PCF-40

User Guide

for the

PCF-40

PASCAL, C, & FORTRAN

Callable Drivers

For The

DAS-40

Revision B - December 1992

Part Number: 24405

KEITHLEY DATA ACQUISITION - Keithley MetraByte/Asyst

440 Myles Standish Blvd., Taunton, MA 02780
TEL. 508/880-3000, FAX 508/880-0179

Contents

■ ■ ■

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The PCF-40 is a software package for programmers using Pascal, C, FORTRAN, and QuickBASIC to write data acquisition and control routines for the DAS-40 Data Acquisition Board. This package is normally supplied on 5.25" low-density diskettes but is also available (upon request) on 3.5" diskette(s). Contents of the package include the following:

- DAS-40 Software Driver
- Language Interface Modules
- Source code for Driver and Language Interface Modules
- Miscellaneous documentation (.DOC) files
- Example program files in all supported languages

1.2 SUPPORTED LANGUAGES

The PCF-40 supports all memory modules of the following languages:

- Microsoft C (V4.0 - 7.0)
- Microsoft Quick C (V1.0 - 2.0)
- Microsoft Pascal (V3.3 - 4.0)
- Microsoft FORTRAN (V4.0, 4.1)
- Microsoft QuickBASIC (V4.0 and higher)
- Borland Turbo Pascal (V3.0 and higher)
- Borland Turbo C (V1.0 - 2.0)
- Lahey Personal FORTRAN (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

In either of these instructions, the *source* diskette will be the diskette containing your Distribution Software; the *target* diskette will be the diskette you will 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 *PCF40* .

1. After making a directory named *PCF40* , place your Distribution Software diskette in your PC's A Drive and log to that drive by typing **`A:`** .
2. Then, type **`COPY *.*`** *`path`***`\PCF40`** , where *path* is the drive designation and DOS path (if needed) to the PCF40 directory.

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

1.4 LOADING/UNLOADING THE DAS-40 SOFTWARE DRIVER

The DAS-40 Device Driver consists of the two files MDAS40.EXE, and VIPARSE.EXE. As executable files, both are self-loading. Loading and unloading may be performed from the DOS command line or from a batch file. Note that a path may have to proceed the commands unless you are logged into the correct directory.

At the DOS command line, type the following:

```
VIPARSE
```

followed by

```
MDAS40 /C0=file0 /C1=file1
```

Where *file0* and *file1* are optional configuration files for Cards #0 and Card #1, respectively. These files are presumed created via the Distribution Software utility CONFIG40.EXE. Refer to the Section 3.4 (*CHANGING THE FACTORY CONFIGURATION*) for more detail. Specification of switches */C0=* and */C1=* is optional. If you specify no switches, Factory Configuration is assumed for up to two physical boards. If you specify both */C0=* and */C1=* , their order on the command line is irrelevant.

These loading commands may be entered automatically on power-up by adding them to your AUTOEXEC.BAT file. Use any ASCII text editor or a word processor to add the commands in the order shown above.

The DAS40 Device Driver (both files) may also be unloaded from the DOS Command line or from a batch file. Unloading the Device Driver returns the memory it occupies to DOS. The Device Driver files must be unloaded in the following order: first MDAS40.EXE, then VIPARSE.EXE.

To unload MDAS40.EXE, type the following:

```
MDAS40 /U
```

and for VIPRASE.EXE,

```
VIPARSE /U
```

Note that VIPARSE and MDAS40 occupy approximately 7600 bytes and 18000 bytes respectively.

1.5 RUNNING THE EXAMPLE PROGRAMS

The PCF-40 Distribution Software includes several example programs written in the various supported languages. These programs are in source-code and executable (.EXE) forms. To run these programs proceed as follows:

1. With the computer powered down, install your Factory Configured DAS-40 board and then power up. Refer to the appropriate section of your User Guide for hardware installation directions.
2. Install VIPARSE and MDAS40 as described in Section 1.4 of this manual.
3. Type the name of the example program you wish to run (for example, *path***MSCEXMPL** followed by < Enter > where *path* is the drive designation and DOS path (if needed).

