PCF-HRES

User Guide

for the

Keithley MetraByte

PCF-HRES

PASCAL, C, & FORTRAN

Callable Drivers

For The

DAS-HRES

Revision B - November 1991

Part Number: 24817

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-HRES is a software package for programmers using Pascal, C, FORTRAN, and QuickBASIC to write data acquisition and control routines for the DAS-HRES 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-HRES Interface Library
- Source code for Interface Library
- Miscellaneous documentation (.DOC) files
- Example program files in all supported languages

1.2 SUPPORTED LANGUAGES

The PCF-HRES 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

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 *PCFHRES*.

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

   ```
   LINK example,,,HRES.LIB;
   ```

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

CHAPTER 2

THE DAS-HRES INTERFACE LIBRARY

2.1 OVERVIEW

The Interface Library was programmed with Microsoft Assembler (MASM), Version 5.1, and consists of the following modules:

- HRES.ASM
- HRESPCF.ASM

These files are included in your PCF-HRES Distribution Software diskette(s).

Interface Modules for various languages are also included in this package. You must LINK your program to the appropriate module. These Modules are as follows:

- HRES.LIB: Use for mode calls from Pascal, C, FORTRAN, and stand-alone QuickBASIC programs.
- HRES.BIN: Use for mode calls from BASIC(A).
- HRES.QLB: Use for mode calls from the QuickBASIC Integrated Development Environment (Ver. 4.0 - 4.5).
- HRESX.QLB: Use for mode calls from the QuickBASIC Extended Environment (Ver. 7.0).
- TPHRES.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	FUNCTION
0	Initialize card,input HRES base address,interrupt level, DMA level,and calibration level.
1	Set multiplexor low and high scan limits.
2	Reads the current channel setting and scan limit settings.
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.
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.

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).
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.
7	Disable DMA and interrupt operations started in MODEs 5,6,18, or 20.
8	Reports status of DMA and interrupt operations started by MODEs 5,6,18 or 20.
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.
10	Set Counter 0 configuration. This is the counter whose output is available at the output connector.
11	Load Counter 0 data.
12	Read counter 0.
13	Output to digital outputs OP0-OP7.
14	Read digital inputs IP0-IP7.
15	Output data to a single DAC channel.
16	Outputs data to both DAC channels.
17	Set programmable timer rate.
18	DAC waveform output & ADC input. (Note that no data correction is performed by MODE 18.
19	Analog trigger function.
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.
21	Set channel gains. Use this MODE to set up gain table prior to calling MODEs 3,4,5 and 20.
22	Output a square wave frequency from counter 0 out. Frequency from 1.5Hz to 25KHz.
23	Allocate memory Segment.
24	Deallocate Memory Segment.

2.2 CALLING THE DAS-HRES DRIVER

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

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

HRESPCF.LIB :

mscs_hres	For Calls from Microsoft C, Small Model
mscm_hres	For Calls from Microsoft C, Medium Model
mscl_hres	For Calls from Microsoft C, Large Model
tcs_hres	For Calls from TURBO C, Small Model
tcm_hres	For Calls from TURBO C, Medium Model
tcl_hres	For Calls from TURBO C, Large Model
msp_hres	For Calls from Microsoft Pascal
msf_hres	For Calls from Microsoft FORTRAN
qb_hres	For Calls from Microsoft QuickBASIC

TPHRES.OBJ :

tp_hres	For Calls from TURBO Pascal

HRES.BIN :

hres	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-HRES driver.

PARAM An array of nine 16-bit integers containing 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-HRES driver.

