DAS-TC User's Guide

Revision C - April 1994
Part Number: 84370

Keithley Instruments, MetraByte Division

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

Contents

Contents

Preface

The *DAS-TC User's Guide* is an operator reference for technicians, engineers, scientists, or other persons using a DAS-TC board to monitor or record thermocouple measurements. To follow the information and instructions contained herein, readers must be familiar with thermocouple principles and measurement techniques, with PC and DOS operation, and with current DOS and/or Windows software-control tools.

The guide focuses primarily on describing the board and its capabilities, setting up the board and its associated software, making typical thermocouple/voltage hookups, and operating the data monitoring and recording software. There are also chapters on calibration and troubleshooting. For programmers in Microsoft® QuickBasic™, Microsoft Professional BASIC, and Microsoft Visual Basic™ for DOS, the guide also includes two chapters on the *Function Call Driver*, a software tool to simplify the writing of applications programs for the DAS-TC.

The *DAS-TC User's Guide* is organized as follows:

- Chapter 1 describes the DAS-TC board's purpose, its analog input features, its software options, its accessories, and its specifications. The chapter also contains a DAS-TC block diagram.
- Chapter 2 discusses unpacking and inspection, hardware configuration, software installation, use of the Configuration File Generator, and hardware installation.
- Chapter 3 shows the preferred methods of making I/O connections, using the STC-TC and STA-TC accessories.
- Chapter 4 describes the function and operation of the DATALOGGER program.
- Chapter 5 describes the single DAS-TC calibration requirement.
- Chapter 6 presents an approach to problem isolation and a troubleshooting table. This chapter also contains instructions for obtaining technical support.
- Chapter 7 is an introduction to the purpose and implementation of the DAS-TC Function Calls, which are programming tools for QuickBASIC, Professional BASIC, and Visual Basic. This chapter also identifies and briefly describes the Function Calls available for the DAS-TC.
- Chapter 8 describes the purpose and use of each Function Call on an individual basis.
- Appendix A is a graphical pinout of the DAS-TC I/O connector.
- Appendix B is a list, with descriptions, of each of the error codes. This appendix also gives the DAS-TC response to over-/under-voltage and over-/under-temperature inputs from the thermocouples.
- Appendix C describes the CJC sensor circuit used on the STA-TC and STC-TC accessory panels.
- Appendix D describes the DAS-TC External Driver and its setup for use with the VIEWDAC, EASYEST LX, and ASYST data acquisition and analysis programs.

■ ■ ■

Chapter 1

DESCRIPTION

The DAS-TC is a thermocouple/voltage measuring board that accepts up to 16 inputs. Inputs may be any mix of thermocouple and other voltage sources. Readings from the DAS-TC are in degrees (Celsius or Fahrenheit) or volts, according to the input type.

This chapter describes the DAS-TC circuit functions, features, software options, accessories, and specifications.

1.1 FEATURES

DAS-TC features are as follows:

- The board fits an accessory slot in an IBM™ PC/XT®, PC-AT®, or equivalent computer.
- All 16 channels are differential input.
- Thermocouple inputs may be any mix of up to seven standard types, as follows: J, K, E, T, R, S, and B.
- Voltage input ranges and corresponding gain levels are as follows:

Gain	Range
1	-2.5 to +10 V
125	-20 to 80 mV
166.7	-15 to 60 mV
400	-6.25 to 25 mV

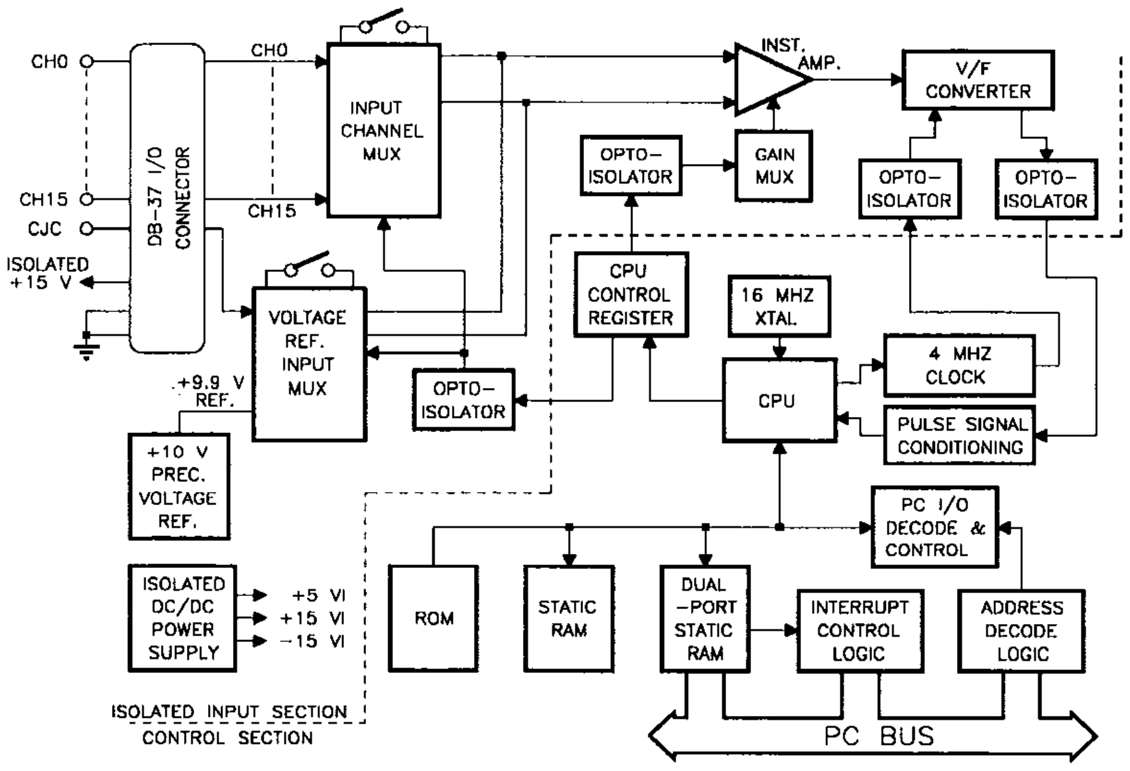
- Thermocouple measurements are linearized by an onboard microprocessor for readings in degrees (Celsius or Fahrenheit).
- Open thermocouple detection is optional.
- Automatic CJC (Cold Junction Compensation) on a per-channel-basis is continuous.
- Calibration of the CJC and board logging parameters is automatic and continuous.
- Inputs connect to the DAS-TC through either a plug-in screw-terminal block (the STC-TC) or an external screw-terminal box (the STA-TC).
- CMRR (Common Mode Rejection Ratio) is greater than 100 dB for gains greater than one.
- Sampling Rate is configurable for 50, 60, or 400 Hz.

- Analog inputs (measurement section) are electrically isolated from the PC.
- The board can perform onboard averaging.

1.2 FUNCTIONAL DESCRIPTION

Principal Sections

DAS-TC circuits are in two principal sections: the isolated input section and the control section. The isolated input section handles the measurement functions, while the control section handles data processing. The following block diagram represents the two sections and their essential stages.



DAS-TC Block Diagram

The isolated input section consists of a CJC sensor input, calibration inputs, a 16-channel input multiplexer, a programmable gain amplifier, and a V/F (voltage-to-frequency) converter. During operation, the DAS-TC continually monitors the CJC input to maintain the accuracy of the board over time and temperature. At the same time, the board switches in a precision 9.9 V input to measure gain error at a gain of 1 and to measure offset errors for all four gain ranges. The board then stores these measurements in onboard memory.

The four ranges of the programmable gain amplifier are selectable to match the thermocouple input ranges and to accommodate the voltage input range of 0-10 V (using a gain of 1).

The V/F converter provides excellent noise rejection and high resolution while generating a square-wave output whose frequency is proportional to voltage input. The square-wave output passes first through a stage of optical isolation and then into a stage of counting, where a count of the square waves over a specified period determines a value for the corresponding voltage input. The period for each count of square waves is set by the value entered in the configuration file for Normal Mode Rejection Frequency (50, 60, or 400 samples/s); the actual rate is one-half this selected value. The longer the count time, the higher the resolution and better the noise rejection.

The control section consists of the microprocessor and its memory. The microprocessor performs all control functions and mathematical calculations, precluding the need for any PC processing. Working from the configuration file, the microprocessor sets up the board for the desired configuration. Board parameters include Interrupt Level and Normal Mode Rejection Frequency. Channel parameters include thermocouple type, number of samples to average, and type of Engineering Units. During the acquisition process, the microprocessor also handles scan order, thermocouple linearization, and calculations for CJC (cold junction compensation).

Onboard memory includes ROM, Static RAM, and Dual-Port Static RAM. ROM contains the DAS-TC control program, which directs the activities of the microprocessor and the thermocouple look-up tables. Static RAM serves as the CPU "scratch pad," providing temporary storage for measurement results and calibration coefficients. The Dual-Port RAM is the buffer for communications and data flow to and from the PC.

Operational Flow

Configuration data for the DAS-TC board and channels is contained in the configuration file. You may use the default configuration file furnished with the DAS-TC software package, or you may generate your own using the Configuration File Generator program (described in Chapter 2). Whichever you use becomes the configuration reference for the onboard CPU.

During DAS-TC initialization, the configuration information is downloaded from the DAS-TC software directory, on the PC hard drive, to the DAS-TC board memory. The CPU then sets up the DAS-TC board and channel parameters to the specified values. Following parameter setup, the CPU performs a calibration and stores the gain and offset coefficients in board memory. Next, the board reads and stores the CJC sensor value.

When the PC initiates a channel scan, the DAS-TC scans the channels in the order specified by the PC. Channels designated for thermocouple input use parameter values from the configuration file. Channels designated for voltage input also use parameter values from the configuration file, unless you overwrite these values.

Channel readings proceed at the rate specified by the value for Normal Mode Rejection Frequency, in the configuration file. The reading for each channel requires a count of the frequency output of the V/F Converter. The CPU compensates for calibration errors in these readings. For channels configured as thermocouple inputs, the CPU also adjusts for CJC and converts the readings to temperature measurements. For channels configured as voltage inputs, the CPU converts the readings to volts.

To convert thermocouple readings to temperature measurements, the CPU refers to *look-up* tables, stored in ROM. There is a separate look-up table for each of the seven thermocouple types accommodated by the DAS-TC. The look-up tables optimize accuracy by using more reference points along ranges of greatest temperature-versus-voltage change than along ranges of minimal change (using the same number of points at fixed intervals would lead to error along ranges of greatest temperature-versus-voltage change).

As soon as all readings and conversions are complete, the DAS-TC issues an Interrupt and transfers all measurements to the PC.

The DAS-TC makes periodic measurements of the CJC and performs self-calibration as a background task.

1.3 OPERATING OPTIONS

To operate the DAS-TC, use one of the following software options:

- The *DATALOGGER* program is a graphical control interface that allows you to monitor and/or record the thermocouple/voltage inputs for one DAS-TC board in a system of up to two boards.
- The *Function Call Driver* supports a comprehensive set of callable functions you may use as programming tools when writing application programs in QuickBASIC, Professional BASIC, and Visual Basic for DOS.
- The ASO-TC Advanced Software Option provides Function Call Drivers for C and Pascal and a Windows Dynamic Link Library.
- The DAS-TC External Driver supports the use of a DAS-TC board with the VIEWDAC, EASYEST LX, and ASYST data acquisition and analysis software.

1.4 ACCESSORIES

The following accessories are available for the DAS-TC:

ASO-TC	Advanced Software Option for the DAS-TC. The ASO-TC includes Function Call Drivers for C and Pascal and a Windows 3.X Dynamic Link Library. ASO software is supplied on both 3.5" and 5.25" diskettes.
STC-TC	The STC-TC is a panel containing a CJC sensor and screw terminals for all I/O connections of the DAS-TC. The STC-TC plugs directly into the DAS-TC I/O Connector and extends from the rear of the computer.
STA-TC	The STA-TC is housed in a plastic enclosure and contains a CJC sensor and screw terminals for all I/O connections of the DAS-TC. You use a C-1800 cable to connect the STA-TC to the DAS-TC.
C-1800	The C-1800 is an 18" ribbon cable for connecting the STA-TC Expansion Box to the DAS-TC. This cable is available in longer lengths but must be specified accordingly (for example, to specify three extra feet, use P/N C-1803).
DAS-TC External Driver	The external driver is a program that supports the use of a DAS-TC board with the following Keithley MetraByte data acquisition and analysis software: VIEWDAC, EASYEST LX, and ASYST.

1.5 SPECIFICATIONS

DAS-TC specifications are as follows:

Analog Specifications

Inputs:	16 differential channels plus a CJC
Thermocouple Types:	J, K, E, T, R, S, and B
Voltage Gains:	1, 125, 166.7, and 400
Overvoltage Protection:	±30 V max. for powered ±20 V max. for unpowered
Isolation to PC:	500 VDC min.
Input Impedance:	100 MΩ min.
CJC Error:	±0.8 °C max. (at 25 °C) ±1.2 °C max. (at 0 to 70 °C)

COMMON MODE REJECTION RATIO

Voltage Range of Gain = 1:	72 dB min. (DC to 60 Hz)
All Other Ranges:	100 dB min. (DC to 60 Hz)

NORMAL MODE REJECTION RATIO

55 dB typ. (at 50/60 Hz)

Open Thermocouple Reading	
Integer Data Format:	+1,176,256,512 (°C or °F)
Real (Floating Point) Data Format:	+10,000.00 (°C or °F)

Conversion Rates

	Conversion Speed (ms/chan) * for	
Rejection Rate	**Temperature**	**Volts**
50	47	43
60	40	36
400	10	6

* The speed values are for integer-output format. For floating-point format, add 1 ms to each value.

Temperature Units (User Configurable)

Degrees:	Celsius or Fahrenheit

Temperature/Voltage Data Format

Integer:	32-bit signed
Real:	32-bit IEEE-754 Standard (floating point)

Accuracy and Resolution (at 25 °C)

All ranges guaranteed monotonic.

VOLTAGE INPUTS:

Gain	Range	Accuracy (Worst case)	Voltage Resolution at 50 Hz	60 Hz	400 Hz
1	-2.5 to 10 V	±0.01% reading ±2.5 mV	312.5 μV	375 μV	2.5 mV
125	-20 mV to 80 mV	±0.02% reading ±26 μV	2.5 μV	3.0 μV	20 μV
166.7	-15 mV to 60 mV	±0.02% reading ±22 μV	1.88 μV	2.25 μV	15.0 μV
400	-6.25 mV to 25 mV	±0.03% reading ±12.5 μV	0.781 μV	0.938 μV	6.25 μV

THERMOCOUPLE INPUTS:

Type	Range	Accuracy (Worst case)	Temperature Resolution at 50 Hz	60 Hz	400 Hz
J	-200 to -1 °C	±1.0 °C	0.1 °C	0.1 °C	0.7 °C
	0 to 750 °C	±0.5 °C	0.04 °C	0.05 °C	0.3 °C
K	-200 to -1 °C	±1.4 °C	0.1 °C	0.2 °C	1.0 °C
	0 to 900 °C	±0.7 °C	0.05 °C	0.06 °C	0.4 °C
	901 to 1250 °C	±0.9 °C	0.06 °C	0.07 °C	0.5 °C
E	-200 to -50 °C	±1.1 °C	0.1 °C	0.1 °C	0.8 °C
	-49 to 1000 °C	±0.6 °C	0.05 °C	0.06 °C	0.4 °C
T	-200 to -120 °C	±0.9 °C	0.05 °C	0.06 °C	0.4 °C
	-119 to 400 °C	±0.5 °C	0.03 °C	0.04 °C	0.2 °C
R	0 to 299 °C	±2.3 °C	0.1 °C	0.2 °C	1.0 °C
	300 to 1768 °C	±1.5 °C	0.08 °C	0.1 °C	0.6 °C
S	0 to 299 °C	±2.3 °C	0.1 °C	0.2 °C	1.0 °C
	300 to 1450 °C	±1.7 °C	0.09 °C	0.1 °C	0.7 °C
B	400 to 799 °C	±3.0 °C	0.2 °C	0.2 °C	1.5 °C
	800 to 1700 °C	±1.7 °C	0.1 °C	0.1 °C	0.8 °C

* Accuracy values do not include CJC error.

Noise

VOLTAGE INPUTS:

Gain	Factor
1	0.75 x resolution rms
125	1.0 x resolution rms
166.7	1.5 x resolution rms
400	4.0 x resolution rms

THERMOCOUPLE INPUTS:

Type	Factor
E	1.0 x resolution rms
J or K	1.5 x resolution rms
B, R, S, or T	4.0 x resolution rms

Maximum Gain Error Temperature Coefficients

VOLTAGE INPUTS:

Gain	Temp Coefficient
1	±7 ppm/°C
125	±10 ppm/°C
166.7	±12 ppm/°C
400	±17 ppm/°C

THERMOCOUPLE INPUTS:

Type	Temp Coefficient
J	±12 ppm/°C
K	±12 ppm/°C
E	±10 ppm/°C
T	±17 ppm/°C
R	±17 ppm/°C
S	±17 ppm/°C
B	±17 ppm/°C

NOTE: The Offset error at all gains is canceled via periodic self-calibration that the microprocessor of the DAS-TC performs as a background task.

Environment

Operating Temperature:	0 to 50° C
Storage Temperature:	-20 to 70° C
Humidity:	0-95% noncondensing
Dimensions:	13.3 L x 4.25 H x .75 D in. (33.8 x 10.8 x 1.9 cm)

Power

+5 V Supply:	950 mA typical; 1500 mA max
+12 V Supply:	35 mA typical; 50 mA max
-12 V Supply:	Not used

Miscellaneous

PC I/O Base Address Interface:	100h to 3FFh DIP switch selectable
Interrupt Levels:	2 - 7

■ ■ ■

Chapter 2

SETUP

This chapter describes how to inspect the DAS-TC, set up the hardware, install the Distribution Software, run the configuration utility and install the board.

Read this chapter before attempting to install and use your DAS-TC board.

2.1 BOARD-HANDLING PRECAUTIONS

When you handle a computer board, the discharge of static electricity from your hands can seriously damage certain electrical components. Therefore, you should discharge static from yourself before handling a board by touching a grounded conductor such as your computer chassis (your computer must be turned off but grounded). When you handle a board, hold it by its edges and try to avoid touching any board components.