Note that most example programs assume that the DAS-40 is factory-configured (the various on-board switches and jumpers are set to factory-default positions. Refer to Section 3.6 of the DAS-40 User Guide for more information. If the Factory Configuration is not adequate for your system (the DMA Channel(s) or the Interrupt Level conflict with other devices in your system), refer to Section 3.4 of the DAS-40 User Guide for instruction on how to re-configure your board.

If you make changes to any of the example programs, refer to the supplied file HOWTOEXE.DOC for steps necessary to recreate a new executable program.

1.6 WRITING YOUR PROGRAM

The example program for the language you are using will provide you with most of the information you need to start your own DAS40-based application program. As an illustration, to create a program using Microsoft C, proceed as follows:

1. Write your program using a text editor or the QuickC Environment.
2. Compile your program using the CL command

   ```
   CL /Ax /c example.c
   ```

   where in the memory-model switch */Ax* , x = S, C, M, or L.
3. Link your program to the Language interface module DAS40.LIB:

   ```
   LINK example,,,DAS40.LIB;
   ```

You now have an executable program ready to run. Repeat all three steps as you modify/fix this program.

■ ■ ■

❑

❑

❑

CHAPTER 2

THE DAS-40 DRIVER

2.1 OVERVIEW

All languages supported by the PCF-40 use the same installable driver, **MDAS40.EXE** . This driver was programmed in Assembly Language with the Microsoft Assembler (MASM), Version 5.1, and consists of the following modules:

- D40drive.ASM
- D40modes.ASM
- D40funct.ASM
- D40utils.ASM
- LU.ASM

The DAS40.EXE *Include* file is **DAS40.DEF** . All of these files are included in your PCF-40 Distribution Software diskette(s).

In addition to the MDAS40 Driver, Interface Modules for various languages are also included in this package. You must LINK your program to the appropriate module for proper communication to the MDAS40 Driver. A typical module occupies approximately 200 bytes. These Modules are as follows:

- DAS40.LIB: Use for mode calls from Pascal, C, FORTRAN, and stand-alone QuickBASIC programs.
- DAS40.BIN: Use for mode calls from BASIC(A).
- DAS40.QLB: Use for mode calls from the QuickBASIC Integrated Development Environment (Ver. 4.0 - 4.5).
- DAS40x.QLB: Use for mode calls from the QuickBASIC Extended Environment (Ver. 7.0).
- DAS40TP.OBJ: Use for mode calls from TURBO Pascal.

The Interface Modules support all MODEs listed here and described in the User Guide. MODEs are as follows:

MODE 0	Initialize and Query Configuration of an installed DAS-40.
MODE 1	Assign a buffer and sample count to each DMA level for both A/D and D/A. Used before calling MODES 4 or 5.
MODE 2	Setup for N A/D or D/A conversions for transfer via DMA. Used before calling MODES 4 or 5.
MODE 3	Setup global start and stop channels and overall gain.
MODE 4	Start A/D conversions and transfer via DMA as setup in MODEs 1,2.
MODE 5	Start D/A conversions and transfer via DMA as setup in MODEs 1,2.
MODE 6	Setup Pacing clock rate.

MODE 7	Perform one A/D conversion on specified channel.
MODE 8	Perform one D/A conversion on DAC channel (1, 2, or both).
MODE 9	Monitor ADC/DAC DMA transfer status.
MODE 10	Setup for Digital Input/Output.
MODE 11	Input Byte/Word.
MODE 12	Set/Query current DAS-40 Card number
MODE 13	Terminate Current DMA transfer - A/D In or D/A Out.
MODE 14	Output Byte/Word.
MODE 15	Tag channel numbers to data.
MODE 16	A/D Triggered Block Scan (DMA).
MODE 17	Transfer A/D Data to a BASIC Array.
MODE 18	Allocate Memory buffer for DMA.
MODE 19	Free DMA Memory Buffer.

2.2 CALLING THE DAS-40 DRIVER

The DAS-40 Driver is called through a single and unique label to the language/model you are using.

The following is a list of CALL routine labels available in the Language Interface Modules.

