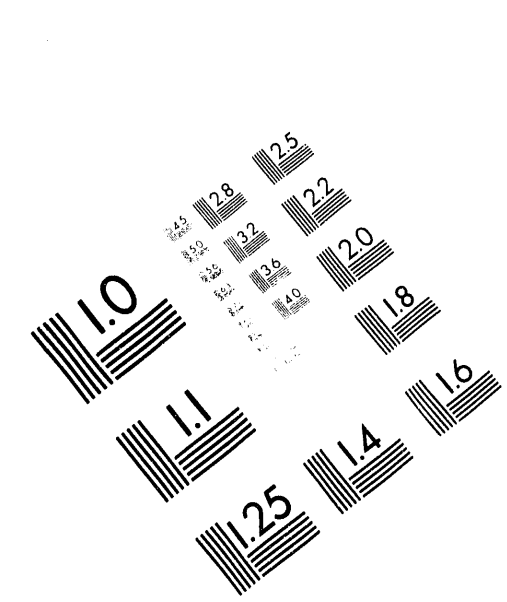
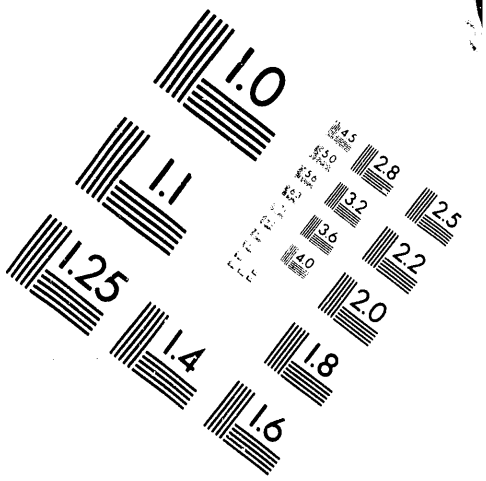
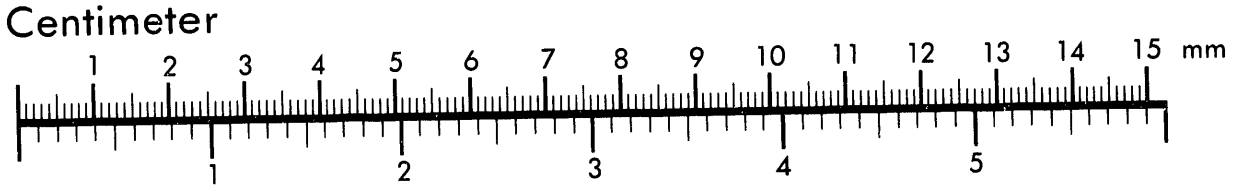
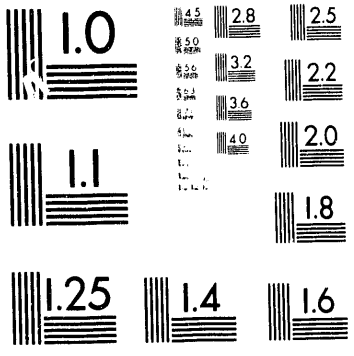
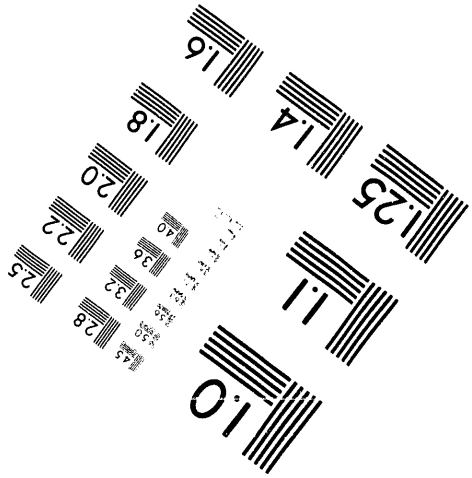


Association for Information and Image Management
1100 Wayne Avenue, Suite 1100
Silver Spring, Maryland 20910
301/587-8202

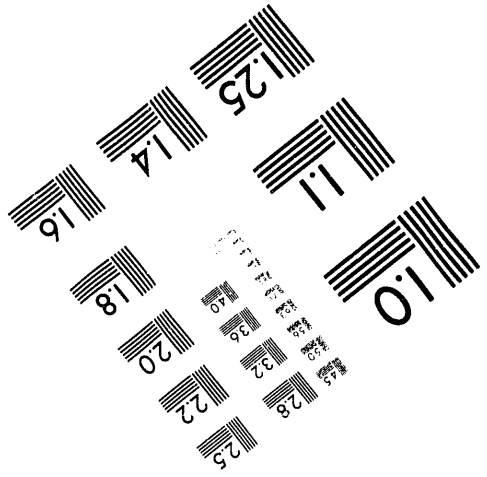








MANUFACTURED TO AIIM STANDARDS
BY APPLIED IMAGE, INC.



1 of 1

BNL-48962
INFORMAL REPORT

MULTI MOTOR CONTROLLER MMC32

USER MANUAL

S. Kate Feng-Berman, D. Peter Siddons

February, 1993

Research Supported by the
OFFICE OF BASIC ENERGY SCIENCES

Table of Contents

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The MMC32 is a versatile stepping motor controller for systems with many motors. The system as currently configured can control up to 32 motors, with all motors capable of full speed operation concurrently in different pulse modes. Each individual motor's position can be monitored in an open loop, a closed loop, or an encoded loop, even when the motor is moving. There are 2 limit switch inputs for each motor, and a further input to accept a reference position marker. The motors can be controlled via a front panel keyboard with display, or by a host computer over an IEEE-488 interface. Both methods can be used together if required. The details for manual operation are in Chapter 4, and for remote computer control are in Chapter 5.

The manual operation is controlled by the front panel keypad with interactive menu display. There is an "emergency stop" key on the front panel keypad to abort the motion of all motors without losing track of the motors' position. Please reference Appendix B for MMC32 motor board connector pinouts.

1.2 FEATURES AND SPECIFICATIONS

The MMC32 is a high density, extremely flexible module that controls incremental motors and other pulse-driven devices. For each channel, the mmc32 provides two kinds of output-pulse modes and three kinds of input-pulse modes in negative true logic at TTL levels. Both input and output modes have the directory mode and pulse mode. In the directory mode it supplies(output)/accepts(input) a direction signal (1=CW[-], 0=CCW[+]) and a pulse-train output/input. In the pulse mode, it supplies(output)/ accepts(input) pulses in both CCW(+) and CW(-) directions. The additional mode for input pulses is called encoded mode which accepts an encoder A-phase input and an encoder B-phase input with 1X, 2X, and 4X multiplication. The following is a list of the MMC32 features and specifications. Please note that the features marked with a '*' symbol are **NEW FEATURES.**

1 Optional remote (IEEE-488) and manual control

2 Abort motion capability

3 Maskable Service Request (SRQ) generation

4 Three Limit-switch inputs for each motor: upper,lower, and reference position

5* Thirty-two independent channels, one for each motor has programmable output and input modes

6* Readout of actual motor's position while motor is moving either from the number of pulses sent out, or the encoded steps depending on the programmed input mode.

7* Programmable output mode for direction/pulse or CCW(+)/CW(-) pulses (section 2.4).

8* Programmable input mode for direction/pulse, CCW(+)/CW(-) pulse, or B/A phase (section 2.4).

9* Programmable start/peak speed(1 pps to 8191 pps).

10* Programmable speed multiplication mode from 0.01x mode to 30x mode (section 2.2).

11* Programmable acceleration/deceleration register value (2 to 16383, see section 2.3)

12 Maximum move per instruction is 16,777,215 steps (24 bits resolution)

13 Load/go function for different motors to move concurrently.

14 Single step capability

15 Automatic completion for remaining pulses after abort

16 Error status, position readout

17 Position calibration

18 Inexpensive

Once initialized, the absolute position of each channel is maintained throughout all operations.

A move operation may be prematurely terminated by either remote control or manual control.

The maskable SRQ capability is provided for all channels. If requested, an SRQ is generated when any motor stops moving for any reason.

The inputs for three limit switches can be provided as either normally closed or normally open (see Appendix B for details). The activation of a limit switch causes the selected motor to perform a decelerating stop. The status of the limit switches is readable by either the remote or manual method.

The LOAD/GO function provides the capability of moving a selected group of motors concurrently and repeatedly.

The output status byte reports various system errors, including the 3 limit switch states, SRQ status, motor busy flag, command error, motor enable/disable, and motion aborted state. If the command error bit is set, an error number may be read which indicates the particular error.

