MSTEP-3
Triple Stepper Motor Controller Board
User Guide

KEITHLEY

User Guide

for the

MSTEP-3

Triple Stepper Motor

Controller Board

Revision B - April 1992

Part Number: 24421

KEITHLEY DATA ACQUISITION - KEITHLEY METRABYTE/ASYST

440 MYLES STANDISH BLVD., Taunton, MA 02780
TEL. 508/880-3000, FAX 508/880-0179

Warranty Information

All products manufactured by Keithley MetraByte are warranted against defective materials and worksmanship for a period of one year from the date of delivery to the original purchaser. Any product that is found to be defective within the warranty period will, at the option of Keithley MetraByte, be repaired or replaced. This warranty does not apply to products damaged by improper use.

Warning

Keithley MetraByte assumes no liability for damages consequent to the use of this product. This product is not designed with components of a level of reliability suitable for use in life support or critical applications.

Disclaimer

Information furnished by Keithley MetraByte is believed to be accurate and reliable. However, the Keithley MetraByte Corporation assumes no responsibility for the use of such information nor for any infringements of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent rights of Keithley MetraByte Corporation.

Note:

Keithley MetraByte™ is a trademark of Keithley Data Acquisition.

Basic™ is a trademark of Dartmouth College.

IBM® is a registered trademark of International Business Machines Corporation.

PC, **XT**, **AT**, **PS/2**, and **Micro Channel Architecture**® are trademarks of International Business Machines Corporation.

Microsoft® is a registered trademark of Microsoft Corporation.

Turbo C® is a registered trademark of Borland International.

Contents

CHAPTER 1: INTRODUCTION

CHAPTER 2: INSTALLATION

CHAPTER 3: PROGRAMMING

CHAPTER 4: MODE CALLS

CHAPTER 5: USING THE ACCESSORIES

APPENDICES

CHAPTER 1

INTRODUCTION

1.1 DESCRIPTION

Keithley MetraByte's MSTEP-3 is a plug-in, 3-axis, stepper-motor, motion-control board for the IBM PC/XT/AT and compatibles. The Board is 12 inches long and requires a full-length expansion slot. All connections necessary for operation with Keithley MetraByte's accessories are made through the rear plate using a standard 50-pin insulation-displacement (mass-termination) connector. Keithley MetraByte offers as options a stepper-motor driver (M3-DRIVE), a compatible power supply (M3-PWR-24), a popular type of stepper motor (STEP-MOT1), as well and a screw-terminal adapter (STA-50) that allows the MSTEP-3 to be connected to drivers other than the M3-DRIVE. Figure 1-1 is a block diagram of the MSTEP-3.

Each independent stepper channel consists of a Sil-Walker PPMC-103C intelligent controller chip capable of executing a variety of motion control commands. The PPMC-103C is one of the more popular Japanese robotics chips, providing essential features with simplicity of use. Keithley MetraByte's driver software further enhances the ease of stepper-motor control by personal computer.

Once a command is loaded into the PPMC-103C controller chip, the host computer is no longer burdened by the execution of the particular motion but may monitor its status as needed. The associated stepper motor may be moved any number of steps up to 24 bits of resolution (±16,777,216 steps) either with a controlled acceleration/deceleration profile or constant stepping rate. Associated with each motor are 5 limit switch inputs as well as a motor-enable input. The limit switches provide normal and emergency stop limits at both ends of travel, plus a home or reference point at any intermediate point. A normal stop is defined as a normal deceleration to rest without loss of the step count due to inertial effects, an emergency stop is a sudden stop that may lead to run on of the motor and hence loss of location from the step count and would normally require recalibration by return to the reference or home point. The emergency stop amounts to an immediate cessation of step pulses regardless of what the motor is doing at the time. In addition to controlling the number of steps travelled by the motor (normal motion), the PPMC-103C controller executes the following commands described below:

Initialization — Controls number of phases driven (3, 4 or 5), logic levels of phase excitation, (normal, inverted), internal/external step clock select and switching excitation on/off at standstill. In addition this command sets the start up, acceleration/deceleration and high speed run rates. The components of this command that relate to the motor configuration cannot be altered by a further initialization command without resetting the PPMC-103C controller.

Move Normal — Moves the motor the desired number of steps with a controlled rate and acceleration/deceleration.

Move Constant — Rotates motor at constant speed for a specified number of steps.

Find Limit — Rotates motor to an outer limit switch.

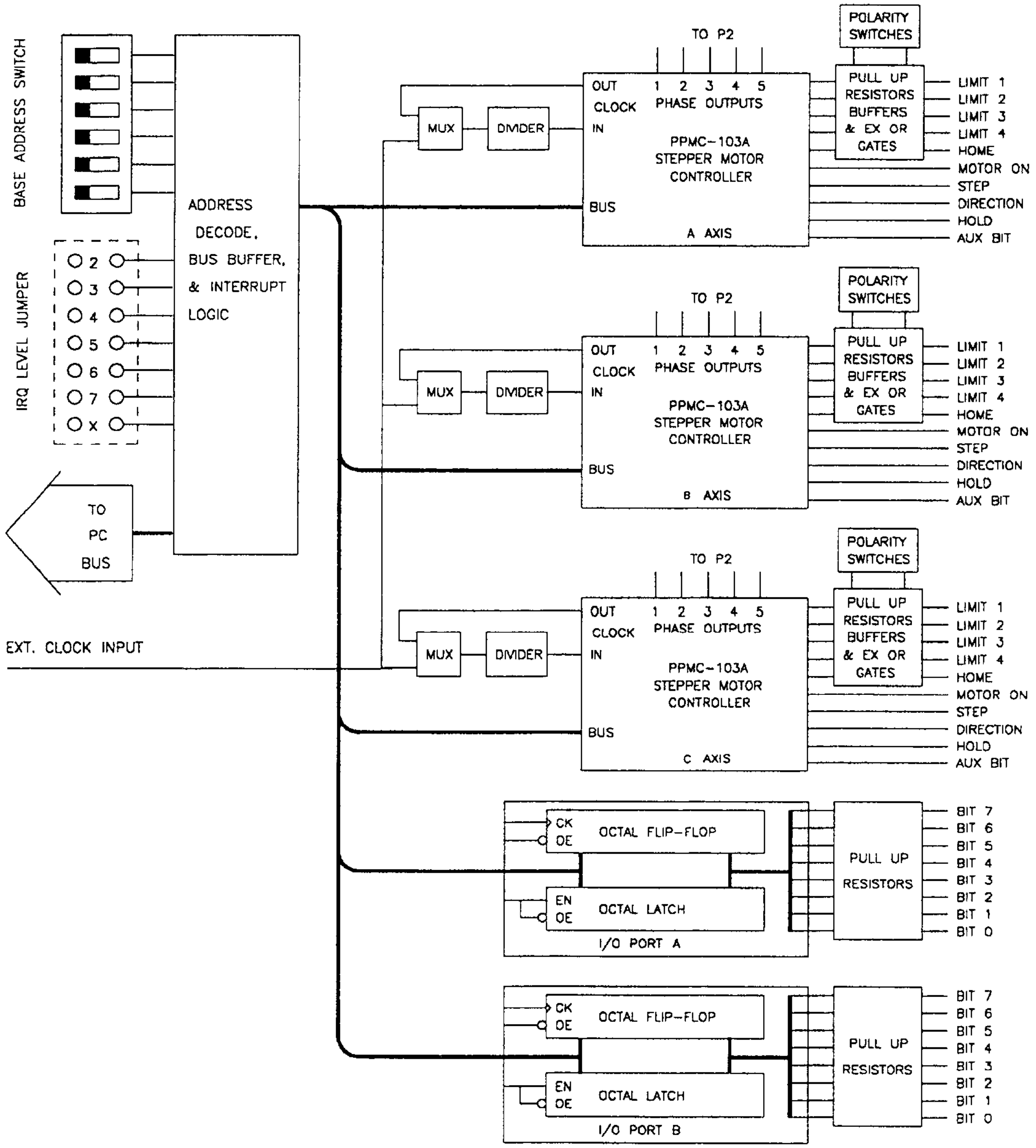


Figure: Block diagram of the MSTEP-3.

Find High-Speed Limit	Rotates motor to a high-speed or inner-limit switch.
Find Base Point	Rotates motor to home or reference-limit switch.
Read status	Read PPMC-103C controller status.
Decelerating Stop	Stops motor normally.
Emergency Stop	Instantly stops motor by removing drive pulses (may lead to loss of true location from step count).
Single Step	Single step or "jog" command.

Each stepper channel provides two different types of outputs. One is a counter-clockwise/clockwise (CCW/CW) signal plus a pulse-train output corresponding to the number of steps to be moved. This is suitable for driving the M3-DRV as well as a wide variety of standard stepper-motor translators available from most stepper-motor manufacturers. The other set of signals consists of 5-phase outputs used to drive power transistors to switch the stepper motor windings directly. The number of outputs enabled and the stepping sequence are controlled by the initialization command and can be matched to 3-, 4-, or 5-phase motors.

The M3-DRIVE sets up quickly to drive most small to medium size (i.e. 1-2 Amps /phase) stepper motors directly. This is a bipolar-chopper type of driver which provides better efficiency and often better torque at high speeds than R/L drives. Since this type of driver is superior for most applications, it is the only one sold by MetraByte for the MSTEP-3. However, if you desire to interface with an R/L driver, the necessary signals are accessible on the MSTEP-3 either through connector J1 at the rear of the card or connector P2 on the top edge of the card.

The step rate is controlled by the clock frequency. The PPMC-103A chip has an internal clock of 25 KHz. This clock has a programmable divider that can divide it by any number in the range of 13-255 (20-255 if not in turbo mode) giving corresponding step rates from 98 to 1923 pps (pulses per second). This range is often adequate, but you also have the option of selecting an external clock. There are two possible sources for this external clock. One is an on-board 800 KHz crystal controlled clock, and the other is a user supplied clock. Whichever source is selected is, in turn, passed through an 8254 clock divider before being fed to the PPMC-103A. Software determines which external clock is being used and the division ratio of the 8254. The division ratio must be selected so that with the chosen clock source, the input to the PPMC-103A is less than 267 KHz. This allows step rates as high as 19,000 pps. With the on-board 800 KHz clock, the lower limit is less than one pulse every 20 seconds (of course with the user supplied clock, there is no lower limit on the pulse rate).

All communications with the MSTEP-3 is via I/O ports (no memory address space is used); 16 contiguous addresses are used in I/O space. The Base Address is selected on a DIP switch and can be anywhere in the range 0 - 3F8 hex; 100-3F8 or 200-3F8 is the usual usable range in an IBM PC/AT/XT. More than one MSTEP-3 may be installed in a PC for multiple-axis control. The number is limited only by available expansion slots.

The PPMC-103C stepper controller chips can also generate interrupts on completion of commands and certain other conditions. This is supported in the MSTEP-3 hardware, interrupts may be jumper selected to any of the IBM PC Interrupt Levels 2-7 and in conjunction with the PPMC-103C Status Registers makes interrupt handshaking a simple procedure and allows the programmer to perform background control.

1.2 GENERAL AREAS OF APPLICATION

Many manufacturers produce stepper motors, stepper motor assemblies such as X-Y tables, etc. A short summary of sources appears in Appendix B. Keithley MetraByte can provide from stock a standard 5V, 1A, 200-step/revolution motor (STEP-MOT1); see Appendix C for specifications. These devices are representative only of typical hardware that can be used with the MSTEP-3.

Generally, stepper motors are suited to applications requiring variable torque, low speeds (not exceeding 500 rpm) and maximum position retention. They also offer the simplicity of open-loop position control simply by counting the number of steps. If your needs dictate higher speeds, fast acceleration and deceleration, and the ability to return to a set position regardless of load perturbations, a closed-loop DC servo drive may be more suitable. This type of drive is generally more complex and costly than a stepper drive but has characteristics that may be essential in certain applications.

The MSTEP-3 may be combined with other Keithley MetraByte measurement and control boards to implement complex "move and measure" type of instrumentation. It has obvious applications in robotics, optics and lasers, mechanical assemblies, remote control, etc.

1.3 UTILITY SOFTWARE

It is possible to program the MSTEP-3 directly using normal I/O port commands (INP and OUT etc.). This is explained in Section 3 and Appendix A, but for the programmer who wants fast results, our accompanying utility software (MSTEP.BIN) will simplify the use of the MSTEP-3 and save a lot of programming time. The utility software is provided on a single-sided PC-DOS 1.10 format 5-1/4" floppy disk (upward compatible with DOS 2.0 and higher revisions):

1. A Microsoft Basic callable driver (MSTEP.BIN) is provided for control of the basic stepper and encoder functions. The fully commented assembly source for this driver (MSTEP.ASM) is also provided. The object module, MSTEP.OBJ, is also on the disk for linking when using compiled BASICs (for example, IBM BASIC Compiler, Microsoft QuickBASIC, etc.).
2. Examples and demonstration programs. A comprehensive demonstration program (DEMO.BAS, DEMO.EXE) is provided. This is excellent both as a programming example and a way of getting the "feel" of the PPMC motion commands and driver software features. It will also be useful in your system setup and test. For further details see Chapters 3 and 7.
3. Instructions for ASSEMBLY LANGUAGE, C, Pascal and Fortran programmers are included in Appendix E.

1.4 TECHNICAL SUPPORT

If you have a problem or need information or advice, please call us at 508/880-3000 and ask for *Applications Engineering* . We will do our best to assist you. If for any reason you are dissatisfied with any Keithley MetraByte product or find it is unsuited to your requirements, you are welcome to return it within the first 30 days of purchase for a full refund. Please call us first for an RMA (Return Material Authorization) number before sending back any hardware. The MSTEP-3 and accessories are warranted against defects in manufacture and material for one year from the date of original purchase.

1.5 ACCESSORIES

Several optional accessories to facilitate the use of the MSTEP-3 are available from Keithley MetraByte. These include:

M3-DRIVE This is a driver which, with the addition of a power supply like the M3-PWR-24, allows the MSTEP-3 to control a stepper motor requiring from 1 to 2 Amps per phase. Screw terminals allow easy connections to M3-PWR-24, the motor, and limit or home switches, as well as one of the two general purpose I/O ports of the MSTEP-3. Up to three M3-DRIVE's can be connected to each MSTEP-3. They are connected in "daisy chain" (the MSTEP-3 is connected to the axis A M3-DRIVE, the Axis A unit is connected to the Axis B unit, and the Axis B unit to the Axis C).

M3-PWR-24 This is a power supply capable of supplying the power needed by one M3-DRIVE.

CDAS-2000 This is the cable to connect the Axis A M3-DRIVE to the MSTEP-3, or the Axis B M3-DRIVE to the Axis A output, or the Axis C M3-DRIVE to the Axis B output.

STA-50 This is a screw terminal adapter which would be used by customers not using M3-DRIVE's to drive their motors.

STEP-MOT1 This is a stepper motor compatible with the MSTEP-3 and the M3-DRIVE.

A complete 3 axis system can be built with one MSTEP-3, three STEP-MOT1s, three M3-DRIVEs, three M3-PWR-24s, and three CDAS-2000s. ■

❑

❑

❑

CHAPTER 2

INSTALLATION

2.1 BACKING UP THE DISK

The back-up software supplied with MSTEP-3 is in DOS 1.10 format and is compatible with DOS 2.0 and higher revisions. You are urged to use this utility to back up your MSTEP-3 Distribution Software at the earliest opportunity. For a direct backup, use the DOS DISKCOPY utility or alternatively COPY *.* to a preformatted disk. For a hard disk, simply use COPY *.* to transfer your Distribution Software to a directory of your choice (the Distribution Software is not copy protected). If for any reason you should misplace or destroy your MSTEP-3 Distribution Software, please contact Keithley MetraByte for a replacement copy.

2.2 UNPACKING AND INSPECTING

After you remove the wrapped board from its outer shipping carton, proceed as follows:

1. Place one hand firmly on a metal portion of the computer chassis (the computer must be turned Off and grounded) to discharge static electricity from the package and your body, thereby preventing damage to board components.
2. Carefully unwrap the board from its antistatic wrapping material.
3. Inspect the board for signs of damage. If any damage is apparent, return the board to the factory.
4. Check the contents of your package against its packing list to be sure the order is complete. Report any missing items to MetraByte immediately.

You may find it advisable to retain the packing material in case the board must be returned to the factory.

2.3 SWITCH & JUMPER SETTINGS

Base Address Switch

MSTEP-3 requires 16 consecutive address locations in I/O space. Some I/O addresses will already be in use by internal I/O and your other peripheral devices. Avoid setting the MSTEP-3 to the same address as any other device already installed in your machine. A conflict of addresses will not cause physical damage but may cause malfunction of the MSTEP-3 and the conflicting adapter and, in some circumstances, the Power On Self Test (POST) diagnostic messages. To avoid conflict with these devices, you may change the MSTEP-3's preset I/O address by resetting the Base Address DIP switch; any new setting should be on an 16-bit boundary anywhere in the PC's available I/O space.

The PC-XT I/O address space extends from decimal 512-1023 (Hex 200-3FF) and the PC-AT I/O address space extends from decimal 256 to 1023 (Hex 100-3FF). In either case, the available space is never likely to be fully occupied and is more than enough to accommodate more than one MSTEP-3 in a single computer.

For your convenience, the reserved I/O addresses for standard IBM devices are detailed in the following table.

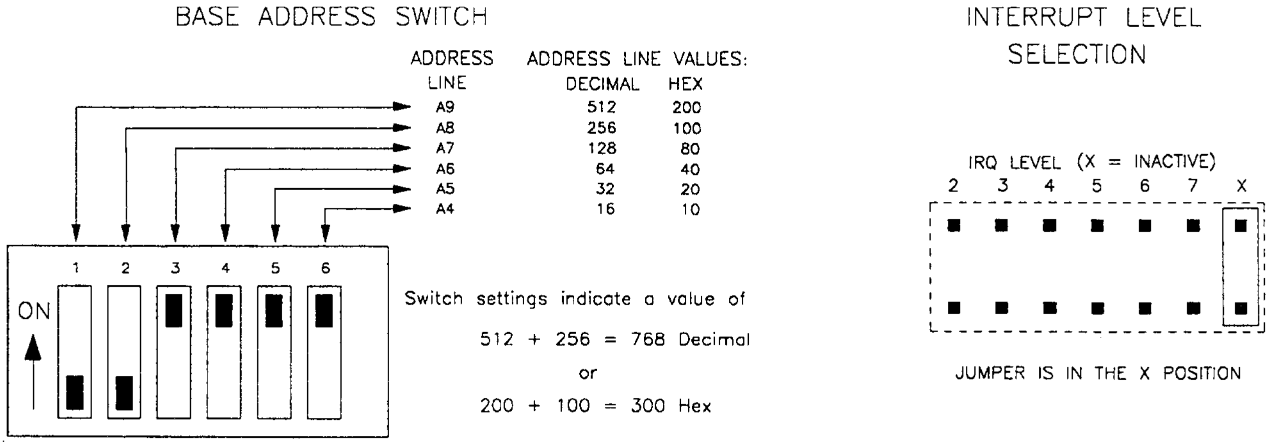
HEX RANGE	USAGE	HEX RANGE	USAGE
000 to 1FF	Internal System	37F to 387	LPT1:
200 to 20F	Game	380 to 38C	SDLC comm.
210 to 217	Expansion unit	380 to 389	Binary comm. 2
220 to 24F	Reserved	3A0 to 3A9	Binary comm. 1
278 to 27F	Reserved	3B0 to 3BF	Mono dsp/LPT1:
2F0 to 2F7	LPT2:	3C0 to 3CF	Reserved
2F8 to 2FF	COM2:	3D0 to 3DF	Color graphics
300 to 31F	Prototype card	3E0 to 3E7	Reserved
320 to 32F	Hard disk	3F0 to 3F7	Floppy disk
		3F8 to 3FF	COM1:

This list covers the standard IBM I/O options (most compatibles are identical), but if you have other I/O peripherals (special hard disk drives, special graphics boards, prototype cards, etc.), they may be making use of I/O addresses not listed in the table.

Usually, a good starting choice of Base Address is 300h or 310h (768 or 784 Decimal). (Note if you are using an IBM prototype board, it uses the 300-31F (Hex) address space and would conflict; 330h or 340h would be a good alternative in this case).

An aid to setting the Base Address DIP switch is the graphical program *DIPSW.EXE* , in your Distribution Software. This program may be run from the DOS prompt by typing **DIPSW** .

When you get the `Desired base address?` prompt, type in your choice in decimal or IBM &H--- format and press < **Enter** > . The program rounds your address to the nearest 8-bit boundary, checks for conflicts with standard I/O devices (and warns you if so), and displays the correct positions of the seven toggles on the Base Address DIP switch. For additional details on Base Address switch settings, see the following diagram.



ADDRESS LINE	DECIMAL	HEX
A9	512	200
A8	256	100
A7	128	80
A6	64	40
A5	32	20
A4	16	10

Switch settings indicate a value of

512 + 256 = 768 Decimal

or

200 + 100 = 300 Hex

Base Address switch and Interrupt Level selection.

Interrupt Level Selection

Referring to the diagram, the choice of Interrupt Level depends on the placement of a jumper. Very likely, you will not initially make use of the interrupt capabilities of the MSTEP-3 and can place the IRQ LEVEL jumper in the X position. If your programming will use interrupts from the MSTEP-3, then select the Interrupt Level (2 thru 7) you intend to use. Take care to avoid selecting a level in use by another adapter card (for example, Level 6 is always used by the floppy disk controller, Level 4 by COM1:, Level 3 by COM2: etc.). For more information on interrupt programming, see MODE 11 in Chapter 3.

HOME/LIMIT Polarity Switch.

DIP Switch S1 controls polarity of the Home and Limit switch inputs. For example, if any axis of your system does not need limit switches you may want to set the Limit Switch for that axis to On (Up position), as this will allow the motor to move with no limit-switch inputs. You may also use this position if you use normally open limit switches.

If, however, fail-safe operation is important, you may want to set the appropriate switch to Off (Down position). Under these conditions, normally closed limit switches must be used to insure that if any wire from a switch to the MSTEP-3 should open up, the MSTEP-3 will cause the motor for that axis to stop.

Similarly, the Home Switches are set for the polarity of the Home Switch used in each axis. Setting a Home Switch On will cause the MSTEP-3 to be looking for a low input when searching for Home, while the Off position will cause the board to look for a high level.

It should be noted that all Limit and Home Switch inputs have a pull-up resistor to +5 Volts and are then buffered through a Schmitt trigger IC in the "LS" family.

2.4 HARDWARE INSTALLATION

WARNING: ANY ATTEMPT TO INSERT OR REMOVE A BOARD WITH THE COMPUTER POWER ON COULD DAMAGE YOUR COMPUTER!

1. Turn Off power to the PC and all attached equipment.
2. Remove the cover of the PC as follows: First remove the cover-mounting screws from the rear panel of the computer. Then, slide the cover of the computer about 3/4 of the way forward. Finally, tilt the cover upwards and remove.
3. Choose any available option slot. Loosen and remove the screw at the top of the blank adapter plate. Then slide the plate up and out to remove.
4. Hold the Board in one hand placing your other hand on any metallic part of the PC/AT chassis (but not on any components). This will safely discharge any static electricity from your body.
5. Make sure the board switches have been properly set (refer to the preceding section).
6. Because of the length of the 50-pin rear connector, you are advised to set its retainer latches out straight (an elastic band will hold them in position), and then to pass it through the rear slot and pivot the Board down into the edge connector. The Board is shaped to allow for this maneuver.
7. Gently press the board downward into the socket. Secure the Board in place by inserting the rear-panel adapter-plate screw.
8. Replace the computer's cover. Tilt the cover up and slide it onto the system's base, making sure the front of the cover is under the rail along the front of the frame. Replace the mounting screws.
9. Plug in all cords and cables. Turn the power to the computer back on.

Remember , TURN OFF THE POWER whenever installing or removing any peripheral board, as it can cause costly damage to the electronics of your computer and/or the MSTEP-3 board.

If for any reason you later remove the MSTEP-3 board, Keithley MetraByte recommends that you retain the special electrostatically shielded packaging and use it for storage. ■

PROGRAMMING

3.1 GENERAL

At the lowest level, the Stepper Motor Controller is programmed using I/O input and output instructions. In BASIC, these are the INP(X) and OUT X,Y functions. In Assembly Language, they are IN AL,DX & OUT DX,AL. Most other high level languages have equivalent instructions. Use of these functions usually involves formatting data and dealing with absolute I/O addresses. Although not demanding, this can require many lines of code and necessitates an understanding of the devices, data format, and architecture of the MSTEP-3. To simplify program generation, a special I/O driver routine "MSTEP.BIN" is included in the MSTEP-3 software package. This may be accessed from BASIC by a single line CALL statement. The CALL MSTEP will perform frequently used sequences of instructions. An example is Mode 1 which performs a normal stop. A routine to perform this operation in BASIC using INP's and OUT's would require many lines of code and would be rather slow and very tedious to program.

A sequence of BASIC INP's and OUT's to write the Stop Command to PPMC channel A would be as follows:

```
x10  BASADR% = &H300                'Set BASE ADDRESS.
x20  CNR% = INP (BASADR%+4)         'Read current data select req.
x30  OUT (BASADR+4),&H1             'Set AORSEL = 1 to read status req.
x40  STAT% = INP (BASADR%+1)        'Read Axis A status Req.
x50                                 'Motor stopped?
x60  IF (STAT% AND &H4) = 0 GOTO 140 'Yes, restore data select req; exit.
x70                                 'Input buffer full?
x80  IF (STAT% AND &H2) <> 0 GOTO 40 'Yes, try again!
x90  CONF% = INP (BASADR%)          'Read and save configuration Req.
x100 CONF% = CONF% AND &HFB
x110 OUT BASADR%+4,CONF%            'Disable Axis A Interrupt.
x120 OUT BASADR%+4,&H1              'Set AORSEL = 1 to write status req.
x130 OUT BASADR%+1,&H40             'Stop motor.
x140 OUT BASADR%+1, CNR%            'Restore data select req. value.
x150 STOP
```

All this code can be circumvented by using the driver:

```
x10  MD% = 1                                 'command - decelerating stop
x20  D%(0)  =  0                             'select channel A
x30  CALL MSTEP (MD%, D%(0), STP#, FLAG%)
```

Obviously, the MSTEP.BIN driver greatly reduces programming time and effort. Both methods of programming are described (see Appendix A for programming the PPMC-103A directly) and you are free to choose either. Usually the BASIC programmer will find the CALL routine method very much simpler to implement. The MSTEP.BIN driver also provides an example interrupt service routine. It is not possible to program interrupt routines directly in BASIC and the driver is the only way of utilizing interrupts from the MSTEP-3 hardware.

3.2 REGISTER LOCATIONS & FUNCTIONS

The following tables describe the locations and functions of the registers within the MSTEP-3 board.

Base Address +0

Configuration Select Register

BIT:	7	6	5	4	3	2	1	0
READ/WRITE:	XCC	CXB	XCA	EINTC	EINTB	EINTA	OEPB	OEPA

XCA, XCB, XCC: These bits determine the source of the clock frequency input into the 8254 divider for each respective axis. Setting any bit low results in the internal 800KHz clock being used, while setting it high results in the user-supplied clock of Pin J1-25 being used. The output of each respective frequency divider is used by the PPMC-103A whenever an external clock is specified.

EINTA, EINTB, EINTC: Setting any of these bits high enables the respective axis to generate an interrupt when the motor stops moving. Note that the jumper on the card must be set to the appropriate Interrupt Level, or the PC will not receive the interrupt.

OEPA, OEPB: These bits control the Output Enable function of Ports A and B, respectively. Setting the Enable Bit of a port high sets the entire port for Output. Both ports can be read whether they are configured for Output or Input.

Base Address +1

A Axis PPMC-103A: this is the address to read or write to the A Axis PPMC-103A. Refer to the Sil-Walker data sheet for more information.

Base Address +2

B Axis PPMC-103A: This is the address to read or write to the B Axis PPMC-103A. Refer to the Sil-Walker data sheet for more information.

Base Address +3

C Axis PPMC-103A: This is the address to read or write to the C Axis PPMC-103A. Refer to the Sil-Walker data sheet for more information.

Base Address +4

Data Select Register

BIT:	7	6	5	4	3	2	1	0
READ/WRITE:	x	x	x	x	x	x	x	AORSEL
READ:	x	x	x	x	\INTC	\INTB	\INTA	-

x: Does not matter.

AORSEL: Controls Bit A0 into both of the PPMC-103As.
Use A0 = 1 to write a command or read the status.
Use A0 = 0 to read or write data.

\INTA, \INTB, \INTC: These bits are set low when the motor stops; they remain low until reset by reading the finish STATUS.

Base Address +5

I/O Port A: This is a general-purpose, 8-bit I/O port. It can be written to or read at any time, but the data written to it will be output only on the appropriate pins of Connector J1 if Port A is set for Output by setting OEPA at Base Address +0 high.

Base Address +6

I/O Port B: This is a general-purpose, 8-bit I/O port. It can be written to or read at any time, but the data written to it will be output only on the appropriate pins of Connector J1 if Port A is set for Output by setting OEPB at Base Address +0 high.

Base Address +7 Through Base Address +11

These addresses are not used in the current MSTEP-3 design.

Base Address +12

8254 Counter 0 (A Axis): This address corresponds to Counter 0 of the 8254, which controls the division ratio that the 8254 applies to the 800KHz clock or the user-supplied clock for the A Axis. For more information on using this divider, refer to an 8254 data sheet.

Base Address +13

8254 Counter 1 (B Axis): This address corresponds to Counter 1 of the 8254, which controls the division ratio that the 8254 applies to the 800KHz clock or the user-supplied clock for the B Axis. For more information on using this divider, refer to an 8254 data sheet.

Base Address +14

8254 Counter 2 (C Axis): This address corresponds to Counter 0 of the 8254, which controls the division ratio that the 8254 applies to the 800KHz clock or the user-supplied clock for the C Axis. For more information on using this divider, refer to an 8254 data sheet.