DAS40.LIB :

mscs_das40	For Calls from Microsoft C, Small Model
mscm_das40	For Calls from Microsoft C, Medium Model
mscl_das40	For Calls from Microsoft C, Large Model
tcs_das40	For Calls from TURBO C, Small Model
tcm_das40	For Calls from TURBO C, Medium Model
tcl_das40	For Calls from TURBO C, Large Model
msp_das40	For Calls from Microsoft Pascal
basdas40	For Calls from Microsoft QuickBASIC
msf-das40	For Calls from Microsoft FORTRAN
lhy_das40	For Calls from Lahey FORTRAN

TPDAS40.OBJ :

tp_das40	For Calls from TURBO Pascal

DAS40.BIN :

das40	For Calls from BASIC(A)

Regardless of the language/model you are using, each call to its label requires the specification of three input parameters, as follows:

MODE	A 16-bit integer quantity containing the number of the MODE to be executed by the DAS-40 driver.
PARAM	An array of ten 16-bit integers containing MODE-dependent arguments required for the successful execution of the MODE. These are also used to return information.
FLAG	A 16-bit integer quantity that contains a number representing any Error Code reported by the DAS-40 driver. Note that the least significant eight bits contain the error number, while the most significant eight bits contain the current MODE number.

Refer to Chapter 3 for details on how to declare and use these variables in the language you are using.

The following is a code fragment on how to declare and use the call parameters.

```
      :
int  Mode;
int  Flag;
int  Params[10];
      :
Mode = 0;
Flag = 0;
Params[0] = 0;
mscl_das40(&Mode, Params, &Flag);
if (Flag != 0)
  printf ("**** Error %u detected in mode 0", Flag & 0xff);
      :
```

2.3 MODIFYING THE DAS-40 DRIVER

DAS40.EXE is a functional and fully tested software driver that supports all documented MODEs of operation, as described in the DAS-40 User Guide. To create a new version of this driver, your development tools must include the assembler (MASM) and the program Linker (LINK), and they must be installed on your system and accessible from your current directory. To assemble a module (D40FUNCT.ASM for example), proceed as follows:

1. Your command-line entry should be **MASM D40FUNCT;**
2. Once all modules are assembled without errors, use the following command to link these modules:

   LINK D40DRIVE+D40MODES+LU+D40FUNCT+D40UTILS,DAS40,DAS40,VI.LIB;

The successful outcome of the LINK step is the installable DAS-40 driver **DAS40.EXE** . Refer to the DAS-40 manual section *LOADING THE DAS-40 DRIVER* for more detail.

Note, the VI.LIB is the interface library to VIPARSE and is included on the distribution diskette(s).

2.4 MODIFYING THE DAS-40 LANGUAGE INTERFACE MODULES

The Interface Modules support all MODEs described in the DAS-40 Manual. To create a new version of any of these modules, your system must have the necessary development tools, including:

MASM.EXE	Microsoft Assembler

LINK.EXE	Microsoft Linker
LIB.EXE	Microsoft Librarian

Other utilities will be specified below, as necessary. All of these tools must be installed on your system and accessible from your current directory. Source code for these modules is in the files DAS40PCF.ASM and DAS40TP.ASM (TURBO Pascal only), which are in the Distribution Software.

DAS40.BIN - BASIC(A) Interface

The BASIC(A) interface is the BLOAD'able DAS40.BIN file. To create this file, you must have access to the following utilities:

EXE2BIN.EXE	A Microsoft **.EXE** to **.COM** file conversion utility.
MAKEBIN.EXE	A **.COM** to **.BIN** file-conversion utility (supplied with the PCF-40 Distribution Software).

To create DAS40.BIN, use the following commands:

```
MASM /DBIN=1 DAS40PCF;
LINK DAS40PCF,,,;
EXE2BIN DAS40PCF.EXE DAS40.COM
MAKEBIN DAS40.COM
```

All four steps must be successful. Note that the LINKing operation generates the warning:

LINK : Warning L4021 : no stack segment

This warning is irrelevant; ignore it and proceed to the next step.

DAS40TP.OBJ - TURBO PASCAL Interface

The TURBO PASCAL interface is the DAS40TP.OBJ file and is *linked* directly into the TURBO PASCAL module, by using the $L compiler directive. To create this file you must have access to the following utilities:

TASM.EXE - TURBO Assembler

To create DAS40TP.OBJ, use the following command:

```
TASM DAS40TP;
```

DAS40.QLB - QuickBASIC Integrated Environment (V4.5) Interface

The interface for QuickBASIC Integrated Environment (up to Ver 4.5) is made using the Quick Library file DAS40.QLB. This file is specified on the command line with the load */L* switch. For example, **QB /L DAS40.QLB** .