CHAPTER 2

BASIC SYSTEM CONCEPTS

2.1 GENERAL

The MMC32 executes motor motion in units of motor steps, or pulses. The maximum number of steps possible in one motion is 16,777,215 steps (i.e., 24 bits). If a user tries to move a motor which is busy or requesting service, the motor controller will wait until that motor finishes moving or finishes serviced to send out the motor motion command.

2.2 PULSE RATE

The default range of pulse rates is 1 to 8191 pps (pulse per second). The end users can multiple the range by changing the multiplication mode from 0.01x to 30x. With the 0.01x mode, the speed range is [0.01, 81.92] pps at an increment of 0.01 pps. With the 30x mode, the speed range is [300, 2,457,60] pps at an increment of 30 pps. The details on how to set the multiplication mode is in section 5.3.1. For normal operation the self starting rate should be smaller than the peak rate (Figure 2-1). If the user set the starting rate higher than the peak rate, then the motor will not move. The error bit of the status byte will be on to indicate error, and the error byte will be 19.

2.3 ACCELERATION/DECELERATION

The range for acceleration/deceleration register value is 2 to 16383. Please see Fig 2-2 for acceleration/deceleration motion. The acceleration and deceleration register can be set with the GPIB command 'A' in the range of [2,16383]. However, the register value does not mean too much for users. Most users prefer to set the number of pulses for acceleration/deceleration, which can be set through the GPIB command 'W' or from the front panel keypad. The math calculation between the number of pulses for acceleration/deceleration and the register value is described in the following:

Tclk : reference clock cycle=1/4.9152MHZ = 0.0000002

RA : Acceleration register

RH : Peak speed

RL : Start speed

Tsu : Acceleration time in second

PA : Number of pulses during Acceleration/deceleration.

PA= (RH*RH - RL*RL)* RA * Tclk

That is to say RA= (RH*RH - RL*RL) * 0.0000002/(pulses #)

Example If RH is equal to 2100 pps, RL is equal to 100, Tclk is equal to 0.0000002, and RA is equal to 1000, we will get the number of steps for acceleration/deceleration to be 440 steps.

The math calculation between the acceleration/deceleration time and the register value is :

Tsu= (RH - RL)* RA * Tclk

Example

If RH is equal to 2100 pps, RL is equal to 100, Tclk is equal to 0.0000002, and RA is equal to 1000, we will get the acceleration/deceleration time to be 0.4 second.

For triangle operation, the total number of pulses must be smaller than two times the acceleration/deceleration pulse number. The hardware limit switch input can be used to trigger the deceleration.

2.4 FLAG BYTE AND SERIAL POLL BYTE

The MMC32 outputs from connector P3 of the motor board for each individual motor could be direction/pulse, which is called directory mode, or CCW(+)/CW(-) pulse, which is called pulse mode. The MMC32 inputs to connector P4 of the motor board for each individual motor could be direction/pulse, CCW(+)/CW(-) pulse, or B/A phase (encoded mode). The connector pinouts of the motor board are described in APPENDIX B in details. Those options are set by the flag byte which can be set via the front panel keypad, or the GPIB interface (section 5.3.1 'F' command). The flag byte indicates the last motion command, connector P3 output mode, connector P4 input mode, and SRQ mask. The following is the table for the flag byte definition:

bit #	Definition
0,1	Operation command code
2,3,4	P4 input mode - step/dir, +/- pulse, open, A/B phase
5	not used
6	SRQ on - 0: No/1: Yes
7	P3 output mode - 0: step,dir (default)/1: CW(-) pulse, CCW(+) pulse

The P4 input mode which is decided by bits 2,3,4 is the input to the P4 connector of the motor board (described in APPENDIX B). The following table is the definitions for bits 2,3,4.

bit 4	bit 3	bit 2	Command Description
0	0	0	step/direction (default)
0	0	1	open (no input)
0	1	0	CCW(+) pulse/CW(-) pulse
1	0	0	quadrature input A/B phase, 1X multiplication
1	1	0	quadrature input A/B phase, 2X multiplication
1	1	1	quadrature input A/B phase, 4X multiplication

The operation command code is the code of last motion, excluding the motor stop command. Hence, bits 0,1 are READ ONLY. The following table is the definitions for bits 0,1.

bit 1	bit 0	Command Description
0	0	Acceleration/deceleration move
0	1	Stop command
1	0	Move until the +/- limit switch
1	1	Move until the home limit switch

**** ATTENTION To Programmer ****

The default for the SRQ flag is off in which case the users will wait polling the output status byte (section 2.5) until it indicates the motor is not busy anymore (bit 2 of the status byte). You should set the SRQ flag on (see section 5.3.1), if you do not want to wait until motors finish moving. Usually, you need the SRQ flag on for an interrupt driven, real time control software so that the user can do something else before a motor finishes moving. If the SRQ flag is set on one motor, an interrupt handling subroutine should be written so that the program will jump to the interrupt handler right after the motor finishing moving. In your interrupt handler you need to do a SERIAL POLL to clear the SRQ signal on the controller bus. The sequence for serial polling is described in the following:

0. Enable Attention control signal of access board.
1. Address MMC32 to be a talker (The default address is set at 6)
2. Send listen address of the access board (MLA: my listen address)
3. Send Serial Poll Enable (SPE)
4. Disable Attention control signal of access board
5. Read response from the device (MMC32).
6. Untalk (UNT) and Unlisten (UNL).

7. Serial Poll Disable (SPD)

The response you read from the serial poll is a byte with bit 6 indicating SRQ on and bits 0-5 indicating which motor needs SRQ.

If your controller software drivers have a function to do the above sequence for the serial poll byte, then you only have to call the function to return the serial poll byte. On example is the GPIB-PC turbo Pascal which uses the function "ibrsp" to return serial poll byte. Please see section 6.1.1 for a GPIB-PC turbo Pascal example on SRQ.

2.5 OUTPUT STATUS BYTE

The output status byte is read only. The following table shows the status byte definition:

bit #	Definition
0	CCW(+) limit switch encountered - 0/1: No/Yes
1	CW(-) limit switch encountered - 0/1: No/Yes
2	Motor busy - 0/1: No/Yes
3	Command Error - 0/1: No/Yes
4	Abort motion by deceleration stop - 0/1: No/Yes
5	Service Requested - 0/1: No/Yes
6	Motor off (Motor Disabled) - 0/1: No/Yes
7	Home position switch activated- 0/1: No/Yes

****** Caution To Programmer *******

If bit 3 is 1, it means the last issued command failed. To find out the error message, the user should use 'IE' to interrogate the error number. The corresponding error message to each error number is described in the next section.

2.6 ERROR MESSAGES

If bit 3 of output status byte is 1, then you can use the computer command 'IE' to find out the error message. Explanations of the integer error numbers follow.

1 motor tends to move over Software CCW(+) limit
2 motor tends to move over Software CW(-) limit
3 motor needs SRQ from last move
4 motor is busy
5 motor is OFF (disabled)
6 motor is at Hardware CCW(+) limit
7 motor is at Hardware CW(-) limit
8 maximum number of steps in a motion is 16,777,216
9 Invalid motor number
10 Motor is already at home position
11 IEEE-488 invalid command
12 IEEE-488 invalid interrogative command

13 Invalid display enable/disable command

14 Invalid self starting speed set

15 Invalid peak speed set

16 Invalid multiplication mode

17 Invalid acc rate set

18 Invalid parameters in LOAD/GO buffer

19 Maximum speed is set lower than minimum speed.

2.7 BUFFER CAPACITY

MMC32 uses first-in-first-out serial rotating buffers (256 bytes) for GPIB commands, GPIB data, SRQ signal, and front panel keypad commands. Since more than one motor can be moving at one time, the SRQ on end-of-motion is also buffered, and multiple SRQs will be generated: one for each motor which was moved.

2.8 CURRENT MEASUREMENT

The following is the measurements of the currents on the motor controller:

1 cpu induces 0.66 ampers of current.
1 MOTOR board induces 1.2 ampers of current.
2 MOTOR boards induce 2.16 ampers of current.
1 cpu + 2 MOTOR boards induce 2.62 ampers of current.