Base Address +15

This address corresponds to the Control Word of the 8254. For more information on using this divider, refer to an 8254 data sheet.

3.3 PPMC-103A STEPPER CONTROLLER INTERNAL REGISTERS

Each PPMC-103A Stepper Controller is a specialized microprocessor that controls the stepper motor for one axis. The I/O address (Base Address +1, +2, or +3,) selects which axis is being addressed. When addressing a PPMC-103A, the AORSEL bit (Bit 0 of Base Address +4) determines which register within the device is being addressed.

AORSEL	READ	WRITE
0	DATA	DATA
1	STATUS	COMMAND

Motion commands are issued to the PPMC controller by writing a command code to the Command Register (with AORSEL = 1). Since the PPMC may be busy executing a command, you must always read the STATUS register to determine whether the PPMC is ready to receive the command. After a command is issued, it may require or produce a variable number of data bytes (depending on the command) which are written or read from the Data Registers (with AORSEL = 0). Apart from controlling access to the Command Register, the Status register also provides additional information on the operation of the PPMC controller.

There are eight motion-control commands as well as the INITIALIZATION command that set the operating conditions of the controller. The functions of the Command and Status Registers are as follows:

INITIALIZATION

Selects the motor type, method of excitation acceleration/deceleration rate, internal/ external step clock, phase output logic type and start up and high speed pulse rate. Once the initialization command has been sent, it may not be overwritten by a further initialization command. The PPMC controller must be reset either by turning the computer power off and on or more conveniently by issuing a hardware clear command (see Sections 3.2.4 and 3.8.13 - mode 12).

OPERATION

This is the user interface mode in which COMMAND selects any of eight motion-control commands. The length of the data to follow depends on the specific command.

STATUS

Before/after the completion of an operation command, the status register provides data on the limit switches, motor in motion or at standstill and input/output data buffer full or valid. It also enables you to read the number of steps remaining to be travelled etc.

Direct programming of the PPMC-103A controllers and their full specification is more fully covered in Appendix A.

3.4 LOADING THE MACHINE-LANGUAGE CALL ROUTINE MSTEP.BIN

In order to make use of the CALL routine MSTEP.BIN, it must first be loaded into memory. You must avoid loading it over any part of memory that is being used by the main body of your program, DOS, or programs such as RAM disks that use high memory. If you do collide with another program, your computer will usually hang up although sometimes the results can be more peculiar. Often you will need to turn the power off and restore it to re-boot the machine, the usual Ctrl-Alt-Delete reset may fail to restore DOS. This may sound ominous, but apart from the frustration, no damage will ever result!

MSTEP.BIN uses about 3 Kbytes of memory and is best loaded outside BASIC's workspace. A typical loading sequence is as follows:

```
xx100 DEF SEG = &H3000       'segment of memory to load link
                             (choose an empty area e.g. @@ 192K)
xx110 BLOAD "MSTEP.BIN",0    'load driver   . .  Continue program
```

The above initializing steps will be the same for any interpreted BASIC program. A more comprehensive example is provided on the disk in DEMO.BAS. Note that the DEF SEG = &H3000 statement in line 100 specifies the load location for the MSTEP.BIN driver. All subsequent CALL's will occur to the last DEF SEG address, so if you add other DEF SEG's in your program, remember to precede your CALL's to MSTEP-3 with the same DEF SEG = &H3000 that you used to load the link (see CALL and DEF SEG in your BASIC reference manual).

Finding a place to load MSTEP.BIN is seldom much of a problem now that most PC's are equipped with at least 256K of memory. The following explanation provides some insight into the process of choosing a memory location for the driver and what to do if memory is in short supply.

DOS occupies the bottom of memory, the amount of memory required being dependent on the version (it grows as each new revision adds extra features!). The simplified memory map below shows what happens after booting up BASICA.

		DOS 1.1		DOS 2.1		DOS 3.0
Bottom:	0K	-------	0K	-------	0K	------
		DOS				
	19K	-------				
				DOS		
						DOS
			47K	-------		
		BASIC			63K	------
				BASIC		
						BASIC
	98K	-------				
		Free				
		memory	126K	-------		
				Free	140K	------
				memory		
						Free
						memory

MSTEP.BIN should be loaded somewhere in the free memory area so that it does not interfere with either BASIC or DOS. This would be above 98K (&H1880) for DOS 1.1, 126K (&H1F80) for DOS 2.1 or 140K (&H2300) for DOS 3.0. If you have 256K (&H4000) or more of memory, then loading the link at DEF SEG = &H2800 or &H3000 is a good solution for all versions of DOS. One further small detail is that if you are using a PC compatible that does not have BASIC in ROM (like the IBM PC), then BASIC (for example, GWBASIC) is usually loaded as an .EXE file from the top of memory down. This is likely to fill up to 64K of the top segment of memory. Some virtual disks or print spoolers will do the same. Also if you are accustomed to using DOS resident programs such as Borland's Sidekick, etc., be aware that these will push the loading floor of BASIC up and require a compensating increase in the location of MSTEP.BIN.

If you are memory limited, or you have so much resident material that there is no longer 64K left for BASIC to load in, then BASIC will attempt to make the most of what it can find. Instead of getting the message when BASIC has loaded:

```
        The IBM Personal Computer Basic
        Version A2.0 Copyright IBM Corp. 1981, 1982, 1983
        60865 Bytes free
        OK
```

You may get only 49345 bytes free (or something less than 60000 bytes) for example. In this case make a note of what space BASIC has found. You can then contract this space further using the CLEAR function and load the link at the end of BASIC. This is more complicated, but just as effective.

Let's suppose you get the message 52000 bytes free. MSTEP.BIN will use 3K bytes, so to be on the safe side let's force BASIC to use 48K. The initializing code would now be:

```
xx100 CLEAR, 48000          'contracts BASIC workspace
```

Next, you must find out where BASIC has loaded in memory, add 48000 to it and load MSTEP.BIN just after the end of BASIC workspace. Memory locations &H510 and &H511 always contain BASIC's load segment:

```
xx110 DEF SEG = 0                       'set up to read &H510 and &H511
x120 LS = 256*PEEK(&H511)+PEEK(&H510)   'load segment
x130 SG = LS + 48000/16                 'remember segment addresses are on
                                        '16-byte (paragraph) boundaries
x140 DEF SEG = SG                       'set up to load link
x150 BLOAD "MSTEP.BIN",0                'load link
   . .
```

Proceed with your program as before. Note that this procedure does not work with GW BASIC.

3.5 STRUCTURE OF THE CALL STATEMENT

If you are unfamiliar with CALL statements, this explanation may assist you in understanding how the CALL transfers execution to the machine language (binary) driver routine (also see CALL in your Basic Reference Manual). Prior to entering the CALL, the DEF SEG = SG statement sets the segment address at which the CALL subroutine is located. The CALL statement for the MSTEP.BIN driver must be of the form

```
xxxx CALL MSTEP (MD%, D%(0), STP#, FLAG%)
```

Let us examine the parameters after CALL one by one:

MSTEP In interpreted BASIC this is a variable that specifies the offset of the start of our routine from the segment defined in the last DEF SEG statement. In our case its value is always set to zero (MSTEP = 0). In compiled BASIC (and most other compiled languages) MSTEP has a different significance - it is the name of the external routine that the linker will look for. Note: We would have liked to use the name STEP instead - it's a better mnemonic than MSTEP, but be warned that STEP is a reserved word (as in FOR I=0 TO 6 STEP 2) and CALL STEP would produce a syntax error.

MD% This is an integer variable that specifies the operation that we wish the driver to perform e.g. MD%=0 performs an emergency stop, MD% = 12 initializes a channel etc. In the case of this driver, valid mode numbers range from 0 to 12.

D%(9) This is a 10-element integer array that passes data to and from the driver. The signifigance of particular data items varies according to the mode (MD%) selected. Not all elements of D%(*) are used in all modes.

STP# This is a double precision variable that specifies the direction and number of steps to travel or returns optical shaft encoder counts. The sign indicates the direction, + clockwise, - counter-clockwise. Not all modes utilize the STP# data , however it must always be included in the call parameter list.

FLAG% Returns an error code if any of the specifying D%(*) or MD% are out of range or if the motor is busy or at standstill in certain commands. In the case of no error, FLAG% is returned zero.

The four variables within brackets are known as the CALL parameters. On executing the CALL, the addresses of the variables (pointers) are passed in the sequence written to BASIC's stack. The CALL routine unloads these pointers from the stack and uses them to locate the variables in BASIC's data space so data can be exchanged with them. Three important format requirements must be met, as follows:

1. The CALL parameters are positional. The subroutine knows nothing of the names of the variables, just their locations from the order of their pointers on the stack. If you write:

   ```
   xxxxx CALL MSTEP (D%(0), FLAG%, MD%, STP#)
   ```

   you will mix up the CALL routine, since it will interpret D%(0) as the mode data, and FLAG% as the D%(0) data variable etc. The parameters must always be written in the correct order, as follows: mode #, data, step count, errors

2. The CALL routine expects its parameters to be of correct type and will write and read to the variables on this assumption: integer, integer array, double precision, integer

   If you slip up and use the wrong variable types in the CALL parameters, the routine will not function correctly and may hang up the program.

3. You **cannot** perform any arithmetic functions within the parameter list brackets of the CALL statement. There can only be a list of variables. Also you are not allowed to replace variables by constants.

Apart from these restrictions, you can name the variables what you want, the names in the examples are just convenient mnemonics. You should always declare the variables before executing the CALL so that BASIC has reserved memory locations for them before entering the CALL. In the case of the integer array, the first element D%(0) should be specified in the CALL parameter list as the data variable so that the CALL routine can locate all of the other remaining data items in the array correctly.

3.6 ERROR CODES

Some value checking is performed on entry data and any errors discovered are returned in FLAG%. This is primarily to prevent you setting up the CALL with obviously incorrect data such as interrupt level 9, mode number -6, base address 2000, byte output data 299 etc. and is intended to help avoid a bad setup of the hardware which could hang the computer. Also certain commands, such as an emergency or decelerating stop are redundant if the motor is already at standstill (FLAG% = 7) and the PPMC controller may not be receptive to further commands if it is already busy executing a command (FLAG% = 1). If a non-zero error code is returned in any mode, execution of that mode will have been abandoned without action since error checking precedes any I/O to the hardware.

ERROR CODE #	PROBLEM
0	No error, OK.
1	Motor busy.
2	Driver not initialized on Channel A.
3	Driver not initialized on Channel B.
4	Driver not initialized on Channel C.
5	Mode number <0 or >15.
6	Hardware error.
7	Step count out of range +/-16,777,215.
8	Motor already at standstill.
9	Motor switching time at standstill; does not set.
10	Error in range of D%(0).
11	Error in range of D%(1).
12	Error in range of D%(2).
13	Error in range of D%(3).
14	Error in range of D%(4).
15	Error in range of D%(5).
16	Error in range of D%(6).
17	Error in range of D%(7).
18	Error in range of D%(8).
19	Error in range of D%(9).
20	Error in range of D%(10).
21	Error in range of D%(11).
22	Error in range of D%(12).
23	Error in range of D%(13).
24	Error in range of D%(14).

Checking for errors is easily performed after each CALL and is recommended if not as a permanent feature then at least while debugging your program:

```
xxx00 CALL MSTEP (MD%, D%(0), STP#, FLAG%)
xx10 IF FLAG%<>0 THEN PRINT "Error number ";FLAG%:STOP
```

Certain error codes are useful in performing chained motion commands, as follows:

```
xxx00 MD% = 3                            'accelerate/decelerate command
xxx10 STP# = 1000                        'steps & direction to move
xxx20 D%(0) = 1                          'select channel B
xxx30 CALL MSTEP (MD%, D%(0), STP#, FLAG%)
xxx40 IF FLAG%<>0 THEN PRINT "Error number ";FLAG%:STOP
xxx50 STP# = -99                         'now do 99 steps in opposite
                                               direction
xxx60 CALL MSTEP (MD%, D%(0), STP#, FLAG%)
xxx70 IF FLAG% = 1 THEN GOTO xxx60       'loop while motor busy
```

```
xxx40 IF FLAG%<>0 THEN PRINT "Error number ";FLAG%:STOP
```

Note that after issuing a motion command in line xxx30 the controller and motor will still be busy executing this command when the program reaches the next motion command in line xxx60. In this case, FLAG% will return 1 and we can keep trying to execute line xxx60 until the previous command of line xxx30 has finished. In this way we can execute a whole series of commands as fast as the hardware will allow.]

3.7 STEPPER MOTOR FUNDAMENTALS

In order to program a stepper motor channel, it is necessary to understand a little bit about the physical setup and characteristics of a stepper motor. The motor itself consists of a permanent magnet rotor rotating within a multipole stator. As the current is switched in the windings of the stator, the magnetic field advances from one pole to the next and pulls the rotor along with it (see Section 6 for more information). The rotor can be turned a precise amount simply by controlling the number of energization/de-energization cycles of the windings (steps).

On turning on the power, the position of the motor is usually unknown, so it is necessary to move it to some known reference position called the home, base or reference point. When the mechanical system is in this position it can operate a switch (microswitch, hall effect or optical interrupter) and this provides the BP (base point) limit switch input. Once at the base point, all further commands can move relative to this known location, and as long as there is no loss of power or emergency stops the subsequent position can always be determined from the step count.

In practical systems, there is usually some physical limit on how far the motor can turn e.g. a lead screw gets wound to an end. To stop the motor, there is provision for 4 sets of limit switches, L1 - L4. L3 and L4 provide inner or deceleration limits where it is desirable to stop a fast moving motor even though it has not reached the absolute physical limit of travel. L1 and L2 are end of travel limits (or emergency stop limits) where it is essential to stop the motor immediately. A typical physical representation of a system is shown in Figure 3-1.

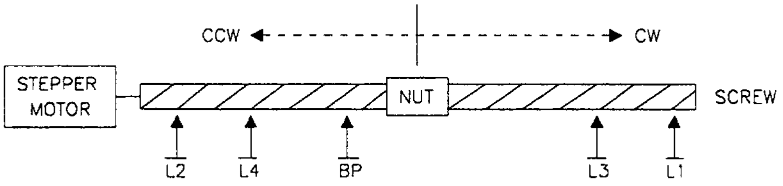


Figure 3-1. Typical mechanical arrangement.

Note that a mechanical system can be simplified and does not have to have 5 limit points, although the PPMC-103A controller can handle it. For example, L2 & L4 & CNP may be one physical limit switch, and L1 & L3 another for a 2 limit switch system. If you do this, you may lose some performance features of the PPMC, but gain in mechanical simplicity. Simply common up the limit switch inputs as required.

A mechanical system has significant inertia and it is not usually possible to start or stop the motor abruptly. The PPMC-103A controllers look after this problem by starting up the motor at a slow rate, ramping it up to full speed over a prescribed number of steps, and slowing it back down again before reaching the final step count (see Figure 3-2). All these parameters are set in the initialization command (mode 12). The inner limit switches (if used) allow you to perform a controlled stop without loss of positional count before hitting the outside or emergency stop limits. Whenever the motor hits an emergency limit switch, L1 or L2, during a motion command, it will stop. Which limit switch it hit and how many pulses remain to be executed in the last command and what was the last motion command can all be determined from a status read (mode 8). In this way the controller prevents overtravel and damage to the mechanical system and the programmer can provide appropriate corrective action depending on which of the limit switches were activated.

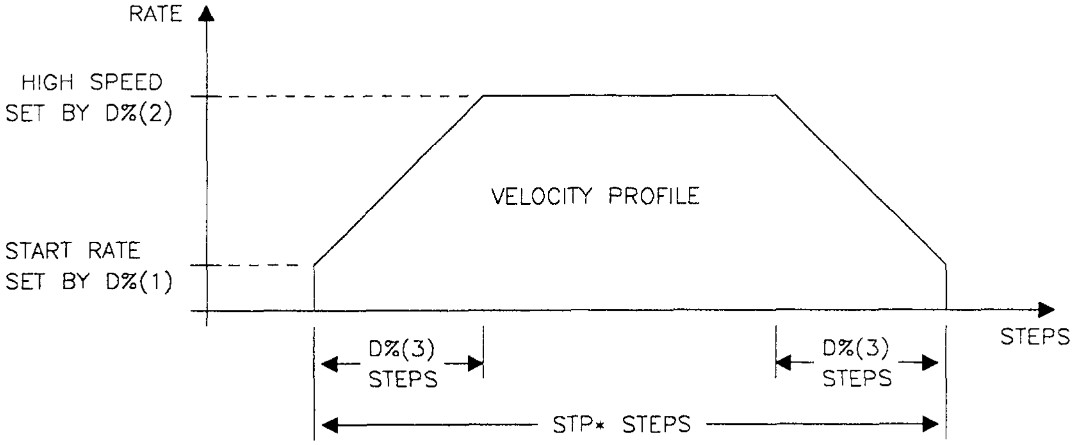


Figure 3-2. Acceleration/deceleration profile (MODE 3).

3.8 CALL MODES

The MSTEP.BIN driver supports sixteen different modes (numbered 0 thru 15). For simplicity, MODEs 0 thru 7 correspond identically to the command numbers in the PPMC-103A data sheet (see Appendix A). Each mode performs a specific operation, as described below:

MD%	= 0	Emergency stop.
	= 1	Decelerating stop.
	= 2	Single step or "jog."
	= 3	Step with acceleration/deceleration.
	= 4	Step at constant speed.
	= 5	Move to a outer limit at constant speed.
	= 6	Move to limit at high speed.
	= 7	Move to base point at constant speed.
	= 8	Read motor status.
	= 9	Load external clock divider.
	= 10	Read data from all ports.
	= 11	Write data to all ports.
	= 12	Write data to one port.
	= 13	Toggle auxiliary bit.
	= 14	Enable/disable an interrupt.
	= 15	Initialization.

The use of each mode is described in the following subsections. **It is essential to perform a channel initialization (MODE 15) separately on each channel before selecting any other MODE (0 - 14).** ■

CHAPTER 4

MODE CALLS

4.1 MODE 0: EMERGENCY STOP

MODE 0 performs an emergency stop or immediate cessation of driving pulses to a motor that is busy executing a motion command. On receipt of the command, inertia may cause a rotating motor to run on, so the step count may no longer be an accurate guide to the motor position. An emergency stop would usually be followed by a re-calibration or return to home position.

Entry Data:

MD% = 0	Emergency stop command
D%(0)	Selects Axis A (0), Axis B (1), or Axis C (2)
D%(1) thru (14)	Value irrelevant
STP#	Value irrelevant
FLAG%	Value irrelevant

Exit Data:

D%(0) thru (14)	Unchanged
STP#	Unchanged
FLAG%	0 if executed OK 2 if driver not initialized on Axis A 3 if driver not initialized on Axis B 4 if driver not initialized on Axis C 5 if MD% <0 or >15 6 if hardware error 8 if motor already at standstill (command redundant) 11 if D%(0) not 0, 1, or 2

A typical program entry preceding MODE 0 might read as follows:

```
xxx10 MD% = 0                           'select MODE number
xxx20 D%(0) = 1                         'select Axis B
xxx30 CALL MSTEP (MD%, D%(0), STP#, FLAG%)
                                        'emergency stop
xxx40 IF (FLAG%<>0 AND FLAG%<>8) THEN PRINT "Error";FLAG%;" in MODE 0":STOP
xxx50                                   'continue program
```

One detail concerning the preceding example: If FLAG% = 8 is returned, it is just a reminder that you told the controller to do an emergency stop when in fact the motor was already stopped. In this case the driver aborts the command and lets you know why through FLAG%.

4.2 MODE 1: DECELERATING STOP

MODE 1 performs a decelerating stop or gradual cessation of driving pulses on a motor that is busy executing a motion command. The deceleration parameters correspond to those set in initialization MODE 12. This type of stop should lead to no loss of positional accuracy of the motor. The number of pulses which the motor would have continued to step if it had not received a decelerating stop command can be determined by a status read (MODE 8). The exact position of the motor can then be determined and a recovery routine initiated if required.

Entry Data:

MD% = 1	Decelerating stop command
D%(0)	Selects Axis A (0), Axis B (1), or Axis C (2)
D%(1) thru (14)	Value irrelevant
STP#	Value irrelevant
FLAG%	Value irrelevant

Exit Data:

D%(0) thru (14)	Unchanged
STP#	Unchanged
FLAG%	0 if executed OK 2 if driver not initialized on Axis A 3 if driver not initialized on Axis B 4 if driver not initialized on Axis C 5 if MD% ,0 or >15 6 if hardware error 8 if motor already at standstill (command redundant) 11 if D%(0) not 0, 1, or 2

A typical program entry preceding MODE 1 might read as follows:

```
10 MD% = 1                 'select MODE number
20 D%(0) = 0                      'select Axis A
30 CALL MSTEP (MD%, D%(0), STP#, FLAG%) 'decelerating stop
40 IF (FLAG%<>0 AND FLAG%<>7) THEN PRINT "Error";FLAG%;" in MODE 1":STOP
50                                'continue program
```

One detail concerning the preceding example: If FLAG% = 8 is returned, it is just a reminder that you told the controller to do a decelerating stop when in fact the motor was already stopped. In this case the driver aborts the command and lets you know why through FLAG%.

4.3 MODE 2: SINGLE STEP OR "JOG"

MODE 2 performs a single step or "jog" of the stepper motor useful in getting a system into final position or for manual control. The direction is set by the sign of the STP# data although the value does not matter. In this respect STP# = 0,1,99 or 16,123,678 would all produce a 1 step clockwise motion, whereas STP# = -1, -234, or -13,456,987 would all produce a 1-step counter-clockwise motion.

Entry Data:

MD% = 2	Single step (jog) command
D%(0)	Selects Axis A (0), Axis B (1), or Axis C (2)
D%(1) thru (14)	Value irrelevant
STP#	Sign sets direction, magnitude irrelevant
FLAG%	Value irrelevant

Exit Data:

D%(0) thru (14)	Unchanged
STP#	Unchanged
FLAG%	0 if executed OK 1 if motor already busy executing last command 2 if driver not initialized on Axis A 3 if driver not initialized on Axis B5 if MD% ,0 or >15 4 if driver not initialized on Axis C 6 if hardware error 11 if D%(0) not 0, 1, or 2

A typical program entry preceding MODE 2 might read as follows:

```
xxx10 MD% = 2                           'select MODE number
xxx20 D%(0) = 0                         'select Axis A
xxx30 CALL MSTEP (MD%, D%(0), STP#, FLAG%)
                                        'jog command
xxx40 IF FLAG%<>0 THEN PRINT "Error in MODE 2, # ";FLAG%:STOP
xxx50                                   'continue program
```

Here's a programming tip. This routine uses the cursor keys to jog clockwise or counterclockwise until another key is pressed:

```
100 MD% = 2                       'select MODE
110 D%(0) = 1                           'select Axis B
120 A$ = INKEY$:IF A$="" GOTO 120       'wait for keypress
130 IF LEN(A$)=1 THEN GOTO 210          'exit on any single key code
140 IF ASC(RIGHT$(A$,1))=75 THEN STP#=-1:GOTO 170
                                        'cursor left
150 IF ASC(RIGHT$(A$,1))=77 THEN STP#=+1:GOTO 170
                                        'cursor right
160 GOTO 210                            'exit on any other double key code
170 CALL MSTEP (MD%,D%(0),STP#,FLAG%)   'jog
180 IF FLAG%=1 THEN GOTO 170            'repeat command if still busy
190 IF FLAG%<>0 THEN ?"Error in jog command":STOP
200 GOTO 120                            'read keyboard again
210                                     'continue your program here
```

4.4 MODE 3: STEP WITH ACCELERATION/DECELERATION

MODE 3 is the basic all purpose trapezoidal motion command that causes the motor to move the number of steps specified by STP# with acceleration, deceleration, start up and high speed run rates set by the initialization parameters of MODE 15. The direction of rotation is controlled by the sign of STP#, positive is clockwise and negative is counter-clockwise. Up to 16,777,215 steps may be performed with one command corresponding to the 24 bit integer data limits of the PPMC-103A controllers.

MODE 3 corresponds to motion command 3 of the PPMC-103A with a few "user friendly" differences introduced by the driver. The PPMC will in fact move 1 step more than the number input, the driver corrects this characteristic by subtracting 1 from the step count before inputting it to the PPMC, so that STP# corresponds exactly with the number of steps moved. Also the PPMC behaves in strange ways with step counts of -1,0 and +1. Again the driver intercepts these singular values, does nothing with a STP#=0 (aborts command with no error) and automatically reverts to a jog command if STP# = +1 or -1 (see Appendix A for further details). This is all transparent to the user, so you can compute values of STP# without worrying about the controller's peculiarities. In fact, you can ignore MODE 2 and do jogs with STP# = +1 or -1 if this rationalization makes sense for you.

If the the total number of steps is less than twice the acceleration/deceleration step count, the velocity profile will be triangular instead of trapezoidal i.e. the motor will never reach its high speed run rate. The PPMC controller will look after this automatically.

Entry Data:

MD% = 3	Accelerate/decelerate command
D%(0)	Selects Axis A (0), Axis B (1), or Axis C (2)
D%(1)	0 = non-Turbo Mode, 1 = Turbo Mode
D%(2) thru (14)	Value irrelevant
STP#	Value and sign set number of steps and direction
FLAG%	Value irrelevant

Exit Data:

D%(0) thru (14)	Unchanged
STP#	Unchanged
FLAG%	0 if executed OK 1 if motor already busy executing last command 2 if driver not initialized on Axis A 3 if driver not initialized on Axis B 4 if driver not initialized on Axis C 5 if MD% ,0 or >15 6 if hardware error 7 if STP# <-16,777,215 or >+16,777,215 11 if D%(0) not 0, 1, or 2

A typical program entry preceding MODE 3 might read as follows:

```
xxx10 MD% = 3                           'select MODE number
xxx20 D%(0) = 1                         'select Axis B
xxx30 D%(1) = 1                         ;select non-Turbo Mode
xxx40 STP# = -5632                      'move counter-clockwise 5,632 steps
xxx40 CALL MSTEP (MD%, D%(0), STP#, FLAG%)
                                        'acel/decel command
xxx50 IF FLAG%<>0 THEN PRINT "Error in MODE 3, # ";FLAG%:STOP
xxx60                                   'continue program
```

Here's a programming tip. The time this command takes to complete depends on the stepping rate and number of steps, for a lot of steps it can take a long time. If the command is reissued while a previous motion is taking place the FLAG% will be returned = 1, indicating that you have to wait before sending the next motion command (unless it's a stop command). Let's say we would like the motor to move 1,237 steps clockwise, 67 steps counter-clockwise, 12,678 steps clockwise etc. This is how to program this trajectory:

```
100 MD% = 3                                 'select MODE
110 D%(0) = 1                               'select Axis B
120 D%(1) = 0                               'select non-Turbo acceleration
130 STP# = 1237                             '1st step count
140 CALL MSTEP (MD%, D%(0), STP#, FLAG%)
                                            '1st motion
                                            'keep trying if motor busy
150 IF FLAG%=1 THEN GOTO 140
160 IF FLAG%<>0 THEN "Error # ";FLAG%:STOP
                                            'major disaster!
                                            '2nd step count
170 STP# = -67
180 CALL MSTEP (MD%, D%(0), STP#, FLAG%)
                                            '2nd motion
                                            'keep trying until 1st motion
190 IF FLAG%=1 THEN GOTO 180                '1st motion finished

200 IF FLAG%<>0 THEN "Error # ";FLAG%:STOP  'major disaster!
                                            '3rd step count

210 STP# = 12678
220 CALL MSTEP (MD%, D%(0), STP#, FLAG%)    '3rd motion
                                            'keep trying until 2nd motion

230 IF FLAG%=1 THEN GOTO 220
                                            'finished
240 IF FLAG%<>0 THEN "Error # ";FLAG%:STOP  'major disaster!
250 'etc.
```

Obviously a good programmer would use a few GOSUBs to minimize the code above. Also note that the program is paced by the rate at which the motor will execute commands. Maybe you would like to go off and do something else if you find FLAG%=1, the option is yours. Also if you ever need to find whether the motor is busy without doing anything before inputting a motion command, use read status, MODE 8 - see MODE 8 for further details.

4.5 MODE 4: STEP AT CONSTANT SPEED

MODE 4 is similar to MODE 3 except that the motor will move the number of steps specified by STP# with constant velocity specified by D%(1). The stepping rate depends on the clock source. For the PPMC internal clock source (D%(7) = 0 in initializing MODE 15), Rate = 25,000 / D%(1) steps/second

For the external user input clock source (D%(7) = 2 in initializing MODE 15),

Rate = Ext. Freq. / D%(1) steps/second

For the on board 800KHz clock + divider (D%(7)=1 in initializing MODE 15),

Rate = 800,000 / (D%(1) * (X+1)) steps/second

where X = divider ratio set in MODE 10

The valid range for D%(1) is 20 (fastest) to 255 (slowest). The direction of rotation is set by the sign of STP#, positive is clockwise and negative is counter-clockwise. Up to 16,777,215 steps may be performed with one command corresponding to the 24 bit integer data limits of the PPMC-103A controllers.

Since the motor has to start up and run at a constant rate, this rate must be within the start up capabilities of the motor. Also, the rate set in this MODE by D%(1) **does not overwrite** the rate specified for acceleration/deceleration in initializing MODE 15 (it is specific to this command only).

If you are operating from the on-board 800KHz clock and on-board divider, it is possible to change the external divider ratio while the motor is in motion and in this way modulate the speed. This is a MODE of operation that is not obtainable with the PPMC controller alone, at least not with one continuous command.

