PCF-16G
&
PCF-μC16G

User Guide

for the

Keithley MetraByte

PCF-16G & PCF-μC16G

PASCAL, C, & FORTRAN Callable Drivers

for the

DAS-16, DAS-16F, DAS-16G, & μCDas-16G Boards

Revision C - January 1992

Part Number: 24841

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-16G is a software package for programmers using Pascal, Turbo PASCAL, C, FORTRAN, and QuickBASIC to write data acquisition and control routines (referred to herein as *Application Code*) for the DAS-16/F/G and uCDAS-16G. The Distribution Software for this package is normally supplied on 5.25" (3.5" for uCDAS-16G) low-density diskettes but is also available (upon request) on 3.5" diskette(s). Contents of the package include the following:

- DAS-16/F/G and uCDAS-16G 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 PCF-16G 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 *PCF16G*.

1. After making a directory named *PCF16G*, place your Distribution Software diskette in your PC's A Drive and log to that drive by typing **A:** .
2. Then, type **COPY *.*** *path***PCF16G**, where *path* is the drive designation and DOS path (if needed) to the PCF16G 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 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 DAS-16 and uCDAS-16 Drivers required for the supported languages. Since the Drivers support the full series of DAS-16 and uCDAS-16 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 DAS-16 (DAS-16 hereinafter refers to DAS-16/F/G and uCDAS-16G) 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 three Driver Source Modules are the essential building blocks for creating a DAS-16 Driver in any language:

DASG.ASM	Core of the driver.
PCDASG.ASM	Core of the driver.
DASGPCF.ASM	Driver interface module for PASCAL, C, FORTRAN, and QuickBASIC.

As mentioned earlier, these three 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 three 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:

BASDASG.LIB:	Driver for Compiled BASIC.
DASG.LIB:	Driver for Pascal, C, FORTRAN, and stand-alone QuickBASIC programs.
DASG.BIN:	Driver for BASIC(A).
DASG.QLB:	Driver for the QuickBASIC Integrated Development Environment (Ver. 4.0-4.5).
DASGX.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 DAS-16/F/G and uCDAS-16G driver software. More detailed explanations of each Mode are available in the main text of the DAS-16/F/G and uCDAS-16G User Guides.

MODE 0: Initialize board, input Base Address, Interrupt Level, and DMA Level.

MODE 1: Set multiplexer low and high scan limits.

MODE 2: Reads the current channel setting and scan limit settings.

MODE 3: This MODE performs a single A/D conversion. Data from the conversion is returned, the multiplexor is incremented, and the next channel gain is selected from the gain table set up in MODE 21. This MODE is slow and runs in the foreground. If calibration MODE 1 is selected in MODE 0 then data will be offset corrected. If calibration MODE 2 is selected in MODE 0 then both gain and offset will be corrected.

MODE 4: This routine will perform N A/D conversions after receipt of an external trigger. Scan rate can be set by the programmable timer (set up by MODE 17), or by the external trigger. This MODE is faster than MODE 3 but still runs in the foreground. MODE 21 sets gain table. If calibration MODE 1 is selected in MODE 0 then data will be offset corrected. If calibration MODE 2 is selected in MODE 0 then both gain and offset will be corrected.

MODE 5: This MODE is similar to MODE 4 except that data transfer is driven by interrupts. Scan rates are set by the programmable timer or the external trigger. Speed is medium and runs in the background. Data is transferred by an interrupt service routine. MODE 21 sets gain table. MODE 9 is used to transfer the data from memory into a BASIC Array (autocalibration is performed in MODE 9).

MODE 6: This MODE is the fastest MODE of data transfer. An array of integers is stored directly into memory by a DMA transfer. Since is no interrupt service routine to select the next channel's gain, all channels are at the same gain. MODE 9 is used to transfer data from memory into a BASIC array. Note that MODE 6 puts data into memory without performing any autocalibration. For calibrated results use MODE 9, which not only takes the data from memory, but also performs that autocalibration operation.

MODE 7: Disable DMA and interrupt operations started in MODEs 5,6,18, or 20.

MODE 8: Reports status of DMA and interrupt operations started by MODEs 5,6,18 or 20.