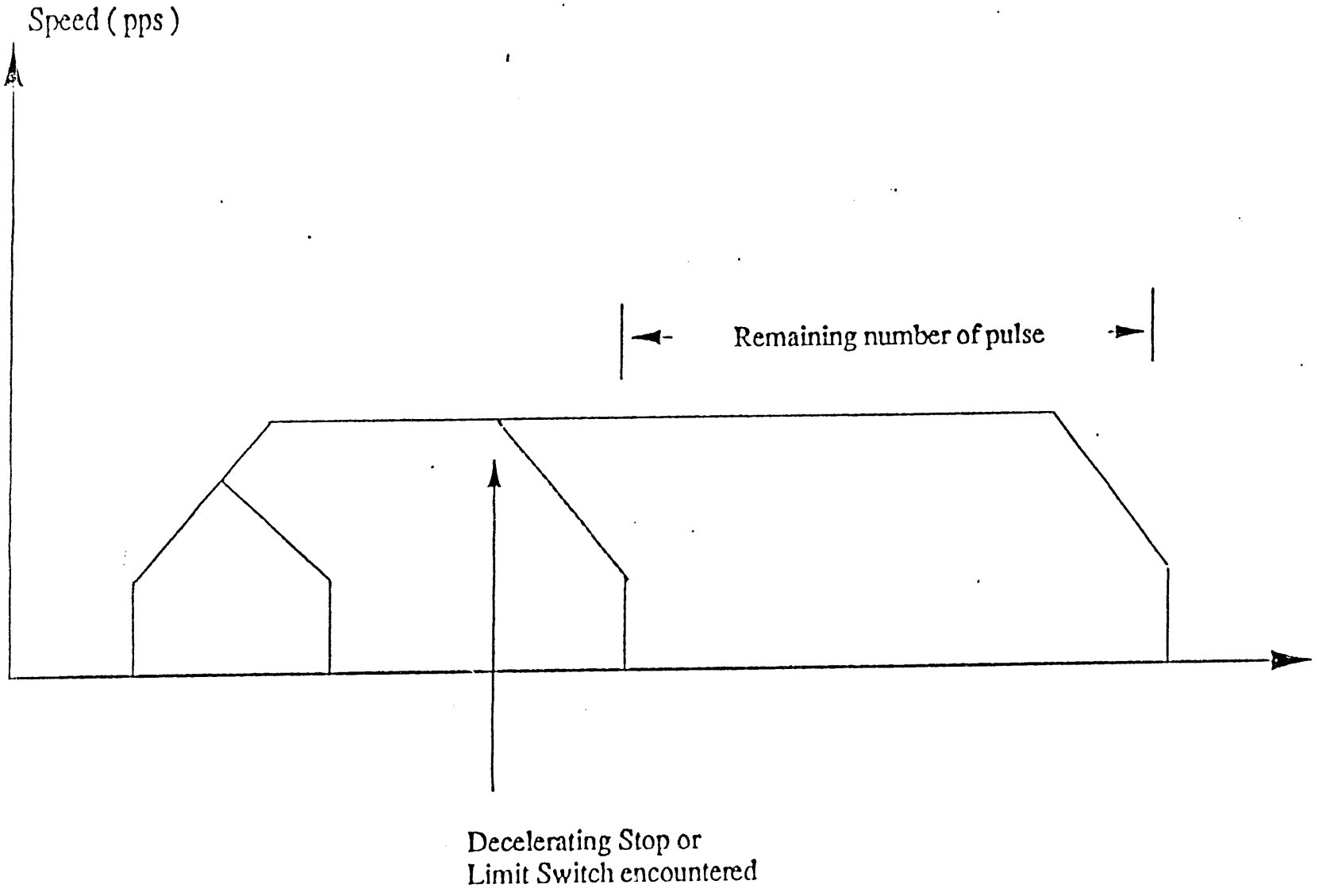


Figure 2-2 Accelerating/Decelerating Motion

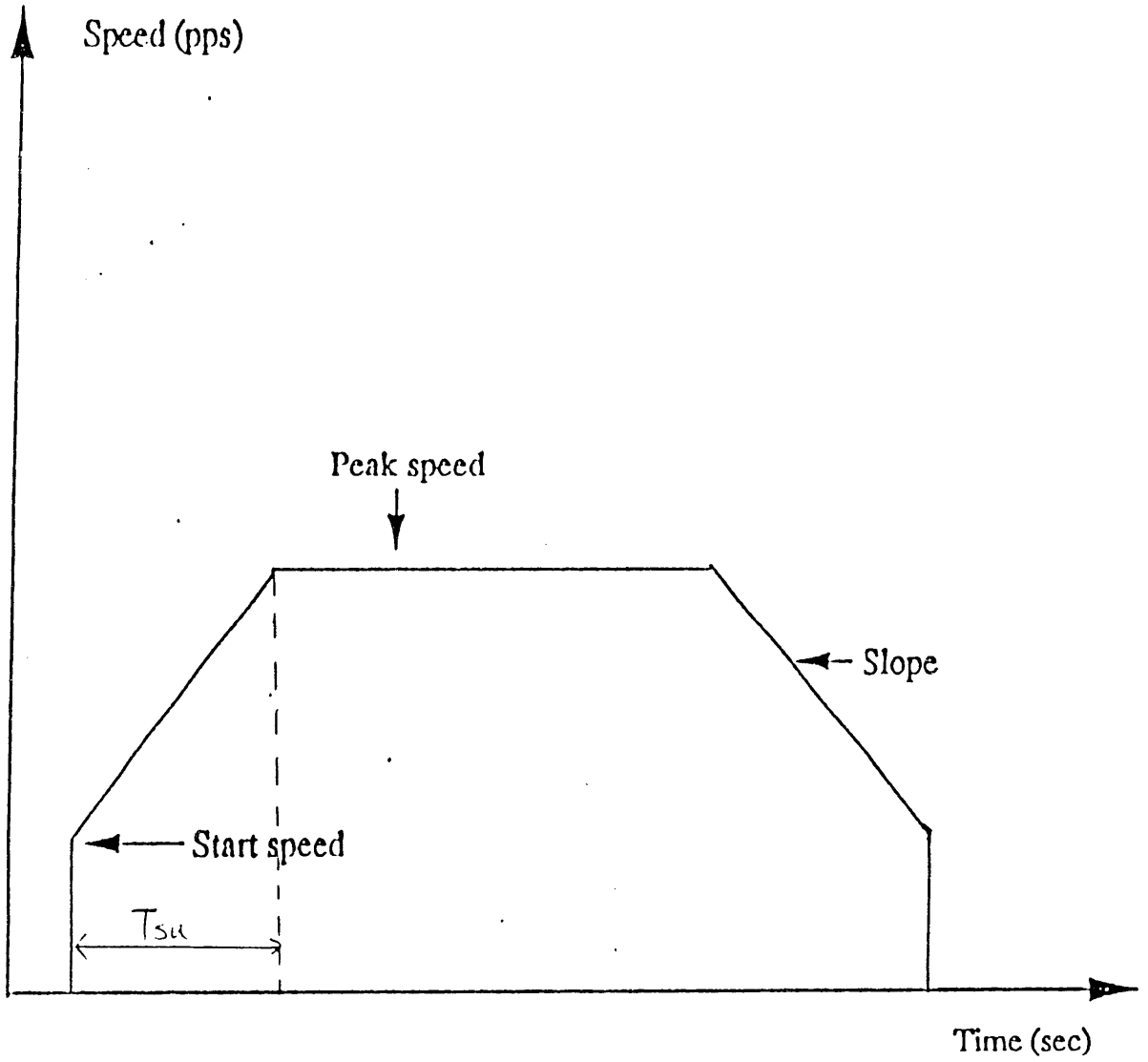


Figure 2-1 Acceleration/Deceleration Motion

CHAPTER 3

INSTALLATION

3.1 INSTALLATION GUIDE

All the motor controllers come with one CPU board marked with 'NSLS-MC32A', and the MOTOR boards marked with 'NSLS-MC32B'. Facing the back panel of the motor controller, on the MOTOR board, from left to right you will see the connector for pulses output (P3), connector for pulses input (P4), connector for CW/CCW limit switches (P5) and connector for home limit switches and motor on/off control (P6). To make sure the connector to the socket on the MOTOR board match with each other, an arrow is marked on each connector and socket indicating pin number one. The pins on the bottom layer are even number pins (2,4,6....), and the pins on the top layer are odd number pins (1,3,5.....). In MMC32, the connectors for step/direction outputs, step/direction input, and CW/CCW limit switches are E500 (a commonly used motor controller) compatible. Please reference APPENDIX B the MOTOR boards' connector pinouts.

