PCF-M3

User Guide

for the

Keithley MetraByte

PCF-M3

PASCAL, C, & FORTRAN

Callable Drivers

For The

MSTEP-3

Revision A - February 1992

Part Number: 24420

Keithley Instruments, Inc. Data Acquisition Division

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

Warranty Information

All products manufactured by Keithley Instruments, Inc. Data Acquisition Division 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 the manufacturer, be repaired or replaced. This warranty does not apply to products damaged by improper use.

Warning

Keithley Instruments, Inc. Data Acquisition Division 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 Instruments, Inc. Data Acquisition Division is believed to be accurate and reliable. However, the Keithley Instruments, Inc. Data Acquisition Division 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 the Keithley Instruments, Inc. Data Acquisition Division.

Note:

Keithley MetraByte™ is a trademark of Keithley Instruments, Inc. Data Acquisition Division.

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

1.1 OVERVIEW

The PCF-M3 is a software package for programmers using Pascal, Turbo PASCAL, C, FORTRAN, and QuickBASIC to write motion control routines (referred to herein as *Application Code*) for the MSTEP-3. The Distribution Software for this package is normally supplied on 5.25" low-density diskettes but is also available (upon request) on 3.5" diskette(s). Contents of the package include the following:

- MSTEP-3 Drivers for each of the supported languages
- Driver Source Modules for creating new Drivers
- Miscellaneous documentation (.DOC) files
- Example program files in all supported languages

1.2 SUPPORTED LANGUAGES

The MSTEP-3 supports all memory modules of the following languages:

- Microsoft C (V4.0 - 6.0)
- Microsoft Quick C (V1.0 - 2.0)
- Microsoft Pascal (V3.0 - 4.0)
- Microsoft FORTRAN (V4.0, 4.1)
- Microsoft QuickBASIC (V4.0 and higher)
- Borland Turbo Pascal (V3.0 - 5.0)
- Borland Turbo C (V1.0 - 2.0)
- GW, COMPAQ, and IBM BASIC (V2.0 and higher)

1.3 COPYING DISTRIBUTION SOFTWARE

As soon as possible, make a working copy of your Distribution Software. You may put the working copy on diskettes or on the PC Hard Drive. In either case, making a working copy allows you to store your original software in a safe place as a backup.

To make a working copy of your Distribution Software, you will use the DOS COPY or DISKCOPY function according to one of the instructions in the following two subsections.

To Copy Distribution Software To Another Diskette

Note that the *source* diskette is the diskette containing your Distribution Software; the *target* diskette is the diskette you copy to. Before you start, be sure to have one (or more,as needed) formatted diskettes on hand to serve as target diskettes.

First, place your Distribution Software diskette in your PC's A Drive and log to that drive by typing **A:** . Then, use one of the following instructions to copy the diskette files.

- If your PC has just one diskette drive (Drive A), type **COPY *.* B:** (in a single-drive PC, Drive A also serves as Drive B) and follow the instructions on the screen.

  If you prefer to use the DOS *DISKCOPY* function, instead of *COPY* , you will type **DISKCOPY A: A:** and follow instructions on the screen. This alternative is faster, but requires access to DISKCOPY.COM, in your DOS files.

- If your PC has two diskette drives (Drive A and Drive B), type **COPY *.* B:** (the same as above) and follow the instructions on the screen.

  If you prefer to use the DOS *DISKCOPY* function, instead of *COPY* , you will type **DISKCOPY A: B:** and follow instructions on the screen. This alternative is faster, but requires access to DISKCOPY.COM, in your DOS files.

To Copy Distribution Software To The PC Hard Drive

Before copying Distribution Software to a hard drive, make a directory on the hard drive to contain the files. While the directory name is your choice, the following instructions use *PCFM3* .

1. After making a directory named *PCFM3* , place your Distribution Software diskette in your PC's A Drive and log to that drive by typing **A:** .
2. Then, type **COPY *.*** *path***PCFM3** , where *path* is the drive designation and DOS path (if needed) to the PCFM3 directory.

When you finish copying your Distribution Software, store it in a safe place (away from heat, humidty, and dust) for possible future use as a backup.

