PCF-M5

* * * * *

User Guide

for the

PCF-M5

Language Drivers

Revision A - April 1991

Part Number: 24413

KEITHLEY METRABYTE CORPORATION

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.

Notes

Keithley MetraByte/Asyst/DAC is also referred to here-in as *Keithley MetraByte.*

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**® (MCA) 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

MetraByte's PCF-M5 is for Pascal, C, and Fortran programmers writing data acquisition and control routines for the MSTEP-5 Board. The PCF-M5 supports all memory models for the following languages;

- Microsoft C (V4.0 - 6.0)
- Microsoft QuickC (V1.0 - 2.0)
- Borland Turbo C (V1.0 - 2.0)
- Microsoft PASCAL (V3.0 - 4.0)
- Borland Turbo PASCAL (V3.0 - 5.0)
- Microsoft FORTRAN (V4.0 - 4.1)

The PCF-M5 consists of several assembly language drivers for the various supported languages along with example programs for each language. This manual is structured to illustrate memory model usage for each of the above languages and to include a brief example program at the end of each language section. Full source listings are included on the supplied disk.

This manual is not an introduction or operating guide to the supported MSTEP-5 boards. You should be familiar with the boards' various operating MODES, PARAMETERS, and ERROR codes before attempting PCF-M5 implementation. Refer to the main sections of this manual supplied with your MSTEP-5 MetraByte board for a complete discussion of hardware and related functionality.

PCF-M5 Distribution Software is furnished on a 5.25" floppy diskette. A 3.5" diskette version is available as an option.

A1.2 IMPLEMENTATION

Before working with this interface package, you are urged to become familiar with MSTEP-5 board functions and specifications. Example programs herein do not assume any knowledge of these boards since the programs are general in nature and do not actually implement features of any specific board. They are limited to the actual language interface for the various languages supported.

In the following chapter, each interface driver (implemented via a CALL statement) consists of three position-dependent parameters. These are MODE, ARGUMENT (or PARAM), STPNUM, and FLAG, as follows:

MODE	Type of function to be executed by the MSTEP-5.
PARAM	Function-dependent arguments required for execution.
STPNUM	Step Number, which is a long integer that specifies the direction and number of steps to travel, or it returns optical shaft encoder counts. The sign indicates direction (+ = clockwise, - = counter-clockwise). Not all MODEs use StpNum data, but it must always be included in the CALL parameter list.
FLAG	Error number, if any, corresponding to selected MODE

* * * * *

CHAPTER 2

INTERFACE DRIVERS

2.1 MICROSOFT C (V4.0 - 6.0) & QUICKC (V1.0 - 2.0)

Small Model

Model:	Small ("/AS") switch on command line
Passes:	word size pointers (offset, no DS register)
Sequence:	Arguments Passed Right to Left
Default Calling Convention:	Arguments Passed by Value (Passing pointers to a subroutine is considered pass-by-value convention)

Example:

'C' Call:	**mscs_mstep (&Mode, Param, &StpNum, &Flag);**
'C' Declaration:	**extern void mscs_mstep(int*,unsigned long *, int*);**

.ASM Subroutine:

The following assembly code shows how the driver handles user arguments:

```
_mscs_mstep proc near
        push bp                 ; save base pointer
        mov bp,sp               ; save stack pointer
        .                       ; [bp+4] holds offset of Mode
        .                       ; [bp+6] holds offset of Param
        .                       ; [bp+8] holds offset of StpNum
        .                       ; [bp+10] holds offset of Flag
        .                       ; Program execution here
        .                       ;
        .                       ;
        pop bp                  ;restore bp & sp prior to exit
        ret   ;return
_mscs_mstep endp
```

Other:

This information is provided for those wishing to create their own drivers:

- _mscs_mstep is declared "PUBLIC" in the .ASM file
- mscs_mstep is declared "extern" in the "C" file

- The .ASM file contains the ".model small" directive (MASM & TASM only)
- Add leading underscore "_" to all mscs_mstep occurrences in .ASM file
- mscs_mstep is a near call
- mscs_mstep must be in a segment fname_TEXT (where fname is the name of the file where mscs_mstep resides) if .ASM file contains mixed model procedures.

Medium Model

Model:	Medium ("/AM") switch on command line
Passes:	Word-size pointers (offset, no DS register)
Sequence:	Arguments Passed Right to Left
Default Calling Convention:	Arguments Passed by Value

Example

'C' Call:	**mscm_mstep (&Mode, Param, &StpNum, &Flag);**
'C' Declaration:	**extern void mscm_mstep(int*,unsigned long*,int*);**

.ASM Subroutine:

The following assembly code shows how the driver handles user arguments:

```
_mscm_mstep proc far        ; far CALL (dword return address)
       push bp              ; save base pointer
       mov bp,sp            ; save stack pointer
       .                    ; [bp+6] holds offset of Mode
       .                    ; [bp+8] holds offset of Param
       .                    ; [bp+10] holds offset of StpNum
       .                    ; [bp+12] holds offset of Flag
       .                    ; Program execution here
       .                    ;
       .                    ;
       pop bp               ;restore bp & sp prior to exit
       ret   ;return
_mscm_mstep endp
```

Other:

This information is provided for those wishing to create their own drivers:

- _mscm_mstep is declared "PUBLIC" in the .ASM file
- mscm_mstep is declared "extern" in the "C" file
- The .ASM file contains the ".model medium" directive (MASM & TASM only)
- Add leading underscore "_" to all mscm_mstep occurrences in .ASM file
- mscm_mstep is a far call

- mscm_mstep must be in a segment fname_TEXT (where fname is the name of the file where mscm_mstep resides), else Linker returns an error.