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

The following is code fragment 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 */
Params[3] = o;         /* Auto-Calibration = OFF */
mscl_hres(&Mode, Params, &Flag);
if (Flag != 0)
  printf ("**** Error %d detected in mode 0", Flag);
    :
```

2.3 MODIFYING THE DAS-HRES LANGUAGE INTERFACE MODULES

The Interface Modules support all MODEs of operation described in the DAS-HRES User Guide. 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 HRES.ASM and HRESPCF.ASM, which are in the Distibution Software.

Warning !!

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

HRES.BIN - BASIC(A) Interface

A BASIC(A) interface is made using the BLOADable HRES.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-HRES Distribution Software).

To create HRES.BIN, use the following commands:

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

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

LINK : Warning L4021 : no stack segment

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

TPHRES.OBJ - TURBO PASCAL Interface

An interface for TURBO PASCAL programs is made using the file TPHRES.OBJ. This file is *included* directly into the TURBO PASCAL module, where it is used in conjunction with the $L compiler directive. To create this file you must have access to the following utilities:

TASM.EXE - TURBO Assembler

To create TPHRES.OBJ, use the following command:

```
TASM /DBIN=0 HRES.ASM TPHRES.OBJ;
```

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

The interface for the QuickBASIC Integrated Environment (up to Version 4.5) requires use of the Quick Library file HRES.QLB. This file is specified on the command line with the load */L* switch. For example, **QB / HRES.QLB** .

To create the HRES.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 HRES.ASM;
MASM /DBIN=0 HRESPCF.ASM;
LINK /q HRES+HRESPCF,HRES,,BQLB45;
```

HRES.LIB - Library Used To Make A Stand-alone QuickBASIC (V4.5) Program

To create the HRES.LIB file, you must have access to MASM (the Microsoft Assembler) and LIB.EXE (the Microsft Library Manager). Use the Following Entries:

```
MASM /DBIN=0 HRES.ASM;
MASM /DBIN=0 HRESPCF.ASM;
LIB HRES-+HRES;
LIB HRES-+HRESPCF;
```

HRESX.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 HRES+HRESPCF,HRESx,,QBXQLB;
```

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

HRESX.LIB - Library Used To Make A Stand-alone QuickBASIC (V7.0) Program

To create the HRESX.LIB file, you must have access to MASM (the Microsoft Assembler) and LIB.EXE (the Microsft Library Manager). Use the Following Entries:

```
MASM /DBIN=0 HRES.ASM;
MASM /DBIN=0 HRESPCF.ASM;
LIB HRESX-+HRES;
LIB HRESX-+HRESPCF;
```

HRES.LIB - PASCAL, C, And FORTRAN Interface

The interface for Pascal, C, or FORTRAN uses the library file *HRES.LIB* . This file is linked to the user's program with the following commands:

```
MASM /DBIN=0 HRES;
MASM /DBIN=0 HRESPCF;
LIB HRES-+HRES;
LIB HRES-+HRESPCF;
```

■ ■ ■

CHAPTER 3

THE LANGUAGE INTERFACE

3.1 OVERVIEW

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-HRES product.

The Software Driver Mode Call Labels

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

```
mscs_hres(int *, int *, int *);      /* MS C Small Model */
mscm_hres(int *, int *, int *);      /* MS C Medium Model */
mscl_hres(int *, int *, int *);      /* MS C Large Model */
tcs_hres(int *, int *, int *);       /* Turbo C Small Model */
tcm_hres(int *, int *, int *);       /* Turbo C Medium Model*/
tcl_hres(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[16];
int Flag;
```

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

An Example

To call MODE 0 of the HRES driver from an MS C Medium Model program,

```
    :
Mode=0;
Flag=0;
Params[0]=0X300;
Params[1]=7;
Params[2]=3;
Params[3]=0;
mscm_hres (&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 DAS-HRES Library

Once you have your C program written, you must compile and LINK it to the Library Module, HRES.LIB. HRES.LIB is where the label *mscm_hres* resides.

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

```
CL /AM /c <your-program>.C
LINK <your-program>.OBJ,,,HRES.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, than you must check whether you LINKed to HRES.LIB correctly.

NOTE: Be sure to use the correct Call Label for the Memory Model you are using. For example, use *mscm_hres* for the Medium Memory Model /AM.

3.2 MICROSOFT PASCAL

The Software Driver Mode Call Labels

In your program, declare the following function prototype:

```
FUNCTION MSP_HRES(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 value starts at 0.