1.4 GENERATING AN APPLICATION PROGRAM

In the Distribution Software, the example program for the language you are using provides most of the information you need to start your own MSTEP-3-based Application Program. The overall procedure for a typical executable program, however, is as follows:

1. Write your Application Code using a text editor or the language environment.
2. Compile your program.
3. Link the compiled program to a Driver (from the Distribution Software) suited to the language of your Application Code.

This procedure gives you an executable Application Program, ready to test. Repeat all three steps as you modify/fix this program.

1.5 THIS MANUAL

Chapter 1 of this manual is introductory material.

Chapter 2 presents information on the MSTEP-3 Drivers required for the supported languages. Since the Drivers support the full series of MSTEP-3 Mode Calls, Chapter 2 also lists and briefly describes the Mode Calls. And since the Drivers may not be perfectly suited to your particular applications, Chapter 2 discusses the Driver Source Modules, which are the source-code files you may use for creating new Drivers. Finally, the chapter includes instructions for creating new Drivers.

Chapter 3 presents brief instructions and examples for using the Drivers with your Application Programs.

■ ■ ■

CHAPTER 2

DRIVER INFORMATION

2.1 OVERVIEW

When you write a program for your own MSTEP-3 application, your program is referred to herein as the *Application Code*. You have a choice of writing this Code in BASIC, QuickBASIC, PASCAL, Turbo PASCAL, C, or FORTRAN. You then compile your Application Code and link the resulting program with a *Driver*. The linking process develops the *Application Program*, which is the program giving you software control of your hardware.

The Driver you link with your Application Code must be suited to the language used for the Code. For example, if you write your Application Code in C, your must link it with a Driver suited to C.

The Distribution Software contains Drivers for BASIC, QuickBASIC, PASCAL, Turbo PASCAL, C, and FORTRAN. The Distribution Software also contains the *Driver Source Modules*, which are the Assembly Language source files provided for the purpose of allowing you to create new Drivers customized to your particular needs.

Section 2.2 of this chapter lists and describes the Driver Source Modules, with which you may create new Drivers. Section 2.3 lists and describes the Drivers available in the Distribution Software. Section 2.4 lists the Mode Calls supported by the Drivers. Section 2.5 instructs you on how to make calls from your Application Code. The final section (Section 2.6) instructs you on how to use the Driver Source Modules to create new Drivers.

2.2 DRIVER SOURCE MODULES

The following two Driver Source Modules are the essential building blocks for creating a MSTEP-3 Driver in any language:

MSTEP.ASM	Core of the driver.
MSTEPPCF.ASM	Driver interface module for PASCAL, C, FORTRAN, and QuickBASIC.

As mentioned earlier, these two modules are available in your Distribution Software. Also available in the Distribution Software is *TURBOPAS.ASM*, which is a Driver Source Module available strictly for Turbo PASCAL. TURBOPAS.ASM actually has a source equivalent to all the above two modules.

For instructions on using these modules to create Drivers, refer to Section 2.6.

2.3 DRIVERS

As a convenience, your Distribution Software contains Drivers for PASCAL, Turbo PASCAL, C, FORTRAN, BASIC, and QuickBASIC. You must link the appropriate Driver with your Application Code; choose the Driver that matches the language used for your Application Code. Available

Drivers are as follows:

MSTEP.LIB:	Driver for Pascal, C, FORTRAN, and stand-alone QuickBASIC programs.
MSTEP.BIN:	Driver for BASIC(A).
MSTEP.QLB:	Driver for the QuickBASIC Integrated Development Environment (Ver. 4.0-4.5).
MSTEPX.QLB:	Driver for the QuickBASIC Extended Environment (Ver. 7.0).
TURBOPAS.OBJ:	Driver for TURBO Pascal.

2.4 MODE CALLS

This list briefly describes the Mode Calls supported by the MSTEP-3 driver software. More detailed explanations of each Mode are available in the main text of the MSTEP-3 User Guide.

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

Refer to the MSTEP-3 User Guide for details of each Mode. It is essential that you perform a channel initialization (MODE 15) on each channel separately before selecting any other Mode (0-14).