Large Model

Model:	Large ("/AL") switch on command line
Passes:	dword size pointers (offset and DS register)
Sequence:	Arguments Passed Right to Left
Default Calling Convention:	Arguments Passed by Value

Example

'C' Call:	**mscl_mstep (&Mode, Param, &StpNum, &Flag);**
'C' Declaration:	**extern void mscl_mstep(int*, unsigned long*, int*);**

.ASM Subroutine:

The following assembly code shows how the driver handles user arguments:

```
_mscl_mstep proc far        ; far CALL (dword return address)
       push bp              ; save base pointer
       mov bp,sp            ; save stack pointer
       .                    ; [bp+6] holds offset of Mode
       .                    ; [bp+10] holds offset of Param
       .                    ; [bp+14] holds offset of StpNum
       .                    ; [bp+18] holds offset of Flag
       .                    ; Program execution here
       .                    ;
       .                    ;
       pop bp               ;restore bp & sp prior to exit
       ret   ;return
_mscl_mstep endp
```

Other:

This information is provided for those wishing to create their own drivers:

- _mscl_mstep is declared "PUBLIC" in the .ASM file
- mscl_mstep is declared "extern" in the "C" file
- The .ASM file contains the ".model large" directive (MASM & TASM only)
- Add leading underscore "_" to all mscl_mstep occurrences in .ASM file
- Both code and data use dword (segment/offset) pointers
- mscl_mstep must be in a segment fname_TEXT (where fname is the name of the file where mscl_mstep resides), else Linker returns an error.

Microsoft 'C' Example

```
/*                                                                    */
/*            Demonstration Program for MSTEP-5                       */
/*            Keithley Metrabyte Corporation                          */
/*                                                                    */
/*            Language:    Microsoft "C"                              */
/*            File:        MSCSDEMO.C                                 */
/*                                                                    */

#include <stdio.H>
#include <stdlib.H>

#define Beep printf("%s","\7")      /* make a beep sound */

#define TRYAGAIN        1
#define REV_VIDEO       7
#define DEF_VIDEO       0
#define READY           1
#define ON              1
#define OFF             0

/*********************************************************************/
/*                                                                   */
/*     The Following are the function Calls for different models:    */
/*                                                                   */
/*     mscs_mstep(&Mode,Param,&StpNum,&Flag) : Microsoft C small Model. */
/*     mscm_mstep(&Mode,Param,&StpNum,&Flag) : Microsoft C medium Model.*/
/*     mscl_mstep(&Mode,Param,&StpNum,&Flag) : Microsoft C large Model. */
/*                                                                   */
/*     This Program Uses the Small Model Function Call               */
/*                                                                   */
/*********************************************************************/
                  /* DECLARE CALL structure */
extern  mscs_mstep(int *,unsigned *,long *,int *);

/*********************************************************************/
int  mode0(void);                          /* mode 0   */
int  mode1(void);                          /* mode 1   */
int  mode2(void);                          /* mode 2   */
int  mode3(void);                          /* mode 3   */
int  mode4(void);                          /* mode 4   */
int  mode5(void);                          /* mode 5   */
int  mode6(void);                          /* mode 6   */
int  mode7(void);                          /* mode 7   */
int  mode8(void);                          /* mode 8   */
int  mode9(void);                          /* mode 9   */
int  mode10(void);                         /* mode 10  */
int  mode11(void);                         /* mode 11  */
int  mode12(void);                         /* mode 12  */
int  setchannel(void);                     /* select channel          */
int     ErrHandler(int);                   /* error handler           */
int     MenuHandler(void);                 /* menu handler            */
int  GetLongInp(int,int,char *,long *);    /* input get long integer  */
void GetIntInp(int,int,char *,unsigned *); /* input unsigned integer  */
void printat(int,int,int,char *);          /* formatted print function */
```

```
void ShowChannel(void);                          /* print current channel    */
void EraseLine(int);                             /* erase line               */
/*************************************************************************/

int     Mode=12,Flag=0,IntLev=7,Interrupt=OFF;
unsigned  Param[10],TP[10];
long    StpNum;

/*************************************************************************/
/*                                                                       */
/*                        Main program                                   */
/*                                                                       */
/*************************************************************************/
main()
{
int     option,status=!TRYAGAIN;

/* default setup for MSTEP-5 */

   TP[0] =   0;        // select channel A
   TP[1] = 255;        // start up rate, 49 pps
   TP[2] = 100;        // high speed run rate
   TP[3] = 200;        // acceleration/deceleration pulse count
   TP[4] =   2;        // 4 phase
   TP[5] =   0;        // full step
   TP[6] =   0;        // inverted S1-5 outputs
   TP[7] =   0;        // internal clock
   TP[8] =   0;        // switching off at standstill
   TP[9] = 768;        // base address
          .
          .
          .
};
          .
          .
          .

/*************************************************************************/
/*                                                                       */
/*      Mode 12: Initialization.                                         */
/*                                                                       */
/* Arguments                                  Return values      */
/*      ---------                               -------------   */
/*      None                                            1: repeat this mode*/
/*                                                         again         */
/*                                                      0: no repeat     */
/*                                                                       */
/*************************************************************************/
int mode12()
{
          .
          .
          .
}
          .
          .
          .
```