An Example

To call MODE 0 of the HRES driver from an MS Pascal program,

```
    :
Mode := 0;
Params[1] := 768;                    (* BOARD ADDRESS *)
Params[2] := 7;                      (* INTERRUPT LEVEL *)
Params[3] := 3;                      (* DMA LEVEL *)
Params[4] := 0;                      (* AUTO-CALIBRATION = OFF *)

Result := MSP_HRES (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 written your MS Pascal program, you must compile and LINK it to the Interface Module, HRES.LIB. HRES.LIB is where the label *MSP_HRES* resides.

For example,

```
PL <your-program>.pas;
LINK <your-program>,,,HRES.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, you must determine whether you LINKed to HRES.LIB correctly.

3.3 BORLAND TURBO PASCAL

The Software Driver Call Label

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

```
FUNCTION TP_HRES(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 declare as [0..15], the index values start at 0.

An Example:

To call MODE 0 of the HRES driver from Turbo Pascal program:

```
    :
Mode := 0;
Params[1] := 768;                          (* BOARD ADDRESS *)
Params[2] := 7;                            (* INTERRUPT LEVEL *)
Params[3] := 3;                            (* DMA LEVEL *)
Params[4] := 0;                            (* AUTO-CALIBRATION = OFF *)
Result := TP_HRES (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 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 *TPHRES.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 TPHRES}
```

Once included, you are ready to compile your program:

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

3.4 MICROSOFT FORTRAN

The Software Driver Call Label

The call label *msf_hres* 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(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 DAS-HRES driver from a Microsoft FORTRAN program,

```
mode=0
i(1)=768                          ! Board Address
i(2)=7                            ! Interrupt Level
i(3)=3                            ! DMA Level
i(4)=0                            ! Auto-Calibration = Off
call msf_hres(mode, i(1), Flag)
if (flag .NE. 0) then
    print *,'Mode = ',mode,' Error # ',flag
endif
```

Linking To The Driver Interface Module

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

```
FL /FPi /c <your-program>.for
LINK <your-program>,,,HRES.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 HRES.LIB correctly.

3.5 MICROSOFT QUICKBASIC

The Software Driver Call Label

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

```
DECLARE SUB QBHRES (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%()
```

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

An Example

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

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

Linking To The Driver Interface Module

The QuickBASIC Interface consists three separate Modules:

HRES.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 HRES <your-program>
```

HRESX.QLB — This is identical to HRES.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 HRESX <your-program>
```

HRES.LIB — Link to this library when you want to make a stand-alone EXE program from your QuickBASIC (4.5) 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>,,,HRES.LIB;
```

HRESX.LIB — Link to this library when you want to make a stand-alone EXE program from your QuickBASIC (7.0) 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>,,,HRESX.LIB;
```

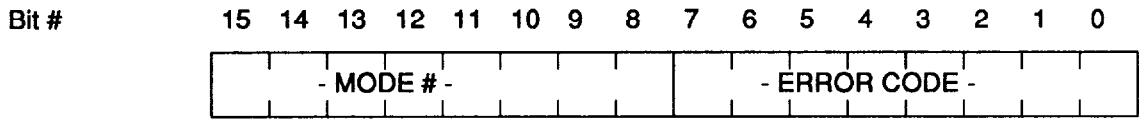
NOTE: All $DYNAMIC data declaration must occur after all COMMON and DECLARE statements in your program. If you get the QB error, COMMON and DECLARE must precede all executable statements; double check the order of all DECLARE, COMMON, and $DYNAMIC declarations.

■ ■ ■

CHAPTER 4

SUMMARY OF ERROR CODES

In general, you receive reports of error conditions via the Error Flag. This flag is of an unsigned integer type (16 bits) and contains the MODE number and Error Code number in the following format:



The Error Code is obtained by ANDing the error flag with 255 (0FFH). An Error Code of zero implies that the MODE Call was successful.

The following list contains Error Code definitions and suggested actions.

Error 1: Driver not initialized.

Meaning:	This error may be issued from any MODE. It indicates that the driver must be re-initialized prior to this step.
Action:	Call MODE 0 before the MODE in which this error occurred.

Error 2: Mode number out of range.