2.2 INSPECTION

Factory packaging of the DAS-TC includes a final wrap of protective, anti-static material. Use the following procedure to remove this final wrap and inspect the board.

1. After taking the board-handling precautions of Section 2.1, carefully remove the board from its anti-static wrapping material. You may wish to store the wrapping material for possible future use.
3. Inspect the board for signs of damage. If damage is apparent, arrange to return the board to the factory (see *Technical Support* in Chapter 6).
4. Check the remaining contents of your package against the packing list to be sure your order is complete. Report any missing items, immediately.
5. When you are satisfied with the inspection, proceed with the hardware setup.

   Note: The DAS-TC is factory calibrated; it requires no further adjustment prior to installation. If at a later time you decide to calibrate the DAS-TC, refer to Chapter 5.

2.3 HARDWARE SETUP

The only configurable component on the DAS-TC is the DIP switch labelled *BASE ADDRESS*. Figure 2-1 shows the 7-position DIP switch with factory settings.

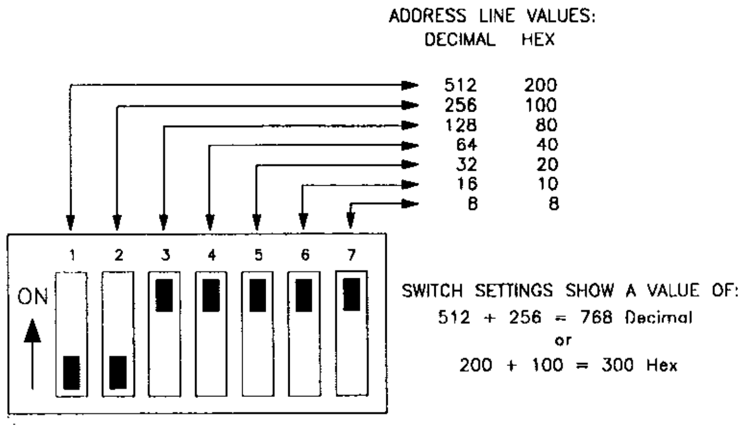


Figure 2-1. DAS-TC Base Address Switch

The DAS-TC uses a block of four non-overlapping addresses in the computer I/O address space between 100h and 3FFh. If the factory-set Base Address conflicts with the address of another board or device in your computer, you need to reset the DIP switch for an unused address.

To learn what addresses are available, consult the manual for your computer or motherboard. To determine base address switch settings for a particular address, use the configuration utilty (refer to Section 2.5).

2.4 SOFTWARE INSTALLATION

This manual refers to the software portion of the DAS-TC package as the *Distribution Software*. The DAS-TC Distribution Software contains programs and files in compressed form on 5.25" and 3.5" diskettes. Software for the basic DAS-TC package is for DOS only, while software for the ASO-TC package is for DOS and Windows. Since the utilities (the configuration utilty and the DATALOGGER programs) for DOS and Windows are similar in appearance and functionally equivalent, they are both covered by this manual.

Before installing your Distribution Software, make a back-up copies and store the original diskettes in a safe environment.

Use the following procedure(s) to decompress your basic Distribution Software and install it on your computer hard drive.

Installing the Basic DAS-TC Distribution Software

You must install your basic Distribution Software from the DOS command line. This installation first decompresses the software then installs it on about 1.4 MB of your computer hard drive. The procedure for this installation is as follows:

1. Place Diskette #1 in the floppy drive of your computer
2. Change to the floppy drive containing the diskette, and enter

```
INSTALL
```

3. Follow the instructions on the screen.

The installation program creates a directory on the hard drive then decompresses and loads the Distribution Software into that directory. Unless you specify otherwise, the default drive is C and the default directory *DASTC*.

Installing the ASO-TC Distribution Software

The ASO-TC Advanced Software Option furnishes Distribution Software in both DOS and Windows versions. Base your choice of which to use on your own preference, since both versions are functionally equivalent.

Before installing your Distribution Software, make back-up copies and store the original diskettes in a safe environment.

Both the DOS and the Windows versions are in compressed form. Therefore, their installation first decompresses the software then installs it on the hard drive. The procedures for the installation of both versions are as follows:

INSTALLING THE ASO-TC FOR DOS

Installation of the ASO-TC DOS version of the Distribution Software requires about 1.5 MB of hard drive space. The installation procedure is as follows:

1. Place Diskette #1 in a floppy drive of your computer.
2. Change to the floppy drive containing Diskette #1, and enter

   ```
   INSTALL
   ```
3. Follow the directions on the screen.

Unless you specify otherwise during the installation, the default drive is C and the default directory *ASOTC*.

INSTALLING THE ASO-TC FOR WINDOWS

Installation of the ASO-TC Windows version of the Distribution Software requires about 2 MB of hard drive space. The installation procedure is as follows:

1. Place the Windows diskette in a floppy drive of your computer.
2. Run the Windows program.
3. From the Program Manager File menu, choose *Run*.
4. In the Command Line text box, type the letter of the drive containing your Windows diskette and follow with *SETUP*. For example, if your diskette is in the A Drive, enter

   A:\SETUP
5. Click on *OK*.
6. Follow the directions on the screen.

Unless you specify otherwise during the installation, the default drive is C and the default directory *ASOTC\WINDOWS*.

FILES.DOC and README.DOC

Your installed Distribution Software contains the ASCII files *FILES.DOC* and *README.DOC*. FILES.DOC lists and describes all the files in your Distribution Software. README.DOC contains any last-minute changes to the software. Both files are readable with any text editor (word processor) or with the DOS *TYPE* command.

2.5 THE CONFIGURATION UTILITY

The configuration utility is a graphical interface program for creating or changing a configuration file. The DOS version of this program is *DASTCCFG.EXE* and is located in your basic DAS-TC Distribution Software (in the DASTC directory). The Windows version of this program is *WDASTCCF.EXE* and is located in the ASO-TC Distribution Software (in the ASOTC directory). You may operate either program with a mouse and/or the keyboard, using the conventions in the *Preface*.

A configuration file contains parameter settings used as a reference by the DATALOGGER program (see Chapter 4) or by the Function Call Driver (see Chapters 7 and 8 in this manual or your *ASO-TC User's Guide*). The default configuration file (in your Distribution Software) is *DASTC.CFG*. You can use this file as is or copy and rename it for modification with the configuration utilty. You can also create a totally new configuration file with the configuration utility program. While you can give new configuration files any name, you will recognize them faster if you give them *.CFG* extensions (for example, *MYFILE.CFG* or *NEWFILE.CFG*).

Starting the Utility

> *Note:* See Appendix D if you are using VIEWDAC, EASYEST LX, or ASYST.

To start the configuration utilty from DOS or Windows, perform the following steps:

1. Start the configuration utility from DOS or Windows as follows:

   - *If you are in the DOS environment,* change to the directory containing DASTCCFG.EXE, and enter the following at the DOS prompt:

     ```
     DASTCCFG
     ```

   - *If you are in the Windows environment,* double click on the .CFG icon in the DAS-TC Group Window, which is within the Program Manager window.

   After the configuration utilty starts, it displays a help screen.

2. Press **[Enter]**.

   The configuration utility displays the path-specification screen, shown in Figure 2-2.

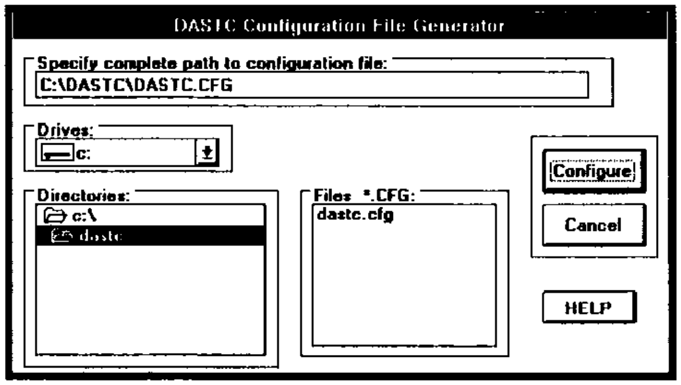


Figure 2-2. Path-specification Screen

Note: The configuration utility screens in this manual are the product of the Windows-based configuration utilty program. While the DOS- and Windows-based screens differ slightly in appearance, they are functionally the same (except that the Windows-based program cannot be used for the DAS-TC External Driver).

Path-specification Screen

In the path-specification box, the default configuration file and its path for the DOS program are *C:\DASTC\DASTC.CFG.* The default configuration file and its path for the Windows program are *C:\ASOTC\DASTC.CFG.*

To accept the path defaults, move the cursor to the *Configure* button and click or press **[Enter]**.

To create a totally new configuration file and path, tab to the path-specification box and use the keyboard to overwrite the contents.

To select a different configuration file and/or path, use the following procedure:

1. Move to the *Drive* selector, use mouse or keyboard to scan the list of available drives, and select a drive. The selected drive appears in the *Directories* box with a list of its directories.
2. Move to the *Directories* box, scan the list of available directories, and select one.
3. Move to the *Files *.CFG* box, scan the list of available configuration files, and select one.
4. Move to the *Configure* button and click or press **[Enter]** to complete the procedure.

   The program then displays the *First Board Main Menu*.

   Note that clicking or pressing **[Enter]** at the *Cancel* button exits the program.

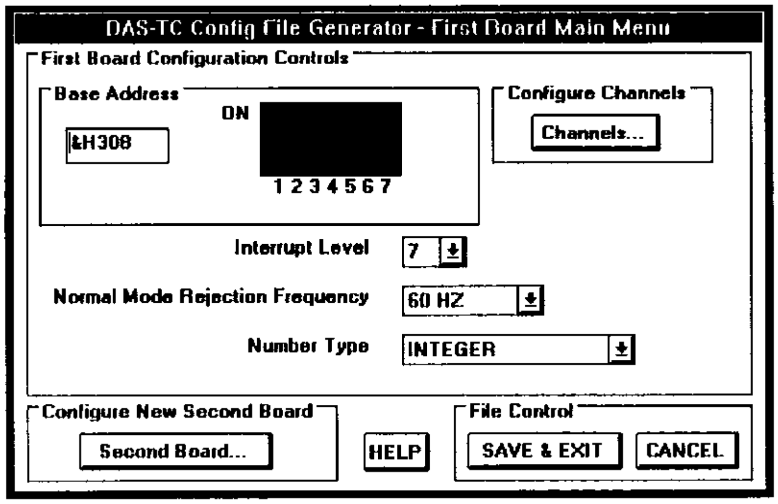
First Board Main Menu

NOTE: The Windows configuration utility cannot be used for the DAS-TC External Driver. In the DOS configuration utility, whether you are configuring for standard operation or for operation with the DAS-TC External Driver, the Main Menu and the Thermocouple Configuration Table are the same except for the following:

• The Main Menu for the DAS-TC External Driver does not offer the FLOATING POINT option under Number Type.

• The Thermocouple Configuration Table for the DAS-TC External Driver does not offer the VOLTS option under Type.

The First Board Main Menu, shown in Figure 2-3, contains configuration options for the First Board and a selector for accessing the Second Board Main Menu.


DAS-TC Config File Generator - First Board Main Menu

First Board Configuration Controls

Base Address: &H308

Interrupt Level: 7

Normal Mode Rejection Frequency: 60 HZ

Number Type: INTEGER

Figure 2-3. Menu of Configuration Options

Configuration options for the First and Second Board Main Menus are as follows:

• ***<u>Base Address</u>***

This option is an aid for determining acceptable settings for the base address DIP Switch. To be acceptable, an address must be between 100h and 3FFh (inclusive) on an 8-byte boundary, and it must differ from the address of a second DAS-TC board. Settings in the DIP Switch diagram change automatically to the address you enter in the address-edit box.

The factory-set base address of 310h works in most computers. However, if you are unable to proceed from the first screen of the DATALOGGER program (described in Chapter 4), you may want to try a new base address. To change a base address, put your keyboard in the overwrite mode (by pressing the [**Insert**] key), place the cursor in the base address box, and overwrite the existing address. If you make an incorrect entry, the program displays an error message and returns to the original address or corrects your entry.

Remember to restore the keyboard to insert mode by pressing the [**Insert**] key again after you complete a Base Address entry. Otherwise, the overwrite mode remains in effect.

- ***Interrupt Level***

  This option selects the interrupt level for the DAS-TC board as 2, 3, 4, 5, 6, or 7. To change the existing value, scan the pull-down list and make a selection. The program does not accept the same value for a second board. Note that the uses of different interrupt levels may vary between computers, making some of the above choices unavailable. For example, Interrupt Level 2 is becoming more commonly used by some computers. If necessary, use a program such as *Norton Utilities* to determine the availability of interrupts.

- ***Normal Mode Rejection Frequency***

  This option selects the Normal Mode Rejection Frequency of the DAS-TC board as 50, 60, or 400 Hz. The selected value must match the frequency of the computer power source to allow DAS-TC operation at the optimum Normal Mode Rejection Ratio. The values of 50, 60, and 400 Hz represent the standard power frequencies of Europe, the United States, and many U.S. military facilities.

- ***Number Type***

  This option selects the format for data returned from the DAS-TC as integer or floating point. To make a change, scan the pull-down list and make your selection.

  When you select *Integer,* the value returned by the DAS-TC for temperature is in centidegrees (degrees/100) and for voltage is in microvolts. To convert an *Integer* temperature reading to degrees, divide it by 100. To convert an *Integer* voltage reading to volts, divide it by 1,000,000.

  When you select *Floating Point,* the value returned by the DAS-TC for temperature is in degrees and for voltage is in volts. No conversion is required. The example program QBEXAMP2.BAS demonstrates the use of floating point numbers; QBEXAMP2.BAS is in your Distribution Software.

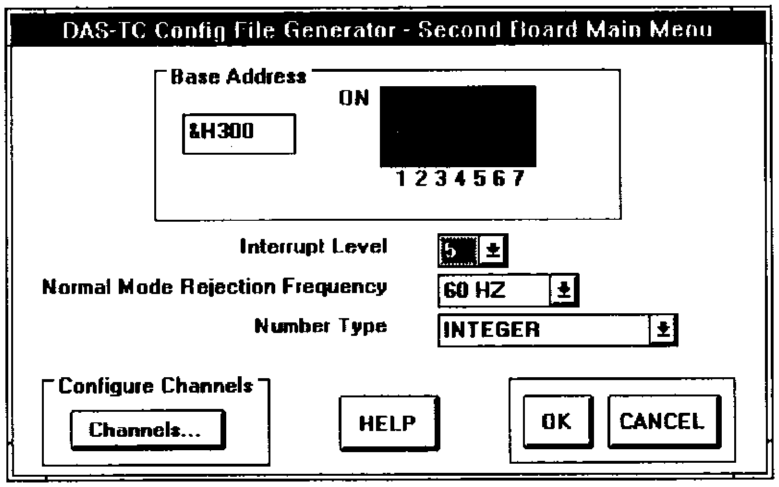
- ***Configure Channels***

  Use this button to transfer to the Thermocouple Configuration Table. Refer to the section entitled *Thermocouple Configuration Table* , on the next page, for descriptions of the options.

- ***Configure/Edit New/Old Second Board***

  If the current configuration file has never been edited for a second board, this box is entitled *New Second Board.* Otherwise, this box is entitled *Edit Old Second Board.* The box converts from *New* to *Old* when you Save & Exit the configuration utyility if you saved the contents of the second-board menus to the configuration file (by pressing *OK,* as described below).

  Use the *Second Board* button to configure a second DAS-TC board. The Second Board Main Menu options are the same as for the First Board, as shown in Figure 2-4.



DAS-TC Config File Generator - Second Board Main Menu

Base Address: &H300

ON 1 2 3 4 5 6 7

Interrupt Level: 5

Normal Mode Rejection Frequency: 60 HZ

Number Type: INTEGER

Figure 2-4. Second Board Main Menu

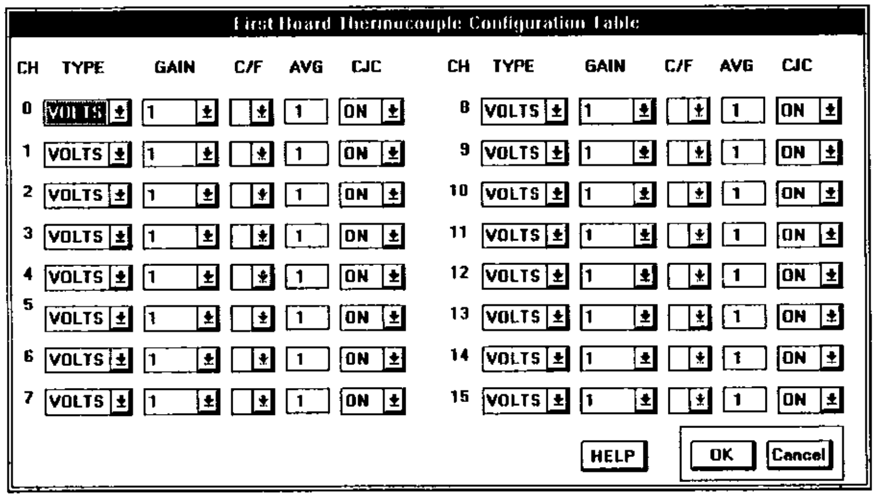
Use the *OK* button of the Second Board Main Menu to return to the First Board Main Menu while retaining any changes; the OK button saves configuration settings--even if they are unchanged. The *Cancel* button exits this menu and loses any changes.

- ***File Control***

  This box contains options for exiting the configuration utilty program. Use *Save & Exit* to save all data to the specified configuration file and then exit to DOS. Use *Cancel* to exit to DOS without saving any data.

Thermocouple Configuration Table

The Thermocouple Configuration Table, shown in Figure 2-5, enables you to configure the 16 channels individually for thermocouple type or voltage, gain (for voltage readings only), Celsius or Fahrenheit, samples to be averaged, and CJC On or Off.