2.5 CALLING THE DRIVER

In your Application Code, you write a call the MSTEP-3 driver through a single label that corresponds to the language used for your Code and to the memory model used for compiling. These labels are the *Call Labels* . MSTEP-3 Call Labels and their corresponding Drivers are as follows:

MSTEP.LIB:

mscs_mstep	For Calls from Microsoft C, Small Model
mscm_mstep	For Calls from Microsoft C, Medium Model
mscl_mstep	For Calls from Microsoft C, Large Model
tcs_mstep	For Calls from TURBO C, Small Model
tcm_mstep	For Calls from TURBO C, Medium Model
tcl_mstep	For Calls from TURBO C, Large Model
msp_mstep	For Calls from Microsoft Pascal
fmstep	For Calls from Microsoft FORTRAN
qbmstep	For Calls from Microsoft QuickBASIC

TURBOPAS.OBJ:

tp_mstep	For Calls from TURBO Pascal

MSTEP.BIN:

mstep	For Calls from BASIC(A)

Regardless of the language/model you are using, with each call to a label you must specify three input parameters, as follows:

MODE — A 16-bit integer containing the number of the mode to be executed by the MSTEP-3 driver.

PARAM — An array of 16-bit integers containing a variable number of mode-dependent arguments required for the successful execution of the mode.

STPNUM — Step Number, which is a long integer that specifies the direction and number of steps to travel. The sign indicates direction (+ = clockwise, - = counter-clockwise). Not all modes use StpNum data, but this parameter must be included in a Call.

FLAG — A 16-bit integer quantity that contains a number representing any error code reported by the MSTEP-3 driver. (See Chapter 4 for error-code definitions.)

The following is code fragment (in C) on how to declare and use the call parameters.

```
    :
int         Mode=15, Flag;
unsigned    Param[15];
long        StpNum=0;
    :

Param[0]  = 0            /* Select Axis A */
Param[1]  = 125;         /* Start-up rate, 49 pps */
Param[2]  = 20;          /* High-speed run rate */
Param[3]  = 200;         /* Acceleration/deceleration pulse count */
Param[4]  = 2;           /* 4 phase */
Param[5]  = 0;           /* Full step */
Param[6]  = 1;           /* Inverted S1-5 outputs */
Param[7]  = 0;           /* Internal clock */
Param[8]  = 1;           /* Switching off at standstill */
Param[9]  = 16;          /* "On Time" of duty cycle */
Param[10] = 32;          /* "Off Time" of duty cycle */
Param[11] = 1;           /* Port A Direction; 0 = input, 1 = output */
Param[12] = 1;           /* Port B Direction; 0 = input, 1 = output */
Param[13] = 0;           /* Auxiliary bit logic; 0 = low, 1 = high */
Param[14] = 0x300;       /* Base Address */

if (Flag != 0)
  printf ("**** Error %d detected in Mode 15", Flag);
    :
```

Refer to Chapter 3 for additional details on how to declare and use these variables in other languages.

2.6 CREATING NEW DRIVERS

General

While the Drivers available to you in the Distribution Software (see Section 2.3) support all the Call Modes described in Section 2.4, they may not suit your particular application. You may remedy this problem by creating a new version of the desired Driver. This section provides the information necessary to create a new Driver for BASIC, QuickBASIC, PASCAL, Turbo PASCAL, C, and FORTRAN.

Note that to create a new version of a Driver, your working directory (generally, the directory containing the Distribution Software) must contain the Driver Source Modules (Section 2.2) and the following development tools:

MASM.EXE	Microsoft Assembler
LINK.EXE	Microsoft Linker
LIB.EXE	Microsoft Librarian

Other utilities will be specified as needed in the instructions of the subsections that follow.

Also, note that in the MASM compile commands you use to create a new Driver, you must define the two symbols *BIN* and *MSTEP*. These definitions use the */D* option for BASIC, QuickBASIC, PASCAL, C, and FORTRAN; for Turbo PASCAL, no symbol definition is required. These symbol definitions are as follows:

BIN = 1: Compile for BASIC(A) Driver. Usage example: **/DBIN=1**.

BIN = 0: Compile for non-BASIC(A) Driver (PASCAL, C, FORTRAN, and QuickBASIC). Usage example: **/DBIN=0**.

WARNING

The manufacturer does not provide technical support for user modications of the driver source code.