MODE 9: Block move of data acquired in MODEs 5, 6, and 20. If calibration MODE 1 is selected in MODE 0 then data will be offset corrected. If calibration MODE 2 is selected in MODE 0 then both gain and offset will be corrected.

MODE 10: Set Counter 0 configuration. This is the counter whose output is available at the output connector.

MODE 11: Load Counter 0 data.

MODE 12: Read Counter 0.

MODE 13: Output to digital outputs OP0 - OP7.

MODE 14: Read digital inputs IP0-IP7.

MODE 15: Output data to a single DAC channel.

MODE 16: Outputs data to both DAC channels.

MODE 17: Set programmable timer rate.

MODE 18: DAC waveform output and ADC input. (Note that no data correction is performed by MODE 18.

MODE 19: Analog trigger function.

MODE 20: A/D block scan on interrupt. This MODE will do a complete scan of channels using gains set by MODE 21. Runs in background at medium speed. Use MODE 9 to transfer data from memory into a BASIC array. Note that MODE 20 puts data into memory without performing any autocalibration. For calibrated results use MODE 9, which not only takes the data from memory, but also performs that autocalibration operation.

MODE 21: Set channel gains. Use this MODE to set up gain table prior to calling MODEs 3, 4, 5, and 20.

MODE 22: Output a square wave frequency from Counter 0 out. Frequency from 1.5Hz to 25KHz.

MODE 23: Allocate memory Segment.

MODE 24: Deallocate Memory Segment.

MODE 25: Initialize PPI chip.

MODE 26: I/O to PPI.

2.5 CALLING THE DRIVER

In your Application Code, you write a call the DAS-16/F/G or uCDAS-16G 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* . DAS-16 Call Labels and their corresponding Drivers are as follows:

DASG.LIB:

mscs_dasg	For Calls from Microsoft C, Small Model
mscm_dasg	For Calls from Microsoft C, Medium Model
mscl_dasg	For Calls from Microsoft C, Large Model
tcs_dasg	For Calls from TURBO C, Small Model
tcm_dasg	For Calls from TURBO C, Medium Model
tcl_dasg	For Calls from TURBO C, Large Model
msp_dasg	For Calls from Microsoft Pascal
msf_dasg	For Calls from Microsoft FORTRAN
basdasg	For Calls from Microsoft QuickBASIC

TURBOPAS.OBJ:

tp_dasg	For Calls from TURBO Pascal

DASG.BIN:

dasg	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 DAS-16 driver.

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

FLAG A 16-bit integer quantity that contains a number representing any error code reported by the DAS-16 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;
int     Flag;
int     Params[16];
    :
Mode = 0;
Flag = 0;

Params[0] = 0x300;      /* Card Base Address */
Params[1] = 7;          /* Selected Interrupt Level */
Params[2] = 3;          /* Selected DMA Level */

mscl_dasg(&Mode, Params, &Flag);
if (Flag != 0)
  printf ("**** Error %d detected in mode 0", 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 *DASG*. These definitions use the */D* option for BASIC, QuickBASIC, PASCAL, C, and FORTRAN. For Turbo PASCAL, only the symbol *DAS16A* requires definition. 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** .

DASG = 1: Support for DAS-16A. Usage example: **/DDASG=1** .

DASG = 0: No support for DAS-16A. Usage example: **/DDASG=0** .

DAS16A = 1: Support for DAS-16A (Turbo PASCAL Driver only). Usage example: **/DDAS16A=1** .

DAS16A = 0: No support for DAS-16A (Turbo PASCAL Driver only). Usage example: **/DDAS16A=0** .

WARNING

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

The DASG.BIN Driver For BASIC(A)

To create a DASG.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 DAS-16/F/G and uCDAS-16 Distribution Software).