First Board Thermocouple Configuration Table

CH	TYPE	GAIN	C/F	AVG	CJC	CH	TYPE	GAIN	C/F	AVG	CJC
0	VOLTS	1		1	ON	8	VOLTS	1		1	ON
1	VOLTS	1		1	ON	9	VOLTS	1		1	ON
2	VOLTS	1		1	ON	10	VOLTS	1		1	ON
3	VOLTS	1		1	ON	11	VOLTS	1		1	ON
4	VOLTS	1		1	ON	12	VOLTS	1		1	ON
5	VOLTS	1		1	ON	13	VOLTS	1		1	ON
6	VOLTS	1		1	ON	14	VOLTS	1		1	ON
7	VOLTS	1		1	ON	15	VOLTS	1		1	ON

Figure 2-5. First Board Thermocouple Configuration Table

The options in the Thermocouple Configuration Table are as follows:

- **Type**: This field offers the following choices:

N.C. . . for	No Connection
B . . for	Type B Thermocouple
E . . for	Type E Thermocouple
J . . for	Type J Thermocouple
K . . for	Type K Thermocouple
R . . for	Type R Thermocouple
S . . for	Type S Thermocouple
T . . for	Type T Thermocouple
VOLTS . . for	Voltage

- **Gain**: When the Type field is set for *VOLTS*, the Gain field offers the following choices:

1 . .	for a gain of 1
125 .	for a gain of 125
166. .	for a gain of 166.67
400 .	for a gain of 400

- **C/F**: When the Type field is set for a thermocouple type, the C/F field offers the following choices:

C . .	for degrees Celsius
F. . .	for degrees Fahrenheit

- **AVG**: This field requires entry of a number between 0 and 99 (inclusive) to represent the number of samples to average. The number 0 specifies just one reading, while any other number specifies the number of readings to be averaged. While the sampling process cycles continuously through specified channels, the readouts at any instant are the average of the number of samples specified in the *AVG* column.
- **CJC**: This field specifies the status of CJC (Cold Junction Compensation), as follows:

  ON turns CJC On
  OF turns CJC Off

- **OK/Cancel**: Use this field to return to the Board Menu. Use *OK* to return to the Board Menu while retaining new entries. Use *Cancel* to return to the Board Menu while losing any new entries.

2.6 BOARD INSTALLATION

WARNING

Any attempt to insert or remove a board with computer power on can damage your computer!

To install the DAS-TC board in a computer, proceed as follows:

1. Turn Off power to the computer and all attached equipment.
2. Remove the computer chassis cover.

3. Select an available accessory slot. Loosen and remove the screw at the top of the blank adapter plate, then slide the blank plate up and out to remove.
4. Make sure the Base Address Switch is properly set (refer to the Section 2.3).
5. Insert and secure the board.
6. Replace the computer cover.
7. Plug in all cords and cables.
8. Return power to the computer.

You are now ready to make I/O connections. Appendix A contains a pinout of the DAS-TC I/O connector. Chapter 3 describes the accessories for I/O connections.

■ ■ ■

Chapter 3

I/O CONNECTION METHODS

DAS-TC I/O connections require the use of either an STC-TC or an STA-TC accessory panel. This chapter describes these accessory panels and their use. Refer to Appendix A for a pinout of the DAS-TC I/O connector.

3.1 USING THE STC-TC

The STC-TC is a screw-terminal board that plugs directly into the I/O connector of the DAS-TC. This board supports all 16 differential channels of the DAS-TC and contains a CJC sensor that provides a correction reference for temperature measurements on all channels. The CJC sensor requires no calibration.

The terminal blocks reside on an isothermal bar, a metal plate that keeps the thermocouple terminations and the CJC sensor at the same temperature. One of the terminal blocks contains four terminals at PC ground potential for grounding shields, and miscellaneous purposes.

The STC-TC also contains two 8-position DIP switches to enable/disable the open-thermocouple-detection circuitry on a channel-by-channel basis; open-thermocouple detection is enabled for a channel when the switch for that channel is on. Board labeling for the switches is *CH0* (for Channel 0) and *CH15* (for Channel 15); these labels are positioned to indicate which switches correspond to which channels. To gain access to these switches, remove the board's top cover. If you are measuring a voltage source (instead of a thermocouple) with a very high output impedance, you may want to disable the corresponding OPEN-THERMOCOUPLE-DETECTION CIRCUITRY to get a more accurate reading.

Figure 3-1 depicts the connection method for the STC-TC and the DAS-TC. Figure 3-2 shows the screw-terminal and switch layout of the STC-TC.

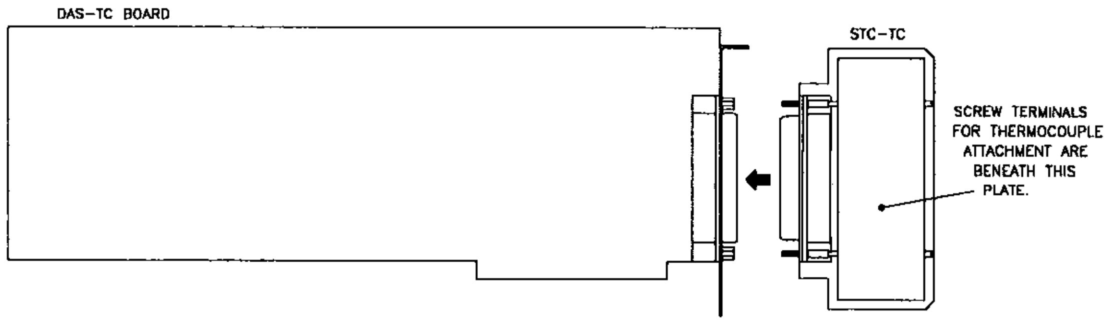

DAS-TC BOARD
STC-TC
SCREW TERMINALS FOR THERMOCOUPLE ATTACHMENT ARE BENEATH THIS PLATE.

Figure 3-1. STC-TC and DAS-TC

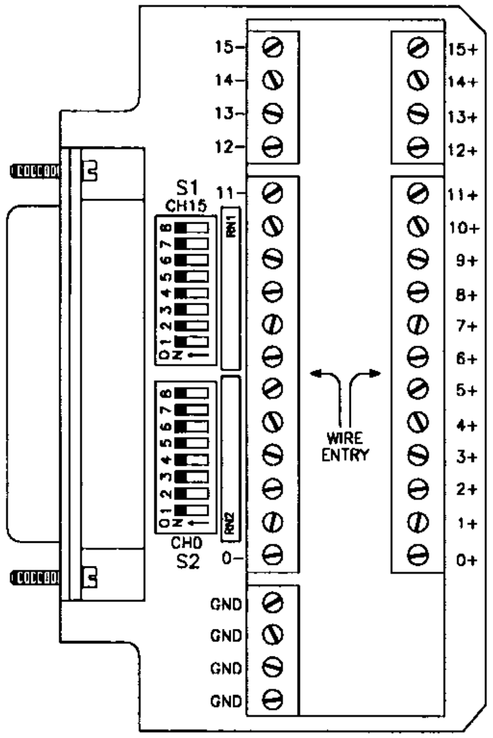


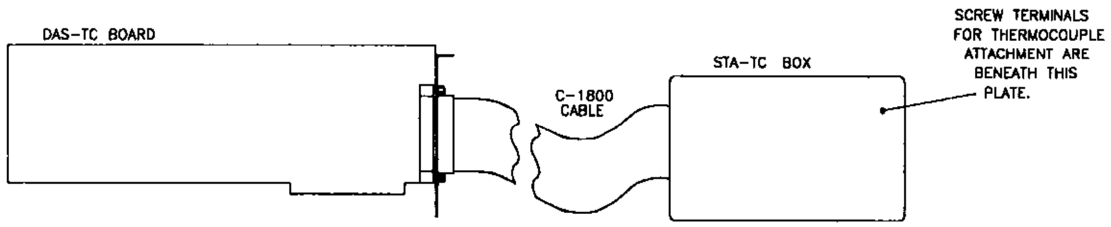
Figure 3-2. STC-TC Terminal and Switch Layout

3.2 USING THE STA-TC

The STA-TC is a screw-terminal expansion box you may use when you can not bring thermocouple leads to the rear panel of the host computer. Use a C-1800 accessory cable to connect this box to the computer.

All terminal blocks reside on an isothermal bar, a metal plate that keeps the thermocouple terminations and the CJC sensor at the same temperature. One of the terminal blocks contains four terminals at PC ground potential for grounding shields, and miscellaneous purposes. Figure 3-3 depicts the connection method for the STA-TC and the DAS-TC. Figure 3-4 shows the screw-terminal and switch layout of the STA-TC.

The STA-TC also uses two 8-position DIP switches to enable/disable the open-thermocouple-detection circuitry on a channel-by-channel basis; open-thermocouple detection is enabled for a channel when the switch for that channel is on. Board labeling for the switches is *CH0* (for Channel 0) and *CH15* (for Channel 15); these labels are positioned to indicate which switches correspond to which channels. If you are measuring a voltage source (instead of a thermocouple) with a very high output impedance, you can disable the corresponding OPEN-THERMOCOUPLE-DETECTION CIRCUITRY to get a more accurate reading.



DAS-TC BOARD
C-1800
CABLE
STA-TC BOX
SCREW TERMINALS FOR THERMOCOUPLE ATTACHMENT ARE BENEATH THIS PLATE.

Figure 3-3. STA-TC and DAS-TC

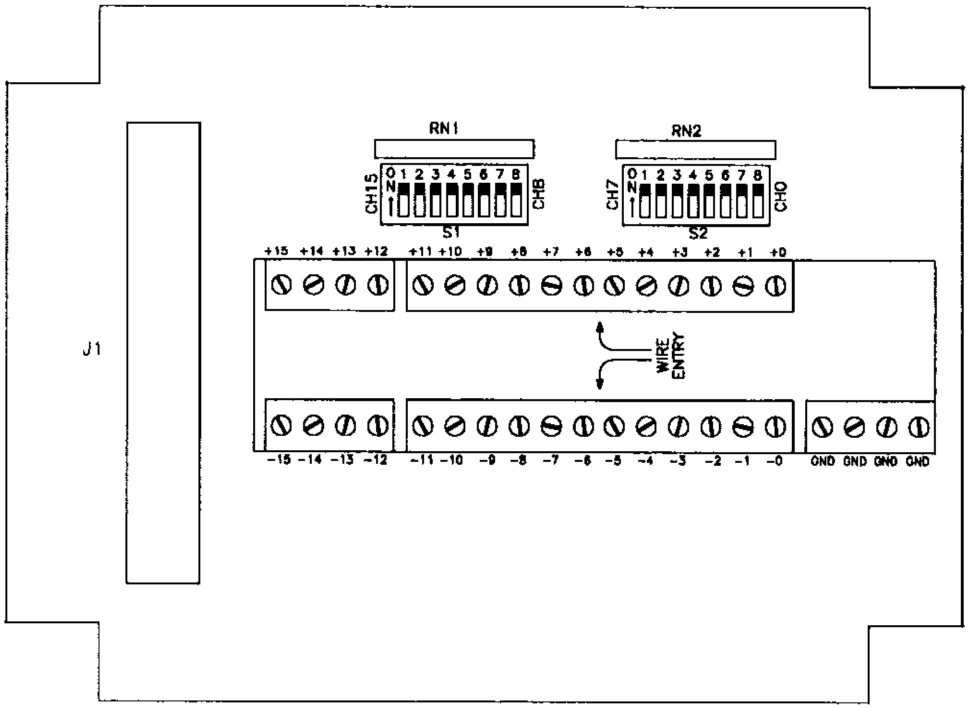


Figure 3-4. STA-TC Terminal and Switch Layout

3.3 THERMOCOUPLE CONNECTIONS

A thermocouple is a *floating source*: it has no connection to ground at the signal source, and it can be wired to the STC-TC or STA-TC inputs as shown in Figure 3-5. Note that a ground connection is available on either accessory panel to connect a shield, in the event the thermocouple wires are shielded.

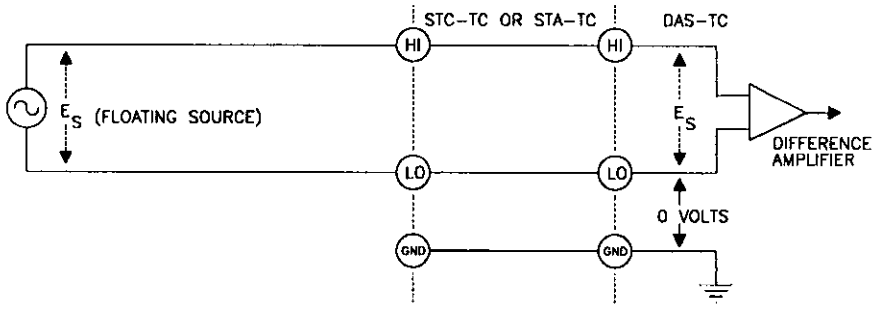


Figure 3-5. Thermocouple Connection

3.4 VOLTAGE SOURCES

A differential input responds only to the difference signals between the high (HI) and low (LO) inputs of a channel. In practice, the signal-source ground and the computer ground (where the DAS-TC is) will be at slightly different potentials, as they are connected through the ground returns of the equipment and the building wiring. The difference between the ground voltages (see Figure 3-6) forms a common-mode voltage (Vcm), a voltage that is common to both inputs and that a differential input is designed to reject (up to a certain limit).

If you have a combination of floating- and ground-referred signal sources, connect the ground-referred signals as shown in Figure 3-6 and the floating signals as shown in Figure 3-7.

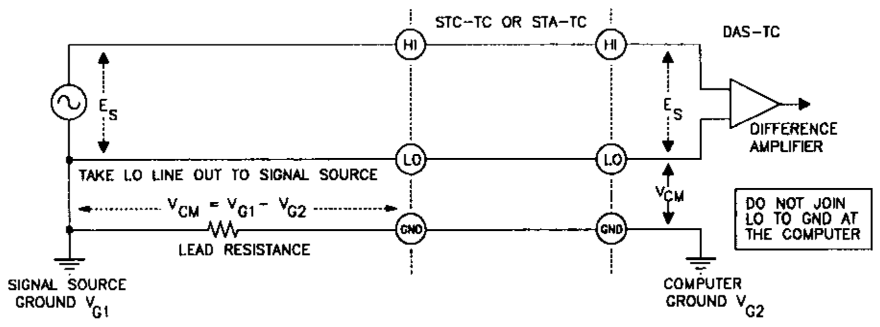

STC-TC OR STA-TC
DAS-TC
HI
HI
E_S
E_S
DIFFERENCE AMPLIFIER
LO
LO
TAKE LO LINE OUT TO SIGNAL SOURCE
$V_{CM} = V_{G1} - V_{G2}$
V_{CM}
DO NOT JOIN LO TO GND AT THE COMPUTER
GND
GND
LEAD RESISTANCE
SIGNAL SOURCE GROUND V_{G1}
COMPUTER GROUND V_{G2}

Figure 3-6. Connecting a Grounded Signal Source.

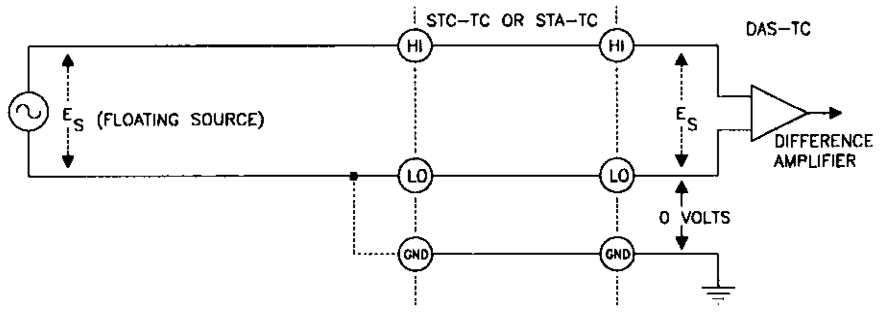


Figure 3-7. Connecting a floating source to a differential input.

Note in Figure 3-7 that you may make a connection between LO and GND at the accessory-panel input to maintain the input voltage being measured within the common-mode range of the DAS-TC. However, the LO inputs of the DAS-TC have 10 kΩ resistors to GND, making most external LO to GND connections unnecessary.

3.5 PRECAUTIONS

A short between computer inputs and AC lines can seriously damage your computer. The manufacturer can accept no liability for this type of accident. To avoid this problem, secure all input connections to prevent signal wires from coming loose and shorting to high voltages.

■ ■ ■

Chapter 4

DATALOGGER

The DATALOGGER program is a visual interface for viewing and recording samples of readings from the DAS-TC. You can use this program to check out the DAS-TC and your application or to acquire data for plotting or analysis by another package. The DATALOGGER allows you to specify the number of channels to be sampled and the order of sampling. The program also permits you to specify a series of sampling scans at convenient intervals.

This chapter explains DATALOGGER requirements, functions, and use. The program operates with mouse and/or keyboard.

4.1 STARTUP

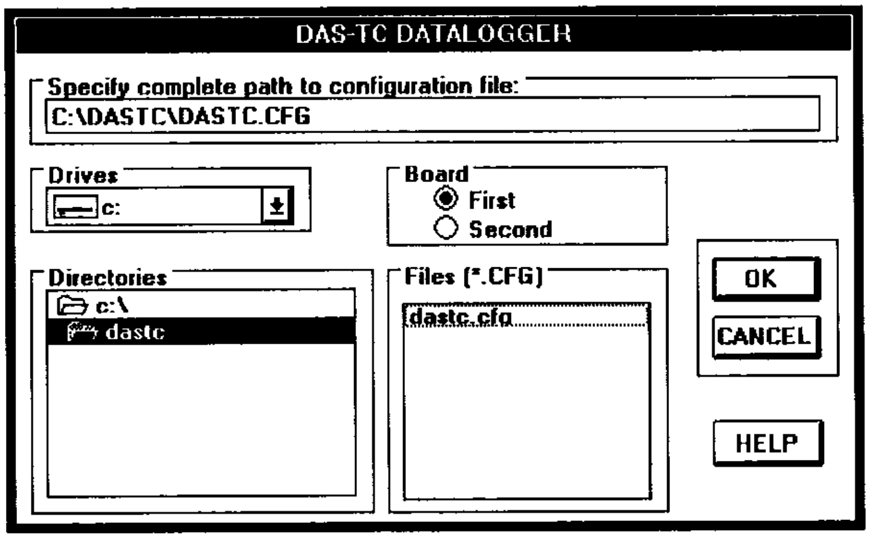
Software for the basic DAS-TC package is for DOS only, while software for the ASO-TC package is for DOS and Windows. Since the utilities (the Configuration File Generator and the DATALOGGER programs) for DOS and for Windows are similar in appearance and functionally equivalent, they are both covered by this manual. This section therefore includes startup instructions for both DOS and Windows.

Note that the DATALOGGER screens in this manual are the product of the Windows-based DATALOGGER program. While the DOS- and Windows-based screens differ slightly in appearance, they are functionally equivalent.