The MSTEP.BIN Driver For BASIC(A)

To create a MSTEP.BIN Driver, you must have access to the following utilities:

EXE2BIN.EXE	A Microsoft .EXE-to-.COM file conversion utility (generally available in DOS files).
MAKEBIN.EXE	A .COM-to-.BIN file-conversion utility (supplied in the MSTEP-3 Distribution Software).

Then, use the following commands:

```
MASM /DBIN=1 MSTEP.ASM;
MASM /DBIN=1 MSTEPPCF.ASM
LINK MSTEPPCF + MSTEP, MSTEP,,;
EXE2BIN MSTEP.EXE MSTEP.COM
MAKEBIN MSTEP.COM
```

All five steps must be successful. Note that the linking operation generates the warning:

LINK : Warning L4021 : no stack segment

Disregard this warning; it is irrelevent.

The TURBOPAS.OBJ Driver For Turbo PASCAL

To create a TURBOPAS.OBJ Driver, you must have access to the following utility:

TASM.EXE TURBO Assembler

Then, use the command **TASM TURBOPAS.ASM**.

The MSTEP.QLB Driver For The QuickBASIC Integrated Environment (V4.5)

To create the MSTEP.QLB Driver, you must have access to the utility *BQLB45.LIB*, which is the QuickBASIC Integrated Environment Library. Then use the following commands:

```
MASM /DBIN=0 MSTEP.ASM;
MASM /DBIN=0 MSTEPPCF.ASM;
LINK /q MSTEP+MSTEPPCF, MSTEP,,BQLB45;
```

The MSTEP.LIB Driver For A Stand-alone QuickBASIC (V4.5) Program

To create the MSTEP.LIB file, you must have access to *MASM* (the Microsoft Assembler) and *LIB.EXE* (the Microsft Library Manager). Then, use the following commands:

```
MASM /DBIN=0 MSTEP.ASM;
MASM /DBIN=0 MSTEPPCF.ASM;
LIB MSTEP-+MSTEP;
LIB MSTEP-+MSTEPPCF;
```

The MSTEPX.QLB Driver For The QuickBASIC Extended Environment (V7.0)

To create a QLB library compatible with QuickBASIC Version 7.0, follow the procedure described for QuickBASIC Version 4.5. However, link with *QBXQLB.LIB* , instead of *BQLB45.LIB* , as follows:

```
LINK /q MSTEP+MSTEPPCF, MSTEPX,,QBXQLB;
```

Note that the output file (from the linker) is renamed *MSTEPX.QLB* to avoid incompatibilities with QuickBASIC 4.5.

The MSTEP.LIB Driver For PASCAL, C, & FORTRAN

When your Application Code is in PASCAL, C, or FORTRAN, use the MSTEP.LIB Driver to compile your Application Program.

To create the MSTEP.LIB file, you must have access to *MASM* (the Microsoft Assembler) and *LIB.EXE* (the Microsft Library Manager). Use the following commands:

```
MASM /DBIN=0 MSTEP;
MASM /DBIN=0 MSTEPPCF;
LIB MSTEP-+MSTEP;
LIB MSTEP-+MSTEPPCF;
```

■ ■ ■

CHAPTER 3

DRIVER USAGE

3.1 OVERVIEW

This chapter presents example of Driver usage in each of the supported languages. Refer to the appropriate section below for details on performing the Mode Calls from the language you are using. The language sections contain brief code fragments for illustration. More information is also available in the example programs (Distribution Software).

3.2 MICROSOFT C/TURBO C

The C Language, with its large run-time libraries and full pointer-manipulation support, provides the most flexible environment for writing Application Code that fully utilizes your MSTEP-3 product.

Function Prototypes

In your Application Code, declare one of the following function prototypes, depending on the Memory Model you will use:

```
mscs_mstep(int *, unsigned *, long *, int *);      /* MS C Small Model */
mscm_mstep(int *, unsigned *, long *, int *);      /* MS C Medium Model */
mscl_mstep(int *, unsigned *, long *, int *);      /* MS C Large Model */
tcs_mstep(int *, unsigned *, long *, int *);       /* Turbo C Small Model */
tcm_mstep(int *, unsigned *, long *, int *);       /* Turbo C Medium Model*/
tcl_mstep(int *, unsigned *, long *, int *);       /* Turbo C Large Model */
```

You have the option of preceding these function prototypes with the C keyword *extern* . Note that each prototype contains a Call Label that corresponds to the Memory Model to be used during compilation.