To create the DAS40.QLB file you must have access to the utility *BQLB45.LIB* , which is the QuickBASIC Integrated Environment Library. Use the following entries:

```
MASM /DBIN=0 DAS40PCF;
LINK /q DAS40PCF,DAS40,,BQLB45;
```

DAS40X.QLB - QuickBASIC Integrated Environment (V7.0) Interface

To create a QLB library compatible with QuickBASIC Version 7.0, follow the procedure described for QB V4.5; however, link with QBXQLB.LIB instead of BQLB45.LIB, as follows:

```
LINK /q DAS40PCF,DAS40x,,QBXQLB;
```

Note that the output file (from the linker) is renamed *DAS40x.QLB* to avoid incompatibility conflicts with QuickBASIC 4.5. To create a stand-alone QuickBASIC *.EXE* program, see the discussion that follows on the PASCAL, C, and FORTRAN Interface.

DAS40.LIB - PASCAL, C, And FORTRAN Interface

The interface for Pascal, C, or FORTRAN (and stand-alone QB .EXEs) uses the library file *DAS40.LIB* . This file is linked to the user's program with the following commands:

```
MASM /DBIN=0 DAS40PCF;
LIB DAS40-+DAS40PCF;
```

Note that DAS40PCF.OBJ is the only module in the output library *DAS40.LIB* . This was done intentionally at development time to eliminate possible confusion between the different versions of DAS40PCF.OBJ (/DBIN=0 or /DBIN=1 ?). However, you may want to bypass the LIB step and add the DAS40PCF.OBJ to your linker input list.

■ ■ ■

❑

❑

❑

CHAPTER 3

THE LANGUAGE INTERFACE

3.1 OVERVIEW

Although your DAS-40 software driver modes are identical for all supported languages, there are differences in the manner in which parameters and parameter values are passed to these modes from different languages. Specifically, the items causing the most confusion are as follows:

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

Refer to the appropriate section below for details on performing the Driver Mode Calls from the language you are using. The language sections below contain brief code fragments for illustration only; refer to the example program(s) on your distribution diskette(s) for more information.

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 programs that fully utilize your DAS-40 product.

The Software Driver Mode Call Labels

In your program, declare one of the following function prototypes:

```
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 */
tcm_dasg(int *, int *, int *);        /* Turbo C Medium Model*/
tcl_dasg(int *, int *, int *);        /* Turbo C Large Model */
```

These function prototypes may be optionally proceeded by the C keyword *extern* .

The Call Parameters

Declare the mode call parameters as follows:

```
int Mode;
int Params[10];
int Flag;
```

The Params[] array index values are 0 thru 9 inclusive.

An Example

To call MODE 0 of the DAS-40 driver from an MS C Medium Model program,

```
     :
Mode=0;
Flag=0;
Params[0]=0;
mscm_das40 (&Mode, Params, &Flag);
if (Flag !=0)
   {
     printf("Mode %d Error Flag = %d\n", Mode, Flag & 0xff);
     exit(1);
   }
    :
```

Note that specifying *Params* in the call statement is the same as *&Params[0]*.

Linking To The Driver/Interface Module

Once you have your C program written, you must compile and LINK it to the Interface Module, DAS40.LIB. DAS40.LIB is where the label *mscm_das40* label resides.

For example, if your program is in the Medium Model,

```
   CL /AM /c <your-program>.C
   LINK <your-program>.OBJ,,,DAS40.LIB;
```

If no errors are reported, you now have the executable file <your-program>.EXE that is ready to test. In the case that errors such as Unresolved External(s) are reported by the LINKer, then you must verify that you have LINKed to DAS40.LIB correctly.

NOTE: Be sure to use the correct Call Label for the Memory Model you are using. For example, use *mscm_das40* when you compile your program using the Medium Memory Model i.e using compiler switch /AM.