MODE 4 corresponds to motion command 4 of the PPMC-103A with a few "user friendly" differences introduced by the driver. The PPMC will in fact move 1 step more than the number input, the driver corrects this characteristic by subtracting 1 from the step count before inputting it to the PPMC, so that STP# corresponds exactly with the number of steps moved. Also the PPMC behaves in strange ways with step counts of -1,0 and +1. Again the driver intercepts these singular values, does nothing with a STP#=0 (aborts command with no error) and automatically reverts to a jog command if STP# = +1 or -1 (see Appendix A for further details). This is all transparent to the user, so you can compute values of STP# without worrying about the controller's peculiarities. In fact, you can ignore MODE 2 and do jogs with STP# = +1 or -1 if this rationalization makes sense for you.

Entry Data:

MD% = 4	Move at constant speed command
D%(0)	Selects Axis A (0), Axis B (1), or Axis C (2)
D%(1)	Sets speed, valid range 20 - 255
D%(2) thru (14)	Value irrelevant
STP#	Value and sign set number of steps and direction
FLAG%	Value irrelevant

Exit Data:

D%(0) thru (14)	Unchanged
STP#	Unchanged
FLAG%	0 if executed OK 1 if motor already busy executing last command 2 if driver not initialized on Axis A 3 if driver not initialized on Axis B 4 if driver not initialized on Axis C 5 if MD% ,0 or >15 6 if hardware error 7 if STP# <-16,777,215 or >+16,777,215 10 if D%(0) not 0, 1, or 2 11 if D%(1) <20 or >255

A typical program entry preceding MODE 4 might read as follows:

```
xxx10 MD% = 4                                   'select MODE number
xxx20 D%(0) = 1                                 'select Axis B
xxx30 D%(1) = 180                               'speed, about 70pps with
                                                     internal clock
xxx40 STP# = 2000                               'move clockwise 2,000 steps
xxx50 CALL MSTEP (MD%, D%(0), STP#, FLAG%)
                                                'constant speed command
xxx60 IF FLAG%<>0 THEN PRINT "Error in MODE 4, # ";FLAG%:STOP
xxx70                                           'continue program
```

As with MODE 3, you can chain motion commands using FLAG%=1 to signal the readiness of the controller to receive the next command (see MODE 3 for programming example).

4.6 MODE 5: MOVE TO OUTER LIMIT AT CONSTANT SPEED

MODE 5 lets you run the motor into either of the outer overtravel limit switches L1 or L2. The motor will move in the direction specified by the sign of STP# with constant velocity specified by D%(1) until it encounters the appropriate limit switch input (L1 for clockwise, L2 for counter-clockwise). The stepping rate depends on the clock source. For the PPMC internal clock source (D%(7) = 0 in initializing MODE 15),

Rate = 25,500 / D%(1) steps/second

For the external user input clock source (D%(7) = 2 in initializing MODE 15),

Rate = Ext. Freq. / D%(1) steps/second

For the on board 800KHz clock + divider, (D%(7)=1 in initializing MODE 15),

Rate = 800,000 / (D%(1) * (X+1)) steps/second

where X = divider ratio set in MODE 10.

The valid range for D%(1) is 20 (fastest) to 255 (slowest). Since the motor has to start up and run at a constant rate, this rate must be within the start up capabilities of the motor. Also, the rate set in this MODE by D%(1) **does not overwrite** the rate specified for acceleration/deceleration in initializing MODE 15 (it is specific to this command only).

If you are operating from the on-board 800KHz clock and external divider, it is possible to change the external divider ratio while the motor is in motion and in this way modulate the speed.

Entry Data:

MD% = 5	Move at constant speed to outer limit L1 or L2.
D%(0)	Selects Axis A (0), Axis B (1), or Axis C (2)
D%(1)	Sets speed, valid range 20 - 255
D%(2) thru (14)	Value irrelevant
STP#	Sign sets direction, magnitude irrelevant
FLAG%	Value irrelevant

Exit Data:

D%(0) thru (14)	Unchanged
STP#	Unchanged
FLAG%	0 if executed OK, otherwise:- 1 if motor already busy executing last command 2 if driver not initialized on Axis A 3 if driver not initialized on Axis B 4 if driver not initialized on Axis C 5 if MD% ,0 or >15 6 if hardware error 7 if STP# <-16,777,215 or >+16,777,215 10 if D%(0) not 0, 1, or 2 11 if D%(1) <20 or >255

A typical program entry preceding MODE 5 might read as follows:

```
xxx10 MD% = 5                                   'select MODE number
xxx20 D%(0) = 0                                 'select Axis A
xxx30 D%(1) = 250                               'speed, about 50pps with
                                                      internal clock
xxx40 STP# = -99                                'move counter-clockwise to L2
xxx50 CALL MSTEP (MD%, D%(0), STP#, FLAG%)
                                                'move to limit L1 or L2
```

```
xxx60 IF FLAG%<>0 THEN PRINT "Error in MODE 5, # ";FLAG%:STOP
xxx70                                   'continue program
```

After you have reached the limit, you can confirm that you are there by performing a MODE 8, read status command. In general, the constant speed should be selected slow enough so that there is no danger of overshooting the limit. It is then possible to use either L1 or L2 as homing reference points as an alternative to the BP (base point) or CNP input.

4.7 MODE 6: MOVE TO LIMIT AT HIGH SPEED

MODE 6 lets you run the motor with controlled acceleration and deceleration to and slightly beyond either of the inner high speed limit switches L3 or L4. The motor will move in the direction specified by the sign of STP# with a trapezoidal velocity profile specified by the initialization parameters of MODE 12. On hitting the appropriate inner or high speed limit switch (L3 clockwise or L4 counter-clockwise) a controlled deceleration will take place, and hopefully (if you have your system set up right!) will stop before reaching an outer limit switch L1 or L2. At this point you can inch it into an outer limit using MODE 5 to perform a calibration, or do whatever else is appropriate in the circumstances. This provides a fast way of getting to the end of travel if it is something you need to do frequently.

Entry Data:

MD% = 6	Move at constant speed to inner limit L3 or L4.
D%(0)	Selects Axis A (0), Axis B (1), or Axis C (2)
D%(1) thru (14)	Value irrelevant
STP#	Sign sets direction, magnitude irrelevant
FLAG%	Value irrelevant

Exit Data:

D%(0) thru (14)	Unchanged
STP#	Unchanged
FLAG%	0 if executed OK 1 if motor already busy executing last command 2 if driver not initialized on Axis A 3 if driver not initialized on Axis B 4 if driver not initialized on Axis C 5 if MD% ,0 or >15 6 if hardware error 7 if STP# <-16,777,215 or >+16,777,215 10 if D%(0) not 0, 1, or 2

A typical program entry preceding MODE 6 might read as follows:

```
xxx10 MD% = 6                                       'select MODE number
xxx20 D%(0) = 0                                     'select Axis A
xxx30 STP# = -8                                     'move counter-clockwise to L4
xxx40 CALL MSTEP (MD%, D%(0), STP#, FLAG%)
                                                    'move to limit L3 or L4
xxx50 IF FLAG%<>0 THEN PRINT "Error in MODE 6, # ";FLAG%:STOP
xxx60                                               'continue program
```

4.8 MODE 7: MOVE TO BASE POINT AT CONSTANT SPEED

MODE 7 lets you run the motor into the BASE point limit switch (this is called CNP on the PPMC data sheet) at constant speed set by D%(1) and direction set by the sign of STP#. The magnitude of STP# is irrelevant.

Not all systems will use a base point reference switch, often one of the outer limits L1 or L2 will do just as well to calibrate the motor position. If however you do provide a base point reference, say in the middle of travel, how do you know which way to go to find it? The answer is that you don't, if you have gone the wrong way you may just as likely stop at one of the outer limit switches. The PPMC is not intelligent enough to reverse direction on hitting an end limit, so you had better be prepared to provide some software to do the hunting! A status read, MODE 8, after MODE 7 will tell you what limit you hit, BP or an outer limit, L1 or L2. If you hit an outer limit, you can engage MODE 7 again in the reverse direction and you are bound to find the BP this time - see MODE 8 for details.

The stepping rate in MODE 7 depends on the clock source. For the PPMC internal clock source (D%(7) = 0 in initializing MODE 15),

Rate = 25,000 / D%(1) steps/second

For the external user input clock source (D%(7) = 2 in initializing MODE 15),

Rate = Ext. Freq. / D%(1) steps/second

For the on board 800KHz clock + divider, (D%(7)=1 in initializing MODE 15),

Rate = 800,000 / (D%(1) * (X+1)) steps/second

where X = divider ratio set in MODE 9

The valid range for D%(1) is 20 (fastest) to 255 (slowest). Since the motor has to start up and run at a constant rate, this rate must be within the start up capabilities of the motor. Also, the rate set in this MODE by D%(1) **does not overwrite** the rate specified for acceleration/deceleration in initializing MODE 12 i.e. it is specific to this command only.

If you are operating from the on-board 100KHz clock and external divider, it is possible to change the external divider ratio while the motor is in motion and in this way modulate the speed.

Entry Data:

MD% = 7	Move at constant speed to base point BP or CNP
D%(0)	Selects Axis A (0), Axis B (1), or Axis C (2)
D%(1)	Sets speed, valid range 20 - 255
D%(2) thru (14)	Value irrelevant
STP#	Sign sets direction, magnitude irrelevant
FLAG%	Value irrelevant

Exit Data:

D%(0) thru (14)	Unchanged
STP#	Unchanged
FLAG%	0 if executed OK 1 if motor already busy executing last command 2 if driver not initialized on Axis A 3 if driver not initialized on Axis B 4 if driver not initialized on Axis C 5 if MD% ,0 or >15 6 if hardware error 7 if STP# <-16,777,215 or >+16,777,215 10 if D%(0) not 0, 1, or 2 11 if D%(1) <20 or >255

A typical program entry preceding MODE 7 might read as follows:

```
xxx10 MD% = 7                                     'select MODE number
xxx20 D%(0) = 0                                   'select Axis A
xxx30 D%(1) = 200                                 'speed, about 62pps with
                                                  'internal clock
xxx40 STP# = 10                                   'move clockwise to BP
xxx50 CALL MSTEP (MD%, D%(0), STP#, FLAG%)        'move to BP
xxx60 IF FLAG%<>0 THEN PRINT "Error in MODE 7, # ";FLAG%:STOP
xxx70                                             'continue program
```

4.9 MODE 8: READ MOTOR STATUS

MODE 8 provides useful information about what the motor is doing or has done. The PPMC-103A controller can only provide full status information if the motor is at rest. If the motor is busy, FLAG% will be returned = 1 and the status data variables D%(2) thru D%(4) will be returned unchanged. This at least tells you that the motor is busy and unable to respond to a motion command other than an emergency or decelerating stop. At this point you can decide whether to let the motor complete whatever its doing and continue looping status reads until FLAG%=0 indicating its finished, or you can intervene and abort whatever it's doing with a stop command.

If FLAG% is returned = 0 then D%(2) thru D%(4) and STP# will contain status data as follows:

D%(2) contains the **FINISH STATUS** consisting of a byte of data:

```
B7   B6   B5   B4   B3   B2   B1   B0
|    |    |    |    |    ____|____
|    |    |    |    |        |
|    |    |    |    |     Last command code 0-7
|    |    |    |    |
|    |    |    |    Decelerating stop on L3 or L4
|    |    |    |
|    |    |    Stop on limit L1 or L2
|    |    |
|    |    Motor on (MC) check flag
|    |
|    Stopped by a motor stop command
|
Finish flag or interrupt flag
```

Bits B0 thru B2 provide the code or MODE number of the last command. Bits B3 thru B7 tell you what happened as follows:

B3	goes to logic "1" after L3 or L4 limit switch inputs have forced a stop. Otherwise it is "0".
B4	goes to logic "1" after L1 or L2 limit switch inputs have forced a stop. Otherwise it is "0".
B5	Is the motor ON signal (inverse of MC input). If the motor loses power, B5 is "1" otherwise it is "0" assuming MC is connected to suitable power sensing circuitry. If MC is left open circuit, then B5 will always be "0". The PPMC-103A will not execute any commands if MC is at logic "0".
B6	Is logic "1" if a motion command was aborted by an emergency stop or decelerating stop command, otherwise it is "0".
B7	Corresponds to the interrupt flag. Whether interrupts are enabled or not, whether a command was aborted or not, B7 is set to "1" at the completion of the command. It is cleared by a second read of the finish status. In multiple PPMC-103A systems sharing a common interrupt, the finish flags of each controller can be polled to see which one generated the interrupt and the interrupt service routine can clear the flag ready for the next command after servicing the interrupt. This is the arrangement on the MSTEP-3 - see the MSTEP.ASM listing & MODE 11.

If all the bits B3-B6 are zero, it indicates that the last command executed to completion and the step count input was actually moved. You can implement various recovery routines on conditions that might arise e.g.:

```
xxx10 IF (D%(2) AND &H08)=&H08 THEN GOSUB aaaa
                             'recover from stop at inner h.s.  limit
xxx20 IF (D%(2) AND &H10)=&H10 THEN GOSUB bbbb
                             'recover from stop at outer limit
xxx30 IF (D%(2) AND &H20)=&H20 THEN GOSUB cccc
                             'notify user there is no motor power
xxx40 IF (D%(2) AND &H40)=&H40 THEN GOSUB dddd
                             'recover from an emergency or decelerating stop
xxx50                        'continue program .  .  .
```

Obviously this can get quite complicated, but at least all the information you need to make a recovery including the remaining step count (STP#) is returned by the MODE 8 status read.

D%(3) contains the **INPUT STATUS** consisting of a byte of data:

```
 B7   B6   B5   B4   B3   B2   B1   B0
 |    |    ----+----  |    |    |    |
 |    |    Not Used   |    |    |   At L4
 |    |               |    |   At L3
 |    |               |   At L2
 |    |              At L1
 |   Motor on (MC)
At base point, BP or CNP
```

All these bits, with the exception of B4 & B5 which are indeterminate, will be high unless the motor is at the limit switch position. Note that this is an instantaneous read of the limit switch states, if you have overshot a limit, it will not store that information, the input status simply tells you the current state of the limit switch inputs.

The main use of the input status is in determining whether the motor is at a limit, what limit it is and hence performing recoveries or re-calibrations (homing to a reference) through appropriate software routines. The individual limit switch inputs can be separated out with ANDing operations as in the previous example.

D%(4) contains the **OUTPUT STATUS** consisting of a byte of data:

```
B7   B6   B5   B4   B3   B2   B1   B0
|    |    |    |    |    |    |    |
S1   S2   S3   S4   S5   |    |    Not used
                         |    Direction
                         Hold signal
```

The output status provides information on the PPMC-103A output state. Bits B7 thru B3 simply reflect the current state of the phase drive outputs S1-S5. B2 is the "HOLD" signal corresponding to HCK on the connector. This goes high 3 milliseconds after the motor comes to a standstill but @B[only if] you have selected switching at standstill on in initialization MODE 12 (D%(8)=1 in initialization). The intent of the external hold output is most likely to let you turn on some sort of electromechanical holding brake as the holding torque with switching on is fairly low.

Bit B1 reflects the last rotation direction, "0" = clockwise, "1"= counterclockwise. Bit B0 is not used and is indeterminate.

STP# contains the remaining step count after a command has aborted. If the command completed STP# will be zero. The sign of STP# is always positive regardless of whether you were going clockwise or counter-clockwise at the time i.e. it is a true remaining step count.

Between FLAG%, D%(2) - D%(4) and STP#, the MODE 8 read status command provides a wealth of information. Note that the byte formats of D%(2)- D%(4) are identical to those described in the PPMC-103A data sheet (Appendix A) under finish status, input status and output status respectively. You may find the data sheet useful in further understanding the various status conditions. MODE 8 actually does 4 consecutive read register operations and returns every conceivable status condition in one operation, this is simply a convenience, you can use what you want and ignore the rest.

Entry Data:

MD% = 8	Read status
D%(0)	Selects Axis A (0), Axis B (1), or Axis C (2)
D%(1) thru (14)	Value irrelevant
STP#	Value irrelevant
FLAG%	Value irrelevant

Exit Data:

D%(0) thru (1)	Unchanged
D%(2)	Finish status
D%(3)	Input status
D%(4)	Output status
D%(5) thru (14)	Unchanged
STP#	Remaining pulses (0 - 16,777,215)
FLAG%	0 if executed OK 1 if motor busy 2 if driver not initialized on Axis A 3 if driver not initialized on Axis B 4 if driver not initialized on Axis C 5 if MD% ,0 or >15 6 if hardware error 10if D%(0) not 0, 1, or 2

Note that if FLAG%=1 then D%(2) thru D%(4) and STP# will be returned unchanged as the PPMC-103A is unable to execute a status command unless the motor is at standstill.

A typical program entry preceding MODE 8 might read as follows:

```
xxx10 MD% = 8                                 'select MODE number
xxx20 D%(0) = 0                               'select Axis A
xxx30 CALL MSTEP (MD%, D%(0), STP#, FLAG%)
                                              'read status
xxx40 IF FLAG%=1 THEN GOTO
xxx30                                         'keep trying until motor stops
xxx50 IF FLAG%<>0 THEN PRINT "Error in MODE 8, # ";FLAG%:STOP
xxx60                                         'continue program, process
                                                    D%(2)-D%(4), STP#
                                                    etc.
```

NOTE: The loop in line xxx40 may not always be action you might want on finding the motor busy. Change it to suit.

4.10 MODE 9: LOAD EXTERNAL CLOCK DIVIDER

MODE 9 sets the division ratio of the external stepper clock divider. It applies only if you have selected operation from the external 800 KHz on-board clock, D%(7)=1 in initializing MODE 12, or user-supplied clock. Note that the frequency of the selected clock divided by the specified divider ratio must be less than 267KHz for the PPMC-103A to work properly.

Entry Data:

MD% = 9	Load external divider
D%(0)	Selects Axis A (0), Axis B (1), or Axis C (2)
D%(1)	Sets divider ratio 2-255
D%(2) thru (9)	Value irrelevant
STP#	Value irrelevant
FLAG%	Value irrelevant

Exit Data:

D%(0) thru (14)	Unchanged
STP#	Unchanged
FLAG%	0 if executed OK 2 if driver not initialized on Axis A 3 if driver not initialized on Axis B 4 if driver not initialized on Axis C 5 if MD% <0 or >14 10 if D%(0) not 0 or 1 11 if D%(1) <1 or >255

A typical program entry preceding MODE 9 might read as follows:

```
xxx10 MD% = 10                                'select MODE number
xxx20 D%(0) = 0                               'select Axis A
xxx30 D%(1) = 100                             'divider ratio = 100
xxx40 CALL MSTEP (MD%, D%(0),STP#, FLAG%)
                                              'load divider
xxx50 IF FLAG%<>0 THEN PRINT "Error in MODE 10, # ";FLAG%:STOP
xxx60                                         'continue program .  .  .
```

4.11 MODE 10: READ DATA FROM ALL PORTS

Mode 10 reads all ports and returns data regardless of whether the ports have been configured as inputs or outputs.

Entry Data:

MD% = 10	Read data from all ports
D%(0)	Selects Port A (0) or B (1)

D%(1) thru (14)	Value Irrelevant
STP#	Value Irrelevant
EFLAG%	Value Irrelevant

Exit Data:

D%(0)	Unchanged
D%(1)	Port A data (range 0-255)
D%(2)	Port B data (range 0-255)
D%(1) thru (14)	Unchanged
STP#	Unchanged
EFLAG%	0 = Execute OK

A typical program entry preceeding Mode 10 might read as follows:

```
xx10 MD% = 10                'Select Mode number
xx20 D%(0) = 0               'Select Port A
xx30 CALL MSTEP (MD%, D%(0), STP#, FLAG%)
xx40 IF FLAG% <> 0 THEN PRINT "ERROR IN MODE 10, #";FLAG%:STOP
xx50                         'Continue program
```

4.12 MODE 11: WRITE DATA TO ALL PORTS

Mode 11 writes data to all ports. No data is written to any port unless all output data is within range, as detailed under Mode 10. Data may be written to ports that are configured as inputs; this will have no effect on the input data.

Entry Data:

MD% = 11	Write data to all ports
D%(0)	Selects Port A (0) or B (1)
D%(1)	Port A data (range 0-255)
D%(2)	Port B data (range 0-255)
D%(3) thru (14)	Value Irrelevant
STP#	Value Irrelevant
EFLAG%	Value Irrelevant

Exit Data:

D%(0) thru (14)	Unchanged
STP#	Unchanged

EFLAG%	0 = Execute OK 11 = D1 <0 or >255 12 = D2 <0 or >255

A typical program entry preceeding Mode 11 might read as follows:

```
xx10 MD% = 11                'Select Mode number
xx20 D%(0) = 0               'Select Port A
xx30D%(1) = 255              'Write 255 to Port A
```

4.13 MODE 12: WRITE DATA TO ONE PORT

Mode 12 writes data to a single port rather than to all ports, as Mode 11 does. D0 selects the axis, D1 selects the port, and D2 provides the data. Note that data should be in the range 0-255 for the PA and PB ports. If a port is in the input mode, the data is still written but will have no effect. If port number or data is out of range, the Mode 12 operation is abandoned, no data is written, and an error code is returned.

Entry Data:

MD% = 12	Write data to one port
D%(0)	Selects Axis A (0), B (1), or C (2)
D%(1)	Selects Port A (0) or B (1)
D%(2)	Data (range 0-255)
D%(3) thru (14)	Value Irrelevant
STP#	Value Irrelevant
EFLAG%	Value Irrelevant

Exit Data:

D%(0) thru (14)	Unchanged
STP#	Unchanged
EFLAG%	0 = Execute OK 11 = D1 not 0 or 1 12 = D2 <0 or >255

A typical program entry preceeding Mode 12 might read as follows:

```
xx10 MD% = 12                'Select mode number
xx20 D%(0) = 0               'Select Axis A
xx30 D%(1) = 0               'Select Port A
xx40 D%(2) = 255             'Write 255 to Port A
xx50 CALL MSTEP (MD%, D%(0), STP#, FLAG%)
xx60 IF FLAG%  <> 0 THEN PRINT "ERROR IN MODE 12, #";FLAG%:STOP
xx70                         'Continue program
```

4.14 MODE 13: TOGGLE AUXILIARY BIT

Mode 13 toggles the auxiliary bit of the PPMC-103A dependent on the selected axis.

Entry Data:

MD% = 13	Toggle auxiliary bit
D%(0)	Selects Axis A (0), B (1), or C (2)
D%(1) thru (14)	Value Irrelevant
STP#	Value Irrelevant
EFLAG%	Value Irrelevant

Exit Data:

D%(0) thru (14)	Unchanged
STP#	Unchanged
EFLAG%	0 = Execute OK 11 = D0 not 0 or 1 12 = D2 <0 or >255

A typical program entry preceeding Mode 13 might read as follows:

```
xx10 MD% = 13                'Select mode number
xx20 D%(0) = 0               'Select Axis A
xx30 CALL MSTEP (MD%, D%(0), STP#, FLAG%)
xx40 IF FLAG <> 0 THEN PRINT "ERROR IN MODE 13, #";FLAG:STOP
xx50                         'Continue program
```

4.15 MODE 14: ENABLE/DISABLE AN INTERRUPT

Either or both PPMC stepper motor controllers are capable of generating an interrupt to signify completion of certain motion control commands. BASIC cannot be used to program interrupt service routines as there is no ON INTERRUPT construct. If you wish to make use of the MSTEP-3 interrupt capabilities, you will have to resort to some assembly language programming and modification of the driver. MODE 14 does a lot of the groundwork for you in installing an example routine and enabling/disabling it. While it is impossible to predict your requirements in terms of what you might want the interrupt service routine to do, as an example, this MODE installs a sample interrupt service routine (label INTH: in the MSTEP.ASM source) which simply sounds a beep on generation of an interrupt. The MSTEP.ASM source can be modified to change the INTH: service routine to your needs and re-assembled.

MODE 14 performs the following:

1. Disables interrupts on the selected level (2-7).
2. Installs interrupt vectors to INTH: for that level.
3. Either enables or disables hardware interrupts on the selected level according to D%(2).

WARNING It is your responsibility to avoid conflict with other peripherals when using interrupts - see below.

It is important to note that MODE 14 does not save and restore vectors to any previous interrupt service routine that may have been serviced on the selected level. If you need this capability for sharing devices on the same level, you need to expand and modify the driver code. The assumption has been made that if you are going to use interrupts at all, you will dedicate one of the hardware interrupt levels 2 thru 7 to the MSTEP-3. If interrupts are disabled (default power up & initializing conditions) then the MSTEP-3 interrupt output is tri-stated and will not interfere with any device on the level. The operating level is selected by the jumper block labelled "IRQ LEVEL" on the MSTEP-3 board.

If interrupts are enabled while the selected motor is busy, they will not take effect until execution of the following command. Only certain motion control commands are able to generate an interrupt on completion. These are:

MODE 2: Single step or jog.
MODE 3: Accelerate/decelerate forSTP# pulses.
MODE 4: Constant speed forSTP# pulses.
MODE 5: Constant speed to outer limit switch.
MODE 6: High speed to inner limit switch.
MODE 7: Move to base or reference point.

Because the interrupts from Axes A, B, and C are OR'd together by the MSTEP-3 hardware, if all are enabled you will need to poll the status of both controllers to determine which one generated the interrupt. This requires a read finish status register command which in turn will clear the interrupt signal from the requesting controller. This may be part of your Interrupt Service Routine (as in the example handler) or provided by your foreground program by a call to MODE 8 as dictated by your needs.

The following list provides the normal IBM reserved functions for Hardware Interrupts 2 thru 7.

LEVEL	FUNCTION
2	Usually free PC & PC/XT, cascade input on PC/AT
3	COM2: (if installed)
4	COM1: (if installed)
5	LPT2: (if installed)
6	Floppy disk (always used)
7	LPT1: (if installed)

If you do not have the particular hardware item installed, it is safe to use that level. Generally, Levels 2 or 5 work out well on PCs and PC/XTs, and Level 5 on PC/ATs. Never use Level 6 as you will mess up the floppy disk operation. Levels 3, 4, and 7 may be available depending on what equipment you have installed and whether you are using it you can often share with a peripheral that your program will not be using as long as the peripheral's interrupt hardware is disabled.

Entry Data:

MD% = 14	Enables/disables interrupt
D%(0)	Selects Axis A (0), Axis B (1), or Axis C (2)

D%(1)	Value irrelevant
D%(2)	Enables (1) or disables (0) interrupt
D%(3)	Interrupt level (2 - 7); if D%(2) = 0 (disable), value of D%(3) is irrelevant.
D%(4) thru (9)	Value irrelevant
STP#	Value irrelevant
FLAG%	Value irrelevant

Exit Data:

D%(0) thru (14)	Unchanged
STP#	Unchanged
FLAG%	0 if executed OK 2 if driver not initialized on Axis A 3 if driver not initialized on Axis B 4 if driver not initialized on Axis C 5 if MD% <0 or >15 10 if D%(0) not 0, 1, or 2 12 if D%(2) not 0 or 1 13 if D%(2)=1 and D%(3) <2 or >7

A typical program entry preceding MODE 11 might read as follows:

```
xxx10 MD% = 11                                'select MODE number
xxx20 D%(0) = 0                               'select Axis A
xxx30 D%(2) = 1                               'enable interrupt
xxx40 D%(3) = 5                               'on level 5
xxx50 CALL MSTEP (MD%, D%(0),STP#, FLAG%)     'enable interrupt
xxx60 IF FLAG%<>0 THEN PRINT "Error in MODE 11, # ";FLAG%:STOP
xxx70                                         'continue program .   .   .
yyy10   .   .   .  and if you want to disable interrupts at any point
yyy20 MD% = 11                                'select MODE number
yyy30 D%(0) = 0                               'select Axis A
yyy40 D%(2) = 0                               'disable interrupt
yyy50 CALL MSTEP (MD%, D%(0),STP#, FLAG%)     'disable interrupt
yyy60 IF FLAG%<>0 THEN PRINT "Error in MODE 11, # ";FLAG%:STOP
yyy70                                         'continue program .   .   .
```

4.16 MODE 15: INITIALIZATION

MODE 15 initializes the driver and MSTEP-3 hardware **and must be executed before using any of the other MODEs**. Initializing must be done separately for each Axis (A, B, and C). This mode performs the following operations:

1. Disables hardware interrupts.
2. Checks and stores the MSTEP-3's Base I/O Address.
3. Checks for presence of MSTEP-3 board at this address.
4. Checks the validity (range) of all initialization data.
5. Resets the PPMC-103A controller.
6. Loads the controller with initialization data.

7. Sets initialization flags to enable selection of other MODEs.

The Base I/O Address is checked to be in the legal range of 256-1016 (Hex 100 - 3F8) for the PC. If not, Error Exit #19 will occur. If OK, the Base I/O Address is stored for use by other MODEs on re-entry to the CALL.

A short read/write test is made to the MSTEP-3 control register which is sufficient to detect the presence or absence of the board at the specified I/O address. If no board is detected (absent board, wrong base I/O address), Error #6, hardware error is returned. Error #6 may also be returned by this MODE or any other MODE if the PPMC-103A controller fails to perform correct handshakes on inputs and outputs of commands and data (see Appendix A). In this case, Error #6 may be indicative of a failure of the PPMC-103A controller especially if one axis works and the other does not.

All the initialization data is checked to be within valid limits. Any of the initializing data variables D%(N) that is outside acceptable limits will return error code 10+N (this is true for all MODEs). Also in the interests of consistency D%(0) always specifies the axis selected (0 = A, 1 = B, 2 = C) and D%(1) the constant speed rate in all MODEs. Initialization MODE 15 makes use of all 10 elements of D%(*) and is the only MODE to require so much input data.