The Call Parameters

Declare the Mode Call parameters as follows:

```
int Mode;
unsigned Param[15];
int Flag;
long stpnum;
```

The Param[] array index values are 0 thru 14, inclusive.

An Example

To call MODE 15 of the MSTEP-3 Driver from an MS C Medium Model program, your commands would be

```
    :
int         Mode=15;        /* Initialize MSTEP-3 */
int         Flag=0;
unsigned    Param[15];
long        StpNum=0;
    :

Param[0]  = 0             /* Select Axis A */
Param[1]  = 125;          /* Start-up rate, 49 pps */
Param[2]  = 20;           /* High-speed run rate */
Param[3]  = 200;          /* Acceleration/deceleration pulse count */
Param[4]  = 2;            /* 4 phase */
Param[5]  = 0;            /* Full step */
Param[6]  = 1;            /* Inverted S1-5 outputs */
Param[7]  = 0;            /* Internal clock */
Param[8]  = 1;            /* Switching off at standstill */
Param[9]  = 16;           /* "On Time" of duty cycle */
Param[10] = 32;           /* "Off Time" of duty cycle */
Param[11] = 1;            /* Port A Direction; 0 = input, 1 = output */
Param[12] = 1;            /* Port B Direction; 0 = input, 1 = output */
Param[13] = 0;            /* Auxiliary bit logic; 0 = low, 1 = high */
Param[14] = 0x300;        /* Base Address */

mscm_mstep(&Mode, Param, &StpNum, &Flag);
if (Flag !=0)
       printf("**** Error %d detected in Mode 15", Flag);
    :
    :
    :
```

Note that specifying *Param* in the Call statment is the same as *&Param[0]*.

Linking To The Driver

After compiling your C Application Code, link it to the MSTEP.LIB Driver (for Call Label *mscm_mstep*) as follows:

LINK *your-program*,,,MSTEP.LIB;

If no error reports occur, you will obtain your Application Program *your-program.EXE* , ready to test. If the Linker reports errors such as Unresolved External(s), determine whether you linked to MSTEP.LIB correctly.

NOTE: Be sure to use the correct Call Label for the Memory Model you are using.

3.3 MICROSOFT PASCAL

Function Prototypes

In your Application Code, declare the following function prototype:

```
MSP_MSTEP(VAR Mode:integer;VAR Param:PArray;VAR StpNum:integer4;VAR Flag:integer): external;
```

The Call Parameters

Declare the Mode Call parameters as follows:

```
TYPE
  PArray = array [0..14] of word ;

VAR
  Param     : PArray;                (* MODE PARAM ARRAY *)
  Mode,Flag : integer;               (* MODE CALL VARIABLES *)
  StpNum    : integer;               (* STEP NUMBER *)
```

The Param[] array index values are 0 thru 14, inclusive. The index value starts at 0.

An Example

To call MODE 0 of the MSTEP Driver from MS Pascal Application Code,

```
    :
Mode      :=15;          /* Initialize MSTEP-3 */
Flag      :=0;
StpNum    :=300;
    :

Param[0]  := 0            /* Select Axis A */
Param[1]  := 125;         /* Start-up rate, 49 pps */
Param[2]  := 20;          /* High-speed run rate */
Param[3]  := 200;         /* Acceleration/deceleration pulse count */
Param[4]  := 2;           /* 4 phase */
Param[5]  := 0;           /* Full step */
Param[6]  := 1;           /* Inverted S1-5 outputs */
Param[7]  := 0;           /* Internal clock */
Param[8]  := 1;           /* Switching off at standstill */
Param[9]  := 16;          /* "On Time" of duty cycle */
Param[10] := 32;          /* "Off Time" of duty cycle */
Param[11] := 1;           /* Port A Direction; 0 = input, 1 = output */
Param[12] := 1;           /* Port B Direction; 0 = input, 1 = output */
Param[13] := 0;           /* Auxiliary bit logic; 0 = low, 1 = high */
Param[14] := 768;         /* Base Address */

MSP_MSTEP(Mode,Param,StpNum,Flag);
IF (Flag > 0) THEN
BEGIN
       WRITELN('INITIALIZATION ERROR ! ');
       RETURN;
END;
    :
    :
    :
```

Linking To The Driver

After compiling your MS PASCAL Application Code, link it to the MSTEP.LIB Driver (for Call Label *MSP_MSTEP*), as follows:

```
LINK your-program,,,MSTEP.LIB;
```

If no error reports occur, you will obtain your Application Program *your-program.EXE* , ready to test. If the Linker reports errors such as Unresolved External(s), determine whether you linked to MSTEP.LIB correctly.