Then, use the following commands:

```
MASM /DBIN=1 /DDASG=1 DASG.ASM;
MASM /DBIN=1 PCDASG.ASM
MASM /DBIN=1 DASGPCF.ASM;
LINK DASPCF + DASG + PCDASG, DASG;
EXE2BIN DASG.EXE DASG.COM
MAKEBIN DASG.COM
```

All four 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 DASG.QLB Driver For The QuickBASIC Integrated Environment (V4.5)

To create the DASG.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 /DDASG=1  DASG.ASM;
MASM /DBIN=0 PCDASG.ASM
MASM /DBIN=0 DASGPCF.ASM;
LINK /q DASG+PCDASG+PCFDASG, DASG,,BQLB45;
```

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

To create the DASG.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 /DDASG=1  DASG.ASM;
MASM /DBIN=0 PCDASG.ASM;
MASM /DBIN=0 DASGPCF.ASM;
LIB DASG-+DASG;
LIB DASG-+PCDASG;
LIB DASG-+DASGPCF;
```

The DASGX.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 DASG+PCDASG+DASGPCF, DASGX,,QBXLB;
```

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

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

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

To create the DASGX.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 /DDASG=1  DASG.ASM;
MASM /DBIN=0 PCDASG.ASM;
MASM /DBIN=0 DASGPCF.ASM;
LIB DASG-+DASG;
LIB DASG-+PCDASG;
LIB DASG-+DASGPCF;
```

■ ■ ■

CHAPTER 3

DRIVER USAGE

3.1 OVERVIEW

Although your DAS-16 (DAS-16 refers hereinafter to DAS-16/F/G and uCDAS-16G) drivers perform similarly for all supported languages, there are differences from language-to-language in how they pass parameters and parameter values. The items causing the most confusion are as follows:

- Memory allocation for DMA buffers.
- Separating a FAR (32-bit) pointer into its Segment and Offset values (two 16-bit values).

This chapter discusses these items and any others of concern in the separate treatment of each supported language. 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 DAS-16 product.

Function Prototypes

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

```
mscs_dasg(int *, int *, int *);      /* MS C Small Model */
mscm_dasg(int *, int *, int *);      /* MS C Medium Model */
mscl_dasg(int *, int *, int *);      /* MS C Large Model */
tcs_dasg(int *, int *, int *);       /* Turbo C Small Model */
tcm_dasg(int *, int *, int *);       /* Turbo C Medium Model*/
tcl_dasg(int *, int *, 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;
int Params[16];
int Flag;
```

The Params[] array index values are 0 thru 15, inclusive.

An Example

To call MODE 0 of the DASG Driver from an MS C Medium Model program, your commands would be

```
    :
Mode=0;
Flag=0;
Params[0]=0X300;
Params[1]=7;
Params[2]=3;
mscm_dasg (&Mode, Params, &Flag);
if (Flag !=0)
   {
     printf("Mode %d Error Flag = %d\n", Mode, Flag);
     exit(1);
   }
    :
```

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

Linking To The Driver

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

```
LINK your-program,,,DASG.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 DASG.LIB correctly.

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

DMA Memory Buffer Allocation

MODE 6 requires memory buffer represented by a special DMA buffer address in form of one 16-bit value (the segment; to complete the address, the Driver assumes an Offset of 0). This special DMA address is available by calling MODE 23 (for detail, refer to the DAS-16/F/G and uCDAS-16G User Guides).

Far Pointer Manipulation

MODE 4, 5, 9, 18, and 20 allows FAR pointers to be passed in the user Params[] integer array. The Segment and Offset of all FAR pointers(32 bits) in C may be retrieved using C macro: FP_OFF and FP_SEG. Refer to your C Run-time library manual for more detail.

For example,

```
int far *       samp_buf_ptr,*chan_buf_ptr;
unsigned int    dest_seg;
int             Mode, Params[16], Flag;

        .
        .
        .

Mode=9;
Params[0] = 50;                     /* Number of Samples to unpack      */
Params[1] = -1;                     /* use FAR pointer                  */
Params[2] = 0;                      /* Start with sample 0              */
Params[3] = FP_OFF(samp_buf_ptr);   /* Offset address of data array     */
Params[4] = FP_SEG(samp_buf_ptr);   /* Segment address of data array    */
Params[5] = FP_OFF(chan_buf_ptr);   /* Offset address of channel array  */
Params[6] = FP_SEG(chan_buf_ptr);   /* Segment address of channel array */
Params[7] = 0;                      /* Memory offset                    */
Params[8] = dest_seg;               /* Memory segment                   */
mscl_dasg(&Mode, Params, &Flag);
if (Flag !=0)
{
       printf("Mode 9 Error Flag = %d\n",Flag);
       exit(1);
}
```

3.3 MICROSOFT PASCAL

Function Prototypes

In your Application Code, declare the following function prototype:

```
FUNCTION MSP_DASG(VAR Mode:integer;VAR Params:PArray;VAR Flag:integer):integer; external;
```

The Call Parameters

Declare the Mode Call parameters as follows:

```
TYPE
  PArray = array [1..16] of word ;

VAR
  Params     : PArray;                (* MODE PARAM ARRAY *)
  Mode,Flag  : integer;               (* MODE CALL VARIABLES *)
```

The Params[] array index values are 1 thru 16 inclusive. Note that if PArray TYPE is declared as [0..15], the index value starts at 0.

An Example

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

```
    :
Mode := 0;
Params[1] := 768;                    (* BOARD ADDRESS *)
Params[2] := 7;                      (* INTERRUPT LEVEL *)
Params[3] := 3;                      (* DMA LEVEL *)

MSP_DASG (Mode, Params, Flag);
if (Flag <> 0) then ReportError;
    :
```

where *ReportError* is a previously declared procedure that displays an error message and terminates the program. Refer to the Microsoft PASCAL example program (in the Distribution Software) for more detail.

Linking To The Driver

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

```
LINK your-program,,,DASG.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 DASG.LIB correctly.

DMA Memory Buffer Allocation

MODE 6 requires memory buffer represented by a special DMA buffer address in form of one 16-bit value (the segment; to complete the address, the Driver assumes an Offset of 0). This special DMA address is available by calling MODE 23 (for detail, refer to the DAS-16/F/G and uCDAS-16G User Guides).

Far Pointer Manipulation

MODEs 4, 5, 9, 18, and 20 allow FAR pointers to be passed in the user Params[] integer array. The Segment and Offset of all FAR pointers (32 bits) in MS PASCAL may be retrieved using the built-in operator ADS and the *.S* and *.R* sub-operators. Refer to your MS Pascal Run-time library manual for more detail.

For example,

```
Type
 DArray = array [1..5000] of integer;
var
 data,chan        : DArray;
```

```
Mode := 9 ;
Flag := 0 ;
Params[1] :=5000;
Params[2] :=-1;
Params[3] :=0;
Params[4] :=ORD((ADS data).R);
Params[5] :=ORD((ADS data).S);
Params[6] :=ORD((ADS chan).R);
Params[7] :=ORD((ADS chan).S);
Params[8] :=0;
Params[9] :=dest_seg;
Result := msp_dasg(Mode,Params,Flag) ;
if(result <> 0) then
Begin
  ReportError;
  Return;
End;
```

3.4 BORLAND TURBO PASCAL

The Call Label

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

```
FUNCTION TP_DASG(VAR Mode:integer;VAR Params:PArray;VAR Flag:integer):integer; external;
```

The Call Parameters

Declare the Mode Call parameters as follows:

```
TYPE
     PArray = array [1..16] of word;
VAR
     Params          : PArray;       (* MODE PARAM ARRAY *)
     Mode,Flag       : integer;      (* MODE CALL VARIABLES *)
     Result          : integer;      (* MODE CALL RETURN VALUE *)
```

The Params[] array index values are 1 thru 16, inclusive. Note that if PArray TYPE is declared as [0..15], the index values start at 0.

An Example:

To call Mode 0 of the DASG Driver from Turbo Pascal Application Code:

```
    :
Mode := 0;
Params[1] := 768;                          (* BOARD ADDRESS *)
Params[2] := 7;                            (* INTERRUPT LEVEL *)
Params[3] := 3;                            (* DMA LEVEL *)
Result := TP_DASG (Mode, Params, Flag);
if (Result <> 0) then ReportError;
    :
```

Where *ReportError* is a previously declared procedure that displays an error message and terminates the Application Code. Refer to the Turbo Pascal example program provided for more detail.

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
```

DMA Memory Buffer Allocation

MODE 6 requires memory buffer represented by a special DMA buffer address in form of one 16-bit value (the segment; to complete the address, the Driver assumes an Offset of 0). This special DMA address is available by calling MODE 23 (for detail, refer to the DAS-16/F/G and uCDAS-16G User Guides).

Far Pointer Manipulation

MODEs 4, 5, 9, 18, and 20 allow FAR pointers to be passed in the user Params[] integer array. The Segment and Offset of all FAR pointers (32 bits) in Turbo Pascal may be retrieved using built-in function *Ofs* and *Seg*. Refer to your Turbo Pascal Run-time library manual for more detail.

For example,

```
Type
 PArray = array [1..16]of integer ;
 DArray = array [1..5000] of integer;
var
 Params              : PArray ;
 data,chan        : DArray;
 Mode,Flag           : integer;

        .
        .
        .