If initialization is successful, any other MODE may be selected on subsequent CALL's. **Trying to select any other MODE before performing initializing MODE 0 will give rise to Error # 2, 3, or 4 depending on whether Axis A, B, or C is uninitialized.**

Entry Data:

MD% = 15	(Value Irrelevant)
D%(0)	Axis selector: A = 0, B = 1, C = 2
D%(1)	Start up rate divider, RA, range 20-255
D%(2)	High speed run rate divider, range 20-255
D%(3)	Acceleration/deceleration pulses, 4-11,220
D%(4)	Motor type, valid codes 1,2 or 3 (see below)
D%(5)	Full step (0) or half step (1) MODE (see below)
D%(6)	Logic polarity of S1-5, 1 = positive, 0 = inverted
D%(7)	Clock source, 0 = internal, 1 = on board external, 2 = external user input
D%(8)	Power switching at standstill, 0 = off, 1 = on
D%(9)	"On" time for power-switching frequency
D%(10)	"Off" time for power-switching frequency
D%(11)	Port A direction: 1 = output, 0 = input
D%(12)	Port B direction: 1 = output, 0 = input
D%(13)	Auxiliary bit: 0 = low, 1 = high
D%(14)	Base I/O address, valid range 100-3F8 hex
STP#	Value irrelevant
FLAG%	Value irrelevant

Exit Data:

D%(0) thru (14)	Unchanged
STP#	Unchanged
FLAG%	0 if initialization OK 5 if MD% <0 or >15 6 if hardware error, no board, incorrect I/O address 10+N if D%(N) not in valid range

A typical start of program initializing sequence would be as follows:

```
100 DEF SEG = &H6000                          'segment to load driver
110 BLOAD "MSTEP.BIN"                         'load it at zero offset
120 MSTEP = 0                                 'call offset
130 DIM D%(14)                                'declare data array
140                                           'declare other CALL variables
150 MD% = 15                                  'mode number
160 FLAG% = 0                                 'call error flag variable
170 STP# = 0                                  'step count (must be double-
                                              'precision)
180 '
190 D%(0) = 0                                 '0 = Axis A
200 D%(1) = 125                               'Start Rate Divider (125 =
                                              'slowest)
210 D%(2) = 20                                'high-speed run rate divider
                                              ' (20 = medium)
220 D%(3) = 200                               'acceleration/deceleration
                                              'steps
230 D%(4) = 2                                 'motor code (2 = 4-phase)
240 D%(5) = 0                                 ''excitation (0 = full step)
250 D%(6) = 1                                 'logic polarity (1 =inverted)
260 D%(7) = 0                                 'clock source (0 = internal)
270 D%(8) = 1                                 'switching at standstill (1 =
                                              'On)
280 D%(9) = 16                                'On time for power switch
                                              'frequency (16 = time on)
290 D%(10) = 32                               'Off time for power switch
                                              'frequency (32 = time off)
300 D%(11) = 1                                'Port A direction (1 = output)
310 D%(12) = 1                                'Port B direction (1 = output)
320 D%(13) = 0                                'Auxiliary Bit (0 = low)
330 D%(14) = &H300                            'Base Address = &H300
340                                           'initialize
350 CALL MSTEP (MD%, D%(0),STP#, FLAG%)
360 IF FLAG%<>0 THEN PRINT "Error in initializing # ";FLAG%:STOP
370                                           'continue program
```

Much data must be provided at initialization, but it is required only once! Let's examine the initializing parameters in more detail:

D%(0) Determines which axis is selected. If D%(0) = 0, the CALL operates on Axis A motor and encoder; if D%(0) = 1 then the CALL operates on Axis B motor and encoder. Note that each each axis must be initialized individually and may have different initializing parameters, motor types etc.; and if D%(0) = 2, then CALL operates on C axis.

3-PHASE MOTOR

FULL STEP

MD% = 12, D%(4) = 1,
D%(5) = 0

	01	02	03
1	▨	▨	
2		▨	▨
3	▨		▨

HALF STEP

MD% = 12, D%(4) = 1,
D%(5) = 1

	01	02	03
1	▨		
2	▨	▨	
3		▨	
4		▨	▨
5			▨
6	▨		▨

4-PHASE MOTOR

FULL STEP

MD% = 12, D%(4) = 2,
D%(5) = 0

	01	02	03	04
1	▨	▨		
2		▨	▨	
3			▨	▨
4	▨			▨

HALF STEP

MD% = 12, D%(4) = 2,
D%(5) =1

	01	02	03	04
1	▨			
2	▨	▨		
3		▨		
4		▨	▨	
5			▨	
6			▨	▨
7				▨
8	▨			▨

5-PHASE MOTOR

FULL STEP

MD% = 12, D%(4) = 3,
D%(5) = 0

	01	02	03	04	05
1	▨	▨			
2		▨	▨		
3			▨	▨	
4				▨	▨
5	▨				▨

▨ DENOTES WINDING ENERGIZED

HALF STEP

MD% = 12, D%(4) = 3,
D%(5) = 1

	01	02	03	04	05
1	▨	▨			
2	▨	▨	▨		
3		▨	▨		
4		▨	▨	▨	
5			▨	▨	
6			▨	▨	▨
7				▨	▨
8	▨			▨	▨
9	▨				▨
10	▨	▨			▨

Figure 4-1. Stepping sequence patterns for 3-, 4-, and 5-phase full/half step motors. Note: when using STA-STEP, set D%(6) = 0.

D%(1) Controls the start up rate of the motor in MODEs 3 and 6 which involve acceleration and deceleration. (D%(1) corresponds to RA in the PPMC data sheet). The start up rate with internal clock (D%(7)=0):

Rate = 25,000 / D%(1) steps per sec.

With external user clock input (D%(7)=2):

Rate = Ext. freq. / D%(1) steps per sec.

With onboard 800KHz external clock (D%(7)=1):
Rate = 800,000 / (D%(1)*(X+1)) steps per sec.
where X = external divider ratio set in MODE 10.

The start up rate should be chosen slow enough so that the motor does not skip pulses on getting moving. If in doubt, set D%(1) = 255 (slowest).

D%(2) Controls the high speed run rate of the motor in MODEs 3 and 6 which involve acceleration and deceleration. For internal clock (D%(7)=0):
Rate = 25,000 / D%(2) steps per sec.

With external user input clock (D%(7)=2):
Rate = Ext. freq. / D%(2) steps per sec.

With onboard 800KHz external clock (D%(7)=1):
Rate = 800,000 / (D%(2)*(X+1)) steps per sec.

where X = external divider ratio set in MODE 10 The high speed run rate should be chosen so that the motor does not skip pulses at high speed. This usually requires some judicious experimentation. If you start off with D%(2)=150 and then reduce it until the motor shows signs of distress and then back off a a little, it will generally work out OK.

D%(3) Controls the number of steps that the motor will accelerate from the start rate to the high speed run rate and vice versa decelerate. The trapezoidal velocity profile and the controlling parameters are shown below in Fig.3.4. The acceleration/deceleration step count D%(3) may be anywhere from 4 to 11,220.

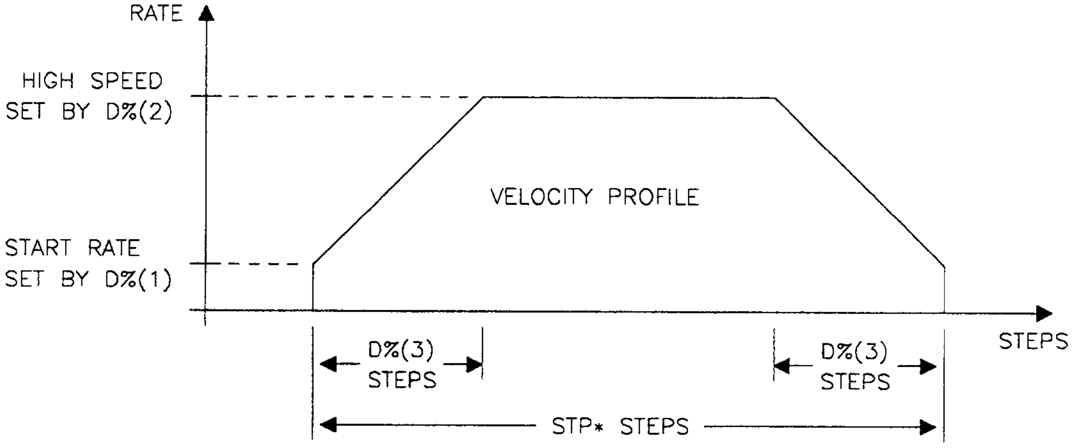


Figure 4-2. Velocity profile controlling parameters.

D%(4) Sets the motor code (1 - 3) depending on the winding arrangement. 3-phase motors are Code 1, 4-phase motors are Code 2, and 5-phase motors are Code 3. Another way to determine the motor code is to determine the required switching sequence as shown in Figure 4-1. Note that 3-phase motors use only phase outputs S1-S3, 4-phase use S1-S4 and 5-phase use all S1-S5. The motors additionally may be operated in full-step or half-step as set by D%(5). If you are not using the PPMC-103A phase outputs (e.g. in conjunction with an STA-STEP) then the motor code is irrelevant. D%(4) corresponds to the motor code parameter on the PPMC-103A data sheet.

D%(5) Selects the stepping MODE, full-step (0) or half-step (1). It corresponds to the excitation parameter of the PPMC-103A data sheet (see Appendix A). A standard 200 step/revolution motor (e.g. STEP-MOT1) will step 1.8 degree increments in full-step MODE or 0.9 degree increments (400 steps/revolution) in half-step. The half-step MODE offers finer resolution and smoother stepping but at the cost of reduced torque and maximum operating speed compared to full-step.

D%(6) Controls the logic polarity of the S1-S5 phase drive outputs. D%(6)=1 selects positive true logic and D%(6)=0 selects active low negative logic. The choice here depends on the driving hardware. For the STA-STEP set D%(6)=0. If you are not using the phase outputs S1-S5, the choice is irrelevant.

D%(7) Selects the clock source: D%(7) = 0 : Selects internal PPMC 12.5KHz source = 1 : Selects on board 100KHz + divider = 2 : Selects external user input For most purposes the internal clock source will be adequate. If you need step rates above 625 pps or synchronized stepping of both axes, use one of the external options.

D%(8) Controls switching at standstill, D%(8) = 0 turns switching off, D%(8) = 1 turns it on. At standstill the motor will normally have maximum current flowing in the windings which causes the greatest ohmic heating and temperature rise of the motor. One way of reducing this heating is to chop the phase drive outputs when the motor has reached standstill. With switching on, chopping commences 100 milliseconds after the motor reaches standstill at a frequency of about 2.2KHz and a duty cycle of 30%. As soon as the motor is commanded to move, chopping ceases and will automatically resume on standstill. Chopping tends to reduce the holding torque, raise the driver transistor switching losses, reduce heating of the motor and may produce audio noise from the motor. There are thus advantages and disadvantages associated with this feature.

D%(9) Set the ON time for the switching frequency of the chopping of the phase outputs, S1 - S5, when the motor is at standstill. The Excitation Signal Switching Bit must be set during Initialization to activate the phase chopping.

D%(10) Set the OFF time for the switching frequency of the chopping of the phase outputs, S1 - S5, when the motor is at standstill. The Excitation Signal Switching bit must be set during Initialization to activate phase chopping.

D%(11) Set the I/O direction for Port A. For D%(11) = 0, Port A is input; for 1, Port A is output.

D%(12) Set the I/O direction for Port B. For D%(12) = 0, Port B is input; for 1, Port B is output.

D%(13) Select the logic level on Pin 36 of PPMC-103A as default. For D%(13) = 0, logic is low; for 1, logic is high.

D%(14) Selects the base I/O address which must correspond to the dipswitch setting on the board. IBM PCs and XTs decode addresses from hex 200 - 3FF. IBM ATs decode addresses from hex 100 - 3FF. The driver checks for a valid address in the range 100-3F8 hex although the hardware can physically be set from address 0 - 3F8 hex.

In our examples of programming all the different MODEs, we have consistently used D%(*) as the data array. You may prefer to keep your initialization parameters in another array e.g. IX%(*) and not overwrite its values when subsequently selecting other MODEs. This is easy to implement:

```
xxx00 DIM IX%(9), D%(9)
xxx10 MD% = 12
xxx20 IX%(0) = 1 : IX%(1) = 200 : IX%(2) = 50 etc.
```

then:

```
xx100 CALL MSTEP (MD%, IX%(0),STP#, FLAG%)    initialize
```

after this use D%(*) for other MODEs:

```
xx200 MD% = 3
xx210 D%(0) = 0
xx220 STP# = 999
xx230 CALL MSTEP (MD%, D%(0),STP#, FLAG%)
                                              'accel/decel
  .   .   .  etc.
```

4.17 INTERPRETED BASIC (GW, COMPAQ, IBM, ETC.)

Example

BASIC Call:	CALL mstep(MD%, D, STP#%, FLAG%)
BASIC Declaration:	NONE NECESSARY IN BASIC SOURCE CODE. However, a "BLOAD" (Binary load of .BIN file) of the binary file containing the external subroutine must be done prior to calling that subroutine.

Example Program Illustrating a BASIC CALL:

```
10 '***********************************************************
20 '*                                                         *
30 '*          MSTEP-3 Demonstration program                  *
40 '*          Keithley MetraByte Corporation                 *
50 '***********************************************************
60 '
70 SCREEN 0,0,0:WIDTH 80:KEY OFF:CLS
80 LOCATE 1,1,0                               'turn off cursor
90 LOCATE 25,1:PRINT"Metrabyte MSTEP-3 Demo program Loading
                                        MSTEP.BIN driver";:LOCATE 1,1
100 '
110 '--- Load MSTEP.BIN driver --------------------------------------------
120 CLEAR, 48000!                             'set workspace to 48000 bytes
                                                (for example)
130 DEF SEG = 0                               'set zero prior to PEEK's to
                                                zero segment
140 LS = 256 * PEEK(&H511) + PEEK(&H510)
                                              'Basic's data segment
150 SG = LS + 48000!/16                       'end of Basic's data segment
160 DEF SEG = SG                              'this is where to load
                                                MSTEP.BIN driver
170 BLOAD "MSTEP.BIN",0 'load driver with zero offset
180 '

          .
          .

330 '--- Declare other CALL variables -------------------------------------
340 MD% = 15                                  'mode number
350 MSTEP = 0                                 'call offset
360 FLAG% = 0                                 'call error flag variable
370 STP#  = 0                                 'step count (must be double
                                                precision)
380 '
```

```
390 '--- Display menu ---------------------------------------------------

          .
          .
          .

2590 '--- MODE 15: Initialize ---------------------------------------------
2600 CLS
2610 LOCATE 25,1:PRINT"Metrabyte MSTEP-3 Demo program
                                        Initialize - mode 15";:LOCATE 1,1
2620 GOSUB 3250   'display axis selected

          .
          .
          .

2830 CALL MSTEP (MD%,D%(0),STP#,FLAG%)
2840 IF FLAG%<>0 THEN GOSUB 3300:IF E%=0 THEN GOTO 2600
2850 RETURN

          .
          .
          .

3460 '
3470 END
```

NOTE Lines 120 through 160 will not work in GW BASIC.

4.18 QUICKBASIC

Example

BASIC Call: CALL QBPIOINT(MD%, VARPTR(D%(0)) STP#%, FLAG%)

BASIC Declaration: DECLARE SUB QBPIOINT (MD%,BYVAL DUMMY%, FLAG%) The Declaration tells QuickBASIC that the subroutine expects three arguments and that the middle argument is to be passed by value. Remember that BASIC normally passes all arguments by reference (address). This is the only method for passing an array to a subroutine in BASIC: passing the value of the address of the array in effect passes the array by reference. To make use of the callable assembly routine, a ".QLB" (Quick Library) file is created out of the original .ASM source file. Although the format of the subroutine is identical to those used by interpreted BASIC packages, both the Quick BASIC integrated development environment (QB.EXE) and the command line complier (BC.EXE) expect the subroutine to be in a specially formatted .QLB library file. Unlike interpreted BASIC packages, Quick BASIC actually links to the assembly .QLB library file so it is not necessary to include the "jmp QBPIOINT" instruction at location 0 (of the source file) as in interpreted BASIC.

Example Program Illustrating a QuickBASIC CALL:

```
     '**********************************************************************
     '*                                                                    *
     '*              MSTEP-3 Demonstration program                         *
     '*                     for Quick Basic                                *
     '*              Keithley MetraByte Corporation                        *
     '*                                                                    *
     '**********************************************************************
     '
     DIM L$(17), P$(14), U$(14), D%(14), U(14), D$(3)
     COMMON SHARED D%()
     DECLARE SUB QBMSTEP (MD%, BYVAL DUMMY%, STP&, FLAG%)
     SCREEN 0, 0, 0: WIDTH 80: KEY OFF: CLS
     LOCATE 1, 1, 0                                 'turn off cursor
     LOCATE 25, 1: PRINT "MSTEP-3 Demo program loading MSTEP.BIN driver";
     LOCATE 1, 1
     '
     '--- Declare arrays ---------------------------------------------------
     '
     'Default parameters for initializing controllers
     P$(1) = "              Start rate divider: ": U(1) = 125  'slowest
     P$(2) = "     High speed run rate divider: ": U(2) = 20   'medium
     P$(3) = " Acceleration/deceleration steps: ": U(3) = 200  '200 steps
     P$(4) = "                      Motor type: ": U(4) = 2    '4 phase
     P$(5) = "                      Excitation: ": U(5) = 0    'full step
     P$(6) = "                  Logic polarity: ": U(6) = 1    'inverted
     P$(7) = "                    Clock source: ": U(7) = 0    'internal
     P$(8) = "         Switching at standstill: ": U(8) = 1    'on
     P$(9) = "   ON time for power switch freq: ": U(9) = 16   'time on
     P$(10) = " OFF time for power switch freq: ": U(10) = 32  'time OFF
     P$(11) = "  Port A direction, 0=in, 1=out: ": U(11) = 1   'output
     P$(12) = "  Port B direction, 0=in, 1=out: ": U(12) = 1   'output
     P$(13) = "   Auxiliary bit, 1=high, 0=low: ": U(13) = 0   'low
     P$(14) = "               Base I/O address: ": U(9) = &H300
     '
     '--- Declare other CALL variables -------------------------------------
     MD% = 15                     'mode number
     FLAG% = 0                    'call error flag variable
     STP& = 0                     'step count (must be double precision)
     '--- Display menu ----------------------------------------------------

          .
          .
          .

3310 '--- MODE 15: Initialize ----------------------------------------------
3320 CLS
     LOCATE 25, 1: PRINT "Metrabyte MSTEP-3 Demo program
                                  Initialize - mode 15"; : LOCATE 1, 1
          .
          .
          .
     FOR I% = 1 TO 14
     IF U(I%)  > 65535! OR U(I%)  < -32768! THEN D%(I%) = -1: GOTO 3550
     IF U(I%)  > 32767! THEN D%(I%) = U(I%) - 65536! ELSE D%(I%) = U(I%)

3550 NEXT I%
          .
          .
          .
     CALL QBMSTEP(MD%, VARPTR(D%(0)), STP&, FLAG%)
     IF FLAG% <> 0 THEN GOSUB 4120: IF E% = 0 THEN GOTO 3320
     RETURN
          .
          .
          .
     '
     END
```

4.19 MULTIPLE MSTEP-3s IN ONE SYSTEM

What if you wish to operate more than one MSTEP-3 in a system? To avoid conflicts, each MSTEP-3 must have a different base I/O address. For simultaneous operation using interrupts select a different interrupt level for each board. It is possible to share an interrupt level if the operations can be sequential instead of simultaneous. Each board must also be assigned its own CALL routine. To do this start by loading the MSTEP.BIN routine at different locations in memory:

```
xxx10 SG1 = &H3000
xxx20 SG2 = &H4000
xxx30 DEF SEG = SG1
xxx40 BLOAD "MSTEP.BIN",0
xxx50 DEF SEG = SG2
xxx60 BLOAD "MSTEP.BIN",0
```

Now the CALL appropriate to each board can be entered as required. Note that each CALL is preceded by a DEF SEG appropriate to that board:

```
yyy10 DEF SEG = SG1
yyy20 CALL MSTEP (MD1%, D1%(0), STP1#, FLAG1%)
yyy30 DEF SEG = SG2
yyy40 CALL MSTEP (MD2%, D2%(0), STP2#, FLAG2%)
     'etc.
```

If your system requires several MSTEP-3s and a compiled language, consult the manufacturer's technical support group for information.

■ ■ ■

❑

❑

❑

CHAPTER 5

USING THE ACCESSORIES

5.1 M3-DRIVE DESCRIPTION

The M3-Drive is built upon the Superior Electric 230-TH stepper motor drive module. The 230-TH is a stepper driver capable of driving a four-phase stepper motor with winding currents ranging from 1 to 2 amps per phase. It accepts the pulse and direction inputs from the MSTEP-3 to produce either full (200 steps per revolution) or half (400 steps per revolution) step motion from the motor. Half-step operation is often the preferred method since it produces smoother motion with less overshoot for each step. The M3-Drive accepts the AUX bit from the MSTEP-3 and uses it to set up the driver according to the following table:

AUX BIT	MODE	STEPS/REV.
0	Half Step	400
1	Full Step	200

The board portion of the Drive contains a jumper pad that allows easy selection of the winding current from the following values:

1.00 Amps per phase

1.25 Amps per phase

1.50 Amps per phase

1.75 Amps per phase

2.00 Amps per phase

Winding current is factory-preset for 1.00A. However, you may change the setting according to the requirements of your motor. Note that if no jumper is installed the current defaults to 2.0 Amps per phase.

The M3-Drive comes with an attached heat sink. Operation without the supplied heat sink or some equivalent is not recommended, since it could cause the driver to overheat.

5.2 CONNECTING THE M3-DRIVE

All connections between MSTEP-3 and the M3-Drives use mass-terminated cables (such as MetraByte's CDAS-2000). All connections to the motors, limit/home switches, and I/O ports use terminal blocks that allow simple screw-driver connections.

Up to three M3-Drives and their power supplies connect to a single MSTEP-3 for 3-axis motion control. Figure 5-1 shows how the MSTEP-3 is connected to the three M3-Drives and the three motors in a "daisy-chain" arrangement. Note that the M3-Drive closest to the MSTEP-3 always controls the A-axis motor, the next one the B-axis, and the last one the C-axis. This order of control is automatic; it requires no setting of jumpers or switches.

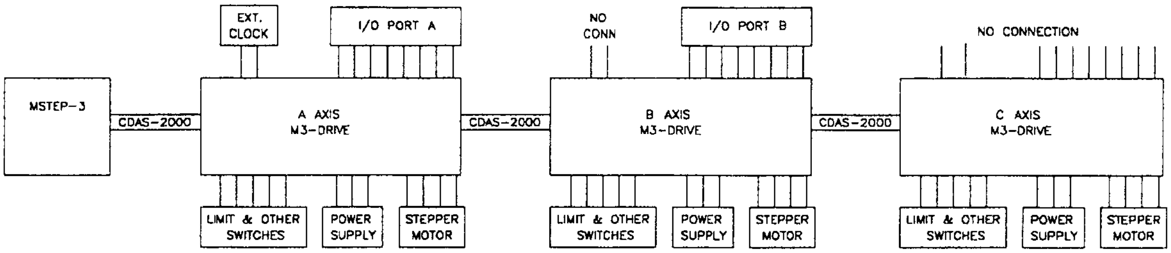


Figure 5-1. MSTEP-3 daisy chained to three M3-Drives.

Access to the I/O ports follows a similar method. The A Axis M3-Drive allows access to I/O Port A, while the B Axis M3-Drive allows access to I/O Port B access. Note that the I/O port terminals on the C-axis M3-Drive are unconnected. Likewise, since there is only one external clock input for the three axes, it is brought out on the A-axis driver. Connecting a clock source to the external clock terminals of the B and/or C axes will have no effect.

5.3 POWER SUPPLY SELECTION

The M3-PWR-24 supply is capable of powering one M3-Drive connected to a 1-to-2 Amp-per-phase motor. To use it, simply connect the wires to the proper power supply terminals on the M3-Drive (the M3-Drive board is marked showing the color of the wires from the M3-PWR-24.).

Other power supplies are also usable. Note that when a phase is turned off, the 230-TH dumps the energy from the motor's collapsing magnetic field back into the power supply. The power supply should therefore have a sufficiently large output capacitor to absorb this charge and prevent damaging voltage spikes. The M3-Drive includes a 4700 f, 50V capacitor to meet this requirement (this is important when powering the unit from a regulated power supply, since regulated supplies often have very little output capacitance).

If you have a need to power several motors from one power supply, the instructions for the 230-TH call for isolation diodes in series with each unit. The M3-PWR-24 supplies the necessary diode (a power Schottky diode to reduce voltage drop and heat) so that no additional diode is required. This diode also protects the unit in the event that the power supply leads are inadvertently connecter in reversed order. If that should happen, there will be no power to the 230-TH and thus no power to the motor.

5.4 CONNECTING A MOTOR TO THE M3-DRIVE

For your convenience, Keithley MetraByte sells a stepper motor in a popular size (Size 23). This motor is rated for 1A per phase; if you plan to use it, set the current jumper accordingly.

In general, stepper motor manufacturers rate their motors for a unipolar driver. This type of driver provides current to only one-half the winding of each phase at any given time. Typically, the center tap is tied to a positive voltage, and one end of the winding is grounded at any given time. The magnetic field is reversed by grounding the opposite end of the winding.

The M3-Drive is a bipolar driver: only two leads from each winding are used, and the driver causes the direction of the current to be reversed within the winding. A user's first instinct is to connect the full winding across the driver, which produces the following three effects:

1. The heat produced by the ohmic power dissipation within the coil doubles, because the same current is now passed through twice the resistance. This effect may be countered by reducing the current to 0.707 times the rated current, producing the same power loss as the full current through a half winding but producing 1.414 times the Ampere-turns within the winding.
2. This increase in the Ampere-turns results in only a slight increase in torque at low step rates, because most motors are designed to be near magnetic saturation at their rated current.
3. Doubling the number of turns actually increases the inductance of the winding by a factor of 4. Since the voltage across the inductor is the same, the rate at which the current increases is reduced by this same factor. This means that at high step rates, the current will not have time to slew to the full rated current.

Thus, it is generally best to only use one-half of a center tapped winding of a motor when connecting it to a bipolar drive such as the M3-Drive. Table 5-1 shows this method of connecting MetraByte's STEP-MOT1 to the M3-Drive.

If you connect a STEP-MOT1 to an M3-Drive according to Table 5-1, you get counter-clockwise rotation (viewed from the shaft-end of the motor) when you specify a positive direction. With other types of motors, you will have to experiment to determine the motor direction. Should you get movement opposite to the direction you need, you may either modify your software to change direction, or swap the leads of any one phase of the motor (swapping the leads to both phases will produce a double reversal back to the original directions). Since Keithley MetraByte obtains STEP-MOT1's from two different sources, the color coding of the wires is shown for either VEXTA or SLO-SYN motors.

Table 5-1. STEP-MOT1 to M3-Drive connections for SUPERIOR ELECTRIC and VEXTA motors.

M3-DRIVE TERMINAL	AMERICAN PRECISION INDUSTRY COLOR	SUPERIOR ELECTRIC COLOR	VEXTA COLOR
A-HIGH	WHITE/GREEN	WHITE/GREEN	BLACK
A-LOW	WHITE	WHITE	YELLOW
B-HIGH	WHITE/RED	WHITE/RED	RED
B-LOW	BLACK	BLACK	WHITE
NO CONN.	RED	RED	BLUE
NO CONN.	GREEN	GREEN	GREEN

The pin labeled /AW0 on the M3-Drive is the All Windings Off pin. With this pin open, the M3-Drive operates normally, but connecting it to a pin marked RET (for Return) causes current to all windings of that motor to be turned off. This allows the motor to be manually moved to the desired position.

5.5 DIRECT CONNECTIONS TO THE MSTEP-3

If you desire to connect some driver other than the M3-Drive to an MSTEP-3, you may do so either directly from the connector at the rear of the MSTEP-3 or by using an STA-50. Since the connector used at the rear of the MSTEP-3 does not follow the convention of odd-numbered pins on one side and even-numbered pins on the other (while the one at the end of the CDAS-2000 cable does), the signals appear on different numbered pins depending on which connector you are looking at. For this reason, Table 5-2 shows the pin numbers at both ends of the cable. If you are using an STA-50 to make connections, use the numbers shown for the connector at the M3-Drive end of the cable.

Should you wish to use a driver that controls the transistors of a unipolar drive directly, the signals for this are brought out via the Connector P2 of the MSTEP-3. You will need to route this cable out of your computer. Table 5-3 shows the connections to P2. Please note that the "TURBO" mode can NOT be used if you are using a drive connected to the P2 connector.

Table 5-2. Connections to the MSTEP-3 via a CDAS-2000 Cable.

	A-AXIS		B-AXIS		C-AXIS	
FUNCTION	MSTEP-3 END	M3-DRIVE END	MSTEP-3 END	M3-DRIVE END	MSTEP-3 END	M3-DRIVE END
LIMIT 1	27	4	6	11	20	39
LIMIT 2	28	6	32	14	45	40
LIMIT 3	26	2	7	13	21	41
LIMIT 4	1	1	33	16	46	42
MON	2	3	8	15	22	43
HOME	3	5	34	18	47	44
STEP	5	9	36	22	24	47
DIRECTION	4	7	10	19	49	48
HOLD	29	8	9	17	48	46
AUX. BIT	30	10	35	20	23	45
GROUND	31	12	37	24	50	50

	I/O PORT A		I/O PORT B	
FUNCTION	MSTEP-3 END	M3-DRIVE END	MSTEP-3 END	M3-DRIVE END
BIT 0	14	27	43	36
BIT 1	39	28	18	35
BIT 2	15	29	19	37
BIT 3	40	30	44	38
BIT 4	13	25	42	34
BIT 5	38	26	17	33
BIT 6	12	23	41	32
BIT 7	11	21	16	31

EXTERNAL CLOCK	25	49	

Table 5-3. Connections to Connector P2.