3.4 BORLAND TURBO PASCAL

The Call Label

The Call Label *TP_MSTEP* is usable from any Turbo Pascal program; declare this label in your Application Code as follows:

```
TP_MSTEP(VAR Mode:integer;VAR Param:PArray;VAR StpNum:longint;VAR Flag:integer):external;
```

The Call Parameters

Declare the Mode Call parameters as follows:

```
TYPE
     PArray = array [0..14] of word;
VAR
     Param           : PArray;       (* MODE PARAM ARRAY *)
     Mode,Flag       : integer;      (* MODE CALL VARIABLES *)
     StpNum          : longint;      (* STEP NUMBER *)
```

The Param[] array index values are 0 thru 14, inclusive. The index values start at 0.

An Example:

To call Mode 15 of the MSTEP Driver from Turbo Pascal Application Code:

```
    :
Mode      :=15;          /* Initialize MSTEP-3 */
Flag      :=0;
StpNum    :=300;
    :

Param[0]  := 0            /* Select Axis A */
Param[1]  := 125;         /* Start-up rate, 49 pps */
Param[2]  := 20;          /* High-speed run rate */
Param[3]  := 200;         /* Acceleration/deceleration pulse count */
Param[4]  := 2;           /* 4 phase */
Param[5]  := 0;           /* Full step */
Param[6]  := 1;           /* Inverted S1-5 outputs */
Param[7]  := 0;           /* Internal clock */
```

```
Param[8]  := 1;          /* Switching off at standstill */
Param[9]  := 16;         /* "On Time" of duty cycle */
Param[10] := 32;         /* "Off Time" of duty cycle */
Param[11] := 1;          /* Port A Direction; 0 = input, 1 = output */
Param[12] := 1;          /* Port B Direction; 0 = input, 1 = output */
Param[13] := 0;          /* Auxiliary bit logic; 0 = low, 1 = high */
Param[14] := 768;        /* Base Address */

TP_MSTEP(Mode,Param,StpNum,Flag);
IF (Flag > 0) THEN
BEGIN
       WRITELN('INITIALIZATION ERROR ! ');
       RETURN;
END;
    :
    :
    :
```

Linking To The Driver

The Turbo Pascal Driver is *TURBODAS.OBJ*. This file is linked into your program using the $L Compiler Directive. Include this command at the beginning of your progam as follows:

```
{$L TURBOPAS}
```

Once included, you are ready to compile your program with the command

```
TPC your-program
```

3.5 MICROSOFT FORTRAN

The Software Driver Call Label

The call label *fmstep* is usable from any MS FORTRAN Application Code; no prototype declaration of the label is required.

The Mode Call Parameters

Declare the Mode Call parameters as follows:

```
integer*2 Param(15)          !Parameter Array
integer*2 mode               !Mode number
integer*2 flag               !Return error flag
integer*4 stpnum             !Step Number
```

Note that by default, FORTRAN array index values begin at 1. The latest versions of FORTRAN, however, allow you override this default to start at Index Value 0. Refer to your FORTRAN Manuals for more detail.

An Example

To call MODE 0 of the Driver from Microsoft FORTRAN Application Code,

```
C       Mode=15:  Initialize MSTEP-3

        Mode=15
        Flag=0
        StpNum=300

        Param(1)    = 0
        Param(2)    = 125
        Param(3)    = 20
        Param(4)    = 200
        Param(5)    = 2
        Param(6)    = 0
        Param(7)    = 1
        Param(8)    = 0
        Param(9)    = 1
        Param(10)   = 16
        Param(11)   = 32
        Param(12)   = 1
        Param(13)   = 1
        Param(14)   = 0
        Param(15)   = 768

          call fmstep(Mode, Param(1),StpNum, Flag)
        if (Flag .GT. 0) then
            WRITE(*,50) Flag
50          format(1x,'INITIALIZATION ERROR : ',i2)
            stop
        endif
```

Linking To The Driver

After compiling your FORTRAN Application Code, link it to the MSTEP.LIB Driver (for the Call Label *fmstep*) as follows:

```
LINK your-program,,,MSTEP.LIB;
```

If no error reports occur, you will obtain your Application Program *your-program.EXE* , ready to test. If the Linker reports errors such as Unresolved External(s), determine whether you linked to MSTEP.LIB correctly.