  Mode := 9 ;
  Flag := 0 ;
  Params[1] :=5000;
  Params[2] :=-1;
  Params[3] :=0;
  Params[4] :=ofs(data);
  Params[5] :=seg(data);
  Params[6] :=ofs(chan);
  Params[7] :=seg(chan);
  Params[8] :=0;
  Params[9] :=dest_seg;
  Result := tp_dasg(Mode,Params,Flag) ;
```

3.5 MICROSOFT FORTRAN

The Software Driver Call Label

The call label *msf_dasg* 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 i(16)              !Parameter Array
integer*2 mode               !Mode number
integer*2 flag               !Return error flag
```

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,

```
mode=0
i(1)=768                          ! Board Address
i(2)=7                            ! Interrupt Level
i(3)=3                            ! DMA Level
call msf_dasg(mode, i(1), Flag)
if (flag .NE. 0) then
    print *,'Mode = ',mode,' Error # ',flag
endif
```

Linking To The Driver

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

```
LINK your-program,,,DASG.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 DASG.LIB correctly.

DMA Memory Buffer Allocation

MODE 6 requires memory buffer represented by a special DMA buffer address in form of one 16-bit value (the segment; to complete the address, the Driver assumes an Offset of 0). This special DMA address is available by calling MODE 23 (for detail, refer to the DAS-16/F/G and uCDAS-16G User Guides).

Far Pointer Manipulation

MODEs 4, 5, 9, 18, and 20 allow FAR pointers to be passed in the user Params() integer array. The Segment and Offset of all FAR pointers that are to represent a memory buffer in FORTRAN may be retrieved using the FORTRAN intrinsic function *LOCFAR()* and some simple calculation. Refer to your FORTRAN Run-time library manual and our FORTRAN example programs for more detail.

For example,

```
      integer*2 data(5000),params(16)
      integer*2 mode,flag
        integer*2 array_off,array_seg
        integer*4 array_addr

                .
                .
                .

C       Get segment and offset address of data array
        array_addr=LOCFAR(data(1))
        array_seg=array_addr/#10000
        array_off=array_addr-(array_seg*#10000)

        mode=9
        flag=0
      params(1)=5000
        params(2)=-1
        params(3)=0
        params(4)=array_off
      params(5)=array_seg
        params(6)=0
        params(7)=0
        params(8)=0
      params(9)=dest_seg
        call fdasg(mode, params(1), flag)
```

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 BASDASG (MD%, BYVAL PARAMS%, FLAG%)
```

Note that all subroutine DECLAREs in your program MUST be made before any $DYNAMIC arrays are allocated. $DYNAMIC data is data that is allocated space in the FAR heap, outside the default data segment. All arrays used for data acquisition must be declared as $DYNAMIC; QuickBasic assumes $STATIC data (Default data segment) unless otherwise specified.

The Call Parameters

Declare the Mode Call parameter array D%(15) as follows:

```
DIM D%(15)
COMMON SHARED D%()
```

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

An Example

To initialize your DAS-16 board, use MODE 0 as follows:

```
    :
180 MD% = 0                          'initialize mode
190 FLAG% = 0                        'declare error variable
200 D%(0) = &H300                    'Card Base Address
210 D%(1) = 7                        'Interrupt Level
220 D%(2) = 3                        'DMA Level
230 D%(3) = 0                        'Auto-Calibration = Off
240 CALL BASDASG (MD%, VARPTR(D%(0)), FLAG%)
250 IF FLAG% <> 0 THEN PRINT "MODE 0 Error # "; FLAG% : STOP
    :
```

Linking To The Driver

The QuickBASIC interface consists three separate Drivers:

DASG.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 DASG <your-program>
```

DASGX.QLB — This is identical to DASG.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 DASGX <your-program>
```

DASG.LIB — Link to this library when you want to make a stand-alone EXE program from your QuickBASIC (all versions) source. To create such a program, use BC and LINK the QuickBASIC compiler and linker as follows:

```
BC <your-program>.bas /o;
LINK <your-program>,,,DASG.LIB;
```

NOTE: All $DYNAMIC data declaration must occur after all COMMON and DECLARE statements in your program. If you get the QB error, COMMON and DECLARE must precede all executable statements; double check the order of all DECLARE, COMMON, and $DYNAMIC declarations.