SIGNAL	A-AXIS PIN	B-AXIS PIN	C-AXIS PIN
PHASE 1	7	6	1
PHASE 2	8	5	2
PHASE 3	9	14	3
PHASE 4	10	4	18
PHASE 5	11	15	17
GROUND	12	13	16

CHAPTER 6

STEPPER MOTORS & TRANSLATORS

6.1 HOW A STEPPER MOTOR WORKS

The diagrams in Figure 7-1 will assist you in understanding how a stepper motor operates. In a real stepper motor there are are more poles on the stator and rotor, but the operation is similar. The rotor of the simplified version consists of a cylindrical magnet sandwiched between two discs each with 3 poles on them. The discs are rigidly clamped together and a half pole pitch out of step with each other (in this case 60 degrees out of alignment). Since the poles stick out (salient poles) on both the stator and the rotor, there are positions where the magnetic circuit or reluctance is at a minimum and the rotor will tend to align in these preferred positions even when the power is off. As soon as power is applied to the windings, this "locking" effect becomes even more pronounced, the external torque required to move the rotor to the next preferred position is known as the holding torque. The holding torque with power applied is usually about 10 times the holding torque with the stator de-energized.

By turning the stator windings on in sequence, it is possible to pull the rotor around in either direction according to the sequence. The diagrams show 4 steps of the motor, note how the windings are alternately energized and the current reversed to accomplish the stepping (remember like poles repel and unlike poles attract!). In this example each switching of the windings results in the rotor moving 30 degrees, so this motor would have a 30 degree step angle and take 12 steps to complete a revolution.

Most commercially available stepping motors have finer step angles of 1.8, 7.5 and 15 degrees with the 1.8 degree (or 200 step/rev) motor being one of the most popular. Smaller steps require more poles on the rotor and stator, in fact for 200 step/rev. motors, the rotor resembles two 50 tooth gearwheels although the stator uses some tricks by grooving the poles to reduce the number of windings and hence complexity. The permanent magnet of the rotor is magnetized to the maximum extent by the manufacturer within a magnetic circuit similar to the stator and cleverly assembled without removing the magnetic circuit. If the rotor is removed from the stator, the loss of the magnetic circuit will tend to self demagnetize the magnet. This leads to a loss of torque on re-assembly as the magnet is no longer as powerful. If you want to see what a real stepper motor looks like inside, bear this in mind before you tear a good motor apart. Some stepper motors do not use a permanently magnetized rotor, but operate similarly to those that do, except that they have zero holding torque when de-energized. Some stepper motor drives use a technique known as microstepping which leads to very small step angles of a few tenths of a degree. Instead of switching the windings completely on or off, the windings are energized partially so that the resultant magnetic field vector is moved in much smaller increments than the pole angles. This requires a much more complex drive circuit and a slightly different type of design of stepping motor. The MSTEP-3 cannot microstep motors directly, although it can provide the pulse and direction outputs for a microstepping translator.

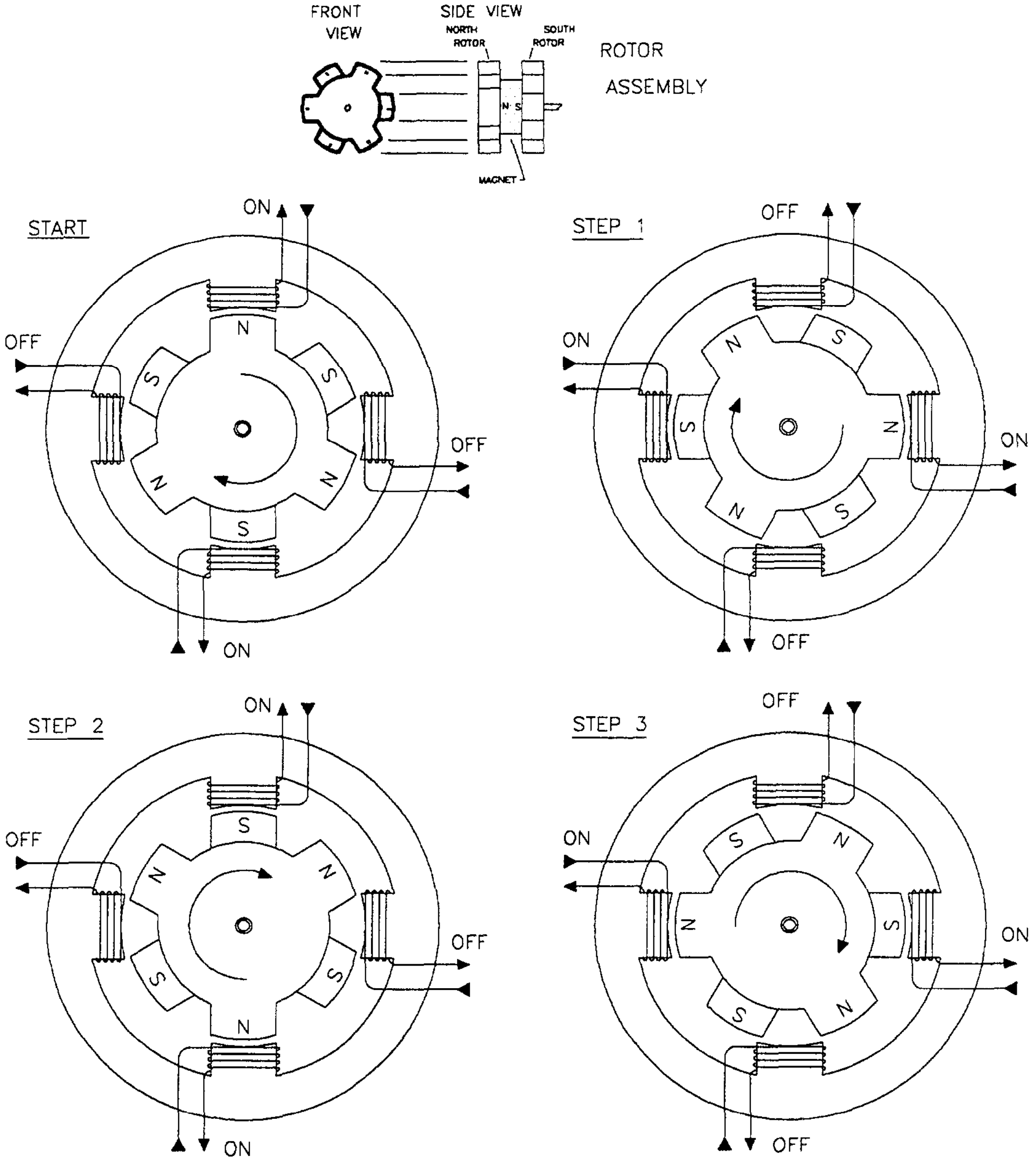


Figure 6-1. Stepper motor operation.

6.2 TORQUE VERSUS SPEED

As the stepping rate or speed of rotation of a stepping motor rises, the torque that the motor can provide tends to fall. A typical torque/speed curve is shown in Figure 6-2.

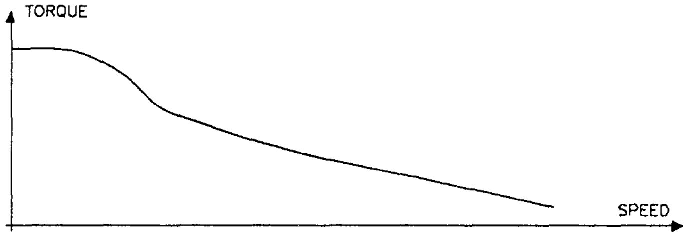


Figure 7-2. Typical torque/speed curve of a stepping motor.

As the speed rises, the winding currents tend to reduce due mainly to two effects. In turn the reduced winding current leads to reduced torque. The two effects that cause this are as follows:

Back EMF

As the magnetized rotor turns within the stator, it induces a voltage in the windings which opposes the driving voltage. This in turn reduces the winding current. The induced EMF is proportional to speed, rising as the speed increases. The solution to this is to increase the driving voltage as the speed rises, but this greatly increases the complexity of the drive electronics.

Inductance

The winding inductance and resistance control the rate of rise of current in accordance with the well known relation

$$I = (1 - E^{-Rt/l}) * V/R$$

The quantity L/R is known as the time constant, the smaller it is, the faster the current will rise on switching the winding on. Plainly we need the smallest inductance and largest resistance to reduce the time constant although this would lead to a very lossy and inefficient winding. In practice we can add a bit of external resistance and improve the performance, although the resistor wastes power. This forms the basis of the simple L/R translator as embodied in the design of the M3-Drive.

6.3 RESONANCE

Each time the rotor is stepped, it tends to overshoot the desired position slightly and performs a small damped oscillation. The damping is mainly electrical due to induced voltage in the windings. The frequency of the damped oscillation depends on the total inertia of the system. It is possible that at certain speed the stepping rate will become a harmonic of the natural oscillation frequency, and under these conditions the oscillation builds up to a point that the motor misses steps. This condition is known as resonance.

Resonance problems can be avoided by

1. Not operating in or close to a resonance region.
2. Accelerating the motor through the resonant point as fast as possible.
3. Providing additional damping in the form of mechanical or viscous damping (Lanchester dampers) and/or altering the moment of inertia.

If the motor is operated at low stepping rates, resonance is not usually likely to be a problem.

6.4 FULL- AND HALF-STEP OPERATION

There are two common switching sequences used for stepper motors. In the full-step sequence, two windings are always energized. This applies whether the motor has three, four, or five phase windings. For example, for a 4-phase motor

FULL STEP:

	WINDING 1	WINDING 2	WINDING 3	WINDING 4
Step 1	on	on	off	off
Step 2	off	on	on	off
Step 3	off	off	on	on
Step 4	on	off	off	on
etc...				

Full step gives the greatest torque and the rated step angle of the motor e.g. 200 steps/rev.

In half step sequence, switching alternates between having 2 windings on and 1 winding on. It takes twice as many steps to travel the same distance, thus a 200 step/rev. motor with 1.8 degree stepping angle in full step sequence will produce 400 steps/rev. and 0.9 degree step angles in half step sequence.

HALF STEP:

	WINDING 1	WINDING 2	WINDING 3	WINDING 4
Step 1	on	off	off	off
Step 2	on	on	off	off
Step 3	off	on	off	off
Step 4	off	on	on	off
Step 5	off	off	on	off
Step 6	off	off	on	on
Step 7	off	off	off	on
Step 8	on	off	off	on
etc...				

Half step sequence offers a finer positioning resolution and somewhat smoother stepping, but since fewer windings are excited on average, the torque and maximum speed capabilities can be lower than full-step. The M3-Drive can drive in either half- or full-step mode (see Section 5.1). When using the outputs of P2 to control the phasing, use the Initialization mode (Mode 15) to set up for full- or half-step mode. When using the M3-Drive, half- or full-step operation can be selected by setting the AUX bit (see Section 5.1).

6.5 TRANSLATORS & INDEXERS

If you are new to stepper motors, you will start reading catalogs and find all sorts of equipment with strange names. This little explanation may assist you in understanding some of the terminology and its relevance to the MSTEP-3.

A TRANSLATOR is a device that takes a step pulse and direction input and drives the stepper motor windings - it is essentially the power drive and pulse pattern determining stage, but it does not keep count of the pulses or position. An INDEXER goes further and will take an input from switches or a keyboard and move the motor the number of steps input at the keyboard, it also keeps track of the motor position.

In this sense, the M3-Drive is a TRANSLATOR whereas an IBM PC, MSTEP-3, M3-Drive, and a suitable program can perform functions equivalent to an INDEXER.

The MSTEP-3 does not have to be used with the M3-Drive. It can just as easily be connected to a commercial TRANSLATOR using the direction and pulse outputs of each channel. Commercial INDEXERS are usually complete self-contained positioning systems and cannot be connected to the MSTEP-3.

6.6 MECHANICAL DESIGN

Simple systems can often be assembled with very little involvement in optimizing the mechanical design. They can almost be put together on a "try it and see" basis, this is one of the advantages of the simplicity of stepper motors. On the other hand, if you want to obtain maximum performance especially with larger drives, you should carefully analyze your mechanical requirements and match the motor and translator to your needs. Torque and power of stepper motors depend on the motor frame size and rating and range from 35 oz/in and fractional H.P. to 5000 oz/in and ratings of 3-4 H.P.

The major problem in mechanical design is sizing the stepper motor to the application. The first determination to make is the maximum torque required. This depends on the static load e.g. friction and the dynamic load set by inertia and acceleration and possibly viscous drag. Also the available torque will decline as the speed rises due to the characteristics of the motor and its translator. Taking these factors into account is a fairly complex exercise and is best described in the design literature provided by stepper motor manufacturers (see Appendix B). One excellent source is the "The Art and Practice of Step Motor Control" by Bert Leenhouts, available @ $84 from Intertec Communications Inc., 2472 Eastman Ave., #33-34, Ventura, CA. 93003-5774, Phone (805)-658-0933.

Other considerations are holding torque, motor size and design - shafts are available on one end or both ends etc. If you need leadscrews, X-Y tables etc. many vendors provide this type of accessory equipment, a short list is provided in Appendix B. ■

❑

❑

❑

CHAPTER 7

TESTING & MAINTENANCE

No periodic calibration or maintenance is required for the MSTEP-3 or M3-Drive, there are no user adjustments.

The best method of testing a system is to run the DEMO program and exercise the motor or shaft encoder using this program. If you receive error code 5 while running the demo program, there are a number of possible conditions to check before assuming that the MSTEP-3 board is faulty, as follows:

1. If you receive Error #6 on initialization of both channels, it means that the driver software is not locating the MSTEP-3 at the I/O address specified in the initializing parameters. This may be caused by the BASE I/O ADDRESS dipswitch being set to an address other than that specified, a conflict with the address of another peripheral or a truly faulty MSTEP-3. The driver software performs a quick write/read test to the MSTEP-3 control register on initialization, and if it detects any discrepancies will return Error #6.
2. If you consistently receive Error #6 on initializing one of the three channels, or receive it in the motion commands, it points to a faulty PPMC-103A controller. These are plug in chips and can be replaced. A PPMC-103A which is not correctly handshaking with the driver when it is ready to receive or provide data will give this hardware fault.

Other errors that are possible are open limit switch wiring, faulty limit switches etc. These can be checked by running the READ STATUS option of the DEMO program and exercising the limit switches.

If you wire the stepper motor up incorrectly with its windings in the wrong sequence, it can lead to bumpy or rough operation or dithering and similar effects. Check the stepper motor wiring and make sure that the windings are switching on and off using a voltmeter and the JOG command.

The DEMO program will expose 99% of the likely problems that you can have and also provides a baseline for comparison if you suspect you have a programming problem. The remaining 1% of hardware problems, stuck bits, open lines etc. can be determined by writing and reading the I/O ports. This tends to be tedious work and a program such as DEBUG which makes I/O operations short and simple is the best choice for this chore.

If you have problems with your MSTEP-3, please call MetraByte applications engineering at (508)-880-3000. If you need to return any hardware, we will issue you an R.M.A. (return material authorization) number to mark on your package. Please do not send us material without an R.M.A. number as it greatly complicates tracing its origins and faults. It is also useful to our test technicians if you can include a brief description of the problem and in what circumstances it occurs. ■

CHAPTER 8

FACTORY RETURNS

Before returning any equipment for repair, please call 508/880-3000 to notify MetraByte's technical service personnel. If possible, a technical representative will diagnose and resolve your problem by telephone. If a telephone resolution is not possible, the technical representative will issue you a Return Material Authorization (RMA) number and ask you to return the equipment. Please reference the RMA number in any documentation regarding the equipment and on the outside of the shipping container.

Note that if you are submitting your equipment for repair under warranty, you must furnish the invoice number and date of purchase.

When returning equipment for repair, please include the following information:

1. Your name, address, and telephone number.
2. The invoice number and date of equipment purchase.
3. A description of the problem or its symptoms.

Repackage the equipment. Handle it with ground protection; use its original anti-static wrapping, if possible.

Ship the equipment to

Repair Department
Keithley MetraByte Corporation
440 Myles Standish Boulevard
Taunton, Massachusetts 02780

Telephone 508/880-3000
Telex 503989
FAX 508/880-0179

Be sure to reference the RMA number on the outside of the package! ■

□

□

□

APPENDIX A

PPMC-103A SPECIFICATIONS & PROGRAMMING

The data sheet of the stepper motor controller chip used in the MSTEP-3 is reproduced in its entirety here by permission of Sil-Walker America [Sil-Walker America Inc., 653 Las Casas Avenue, Pacific Palisades, CA. 90272 Phone:(213)-454-4772]. If you intend programming the stepper channels directly, want to understand how the MSTEP.BIN driver works, write your own routines, or know more about the limit switch operation or fine details of the controllers, these data pages will provide most of the answers.

Due to the mapping of the PPMC-103A registers into the I/O map of the MSTEP-3 the following correspondences apply:

A0RSEL of the MSTEP-3 Data Select Register = A0 of all PPMC-103As

RESET of a PPMC-103A is accomplished on the MSTEP-3 by the active RESET line of the IBM PC bus (on power-up). Software RESET is also available.

The PPMC-103A has a couple of minor programming quirks to watch. The motion commands accelerate/decelerate or move at constant speed will always step one pulse more than the number input. The MSTEP.BIN driver corrects for this characteristic in these two commands by subtracting one from the data before sending it along to the PPMC-103A's. Also the singular data values 0 and 1 are caught in the driver - see MSTEP.ASM. If zero data is applied, the PPMC-103A will step 1 pulse, whereas if 1 is supplied it will step 2 pulses. The MSTEP.BIN driver grabs these two conditions and does nothing if zero is applied or aborts to a jog command if 1 is applied. In this way the MSTEP.BIN driver always does what you would expect - the PPMC-103A has somewhat different although consistent rules.

A complete English language edition of the PPMC103-A data manual is not presently available. Until such an edition becomes available, Keithley MetraByte will offer a sheet outlining the differences between the PPMC-103A and the PPMC-101C along with the complete PPMC-101C data sheet. Meanwhile, keep in contact with Keithley MetraByte to learn when more complete information will become available.

Sil-Walker

PPMC-103A *New for 1990*

The **PPMC-103** is an improved, **CMOS** version of the PPMC-101/102 motor control IC. The use of CMOS over conventional NMOS reduces the power consumption from 625 mW (PPMC-101) to 150 mW for the PPMC-103. The command set is downward compatible with the PPMC-101/102 and adds three new functions. The PPMC-103 also increases the maximum pulse rate to **19K** pps and increases the smoothness of the acceleration/deceleration operation and increases the number of acceleration/deceleration steps to 11,220 max. (The PPMC-101/102 had a max of 8,160 accel/decel steps).

The major improvements are;

1 - Increase pulse rate to 19K pps maximum (P-out mode)
2 - Increase pulse rate to 12K pps maximum (phase outputs)
3 - CMOS technology. Decrease power consumption to 150 mW.
4 - Increase smoothness of acceleration/deceleration.
5 - Increase number of acceleration/deceleration steps to 11,220 max.
6 - Add a "SOFTWARE RESET COMMAND".
7 - Add an "AUXILIARY OUTPUT COMMAND" which controls the logic level
8 - Add a "SWITCHING RATE CONTROL COMMAND". This allows the duty cycle and the frequency of "EXCITATION SIGNAL SWITCHING" to be controlled by the user. This function is used to reduce power consumption of the motor at standstill by chopping the phase outputs, S1-S5.

The PPMC-103A is pin-compatible with the 101/102 versions, except that a previously unused pin (pin 36) is now used as an output. If this pin was left open or pulled up to 5V through a 3.3kΩ resistor (as per the manual) on designs using the 101/102, then the 103A can be substituted directly. Of course, the crystal must be upgraded to 12MHz for the 103A to give its' top performance.

When using the PPMC103A, substitute the following page 2 for the original page 2 appearing in the PPMC101/102 manual. All other pages should be treated as an addendum and inserted at the back of the manual.

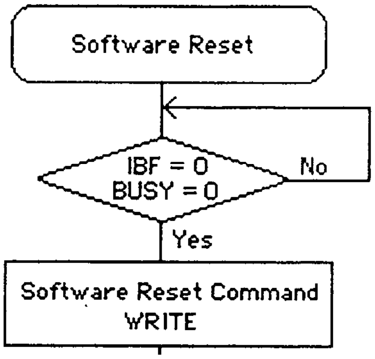
PPMC-103A COMMANDS

SOFTWARE RESET COMMAND

This is a NEW command, available on the PPMC-103A. This command causes a "RESET" of the PPMC with the same effect as a "hardware reset" (ie, the removal of power). After this command, the PPMC MUST be re-INITIALIZED. Any parameter in the "INITIALIZATION COMMAND" can be changed at this time. The "SOFTWARE RESET" will also cause all phase outputs, S1-S5, to turn off.

"SOFTWARE RESET" cannot be sent while the motor is in motion. The flowchart below must be followed.

7	6	5	4	3	2	1	0	
1	1	0	0	0	0	0	0	} Software Reset Command



AUXILIARY OUTPUT COMMAND

This command controls the logic level on pin 36. (On the PPMC-101/102, pin 36 is unused). This command can be sent at any time, even while the motor is running. Pin 36 assumes the appropriate logic level about 29 µsec after the command is sent. (12 MHz operation)

7	6	5	4	3	2	1	0	
1	1	0	0	0	0	0	1	} AUXILIARY OUTPUT "OFF" (pin 36 = 0)

7	6	5	4	3	2	1	0	
1	1	0	0	0	0	1	0	} AUXILIARY OUTPUT "ON" (pin 36 = 1)

SWITCHING TIME SETTING COMMAND

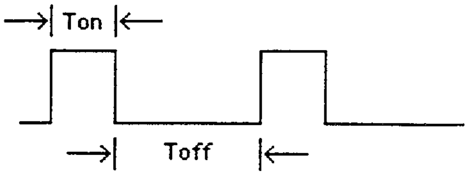
This new command allows programmable control over the frequency and duty cycle of the chopping of the "phase outputs", S1-S5, when the motor is at standstill. The "EXCITATION SIGNAL SWITCHING" bit must be set during "INITIALIZATION" to activate the phase chopping.

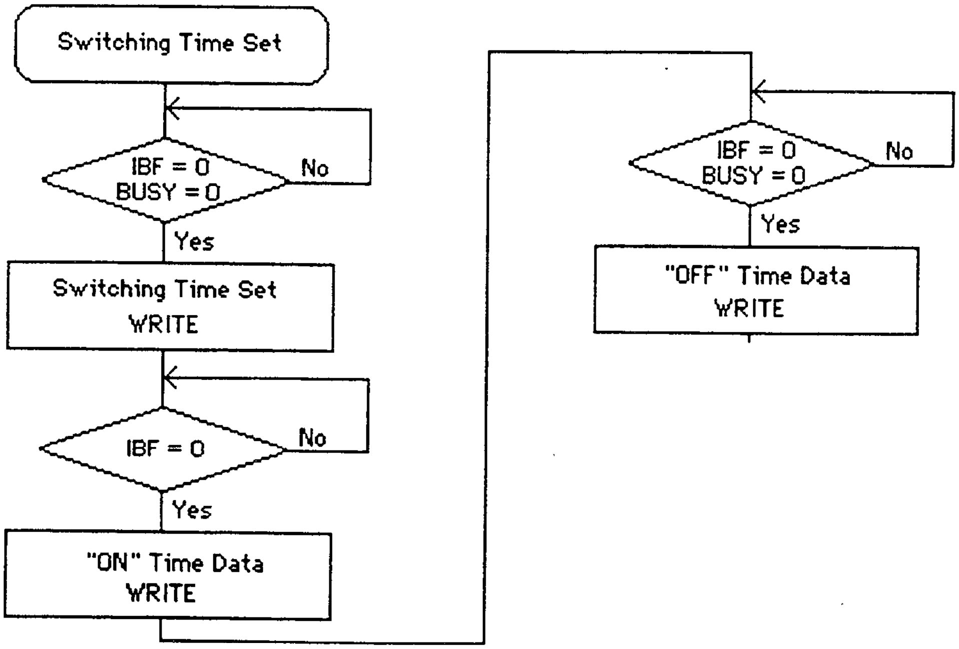
7	6	5	4	3	2	1	0	
1	0	0	0	0	1	0	0	} Switching Time Set Command
								} "ON" Time (Pon)
								} "OFF" Time (Poff)

Ton = (Pon x 2.5) + 2.5 (µseconds)

Toff = (Poff x 2.5) + 28.75 (µseconds)

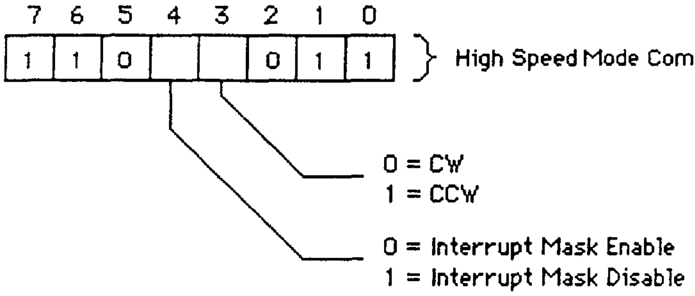
$$\text{Switching Frequency} = \frac{1}{(\text{Ton} + \text{Toff})} \quad \text{(Hz)}$$





HIGH SPEED MODE COMMAND (P-out only, Accel/Decel only)

This is a NEW command, to allow the P-OUT pulse rate to reach 19kpps. In this mode the "phase outputs", S1-S5 are disabled. The maximum pulse rate can only be achieved if the "external clock input", pin 39, is driven from the "SYNC" output, pin 11. With a 12 MHz crystal driving the PPMC-103A, the "SYNC" output = 800kHz. The maximum pulse rate is obtained when the external CLOCK input, pin 39, is driven at 266.67 kHz and RH = 13. The SYNC output must be divided by 3 to obtain 266.67kHz.

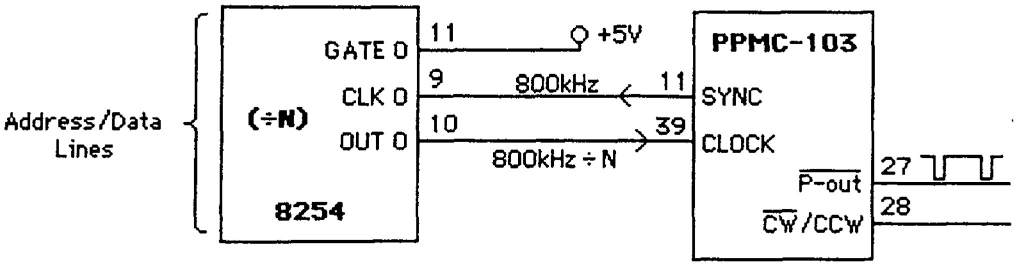


The flowchart for this command is identical to that of the Accel/Decel Command.

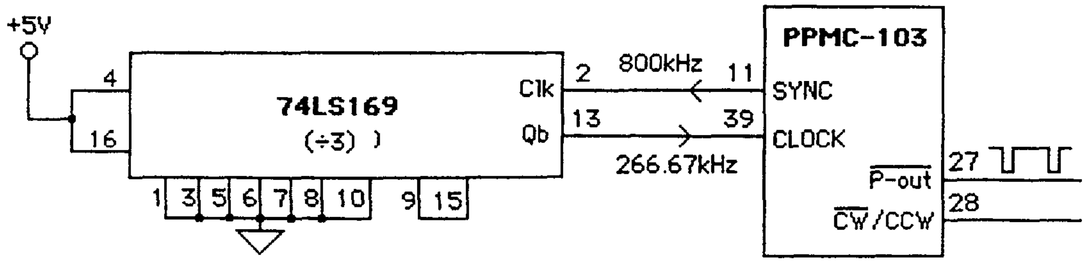
The output pulse rate (speed of motor) is calculated from the following formula;

$$\boxed{\text{Speed} = \frac{10^6}{(RH + 1) \times (T\text{clock}) + (7.5\mu\text{sec.})} = \text{pulses-per-second}} \quad \text{for } RH \geq 13$$

There are several ways to divide the SYNC output to levels useable by the PPMC-103A. The most versatile way is to use a programmable divider such as the INTEL CORP. 8254, which contains three 16 bit counter-timers. Programming the 8254 to divide by 3 gives (800kHz ÷ 3) = 266.67 kHz from SYNC output. Applying this to the CLOCK input at pin 39 will allow the maximum pulse rate of 19kHz. The beauty of using a software programmable divider is the wide range of speeds then available to the PPMC-103A. For example, using the 8254 (or any 16-bit divider), we can divide the SYNC clock by up to 65,536. This translates to a range of pulses-per-minute to 19k pulses-per-second.