3.6 MICROSOFT QUICKBASIC

The Call Label

You must declare the Call Label in your Application Code. Make this declaration by inserting the following command at the beginning of your Code:

```
DECLARE SUB QBMSTEP (MD%, BYVAL PARAMS%, STP&, FLAG%)
```

The Call Parameters

Declare the Mode Call parameter array D%(14) as follows:

```
DIM D%(14)
COMMON SHARED D%()
```

The term *COMMON SHARED* allows the use other modules and subroutines in this array.

An Example

To initialize your MSTEP-3 board, use MODE 0 as follows:

```
    :
MD% = 15                        'Initialize MSTEP-3
STP& = 300
FLAG% = 0

D%(0) = 0                 'Select Axis A
D%(1) = 125;              'Start-up rate, 49 pps
D%(2) = 20;               'High-speed run rate
D%(3) = 200;              'Acceleration/deceleration pulse count
D%(4) = 2;                '4 phase
D%(5) = 0;                'Full step
D%(6) = 1;                'Inverted S1-5 outputs
D%(7) = 0;                'Internal clock
D%(8) = 1;                'Switching off at standstill
D%(9) = 16;               '"On Time" of duty cycle
D%(10) = 32;              '"Off Time" of duty cycle
D%(11) = 1;               'Port A Direction; 0 = input, 1 = output
D%(12) = 1;               'Port B Direction; 0 = input, 1 = output
D%(13) = 0;               'Auxiliary bit logic; 0 = low, 1 = high
D%(14) = 768;             'Base Address

PRINT "Current Acceleration/Deceleration step rate = "; D%(3)
CALL QBMSTEP (MD%, VARPTR(D%(0)), STP&, FLAG%)
IF FLAG% <> 0 THEN PRINT "INITIALIZATION ERROR ! "
    :
```

Linking To The Driver

The QuickBASIC interface consists three separate Drivers:

MSTEP.QLB — Use when you load the QuickBASIC Enviroment Version 4.5 and you plan to run your program from within the Environment (no EXE envolved here). Use the /L switch to load this Quick Library into QuickBASIC:

QB /L MSTEP *your-program*

MSTEPX.QLB — This is identical to MSTEP.QLB except that it is designed for QuickBASIC Extended Environment Version 7.0 (QBX). Use the /L switch to load this Quick Library into QuickBASIC:

QBX /L MSTEPX *your-program*

■ ■ ■

CHAPTER 4

SUMMARY OF ERROR CODES

In general, the Error Flag is the parameter that returns any reports of error conditions. This flag is an integer type (16 bits) and contains the Error Code number.

The following list contains Error Code definitions and suggested actions.

Error 1: Motor busy.

Meaning:	This error is generated by MODEs 2, 3, 4, 5, 6, 7, and 8; it indicates that the motor is under operation.
Action:	Call MODE 0 to stop the motor and try again.

Error 2: Driver not initialized on Channel A.

Meaning:	Any mode can return this error; it indicates that the driver must be initialized for Channel A prior to this MODE.
Action:	Call MODE 15 to initialize Channel A before the MODE in which this error occurred.

Error 3: Driver not initialized on Channel B.

Meaning:	Any mode can return this error; it indicates that the driver must be initialized for Channel B prior to this MODE.
Action:	Call MODE 15 to initialize Channel B before the MODE in which this error occurred.

Error 4: Driver not initialized on Channel C.

Meaning:	Any mode can return this error; it indicates that the driver must be initialized for Channel C prior to this MODE.
Action:	Call MODE 15 to initialize Channel C before the MODE in which this error occurred.

Error 5: Mode number < 0 or > 15.