Figure 3-1 is the layout of a CPU board, and figure 3-2 is the layout of a MOTOR board. On the CPU board, the jumpers-setting on J1 and position 6 to 8 of SW2 decide the number of MOTOR boards being used. Besides, the position 1 (least significant bit) to 5 (most significant bit) of SW2 decides the GPIB address of the motor controller (default is 6). The following table is the definition of the position 6 to 8 of SW2.

pos 8	pos 7	pos 6	number of MOTOR boards
0	0	0	1 MOTOR board
0	1	0	2 MOTOR boards
1	0	1	3 MOTOR boards
1	1	1	4 MOTOR boards

If you are using only one MOTOR board, please do not jumper pins 3,4 of J1 but the rest of the pins, and the positions 6,7 and 8 of SW2 are all off. If you are using two MOTOR boards, please do not jumper pins 3,4 and pins 5,6 of J1, and the positions 6,7 and 8 of SW2 are off, on, off respectively. If you are using three MOTOR boards, you only have to jumper pins 1,2 and pins 9,10 of J1 respectively, and the positions 6,7 and 8 of SW2 are on,on,off respectively. If you are using four MOTOR boards, you only have to jumper pins 1,2, and the positions 6,7 and 8 of SW2 are all on.

The jumper J2 of the CPU board must be jumpered at pins 3,4.

The jumper J3 on the CPU board is configured for two different kinds of U12 chips. If U12 chip is a normal RAM (usually SONY 5864), then J3 should be jumpered on position 1,2, and 5,6, and 11,12. In the future, if U12 chip is tested successfully with

static RAM with on-chip battery, then J3 should be jumpered on position 3,4, and 7,8, and 9,10.

The jumper J4 is configured for two different kinds of ROM chips. J4 should be jumpered on position 1,2 if two of 16K ROM chips (27128) sitting in chip U14, U15 respectively. Otherwise, J4 should be jumpered on position 3,4 for one 32K ROM chip on U15.

The number of MOTOR boards is one for eight-motors-control, two for sixteen-motors-control, three for twenty-four-motors-control, and four for thirty-motors-control. By jumpering P5 on the MOTOR board, you can set the motor number for the board. For example, if you jumper pins 7,8 and pins 15,16, the board controls motor 0 to motor 7. If you jumper pins 5,6 and pins 13,14, the board controls motor 8 to motor 15. If you jumper pins 3,4 and pins 11,12, the board controls motor 16 to motor 23. If you jumper pins 1,2 and pins 9,10, the board controls motor 24 to motor 31. Please reference Figure 3-2.