Meaning:	This error may be issued from any MODE. It indicates that the mode number specified is not allowed.
Action:	Check the mode number to be within the range 0 thru 24 inclusive

Error 3: Board's base address out of range.

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

Error 4: Interrupt Level used is out of range.

Meaning:	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.

Action: Supply MODE 0 with an Interrupt Level in the range 2-7, inclusive.

Error 5: DMA Channel not 1 or 3.

Meaning: This error is generated by MODE 0 whenever the specified DMA Channel is not allowed. Note that, unlike the Interrupt Level selection, the selected DMA Channel must match the DMA Channel Switch position on the board!

Action: Supply MODE 0 with DMA Channel 1 or 3.

Error 6: Mux scan limits out of range.

Meaning: This error is generated by MODE 1 whenever the Differential Input Channel scan limits are out of range.

Action: Check the channel numbers passed to MODE 1. They must be within 0 thru 15, inclusive.

Error 7: NOT USED.

Error 8: NOT USED.

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

Meaning: This error is generated by MODE 3 whenever an EOC signal expected from the DAS-HRES is not recieved. This indicates a hardware failure.

Action: Check the board's Base Address.

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

Meaning: This error is generated by MODE 17 whenever one or both Counter divisors are specified as 0 or 1.

Action: Check both Counter divisors for values within the range 2-65535, inclusive.

Error 11: Number of conversions out of range.

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

Action: Check the number of conversions.

Error 12: Counter 0 Configuration number out of range.

Meaning:	This error is generated by MODE 10 whenever Counter 0 configuration number is not within the allowed range of 0-5, inclusive. Refer the manual for detail on the different configuration modes.
Action:	Check the configuration number.

Error 13: Digital Output data out of range.

Meaning:	This error is generated by MODE 13 whenever the output data value is not within 0-255, inclusive.
Action:	Check the Digital Output value.

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

Meaning:	This error is generated by MODEs 15 and 16 whenever the specified D/A value is out of the allowed range of -32768 to 32767, inclusive.
Action:	Check the DAC value.

Error 15: DAC Channel number not 0 or 1.

Meaning:	This error is generated by MODEs 15 and 18 whenever the specified DAC channel number is not allowed. the DAS-HRES boards support two DAC channels: 0 and 1.
Action:	Check the DAC channel number.

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

Meaning:	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.
Action:	Check Read Operation number.

Error 17: Starting Sample number out of range.

Meaning:	This error is generated by MODE 9 whenever the Starting sample number is out of range. The Starting sample number must within 0-32767, inclusive. 0 specifies the 32768.
Action:	Check the Stating sample number.

Error 18: Requested sample count out of range.

Meaning:	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 be within 0-32767, inclusive. 0 specifies 32768.
Action:	Check the Sample Count.

Error 19: Trigger mode must be 0 or 1.

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

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

Meaning:	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.
Action:	Use Mode 7 to terminate current acquisition operation.

Error 21: DMA Page Wrap.

Meaning:	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.
Action:	Use MODE 23 to determine a 'good' DMA Segment for MODE 6.

Error 22: Board not present.

Meaning:	The Board presence test perform in MODE 0 has failed. The board's Base Address passed to MODE 0 must match the actual address selected on the board.
Action:	Check the board's base address.

Error 23: Analog Trigger channel out of range.

Meaning:	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).
Action:	Check the channel number.

Error 24: Analog Trigger data out of range.

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

Error 25: Analog Trigger slope not 0 or 1.

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

Error 26: Gain code out of range.

Meaning:	This error is generated by MODEs 6, 19, and 21 whenever the specified Gain Code is out of range. The allowed Gain Codes are 0-3, inclusive.
Action:	Specify a valid Gain Code.

Error 27: Auto-Calibration code out of range.

Meaning:	This error is generated by MODE 0 whenever the Auto-calibration Code for initialization is out of range. The allowed Codes are 0 thru 2, inclusive.
Action:	Specify a valid Code.

Error 28: Possible memory index overflow.

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

Error 29: Memory allocation/deallocation error.

Meaning:	This error is generated by MODE 23 or MODE 24 whenever memory can not be allocated or deallocated through DOS.
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.

■ ■ ■