Meaning:	Any mode can return this error; it indicates that the mode number is not allowed.
Action:	Check that the mode number is within 0 thru 15, inclusive.

Error 6: Hardware error.

Meaning:	MODEs 0, 1, 2, 3, 5, 6, 7, 8, and 15 return this error; it indicates that the PPMC motor controller is not taking commands from the MODE in which this error occurred.
Action:	Call MODE 15 to re-initialize.

Error 7: Step count out of range ±16,777,215.

Meaning:	MODE 3, and 4 whenever thespecified Step Count is out of range.
Action:	Check the mode number to be within the range - 16,777,215 thru + 16,777,215 inclusive.

Error 8: Motor already standstill.

Meaning:	MODEs 0 and 1 return this error whenever the motor is already standstill.
Action:	No action is needed.

Error 9: Motor switching time at standstill; does not set.

Meaning:	MODE 15 returns this error; it indicates that MODE 15 fails to set up the "EXCITATION SIGNAL SWITCHING" timing.
Action:	Check the "ON" (Param [9]) and "OFF" (Param [10]) time for switching frequency in MODE 15 initialization parameters.

Error 10: Error in range of Param[0].

Meaning:	This error may be returned by any MODE; it indicates that the first parameter (Param [0]) passed to the driver is out of range.
Action:	Check the range of the parameter.

Error 11: Error in range of Param[1].

Meaning:	This error may be returned by MODEs 4, 5, 7, 9, 11, 12, and 15; it indicates that the second parameter (Param [1]) passed to the driver is out of range.
Action:	Check the range of the parameter.

Error 12: Error in range of Param[2].

Meaning:	This error may be returned by MODE 12, 14, and 15; it indicates that the third parameter (Param [2]) passed to the driver is out of range.
Action:	Check the range of the parameter.

Error 13: Error in range of Param[3].

Meaning:	This error may be returned by MODE 14 and 15; it indicates that the fourth parameter (Param [3]) passed to the driver is out of range.
Action:	Check the range of the parameter.

Error 14: Error in range of Param[4].

Meaning:	This error may be returned by MODE 15; it indicates the fifth parameter (Param [4]) passed to the driver is out of range.
Action:	Check the range of the parameter.

Error 15: Error in range of Param[5].

Meaning:	This error may be returned by MODE 15; it indicates that the sixth parameter (Param [5]) passed to the driver is out of range.
Action:	Check the range of the parameter.

Error 16: Error in range of Param[6].

Meaning:	This error may be returned by MODE 15; it indicates that the seventh parameter (Param [6]) passed to the driver is out of range.
Action:	Check the range of the parameter.

Error 17: Error in range of Param[7].

Meaning:	This error may be returned by MODE 15; it indicates that the eighth parameter (Param [7]) passed to the driver is out of range.
Action:	Check the range of the parameter.

Error 18: Error in range of Param[8].

Meaning:	This error may be returned by MODE 15; it indicates that the ninth parameter (Param [8]) passed to the driver is out of range.
Action:	Check the range of the parameter.

Error 19: Error in range of Param[9].

Meaning:	This error may be returned by MODE 15; it indicates the tenth parameter (Param [9]) passed to the driver is out of range.
Action:	Check the range of the parameter.

Error 20: Error in range of Param[10].

Meaning:	This error may be returned by MODE 15; it indicates that the eleventh parameter (Param [10]) passed to the driver is out of range.
Action:	Check the range of the parameter.

Error 21: Error in range of Param[11].

Meaning:	This error may be returned by MODE 15; it indicates that the twelveth parameter (Param [11]) passed to the driver is out of range.
Action:	Check the range of the parameter.

Error 22: Error in range of Param[12].

Meaning:	This error may be returned by MODE 15; it indicates that the thirteenth parameter (Param [12]) passed to the driver is out of range.
Action:	Check the range of the parameter.

Error 23: Error in range of Param[13].

Meaning:	This error may be returned by MODE 15; it indicates that the fourteenth parameter (Param [13]) passed to the driver is out of range.
Action:	Check the range of the parameter.

Error 24: Error in range of Param[14].

Meaning:	This error may be returned by MODE 15; it indicates that the fifteenth parameter (Param [14]) passed to the driver is out of range.
Action:	Check the range of the parameter.

■ ■ ■