DMA Memory Buffer Allocation

MODE 1 requires the specification of one or two far pointers to DMA buffers each in the form of two 16-bit values: Segment and Offset. These pointers should be previously obtained via calls to MODE 18. The following C program fragment illustrates the use of Modes 18 and 1:

```
/* ALLOCATE A DMA BUFFER */
Mode = 18;
ErrFlag = 0;
Params[0] = Asamples;
mscl_das40 (&Mode, Params, &ErrFlag);
if (ErrFlag != 0) ProcessErr();
```

```
/* SAVE ACTUAL BUFFER SIZE, SEGMENT AND OFFSET */
Asamples = Params[0];
BufAOffset = Params[1];
BufASegment = Params[2];
ActBufAOffset = Params[3]
ActBufASegment = Params[4];

/* SPECIFY BUFFER FOR DMA CONTROLLER */
Mode = 1;
ErrFlag = 0;
Params[0] = BufAOffset
Params[1] = BufASegment
Params[2] = Asamples;
:
```

Note that the contents of Params[1] and Params[2] as returned by Mode 18 are passed to Mode 1; while the contents of Params[3] and Params[4] should be later used by Mode 19 to free the allocated memory block.

Refer to your DAS-40 User Guide for a complete description of the Modes mentioned here.

Note for Borland Turbo C/C++ users

Similar to 'allocmem()' in the Borland Turbo C/C++ run-time library function, MODE 18 uses DOS' INT 0x21 function 0x48 to allocate a free memory block from the far heap. According to the Borland Turbo C/C++ Library Reference, allocmem() function remarks: 'allocmem and _dos_allocmem can not coexist with malloc.' Furthermore, MODE 18, like allocmem(), can not coexist with all C/C++ functions that indirectly use malloc such as fopen(), fprintf(), etc... this restriction applies to all Borland Turbo C/C++ versions.

However, as the supplied example TCexamp2.c illustrates, Turbo C functions such as 'fopen()' may be safely used before the first call to MODE 18.

Far Pointer Manipulation

MODEs 1, 2, and 17 require that FAR pointers be passed in the user Params[] integer array. The Segment and Offset of all FAR pointers (32 bit) in C may be retrieved using the built-in functions: FP_OFF and FP_SEG. Refer to your C Run-time library manual for more detail.

The following code fragment illustrates how to pass the addresses of user arrays Chans[] and Gains[] to Mode 2:

```
int far *pChans = &Chans[0];
Mode = 2;
Params[0] = (int) FP_OFF(pChans);
Params[1] = (int) FP_SEG(pChans);
Params[2] = (int) FP_OFF(pGains);
Params[3] = (int) FP_SEG(pGains);
```

3.3 MICROSOFT PASCAL

The Software Driver Mode Call Labels

In your program, declare the following function prototype:

```
FUNCTION MSP_DAS40(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..10] 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 through 10 inclusive. Note that if PArray TYPE is declared as [0..9] then the index value starts at 0.

An Example

To call MODE 0 of the DAS-40 driver from MS Pascal program,

```
    :
Mode := 0;
Params[1] := 0;                    (* BOARD NUMBER 0 *)
Result := MSP_DAS40 (Mode, Params, Flag);
if (Result <> 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/Interface Module

Once you have your MS Pascal program written you must compile and LINK it to the Interface Module, DAS40.LIB. DAS40.LIB is where the label *MSP_DAS40* label resides.

For example,

```
     PL <your-program>.pas;
     LINK <your-program>,,,DAS40.LIB;
```

If no errors are reported, you now have the executable file <your-program>.EXE that is ready to test. In case errors such as Unresolved External(s) are reported by the LINKer, then you must determine whether you LINKed to DAS40.LIB correctly.