Starting from DOS

Use the following steps to start the Configuration File Generator program from DOS:

1. Change to the DASTC directory and type **TCDLOGER** + **[Enter]** at the DOS prompt.
2. When the Help screen appears, press **[Enter]** to transfer to the path-specification screen, which is similar to the diagram in Figure 4-1.


DAS-TC DATALOGGER

Specify complete path to configuration file: C:\DASTC\DASTC.CFG

Drives: c:

Board: First [✓] / Second

Directories: c:\ dastc

Files (*.CFG): dastc.cfg

Figure 4-1. Configuration File Path-specification Screen

Starting from Windows

Use the following steps to start the DATALOGGER program from Windows:

1. Run the Windows program.
2. In the Program Manager screen, click twice on the *DAS-TC* icon to expand it to the DAS-TC program screen.
3. From the DAS-TC program screen, click twice on the *DATALOGGER* icon to run the DATALOGGER program.
4. When the Help screen appears, press [**Enter**] to transfer to the path-specification screen, shown in Figure 4-1.

Path-specification Screen

In the path-specification box of the screen in Figure 4-1, the default configuration file and its path for the DOS program are *C:\DASTC\DASTC.CFG*. The default configuration file and its path for the Windows program are *C:\ASOTC\DASTC.CFG*. The default board is *First*.

To accept the path defaults, move the cursor to the *OK* button and click, or press [**Enter**].

To create a totally new configuration file and path, tab to the path-specification box and use the keyboard to overwrite the contents.

To select a different configuration file and/or path, use the following procedure:

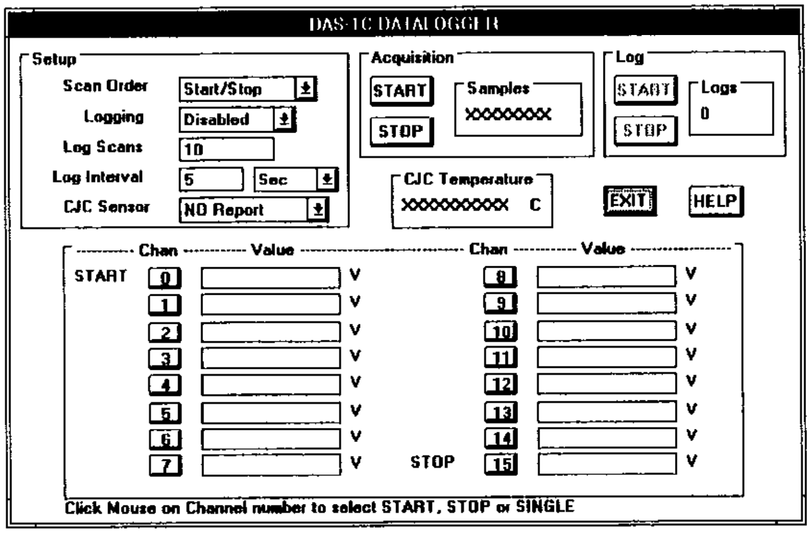
1. Move to the *Drive* selector, and use mouse or keyboard to scan the list of drives available on your system and select a drive. The selected drive appears in the *Directories* option box with a list of its directories.
2. Move to the *Directories* box, scan the list of available directories, and select one.
3. Move to the *Files *.CFG* box, scan the list of available configuration files, and select one.

4. Move to the *Board* box, and choose *First* to specify the configuration file parameters for First board or choose *Second* to specify configuration file parameters for the Second board. Note that the DATALOGGER program addresses only one board per configuration session. To switch the DATALOGGER from one board to the other, exit and restart the program; then, specify the other board in the path-specification screen.

4. Move to the *OK* button, and click or press **[Enter]** to transfer to the DATALOGGER screen. Note that clicking or pressing **[Enter]** at the *Cancel* button ends the DATALOGGER program.

4.2 THE DATALOGGER SCREENS

Figure 4-2 shows the DATALOGGER screen for a Start/Stop scan order, and Figure 4-3 shows the DATALOGGER screen for a Queue scan order.



DAS-TC DATALOGGER

Setup: Scan Order: Start/Stop; Logging: Disabled; Log Scans: 10; Log Interval: 5 Sec; CJC Sensor: NO Report

Acquisition: START, STOP; Samples: XXXXXXXX

Log: START, STOP; Logs: 0

CJC Temperature: XXXXXXXXXX C

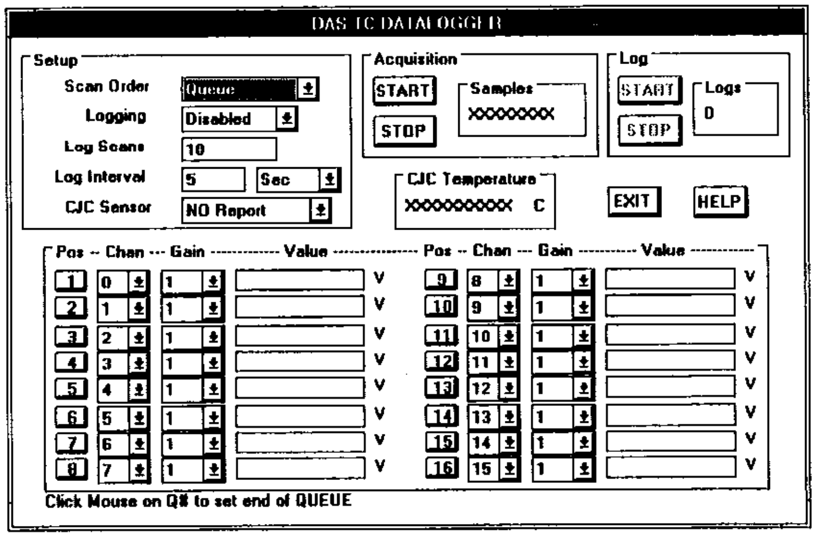
EXIT HELP

Chan — Value — Chan — Value

START 0 … 7 V; 8 … 14, STOP 15 V

Click Mouse on Channel number to select START, STOP or SINGLE

Figure 4-2. DATALOGGER Screen for Start/Stop Scan Order



DAS-TC DATALOGGER

Setup: Scan Order: Queue; Logging: Disabled; Log Scans: 10; Log Interval: 5 Sec; CJC Sensor: NO Report

Acquisition: START, STOP; Samples: XXXXXXXX

Log: START, STOP; Logs: 0

CJC Temperature: XXXXXXXXXX C

EXIT HELP

Pos	Chan	Gain	Value	Pos	Chan	Gain	Value
1	0	1	V	9	8	1	V
2	1	1	V	10	9	1	V
3	2	1	V	11	10	1	V
4	3	1	V	12	11	1	V
5	4	1	V	13	12	1	V
6	5	1	V	14	13	1	V
7	6	1	V	15	14	1	V
8	7	1	V	16	15	1	V

Click Mouse on Q# to set end of QUEUE

Figure 4-3. DATALOGGER Screen for Queue Scan Order

Control options for the two DATALOGGER screens are *Setup, Acquisition,* and *Log*. These options are discussed in the following sections.

Setup

The Setup box contains control options for scan order, logging, log scans, log intervals, and the CJC sensor, as follows:

- ***Scan Order***

  Choose *Start/Stop* or *Queue*. Each of these two choices has its own channel-setup screen, as shown in Figures 4-2 and 4-3.

  A Start/Stop scan begins at the channel marked *Start* and progresses sequentially to the channel marked *Stop*. You may set any channel as a Start channel and any other channel as a Stop channel.

  A Queue scan begins at Position 1 and progresses sequentially through the remaining positions. To set a series of positions for a particular sequence of channels, set the channel numbers at each position to follow the desired sequence. For example, to scan Channel 10 first, Channel 6 second, Channel 13 third, set Position 1 for Channel 10, Position 2 for Channel 6, and Position 3 for Channel 13. Note that channel numbers can be repeated in a sequence.

- ***Logging***

  Choose *Disabled* to turn logging off or *Enabled* to turn logging on. You must enable logging before you can set the *Log* box for *START*.

  When you choose *Enabled*, you are presented with a screen for specifying a path to the logging file. The default file and path are C:\DASTC\DASTC.LOG for DOS and C:\ASOTC\DASTC.LOG for Windows. To accept or change the default, refer to the instructions for the path-specification screen, in Section 4.1.

  When you accept an existing logging file and path, you are asked whether to *Append* or *Overwrite*. If you choose to Append, the new log will begin at the end of the existing logging file contents. If you choose to Overwrite, the new log will overwrite the existing logging file contents. If you choose *Cancel*, the screen for specifying a path to the logging file returns so that you can change the file name or exit the logging function.

- ***Log Scans***

  A log scan is a single sampling and recording of the block of channels defined by the Start/Stop screen or the Queue screen. Your entry in this box determines the number of times a log scan is to be performed. Enter any number that fits the Log Scans box.

- ***Log Interval***

  This entry sets the interval between successive log scans in seconds or minutes. Enter a number between 1 and 100 (inclusive), and choose *Sec* for seconds or *Min* for minutes.

- ***CJC Sensor***

  Choose *NO Report* to turn off CJC temperature reporting or choose *Report* to turn it on. When reporting is on, the temperature of the CJC sensor (on the STA-TC or the STC-TC) appears in the CJC Temperature box as soon as data acquisition begins.

Acquisition

This box starts/stops the channel sampling process for a block of channels specified by the Scan Order. If you wish to visually monitor sampling activity but not to record it, you can start channel sampling while logging is off; however, if you wish to record sampling activity, you must turn on both the logging and the channel sampling. The *Acquisition* options are as follows:

- ***Start***

  Turns channel sampling On. When sampling starts, *Stop* is highlighted. Sampling proceeds according to the specifications you have defined in the *Setup* box. Readings for the scanned channels appear in the *Value* column of the channel table.

- ***Stop***

  Turns channel sampling Off. When sampling stops, *Start* is highlighted.

- ***Samples***

  Tracks the number of samples. This number increments by the number of sampled channels each time a scan is completed.

- ***CJC Temperature***

  This box presents the temperature of the CJC sensor as soon as data acquisition begins. The temperature reading is updated with each new scan.

Log

This box controls logging activity. Before logging can start, however, *Logging* must be set to *Enabled* and *Acquisition* must be set for *Start*. The *Log* options are as follows:

- ***Start***

  Start turns logging On. When logging starts, *Stop* is highlighted and logging proceeds through the specified number of log scans. At the same time, readings in the *Value* column of the channel table are written to the specified log file.

- ***Stop***

  Turns logging Off. When logging stops, *Start* is highlighted.

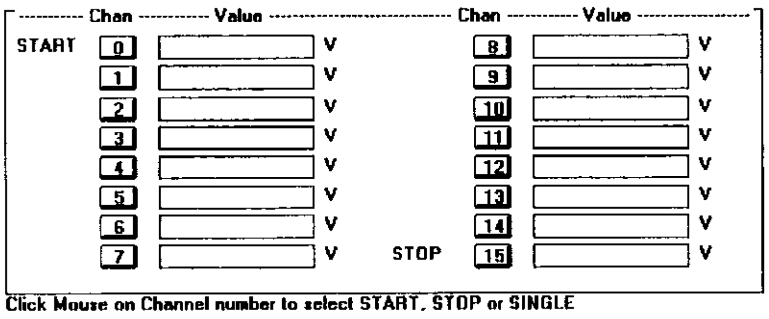
- ***Logs***

  Tracks the number of logging scans. This number increments by one with each completed logging of a scan.

  > *NOTE:* You may restart logging as often as you wish after it ends by itself or you stop it.

Start/Stop Scan Table

When the *Scan Order* is set for *Start/Stop,* the channel scan table appears as shown in Figure 4-4.



	Chan	Value			Chan	Value	
START	0		V		8		V
	1		V		9		V
	2		V		10		V
	3		V		11		V
	4		V		12		V
	5		V		13		V
	6		V		14		V
	7		V	STOP	15		V

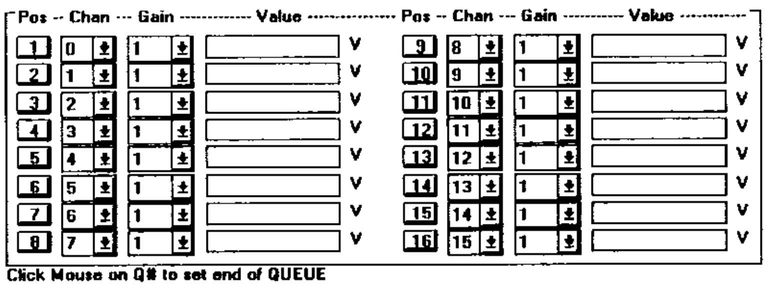
Click Mouse on Channel number to select START, STOP or SINGLE

Figure 4-4. Start/Stop Scan Table

Clicking or pressing [**Enter**] on any number in the *Chan* column displays a menu with the choices of *Start, Stop,* or *Single.* Choose *Start* to specify the corresponding channel as the start point for a scan. Choose *Stop* to specify the end point for a scan. Choose *Single* to specify the corresponding channel as the only channel to be sampled.

Queue Scan Table

When the *Scan Order* is set for *Queue,* the scan table appears as shown in Figure 4-5.



Pos	Chan	Gain	Value		Pos	Chan	Gain	Value	
1	0	1		V	9	8	1		V
2	1	1		V	10	9	1		V
3	2	1		V	11	10	1		V
4	3	1		V	12	11	1		V
5	4	1		V	13	12	1		V
6	5	1		V	14	13	1		V
7	6	1		V	15	14	1		V
8	7	1		V	16	15	1		V

Click Mouse on Q# to set end of QUEUE

Figure 4-5. Queue Scan Table

Control options for the Queue scan table are as follows:

- ***Pos***

  The number in the *Pos* column indicates the position in the scan order, with Position 1 as the starting position. Click or press [**Enter**] on any position number to mark the end of a scan.

- ***Chan***

  The number at any position in the *Chan* column indicates the channel to be scanned at that position. Click or press [**Enter**] on any number in the *Chan* column to display a list of channel numbers (0 to 15). Choose the channel number to be scanned at that position of the scan order.

- ***Gain***

  Refers to the gain setting for voltage measurements. Use this setting only for channels you have assigned (in the configuration file) to measure voltage. Click or press **[Enter]** on a Gain box to display a list of gains. Select the desired gain. The gain you select supersedes the gain setting in the configuration file. If you set no gain for a voltage channel, the gain setting in the configuration file remains in effect.

Exit

Use *Exit* to end the program.

4.3 THE LOG

The following text and data is an example of a logging file for a block of five channels. Note that the recorded data is preceded by a header containing configuration settings. Also note that the user stopped the logging process and that stop time is recorded.

```
DASTC LOGFILE 05-13-1993
NUMBER TYPE = INTEGER
NMR Frequency = 60 HZ
Logging Interval = 5 Sec

CJC Reporting is Enabled

START/STOP CONFIGURATION

CHANNEL       TC TYPE     GAIN          UNITS          FILTER         CJC
 0             VOLTS        1             C              1              ON
 1             VOLTS        1             C              1              ON
 2             VOLTS        1             C              1              ON
 3             VOLTS        1             C              1              ON
 4             VOLTS        1             C              1              ON
 5             VOLTS        1             C              1              ON

11:37:03 25.62 8.108375 7.599875 7.094375 6.58475 6.075125 5.565875

11:37:08 25.64 8.108375 7.599875 7.094375 6.585125 6.0755 5.565875

11:37:13 25.62 8.108 7.599875 7.094 6.585125 6.0755 5.565875

11:37:18 25.62 8.108375 7.60025 7.094375 6.58475 6.075125 5.5655

11:37:23 25.62 8.108375 7.599875 7.094375 6.5855 6.0755 5.5655

User stopped Log Run at 11:37:24
```

Note that when CJC Reporting is enabled, the CJC temperature is the first reading (following the time) on each line.

■ ■ ■

❐

❐

❐

Chapter 5

CALIBRATION

DAS-TC calibration is a simple process of setting the voltage across two of the board's test points, which are shown in Figure 5-1. The process requires a 5 1/2 digit DVM (Digital Voltmeter) or a DMM (Digital Multimeter) such as a Keithley Instruments 196 DMM.

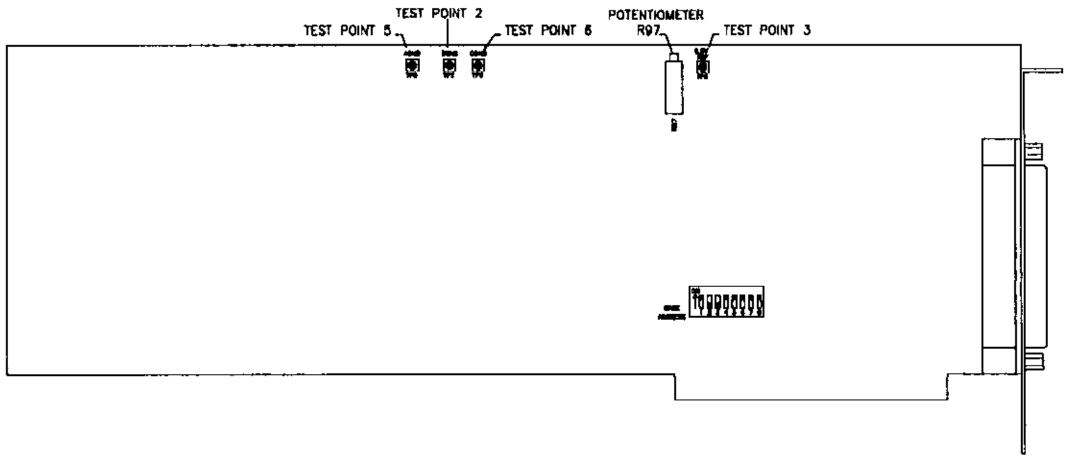


Figure 5-1. Test Points

Perform the DAS-TC calibration process as follows:

1. Referring to the diagram, connect the positive (+) lead of the meter to TP3 (Test Point 3), the 9.9 V reference.
2. Connect the negative (-) lead of the meter to TP2, the BGND point.
3. Adjust Potentiometer R97 for a meter reading of 9.9000 ±0.0001 V.

■ ■ ■

❐

❐

❐

Chapter 6

TROUBLESHOOTING

This chapter discusses an approach to isolating a problem in a DAS-TC system. If the problem appears to be on the DAS-TC board, you may refer to the troubleshooting section of this chapter to look for a possible cause and its solution. If the problem appears serious enough to require technical support, the last section of this chapter discusses what you must do to contact an applications engineer.