To link your Application Code to a QuicBASIC Driver, you must specify the Driver in the command line using the Load Switch */L* , as follows:

```
QB /L DASG.QLB
```

DMA Memory Buffer Allocation

MODE 6 requires memory buffer represented by a special DMA buffer address in form of one 16-bit value (the segment; to complete the address, the Driver assumes an Offset of 0). This special DMA address is available by calling MODE 23 (for detail, refer to the DAS-16/F/G and uCDAS-16G User Guides).

Far Pointer Manipulation

QuickBASIC provides the built-in functions VARPTR and VARSEG for obtaining the Offset and Segment of a given variable. If the variable is declared in the $STATIC area (by default), VARSEG returns the default data segment. If the variable is declared as $DYNAMIC, then it is placed in the FAR heap and VARSEG for such a variable returns a unique Segment value outside the default data segment.

For example,

```
DIM DT%(1000)                ' Data array used by MODE 9
DIM CH%(1000)                ' Channel array used by MODE 9

     .
     .
     .

MD% = 9                           'mode 9 - data transfer
PARAM%(0) = 1000                  'number of words to transfer
PARAM%(1) = DMASEG%               'Flag to look for SEG:OFF pairs below...
PARAM%(2) = 0                     'start transferring at 1st sample
PARAM%(3) = VARPTR(DT%(0))        'Offset of DT%(*) array
PARAM%(4) = VARPTR(CH%(0))        'Offset of CH%(*) array
CALL BASDASG(MD%, VARPTR(PARAM%(0)), FLAG%)   'make transfer
IF FLAG% <> 0 THEN PRINT "Mode 9 data transfer error # "; FLAG%: STOP
```

■ ■ ■

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: Driver not initialized.

Detail:	This error may be issued from any mode; it indicates that the driver must be re-initialized prior to this step.
Recommended Action:	Call Mode 0 before calling the mode in which this error occurred.

Error 2: MODE number out of range.

Detail:	This error may be issued from any mode; it indicates the the specified mode number is not allowed.
Recommended Action:	Check the mode number; it should be within 0 to 25.

Error 3: Base Address out of range.

Detail:	This error is generated by MODE 0 whenever the Base Address specified is not allowed.
Recommended Action:	Specify a Base Address within the range 256 (100h) and 1008 (3F0h), inclusive.

Error 4: Interrupt Level out of range.

Detail:	This error is generated by MODE 0 whenever the specified Interrupt Level is not allowed. Use a level that does not conflict with other devices in your system.
Recommended Action:	Supply MODE 0 with an Interrupt Level in the range 2 thru 7, inclusive.

Error 5: DMA channel not 1 or 3.

Detail:	This error is generated by MODE 0 whenever the specified DMA Channel is not allowed. Note that, unlike the Interrupt Level selection, the DMA Channel selected here must match the DMA Channel Switch position on the board!
Recommended Action:	Supply MODE 0 with DMA Channel 1 or 3.

Error 6: Differential scan limits out of range.

Detail:	This error is generated by MODE 1 whenever the differential input channel scan limits are out of range.
Recommended Action:	Check the channel numbers passed to MODE 1. They must be within 0 thru 7, inclusive.

Error 7: Single Ended scan limits out of range.

Detail:	This error is generated by MODE 1 whenever the Single Ended input channel scan limits are out of range.
Recommended Action:	Check the channel numbers passed to MODE 1. They must be within 0 thru 15, inclusive.

Error 8:

Detail:

Recommended Action:

Error 9: No End-Of-Conversion (EOC) signal from A/D subsystem.

Detail:	This error is generated by MODE 3 whenever an expected EOC signal from the DAS-16 is not recieved. This indicates a hardware failure.
Recommended Action:	Check the board's Base Address.

Error 10: One or both counter values out of range.

Detail:	This error is generated by MODE 17 whenever one or both counter divisors are specified as 0 or 1.
Recommended Action:	Check both counter divisors for values within the range 2 thru 65535, inclusive.