2.2 BORLAND TURBO 'C' (V1.0 - 2.0)

Small Model

Model:	Small ("-ms") switch on command line
Passes:	word size pointers (offset, no DS register)
Sequence:	Arguments Passed Right to Left
Default Calling Convention:	Arguments Passed by Value

Example

'C' Call:	**tcs_mstep (&Mode, Param, &StpNum, &Flag);**
'C' Declaration:	**extern void tcs_mstep(int*,unsigned long*,int*);**

.ASM Subroutine:

The following assembly code shows how the driver handles user arguments:

```
_tcs_mstep   proc near
       push bp              ; save base pointer
       mov bp,sp            ; save stack pointer
       .                    ; [bp+4] holds offset of Mode
       .                    ; [bp+6] holds offset of Param
       .                    ; [bp+8] holds offset of StpNum
       .                    ; [bp+10] holds offset of Flag
       .                    ; Program execution here
       .                    ;
       .                    ;
       pop bp               ;restore bp & sp prior to exit
       ret   ;return
_tcs_mstep endp
```

Other:

This information is provided for those wishing to create their own drivers:

- _tcs_mstep is declared "PUBLIC" in the .ASM file
- tcs_mstep is declared "extern" in the "C" file
- The .ASM file contains the ".model small" directive (MASM & TASM only)
- Add leading underscore "_" to all tcs_mstep occurrences in .ASM file
- tcs_mstep is a near call
- tcs_mstep must be in a segment fname_TEXT (where fname is the name of the file where tcs_mstep resides), else Linker returns an error.

Medium Model

Model:	Medium ("-mm") switch on command line
Passes:	word size pointers (offset, no DS register)
Sequence:	Arguments Passed Right to Left
Default Calling Convention:	Arguments Passed by Value

Example

'C' Call:	**tcm_mstep (&Mode, Param, &StpNum, &Flag);**
'C' Declaration:	**extern void tcm_mstep(int*,unsigned long*, int*);**

.ASM Subroutine:

The following assembly code shows how the driver handles user arguments:

```
_tcm_mstep proc far          ; dword pointer return address
       push bp               ; save base pointer
       mov bp,sp             ; save stack pointer
       .                     ; [bp+6] holds offset of Mode
       .                     ; [bp+8] holds offset of Param
       .                     ; [bp+10] holds offset of StpNum
       .                     ; [bp+12] holds offset of Flag
       .                     ; Program execution here
       .                     ;
       .                     ;
       pop bp                ;restore bp & sp prior to exit
       ret   ;return
_tcm_mstep endp
```

Other:

This information is provided for those wishing to create their own drivers:

- _tcm_mstep is declared "PUBLIC" in the .ASM file
- tcm_mstep is declared "extern" in the "C" file
- The .ASM file contains the ".model medium" directive (MASM & TASM only)
- Add leading underscore "_" to all tcm_mstep occurrences in .ASM file
- tcm_mstep must be in a segment fname_TEXT (where fname is the name of the file where tcm_mstep resides), else Linker returns an error.

Large Model

Model:	Large ("-ml") switch on command line
Passes:	dword size pointers (offset and DS register)
Sequence:	Arguments Passed Right to Left

Default Calling Convention: Arguments Passed by Value

Example

'C' Call: **tcl_mstep (&Mode, Param, &StpNum, &Flag);**

'C' Declaration: **extern void tcl_mstep(int*, unsigned long*, int*);**

.ASM Subroutine:

The following assembly code shows how the driver handles user arguments:

```
_tcl_mstep proc far          ; dword pointer return address
       push bp               ; save base pointer
       mov bp,sp             ; save stack pointer
       .                     ; [bp+6] holds offset of Mode
       .                     ; [bp+10] holds offset of Param
       .                     ; [bp+14] holds offset of StpNum
       .                     ; [bp+18] holds offset of Flag
       .                     ; Program execution here
       .                     ;
       .                     ;
       pop bp                ;restore bp & sp prior to exit
       ret   ;return
_tcl_mstep endp
```

Other:

This information is provided for those wishing to create their own drivers:

- _tcl_mstep is declared "PUBLIC" in the .ASM file
- tcl_mstep is declared "extern" in the "C" file
- The .ASM file contains the ".model large" directive (MASM & TASM only)
- Add leading underscore "_" to all tcl_mstep occurrences in .ASM file
- Both code & data use dword (segment/offset) pointers
- tcl_mstep must be in a segment fname_TEXT (where fname is the name of the file where tcl_mstep resides), else Linker returns an error.