DMA Memory Buffer Allocation

MODE 1 requires the specification of one or two far pointers to DMA buffers each in the form of two 16-bit values: Segment and Offset. These pointers should be previously obtained via calls to MODE 18. The following MS Pascal program fragment illustrates the uses of Modes 18 and 1:

```
(* ALLOCATE MEMORY FOR DMA BUFFER *)
Asamples := 20000;
Mode := 18;
Params[1] := Asamples;
Result := MSP_DAS40(Mode,Params,Flag) ;
if(result <> 0) THEN ReportError;
```

```
(* SAVE RETURNED VALUES *)
Asamples    := Params[1];
AOffset     := Params[2];
ASegment    := Params[3];
AOffActual  := Params[4];
ASegActual  := Params[5];

(* PASS ADDRESS OF DMA BUFFER TO MODE 1 *)
Mode := 1;
Params[1]  := AOffset;
Params[2]  := ASegment;
Params[3]  := Asamples;
```

Note that the contents of Params[1] and Params[2] as returned by Mode 18 are passed to Mode 1; while the contents of Params[3] and Params[4] should be later used by Mode 19 to free the allocated memory block.

Refer to your DAS-40 User Guide for a complete description of the Modes mentioned here.

FAR Pointer Manipulation

MODEs 1, 2, and 17 require that FAR pointers be passed in the user Params[] integer array. The Segment and Offset of all FAR pointers (32 bit) in MS Pascal may be retrieved using the built-in operator ADS and the .S and .R sub-operators! Refer to your MS Pascal manual(s) for more detail.

The following code fragment illustrates how to pass the addresses of user arrays Chans[] and Gains[] to Mode 2:

```
Mode       := 2;
Params[1]  := (ADS Chans).R;
Params[2]  := (ADS Chans).S;
Params[3]  := (ADS Gains).R;
Params[4]  := (ADS Gains).S;
```

3.4 BORLAND TURBO PASCAL

The Software Driver Call Label

The call label TP_DAS40 is provided to use from any Turbo Pascal program; declare this label in your program as follows:

```
FUNCTION TP_DAS40(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..10] 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 10, inclusive. Note that if PArray TYPE is declared as

[0..9], then the index values start at 0.

An Example:

To call MODE 0 of the DAS-40 driver from Turbo Pascal program:

```
      :
Mode  := 0;
Params[1]  := 0;                              (* BOARD NUMBER 0 *)
Result := TP_DAS40 (Mode, Params, Flag);
if (Result <> 0) then ReportError;
      :
```

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

Linking To The Driver Interface Module

The Turbo Pascal Interface Module is supplied in the form of the OBJect file *DAS40TP.OBJ*. This file is linked into your program using the $L Compiler Directive; include this command at the beginning of your program as follows:

```
      {$L DAS40TP}
```

Once included, you are now ready to compile your program:

```
      TPC <your-program>.pas
```

DMA Memory Buffer Allocation

MODE 1 requires the specification of one or two far pointers to DMA buffers each in the form of two 16-bit values: Segment and Offset. These pointers should be previously obtained via calls to MODE 18; refer to the DAS-40 User Guide for complete description of this mode. For this MODE to work correctly from Turbo Pascal, you must set aside some free memory as the DOS far heap; use the $M Compiler Directive to configure this heap.

By default, Turbo Pascal reserves all memory, that is normally given back to DOS (while your program is running), for its own internal Heap Manager. The Heap manager is the part of Turbo Pascal that your program goes through when using the Pascal functions GetMem() and FreeMem(). Using the $M Directive in your program, you can specify the amount of memory to reserve for the program stack and for the Heap Manager. The remaining memory not allocated for programs, data, Stack or the Heap manager constitutes DOS' far heap; the total size of this heap depends on your system.

Examples:

1. To reserve all free memory to the Pascal heap manager:

   $M (16384, 0, 655360)

This is the default where calls to MODE 18 WILL ALWAYS FAIL:

2. To reserve 16K bytes for the stack and 64K bytes for the Pascal heap manager:

$M (16384, 0, 65536)

The remaining memory not allocated for programs, data less 16K bytes for Stack and less 64K bytes for the Heap manager constitutes the new DOS heap.

3. To reserve 16K bytes for stack no space for the heap manager:

$M (16384, 0, 0)

These parameters are recommended when not using any TURBO Pascal heap manager functions (e.g. GetMem(), FreeMem()...) and using MODE 18 in your DAS-40 application programs. Using these $M parameters reserves the remaining memory not allocated for programs or data, less 16K bytes for Stack as the new DOS heap. The following Borland Turbo Pascal program fragment illustrates the uses of Modes 18 and 1:

```
(* ALLOCATE MEMORY FOR DMA BUFFER *)
Mode  := 18;
Params[1]  := 10000;              (* 10K SAMPLES *)
Result  := TP_DAS40(Mode,Params,Flag) ;
if(result <> 0) THEN ReportError;

(* SAVE MODE 18 RESULTS *)
Asamples    := Params[1];
AOffset     := Params[2];         (* DMA BUFFER OFFSET *)
ASegment    := Params[3];         (* DMA BUFFER SEGMENT *)
ActAOffset  := Params[4];         (* MEMORY BLOCK OFFSET *)
ActASegment:= Params[5];          (* MEMORY BLOCK SEGMENT *)

(* SPECIFY PARAMS OBTAINED THRU MODE 18 *)
Mode  := 1;
Params[1]  := AOffset;
Params[2]  := ASegment;
Params[3]  := Asamples;
```

Note that the contents of Params[1] and Params[2] as returned by Mode 18 are passed to Mode 1; while the contents of Params[3] and Params[4] should be later used by Mode 19 to free the allocated memory block.

Refer to your DAS-40 User Guide for a complete description of the Modes mentioned here.

FAR Pointer Manipulation

MODEs 1, 2, and 17 require that FAR pointers be passed in the user Params[] integer array. The Segment and Offset of all FAR pointers (32 bit) in Turbo Pascal may be retrieved using the built-in functions *Ofs* and *Seg*. Refer to your Turbo Pascal manual(s) for more detail.

The following code fragment illustrates how to pass the addresses of user arrays Chans[] and Gains[] to Mode 2:

```
Mode  := 2;
Params[1]  := Ofs  (Chans);
Params[2]  := Seg  (Chans);
Params[3]  := Ofs  (Gains);
Params[4]  := Seg  (Gains);
```

3.5 MICROSOFT FORTRAN

The Software Driver Call Label

The call label *msf_das40* is provided to use from any MS FORTRAN program; no prototype declaration of the label is required.

The Call Parameters

Declare the mode call parameters as follows:

```
integer*2 i(10)              !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 initialize DAC#0 to plus-full-scale (MODE 8 of the DAS-40 driver) from a MS FORTRAN program:

```
mode=8
i(1)=4095                    ! + Full Scale
i(2)=0                       ! DAC # 0
call msf_das40(mode, i(1), Flag)
if (flag .NE. 0) then
    print *,'Mode = ',mode,'     Error # ',flag .AND. 255
endif
```

Linking To The Driver Interface Module

The FORTRAN Interface consists of LINKing to the supplied DAS40.LIB. This Library is where the label *msf_das40* resides.

```
FL /FPi /c <your-program>.for
LINK <your-program>,,,DAS40.LIB;
```

If no errors are reported, you have the executable file <your-program>.EXE ready to test. In the case that errors such as Unresolved External(s) are reported by the LINKer, you must check on whether you LINKed to DAS40.LIB correctly.

DMA Buffer Memory Allocation

MODE 1 requires the specification of one or two far pointers to DMA buffers each in the form of two 16-bit values: Segment and Offset. These pointers should be previously obtained via calls to MODE 18; refer to the DAS-40 User Guide for complete description of this mode.

The following FORTRAN program fragment illustrates the uses of Modes 18 and 1:

```
C ALLOCATE 20K WORDS OF MEMORY
mode=18
i(1)=20000               ! NUMBER OF SAMPLES
call msf_das40(mode, i(1), Flag)

C SAVE RETURNED PARAMETERS
SAMPLES = i(1)           ! ACTUAL NUMBER OF SAMPLES
BUFOFF = i(2)            ! DMA BUFFER OFFSET
BUFSEG = i(3)            ! DMA BUFFER SEGMENT
ACTBUFOFF = i(4)         ! MEMORY BLOCK OFFSET
ACTBUFSEG = i(5)         ! MEMORY BLOCK SEGMENT

C SETUP DATA BUFFER FOR DMA OPERATION
mode=1
i(1)=BUFOFF              ! MEMORY OFFSET
i(2)=BUFSEG              ! MEMORY SEGMENT
i(3)=SAMPLES             ! NUMBER OF SAMPLES
```

Note that the contents of i(2) and i(3) as returned by Mode 18 are passed to Mode 1; while the contents of i(4) and i(5) should be later used by Mode 19 to free the allocated memory block. Refer to your DAS-40 User Guide for a complete description of the Modes mentioned here.