Error 11: Number of conversions out of range.

Detail:	This error is generated by MODEs 4, 5, 6, or 20 whenever the number of conversions specified is out of range. Note that the number of conversions is limited to 0 thru 32767, inclusive; where, 0 is used to specify the 32768 – the maximum number of conversions possible.
Recommended Action:	Check the number of conversions.

Error 12: Counter 0 Configuration Number out of range.

Detail:	This error is generated by MODE 10 whenever Counter 0 configuration number is not within the allowed range of 0 thru 5 inclusive. Refer to section ??? for detail on the different configuration modes.
Recommended Action:	Check the configuration number.

Error 13: Digital output data out of range.

Detail:	This error is generated by MODE 13 whenever the output data value is not within 0 thru 15 inclusive. There are four (4) Digital Output lines available on the DAS-16.
Recommended Action:	Check the Digital output value.

Error 14: Analog output (D/A) data out of range.

Detail:	This error is generated by MODEs 15 and 16 whenever the D/A value specified is out of the allowed range of 0 thru 4095, inclusive.
Recommended Action:	Check the DAC value.

Error 15: DAC channel number not 0 or 1.

Detail:	This error is generated by MODEs 15 and 18 whenever the DAC channel number specified is not allowed. the DAS-16 family of boards supports two DAC channels: 0 and 1.
Recommended Action:	Check the DAC channel.

Error 16: Counter 0 Read operation number out of range.

Detail:	This error is generated by MODE 12 whenever an illegal Counter Read Operation is attempted. Two types of Reads are allowed: 0 = Normal Read, and 1 = Latched Read.
Recommended Action:	Check Read operation number.

Error 17: Starting sample number out of range.

Detail:	This error is generated by MODE 9 whenever the starting sample number is out of range. The number must within 0 thru 32767, inclusive.
Recommended Action:	Check the starting sample number.

Error 18: Requested sample count out of range.

Detail:	This error is generated by MODEs 9 and 18 whenever the number of samples is out of range. The sample count for these modes must within 1 thru 32767, inclusive.
Recommended Action:	Check the sample count.

Error 19: Trigger Mode must be 0 or 1.

Detail:	This error is generated by a multiple of MODEs whenever the A/D Conversions Trigger source is not valid.
Recommended Action:	Specify 0 for Internal Trigger or 1 for External Trigger.

Error 20: Interrupt or DMA Mode(s) already active.

Detail:	This error is generated by MODEs 5, 6, 18, or 20 whenever an Interrupt or DMA Mode is attempted while another is already active. Interrupt and DMA acquiistion modes share the same resources and can not be used simultanuously.
Recommended Action:	Use Mode 7 to terminate current acquisition operation.

Error 21: DMA Page Wrap.

Detail:	This error is generated by MODE 6 whenever the combination of the DMA Segment (DIO%(1)) and the Number of Samples (DIO%(0)) create a Page Wrap condition in the PC's DMA Controller. The DMA operation is never started.
Recommended Action:	Use MODE 23 to determine a 'good' DMA Segment to supply MODE 6.

Error 22: Board not present.

Detail:	The Board Presence Test performed in MODE 0 has failed. The Base Address passed to MODE 0 must match the actual address selected on the board.
Recommended Action:	Check the board's Base Address.

Error 23: Analog Trigger Channel out of range.

Detail:	This error is generated by MODE 19 whenever the specified Analog Input Channel is out of range. This channel number must between 0 and 7 (Differential) or 0 and 15 (Single Ended), inclusive.
Recommended Action:	Check the channel number.

Error 24: Analog trigger data out of range.

Detail:	This error is generated by MODE 19 whenever trigger value is out of range.
Recommended Action:	Specify a value between 0 and 4095 (Unipolar) or -2048 and 2047 (Bipolar), inclusive.

Error 25: Analog trigger slope not 0 or 1.

Detail:	For MODE 19, the user is allowed to specify 0 to trigger on a Positive slope or 1 for a Negative slope.
Recommended Action:	Sepcify 0 or 1 for the Trigger Slope.

Error 26: Gain Code out of range.