Turbo C Example

```
/*                                                              */
/*      Demonstration Program for MSTEP-5                       */
/*      Keithley Metrabyte Corporation                          */
/*                                                              */
/*      Language:     Borland Turbo "C"                         */
/*      File:         TCSDEMO.C                                 */
/*                                                              */
```

```
#include <stdio.H>
#include <stdlib.H>
#include <string.H>

#define Beep printf("%s","\7")      /* make a beep sound */

#define TRYAGAIN        1
#define REV_VIDEO       7
#define DEF_VIDEO       0
#define READY           1
#define ON              1
#define OFF             0

/*************************************************************************/
/*                                                                       */
/*     The Following are the function Calls for different models:        */
/*                                                                       */
/*     tcs_mstep(&Mode,Param,&StpNum,&Flag)   : Turbo C small Model.     */
/*     tcm_mstep(&Mode,Param,&StpNum,&Flag)   : Turbo C medium Model.    */
/*     tcl_mstep(&Mode,Param,&StpNum,&Flag)   : Turbo C large Model.     */
/*                                                                       */
/*     This Program Uses the Small Model Function Call                   */
/*                                                                       */
/*************************************************************************/
                /* DECLARE CALL structure */
extern  tcs_mstep(int *,unsigned *,long *,int *);

/*************************************************************************/
int  mode0(void);                           /* mode 0   */
int  mode1(void);                           /* mode 1   */
int  mode2(void);                           /* mode 2   */
int  mode3(void);                           /* mode 3   */
int  mode4(void);                           /* mode 4   */
int  mode5(void);                           /* mode 5   */
int  mode6(void);                           /* mode 6   */
int  mode7(void);                           /* mode 7   */
int  mode8(void);                           /* mode 8   */
int  mode9(void);                           /* mode 9   */
int  mode10(void);                          /* mode 10  */
int  mode11(void);                          /* mode 11  */
int  mode12(void);                          /* mode 12  */
int  setchannel(void);                      /* select channel        */
int     ErrHandler(int);                    /* error handler         */
int     MenuHandler(void);                  /* menu handler          */
int  GetLongInp(int,int,char *,long *);     /* input get long integer    */
void GetIntInp(int,int,char *,unsigned *);  /* input unsigned integer*/
void printat(int,int,int,char *);           /* formatted print function  */
void ShowChannel(void);                     /* print current channel */
void EraseLine(int);                        /* erase line            */
/*************************************************************************/

int     Mode=12,Flag=0,IntLev=7,Interrupt=OFF;
unsigned  Param[10],TP[10];
long    StpNum;
```

```
/*************************************************************************/
/*                                                                       */
/*                        Main program                                   */
/*                                                                       */
/*************************************************************************/
main()
{
int     option,status=!TRYAGAIN;

/* default setup for MSTEP-5 */

  TP[0] =   0;        /* select channel A                          */
  TP[1] = 255;        /* start up rate, 49 pps                     */
  TP[2] = 100;        /* high speed run rate                       */
  TP[3] = 200;        /* acceleration/deceleration pulse count     */
  TP[4] =   2;        /* 4 phase                                   */
  TP[5] =   0;        /* full step                                 */
  TP[6] =   0;        /* inverted S1-5 outputs                     */
  TP[7] =   0;        /* internal clock                            */
  TP[8] =   0;        /* switching off at standstill               */
  TP[9] = 768;        /* base address                              */

          .
          .

}

          .
          .

/*************************************************************************/
/*                                                                       */
/*      Mode 12: Initialization.                                         */
/*                                                                       */
/* Arguments                              Return values       */
/*      ---------                            -------------   */
/*      None                                          1: repeat this mode */
/*                                                       again           */
/*                                                    0: no repeat       */
/*                                                                       */
/*************************************************************************/
int mode12()
{

          .
          .
          .

}

          .
          .
          .
```

2.3 MICROSOFT PASCAL (V3.0 - 4.0)

Medium Model

Model:	Medium
Passes:	word size pointers (offset address only)
Sequence:	Arguments Passed Left to Right
Default Calling Convention:	Arguments Passed by Value

Example

PASCAL Call: Result:	msp_mstep (Mode, Param,StpNum,Flag)
PASCAL Declaration:	type darray = array[0-9] of word;
PROCEDURE:	msp_mstep(VAR Mode: integer; VAR param; darray; VAR StpNum: integer4; VAR Flag: integer):external;

.ASM Subroutine:

The following assembly code shows how the driver handles user arguments:

```
msp_mstep proc far          ; far call (dword return address)
       push bp              ; save base pointer
       mov bp,sp            ; save stack pointer
       .                    ; [bp+6] holds offset of Flag
       .                    ; [bp+8] holds offset of StpNum
       .                    ; [bp+10] holds offset of Param
       .                    ; [bp+12] holds offset of Mode
       .                    ; Program execution here
       .                    ;
       .                    ;
       mov ax,n             ; Return Value for Function In ax register
       pop bp               ;
       ret 8 ; return and pop bp & sp values prior to exit
msp_mstep endp
```