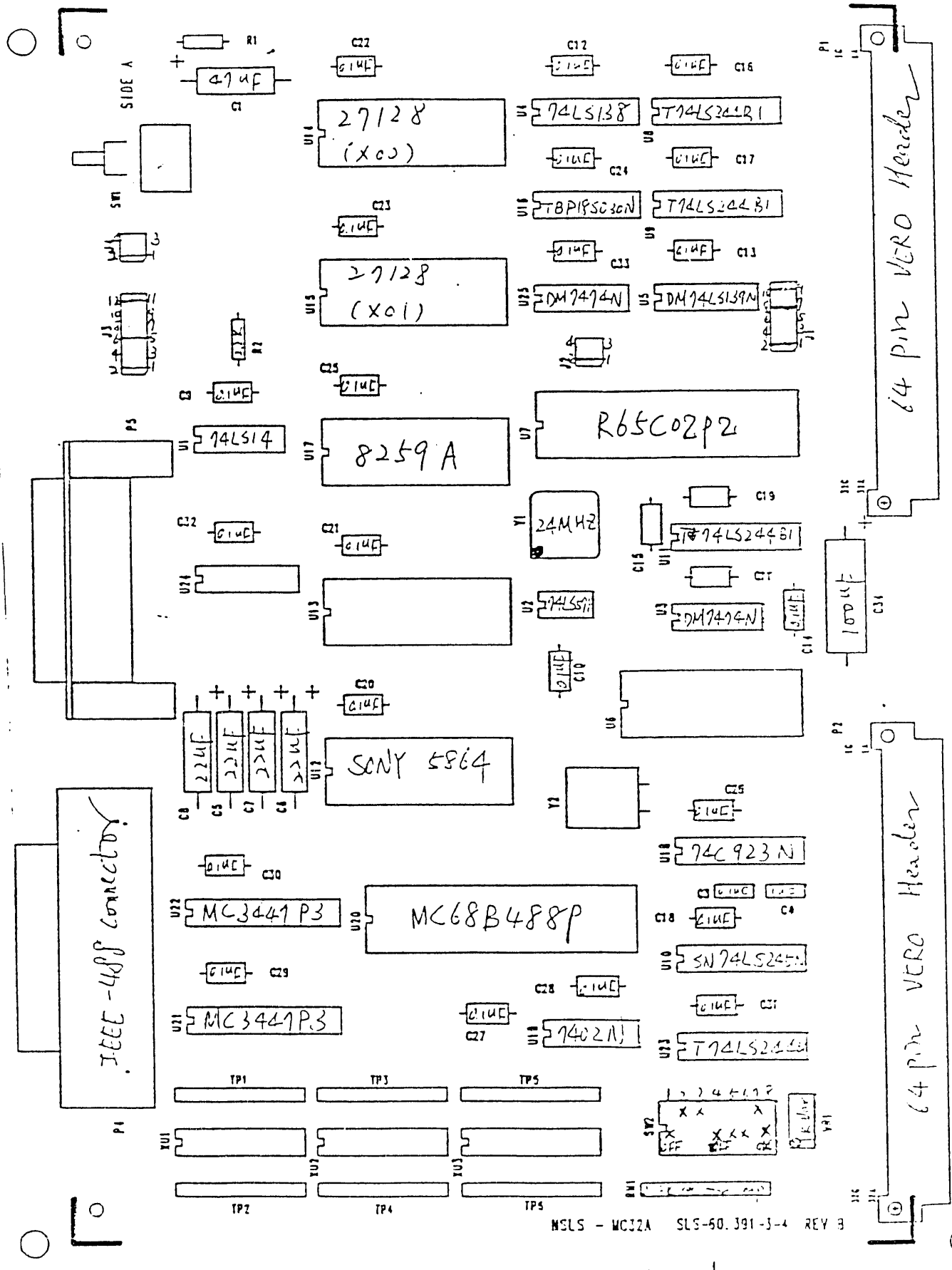


Figure 3-1 A layout of a cpu board

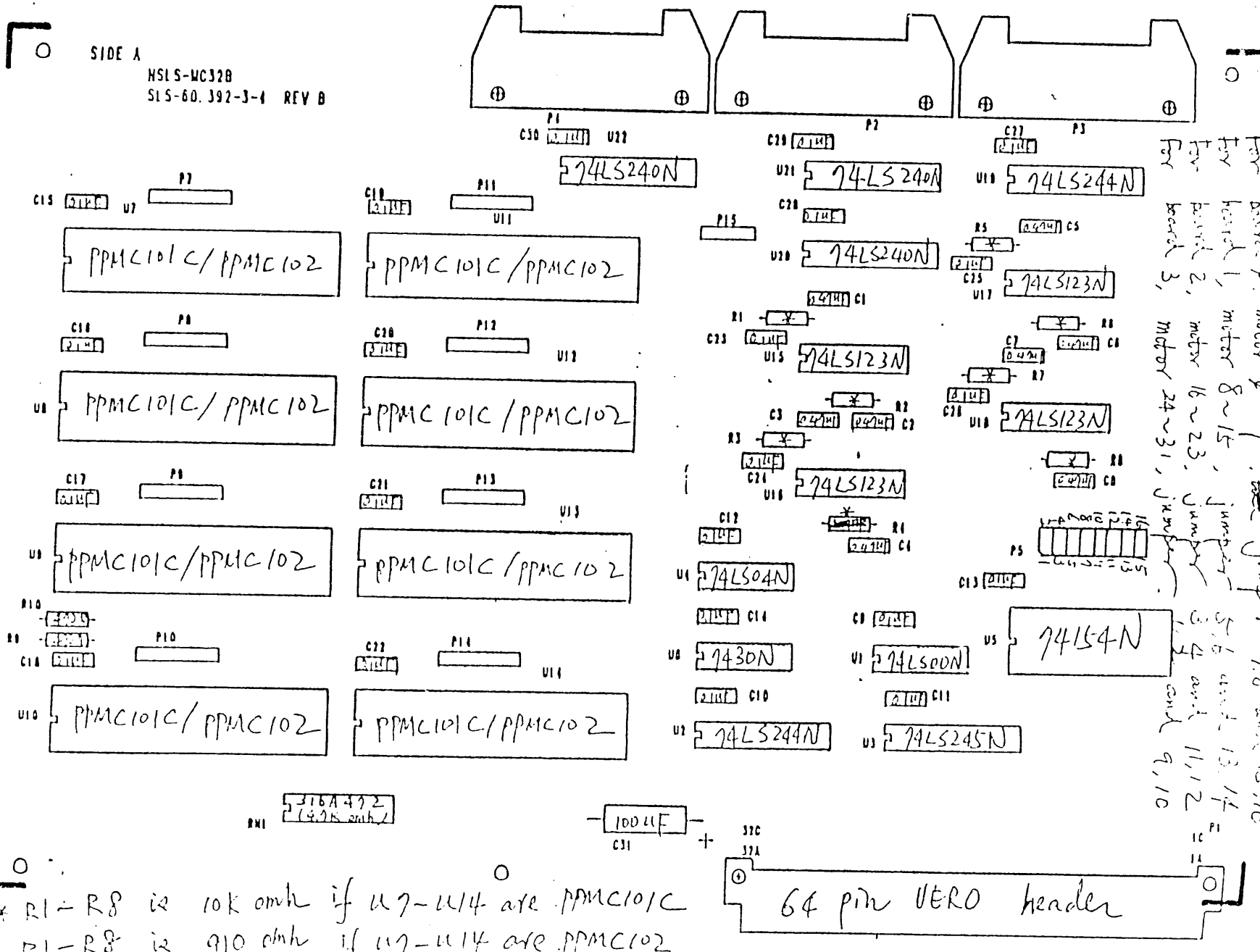


Figure 3-2. The layout of a PPMC board.

For board #1, motor 0~7, base jumper 0,8 and 15,16
For board 2, motor 8~15, jumper 5,6 and 13,14
For board 3, motor 16~23, jumper 3,4 and 11,12
For motor 24~31, jumper 1,2 and 9,10

* R1-R8 is 10K ohm if U7-U14 are PPMC101C
R1-R8 is 910 ohm if U7-U14 are PPMC102

CHAPTER 4

LOCAL MANUAL CONTROL

4.1 OVERVIEW

The Local Manual Control is performed using the MMC32 front panel keypad and display (See Figure 4-1). There are four rows available on the front panel display. Each row displays different kinds of information:

Row 1: Instruction
Row 2: Response Message
Row 3: Keyin Character + Error Message
Row 4: Menu

4.2 MENU FLOWCHART

The MMC32 manual operation is very user friendly because of the interactive information on the MMC32 front panel display. The front Panel Menu is designed like a tree structure starting from ROOT. The Root is the main menu which contains 4 functions: (1) Setup (2) Select Active Motor Number (3) Motor Motion (4) Display motor status. The menu flowchart is in Figure 4-2.

4.3 INDIVIDUAL FUNCTION HIGHLIGHT

The manual and computer controls have almost the same functions except the following.

4.3.1 SETUP

In the setup menu, you can set up different parameters or just look at the parameters by choosing the next parameter function key without inputting any number for the last parameter function key. If the setup parameter is out of the valid range, the error message will appear on row 3 of the display.

4.3.2 DELETE NUMBER KEY

On the front panel keypad, the key 'DELETE' is used to delete the last typed-in digit (-,0,....,9). You can also use this key to cancel the current function progress when you are requested to input a number. For example, if you type the 'LmtMv' (limit move) function key accidentally, when the direction question comes on (0/1), you can type the 'DELETE' key to cancel this function before typing in any digit. Then, the motor will not move, the original menu remains and an error message 'Invalid func. key <- ' appear.

4.3.3 LOAD/GO

In the 'LOAD' function you can load any of the 32 motors' motion parameters sequentially for 36 loops (you can repeat the same motor), but you have to end the loading sequence by inputting 255 for motor number selection. If any of the loaded motors tends to move beyond its software limit setting, the 'LOAD' function is ended. Then, the execution function 'GoForIt' will not do anything. You can repeat the 'GoForIt' command without changing the 'LOAD' parameters.

4.3.4 EMERGENCY STOP

This is a convenient key to stop all motors' motion. It is actually a smooth stop (decelerating stop) without losing track of the motors' position.

4.3.5 Flush input buffer

This key marked with 'FLB' is a key to flush the GPIB command buffer in case the GPIB commands time out due to GPIB communication hardware failure or I/O protocol failure.