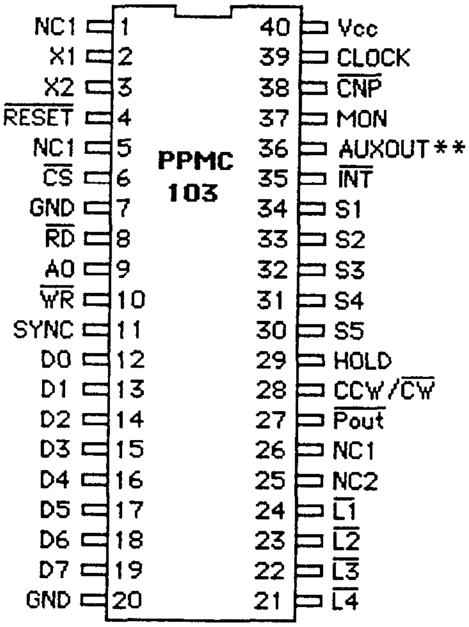
Alternatively, a fixed divide-by-three circuit may be used where only the highest range of speeds is desired;



PPMC103A

Number of steps	:	16,777,216 max
Acceleration/Deceleration pulse number	:	4 - 11,220
Maximum pulse rate	:	19k pps (P-out mode, ext clock = 250KHz, RA = 13)
	:	12k pps (Phase outputs S1-S5, ext clock = 250KHz, RA = 20)
Power Supply	:	5 volts ± 10% 30 ma max with 12MHz x'tal **(CMOS)**

2. TERMINAL ASSIGNMENT AND FUNCTIONS


PPMC
103
NC1 1
X1 2
X2 3
$\overline{\text{RESET}}$ 4
NC1 5
$\overline{\text{CS}}$ 6
GND 7
$\overline{\text{RD}}$ 8
A0 9
$\overline{\text{WR}}$ 10
SYNC 11
D0 12
D1 13
D2 14
D3 15
D4 16
D5 17
D6 18
D7 19
GND 20
40 Vcc
39 CLOCK
38 $\overline{\text{CNP}}$
37 MON
36 AUXOUT**
35 $\overline{\text{INT}}$
34 S1
33 S2
32 S3
31 S4
30 S5
29 HOLD
28 CCW/$\overline{\text{CW}}$
27 $\overline{\text{Pout}}$
26 NC1
25 NC2
24 $\overline{\text{L1}}$
23 $\overline{\text{L2}}$
22 $\overline{\text{L3}}$
21 $\overline{\text{L4}}$

** AUXOUT, pin 36, was previously designated as unconnected in the PPMC-101/102.

SIGNAL	PIN #	I/O	DESCRIPTION
X1, X2	2, 3	I	Crystal inputs (12MHz max.)
$\overline{\text{RESET}}$	4	I	RESET input, active low
$\overline{\text{CS}}$	6	I	Chip Select input, active low
$\overline{\text{RD}}$	8	I	Read strobe, active low
A0	9	I	Address 0
$\overline{\text{WR}}$	10	I	Write strobe, active low
SYNC	11	O	Timing output
D0 - D7	12-19	I/O	Data Bus, 8-bits
$\overline{\text{L4}}$	21	I	Reverse high-speed limit input
$\overline{\text{L3}}$	22	I	Forward high-speed limit input
$\overline{\text{L2}}$	23	I	Reverse absolute limit input
$\overline{\text{L1}}$	24	I	Forward absolute limit input
$\overline{\text{P-out}}$	27	O	Pulse output
$\overline{\text{CW}}$/CCW	28	O	Direction output (forward/reverse)
HOLD	29	O	Motor status output
S5	30	O	Motor 5th phase output
S4	31	O	Motor 4th phase output
S3	32	O	Motor 3rd phase output
S2	33	O	Motor 2nd phase output
S1	34	O	Motor 1st phase output
$\overline{\text{INT}}$	35	O	Interrupt output
AUX OUT	36	O	Auxiliary output (software controlled)
MON	37	I	External control of motor power '1' = Motor ON '0' = Motor OFF
$\overline{\text{CNP}}$	38	I	Base point limit input
CLOCK	39	I	External clock input (250 KHz max)
Vcc	40	I	+5v DC power supply
GND	7, 20	I	Ground (0 volts)
NC1	1, 5, 26	I	Pull up to Vcc with 3.3K resistor
NC2	25	O	Leave this pin open (not connected)

INTRODUCTION

PPMC101C/102A is a unique one-chip LSI specially designed to interface a stepper motor to an 8-bit micro computer with no additional hardware. PPMC101C/102A provides 8 kinds of different operations by the command of master CPU including acceleration/deceleration and constant speed operation.

Operating frequency and number of phase for stepper motor are programmable. Distribution signal to excitation driving circuit can also be programmble for selection of 2-phase or 1-2 phase excitation (2-3 phase excitation for 5-phase motor) for 3, 4 and 5 phase motor.

In addition, PPMC101C/102A provides five kinds of "limit" switch input. Complete function necessary to control stepper motor is included in one chip LSI. The PPMC101C/102A can be easily interfaced with a microcomputer system.

1. PPMC101C/102A SPECIFICATIONS

Operation Command

- * *Emergency Stop*
- * *Decelerating Stop*
- * *Single Step*
- * *Acceleration & Deceleration*
- * *Constant Speed Operation*
- * *To move to the "Limit"*

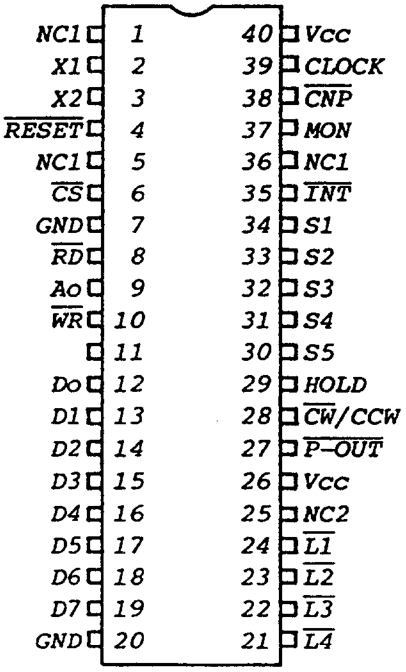
  (1) To move to the high speed limit
  (2) To move to the base point

Excitation Method

Motor	*Excitation*
3-phase	*2 phase* *1-2 phase*
4-phase	*2 phase* *1-2 phase*
5-phase	*2 phase* *2-3 phase*

Number of steps	:	*16,777,216 max*
Number of pulse for acceleration/deceleration	:	*4 - 8,160*
Maximum pulse rate	:	*PPMC101C ... 5K pps (RA=20, fo=100KHz)* *PPMC102A ... 10K pps (RA=20, fo=200KHz)*
Power supply	:	*5V ± 5% 125mA max*

2. TERMINAL ASSIGNMENT AND FUNCTIONS



(Top View)

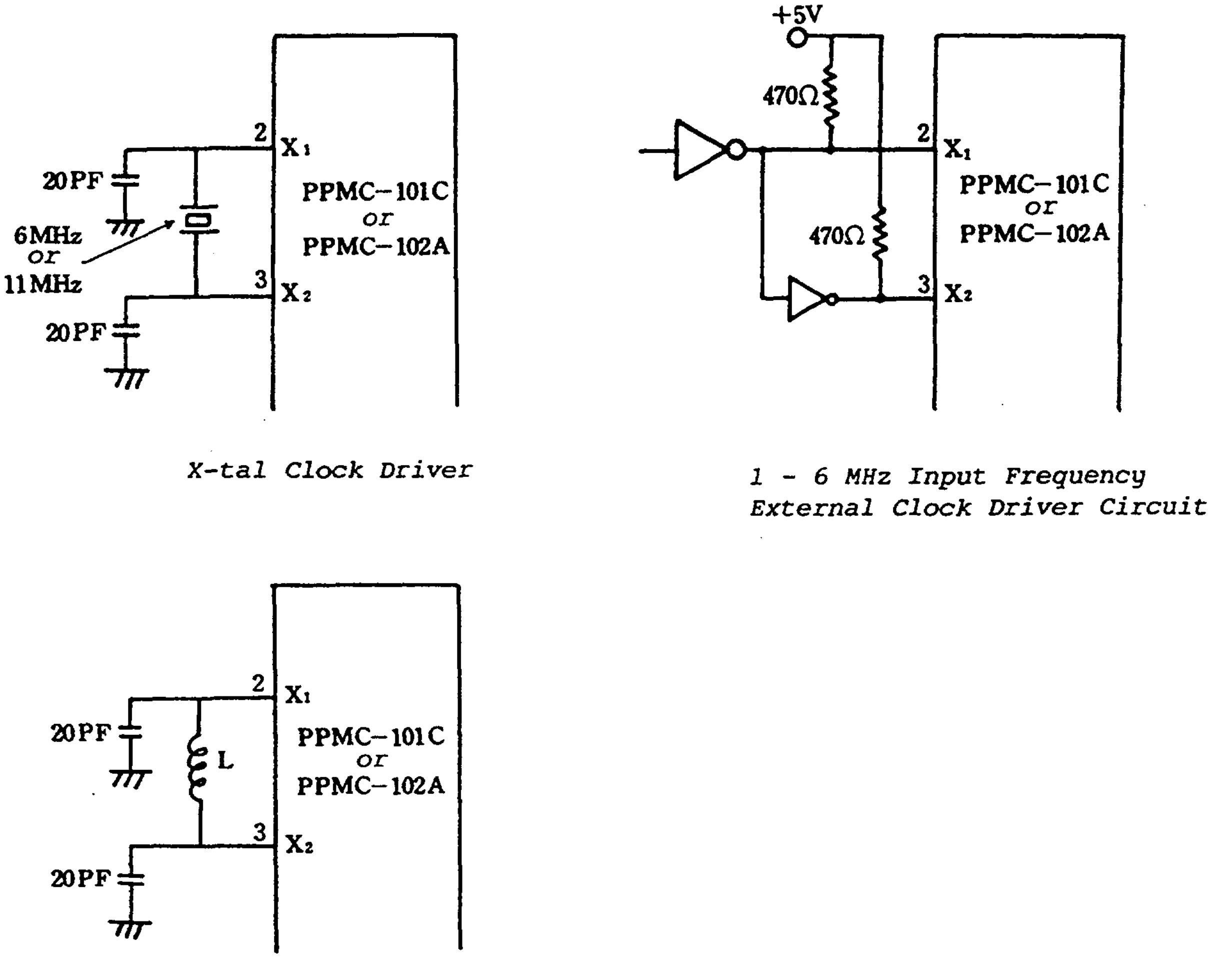
40 pin Dual-In-Line

Signal	*Pin#*	*I/O*	*Description*
X1, X2	2, 3	I	X-tal
$\overline{RESET}$	4	I	RESET input
$\overline{CS}$	6	I	Chip Select
$\overline{RD}$	8	I	READ strobe
Ao	9	I	Address 0
$\overline{WR}$	10	I	WRITE strobe
SYNC	11	O	Timing output
Do - D7	12-19	I/O	Data Bus 8-bit
$\overline{L4}$	21	I	Reverse high speed limit input
$\overline{L3}$	22	I	Forward " " " "
$\overline{L2}$	23	I	Reverse limit input
$\overline{L1}$	24	I	Forward " "
$\overline{P\text{-}OUT}$	27	O	Pulse output
CCW/$\overline{CW}$	28	O	Forward/Reverse status '0' = Forward '1' = Reverse
HOLD	29	O	Motor HOLD output
S5	30	O	Motor 5th phase output
S4	31	O	" 4th " "
S3	32	O	" 3rd " "
S2	33	O	" 2nd " "
S1	34	O	" 1st " "
$\overline{INT}$	35	O	Interrupt signal
MON	37	I	External control '0' = Motor ON '1' = Motor OFF
$\overline{CNP}$	38	I	Base point signal input
CLOCK	39	I	External clock input
Vcc	26,40	I	+5V DC
GND	7,20	I	0 V
NC1	1,5,36	I	pull up to Vcc with 3.3K ohm or open
NC2	25	O	OPEN

PIN DESCRIPTION

2-1 X1, X2

Inputs for a crystal. PPMC101C is normally operated with a 6 MHz crystal and PPMC102A with a 11 MHz crystal. You may also drive the clock inputs with an LC turned circuit or an external clock source (2 phases) as shown in Fig 2-1. 1 to 6 MHz for PPMC101C and 1 to 11 MHz for PPMC102A can also be used for driving frequency, but the operating speed slows down in accordance with the clock frequency.



X-tal Clock Driver

1 - 6 MHz Input Frequency External Clock Driver Circuit

LC Turned Circuit Clock Driver

Fig 2-1

PPMC-101C		PPMC-102A	
L=130μH	3 MHz	L=120μH	3.2 MHz
L= 40μH	5 MHz	L= 45μH	5.2 MHz

2–2 $\overline{\text{RESET}}$

Input used to reset status flip-flops and to set the program counter to zero. This pin should be connected to the RESET signal of a user's system. 50msec after the RESET signal rising edge the PPMC101C/102A is operative for initialization and operation command. The pulse width of the RESET signal must be no less than 2.5 µsec.

2–3 $\overline{\text{CS}}$

Input for chip select. To input the decorded signal from upper bits of ADDRESS. PPMC101C/102A is accessible at a low level 'O' on CS.

2–4 $\overline{\text{RD}}$

I/O read input which enables the master CPU to write data and status words from the PPMC101C/102A. The OUTPUT DATA BUS BUFFER or status register can be READ at a low level 'O' on $\overline{CS}$ and $\overline{RD}$.

2–5 $\overline{\text{WR}}$

I/O write input which enables the master CPU to write data and commands words to the PPMC101C/102A. Data on INPUT DATA BUS BUFFER can be written at a low level 'O' on $\overline{CS}$ and $\overline{RD}$.

2–6 AO

Address input used by the master processor to indicate whether the byte transfer is data or command as shown in the following table

Ao	*$\overline{RD}$*	*$\overline{WR}$*
0	*Data Resister*	*Data Resister*
1	*Status Resister*	*Command Resister*

Table 1

2–7 SYNC

Output signal which occurs once per execution of internal command in the PPMC101C/102A. It is also used to synchronize the single step operation. It is to be normally OPEN and used to check IC operation.

2–8 DO–D7

Tri-state, bidirectional DATA BUS BUFFER lines used to interface the PPMC101C/102A to an 8-bit master system data bus.

2-9 $\overline{L1}$ – $\overline{L4}$, $\overline{CNP}$

Inputs for external 'Limit' switches. Each signal is activated at a low level 'O'. Fig 2-2 shows the idea of 'Limit" switches.

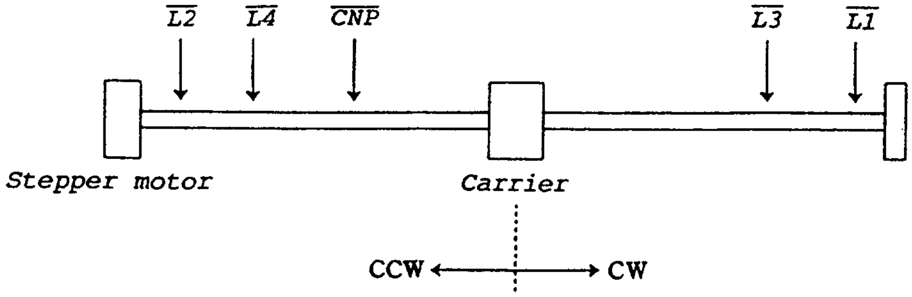


Fig 2-2

L1, L2 :

These switches are set at a maximum limit position where the carrier does not move further in CCW or CW. The motor will stop immediately when the carrier moves to these points regardless of the operation command. The carrier will no longer move further in the same direction even when it receives a command to move in the same direction. The carrier will start to move in the opposite direction only when it receives the command to move in reverse.

L3, L4 :

These switches must be positioned between L1 and L2, at a minimum distance corresponding to the number of deceleration steps. The stepper motor begins to decelerate at these positions (L3 or L4) in order to stop inside of L1 or L2.

$\overline{CNP}$:

Signal from $\overline{CNP}$ is used to establish a convenient reference point (Base point) with which the PPMC can monitor the position of the carrier. It does this by counting the number of steps in the data register. For example, in figure 2-2, in order to establish a convenient base point the command "move to base point" is used (see section 3-3-8). The motor will move the carrier to the position marked $\overline{CNP}$ and stop. Work can then proceed from this point.

2-10 P-OUT, CCW/CW

P-OUT is used for pulse output for other stepper motor driving modules without using PPMC phase output. It is useful for bipolar drive, switching drive and other special type of excitation method. It is recommended to use a decoder for CW or CCW pulse generation in combination with one-shot TIMER as shown in Fig 2-3 because driving module sometimes require 10 to 20 µsec pulse width. The signal from pulse output is a 5 µsec negative pulse and signal for direction is indicated by its level. In addition, these signals can be used for monitoring direction or number of pulses for rotation.

CCW/CW can be activated only when P-OUT is active.

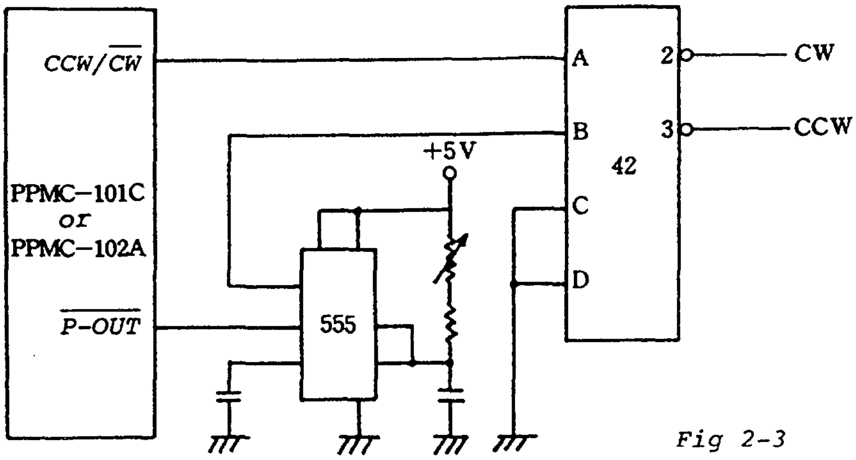


Fig 2-3

P-OUT is always available as well, in use of any type of phase output. (see page 13. motor code 01, 10, 11)

2-11 HOLD

HOLD output is high 3 msec after motor stops, but it is active only when bit 5 of the initialization command is set. (see page 13)

2-12 S1-S5

Provides signal for motor excitation drive.

Motor	Control
3 phase by	S1 - S3
4 " "	S1 - S4
5 " "	S1 - S5

Fig 2-4 shows the form of output.

The logic can be interchanged, positive to negative logic, and visa versa. Typical circuit is shown in Fig 2-5.

EXCITATION PULSE OUTPUT

2-phase excitation

	S1	S2	S3
1	▨	▨	
2		▨	▨
3	▨		▨

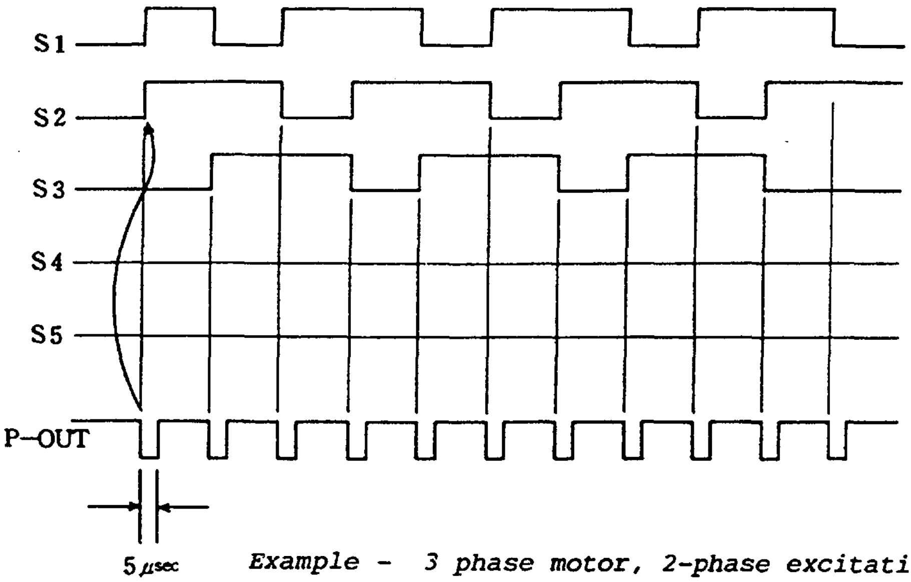
(3-phase motor)

	S1	S2	S3	S4
1	▨	▨		
2		▨	▨	
3			▨	▨
4	▨			▨

(4-phase motor)

	S1	S2	S3	S4	S5
1	▨	▨			
2		▨	▨		
3			▨	▨	
4				▨	▨
5	▨				▨

(5-phase motor)



Example - 3 phase motor, 2-phase excitation positive logic

Fig 2-4

1-2 phase excitation

	S1	S2	S3
1	▨		
2	▨	▨	
3		▨	
4		▨	▨
5			▨
6	▨		▨

(3-phase motor)

	S1	S2	S3	S4
1	▨			
2	▨	▨		
3		▨		
4		▨	▨	
5			▨	
6			▨	▨
7				▨
8	▨			▨

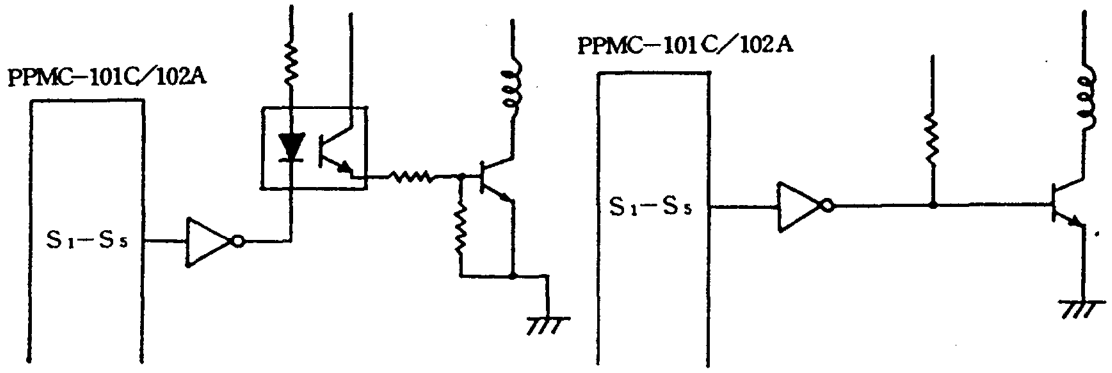
(4-phase motor)

2-3 phase excitation

	S1	S2	S3	S4	S5
1	▨	▨			
2	▨	▨	▨		
3		▨	▨		
4		▨	▨	▨	
5			▨	▨	
6			▨	▨	▨
7				▨	▨
8	▨			▨	▨
9	▨				▨
10	▨	▨			▨

(5-phase motor)

Output logic level can be switched by using positive or negative logic as shown in Fig. 2-5.



Output
Positive logic convention

Output
Negative logic convention

Fig. 2-5

2-13 $\overline{\text{INT}}$

Interrupt request is assertive 'O' when motor stops. $\overline{\text{INT}}$ can be cleared by reading the finish STATUS. This figure is not an open collector and OPEN COLLECTOR BUFFER is required as shown in Fig 2-6, when a multiple INTERUPT is expected.

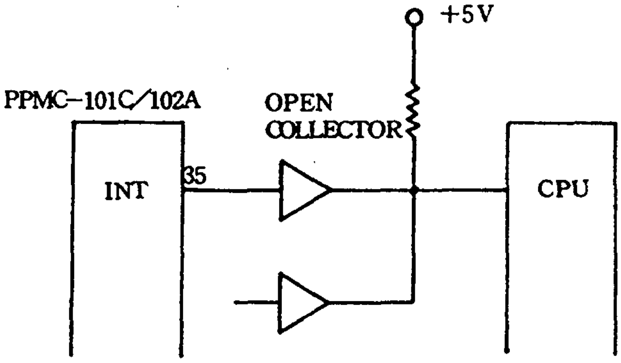


Fig 2-6

2-14 MON

When motor on input is 'O', PPMC does not output driving pulse. An example of an application is indicated in Fig 2-7, in which a thermal relay on the motor is used to protect overheating. MON input is ignored during operation of PPMC, and should be checked only before motor operation.

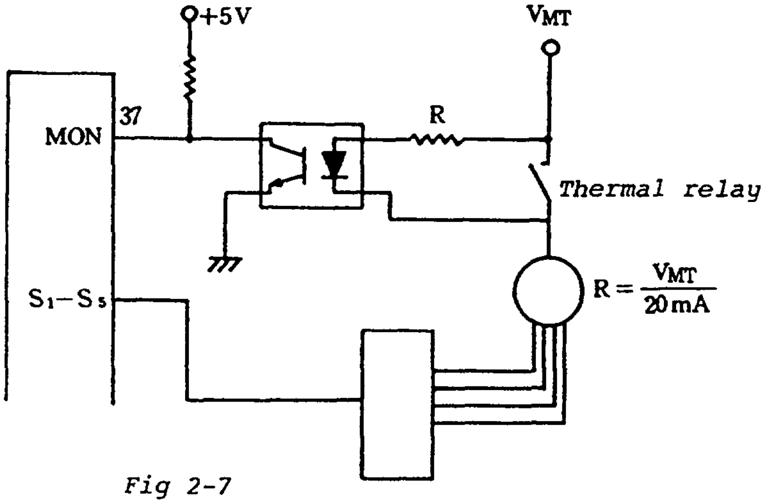


Fig 2-7

Driving circuit

2-15 CLOCK-external clock

Basic signal to control speed of the stepper motor. The speed can be controlled between 400pps and 5K pps by the 100KHz clock input to PPMC101C and between 800pps and 10K pps by 200KHz clock input to PPMC102A. The clock frequency must be below 1/45 of X1, X2 clock. For example, when 6MHz X-tal is applied for X1 and X2, external clock input must be less than 133KHz (in case 11MHz is applied, external clock must be less than 244KHz). High level of the clock pulse must be more than 500 nsec.(250 nsec for 11MHz)

3. COMMUNICATION BETWEEN PPMC AND MASTER CPU

The communication between PPMC and master CPU consists of following 3 types of modes :

(1) Initialization

It designates type of motor, method of excitation, data for acceleration/deceleration and other parameters (see page 13 for details). After power 'ON', initialization is needed before operation command. Note : Some parameters cannot be changed once it is set. Re-initialization is not possible during operation.

(2) Operation Command

8 kinds of operation commands are available for stepper motor. The length of data to follow depends on the command.

(3) Register for PPMC

After completion of (2), master CPU reads the cause of operation finish, status of input/output terminal, and the number of remaining pulse.

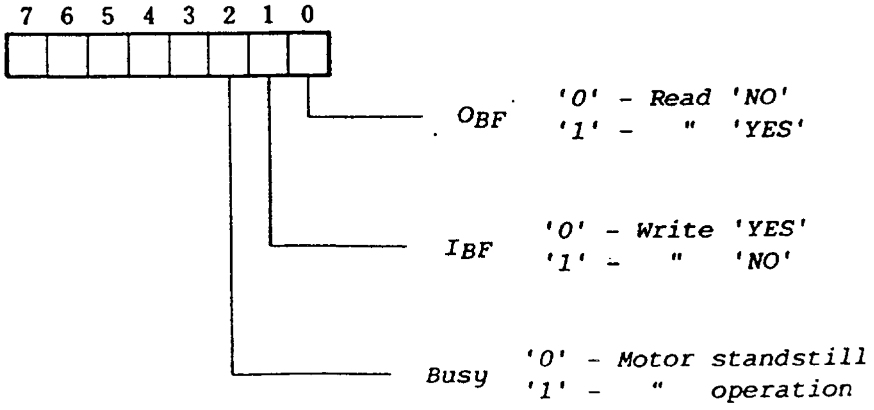
3-1 Register for PPMC

2 read only registers, and 2 write only registers are accessible to the uses.

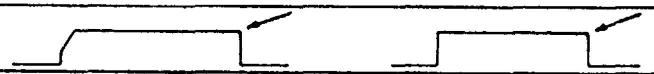
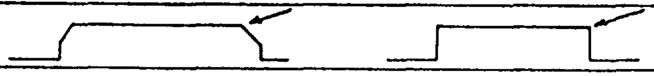
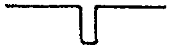
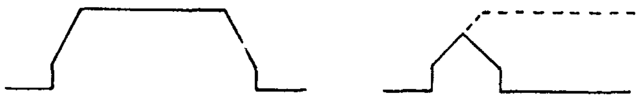
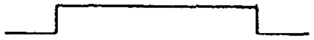
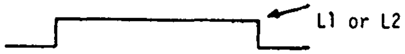
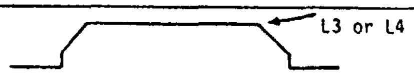
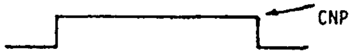
$\overline{CS}$	A_0	$\overline{RD}$	$\overline{WR}$	
1	×	×	×	DISABLE
0	0	0	1	READ DATA
0	1	0	1	READ STATUS
0	0	1	0	WRITE DATA
0	1	1	0	WRITE COMMAND

Table 3-1

3-1-1 Status Register



COMMAND TABLE

		COMMAND DATA		FUNCTION
INITIALIZATION		1 2 3 4 5	0 0 Self-starting pulse rate High speed pulse rate Accelerating/Decelerating pulse rate	
OPERATING COMMAND	Emergency Stop	1	0 1 0 0 0	
	Decelerating Stop	1	0 1 0 0 1	
	Single Step	1	0 1 0 1 0	 1 pulse
	Acceleration/ Deceleration	1 2 3 4	0 1 0 1 1 Number of operating step	
	Constant speed operation	1 2 3 4 5	0 1 1 0 0 Constant speed pulse rate Number of operating step	
	To move until the limit at constant speed	1 2	0 1 1 0 1 Constant speed pulse rate	
	To move until high speed limit	1	0 1 1 1 0	
	To move to the base point	1 2	0 1 1 1 1 Contant speed pulse rate	
READ REGISTER	Finish Data	1	1 0 0 0 0 0 0 0	To read data for reason of FINISH, etc. .... 1 byte
	Input signal	1	1 0 0 0 0 0 0 1	To read data for limit switch, etc. .... 1 byte
	Output signal	1	1 0 0 0 0 0 1 0	To read data for motor phase output and direction ... 1 byte
	Remaining step numbers	1	1 0 0 0 0 0 1 1	To read remaining number of steps .... 3 byte

3-1-1-1 OBF (Output Buffer Full)

This bit checks the status in order to read the data from PPMC. 'O' indicates that there is no data in the buffer. It can only read the data when OBF is '1'.

3-1-1-2 IBF (Input Buffer Full)

This bit checks the status in order to write commands or input data to PPMC. '1' indicates that the data is full in the buffer and therefore, it is not possible to write new data. IBF must be 'O' when you write data or give commands. If you were to write data at IBF '1' the former data would be erased.

3-1-1-3 BUSY (Motor Busy)

This flag outputs '1' during motor operation. It is only possible to input emergency stop and decelerating stop commands at that time. The IBF and BUSY bits must be checked before you input a command. This is 'O' 2.5 µsec after INT output.