Other:

This information is provided for those wishing to create their own drivers:

- msp_mstep is declared "PUBLIC" in the .ASM file
- msp_mstep is declared external in the calling program
- msp_mstep resides in segment_TEXT (default of the .model command)

Microsoft PASCAL Example

```
PROGRAM MSPDEMO(Input,Output);
(*************************************************************************)
(* Demonstration program for MSTEP-5                                     *)
(* Keithley Metrabyte Corporation                                        *)
(*                                                                       *)
(* Language:     Microsoft Pascal                                        *)
(* File:         MSPDEMO.PAS                                             *)
(*                                                                       *)
(* To Compile:                                                           *)
(*               Type:      PL MSPDEMO.PAS ;                           *)
(*                                                                       *)
(*************************************************************************)
TYPE DARRAY = ARRAY[0..9] of WORD;

FUNCTION TICS:WORD;EXTERN;
FUNCTION KEYRD:WORD;EXTERN;

PROCEDURE MSP_MSTEP(
VAR Mode:INTEGER;VAR Param:DARRAY;VAR StpNum:INTEGER4;VAR
Flag:INTEGER);EXTERN;

          .
          .
          .

(*************************************************************************)
(*                                                                       *)
(*Mode 12: Initialization                                               *)
(*                                                                       *)
(*************************************************************************)
PROCEDURE  Mode12;
VAR  count,option:INTEGER;
BEGIN
          .
          .
          .

END;

          .
          .
          .

(*************************************************************************)
(*                                                                       *)
(*               Main                                                    *)
(*                                                                       *)
(*************************************************************************)
```

```
BEGIN

  TP[0] :=   0;       (* select channel A                        *)
  TP[1] := 255;       (* start up rate, 49 pps                   *)
  TP[2] := 100;       (* high speed run rate                     *)
  TP[3] := 200;       (* acceleration/deceleration pulse count   *)
  TP[4] :=   2;       (* 4 phase                                 *)
  TP[5] :=   0;       (* full step                               *)
  TP[6] :=   0;       (* inverted S1-5 outputs                   *)
  TP[7] :=   0;       (* internal clock                          *)
  TP[8] :=   0;       (* switching off at standstill             *)
  TP[9] := 768;       (* base address                            *)
  StpNum := 0;Flag := 0;IntLev := 7;

          .
          .
          .

END.
```

2.4 BORLAND TURBO PASCAL (VER 3.0 - 4.0)

Borland's Turbo PASCAL supports a compact and a large memory model. The compact model supports one code segment and multiple data segments. In this model, the code segment is limited to 64K with assembly routine calls being near calls. The data segment is unlimited. The large model permits unlimited code and data segments with assembly calls and data access being far calls.

The program (TINST.EXE) shipped with TURBO PASCAL can change the calling convention so that the user may not know which convention they are using. The default state is "OFF" or compact mode. In order to ascertain which mode you are using, run the "TINST.EXE" program.

Compact Model

Model:	Compact (Forces far call "OFF" in TINST.EXE)
Passes:	dword size pointers (offset and segment)
Sequence:	Arguments Passed Left to Right
Default Calling Convention:	Arguments Passed by Value

Example

PASCAL Call:	tp_mstep (Mode, Param, StpNum, Flag);
PASCAL Declaration:	type darray = array[0-9] of word;
PROCEDURE:	tp_mstep(VAR Mode: integer;VAR Param: darray;VAR StpNum: integer4;VAR Flag: integer):external;

.ASM Subroutine:

(Either Model)

The following assembly code shows how the driver handles user arguments:

```
tp_mstep  proc near          ; near call (single word return address)
        push bp              ; save base pointer
        mov bp,sp            ; save stack pointer
        .                    ; [bp+4] holds offset of Flag
        .                    ; [bp+8] holds offset of StpNum
        .                    ; [bp+12] holds offset of Param
        .                    ; [bp+16] holds offset of Mode
        .                    ; Program execution here
        .                    ;
        .                    ;
        mov ax,n             ; return Value for Function In ax register
        pop bp
        ret 16               ; return & pop values prior to exit
tp_mstep  endp
```

Other:

This information is provided for those wishing to create their own drivers:

- Use the $L 'Metacommand' to link the object file containing external function tp_mstep, i.e. {$l turbopas} (Link to file turbopas.obj).
- The VAR declarative forces pass by reference (address of variable) in the function declaration. Default is pass by value (pushing the actual integer value onto the stack).
- tp_mstep is declared external in the calling program . Remember that in PASCAL, functions return a value whereas procedures never do.
- The .ASM file contains an explicit declaration of the code segment containing tp_mstep. Turbo PASCAL handles segments in a primitive manner which is not compatible with the '.model' statements available in MASM or TASM. The function tp_mstep must reside in a segment called 'CODE'! Turbo PASCAL will not accept any other segment name. If tp_mstep is not in segment "CODE", the linker returns an "unresolved external" error. The Segment Declaration for "CODE" in the .ASM file must appear as:

```
CODE SEGMENT WORD PUBLIC
ASSUME CS:CODE
.
.       ; CODE GOES HERE
.
CODE  ENDS
```