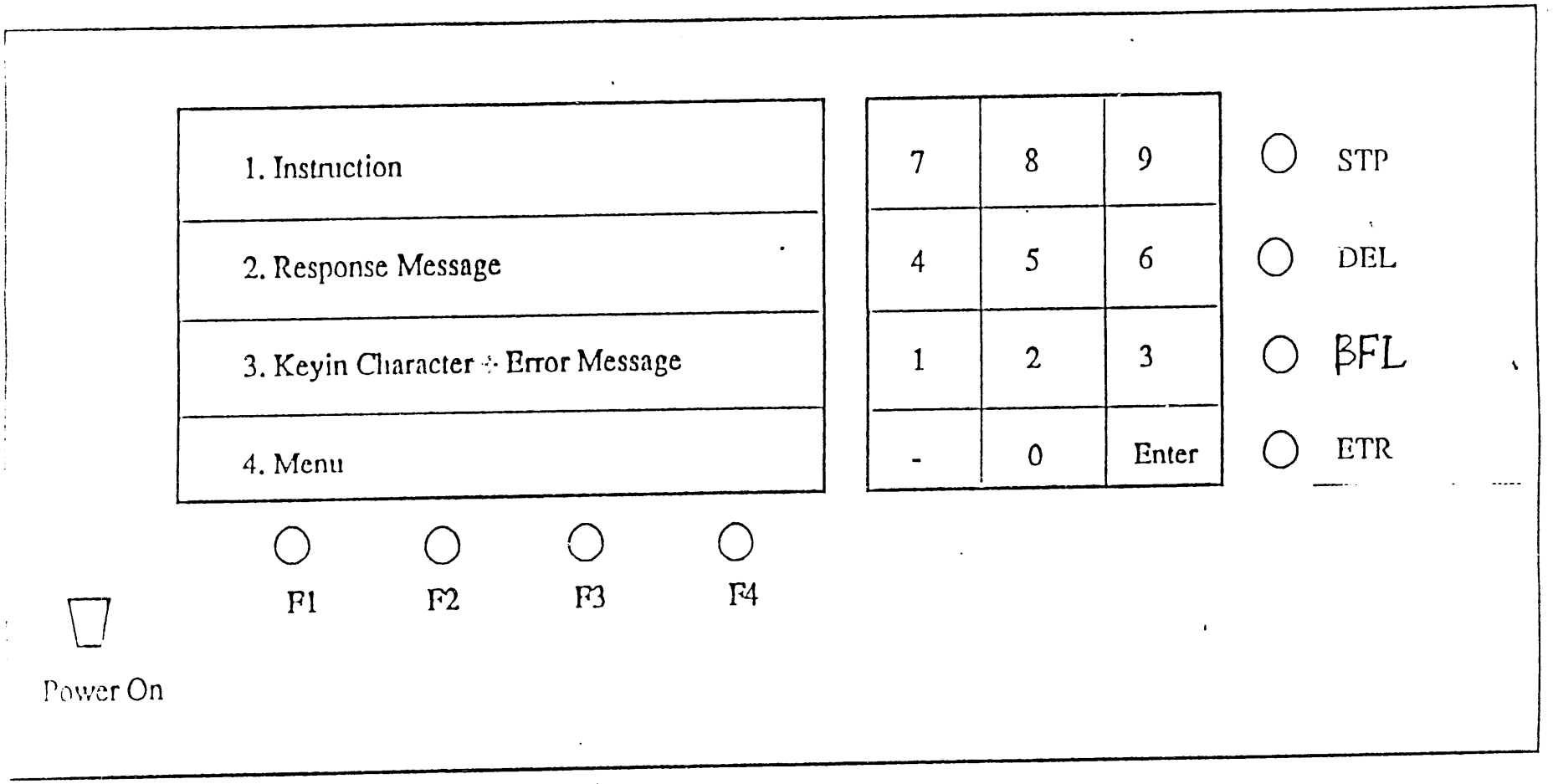


Figure 4 -1 MC32 Front Panel Controls and Displays

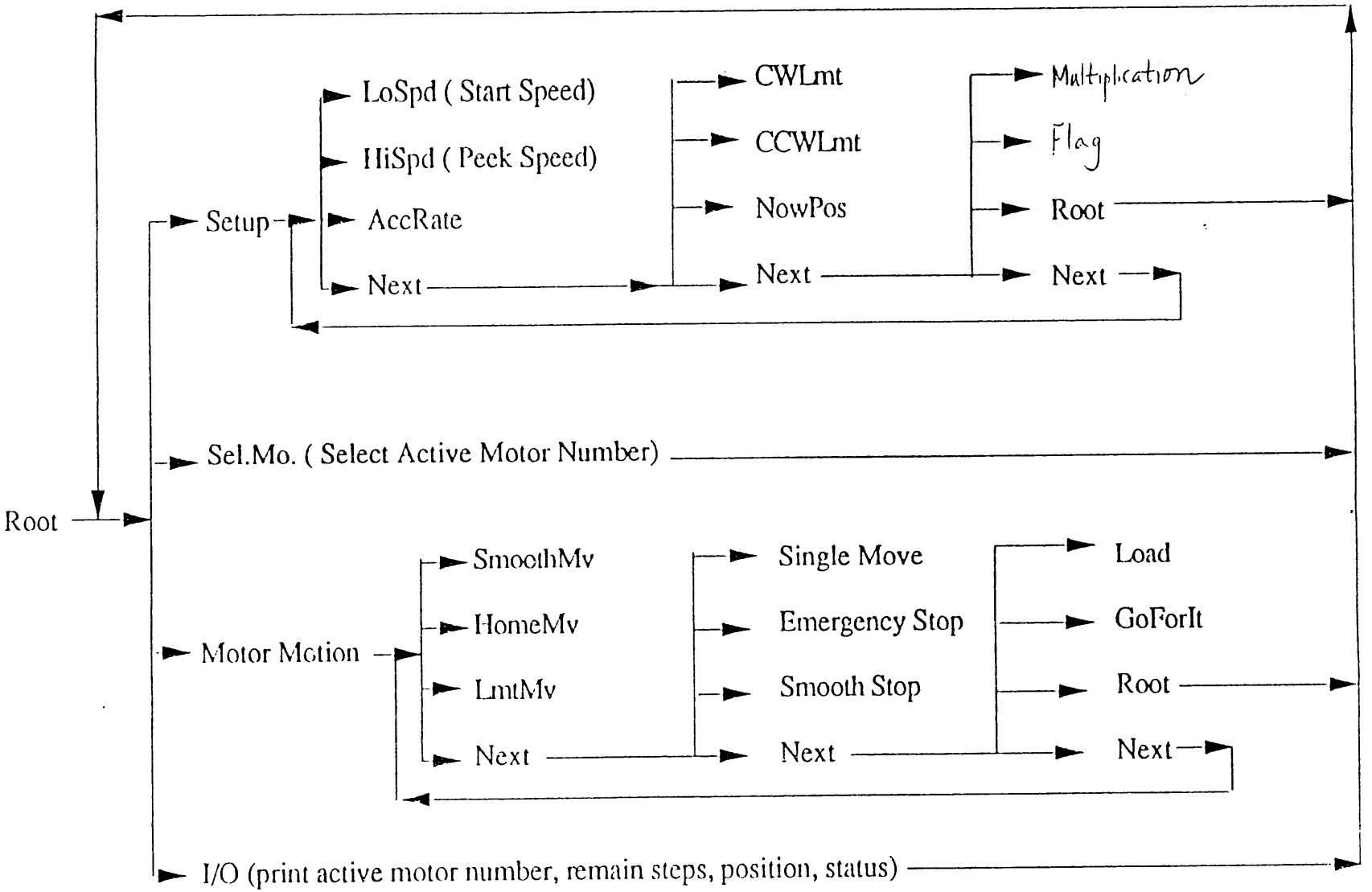


Figure 4 MC32 Front Panel Menu FlowChart

CHAPTER 5

COMPUTER-GPIB COMMAND

5.1 OVERVIEW

This chapter describes the computer commands sent through the IEEE-4888 interface to the MMC32. All commands can be either lower-case or upper-case letters.

5.2 COMPUTER COMMAND CONVENTIONS and DEFINITIONS

5.2.1 DELIMITER

All individual commands end in a delimiter that signifies that the command is completed. A delimiter serves the same function as the space between words in a sentence. The delimiter, which is part of the command, is a space character or a <LF> (i.e. Char(10) for Line Feed)

5.2.2 INDIVIDUAL COMMANDS

An individual MMC32 command controls or reads a single parameter, function, or action such as acceleration, velocity, position, load, go, etc. There are two classes of individual commands, INTERROGATIVE and EXECUTIVE.

INTERROGATIVE COMMANDS: These commands read a single parameter, the result of a function, or the status of an action. They are executed immediately on receipt and the requested data will be placed in the MMC32 output buffer (256 bytes) sequentially waiting for the user to fetch the requested data. Hence, the user must fetch the requested data before he sends the next interrogative command. All interrogative commands' initial is 'I' meaning Interrogative.

EXECUTIVE COMMANDS: These commands modify a single parameter, execute a function, or initiate an action. On receipt by MMC32, they are stored in the MMC32 input buffer(256 bytes) sequentially and executed by MMC32 in the order they are sent.

Individual commands vary in length, and consist of one or more letters with a delimiter. Each command is entered as a character/delimiter combination. Some commands include a sign (+/-) to denote direction of motion. The number of characters used depends on the type of command entered.

When two or more individual commands are entered on the same line, they should be separated by spaces. Please see Chapter 6 for command examples.

5.3 COMMAND GROUPS AND DESCRIPTION

The basic MMC32 individual commands can be divided into four categories: Parameters Setup commands; Data Interrogating commands; Motor Motion commands; and Execution commands. These commands are discussed in section 5.3.1 through section 5.3.4. An alphabetical command summary is included in Appendix A.

5.3.1 PARAMETERS SETUP COMMANDS

These commands set up parameters to control motor motion, calibrate motor position, enable/disable front panel control, and so on. If the selected motor is busy, the motor controller will wait until that motor finishes moving to change its parameters.

Annnn	Acceleration/deceleration register value setup on the active motor -- The range of choices is [2, 16383] A100 (set acceleration register value to be 100) A700 (set acceleration register value to be 700)
B	Flush the GPIB and keypad input buffer in case of GPIB time out or protocol failure
DK	Disable local mode (the front panel keypad control), except the key for emergency stop and the key to enable/disable front panel control. Note: The remote mode (computer control) is always enabled. By default, the MMC32 local mode is enabled.
EK	Enable local mode (the front panel keypad control- default).
Fnnn	Flag setup on the active motor -- Please see section 2.4 for flag bits definition. The bits 0,1 of the flag byte are not programmable. F16 (P4 input is quadrature input A/B phase 1X, P3 output is step/direction, Disable SRQ) F64 (Enable SRQ)
Nnnn	Motor Number -- Select the active motor number. All succeeding commands relate to this selected active motor before the next 'N' command is executed. Valid motor numbers are 0 to 31. N2 (select motor 2 to be the active motor number) N0 (select motor 0 to be the active motor number)

P+/-nnnnnnnn	Calibrate the active motor position (32 bits long) in motor steps. P0 (sets current position to be 0)
Unnnn	Start velocity setup on the active motor -- The default range of choices is [1, 8191] pulses per second. U300 (set start speed to be 300 pulse/sec) U245 (set start speed to be the 245 pps)
Vnnnn	Peak Velocity setup on the active motor -- The default range of choices is [1, 8191] pulses per second. V2000 (set peak velocity to be 2000 pulse/sec) V3125 (set peak velocity to be 3125 pps)
Wnnnn	The number of pulses for acceleration/deceleration W100 (set number of pulses during acceleration/deceleration to be 100)
Zffff	Setup the active motor speed multiplication mode from 0.01x to 30x. Note, the input is a floating number from 0.01 to 30. Z0.01 (set speed range to be [0.01, 81.92] pps at an increment of 0.01pps) Z30 (set speed range to be [30, 245730] pps at an increment of 30 pps)

5.3.2 DATA INTERROGATING COMMANDS

These commands request the setup value or read the motor status. All the interrogating commands start with an 'I' followed by a single character defining which data is requested. After the command is issued, you should fetch the interrogated data. The MMC32 will send LF (#10) as end of data (EOD). These commands are:

IA	Interrogate active motor Acceleration/deceleration value.
IE	Interrogate command Error number message.
IF	Interrogate the active motor Flag settings. Bits 0,1,2 give the operation code of last motion, excluding the motor stop command, and bit 6 is the SRQ mask bit. The flag byte definition is described in section 2.6.