6.1 PROBLEM ISOLATION

If a problem appears in your DAS-TC system, proceed as follows to determine whether it is in the PC, on a DAS-TC board, or in the I/O circuitry:

1. Remove power connections to the host PC.
2. While keeping thermocouple/voltage connections intact on the STC-TC or STA-TC, unplug the accessory connector(s) from the DAS-TC(s). Remove the DAS-TC(s) from the PC and visually check for damage. If a board is obviously damaged, refer to Section 6.3 and request technical support.
3. With the DAS-TC board(s) out of the PC, check the PC for proper operation. Power up the PC and perform any necessary diagnostics.
4. When you are sure the PC is operating properly, remove PC power again, and install a DAS-TC you know to be functional (if one is available). Do not make any I/O connections.
5. Apply PC power and check operation with the functional DAS-TC in place. This test checks the PC accessory slot. If the PC contained two DAS-TCs when the problem appeared, check the other slot, as well.
6. If the accessory slots are functional, check the I/O hookups. Connect the STC-TC and/or STA-TC with thermocouple hookups, one at a time, and check operation.
7. If operation fails for an I/O hookup, check the individual thermocouples and the STA-TC or STC-TC. Note that if a thermocouple circuit is open, the reading for its corresponding channel will be well above normal. The reading for a shorted thermocouple circuit, however, will equate to 0 V and may be difficult to detect unless it is much lower than the expected reading.
8. If operation remains normal to this point, the problem is in the DAS-TC board(s) originally in the system. If there were two boards, try them one at a time in the PC to determine which is faulty. Then, use the troubleshooting information in the next section to try to isolate the problem.
9. If you cannot isolate a problem, refer to Section 6.3 for instructions on obtaining technical support.

6.2 TROUBLESHOOTING TABLE

The following table lists general symptoms and possible solutions for problems with the DAS-TC. If checking out a board according to this table fails to lead to a satisfactory solution, refer to Section 6.3 for instructions on obtaining technical support.

Symptom	Possible Cause	Possible Solution
Dead board:	1. Base Address is unacceptable.	Reset Base Address DIP Switch (refer to Chapter 2 for instructions).
	2. Incorrect alignment of board in accessory slot.	Check installation.
	3. Damaged board.	Contact technical support (see Section 6.3).
Intermittent operation:	1. Vibration or loose connections.	Cushion source of vibration and tighten connections.
	2. Overheating.	Check environmental ambient.
	3. Electrical noise.	Check need for better shielding or thermocouple wires, or reroute wiring.
Thermocouple reading is inordinately high:	1. Open thermocouple circuit.	Check thermocouple and wiring. Check Open Thermocouple Detection jumpers on STC-TC/STA-TC. Check DATALOGGER channel table list to see that problem channel is included.
	2. Incorrectly specified thermocouple type.	Check type definition in configuration file.
	3. Incorrect thermocouple type.	Check thermocouple type temperature range.
Voltage reading is inordinately high:	1. Open connection.	Check wiring to accessory terminal block. Check DATALOGGER channel table to see that channel is included.
	2. Configuration file error.	Check configuration file for correct entries.

Thermocouple reading is inordinately low:	1. Incorrect thermocouple type.	Check thermocouple type and temperature range.
	2. Configuration file error.	Check configuration file for correct entries.
Voltage reading is inordinately low:	1. Open circuit.	Check wiring to accessory terminal block.
	2. Configuration file error.	Check configuration file for correct entries.
Apparent system lockup after certain Function Calls:	Timing error.	Try pressing [**Ctrl**] + [**Break**] keys.

6.3 TECHNICAL SUPPORT

Before returning any equipment for repair, please call 508/880-3000 to notify Keithley MetraByte applications engineering. To save time, please be ready to furnish the following information:

- The invoice or order number (located on the shipping paper) for the equipment.
- All board names with serial numbers and revision codes (refer to the sticker on the component side of the equipment).
- The base address setting.
- The interrupt level setting (if used).
- The revision number of the Function Call Driver (if used).
- The name of the language or software you are using with its revision number.
- The name of the Keithley MetraByte software package with serial and revision numbers (and ESP number, if applicable).
- A profile of your computer, as follows:
  Brand name (if available).
  Type (XT, AT, PS1, PS2, and so on).
  CPU (8086, 80286, 80386, and so on).
  Monitor (EGA, VGA, and so on)
  Amount of RAM.
  Coprocessor type (if applicable).

If possible, the applications engineer will resolve your problem by telephone. If a telephone resolution is not possible, the applications engineer will issue you a Return Material Authorization (RMA) number and ask you to return the equipment. Please reference the RMA number in any documentation regarding the equipment and on the outside of the shipping container.

Note that if you are submitting your equipment for repair under warranty, you must furnish the invoice number and date of purchase.

When returning equipment for repair, please include the following information:

1. Your name, address, and telephone number.
2. The invoice number and date of equipment purchase.
3. A description of the problem or its symptoms.
4. The RMA number on the outside of the package.

Repackage the equipment. Handle it with ground protection; use its original anti-static wrapping, if possible. Ship the equipment to

ATTN: RMA #__________
Repair Department
Keithley MetraByte
440 Myles Standish Boulevard
Taunton, Massachusetts 02780

Telephone 508/880-3000
Telex 503989
FAX 508/880-0179

■ ■ ■

Chapter 7

FUNCTION CALL OVERVIEW

If you prefer writing your own programs for DAS-TC applications, your Distribution Software includes supporting files for a *Function Call Driver.* This driver is a library of the DAS-TC functions you are most apt to use and is included in the Distribution Software.

Each function in the Function Call Driver has its own name. Thus, each function in the driver is individually executable from an application program by means of a call statement, referred to herein as a *function call.* The Function Call Driver supports application programs in the following BASIC languages:

- QuickBASIC, Versions 4.0 and 4.5
- Professional BASIC, Version 7.0 and above
- Visual Basic For DOS, Version 1.0

Each Function Call incorporates all required communications with the DAS-TC microprocessor. You do not have to concern yourself with the additional programming required for these processes. Instead, you need only familiarize yourself with the purposes of the individual Function Calls and their parameter requirements.

This chapter discusses the operations available for use with function calls and covers the setup and use of the Function Call Driver for the languages listed above. As a reference for setup and usage in typical applications, the Distribution Software contains example programs. The programs QBEXAMP1.BAS through QBEXAMP3.BAS are compatible with QuickBASIC, Professional BASIC, and Visual Basic. You may wish to refer to one or more of these programs while reading this chapter.

The final section of this chapter lists and briefly describes each of the calls in the DAS-TC Function Call Driver. Chapter 8 gives detailed descriptions and requirements for each call.

7.1 AVAILABLE OPERATIONS

The DAS-TC provides you with two types of analog-to-digital (A/D) input operations:

- Single-call
- Frame-based

The following subsections describe these operations in more detail.

You can implement both types of operations with the function calls. As with any call, you declare corresponding arguments before making the call, and you pass parameters while making the call.

Single-call A/D Input Operations

In a single-call A/D input operation, you read an analog input value using a single call to a function. The function automatically performs Analog-to-Digital conversion. You specify the attributes of the operation, such as the board to execute the operation, the channel from which to read data, and the buffer in which to store the data, as arguments to the function. The function returns data as a voltage or temperature value; voltage is in volts, temperature in degrees.

> *Note*: The Function Call Driver reads the configuration file to determine the gain; therefore, the gain parameter must be set to 0.

Use the **KADRead** function to read a single analog input value from a specified analog input channel.

The **DASTCGETCJC** function is a special-purpose single-call function for obtaining reading the value of the CJC (cold junction compensation) sensor in degrees (Celsius). You can use the CJC sensor value for temperature correction of a thermocouple not supported by the DAS-TC.

If you wish, you may use **KADRead** or **DASTCGETCJC** in a software loop to acquire more than one value from the channel. Typically, however, you may want to acquire more than one value from a single channel, exercise more control over the data transfer than is possible with single-call operations, or do both. In such cases, use a frame-based operation, described next.

Frame-based A/D Input Operations

A frame-based input operation normally samples more than one value from one or more channels. In the case of the DAS-TC, the data returned consists of as many voltage or temperature values as there are analog input samples. The operation returns the values in volts or degrees.

A frame-based operation uses a single data structure called a *frame* to represent the controllable attributes of the operation. The Function Call Driver for the DAS-TC provides four A/D frames per board. The controllable attributes of the operation, such as the start channel, stop channel, and number of samples, are the frame's *elements*. The attributes and their settings are represented in the frame's internal data structure.

A frame-based operation is a sequence of function calls. At minimum, a frame-based sequence includes functions that manage and set frame elements, followed by a function that performs the actual transfer of data values.

The table below lists the frame elements available for the DAS-TC and the corresponding function call used to set each frame element. You may find valid settings for a frame element in the description of the function that sets the element; the descriptions of each function are in Chapter 8.

Element	**Function Call**
Start/Stop Channel	KSetStartStopChn
Channel-Gain Array Address	KSetChnGAry
Number of Samples	KSetBufL or KSetBufR
Data Buffer Address	KSetBufL or KSetBufR
Buffering Mode	KSetContRun, KClrContRun

Once you set the frame's elements, you pass the all of the settings using only the frame handle (which identifies the frame) to the function that starts the A/D operation. By passing the same frame handle, other A/D operations can use the same settings.

> *Note*: Each of the BASIC programming languages is supported by an Include file containing a definition of the FRAMEH variable type. Therefore, you must declare all frame handles to be of this type.

In the DAS-TC, frame-based A/D operations run in either of two modes:

- Synchronous
- Interrupt

The difference between the two modes is the way each executes the data acquisition and transfer (to a buffer) phase of an application program. Data acquisition and transfer is the final phase of an application program; it is preceded by loading the INCLUDE file, declaring and dimensioning all variables, opening the hardware configuration file, and initializing the Function Call Driver and the board, as explained in Section 7.2. In the DAS-TC, you begin the data acquisition and transfer phase using either **KSyncStart** for *synchronous* mode or **KIntStart** for *interrupt* mode.

KSyncStart starts the synchronous mode by immediately shifting control of operations from the application program to the Function Call Driver. Control remains with the driver until the data acquisition and all transfers to buffer are complete. As long as the driver controls operations, the application program is unable to perform any other functions. If your application program is intended strictly for data acquisition and transfers to buffer, synchronous mode is a good choice. However, if you wish to add other functions during the board's data acquisition periods, you may want to try interrupt mode.

To use the interrupt mode, you must have access to one of the supported interrupt levels (2, 3, 4, 5, 6, or 7). **KIntStart** starts the interrupt mode without a shift of control. Instead, the control of operations remains with the applications program through the acquisition of each block of data. Each time the DAS-TC RAM accumulates a block of data, the application program's interrupt handler issues an interrupt and shifts control of operations to the Function Call Driver for the duration of the interrupt. The interrupt lasts just long enough to allow transfer of the block of data to the buffer. The period of the interrupt is minute in comparison with the time required for the DAS-TC RAM to accumulate a block of data. Though the application program is in control of operations during data accumulation, it is idling while the DAS-TC is working. The idle time is actually enough to allow the program to conduct other functions. You may insert the code for other functions after your call for **KIntStart**.

NOTE

The DAS-TC transfers data in blocks, where block size = the number of channels specified. Suppose, for example, you have requested 43 samples using ten channels. The Function Call Driver actually acquires 50 values in five blocks of ten samples each. The first 40 values are transferred from the first four blocks that have been acquired, and the remaining three samples are transferred from the fifth acquired block of ten samples.

The Function Call Driver stores acquired samples in a buffer that you define. Once you have defined a buffer, use **KSetBufL** or **KSetBufR** to pass the buffer address to the Function Call Driver. You must define a buffer as an array before you call **KSetBufL** or **KSetBufR**.

You can specify either Single-cycle or Continuous buffering mode for interrupt operations. In Single-cycle mode, the specified number of samples is stored in the buffer and the operation stops automatically. Use the **KClrContRun** function to specify Single-cycle buffering mode. In Continuous mode, the board keeps acquiring the same number of new values, placing the data in the buffer until it receives the stop function, **KIntStop**. The transfer index and buffer pointers are reset before another transfer cycle is initiated, and acquired values in the buffer are overwritten. Use the **KSetContRun** function to specify Continuous buffering mode. If you do not specify Continous buffering mode, DAS-TC defaults to Single-cycle.

NOTE

If you are acquiring data using interrupts and Continuous buffering (which repeatedly acquires the same number of samples until you call **KIntStop**), as soon as the last block of samples is transferred,

- the transfer index and buffer pointer are reset to zero,
- **KIntStatus** assigns the value zero to the index parameter rather than the requested sample size, and
- the driver begins to overwrite your buffer's data.

If your application requires consecutive blocks of data, you should begin processing your buffer *before* your buffer is full, using **KIntStatus** to determine how many blocks have been transferred (this call's *index* parameter increments by block size).

7.2 OVERVIEW OF PROGRAMMING WITH THE FUNCTION-CALL DRIVER

The procedure to write a Function Call Driver program is as follows:

1. Define the application requirements.
2. Invoke the QuickBASIC or Visual Basic environment.
3. Write the program code, using the following steps:
    a. Load the Function Call Driver INCLUDE (.BI) files.
    b. Declare and dimension all variables.
    c. Open the DAS-TC hardware configuration file.
    d. Initialize the driver and board.
    e. Perform the operation-specific programming tasks.

The subsections that follow describe the details of each of these steps.

Defining the Application Requirements

Before you write the program code, have a clear idea of the operations you expect your program to execute. Also, determine the order of execution and the characteristics (number of samples, start and stop channels, and so on) of these operations. You may find it helpful to review the list of available operations in Section 7.1 and to browse through the short descriptions of the function calls, at the end of this chapter.

Invoking the QuickBASIC or Visual Basic Environment

When you invoke the QuickBasic or Visual Basic environment, you must also load the Function Call Driver Libraries if your program is to communicate with the driver. You can invoke the environment and load the driver with a single statement from the DOS command line, as follows:

- For QuickBASIC V4.0, use the following statement:

  ```
  path QB /L DTCQB40.QLB program_name
  ```

- For QuickBASIC V4.5, use the following statement:

  ```
  path QB /L DTCQB45.QLB program_name
  ```

- For Professional BASIC Version 7.0 and greater, use the following statement:

  ```
  path QBX /L DTCQBX.QLB program_name
  ```

- For Visual Basic for DOS Version 1.0, use the following statement:

  ```
  path VBDOS /L DTCVBDOS.QLB program_name
  ```

Note that *program_name* option may be an existing or a new program, and it must include the path to the name.

Writing the Program Code

Use the following steps to write the program code:

1. Load the Function Call Driver INCLUDE file

You must insert an INCLUDE statement. In QuickBASIC Versions 4.0 and 4.5, use the following statement:

```
' $INCLUDE: 'Q4IFACE.BI'
```

In Professional BASIC Version 7.0 and Visual Basic for DOS Version 1.0, use the following statement:

```
' $INCLUDE: 'Q7IFACE.BI'
```

2. Declare and dimension the variables

Declare all variables before opening the driver. Examples of these declarations are as follows:

```
DIM GAIN, CHAN AS INTEGER          'declare a variable type
DIM NumOfBrds AS INTEGER
DIM A$                             'declare A as a string
```

```
DIM DASTC AS LONG                   'declare long (32-bit) integers
DIM Derr AS INTEGER                 'error status
DIM ADValue AS LONG
```

Remember! Declare all variables before opening the driver. Using an undeclared variable in the program body (even the `I` in a simple `FOR I= 1 to 10:NEXT I` loop) causes device pointers to shift and may result in program failures during interrupt-mode operation!

3. Open the hardware configuration file

Use the **DASTCDevOpen** Function Call to open the Function Call Driver and read the configuration file (DASTC.CFG; see Chapter 2). **DASTCDevOpen** and other required initialization functions are shown in the example program at the end of this section. A quick example of using **DASTCDevOpen** and DASTC.CFG is as follows:

```
A$ = "DASTC.CFG" + CHR$(0)          'read DAS-TC config file
Derr = DASTCDEVOPEN%(SSEGADD(A$), NumOfBrds) 'execute call
```

The *SSEGADD* command tells the Function Call Driver where to look for the DASTC.CFG configuration file. SSEGADD is a standard command in Professional BASIC, QuickBASIC, and Visual Basic. And while SSEGADD is not defined in QuickBASIC Version 4.5 or lower, using the INCLUDE file Q4IFACE.BI allows its use with these versions (this INCLUDE file is loaded according to statements given in the earlier section titled *Load the Function Call "Include" File*).

4. Initialize driver and board

To allow the use of multiple boards, the Function Calls require an identifier for each board. The board identifier is called a *handle*. Without handles, a **KIntStart** or **KSyncStart** function call would not know which board (in a multi-board system) to call. The board handle is the first parameter in each of the DAS-TC function calls.

The handle for each board is determined automatically by the **DASTCGetDevHandle** Function Call.

> *Note*: A handle is a variable whose value identifies an installed board. The purpose of a handle is to provide a mechanism through which the Function Call Driver can access your board. A handle is also a convenient method for different function calls to reference the same board.

Declare all device handles to be of long type, as follows:

```
DIM DEVHANDLE AS LONG
```

5. Program the Single-call/Frame-based operations

The final step is to add the programming for completing your application objectives. Depending on the nature of your application, this step requires the use of function calls for Single-call or Frame-based operation. Refer to Section 7.3 for the options open to you for programming these operations.

7.3 PROGRAMMING OPTIONS FOR SINGLE-CALL AND FRAME-BASED OPERATIONS

Descriptions for Single-call and Frame-based operations are as follows:

- For Single-Call A/D Operations - The only task required is using the appropriate single-call A/D function (**KADRead** or **DASTCGETCJC**).
- For Frame-based A/D Operations - The tasks required for Frame-based A/D operations depend on whether you are using synchronous or interrupt mode (discussed earlier un *Frame-based A/D Operations*) and whether you are using stop and start channels or channel-gain arrays. The following table lists the four typical frame-based sequences for QuickBASIC, Professional BASIC, and Visual Basic.

Operation Mode	Method of Specifying Acquisition Channels
Synchronous	Start/Stop channels
Synchronous	Channel-gain array
Interrupt	Start/Stop channels
Interrupt	Channel-gain array

These sequences are summarized in the following subsections.