3-1-2 Read Register Data

Register data can be read out after checking OBF and input of READ REGISTER COMMAND.

3-1-3 Write Command

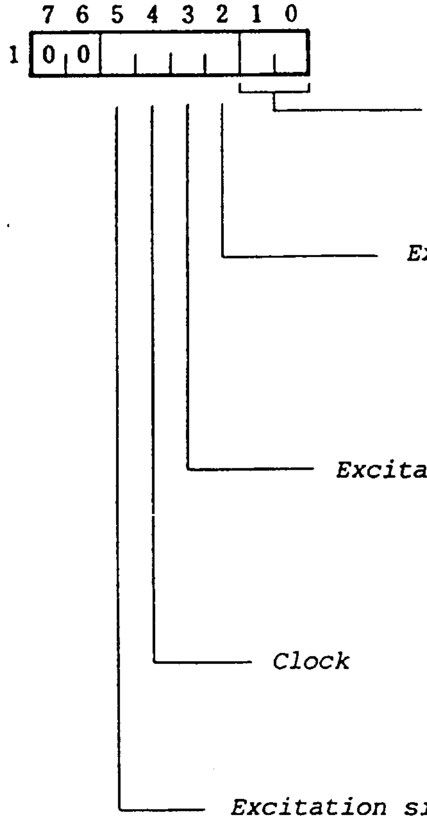
Before inputting initialization, operation command or read register command, check IBF and BUSY bit in the status register.

3-1-4 Write Data

Check IBF before writing data for pulse rate or number of steps. The order of input data must follow as indicated in command table (page 14). PPMC101C will start operating in accordance with the command 400 µsec and PPMC1Q2A does 200 µsec after the data is written.

3–2 Initialization

Initialization Command



Motor Code

0 1 ... 3 phase motor
1 0 ... 4 " "
1 1 ... 5 " "

Excitation Method

0 ... 2 phase excitation

1 ... 1-2 phase excitation
(2-3 phase excitation for 5 phase motor)

Excitation output logic level (S1, S2, S3, S4, S5)

0 ... Negative logic convention

1 ... Positive " "

Clock 0 ... Internal clock (12.5KHz ... at 6MHz X1, X2)
(22.9KHz ... at 11MHz X1, X2)

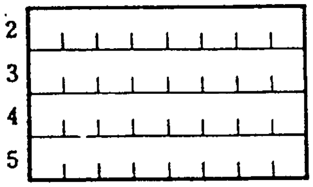
1 ... External clock (Clock signal at pin#39)

Excitation signal switching output at motor standstill

0 ... Switching 'NO'

1 ... " 'YES'

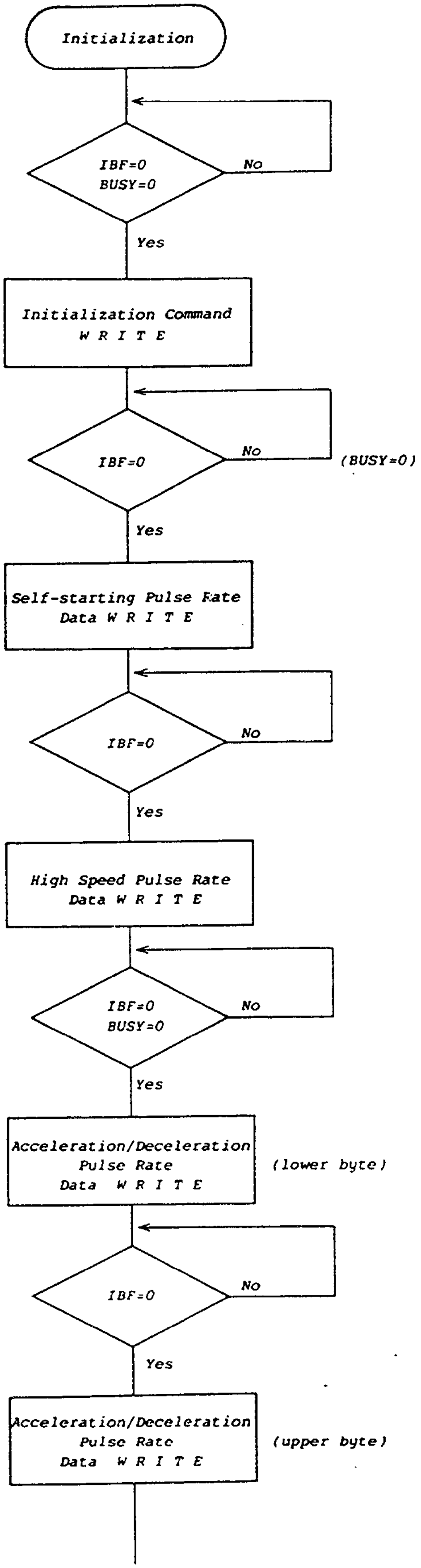
Initialization Data



self-starting pulse rate .... RA max
high speed pulse rate .... RA min
Acc/Deceleration pulse rate .... lower byte
" " " .... upper byte

Initialization command to be input in the above order (1, 2, 3, 4, 5) after power 'ON'.

It can be shown in the following flow chart.


Initialization
IBF=0
BUSY=0
No
Yes
Initialization Command
WRITE
IBF=0
No
(BUSY=0)
Yes
Self-starting Pulse Rate
Data WRITE
IBF=0
No
Yes
High Speed Pulse Rate
Data WRITE
IBF=0
BUSY=0
No
Yes
Acceleration/Deceleration
Pulse Rate
Data WRITE
(lower byte)
IBF=0
No
Yes
Acceleration/Deceleration
Pulse Rate
Data WRITE
(upper byte)

3-2-1 Initialization Command

1) Motor code

The type of motor code used must match the spec of the motor.

2) Excitation method

The excitaion method used must match the spec of the motor.

3) Logic level of excitation output (S1, S2, S3, S4, S5)

In positive logic convention, the current will be flowing through the coil of the motor when output of PPMC is high. In negative logic convention, the current will be flowing through the coil of the motor when output of PPMC is low.

Fig 3-1 shows the circuits of logic level of excitation output.

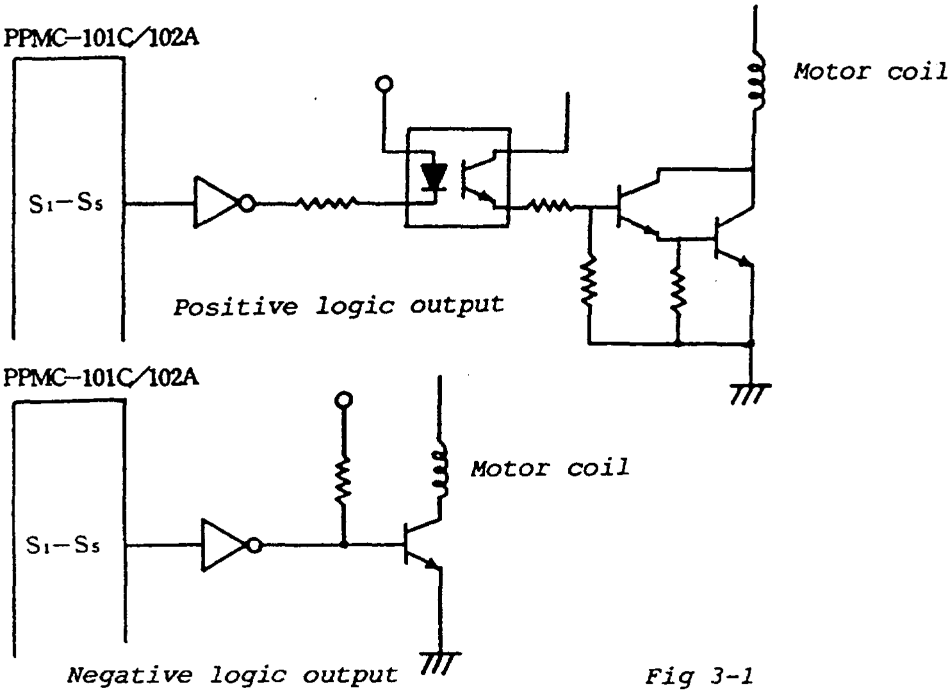


Fig 3-1

4) Clock

This is to select an internal or external basic clock for the motor. The internal clock will use frequency generated inside the PPMC. In the external clock mode the clook is provided by the input on pin#39.

It is possible for PPMC101C to control speed from 50pps to 600pps with the internal clock 12.5KHz and for PPMC102A to control speed from 100pps to 1,200pps. To control speed from 400pps to 5Kpps a 100KHz external clock should be used for PPMC101C. The external clock must be lower than 133KHz. In case of PPMC102A, speed can be controlled from 800pps to 10Kpps with 200KHz external clock which must be lower than 244KHz.

5) Excitation Signal Switching Output

Usually the current into the stepper motor remains at maximum current during stanstill. This maximum current which holds the motor can cause overheating. Bit 5 is used to prevent this type of problem by switching '1' or'0'. PPMC can switch the excitation output to minimize the excitation current. Switching frequency is about 2.2KHz for PPMC101C with a duty cycle of 30% and 4KHz for PPMC102A with a duty cycle of 35%.

About 100 msec after the output of a phase excitation signal, the motor will start operating from a standstill when "switching" is selected.

Type of motor code, excitation method and logic level of excitation pulse output cannot be changed once they are set after RESET, while clock, excitation pulse switching output and initialization data can be changed.

3-2-2 Initialization Data

3-2-2-1 Pulse Rate

PPMC applies the idea of pulse rate (RA) to decide speed of the stepper motor. The relationship between pps and RA is expressed in the following equations :

$$pps = \frac{fo}{RA}$$

fo : Basic clock frequency

RA : Pulse Rate

pps : Motor pulse per second

Basic clock is either a 12.5KHz for PPMC101C (22.9KHz for PPMC102A) clock generated inside PPMC or external clock applied to pin#39. Bit 4 (clock command bit) in initialization command is used to select either the internal/external clock.

PPS Table 3-1 shows various logical figure of RA and practical use.

RA fo		*Logical* (2 – 255)	*Practical* (20 – 255)	
Internal clock	12.5 KHz	49 Hz – 6.25KHz	49 Hz – 625Hz	*(PPMC101C)*
	22.9 KHz	89 Hz –11.4 KHz	89 Hz – 1,140Hz	*(PPMC102A)*
External clock	100 KHz	392 Hz – 50 KHz	392 Hz – 5 KHz	*(PPMC101C)*
	200 KHz	784 Hz –100 KHz	784 Hz – 10 KHz	*(PPMC102A)*

The appropriate number of pulse for acceleration/deceleration should be decided by the customer's experience, because it depends primarily on type of motor, inertia moment of load, etc. In case of large inertia moment of load, a large number of pulse for acceleration/deceleration should be selected for slow operation. PPMC can be adapted quite well to the majority of the load. With 2 byte to store the number of pulse needed for acceleration/ deceleration, 4 - 8,160 steps can be set.

Some motors have a sympathetic point where there is no torque at certain frequency as shown in Fig 3-2. In such cases, the motor has to be started with a speed lower than the sympathetic point in order to fly into a higher speed area. To minimize the time to stay on the sympathetic point, higher speed for acceleration/deceleration must be applied. It is recommended that a damper should be used to increase the inertia moment if the motor goes into the sympathetic point with a small load.

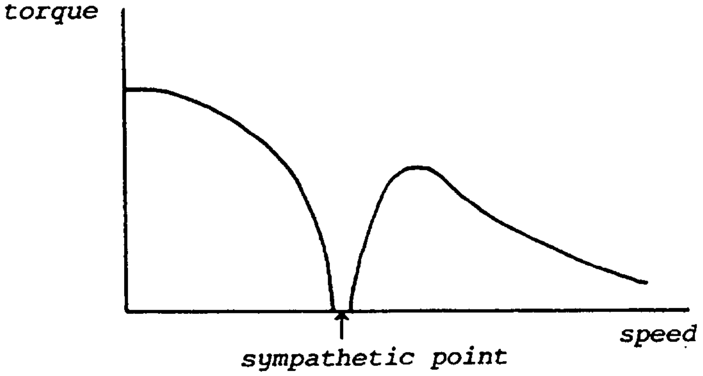


Fig 3-2

Initialization Data

Stepper motor has two types of operation as follows :

(A) Constant speed operation at lower speed of self-starting frequency.

(B) Ramp up/down operation.

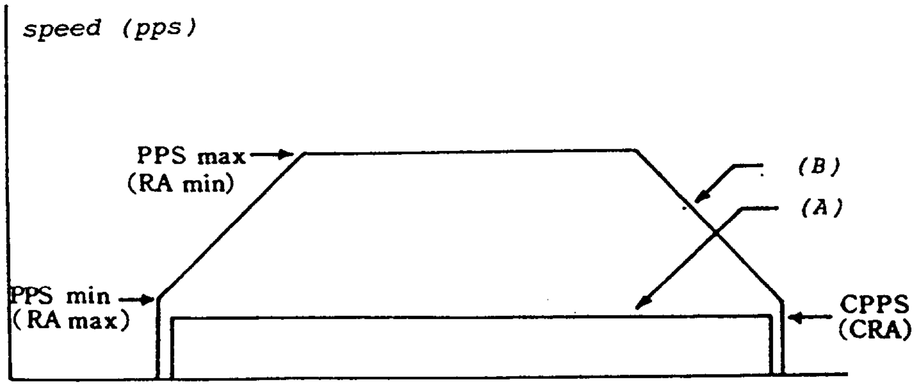


Fig 3-3

Self-starting frequency, parameters for high speed operation and acceleration/deceleration depend on the type of motor, excitation method, inertia or nature of load, etc. Relation among those are indicated below :

$$255 \geq CRA \geq RA\ max > RA\ min \geq 20$$

Pulse Rate		Symbol
At constant speed operation	.....	CRA
At self-starting	.....	RA max
At high speed operation	.....	RA min

CRA and RA max are to be set as large as possible within the above limit. The hardware limits the external clock rate to a maximum of 133KHz for PPMC101C (244KHz for PPMC102A). For example with 100KHz external clock for PPMC101C and 200KHz for PPMC102A the above relation can be converted into pps (pulse per second) as following.

PPMC101C	$392Hz \leq CPPS \leq PPS\ min < PPS\ max \leq 5KHz$
PPMC102A	$784Hz \leq CPPS \leq PPS\ min < PPS\ max \leq 10KHz$

In conclusing, with a 100KHz external clock for PPMC101C or 200KHz for PPMC102A, the stepper motor can be controlled from 400pps to 5Kpps for PPMC101C or 800pps to 10Kpps for PPMC102A. For lower speed operation, external clock frequency should be slowed down accordingly.

3-2-2-3 Aberration of motor speed

There are two major sources that cause the motor speed to deviate from the theoretical value (see Fig 3-4). The first source of error derives from the execution time of routine that outputs the excitation. A 50 µsec overhead time is needed in addition to the delay timing for the pulse output. Therefore for slow speed operations, the 50 µsec error is insignificant. The % error of the output speed will increase with an increase in motor speed.

The second source of error is the non-synchronization of the basic clock and the internal timer. A randam error corresponding to ±1 basic clock pulse in the timer counter is possible. Note the percentage error will be larger at a slower clock rate. For example at a clock rate of 20Hz, the randam error is ±5%, which may be acceptable in practical application.

Following is the curve that shows the differnce between theoretical value and practical speed at a basic clock frequency of 100KHz (PPMC101C) and 200KHz (PPMC102A). The graph shows that at RA=15, the motor speed is 5Kpps for PPMC101C and 10KHz for PPMC102A.

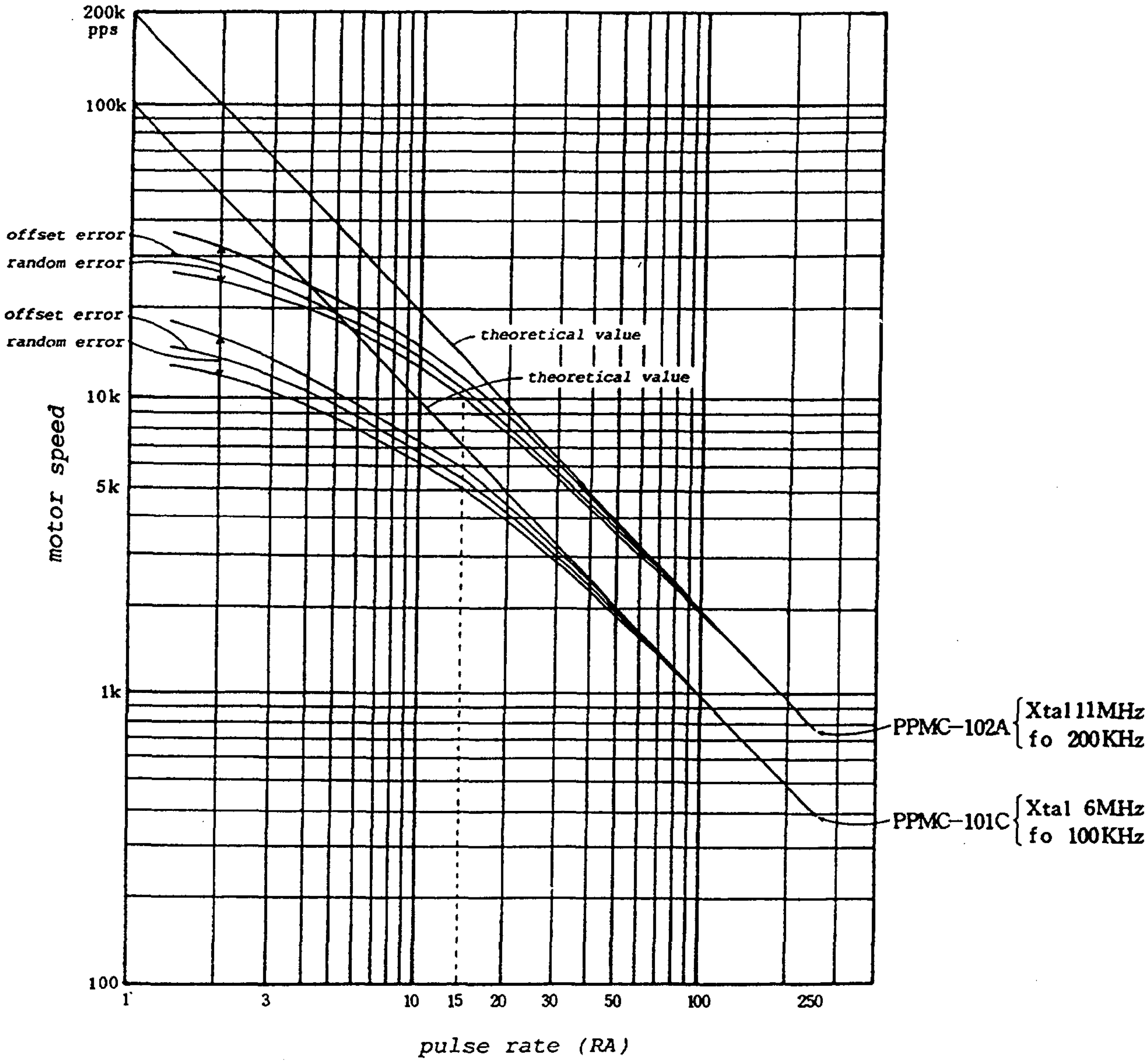
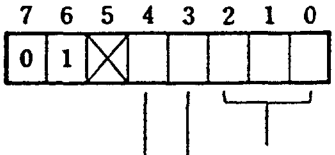


Fig 3-4

3–3 Operation Command



Operation command code

0 0 0	*Emergency stop*
0 0 1	*Decelerating stop*
0 1 0	*Single step*
0 1 1	*Acceleration/Deceleration*
1 0 0	*Constant speed operation*
1 0 1	*To move until the limit*
1 1 0	*To move until the high speed limit*
1 1 1	*To move until the base point*

Direction for rotation of motor

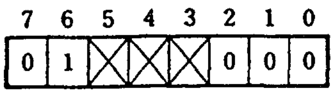
0	*C W*
1	*C C W*

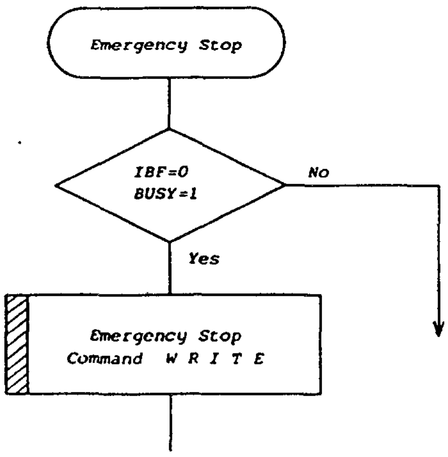
Interupt mask for FINISH signal

0	*Enable*
1	*Disable*

3-3-1 Emergency Stop

To stop rotation of motor instantaneously by inhibiting excitation output during any type of operation, whether it be acceleration/deceleration or constant speed operation. In high speed operation, the phase output stops instantaneously, but the motor will run off with inertia. Therefore, position data is no longer valid. It is necessary to reestablish the base point. During constant speed operation at self-starting frequency, motor can stop instantaneously and restarting is possible from that point by reading the number of operating pulse with the READ REGISTER COMMAND. Emergency stop requires only 1 byte operation command, and no data is necessary. Check whether the condition IBF=0, and BUSY=1 is satified before writing the emergency stop command as shown below.

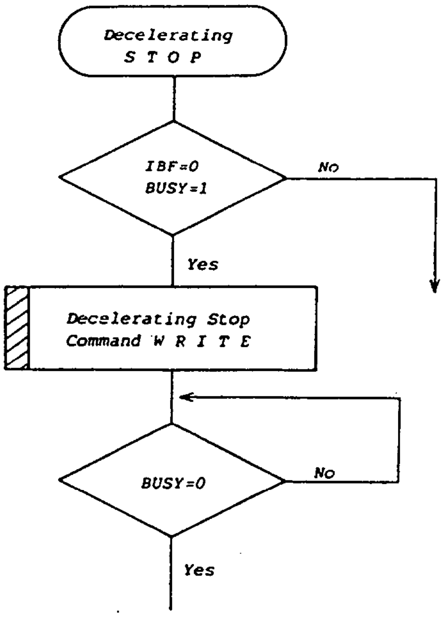




3-3-2 Decelerating Stop

When the decelerating stop command is input during acceleration/deceleration, the motor will decelerate to stop. The motor will stop instantaneously during constant speed operation at self-starting frequency, the remaining number of pulse can be read, by the READ REGISTER COMMAND and the motor can be re-started from where it stopped. The bits for direction of motor rotation and FINISH INTERRUPT become assertive when the motor stops. Refer to the following flow chart for proper sequence of operation.

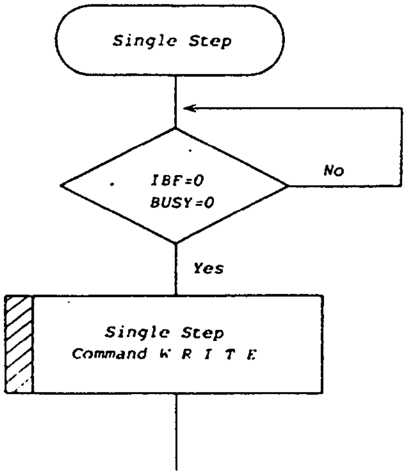
7	6	5	4	3	2	1	0
0	1	☒	☒	☒	0	0	1



3-3-3 Single Step

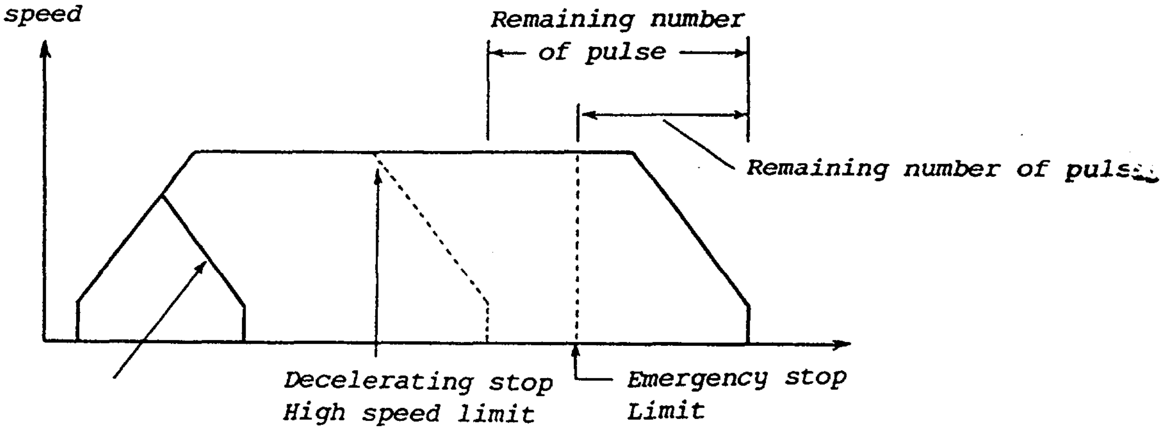
This is the command to move the stepper motor at a single step. It is useable when the master CPU needs to find out its position by itself. When this command is released continuously, timing must be controlled by the master CPU. All command modes are effective, this command consists of a single byte. No other data is necessary. Refer to the following flow chart for proper sequence of operation.

7	6	5	4	3	2	1	0
0	1	☒			0	1	0



3-3-4 Acceleration/Deceleration

This command for acceleration/deceleration in accordance with the data at the time of initialization. In addition to the command itself this operation requires 3 bytes of data, which store the total number of pulses to be output. For triangle operation, total number of pulse must be smaller than two times the acceleration/deceleration pulse number. The limiting switch input L3, L4 can be used to trigger the deceleration (see Fig 2-2) and L1 and L2 can be used to stop the motor. Note that irrelevant signals from L1-L4 will be ignored. For example in Fig 2-2, if the carrier is moving CW, the signals from L2 and L4 will be ignored.



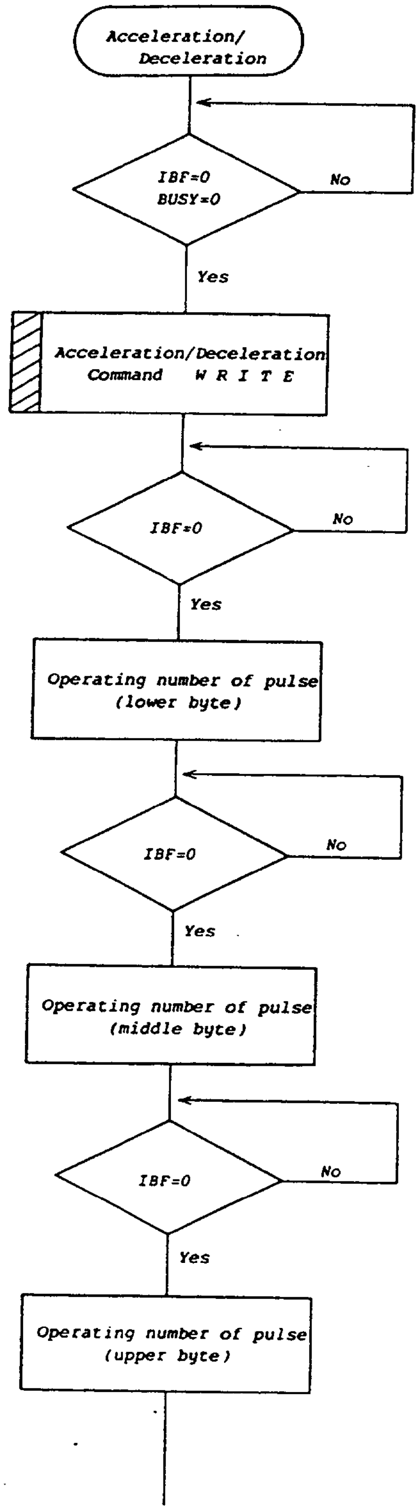
The number of pulse (step number requested for operation - 1) can be got with 3 bytes.
FFFFFF (Hexa decimal) input moves 16,777,216 steps which is the maximum number of steps to move at a time.

- Example -

To move 1,000 steps, 1,000 - 1 = 999 should be converted into Hexa decimal (0003E7) for input. Data must be input from the lower byte.

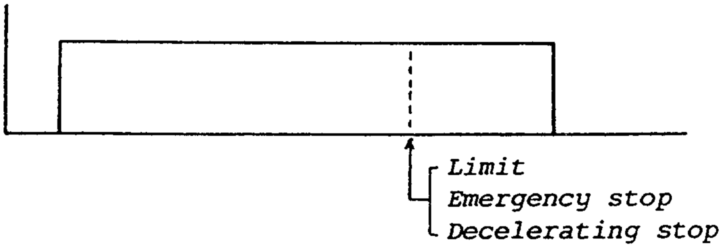
7	6	5	4	3	2	1	0
0	1	☒			0	1	1

	7	6	5	4	3	2	1	0	*Operating number of pulse*
(E7)	1	1	1	0	0	1	1	1	*(lower byte)*
(03)	0	0	0	0	0	0	1	1	*(middle byte)*
(00)	0	0	0	0	0	0	0	0	*(upper byte)*


Acceleration/ Deceleration
IBF=0
BUSY=0
No
Yes
Acceleration/Deceleration Command WRITE
IBF=0
No
Yes
Operating number of pulse (lower byte)
IBF=0
No
Yes
Operating number of pulse (middle byte)
IBF=0
No
Yes
Operating number of pulse (upper byte)

3-3-5 Constant Speed Operation

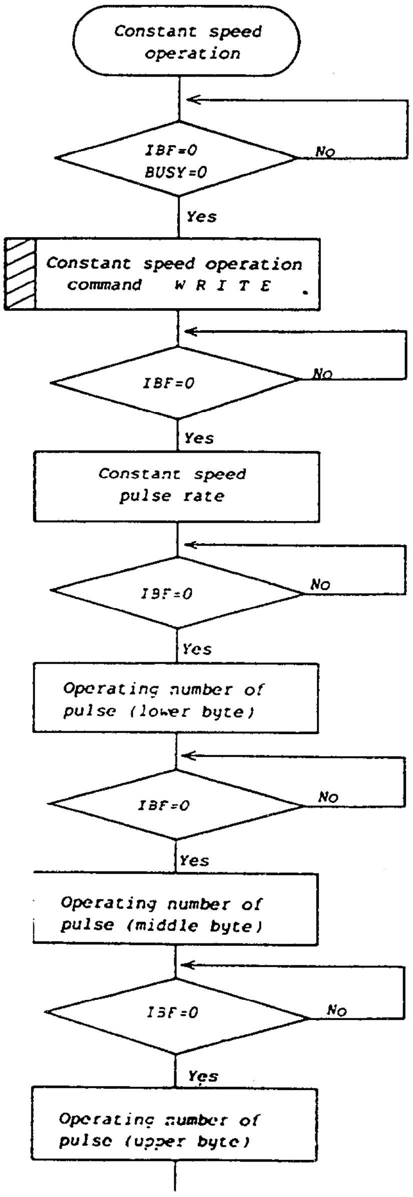
This command requires one byte of data for constant speed pulse rate as well as 3 bytes of data for the total number of pulse. The proper sequence of execution is shown in the following flow chart. The command causes the motor to rotate at a constant speed up to the designated distance. The speed is set by the pulse rate data, which has to be within the self-starting frequency of the motor. L1-L4 limit switch input can be used to decelerate and stop the motor. The READ REGISTER COMMAND can be used to readout the remaining number of pulse and the cause for stopping.