Large Model

Model:	Large (Forces far call "ON" in TINST.EXE)
Passes:	dword size pointers (offset and segment)
Sequence:	Arguments Passed Left to Right

Default Calling Convention:	Arguments Passed by Value

Example

PASCAL Call:	tp_mstep (Mode, Param, StpNum, Flag);
PASCAL Declaration:	type darray = array[0-9] of word;
PROCEDURE:	tp_mstep(VAR Mode: integer;VAR Param: darray;VAR StpNum: integer4;VAR Flag: integer):external;

.ASM Subroutine:

(Either Model)

The following assembly code shows how the driver handles user arguments:

```
tp_mstep  proc far          ; far call (dword return address)
        push bp             ; save base pointer
        mov bp,sp           ; save stack pointer
        . ; [bp+6] holds dword of VAR4
        . ; [bp+10] holds dword of VAR3
        . ; [bp+14] holds dword of VAR2
        . ; [bp+18] holds dword of VAR1
        . ; Program execution here
        . ;
        . ;
        mov ax,n            ; return Value for Function In ax register
        pop bp
        ret 16              ; return & pop values prior to exit
tp_mstep  endp
```

Other:

This information is provided for those wishing to create their own drivers:

- Use the $L 'Metacommand' to link the object file containing external function tp_mstep. For example; {$l turbopas} (Link file turbopas.obj).
- The VAR declarative forces pass by reference (address of variable) in the function declaration. Default is pass by value (pushing the actual integer value onto the stack).
- tp_mstep is declared external in the calling program along with the type of return value (integer). Remember, in PASCAL, functions return a value procedures don't.
- The .ASM file contains an explicit declaration of the code segment containing tp_mstep.

Turbo PASCAL Example

```
PROGRAM TP_DEMO(Input,Output);
{$L TURBOPAS}
{$I-}
USES CRT,DOS;
```

```
(*************************************************************)
(* Demonstration program for MSTEP-5                         *)
(* Keithley Metrabyte Corporation                            *)
(*                                                           *)
(* Language:      Borland Turbo Pascal                       *)
(* File:          TP_DEMO.PAS                                *)
(*                                                           *)
(* To Compile:                                               *)
(*                Type:      TPC TP_DEMO.PAS ;               *)
(*                                                           *)
(*************************************************************)
TYPE DARRAY = ARRAY[0..9] of WORD;
LABEL  CONTINUE,QUIT;
PROCEDURE TP_MSTEP(
VAR Mode:INTEGER;VAR Param:DARRAY;VAR StpNum:LONGINT;VAR
Flag:INTEGER);EXTERNAL;

          .
          .
          .

(*************************************************************)
(*                                                           *)
(*Mode 12: Initialization                                   *)
(*                                                           *)
(*************************************************************)
PROCEDURE  Mode12;
LABEL  BREAKOUT;
VAR  count,option:INTEGER;
  DONE     : BOOLEAN;
BEGIN

          .
          .
          .

END;

          .
          .
          .

(*************************************************************)
(*                                                           *)
(*               Main                                        *)
(*                                                           *)
(*************************************************************)
```

```
BEGIN

  TP[0] :=   0;       (* select channel A                          *)
  TP[1] := 255;       (* start up rate, 49 pps                     *)
  TP[2] := 100;       (* high speed run rate                       *)
  TP[3] := 200;       (* acceleration/deceleration pulse count     *)
  TP[4] :=   2;       (* 4 phase                                   *)
  TP[5] :=   0;       (* full step                                 *)
  TP[6] :=   0;       (* inverted S1-5 outputs                     *)
  TP[7] :=   0;       (* internal clock                            *)
  TP[8] :=   0;       (* switching off at standstill               *)
  TP[9] := 768;       (* base address                              *)

  StpNum := 0;Flag := 0;IntLev := 7;

          .
          .
          .

END.
```

2.5 MICROSOFT FORTRAN (V4.0 AND UP)

Large Model

Model:	Large
Passes:	dword size pointers (offset and DS register)
Sequence:	Arguments Passed Left to Right
Default Calling Convention:	Arguments Passed by Reference

Example

FORTRAN Call:	call fmstep(Mode, Param(1), StpNum, Flag);
FORTRAN Declaration:	None necessary in FORTRAN source file (Fortran assumes that undeclared subroutines or functions are external. It is left to the linking process to provide the required .LIB or .OBJ files. However, the function name should conform to ANSI FORTRAN rules for integer functions.