Synchronous, Start/Stop Channels

Use this calling sequence if performing a synchronous transfer, using Start/Stop channels and a buffer only. Before calling the functions in the sequence, define a local buffer as an array of four-byte elements.

1. Call **KGetADFrame** to get the handle to an A/D frame.
2. Call **KSetBufL** or **KSetBufR** to assign the buffer address previously obtained to the Buffer Address element in the frame.
3. Call **KSetStartStopChn** to assign values to the Start and Stop Channel elements in the frame.
4. Call **KSyncStart** to start the operation. Data is stored in the buffer.
5. Call **KFreeFrame** to return the frame to the pool of available frames obtained, unless you are starting another sequence that uses the same frame.

Synchronous, Channel-gain Array

Use this calling sequence if performing a synchronous transfer, using a channel-gain array. Before calling the functions in the sequence, define a buffer as an array of 4-byte elements.

1. Call **KGetADFrame** to get the handle to an A/D frame.
2. Define and assign values to a channel-gain array.
3. Call **KSetBufL** or **KSetBufR** to assign the buffer address previously declared to the buffer address element in the frame.

4. Call **KSetChnGAry** to assign the channel-gain array to the channel-gain array address element in the frame.
5. Call **KSyncStart** to start the operation. Data is stored in the buffer.
6. Call **KFreeFrame** to return the frame to the pool of available frames.

Interrupt, Start/Stop Channels

Use this calling sequence if performing an interrupt-mode operation using start/stop channels. Before calling the functions in the sequence, define a buffer as an array of 4-byte elements.

1. Call **KGetADFrame** to get the handle to an A/D frame.
2. Call **KSetBufL** or **KSetBufR** to assign the buffer address previously declared to the buffer address element in the frame.
3. Call **KSetStartStopChn** to assign values to the start- and stop-channel elements in the frame associated with the frame handle previously obtained.
4. Call **KIntStart** to start the operation. If you want the application program to perform additional functions during the data-acquisition periods, add the code after the **KIntStart** call.
5. Call **KIntStatus** to monitor the status of the operation. When completion is detected, the data is available in the buffer.
6. Call **KFreeFrame** to return the frame to the pool of available frames, unless you are starting another sequence that uses the same frame.

Interrupt, Channel-gain Array

Use this calling sequence if performing an interrupt transfer, using a channel-gain array. Before calling the functions in the sequence, define a buffer as an array of 4-byte elements.

1. Call **KGetADFrame** to get the handle to an A/D frame, unless you are starting another sequence that uses the same frame.
2. Define and assign values to a channel-gain array.
3. Call **KSetBufL** or **KSetBufR** to assign the address of the buffer previously declared to the buffer address element in the frame.
4. Call **KSetChnGAry** to assign the channel-gain array previously obtained to the channel-gain array address element in the frame.
5. Call **KIntStart** to start the operation. If you want the application program to perform additional functions during the data-acquisition periods, add the code after the **KIntStart** call.
6. Call **KIntStatus** to monitor the status of the operation. When completion is detected, data is available in the buffer.
7. Call **KFreeFrame** to return the frame to the pool of available frames, unless you are starting a another sequence that uses the same frame.

7.4 EXAMPLE PROGRAM

The following example is a copy of the file QBEXAMP2.BAS, in your Distribution Software. This program demonstrates an interrupt-mode transfer using channel/gain array.

```
'***************************************************************************
'
' For this example ONLY;
'       the configuration file must specify FLOATING POINT.
'
'***************************************************************************

' Define ALL local variables required by the program here.  NOTE
' that you must avoid declaring and using QuickBASIC variables on the
' fly.

DIM BUFFA(20) AS SINGLE
DIM CHANGAINARRAY(50) AS INTEGER

DIM NumOfBoards AS INTEGER
DIM DERR AS INTEGER
DIM DEVHANDLE AS LONG
DIM ADHANDLE AS LONG
DIM STATUS AS INTEGER
DIM count AS LONG
DIM BoardNum AS INTEGER
DIM SAMPLES AS INTEGER
DIM CJC AS SINGLE

CLS

'---------------------------------------------------------------------------
' This step initializes the internal data tables according to the
' information contained in the configuration file DASTC.CFG.

  PRINT "Initializing the board - - - PLEASE wait"

  A$ = "DASTC.CFG" + CHR$(0)
  DERR = DASTCDEVOPEN%(SSEGADD(A$), NumOfBoards)
  IF DERR <> 0 THEN BEEP: PRINT "ERROR "; HEX$(DERR); " OCCURED DURING '..DEVOPEN'": STOP

'---------------------------------------------------------------------------
' This step establishes communication with the driver through
' the Device Handle.

    BoardNum = 0
    DERR = DASTCGETDEVHANDLE%(BoardNum, DEVHANDLE)
    IF (DERR <> 0) THEN BEEP: PRINT "ERROR, DEVICE HANDLE IS null. . .": STOP

    DERR = DASTCGETCJC%(BoardNum, CJC)
    IF (DERR <> 0) THEN BEEP: PRINT "ERROR GETTING CJC TEMPERATURE. . .": STOP

    PRINT "CJC TEMPERATURE = "; CJC
    PRINT

'-------------------------------------------------------------------
```

```
' To perform any A/D operations, you must first get a Handle to an
' A/D Frame (Data tables inside the driver pertaining to A/D
' operations).

    DERR = KGetADFrame%(DEVHANDLE, ADHANDLE)
    IF (DERR <> 0) THEN BEEP: PRINT "ERROR, A/D FRAME HANDLE IS null. . .": STOP

'---------------------------------------------------------------------
' Program example code begins here

    SAMPLES = 20
    DERR = KSetBufR%(ADHANDLE, BUFFA(0), SAMPLES)
    IF DERR <> 0 THEN BEEP: PRINT "ERROR "; HEX$(DERR); " OCCURED DURING 'KSetBufR'": STOP

    CHANGAINARRAY(0) = 16
    CHANGAINARRAY(1) = 0: CHANGAINARRAY(2) = 0
    CHANGAINARRAY(3) = 2: CHANGAINARRAY(4) = 0
    CHANGAINARRAY(5) = 4: CHANGAINARRAY(6) = 0
    CHANGAINARRAY(7) = 6: CHANGAINARRAY(8) = 0
    CHANGAINARRAY(9) = 8: CHANGAINARRAY(10) = 0
    CHANGAINARRAY(11) = 10: CHANGAINARRAY(12) = 0
    CHANGAINARRAY(13) = 12: CHANGAINARRAY(14) = 0
    CHANGAINARRAY(15) = 14: CHANGAINARRAY(16) = 0
    CHANGAINARRAY(17) = 1: CHANGAINARRAY(18) = 0
    CHANGAINARRAY(19) = 3: CHANGAINARRAY(20) = 0
    CHANGAINARRAY(21) = 5: CHANGAINARRAY(22) = 0
    CHANGAINARRAY(23) = 7: CHANGAINARRAY(24) = 0
    CHANGAINARRAY(25) = 9: CHANGAINARRAY(26) = 0
    CHANGAINARRAY(27) = 11: CHANGAINARRAY(28) = 0
    CHANGAINARRAY(29) = 13: CHANGAINARRAY(30) = 0
    CHANGAINARRAY(31) = 15: CHANGAINARRAY(32) = 0

    DERR = KFormatChanGAry%(CHANGAINARRAY(0))
    IF DERR <> 0 THEN BEEP: PRINT "ERROR "; HEX$(DERR); " OCCURED DURING 'KFormatChnGAry'": STOP

    DERR = KSetChnGAry%(ADHANDLE, CHANGAINARRAY(0))
    IF DERR <> 0 THEN BEEP: PRINT "ERROR "; HEX$(DERR); " OCCURED DURING 'KSetChnGAry'": STOP

    ' un-comment this block of code for continuous run
    'PRINT "Continuous Run Selected."

    'DERR = KSETCONTRUN(ADHANDLE)
    'IF DERR <> 0 THEN BEEP: PRINT "ERROR "; HEX$(DERR); " OCCURED DURING 'KSetContRun'": STOP

    DERR = KINTStart%(ADHANDLE)
    IF DERR <> 0 THEN BEEP: PRINT "ERROR "; HEX$(DERR); " OCCURED DURING 'KINT'": STOP

    DO
    DERR = KINTStatus%(ADHANDLE, STATUS, count)

        IF DERR <> 0 THEN
                BEEP
                PRINT "ERROR "; HEX$(DERR); " OCCURED DURING 'KINTStatus'"
                DERR = KINTStop%(ADHANDLE, STATUS, count)
                IF DERR <> 0 THEN BEEP: PRINT "ERROR "; HEX$(DERR); " OCCURED DURING 'KINTStop'"
                STOP
        END IF
    LOCATE 16, 1: PRINT "Conversions completed= "; count

LOOP WHILE (((STATUS AND 1) = 1) AND INKEY$ = "")

' When using RECYCLE, this is the ONLY way to stop and reset the interrupt
'  controller. FAILURE to execute this call may cause your system to CRASH!
DERR = KINTStop(ADHANDLE, STATUS, count)
IF DERR <> 0 THEN BEEP: PRINT "ERROR "; HEX$(DERR); " OCCURED DURING 'KINTStop'": STOP

DERR = KFreeFrame%(ADHANDLE)
IF DERR <> 0 THEN BEEP: PRINT "ERROR "; HEX$(DERR); " OCCURED DURING 'KFreFrm'": STOP

DERR = KRestoreChanGAry%(CHANGAINARRAY(0))
IF DERR <> 0 THEN BEEP: PRINT "ERROR "; HEX$(DERR); " OCCURED DURING 'KRestoreChnGAry'": STOP

FOR I = 0 TO SAMPLES - 1
    PRINT "Sample No "; I + 1; (BUFFA(I))
NEXT I

END
```

7.5 LIST OF FUNCTION CALLS

The following list groups the function calls according to the categories of *Driver Initialization, Frame Management, Memory Management, Frame Setting, Frame Confirmation, Frame Operation, Single-call I/O,* and *Miscellaneous.*

Driver Initialization

DASTCDevOpen	Initialize a DAS driver using a configuration file or built-in defaults.
DASTCGetDevHandle	Get a handle to the logical device associated with the board. Verify that board configuration is according to specifications in DASTCDevOpen.
KDASDevInit	Initialize a DAS driver and all aspects of the board to a well-defined state.

Frame Management

KGetADFrame	Get a logical frame for Analog to Digital (A/D) operations.
KFreeFrame	Release a frame no longer in use to the driver.
KClearFrame	Initializes the specified frame to a pre-defined state.
KInitFrame	If frame operation is not in process, this function clears software driver to free state. Returns an error if Frame operation is in process.

Frame Setting

KClearContRun	Disable cyclic buffering.
KFormatChanGAry	Changes format of channel-gain-array data from integer to unsigned bytes.
KRestoreChanGAry	Changes format of channel-gain-array data from unsigned bytes to integer.
KSetBufL	Set the values of a frame's buffer-address and number-of-samples elements for user-defined long-integer arrays (Visual Basic for Windows only).
KSetBufR	Set the values of a frame's buffer-address and number-of-samples elements for user-defined floating-point arrays (Visual Basic for Windows only).
KSetChnGAry	Insert a reference to an A/D channel gain list into a frame.
KSetContRun	Enable cyclic buffering.
KSetStartStopChn	Set the start and stop channels for A/D input associated with a particular frame.

Frame Confirmation

KGetBuf	Obtain the data buffer address and size associated with a particular Frame.
KGetChnGAry	Obtain the channel/gain array associated with a particular Frame.
KGetContRun	Test for cyclic buffering enabled.
KGetStartStopChn	Get the start and stop channels for A/D input associated with a particular frame.

Frame Operation

KIntStart	Starts an interrupt transfer.
KIntStatus	Typically used to detect completion of a non-cyclic interrupt driven transfer or the current status of a cyclic interrupt-driven transfer.
KIntStop	Typically used to stop a cyclic interrupt-driven transfer.
KSyncStart	Perform a synchronous-input operation.

Single-call I/O

DASTCGETCJC	Returns the temperature (in ° C) used by the DAS-TC for CJC. You may use this temperature to correct the temperature reading from an unsupported thermocouple.
KADRead	Read a single A/D value.

Miscellaneous

KGetVer	Get driver specification and driver revision levels.

■ ■ ■

❒

❒

❒

Chapter 8

FUNCTION CALL DESCRIPTIONS

This chapter describes each function call and its requirements for use in programs you can write in QuickBASIC (Version 4.0 and 4.5), Professional BASIC (Version 7.0 and above), and Visual BASIC For DOS (Version 1.0). Descriptions are in the alphabetical order of the function call names. Each of the descriptions discusses the following subjects: *Purpose, Prototype, Parameters, Notes,* and an *Example.*

> *Note*: Many of the Function Calls in this chapter use *handles.* This variable may be declared as a long integer, as shown in the example programs of this chapter. A handle is a 4-byte, user-declared variable used strictly by the software driver.

♦ ♦ ♦

DASTCDevOpen

Purpose Initialize and configure the driver.

Prototype DASTCDevOpen (ByVal *cfgFile$*, *numOfBoards* As Integer) As Integer

Parameters

`cfgFile`	Driver configuration file
`numOfBoards`	Number of boards defined in cfgFile; 1 or 2

Notes DASTCDevOpen initializes the driver according to the information in `cfgFile`. On return, `numOfBoards` contains the number of boards for which `cfgFile` contains configuration information.

Specify -1 for `cfgFile` to make the driver search for DASTC.CFG, the configuration file, and as 0 to set the driver to use its built-in configuration defaults. The built-in configuration defaults are as follows:

	Board 0	Board 1
Base Address:	300h	308h
Interrupt Level:	7	5
Normal Mode Rejection Frequency:	60 Hz	60 Hz
Number Type:	Integer	Integer
Units:	C	C
Number of Readings to Average:	1	1
CJC Correction:	On	On

Example

```
DASErr% = DASTCDEVOPEN%(cfgFile$, NUMOFBOARDS%)
IF DASErr% <> 0 THEN BEEP: STOP:
```

♦ ♦ ♦

DASTCGETCJC

Purpose Returns the temperature (in ° C) used by the DAS-TC for CJC. You may use this temperature to correct the temperature reading from an unsupported thermocouple.

Prototype DASTCGETCJC (ByVal *brd* As Integer, *CJCtemp* As Single) As Integer

Parameters

`brd`	Board number; 0 or 1
`CJCtemp`	CJC sensor temperature value in degrees Celsius

Notes On return, `CJCtemp` contains the CJC sensor temperature (in ° C) for the board identified by `brd`. Note that the value returned in `CJCtemp` is floating point regardless of the format specified in the configuration file.

If you are acquiring data from a thermocouple type not supported by the DAS-TC, you must perform your own correction for the temperature at the STA-TC or STC-TC terminals. Use the temperature returned by DASTCGETCJC to look up the correction. If the temperature at the CJC sensor is not stable, you may need to make frequent calls for the temperature.

Note that a call to DASTCGETCJC during an interrupt transfer results in an error.

Example

```
DasErr% = DASTCGETCJC%(0, CJCtemp!)          'Board #0
IF (DasErr% <> 0) THEN BEEP: STOP:
```

♦ ♦ ♦

DASTCGetDevHandle

Purpose Obtain a board handle.

Prototype DASTCGetDevHandle (ByVal *boardNumber* As Integer, *boardHandle* As Long) As Integer

Parameters

`boardNumber`	Board number; 0 or 1
`boardHandle`	Board handle

Notes On return, `boardHandle` contains the handle associated with the board identified by `boardNumber`.

The value returned in `boardHandle` is intended for exclusive use as an argument to functions that require a board handle. Your program should not modify the value returned in `boardHandle`.

The driver supports up to two boards; you use DASTCGetDevHandle to assign a unique handle to each supported board (if you try to assign the same handle to a second board, you get an error).

DASTCGetDevHandle also performs the following tasks:

- Aborts all in-progress A/D operations.
- Checks for the presence of the board identified by boardHandle.
- Initializes board to internal defaults or to configuration file parameters (as set up by DASTCDevOpen).

Example

```
DasErr% = DASTCGETDEVHANDLE%(0, boardHandle)         'Board #0
IF (DasErr% <> 0) THEN BEEP: STOP:
```

♦ ♦ ♦

KADRead

Purpose Read a single A/D value.

Prototype KADRead (ByVal *boardHandle* As Long, ByVal *channel* As Integer, ByVal *gainCode* As Integer, *ADValue* As Long) As Integer

Parameters

boardHandle	Handle to acquisition board
channel	Input channel; 0, 1, ..., 15
gainCode	Set to 0 for this function, as the driver ignores this value and uses the value in the configuration.
ADValue	Storage location of acquired A/D value

Notes KADRead uses the board identified by *boardHandle* to perform a single A/D acquisition on *channel*. KADRead then stores the value obtained by the single acquisition in *ADValue*. Return values are in microvolts or centidegrees for integer types and not scaled for floating point.

Since return values for integers are in microvolts or 0.01 degrees, you may wish to convert to volts or degrees. Divide microvolts by 1,000,000 to get volts, and divide centidegrees by 100 to get degrees.

gainCode specifies the gain (as a gain code) to be applied to channel; however, the driver ignores this value and uses the value contained in the configuration file.

Note that a call to KADRead during an interrupt transfer results in an error.

Example

```
DIM ADVALUE AS LONG
DASErr% = KADRead%(boardHandle&, channel%, gainCode%, ADValue&)
IF DASErr% <> 0 THEN BEEP: STOP:
```

♦ ♦ ♦

KClearFrame

Purpose Set all of the elements of a frame to their initial values of zero.

Prototype KClearFrame (ByVal *frameHandle* As Long) As Integer

Parameters

`frameHandle`	Frame handle

Notes KClearFrame initializes the values of all of the elements in the frame identified by `frameHandle` to zero. Frame elements include Buffer Address, Start Channel, Stop Channel, and so on.

Example

```
DASErr% = KClearFrame%(frameHandle&)
IF DASErr% <> 0 THEN BEEP: STOP
```

♦ ♦ ♦

KClrContRun

Purpose Set the value of a frame's Buffering Mode element to single-cycle.

Prototype KClrContRun (ByVal *frameHandle* As Long)

Parameters

frameHandle	Frame handle

Notes KClrContRun sets the Buffering Mode to single-cycle in the frame identified by *frameHandle*.

Examples

```
DASErr = KClrContRun(frameHandle)
IF DASErr <> 0 THEN BEEP: STOP:
```

♦ ♦ ♦

KDASDevInit

Purpose Reset and initialize the board and driver.