IN — Interrogate the active motor Number.

IO — Interrogate the active motor Output Status. The response is an ASCII string representing a single byte of data. The corresponding bit definition is discussed in section 2.4.

IP — Interrogate the active motor current position in steps.

IR — Interrogate the remaining number of steps from last motion of the active motor. If it is -16777216 (i.e. -$1000000 in hex), then it means the last motion command was 'move to limit switch' (e.x. T0,T1, or H) and it was terminated by a 'stop command' before it reached the limit. Under this circumstance, MMC32 does not know the remaining number of steps, so it sends a number -16777216 (i.e. -$1000000 in hex) to warn the user.

IU — Interrogate the active motor start velocity.

IV — Interrogate the active motor peak Velocity.

IX — Interrogate the active motor software CW(-) limit position.

IY — Interrogate the active motor software CCW(+) limit position.

IZ — Interrogate the active motor seek-home velocity.

5.3.3 MOTOR MOTION COMMANDS

The motor motion commands are:

C0 — Complete ALL MOTORS which have a non-zero remaining number of steps from the last move. If the last move command is 'move to limit switch' and not finished because of a stop command, then the motor will continue moving until it encounters the limit. User should be aware of which motors have uncompleted motions from the last move before he sends this command.

C1 — Complete the ACTIVE MOTOR motion if it has a non-zero remaining number of steps from the last move. If the last move command was 'move to limit switch' and unfinished because of a stop command, then the motor will continue moving until it

encounters the limit.

H — Move the active motor to the home position (Hardware base point).

M+/-nnnnnnnn — Move the active motor to position nnnnnnnn steps (Absolute move).

M3000 (move to position 3000)
M-1000 (move to position -1000)

R+/-nnnnnnnn — Move the active motor nnnnnnnn steps from the current position (Relative move).

R3000 (move 3000 steps from current position)
R-1000 (move -1000 steps from current position)

S1 — Move the active motor one step in the CCW(+) direction.

S0 — Move the active motor one step in the CW(-) direction.

T1 — Move the active motor to the hardware CCW(+) limit.

T0 — Move the active motor to the hardware CW(-) limit.

5.3.4 OTHER EXECUTION COMMANDS

The remaining commands are a) LOAD/GO , and b) Kill motion.

a) LOAD/GO

The LOAD/GO function is used for the 'LOAD AND GO' option. Prior to any move, the microprocessor in the MMC32 must calculate a number of move parameters. In some applications this delay in calculation may be excessive. The delay between the sending of the motion command and the actual motor motion can be reduced by the use of the 'LOAD AND GO' option, especially if you want to move more than one motor at the same time. You can load any of the 32 motors' motion parameters sequentially before you execute the motion using the 'GO' function.

L — Load move command parameters, which include a space, the motor number immediately followed by an 'r' (relative move) or an 'm'(absolute move), a space, and then the number of steps. Use the same format for the next motor motion parameters until you terminate it with 255 (EOM : End Of Motor). If any motor

is commanded to move beyond the limit settings, the 'G' (execute) command will not do anything.

L 0r 1000 1m -3000 255 (motor 0 moves 1000 steps from current position, and motor 1 moves to absolute position -3000 steps)

L 6r -700 3m 400 0r 1000 255 (motor 6 moves -700 steps from current position, motor 3 moves to absolute position 400 steps, and motor 0 moves 1000 steps from current position)

G — Execute the last loaded motion parameters in the LOAD buffer. It is not effective when previous LOAD command tries to move any motor beyond its own limit setting and the commands will cause the previous motion to be re-executed. However, the user has to be careful not to move any motor beyond the limit switches in repeating the 'G' command, because doing so does not check the status of the motors' limit switch states before moving.Hence, repeating 'G' command without 'LOAD' command repeated beforehand is not recommended. It is used only for keyboard mode convenience.

b) KILL MOTION

K — Decelerating stop to abort the motion of all motors without losing track of the motors' position. It is called a soft abort.

k (abort the motion of all motors softly)

Q — Stop immediately the motion of all motors without losing track of the motors' position. It is called a hard abort.

q (abort the motion of all motors immediately)

CHAPTER 6

EXAMPLES

6.1 TURBO PASCAL 3.0 ON DOS-PC

Following is an example for a PC-GPIB interface to the MMC32 using the TURBO PASCAL 3.0 language on a PC running under DOS. Comments are bracketed by { }.

6.1.1 SRQ handling

The following program moves motors 0,1 at the same time and finds out which motors finished moving.

```
Program MMC32Test;

  Data,Line,Keyin:string[80];
  i,x,Result,IeeeID,Bcount,v:integer;

begin
   { find out GPIB address ID }
   IeeeID:=IbFind('DEV6');
   v:=XEOS+REOS+LF;
   IbEos(IeeeID,v);
   { set motor 0,1 SRQ on }
   keyin:='n0 F64 n1 F64 ';
   Bcount:=Length(keyin);
   Send(IeeeID,keyin,Bcount);
   { load motor 0,1 motion parameters }
   { motor 1 moves 3000 steps in (+) direction.  }
   { motor 0 moves -3000 steps in (-) direction. }
   keyin:='L 1r 3000 0r -3000 255 ';
   Bcount:=Length(keyin);
   Send(IeeeID,keyin,Bcount);
   Writeln('Load command is sent out');
   { After loading moving parameters, execute GO command }
   keyin:='G ';
   Bcount:=Length(keyin);
   Send(IeeeID,keyin,Bcount);
   Writeln('Go command is sent out');
   for i:=1 to 2 do EndMove(IeeeID);
end.

procedure EndMove(IeeeID:integer);
var v,x:integer;
begin
```

```
  { put motor in Serial Poll Active state. }
  { If bit 6 (SRQ bit) is on, then you can check bit 0-5 }
  { to find out which motor finished moving. }
  v:=0;
  While (( v and 64 )=0) do ibrsp(IeeeID,v);
  x:=( v and 63);
  Writeln('Motor ',x,' finished moving');
end;

{ ------- Subroutine to send data to  motor controller.  ---------- }
procedure Send(Dev:integer; Message:DataString; Bcnt:integer);

var I:integer;

begin
    for I := 1 to Bcnt do ibbuf[I] := Message[I];
    IBWRT(Dev,ibbuf,Bcnt);
    if ibsta < 0 then Writeln('IEEE Write error from Send');
end;
```

6.2 C on UNIX-PC286

Following is an example for a CAMAC-GPIB interface to the MMC32 using the C language on a PC running under UNIX. Comments are bracketed by /* */.

6.2.1 Subroutine to send command string

The following subroutine sends the command string to the motor controller through the CAMAC-GPIB interface (model 3388-G1A).

```
smc_send(string)
char *string;
{
  int i,q2;
  long status;
  ff=26; aa=0; camdata=0;   /* attention */
  do { camac(nn,aa,ff,&qq,&xx,&camdata); } while (qq==0);
  ff=16; aa=0; camdata=58;   /* listen */
  do { camac(nn,aa,ff,&qq,&xx,&camdata); } while (qq==0);
  ff=16; camdata=58;   /* controller talk */
  do { camac(nn,aa,ff,&qq,&xx,&camdata); } while (qq==0);
  ff=24; camdata=0;   /* controller talk */
  do { camac(nn,aa,ff,&qq,&xx,&camdata); } while (qq==0);
  for ( i=0; i<strlen(string); i++) {
  ff=16; aa=0; camdata=string[i];   /* send byte */
  do { camac(nn,aa,ff,&qq,&xx,&camdata);
```

```
      if (qq==0) {
        do {camac(nn,aa,1,&q2,&xx,&status);} while (q2==0);
        if (status&0x20) return(0); }
    } while (qq==0);
  }
  return(1);
}
```