.ASM Subroutines:

NOTE: FORTRAN integer functions (beginning with letters i, j, or k) return results in the ax register whereas non-integer functions reserve 4 bytes on the calling stack for a far pointer to the result. Non-integer functions pass their arguments starting at location bp+18 after the "push bp" and "mov bp,sp" instructions have been executed. Keithley MetraByte's FORTRAN <--> Assembly routines predominantly use type integer to avoid the non-integer problem. Using non-integer functions may be a problem when returning pointers, floating point results, long integers, etc. The user should use the IMPLICIT INTEGER (A-Z) declaration causing all Functions and Variables to be implicitly type

integer unless declared otherwise. Also note that FORTRAN calls by Reference. This method places the address of the passed parameters (rather than the parameters themselves) onto the stack at the time of the call to any function or subroutine. As a convenience, PCF-M5 provides functions (INBYT and OUTBYT) for directly addressing the registers and (KEYHOT and KEYRD) for checking Hot Key and reading a key from the keyboard.

Integer (Default) Function or Subroutine

The following assembly code shows how the driver handles user arguments:

```
fmstep proc far              ; dword pointer return address
       push bp               ; save base pointer
       mov bp,sp             ; save stack pointer
       .                     ; [bp+6] holds offset of Flag
       .                     ; [bp+10] holds offset of StpNum
       .                     ; [bp+14] holds offset of Param
       .                     ; [bp+18] holds offset of Mode
       .                     ; Program execution here
       .                     ;
       .                     ;
       mov ax,n              ; return Value for Function In ax register
       pop bp
       ret   ;
fmstep endp
```

NOTES:

1. VAR3 = Return Value of Function
2. Function fmstep must be declared as an integer * 2 fucntion.

Microsoft FORTRAN Example

```
C*****************************************************************
C
C       Demonstration program for MSTEP-5
C       Keithley Metrabyte Corporation
C
C       Language:     Microsoft Fortran
C       File:         MSFDEMO.FOR
C
C*****************************************************************
        program msfdemo

        character NULL,ESC,ch
        integer*2 key,Mode,Flag,IntLev,Param(10),TemPar(10)
        integer*4 StpNum

        COMMON    /ASCII/NULL,ESC
        COMMON    /ARG/Mode,Flag,Param,StpNum
        COMMON    /TEMPLATE/TemPar
        COMMON    /INTERRUPT/IntLev
```

```
        ESC     = char(27)
        NULL    = char(0)

C               select channel A
        TemPar(1)  = 0
C               start up rate, 49 pps
        TemPar(2)  = 255
C               high speed run rate
        TemPar(3)  = 100
C               acceleration/deceleration pulse count
        TemPar(4)  = 200
C               4 phase
        TemPar(5)  = 2
C               full step
        TemPar(6)  = 0
C               inverted S1-5 outputs
        TemPar(7)  = 0
C               internal clock
        TemPar(8)  = 0
C               switching off at standstill
        TemPar(9)  = 0
C               base address
        TemPar(10) = 768

                .
                .
                .

        end

                .
                .
                .

C*****************************************************************
C
C       Mode 12: Initialization
C
C*****************************************************************
        subroutine mode12(*)

        integer*2 Flag,Param(10),Mode,TemPar(10),index,i
        integer*4 StpNum
        COMMON    /ARG/Mode,Flag,Param,StpNum
        COMMON    /TEMPLATE/TemPar

        Mode=12
        Flag=0
                .
                .
                .
        end
                .
                .
```

2.6 MSTEP.LIB GENERAL PURPOSE LIBRARY

mstep.LIB — This is a general purpose library file which provides control of the MSTEP-5 boards. This file can be linked with programs written in C, PASCAL, or FORTRAN to provide access to the MSTEP-5 operating modes.

NOTE: This library cannot be used with TurboPASCAL. However, TurboPASCAL may be used with Turbops.obj (see below).

The following is a brief description of the available call routines:

mscs_mstep(mode,param,stpnum,flag)	:	Call from Microsoft C Small Model
mscm_mstep(mode,param,stpnum,flag)	:	Call from Microsoft C Medium Model
mscl_mstep(mode,param,stpnum,flag)	:	Call from Microsoft C Large Model
tcs_mstep(mode,param,stpnum,flag)	:	Call from Turbo C Small Model
tcm_mstep(mode,param,stpnum,flag)	:	Call from Turbo C Medium Model
tcl_mstep(mode,param,stpnum,flag)	:	Call from Turbo C Large Model
msp_mstep(mode,param,stpnum,flag)	:	Call from Microsoft PASCAL
fmstep(mode,param,stpnum,flag)	:	Call from Microsoft FORTRAN

Linking the Library "mstep.lib" to the user program is accomplished after program compilation by including it in the link line as follows:

```
link userprog.obj,userprog,,user.lib_mstep.LIB;
```

userprog.obj is an object module produced by compilation of the user program.
userprog should be used for the resultant executable .EXE file.
user.lib is any other user library, if applicable.