Prototype KDASDevInit (ByVal *boardHandle* As Long) As Integer

Parameters

boardHandle	Board handle

Notes KDASDevInit performs the following tasks:

- Aborts all in-progress A/D operations.
- Initializes board to the parameter values in the internal defaults or the configuration file.
- Verifies that the board identified by *boardHandle* is present.

Example

```
DASErr% = KDASDevInit%(boardHandle&)
IF DASErr% <> 0 THEN BEEP: STOP:
```

♦ ♦ ♦

KFormatChanGAry

Purpose Reformats a user channel/gain array for use by internal objects.

Prototype KFormatChanGAry (Seg *array* As Integer) As Integer

Parameters *array* Channel Gain Array in the following form:

INTEGER OFFSET	DESCRIPTION OF VALUE	
0	# OF PAIRS (= N)	
1	CHANNEL	PAIR 1
2	GAIN	
3	CHANNEL	PAIR 2
4	GAIN	
⋮	⋮	⋮
2N−1	CHANNEL	PAIR N
2N	GAIN	

Notes You can specify different gains for different input channels using a channel/gain array. A reference to this array is passed to the driver using KSetChnGAry. For the driver to accept this array, it must be in specific format; this format is not achievable directly from BASIC.

This call operates on the array that is actually passed to your BASIC program, making it unreadable. To restore the array so that it is readable from BASIC, use the complementary function KRestoreChanGAry.

Example

```
DIM array(20) AS INTEGER
    :
array(0) = 2                                ' Number of Chan/Gain
pairs
array(1) = 0: array(2) = 0                  ' Chan 0 Gain 1
array(5) = 2: array(6) = 0                  ' Chan 2 Gain 1
DASErr% = KFormatChanGry%(array%)
IF DASErr% <> 0 THEN BEEP: STOP
    :
```

♦ ♦ ♦

KFreeFrame

Purpose	Free a frame and return it to the pool of available frames.
Prototype	KFreeFrame (ByVal *frameHandle* As Long) As Integer
Parameters	`frameHandle` Frame handle
Notes	KFreeFrame frees the frame identified by `frameHandle`, then returns that frame to the pool of available frames. The pool of available frames initially contains four frames for the A/D operation type.

Example

```
DASErr% = KFreeFrame%(frameHandle&)
IF DASErr% <> 0 THEN BEEP: STOP:
```

♦ ♦ ♦

KGetADFrame

Purpose Obtain the handle to an A/D frame.

Prototype KGetADFrame (ByVal *boardHandle* As Long, *frameHandle* As Long) As Integer

Parameters

boardHandle	Board handle
frameHandle	Handle to A/D frame

Notes On return, *frameHandle* contains the handle to an A/D frame associated with the board identified by *boardHandle*.

Example

```
DasErr% = KGetADFrame%(boardHandle&, frameHandle&)
IF (DasErr% <> 0) THEN BEEP: STOP:
```

♦ ♦ ♦

KGetBuf

Purpose	Get the values of a frame's Buffer Address and Number of Samples elements.
Prototype	KGetBuf (ByVal *frameHandle* As Long, *bufAddr* As Long, *samples* As Long) As Integer

Parameters

`frameHandle`	Frame handle
`bufAddr`	Buffer address
`samples`	Number of samples

Notes On return, *`bufAddr`* contains the Buffer Address and *`samples`* contains the Number of Samples in the frame identified by *`frameHandle`*.

Example

```
DASErr% = KGetBuf%(frameHandle&, bufAddr&, samples&)
IF (DASErr% <> 0) THEN BEEP: STOP:
```

♦ ♦ ♦

KGetChnGAry

Purpose	Get the value of a frame's Channel-gain Array address element.	
Prototype	KGetChnGAry (ByVal *frameHandle* As Long, *chanGainArray* As Long) As Integer	
Parameters	`frameHandle`	Frame handle
	`chanGainArray`	Channel-gain Array address

Notes

On return, `chanGainArray` contains the Channel-gain Array address in the frame identified by `frameHandle`.

Refer to KSetChnGAry for a description of Channel-gain Arrays.

Example

```
DasErr% = KGetChnGAry%(frameHandle&, chanGainArray&)
IF (DasErr% <> 0) THEN BEEP: STOP:
```

♦ ♦ ♦

KGetContRun

Purpose Get the value of a frame's Buffering Mode element.

Prototype KGetContRun (ByVal *frameHandle* As Long, *mode* As Integer) As Integer

Parameters

`frameHandle`	Handle to A/D frame
`mode`	Code that indicates Buffering Mode; 0 = Single-cycle, 1 = Continuous

Notes On return, `mode` contains a code that indicates the Buffering Mode in the frame identified by `frameHandle`.

Example

```
DasErr% = KGetContRun%(frameHandle&, mode%)
IF (DasErr% <> 0) THEN BEEP: STOP:
```

♦ ♦ ♦

KGetStartStopChn

Purpose Get the values of a frame's Start Channel and Stop Channel elements.

Prototype KGetStartStopChn (ByVal *frameHandle* As Long, *start* As Integer, *stop* As Integer) As Integer

Parameters

`frameHandle`	Frame handle
`start`	Start Channel; 0, 1, ... 15
`stop`	Stop Channel; 0, 1, ... 15

Notes On return, `start` and `stop` contain the values of the Start Channel and Stop Channel, respectively, in the frame identified by `frameHandle`.

Examples

```
DASErr = KGetStartStopChn%(frameHandle&, start%, stop%)
IF DASErr <> 0 THEN BEEP: STOP
PRINT "The Channel Scan is "; Start; " to "; Stop:
```

♦ ♦ ♦

KGetVer

Purpose Determine the driver revision and driver specification.

Prototype KGetVer (ByVal *boardHandle* As Long, *spec* As Integer, *version* As Integer) As Integer

Parameters

boardHandle	Board handle
spec	Driver specification
version	Driver version

Notes On return, *spec* contains the revision number of the Keithley MetraByte DAS Driver Specification to which the driver conforms; *version* contains the version number of the driver itself.

spec and *version* are 2-byte integers; the high byte contains the major number (the number left of the decimal point), the low byte contains the minor number (the number right of the decimal point). For example, if *spec* returns a high byte of 2 and a low byte of 3, the revision level is 2.03.

Use the following equations to extract the major and minor numbers from the values returned in *spec* and *version*:

major number = (*returned value*) / 256

minor number = (*returned value*) MOD 256

Example

```
DASErr = KGetVer%(boardHandle&, spec%, version%)
IF DASErr <> 0 THEN BEEP: STOP:
```

♦ ♦ ♦

KInitFrame

Purpose Initialize the Function Call Driver for A/D operations when the driver is not doing an Interrupt-mode operation.

Prototype KInitFrame (ByVal *ADHandle* As Long)

Parameters

ADHandle	A/D frame handle

Notes If an Interrupt-mode A/D operation is not in process, KInitFrame checks the validity of the frame identified by ADHandle then enables an A/D operation. If an Interrupt-mode operation A/D operation is in process, KInitFrame returns an error indicating the board is busy.

Note that a call to KInitFrame during an interrupt transfer results in an error.

Example

```
DASErr% = KInitFrame%(ADHandle&)
IF DASErr% <> 0 THEN BEEP: STOP:
```

◆ ◆ ◆

KIntStart

Purpose Start an Interrupt-mode A/D operation.

Prototype KIntStart (ByVal *frameHandle* As Long) As Integer

Parameters *frameHandle*	Frame handle

Notes KIntStart starts the Interrupt-mode A/D operation defined in the frame identified by *frameHandle*. Acquired samples are stored at a location specified by the Buffer Address element of the frame identified by *frameHandle*.

Returned values are in microvolts or centidegrees for integer types and are not scaled for floating point. Divide microvolts by 1,000,000 to obtain volts; divide centidegrees by 100 to obtain degrees.

Note that a call to KIntStart during an interrupt transfer results in an error.

Example

```
DASErr = KIntStart%(frameHandle&)
IF DASErr <> 0 THEN BEEP: STOP:
```

♦ ♦ ♦

KIntStatus

Purpose Determine the status of an Interrupt-mode operation.

Prototype KIntStatus (ByVal *frameHandle* As Long, *status* As Integer, *index* As Long) As Integer

Parameters

frameHandle	Frame handle
status	Code that indicates the status of an Interrupt operation; 0 = interrupt operation idle 1 = interrupt operation active
index	Number of the array element to be written to next

Notes On return, *status* contains a code that indicates the status of the interrupt operation defined by the frame identified by *frameHandle*; *index* contains the number of the array element to receive the next block of samples.

In the Continuous Buffer mode, *index* resets to 0 when the last block transfer is complete. Therefore, in Continuous Buffer mode, an *index* of 0 indicates either the end of a transfer or the beginning one.

Example

```
DASErr = KIntStatus%(frameHandle&, status%, count&)
IF DASErr <> 0 THEN BEEP: STOP:
```

♦ ♦ ♦

KIntStop

Purpose Abort an Interrupt-mode A/D operation.

Prototype KIntStop (ByVal *frameHandle* As Long, *status* As Integer, *count* As Long) As Integer

Parameters

`frameHandle`	Handle to A/D frame
`status`	Code that indicates status of interrupt operation; 0 = interrupt operation idle 1 = interrupt operation active
`count`	Number of samples already transferred to/from interrupt buffer

Notes KIntStop aborts the interrupt operation defined by the frame identified by `frameHandle`. On return, `status` contains a code that indicates the status of the operation (at the time the function was called) and `count` contains the number of samples already transferred to/from the interrupt buffer at the time the function was called.

KIntStop does nothing if an Interrupt-mode A/D operation is not in progress. If *status* = 1, the interrupt was running and was stopped.

Example

```
DASErr = KIntStop%(frameHandle&, status%, count&)
IF DASErr <> 0 THEN BEEP: STOP:
```

♦ ♦ ♦

KRestoreChanGAry

Purpose	Restore a user array that was put in driver format by KFormatChanGAry for re-use by BASIC.
Prototype	KRestoreChanGAry (Seg *chanGain(0)* as Integer) As Integer
Parameters	*chanGain(0)* array of Channel/Gain pairs to be restored.

Notes

This function restores the user's channel/gain array previously modified by KFormatChanGAry such that it is readable again from BASIC.

Do not call this function until a KSyncStart or an interrupt has been completed.

Example

```
DIM ChanGain(20) AS INTEGER
     :
DASErr% = KRestoreChanGAry%(chanGain(0)%)
IF DASErr% <> 0 THEN BEEP: STOP
     :
```

♦ ♦ ♦

KSetBufL

Purpose Set the values of a frame's buffer-address and number-of-samples elements for user-defined long-integer arrays (Visual Basic for Windows only).

Prototype KSetBufL (ByVal *frameHandle* As Long, Seg *bufAddr* As Long, ByVal *samples* As Long) As Integer

Parameters

frameHandle	Frame handle
bufAddr	Address of user-created buffer
samples	Number of samples to be stored in buffer; 1 to 65,535

Notes KSetBufL sets the Buffer Address to *bufAddr* and the Number of Samples to *samples* in the frame identified by *frameHandle*. The maximum number of *samples* is 65.535.

Example

```
DASErr% = KSetBufL%(frameHandle&, bufAddr&, samples&)
IF DASErr% <> 0 THEN BEEP: STOP
    :
```

♦ ♦ ♦

KSetBufR

Purpose Set the values of a frame's buffer-address and number-of-samples elements for user-defined floating-point arrays (Visual Basic for Windows only).

Prototype KSetBufR (ByVal *frameHandle* As Long, Seg *bufAddr* As Single, ByVal *samples* As Long) As Integer

Parameters

`frameHandle`	Frame handle
`bufAddr`	Address of user-created buffer
`samples`	Number of samples to be stored in buffer; 1 to 65,535

Notes KSetBufR sets the buffer address to `bufAddr` and the number of samples to `samples` in the frame identified by `frameHandle`. The maximum number of `samples` is 65,535.

Example

```
DASErr% = KSetBufR%(frameHandle&, bufAddr!, samples&)
IF DASErr% <> 0 THEN BEEP: STOP
     :
```

♦ ♦ ♦

KSetChnGAry

Purpose Set the value of a frame's Channel-gain Array address element.

Prototype KSetChnGAry (ByVal *frameHandle* As Long, *chanGainArray* As Integer) As Integer

Parameters

`frameHandle`	Frame handle
`chanGainArray`	Channel-gain Array address

Notes KSetChnGAry sets the Channel-gain array address in the frame identified by `frameHandle`.

Channel-gain Array defines two characteristics of an A/D operation: the sequence in which the input channels are sampled and the gain applied to each channel in that sequence (if the channel is configured for voltage). The gains for thermocouple channels are taken from the configuration at load time. The following table illustrates the required format of a channel-gain array.

Byte	0	1	2	3	4	5	...	2N	2N + 1
Value	N		Chan	Gain	Chan	Gain	...	Chan	Gain
	# of pairs		Pair 1		Pair 2		...	Pair N	

The gain must be specified as a gain code; refer to the table.

Code	Gain
0	1
1	125
2	166.67
3	400

When a reference to the Channel-gain Array is passed to the driver, any changes made while an interrupt is in process have an immediate effect.

Example

```
DASErr = KSetChnGAry%(frameHandle&, chanGainArray%)
IF DASErr <> 0 THEN BEEP: STOP:
```

♦ ♦ ♦

KSetContRun

Purpose	Set the value of a frame's Buffering Mode element to Continuous.
Prototype	KSetContRun (ByVal *frameHandle* As Long) As Integer
Parameters	`frameHandle` Frame handle
Notes	KSetContRun sets the Buffering Mode to Continuous in the frame identified by `frameHandle`.

Example

```
DASErr = KSetContRun%(frameHandle&)
IF DASErr <> 0 THEN BEEP: STOP:
```

♦ ♦ ♦

KSetStartStopChn

Purpose	Set the values of a frame's Start Channel and Stop Channel elements.
Prototype	KSetStartStopChn (ByVal *frameHandle* As Long, ByVal *start* As Integer, ByVal *stop* As Integer) As Integer

Parameters

`frameHandle`	Handle to A/D frame
`start`	Start Channel; 0, 1, ... 15
`stop`	Stop Channel; 0, 1, ... 15

Notes

KSetStartStopChn sets Start Channel to `start`, Stop Channel to `stop` in the frame identified by `frameHandle`.

During a Start/Stop sequence, the board's gains are either the internal defaults or those read from the configuration file at load time.

Use KSetChnGAry to specify a non-ordinal channel-scanning sequence.

If Stop > Start,

Sequence = Start, -, -, -, -, -, Stop

Else If Stop < Start,

Sequence = Start, -, -, -, 15, 0, -, -, -, Stop

Examples

```
DASErr = KSetStartStopChn%(frameHandle&, 0, 3)
IF DASErr <> 0 THEN BEEP: STOP:
```

♦ ♦ ♦

KSyncStart

Purpose	Start a synchronous operation.
Prototype	KSyncStart (ByVal *frameHandle* As Long) As Integer
Parameters	`frameHandle` Frame handle

Notes

KSyncStart starts the Synchronous operation defined in the frame identified by `framehandle`. When the configuration file specifies the Number Type as *Integer*, the returned value is in microvolts or centidegrees. When the Number Type is *Floating Point*, the returned value is in volts or degrees. Divide microvolts by 1,000,000 to obtain volts; divide centidegrees by 100 to obtain degrees.

Refer to Chapter 7 for a description of Synchronous operations.

Note that a call to KSyncStart during an interrupt transfer results in an error.

Example

```
DASErr = KSync%(frameHandle&)
IF DASErr <> 0 THEN BEEP: STOP:
```

■ ■ ■

❐

❐

❐

Appendix A

I/O CONNECTOR PIN ASSIGNMENTS

Figure A-1 shows the signals assigned to the pins of the DAS-TC I/O connector.

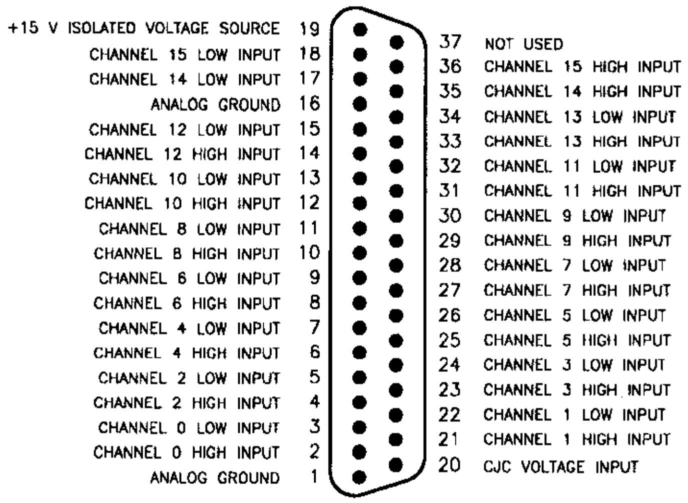


Figure A-1. DAS-TC I/O Connector Pin Assignments

■ ■ ■

❑

❑

❑

Appendix B

ERROR SUMMARY

B.1 ERROR CODES

The following list contains the full range of DAS-TC error codes and their descriptions.

0h	**No error**
6000h	**Error in configuration file**
Cause	The configuration file supplied to DASTCDevOpen is corrupt or does not exist. If the file is known to be good, it probably contains one or more undefined keywords.
Solution	Make sure the file exists at the specified path. Use the DASTCCFG.EXE utility do it to check for illegal keywords in file.
6001h	**Illegal base address in configuration file**
6002h	**Illegal IRQ level in configuration**
6004h	**Error opening configuration file**
6005h	**Illegal channel number**
Cause	The specified I/O operation channel is out of range. The legal range of channels is 0 to 15.
Solution	Specify legal channel number.
6006h	**Illegal gain**
6008h	**Bad number in configuration file**
Cause	An illegal specification of a number is detected in the configuration file. Note that if specifying a hexadecimal number for the base address, you must prefix the number with *&H.*
Solution	Check the number following *Address* in the configuration file.
6009h	**Incorrect version number**

600Ah	**Configuration file not found**
Cause	This error is returned by DASTCDevOpen whenever the specified configuration file is not found.
Solution	Check the configuration file name (spelling), path, and so on.
600Ch	**Error in returning interrupt buffer**
600Dh	**Bad frame handle**
Cause	This error is usually returned by a Frame Management or an Operation Function Call whenever an illegal frame handle is passed to one of these functions.
Solution	Check the frame handle.
600Eh	**No more frame handles**
600Fh	**Requested interrupt buffer too large**
6010h	**Cannot allocate interrupt buffer**
6011h	**Interrupt buffer already allocated**
6012h	**Interrupt de-allocation error**
6013h	**Interrupt buffer never allocated**
7000h	**No board name**
Cause	DASTCDevOpen did not find a board *name* in the specified configuration file.
Solution	Make sure that a name is specified in your configuration file. The legal name is DASTC.
7001h	**Bad board name**
Cause	DASTCDevOpen found the board *name* in the specified configuration file to be illegal. The legal name is DASTC.
Solution	Check the keyword following *name* in your configuration file.
7002h	**Bad board number**
Cause	DASTCDevOpen found the *board* number in the specified configuration file to be illegal. Legal board numbers are 0 and 1.
Solution	Check the number following *board* in your configuration file.