FAR Pointer Manipulation

To obtain the Segment and Offset values for a given FAR variable address, use the functions *KM_segadr()* and *KM_offadr()*. These functions are designed for use from Large Model programs only because FORTRAN does not provide any straight-forward way to perform them.

KM_segadr(far pointer)	Returns a 16-bit Segment value of the FAR pointer specified.
KM_offaddr(far pointer)	Returns a 16-bit Offset value of the FAR pointer specified.

The following code fragment illustrates how to pass the address of user array 'buffer' to Mode 17:

```
mode=17
i(1)=BUFOFF              !SOURCE ADDRESS OFFSET
i(2)=BUFSEG              !SOURCE ADDRESS SEGMENT
i(3)=KM_offadr(buffer)   !DESTINATION ADDRESS OFFSET
i(4)=KM_segadr(buffer)   !DESTINATION ADDRESS SEGMENT
i(5)=2000                !SAMPLES TO TRANSFER
i(6)=0                   !INDEX OF 1ST SAMPLE
```

3.6 MICROSOFT QUICKBASIC

The Software Driver Call Label

You must declare the CALL label to make it known to your application; you make this declaration by inserting the following at the beginning of your program:

```
DECLARE SUB BASDAS40 (MODE AS INTEGER, BYVAL PARAMS AS INTERGER,
FLAG AS INTERGER)
```

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%(10) as follows:

```
DIM D%(10)
COMMON SHARED D%()
```

By making it COMMON SHARED, other modules and subroutines can use this array.

An Example

To initialize your DAS-40 board use MODE 0 as follows:

```
    :
180 MD% = 0                     'initialize mode
190 FLAG% = 0                   'declare error variable
200 D%(0) = 0                   'Card #0
210 CALL BASDAS40(MD%, VARPTR(D%(0)), FLAG%)
220 IF FLAG% <> 0 THEN PRINT "MODE 0 Error # "; FLAG% AND 255: STOP
    :
```

Linking To The Driver Interface Module

The QuickBASIC Interface consists three separate Modules:

DAS40.QLB — Use when you load the QuickBASIC Environment Version 4.5 and you plan to run your program from within the Environment (no EXE involved here). Use the /L switch to load this Quick Library into QuickBASIC:

```
QB /L DAS40 <your-program>
```

DAS40X.QLB — This is identical to DAS40.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 DAS40X <your-program>
```

DAS40.LIB — Link to this library when you want to make a stand-alone EXE program from your QuickBASIC 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>,,,DAS40.LIB;
```

DMA Buffer Memory Allocation

MODE 1 requires the specification of one or two far pointers to DMA buffers each in the form of two 16-bit values: Segment and Offset. The DMA buffers are obtained differently depending on whether you are running your application program from the QuickBASIC Integrated Environment or as a Stand-alone EXE from the DOS command line.

When running as stand-alone EXE programs, use MODE 18 to allocate DMA buffers; refer to the DAS-40 User Guide for complete detail.

When running your programs from within the Integrated Environment, your program can not include calls to MODE 18. This mode call will always fail because the QuickBASIC Environment controls all free memory. In the QB Environment, you can declare large $DYNAMIC arrays as your DMA buffers. $DYNAMIC arrays are placed by QuickBASIC in the far heap (controlled by QB only!) and are generally suitable for DMA.

```
Example:
REM $DYNAMIC
DIM DMABUF(20000) AS INTEGER
REM $STATIC
:
MODE = 1
PARAMS(0) = VARPTR(DMABUF(0))
PARAMS(1) = VARSEG(DMABUF(0))
PARAMS(2) = 20000
CALL BASDAS40(MODE, VARPTR(PARAMS(0)), FLAG)
```

Note that all $DYNAMIC data declarations must occur after all COMMON and DECLARE statements in your program. If you get the QB error: "COMMON and DECLARE must proceed all executable statements", then double check the order of all your DECLARE, COMMON and $DYNAMIC declarations.

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. Refer to the previous section for an example.

■ ■ ■

❑

❑

❑