To create the MSTE.LIB Library for PASCAL, C, or FORTRAN:

```
MASM /DBIN=0 MSTEP.ASM
MASM /DBIN=0 MSTEPPCF.ASM

LIB MSTEP+MSTEP+MSTEPPCF;
```

To create the MSTEP.QLB library for QuickBASIC 4.5:

```
MASM /DBIN=0 MSTEP.ASM
MASM /DBIN=0 MSTEPPCF.ASM

LINK /Q MSTEP.OBJ QBX.LIB,,,QBXQLB.LIB
```

To create the MSTE.BIN for BASIC:

```
MASM /DBIN=1 MSTE.ASM
MASM /DBIN=1 MSTEPPCF.ASM
```

```
LINK MSTEPPCF + MSTE,MSTEP;
EXE2BIN MSTEP.EXE MSTEP.COM
MAKEBIN MSTEP.COM
```

For TurboPASCAL, the entry point is:

```
tp_mstep(mode,param,stpnum,flag)      :Call from TurboPASCAL program
```

The user program should have the directive

```
{$L turbopas}
```

at the beginning of the user program. This directive will ensure that the TPC compiler/linker will include the proper interface object.

TURBOPAS.OBJ (from the PCF-PIOINT disk)

* * * * *

CHAPTER 3

BASIC INTERFACE DRIVERS

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

Medium Model (Only Model Available)

Model:	Medium (Far Calls, Single Data)
Passes:	word size pointers (offset and no DS Register)
Sequence:	Arguments Passed Left to Right
Default Calling Convention:	Arguments Passed by Reference

.ASM Subroutine:

The following assembly code shows how the driver handles user arguments:

```
Location 0 (Beginning of Code Segment)

        jmp mstep

        .
        .
mstep   proc far            ; far call (dword return address)
        push bp             ; save base pointer
        mov bp,sp           ; save stack pointer
        .                   ; [bp+6] holds offset of Flag%
        .                   ; [bp+8] holds offset of STP#
        .                   ; [bp+10] holds offset of D%
        .                   ; [bp+12] holds offset of MD%
        .
        .                   ; Program execution here
        .
        .
        pop bp              ; restore bp & sp prior to exit
        ret
mstep   endp
```

Example:

Refer to Section 4.15 of the MSTEP-5 User Guide.

NOTE BASIC requires that the .BIN file containing the callable subroutine "mstep(Mode%, D%(0), STP#, Flag%)" reside at location 0 in the .ASM segment or to "jmp" (unconditional jump) to the .BIN file. A BASIC "jmp " will always jump to location 0 in the .ASM code segment.

Creation of a .BIN file is accomplished as follows:

1. Create the .ASM Source Code File 'EXAMPLE.ASM'
2. Assemble 'EXAMPLE.ASM' thus creating 'EXAMPLE.OBJ'
3. Link 'EXAMPLE.OBJ' to create 'EXAMPLE.EXE'
4. Run EXE2BIN on 'EXAMPLE.EXE' (DOS Utility) to create 'EXAMPLE.COM'
5. Run MAKEBIN.EXE (Keithley MetraByte Utility) on 'EXAMPLE.COM' to create 'EXAMPLE.BIN'

MASM EXAMPLE;

LINK EXAMPLE;

EXE2BIN EXAMPLE.EXE EXAMPLE.COM

MAKEBIN EXAMPLE.COM

3.2 QUICKBASIC

Medium Model (Only Model Available)

Model:	Medium (far Calls, Single Data).
Passes:	Word-size pointers (Offset and no DS Register).
Sequence:	Arguments passed left-to-right.
Default Calling Convention:	Arguments passed by reference.

.ASM Subroutine:

The following assembly code shows how the driver handles user arguments:

```
QBPIOINT  proc far          ; far call (dword return address)
        push bp             ; save base pointer
        mov bp,sp           ; save stack pointer
        .                   ; [bp+6] holds offset of Flag%
        .                   ; [bp+8] holds offset of D%
        .                   ; [bp+10] holds offset of STP#
        .                   ; [bp+12] holds offset of MD%
        .
        .                   ; Program execution here
        .
        .
        pop bp              ; restore bp & sp prior to exit
        ret
QBPIOINT  endp
```

Example:

Refer to Section 4.16 of the MSTEP-5 User Guide.

NOTE When creating a .QLB file, it is good practice to make a .LIB of the same version as a backup file.

Creation of a .QLB file is accomplished as follows:

1. Create the .ASM Source Code File 'EXAMPLE.ASM'

2. Assemble 'EXAMPLE.ASM' thus creating 'EXAMPLE.OBJ'

3. Link 'EXAMPLE.OBJ' with the "/q" option to create 'EXAMPLE.QLB'

MASM EXAMPLE;
LINK /q EXAMPLE;

A .LIB file is created by:

1. Create the .ASM Source Code File 'EXAMPLE.ASM'

2. Assemble 'EXAMPLE.ASM' thus creating 'EXAMPLE.OBJ'

3. Use Utility LIB.EXE to add EXAMPLE.OBJ to 'EXAMPLE.LIB'

(Remove old EXAMPLE.OBJ from Library)

LIB EXAMPLE.LIB -EXAMPLE

(Create New .OBJ)	**MASM EXAMPLE;**
(Add New .OBJ to Library)	**LIB EXAMPLE,LIB +EXAMPLE;**

4. To use the .QLB file in the QB integrated environment/editor, invoke QB.EXE with the /l option (QB /l qlbname.qlb,) where qlbname.qlb is the file containing BASICsub.

5. To use the .LIB file with the command line complier (BC.EXE), simply specify "EXAMPLE.LIB" in the link process.

* * * * *