Refer to the following flow chart for proper sequence of operation.

7	6	5	4	3	2	1	0
0	1	☒			1	0	0

Operating number of pulse	7	6	5	4	3	2	1	0	
									constant speed pulse rate
(lower byte)									
(middle byte)									
(upper byte)									

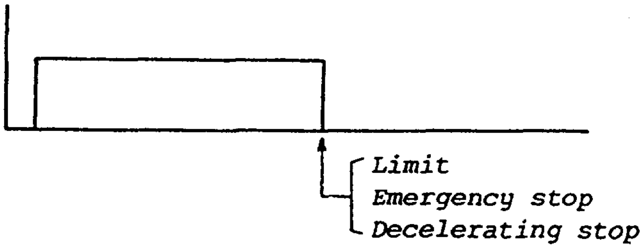

Constant speed operation
IBF=0
BUSY=0
No
Yes
Constant speed operation command WRITE
IBF=0
No
Yes
Constant speed pulse rate
IBF=0
No
Yes
Operating number of pulse (lower byte)
IBF=0
No
Yes
Operating number of pulse (middle byte)
IBF=0
No
Yes
Operating number of pulse (upper byte)

3-3-6 To Move At Constant Speed Until Limit Switch

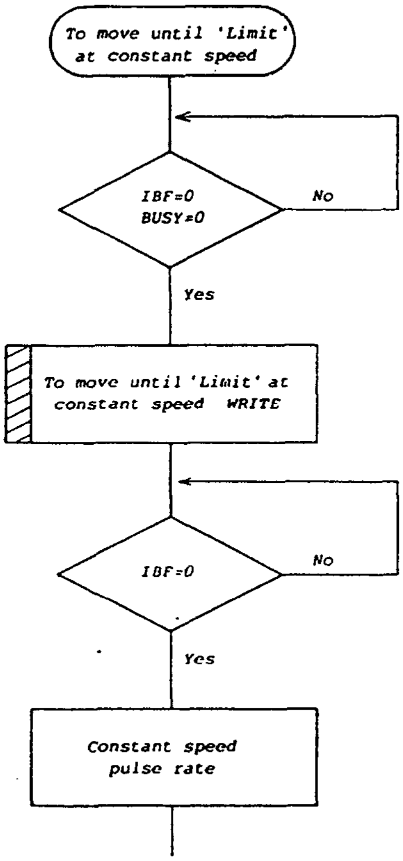
This mode of operation is similar to the previous one except that the data indicating the number of pulse is omitted. The motor will keep on moving until a signal from a limit switch (L1 or L2) is received. The signal from irrelevant L1 or L4 is ignored. Limit input in the same operating direction means as follows :

Operation	*Input*	
C W	*L1*	*limit input*
C C W	*L2*	*" "*

During a CW operation, L2 input will be ignored.



This command is normally used right after POWER ON or to re-start after motor run off.


To move until 'Limit' at constant speed
IBF=0
BUSY=0
No
Yes
To move until 'Limit' at constant speed WRITE
IBF=0
No
Yes
Constant speed pulse rate

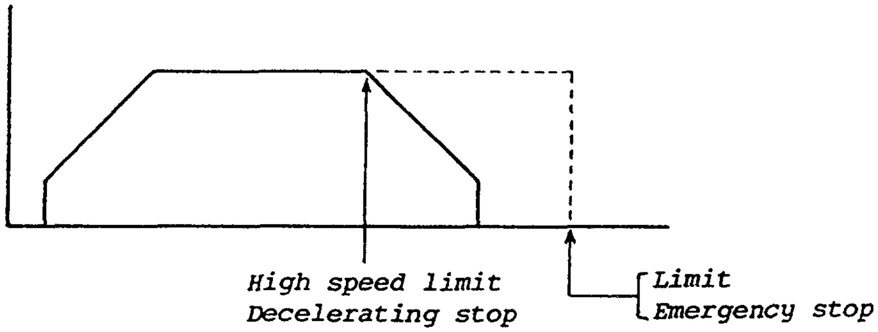
7	6	5	4	3	2	1	0
0	1	☒			1	0	1

7	6	5	4	3	2	1	0

constant speed pulse rate

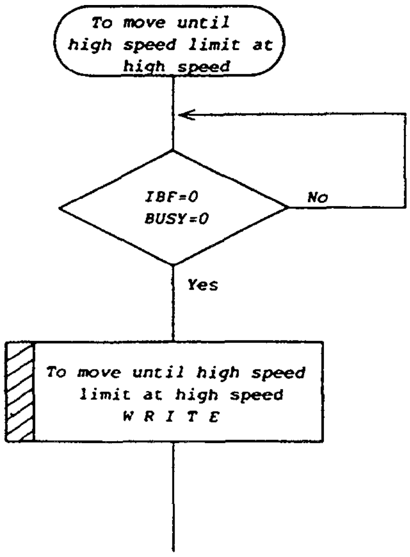
3-3-7 To Move At High Speed Until High Speed Operation Limit

This is the command to accelerate for high speed operation in accordance with the data at the time of initialization. Under this command the motor will rotate at high speed until a limiting signal is received. Then it will decelerate to stop according to the number of decelerating pulse. Once deceleration begins, it continues to decelerate even if limit input turns out '1'. Limit switch L1 for CW and L2 for CCW rotation can force the motor to stop, but inertia may cause the motor to overrun the desired stopping point.



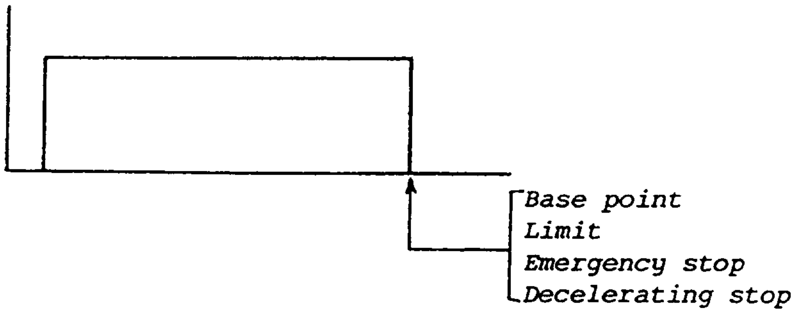
This command is also used right after POWER ON or to re-start motor after run-off. The decision whether to use this command or the command 'to move at constant speed until limit switch' (3-3-6) depends on the distance, time, accuracy of position, etc.

7	6	5	4	3	2	1	0
0	1	☒			1	1	0


To move until high speed limit at high speed
IBF=0
BUSY=0
No
Yes
To move until high speed limit at high speed
WRITE

3-3-8 To Move To Base Point

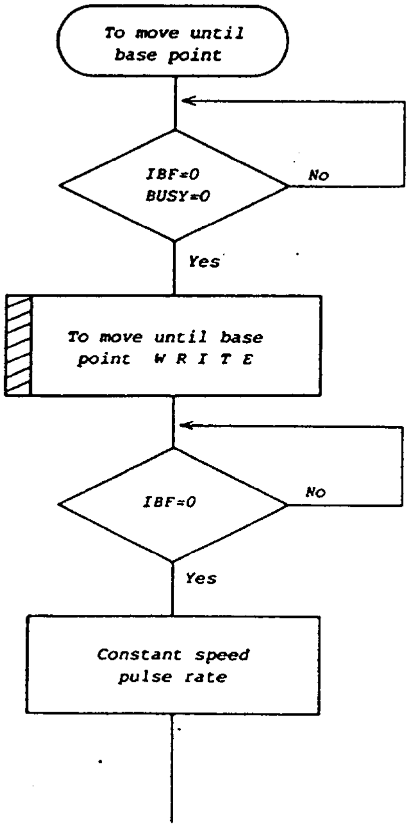
In this mode of operation, the motor rotates at constant speed until a $\overline{CNP}$ signal is detected. The motor can also be stopped by the relevant limit input L1-L4, emergency stop or decelerating stop.



7	6	5	4	3	2	1	0
0	1	☒			1	1	1

7	6	5	4	3	2	1	0

constant speed pulse rate



3-4 READ REGISTER

READ REGISTER COMMAND is used to read three kinds of status and a 3 byte data during standstill of motor. The proper format of the command is indicated below :

7	6	5	4	3	2	1	0
1	0	0	0	0	0		

Register code

0 0 ..... FINISH status

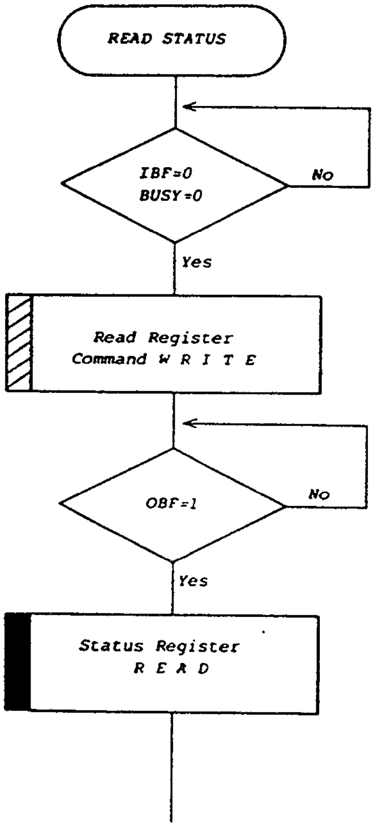
0 1 ..... Input signal status

1 0 ..... Output signal status

1 1 ..... Remaining number of pulse

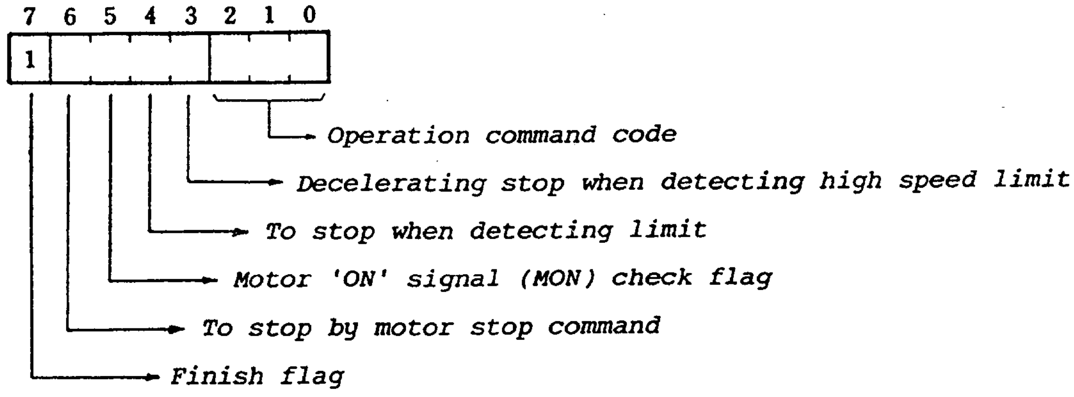
3-4-1 READ FINISH STATUS

The proper sequence to read the various status is shown in the following flow chart.



3-4-1-1 FINISH STATUS

The finish status register contains the following information :


7 6 5 4 3 2 1 0
1
Operation command code
Decelerating stop when detecting high speed limit
To stop when detecting limit
Motor 'ON' signal (MON) check flag
To stop by motor stop command
Finish flag

The lower 3 bytes contain the operation command code.

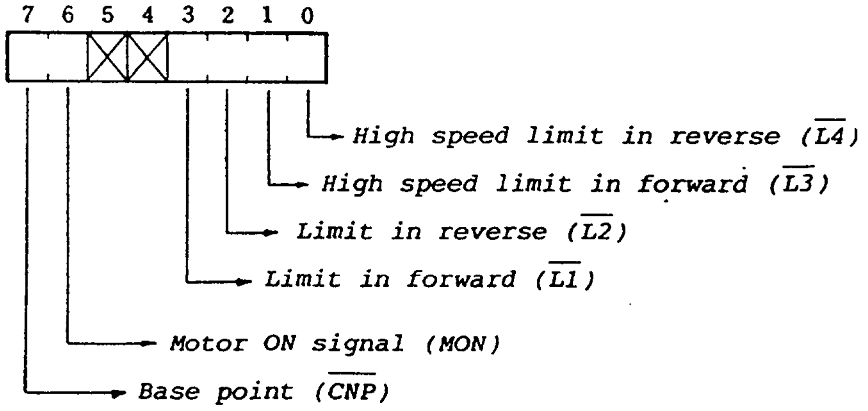
Bit 3 or 4 goes up to '1' after high speed limit ($\overline{L3}$, $\overline{L4}$) or limit ($\overline{L1}$, $\overline{L2}$) is input for motor stop. Bit 5 shows '1' when motor cannot operate with 'motor on signal' (MON) = 0.

A '1' in bit 6 indicates either an emergency stop or decelerating stop. When all number of pulses for acceleration/deceleration and constant speed operation are completely consumed to stop, all bits from 3 to 6 turns to '0'.

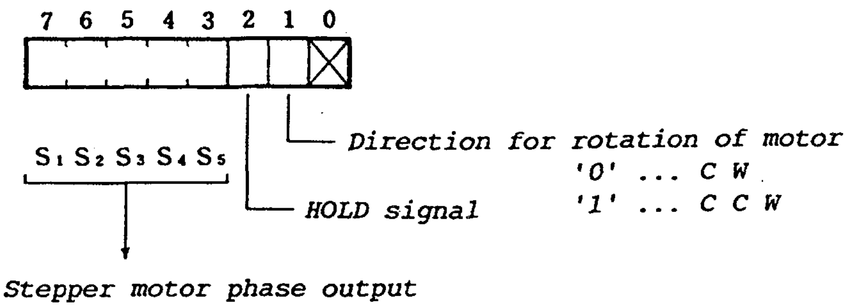
In the absence of finish interrupt mask, $\overline{INT}$ becomes assertive at the end of the operation, INT signal can be cleared to '1' by reading finish status and released.

3-4-1-2 INPUT signal

The input signal register reflects the state of various inputs shown below at the point where the motor stops.


7 6 5 4 3 2 1 0
High speed limit in reverse ($\overline{L4}$)
High speed limit in forward ($\overline{L3}$)
Limit in reverse ($\overline{L2}$)
Limit in forward ($\overline{L1}$)
Motor ON signal (MON)
Base point ($\overline{CNP}$)

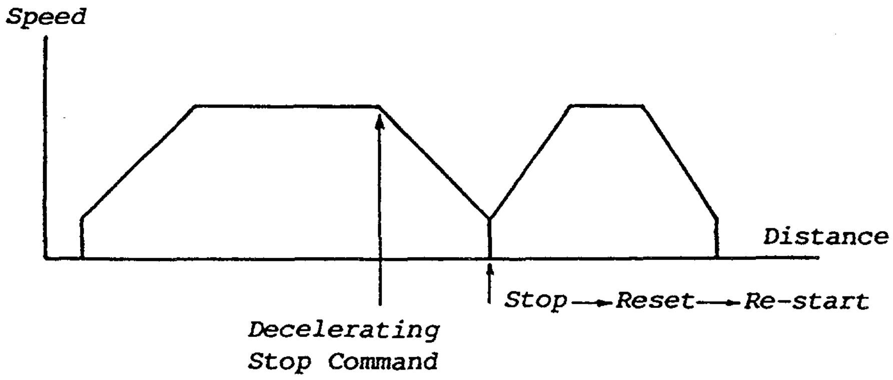
3-4-1-3 OUTPUT Signal



Phase outputs for stepper motor can be checked by bit 3 to 7

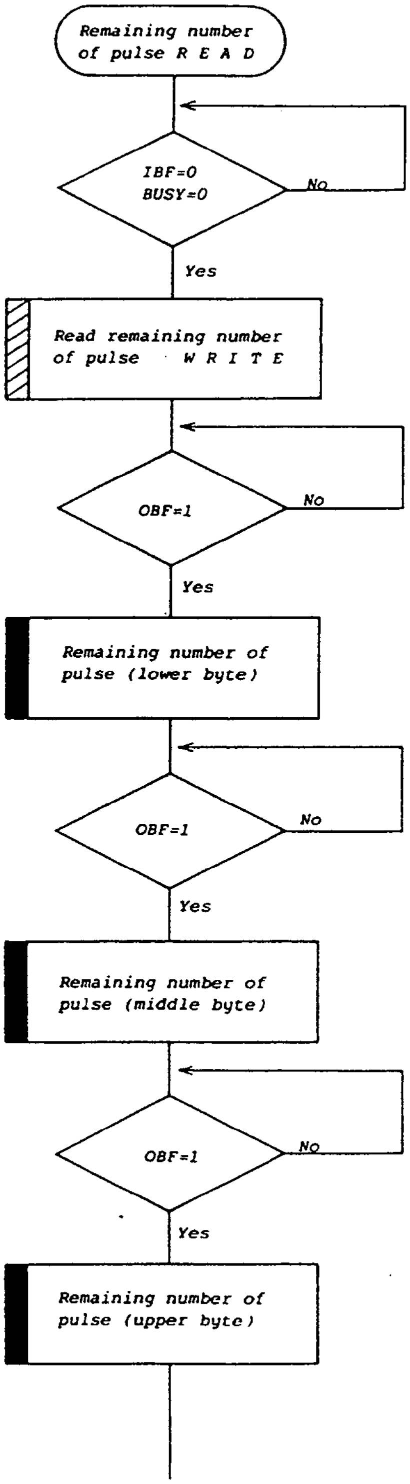
3-4-2 Remaining number of pulse

When the motor is stopped by limit switch or stop command during acceleration/deceleration or constant speed operation, the remaining number of pulse can be read by using this mode. If it is decided to finish the operation after the stop command, the original command and the remaining number of pulse can be input again to restart the mode.



The data becomes '0' when the operation has been successfully terminated.

The following flow chart shows the proper sequence of programming in order to read the data from PPMC.


Remaining number of pulse READ
IBF=0
BUSY=0
No
Yes
Read remaining number of pulse WRITE
OBF=1
No
Yes
Remaining number of pulse (lower byte)
OBF=1
No
Yes
Remaining number of pulse (middle byte)
OBF=1
No
Yes
Remaining number of pulse (upper byte)

7	6	5	4	3	2	1	0	
1	0	0	0	0	0	1	1	*Command*

Remaining number of pulse

7	6	5	4	3	2	1	0	
								(lower byte)
								(middle byte)
								(upper byte)

❑

❑

❑

APPENDIX B

SOURCES FOR STEPPER MOTORS & ASSEMBLIES

B.1 GENERAL INFORMATION

Keithley MetraByte stocks a standard stepper motor (Model STEP-MOT1). Specifications are provided in Appendix C. Although these have been selected from very popular types, they may not be suitable for your application. This Appendix details vendors who specialize in stepper motors and associated mechanical components.

In the course of assembling a stepper motor system, you may need to procure stepper motors, lead screws, X-Y tables, shaft encoders etc. Many vendors provide this type of equipment and the following list has been compiled to assist you in locating components. Inclusion of a vendor in this list does not imply that MetraByte Corporation endorses the vendor or is in any way responsible for the performance of its equipment. Likewise the list is not complete, and omission of any vendor is purely coincidental. This list is provided purely for your convenience and is not exhaustive.

B.2 STEPPER MOTORS

1. Superior Electric Company
   383 Middle Street
   Bristol CT 06010 Phone: 203-582-9561

2. B & B Motor & Control Corp.
   Apple Hill Commons
   Burlington CT. 06013 Phone: 203-673-7151

3. Bodine Electric Company
   2500 W. Bradley Place
   Chicago IL. 60618 Phone: 312-478-3515

4. Litton Clifton Precision
   P.O. Box 160
   Murphy N.C. 28906-0160 Phone: 704-837-5115

5. Oriental Motor U.S.A. Corp.
   2701 Plaza Del Amo Suite 702
   Torrance, CA 90503 Phone: 213-515-2264

B.3 X-Y TABLES, SLIDES, LEADSCREWS ETC.

1. Daedal Inc.
   P.O. Box G
   Sandy Hill Road
   Harrison City PA. 15636 Phone: 1-800-245-6903
   (in PA) 412-744-4451

2. New England Affiliated Technologies Inc.
   620 Essex Street
   Lawrence MA. 01841 Phone: 617-685-4900

3. B & B Motor & Control Corp.
   Apple Hill Commons
   Burlington CT. 06013 Phone: 203-673-7151

4. Klinger Scientific Corporation
   110-120 Jamaica Avenue
   Richmond Hill NY. 11418 Phone: 718-846-3700

B.4 TRANSLATORS, STEPPER DRIVERS

1. Superior Electric Company
   383 Middle Street
   Bristol CT. 06010 Phone: 203-582-9561

2. Bodine Electric Company
   2500 W. Bradley Place
   Chicago IL. 60618 Phone: 312-478-3515

3. Oriental Motor U.S.A. Corp.
   369 Passaic Avenue
   Fairfield N.J. 07006 Phone: 201-882-0480

B.5 TOOTHED BELTS, GEARS, DRIVE COMPONENTS ETC.

1. Precision Industrial Components Corporation
   P.O. Box 1004
   Benson Road
   Middlebury CT. 06762 Phone: 1-800-243-6125
   in CT: 203-758-8272

B.6 OTHER SOURCES

If you are on a tight budget, Herbach & Rademan usually carry a range of new and used stepper motors and other mechanical odds and ends. Ask for their catalog, their address is

Herbach & Rademan Corporation
401 E. Erie Avenue
Philadelphia PA. 19134 Phone: 215-426-1708

■ ■ ■

APPENDIX C

STEP-MOT1 SPECIFICATIONS

Electrical Specifications

Nominal DC Volts	5.0 volts
Winding Resistance	5.0 ohms at 25 deg. C.
Rated Current	1.0 amps per winding
Winding Inductance	10.4 millihenries
Winding Type	4-phase, 6-lead unipolar
Time for Single Step	2.5ms with 24VDC drive

Mechanical Specifications

Step angle	1.8 degrees full step.
Angle Accuracy	5%
Holding Torque	53 oz.-in. min. (2 windings energized)
Residual Torque	1.25 ox.-in. min.
Rotor Inertia	0.04 lb.-in^2
Torque/Inertia Ratio	32,000 typ.
Shaft Diameter	0.25 inch
Radial Load	15 lbs. max.
Axial Load	25 lbs. max.
Weight 20 oz.	

■ ■ ■

❑

❑

❑

APPENDIX D

MSTEP-3 & M3-DRIVE SPECIFICATIONS

D.1 POWER CONSUMPTION

+5v supply	385mA typ; 500mA max.
+12v supply	not used
-12v supply	not used
-5v supply	not used

D.2 MSTEP-5 SPECIFICATIONS

Stepper channels	3 (individually programmable)
Maximum step count	+/-16,777,215
Maximum step rate	15,000 p.p.s.
Acceleration/deceleration ramping	automatic trapezoidal programmable start, run & ramping rates.
Limit switch inputs (active low, open collector, TTL or mechanical switch to ground)	5 per channel (end of travel, high speed & base point)
Translator Drive	CCW/CW and positive going pulse.
Phase (winding) drives	TTL compatible signals for 3,4 or 5 phase motor windings.
Phase drive logic polarity	programmable
Phase drive sequence	programmable full or half step.
Power chopping at standstill	programmable (reduces motor heating)

D.3 M3-DRIVE SPECIFICATIONS

Maximum motor voltage	36V.
Maximum winding current	2A.
Driver type	Bipolar chopping.
Maximum step rate	10,000 steps per second, depending on motor and load.
Driver transistor on resistance	

D.4 LOGIC OUTPUTS

All outputs	TTL compatible 0.4v max output low voltage 2.4v min output high voltage
PULSE, output pulse width	68 ns ±17us
PULSE, DIR sink current	24mA at 0.5V
PULSE, DIR source current	-24mA at 2.4V
All other outputs:	
Sink current	6mA at 0.3V
Source current	-6mA at 2.4V

D.5 LOGIC INPUTS

All inputs	TTL/CMOS compatible 0.8v max input low voltage 2.0v min input high voltage
Pullups	All inputs have internal 10K pullups to +5v.

D.6 POWER OUTPUTS

IBM P.C. buss supplies	None.

D.7 PHYSICAL & ENVIRONMENTAL SPECIFICATIONS

Operating temperature range	0 to 50 deg.C.
Storage temperature range	-20 to 70 deg.C.
Humidity	95% non-condensing
Weight	8oz. (230 gm.)

D.8 CONNECTOR PINOUT

A AXIS LIMIT 4	(I)	1	26	(I)	A AXIS LIMIT 3
A AXIS MOTOR ON (MON)	(I)	2	27	(I)	A AXIS LIMIT 1
A AXIS HOME (BASE)	(I)	3	28	(I)	A AXIS LIMIT 2
A AXIS DIRECTION	(I)	4	29	(O)	A AXIS HOLD
A AXIS STEP PULSE	(O)	5	30	(O)	A AXIS AUX. BIT
B AXIS LIMIT 1	(I)	6	31		GROUND
B AXIS LIMIT 3	(I)	7	32	(I)	B AXIS LIMIT 2
B AXIS MOTOR ON (MON)	(I)	8	33	(I)	B AXIS LIMIT 4
B AXIS HOLD	(O)	9	34	(I)	B AXIS HOME (BASE)
B AXIS DIRECTION	(O)	10	35	(O)	B AXIS AUX. BIT
PORT A BIT 7	(I/O)	11	36	(O)	B AXIS STEP PULSE
PORT A BIT 6	(I/O)	12	37		GROUND
PORT A BIT 4	(I/O)	13	38	(I/O)	PORT A BIT 5
PORT A BIT 0	(I/O)	14	39	(I/O)	PORT A BIT 1
PORT A BIT 2	(I/O)	15	40	(I/O)	PORT A BIT 3
PORT B BIT 7	(I/O)	16	41	(I/O)	PORT B BIT 6
PORT B BIT 5	(I/O)	17	42	(I/O)	PORT B BIT 4
PORT B BIT 1	(I/O)	18	43	(I/O)	PORT B BIT 0
PORT B BIT 2	(I/O)	19	44	(I/O)	PORT B BIT 3
C AXIS LIMIT 1	(I)	20	45	(I)	C AXIS LIMIT 2
C AXIS LIMIT 3	(I)	21	46	(I)	C AXIS LIMIT 4
C AXIS MOTOR ON (MON)	(I)	22	47	(I)	C AXIS HOME (BASE)
C AXIS AUX. BIT	(O)	23	48	(O)	C AXIS HOLD
C AXIS STEP PULSE	(O)	24	49	(O)	C AXIS DIRECTION
EXTERNAL CLOCK INPUT	(I)	25	50		GROUND

Rear view of 50-pin connector, J1; (I) = input, (O) = output, (I/O) = input/output.

Keying blocks are present between pins 3 & 5 and 47 & 49. Connector attached to board is 3M (Scotchflex) # 3433-5303. Mating half for ribbon (insulation displacement cable) is 3M # 3425-6050

C AXIS S1	1	20	NO CONNECTION
C AXIS S2	2	19	NO CONNECTION
C AXIS S3	3	18	C AXIS S4
B AXIS S4	4	17	C AXIS S5
B AXIS S2	5	16	GROUND
B AXIS S1	6	15	B AXIS S5
A AXIS S1	7	14	B AXIS S3
A AXIS S2	8	13	GROUND
A AXIS S3	9	12	GROUND
A AXIS S4	10	11	A AXIS S5

Top view of the P2 Connector (at top of Board).

D.8 SIGNAL DESCRIPTIONS

Phase outputs S1-S5 — TTL outputs that provide signals for motor winding excitation. Logic polarity and step sequence (full/half) are programmable.

CCW/CW — Direction signal output 0 = clockwise 1 = counterclockwise.

PULSE OUT — Pulse output corresponding to steps. 5 microsecond negative going pulses. CCW/CW and PULSE can be used to drive a translator.

HOLD ACK. — If switching at standstill is enabled, HOLD ACK. goes high 3 milliseconds after motor stops otherwise it is always low.

L1 & L2 limit inputs — Active low overtravel limit inputs. Motor will stop immediately on encountering either of these limits. L1 - clockwise limit L2 - counterclockwise limit

L3 & L4 limit inputs — Active low high speed limit inputs. Motor will perform a decelerating stop at either of these limits if executing an accelerate or proceed to high speed limit command. L3 - clockwise limit L4 - counterclockwise limit

Base point, CNP input — Active low home or reference point limit switch input. Motor will stop at base point if executing a proceed to base point command.

Motor control input — Motor control input must be high for controller to execute commands. It can be used to monitor power on the stepper motor.