APPENDIX A

ALPHABETICAL COMMAND SUMMARY

Command	Description	Page
Annnn	setup Acceleration/deceleration register value	22
B	flush the GPIB and keypad input buffer	22
C0	Complete all motors' remaining steps from last move	24
C1	Complete the active motor's remaining steps from last move	24
DK	Disable keyboard mode except emergency stop and key mask	22
EK	Enable keyboard (local) mode	22
Fnnn	setup Flag status	22
G	Go (start executing move for load/go)	26
H	move to Home position	25
IA	Interrogate Acceleration/deceleration value	23
IE	Interrogate command Error message number	23
IF	Interrogate motor Flag status	23
IN	Interrogate active motor Number	24
IO	Interrogate active motor Output status	24
IP	Interrogate active motor current Position	24
IR	Interrogate active motor remaining steps	24
IU	Interrogate active motor start velocity	24
IV	Interrogate active motor peak Velocity	24
IX	Interrogate active motor CW (-) limit	24
IY	Interrogate active motor CCW (+) limit	24
IZ	Interrogate active motor go-home velocity	24
K	Kill motor motion softly	26
L	Load motor move parameters	25
M+/-nnnnnnnn	Move to position +/-nnnnnnnn steps	25
Nnnn	select active motor Number	22
P+/-nnnnnnnn	calibrate Position (32 bits long)	23
Q	Kill motor motion immediately	26
R+/-nnnnnnnn	move +/-nnnnnnnn steps (Relative move)	23
S1	Single step move in CCW (+) direction	25
S0	Single step move in CW (-) direction	25
T1	move to hardware CCW (+) limit	25
T0	move to hardware CW (-) limit	25
Unnn	setup start velocity	23
Vnnn	setup peak velocity	23
Wnnnn	set the number of pulses for Acceleration/deceleration	23
Zfff	set the output pulse rate multiplication value	23

APPENDIX B

MMC32 motor boards' CONNECTOR PINOUTS TABLE

P3	1	GROUND
	2	GROUND
	3	MOTOR 0 DIRECTION OUTPUT or CCW(+) PULSE OUTPUT
	4	MOTOR 0 PULSE OUTPUT or CW(-) PULSE OUTPUT
	5	MOTOR 1 DIRECTION OUTPUT or CCW(+) PULSE OUTPUT
	6	MOTOR 1 PULSE OUTPUT or CW(-) PULSE OUTPUT
	7	MOTOR 2 DIRECTION OUTPUT or CCW(+) PULSE OUTPUT
	8	MOTOR 2 PULSE OUTPUT or CW(-) PULSE OUTPUT
	9	MOTOR 3 DIRECTION OUTPUT or CCW(+) PULSE OUTPUT
	10	MOTOR 3 PULSE OUTPUT or CW(-) PULSE OUTPUT
	11	MOTOR 4 DIRECTION OUTPUT or CCW(+) PULSE OUTPUT
	12	MOTOR 4 PULSE OUTPUT or CW(-) PULSE OUTPUT
	13	MOTOR 5 DIRECTION OUTPUT or CCW(+) PULSE OUTPUT
	14	MOTOR 5 PULSE OUTPUT or CW(-) PULSE OUTPUT
	15	MOTOR 6 DIRECTION OUTPUT or CCW(+) PULSE OUTPUT
	16	MOTOR 6 PULSE OUTPUT or CW(-) PULSE OUTPUT
	17	MOTOR 7 DIRECTION OUTPUT or CCW(+) PULSE OUTPUT
	18	MOTOR 7 PULSE OUTPUT or CW(-) PULSE OUTPUT
	19	+5 VOLTS
	20	+5 VOLTS

Note: In directory mode, the output is direction/pulse, and in the pulse mode, the output is CCW(+)/CW(-).

P4	1	GROUND
	2	GROUND
	3	MOTOR 0 DIRECTION INPUT or CCW(+) PULSE INPUT or B phase INPUT
	4	MOTOR 0 PULSE INPUT or CW(-) PULSE INPUT or A phase INPUT
	5	MOTOR 1 DIRECTION INPUT or CCW(+) PULSE INPUT or B phase INPUT
	6	MOTOR 1 PULSE INPUT or CW(-) PULSE INPUT or A phase INPUT
	7	MOTOR 2 DIRECTION INPUT or CCW(+) PULSE INPUT or B phase INPUT
	8	MOTOR 2 PULSE INPUT or CW(-) PULSE INPUT or A phase INPUT
	9	MOTOR 3 DIRECTION INPUT or CCW(+) PULSE INPUT or B phase INPUT
	10	MOTOR 3 PULSE INPUT or CW(-) PULSE INPUT or A phase INPUT
	11	MOTOR 4 DIRECTION INPUT or CCW(+) PULSE INPUT or B phase INPUT
	12	MOTOR 4 PULSE INPUT or CW(-) PULSE INPUT or A phase INPUT
	13	MOTOR 5 DIRECTION INPUT or CCW(+) PULSE INPUT or B phase INPUT
	14	MOTOR 5 PULSE INPUT or CW(-) PULSE INPUT or A phase INPUT
	15	MOTOR 6 DIRECTION INPUT or CCW(+) PULSE INPUT or B phase INPUT
	16	MOTOR 6 PULSE INPUT or CW(-) PULSE INPUT or A phase INPUT
	17	MOTOR 7 DIRECTION INPUT or CCW(+) PULSE INPUT or B phase INPUT
	18	MOTOR 7 PULSE INPUT or CW(-) PULSE INPUT or A phase INPUT
	19	+5 VOLTS
	20	+5 VOLTS

Note: In the directory mode, the input is direction/pulse, in the pulse mode, the input is CCW(+)/CW(-), and in the encoded mode, the input is B/A phase.

P5	1	GROUND
	2	GROUND
	3	MOTOR 0 CCW (+) LIMIT SWITCH INPUT
	4	MOTOR 0 CW (-) LIMIT SWITCH INPUT
	5	MOTOR 1 CCW (+) LIMIT SWITCH INPUT
	6	MOTOR 1 CW (-) LIMIT SWITCH INPUT
	7	MOTOR 2 CCW (+) LIMIT SWITCH INPUT
	8	MOTOR 2 CW (-) LIMIT SWITCH INPUT
	9	MOTOR 3 CCW (+) LIMIT SWITCH INPUT
	10	MOTOR 3 CW (-) LIMIT SWITCH INPUT
	11	MOTOR 4 CCW (+) LIMIT SWITCH INPUT
	12	MOTOR 4 CW (-) LIMIT SWITCH INPUT
	13	MOTOR 5 CCW (+) LIMIT SWITCH INPUT
	14	MOTOR 5 CW (-) LIMIT SWITCH INPUT
	15	MOTOR 6 CCW (+) LIMIT SWITCH INPUT
	16	MOTOR 6 CW (-) LIMIT SWITCH INPUT
	17	MOTOR 7 CCW (+) LIMIT SWITCH INPUT
	18	MOTOR 7 CW (-) LIMIT SWITCH INPUT
	19	+5 VOLTS
	20	+5 VOLTS

P6	1	GROUND
	2	GROUND
	3	MOTOR 0 ON/OFF SWITCH INPUT
	4	MOTOR 0 HOME LIMIT SWITCH INPUT
	5	MOTOR 1 ON/OFF SWITCH INPUT
	6	MOTOR 1 HOME LIMIT SWITCH INPUT
	7	MOTOR 2 ON/OFF SWITCH INPUT
	8	MOTOR 2 HOME LIMIT SWITCH INPUT
	9	MOTOR 3 ON/OFF SWITCH INPUT
	10	MOTOR 3 HOME LIMIT SWITCH INPUT
	11	MOTOR 4 ON/OFF SWITCH INPUT
	12	MOTOR 4 HOME LIMIT SWITCH INPUT
	13	MOTOR 5 ON/OFF SWITCH INPUT
	14	MOTOR 5 HOME LIMIT SWITCH INPUT
	15	MOTOR 6 ON/OFF SWITCH INPUT
	16	MOTOR 6 HOME LIMIT SWITCH INPUT
	17	MOTOR 7 ON/OFF SWITCH INPUT
	18	MOTOR 7 HOME LIMIT SWITCH INPUT
	19	+5 VOLTS
	20	+5 VOLTS

********** Caution to User **********

Please note that the P3, P4, and P5 pinouts are E500 compatible. All the signals are in negative true logic at TTL levels. Facing the back panel, you will see connector P3 (labeled P3 on the board), P4, P5, P6 apearing from left to right. The pin number one on all connectors is marked with an arrow. The pins on the top layer are odd number pins (1,3,5.....), and the ones on the bottom layer are even number pins (2,4,6....).

If the chip U20 and the chip U21 on your PPMC boards are 74LS240 instead of 74LS244, then the limit switch is activated if the input signal is logical high (5 volts). That means if connector P4 is left float, or the input signal to the connector P4 on any pin 3 to pin 18 is high, then the limit switch is going to be activated, and you can no. move the motor. If you do not have any hardware limit switch installed, then you can not move any motor at all unless you have a shorting plug to connect the limit-switch pins to GROUND (pin number one and two). If you want the logic on the limit switches reversed, then you can exchang chips U20 and U21 with 74LS244.

DATE
FILMED
8/31/93
END