Detail:	This error is generated by MODEs 6, 19, and 21 whenever the specified Gain Code is out of range. The allowed Gain Codes are 0 thru 3, inclusive. Note that an Analog Input Gain Code is relevant only to the DAS-16G1 and G2 boards. This value is ignored when specified for other boards.
Recommended Action:	Specify a valid Gain Code.

Error 27: Old BASIC program compatibility problem.

Detail:	You will get this error when you use the new DAS-16 driver/library feature (Rev 4.10 or newer) from a BASIC program written for a previous revision of the driver.
Recommended Action:	Replace all: *CALL DAS16* with *CALL DASG* , and increase your parameter array D%(4) dimention to D%(15).

Error 28:

Detail:

Recommended Action:

Error 29: Possible memory index overflow.

Detail:	This is a rare error that is generated by MODEs 4 or 5. If you are unlucky enough to see it, reduce your number of samples by 16!
Recommended Action:	Reduce your number of samples by 16, and retry.

Error 30: Memory allocation/deallocation error.

Detail:	This error is generated by MODE 23 or MODE 24 whenever memory can not be allocated or deallocated through DOS.
Recommended Action:	If error is generated from Mode 23, reduces the size of allocating memory. If error is generated from Mode 24, make sure the right memory segment to be deallocated is passed to the driver correctly.

Error 31: 8255A PPI control word out of range (Mode 25 for DAS-16A only!).

Detail:	This error is generated by MODE 25 whenever the control word for PPI 8255A is out of range.
Recommended Action:	Check the control word passed to MODE 25. It must be within 0 -255.

Error 32: Data present flag out of range (Mode 25 for DAS-16A only!).

Detail:	This error is generated by MODE 25 whenever the Data Present flag is neither 0 (data is presented) nor 1 (data is not presented).
Recommended Action:	Make sure the flag passed to MODE 25 is either a *0* or *1* .

Error 33: 8255A PPI mode out of range (Mode 25 for DAS-16A only!).

Detail:	This error is generated by MODE 25 whenever the 8255A PPI mode is out of range.
Recommended Action:	Make sure the mode number passed to MODE 25 is either a 0, 1, or a 2.

Error 34: Port A direction out of range (Mode 25 for DAS-16A only!).

Detail:	This error is generated by MODE 25 whenever the Port A direction is invalid.
Recommended Action:	Make sure the port direction is 00h, 01h, 10h, or 11h.

Error 35: Port B direction out of range (Mode 25 for DAS-16A only!).

Detail:	This error is generated by MODE 25 whenever the Port B direction is invalid.
Recommended Action:	Make sure the port direction is 00h, 01h, 10h, or 11h.

Error 36: Port C (Hi) direction out of range (Mode 25 for DAS-16A only!).

Detail:	This error is generated by MODE 25 whenever the Port C (Hi) direction is invalid.
Recommended Action:	Make sure the port direction is either 00h, 01h, 10h, or 11h.

Error 37: Port C (Lo) direction out of range (Mode 25 for DAS-16A only!).

Detail:	This error is generated by MODE 25 whenever the Port C (Lo) direction is invalid.
Recommended Action:	Make sure the port direction is either 00h, 01h, 10h, or 11h.

Error 38: Incorrect Port B (I/O) mode (for 8255A PPI Mode 2 only!; Mode 25 for DAS-16A only!).

Detail:	This error is generated by MODE 25 whenever the Port B (I/O) mode is invalid.
Recommended Action:	Make sure the mode is either 0, 1.

Error 39: 8255A PPI chip is not initialized (Mode 26 for DAS-16A only!).

Detail:	This error is generated by MODE 26 whenever this MODE is called before calling MODE 25.
Recommended Action:	Calls MODE 25 first.

Error 40: Port A output data out of range (Mode 26 for DAS-16A only!).

Detail:	This error is generated by MODE 26 whenever the output data to Port A is out of range.
Recommended Action:	Make sure the output data is between 0 and 255.

Error 41: Port B output data out of range (Mode 26 for DAS-16A only!).

Detail:	This error is generated by MODE 26 whenever the output data to Port A is out of range.
Recommended Action:	Make sure the output data is between 0 and 255.

Error 42: Port C output data out of range (Mode 26 for DAS-16A only!).

Detail:	This error is generated by MODE 26 whenever the output data to Port A is out of range.
Recommended Action:	Make sure the output data is between 0 and 255.

■ ■ ■