7003h	**Bad base address**
Cause	DASTCDevOpen found the board's base address in the specified configuration file to be illegal. Legal addresses are 100h through 3FFh in increments of 8h, inclusive.
Solution	Check the number following address in your configuration file. NOTE that to specify a Hex number, the address must be preceded by *&H* .
7004h	**Bad interrupt level**
Cause	DASTCDevOpen found the interrupt level in the specified configuration file to be illegal. Legal interrupt levels are 2, 3, 4, 5, and 7.
Solution	Check the number following *IntLevel* in your configuration file.
7005h	**Bad Normal Mode Rejection frequency**
7006h	**Bad number type**
7007h	**Bad channel configuration**
Cause	Channel number is out of range.
Solution	Check the channel number; it should be 0 to 15.
7008h	**Checksum error**
Cause	Checksum test in the communications packet failed.
Solution	Retry the KDASDevInit function. If the problem persists, call for technical support.
7009h	**Board not initialized**
Cause	A function was called before the KDASDevInit call was made. The PC-side board diagnostics performed during KDASDevInit failed. The return of the DAS-TC ID failed. Incorrect base address.
Solution	Check the base address and all other boards to be sure there is no conflict. Call for technical support if you cannot resolve the problem.
700Ah	**Protocol communication error**
8000h	**No error**
8001h	**Function not supported**
Cause	A request is made to a function not supported by the DAS-TC driver. This error should not occur in standard release software.
Solution	Contact the Keithley MetraByte applications engineers.
8002h	**Function out of bounds**
Cause	Illegal function number is specified. This error should not occur in standard release software.
Solution	Contact the Keithley MetraByte applications engineers.

8003h	**Nonvalid board number**
Cause	The DAS-TC driver supports up to two boards: Board 0 and Board 1.
Solution	Check the board number parameter in your call to DASTCGetDevHandle.

8005h	**Board not found at configured address**
Cause	This error is issued during KDASDevInit whenever the board presence test fails. The error is normally caused by a conflict between the specified board I/O address and the actual I/O address the board is configured for. Also, this error is issued when the board is not present in the system.
Solution	Check the board's base I/O address DIP switch and make sure it matches the base address in your configuration file.

8006h	**A/D not initialized**
Cause	A function was called before KDASDevInit was complete.
Solution	Retry KDASDevInit.

801Ah	**Interrupts already active**
Cause	An attempt is made to start an interrupt-based operation while another is already active.
Solution	Stop current interrupt mode first and retry.

B.2 INPUT ERROR CONDITIONS

When a voltage or thermocouple input is under or over the range setting of a particular channel, the DAS-TC returns a specific value for each condition. These values are listed in the following tables.

Values Returned for Voltage Inputs Outside the Range Setting

Condition	Value Returned for Floating Point	Value Returned for Integer
Voltage input under the voltage range setting:	-10,000.00 V	-971,227,136 mV
Voltage input over the voltage range setting:	+10,000.00 V	+1,176,256,512 mV

Values Returned for Thermocouple Inputs Outside the Range Setting

Condition	Value Returned for Floating Point	Value Returned for Integer
Thermocouple input under the temperature range setting:	-10,000.00 °C or °F	-971,227,136 °C or °F
Thermocouple input over the temperature range setting:	+10,000.00 °C or °F	+1,176,256,512 °C or °F

Appendix C

THE CJC SENSOR

The CJC sensor circuit, used on the STA-TC and STC-TC accessory panels, is an Analog Devices AD592CN in series with a 20 kΩ, 0.01% resistor. DAS-TC I/O connections for this circuit are shown in Figure C-1.

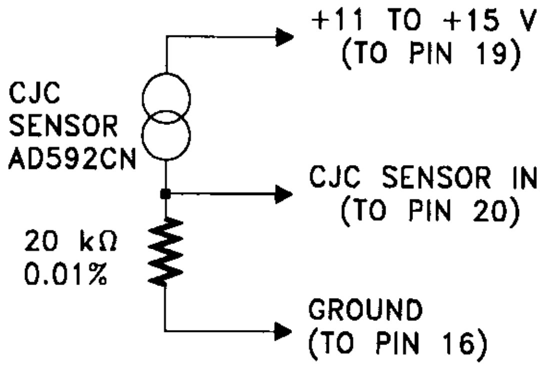


Figure C-1. DAS-TC I/O Connections for the CJC Sensor

■ ■ ■

❐

❐

❐

Appendix D

THE DAS-TC EXTERNAL DRIVER

The DAS-TC External Driver is a terminate-and-stay-resisident (TSR) program that supports the use of a DAS-TC board with the following Keithley MetraByte data acquisition and analysis software packages:

- VIEWDAC
- EASYEST LX
- ASYST

The DAS-TC External Driver supports a 32-bit mode for VIEWDAC and a 16-bit mode for EASYEST LX and ASYST. The 32-bit mode allows VIEWDAC to resolve higher-resolution data.

The driver also supports analog inputs on up to 16 differential, 16-bit channels. Other options supported by the driver include interrupt and synchronous mode A/D operations.

This appendix describes the DAS-TC External Driver and its use with DAS-TC boards.

D.1 DRIVER FILES

The DAS-TC External Driver software package includes the following files:

- **DASTC.EXE** - The DAS-TC External Driver program
- **DASTCCFG.EXE** - The configuration utility for creating or modifying configuration files
- **DASTC.CFG** - The default configuration file
- **VWDAC.SEQ** - A VIEWDAC example of a synchronous or interrupt analog input
- **FILES.DOC** - A list and description of the files in the software package
- **README.DOC** - A history of revisions along with last-minute information

The external driver requires a specially adapted configuration file, either the default configuration file (DASTC.CFG, as supplied in the DAS-TC External Driver package) or a file created with the configuration utility. Refer to the next section for information on this file.

D.2 DAS-TC CONFIGURATION FOR THE EXTERNAL DRIVER

To operate a DAS-TC board with the DAS-TC External Driver, you must set up the board's configuration file for use with the external driver and for either the 16- or 32-bit mode of external-driver operation. You perform this setup using the DOS configuration utility, DASTCCFG.EXE. The steps for starting the configuration utility and setting up the configuration file specifically for the external driver are as follows:

1. Start the configuration utilty by changing to the directory containing the DASTCCFG.EXE program and enter the following at the DOS prompt:

   ```
   DASTCCFG
   ```

   The configuration utility starts by displaying a help screen.
2. Press **[Enter]**.

   The configuration utility asks,

   Is this a configuration file for the DAS-TC External Driver?
3. Select *Yes*.

   The configuration utility asks,

   Is this a configuration file for VIEWDAC?
4. Select *Yes* if you are using VIEWDAC. The configuration utility sets the configuration file to initialize the DAS-TC External Driver in its 32-bit mode. The utility then displays the path-specification screen, shown in Figure 2-2.

   Select *No* if you are using EASYEST LX or ASYST. The configuration utility sets the configuration file to initialize the DAS-TC External Driver in its 16-bit mode. The utility then displays the path-specification screen, shown in Figure 2-2.
5. Specify the path as instructed in the section *Path-specification Screen*, in Chapter 2.

   After you specify and indicate acceptance of the path specification, the program displays the First Board Main Menu.
6. Proceed with the First Board Main Menu and the remainder of the program as instructed in Chapter 2.

The screens for the remainder of the configuration utility program are the same as those shown in Chapter 2 except for the following two differences:

- The Main Menu for the external-driver version does not offer the FLOATING POINT option under Number Type because values are returned in integer format only.
- The Thermocouple Configuration Table for the external-driver version does not offer the VOLTS option under Type because values are returned in degrees only.

D.3 EXTERNAL DRIVER STARTUP FOR ONE OR TWO BOARDS

As described earlier, the DAS-TC External Driver is a TSR. Before you can use a TSR, you must load it into computer memory. Once loaded, the TSR resides in memory until you turn off or re-boot your computer.

To load the external driver into computer memory, change to the directory containing DASTC.EXE and enter the following at the DOS prompt:

```
DASTC  configuration_filename
```

where *configuration_filename* is the name (and path) of the configuration file you intend to use. If you do not name a configuration file, the driver searches for the default configuration file (DASTC.CFG) in the directory you are working from.

> **Note:** By adding the above startup entry to your AUTOEXEC.BAT file, you can instruct your computer to load the driver whenever it boots.

D.4 EXTERNAL DRIVER STARTUP FOR MORE THAN TWO BOARDS

The number of available accessory slots in your computer determines the maximum number of DAS-TC boards your computer can accept. The number of DAS-TC boards a DAS-TC External Driver can support is limited to two. You can, however, load more than one DAS-TC External Driver by using a different configuration file for each loading. The number of times you can load a DAS-TC External Driver is limited only by the amount of available memory in your computer.

For example, to use three DAS-TC boards, you would use the configuration utility to create two configuration files. For the sake of this explanation, you could name the new configuration files *DASTC-1.CFG* and *DASTC-2.CFG*. The DASTC-1.CFG file could contain configuration settings for two DAS-TC boards at addresses 300h and 310h, while the DASTC-2.CFG file could contain configuration settings for one board at address 340h. You would then load these drivers with the following entries:

```
DASTC  DASTC-1.CFG [Enter]
DASTC  DASTC-2.CFG [Enter]
```

D.5 EXTERNAL DRIVER ACCESS

VIEWDAC and EASYEST LX application programs access the loaded DAS-TC External Driver automatically if it is the only installed external driver. If you are using VIEWDAC with more than one external driver, you must select the external driver with *DAS Device* (board name) from within a VIEWDAC DAS Task. If you are using EASYEST LX with more than one external driver, you must perform the following steps:

1. Select *Devices* from the EASYEST LX menu bar.
2. Select option 1, *DAS Board Selection*, from the *DAS Configuration* screen.
3. Select the appropriate external driver (board name) from the displayed list.

ASYST can access the DAS-TC External Driver only after you perform the following steps:

1. After you load the DAS-TC External Driver into computer memory, start up ASYST version 2.10 or greater and permanently load the Data Acq Master and the Ext DAS Driver Support system overlays from the Data Acquisition Menu. ASYST then automatically searches for a driver. Upon finding the DAS-TC External Driver, ASYST creates a virtual device called DASTC.
2. To make DASTC the current device, enter the following at the OK prompt:

```
DASTC
```

D.6 SPECIAL CHARACTERISTICS

This section identifies and describes special operating charactersitics of the DAS-TC External Driver, as follows:

- **Synchronous A/D** - The synchronous A/D mode supports internal clocking and internal triggering.
- **Interrupt-driven Operations** - The DAS-TC board supports interrupt acquisitions for conversions. Only one such operation may be active at a time.

  The only error that can occur during an interrupt acquisition is a protocol-handshake failure. You can connect your application's critical-error handler to deal with this situation. This error is also reported by the interrupt status call.
- **Interrupt A/D** - The interrupt A/D mode supports single, non-cyclic buffering with internal clocking and triggering. Data is placed in the application buffer once per scan.
- **Internal Clocking** - A conversion begins without waiting for an external-clock signal and proceeds at a rate determined by the setting for the Normal Mode Rejection Frequency, in the configuration file. The conversion delay, which is the time between conversions, is equal to the period of two cycles of the Normal Mode Rejection Frequency.
- **Software Interrupt Vectors** - The DAS-TC External Driver uses two software-interrupt vectors for communicating between itself and the application program. These vectors are in the "user interrupt" range of 60h to 67h, set aside by MS-DOS. Do not confuse these (software) interrupt vectors with the hardware interrupt lines used by the DAS-TC.

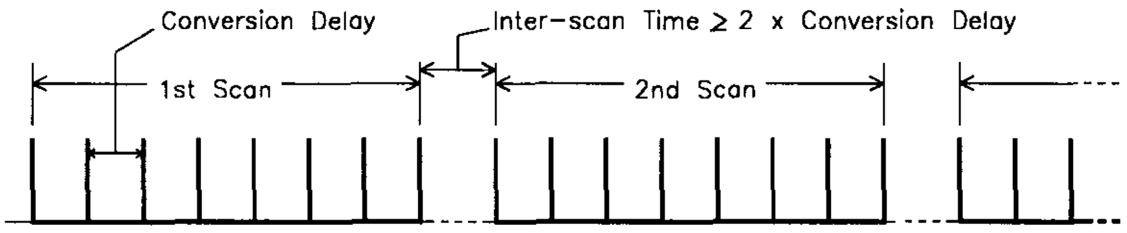
  To prevent conflicts with other devices or programs, you can change one or both of the default interrupt vector numbers (65h and 66h) to any number between 60h and 67h, using the SET command from DOS. The SET command saves a string in the DOS environment that the DAS-TC External Driver searches for during loading. Use the information in the following table to set the environmental string.

Interrupt	**Default #**	**Environmental String**
Device linking vector	66h	DAS DS=*xx*
Critical error vector	65h	DAS CE=*xx*

  xx is between 60h and 67h

- **Scan Size** - Scan size is the number of channels in a start/stop sequence. You can set the scan size to be any number of channels from 1 to 16.
- **Scan Timing** - During a scan, the DAS-TC control software takes a reading of each of the channels in the start/stop sequence. Timing for these readings (or *conversions*) is determined by the Normal Mode Rejection Frequency, which is controlled by a setting in the configuration file. The period of a conversion is the *conversion delay* and is equal to the period of two cycles of the Normal Mode Rejection Frequency.

  Normally, the period between multiple scans is twice the conversion delay for a single conversion. If you wish to control the period between multiple scans, you should limit the number of conversions to the scan size. This limit lets the application program control the period between multiple scans. The timing for scans is illustrated in the following diagram, in which the scan size is eight channels.



Scan Timing for an 8-channel Sequence

- **Gain** - The DAS-TC External Driver ignores the gain parameter in an application program.
- **Temperature Measurements with VIEWDAC** - When you are controlling temperature measurements with VIEWDAC, the A/D Task Panel is unable to show temperature range and indicates an overflow condition (a series of asterisks), instead.
- **Number Types** - When you make your settings with the configuration utility, your choice of number type is limited to Integer. The DAS-TC External Driver does not pass floating-point numbers. The format for integers is twos complement, to represent a signed number.
- **Parameters from the Thermocouple Configuration Table** - The DAS-TC External Driver uses parameters from the Thermocouple Configuration Table, in the configuration utility, as follows:
  - **TYPE** - The VOLTS selection is not available for the DAS-TC External Driver because values are returned as temperature..
  - **GAIN** - The GAIN selection is not available for the DAS-TC External Driver because values are returned as temperature.
  - **C/F** - This selection determines whether temperature readings are in Celsius or Fahrenheit.
  - **AVG** - This entry determines the number of readings to be averaged in the range of 0 to 100. The program uses the following codes:

    0 = no averaging
    1 = no averaging
    2 ... 100 = actual number of averaged readings
  - CJC - This selection determines whether cold-junction compensation is to be used, as follows:

    ON turns CJC on
    OFF turns CJC off
- **Values Returned to Application** - the temperature values returned by the DAS-TC to your application are scaled according to the thermocouple type specified in your configuration file. To display the results, do the following:
  - *VIEWDAC* - Use a normal A/D task for your acquisition. Since the DAS-TC performs the thermocouple scaling, you have no need for the Viewdac Thermocouple Task.

- *EASYEST* - Make sure the Engineering Units are set to voltage. To verify this setting, select the Devices menu followed by the Engineering Units Conversion option. Since the DAS-TC performs the thermocouple scaling, you have no need to select the thermocouple type.
- *ASYST* - The temperature value returned to ASYST is scaled according to thermocouple type. However, you must adjust this value for temperature range. Use the A/D.SCALE command in ASYST to make this adjustment, as follows:

  ADValue -3276.8 3276.7 A/D.SCALE *Temperature* :=

  where *ADValue* is the value returned to ASYST by the DAS-TC, -3276.8 is the minimum returnable value (see Section D.8), 3276.7 is the maximum returnable temperature value (see Section D.8), and *Temperature* is the temperature reading in ° C or ° F.

D.7 ERROR CODES

The following list contains the full range of DAS-TC error codes and their descriptions. The error codes are in decimal.

0 **No error**

24576 **Error in configuration file**

Cause The configuration file supplied to DASTCDevOpen is corrupt or does not exist. If the file is known to be good, it probably contains one or more undefined keywords.

Solution Make sure the file exists at the specified path. Check for illegal keywords in the file; the best way to fix illegal keywords is to let the supplied DASTCCFG.EXE utility do it.

24577 **Illegal base address in configuration file**

24578 **Illegal IRQ level in configuration**

24580 **Error opening configuration file**

24581 **Illegal channel number**

Cause The specified I/O operation channel is out of range. The legal range of channels is 0 to 15.

Solution Specify legal channel number.

24582 **Illegal gain**

24584 **Bad number in configuration file**

Cause An illegal specification of a number is detected in the configuration file. Note that if specifying a hexadecimal number for the base address, you must prefix the number with *&H*.

Solution	Check the number following *Address* in the configuration file.
24585	**Incorrect version number**
24586	**Configuration file not found**
Cause	This error is returned by DASTCDevOpen whenever the specified configuration file is not found.
Solution	Check the configuration file name (spelling), path, and so on.
24588	**Error in returning interrupt buffer**
24589	**Bad frame handle**
Cause	This error is usually returned by a Frame Management or an Operation Function Call whenever an illegal frame handle is passed to one of these functions.
Solution	Check the frame handle.
24590	**No more frame handles**
24591	**Requested interrupt buffer too large**
24592	**Cannot allocate interrupt buffer**
24593	**Interrupt buffer already allocated**
24594	**Interrupt de-allocation error**
24595	**Interrupt buffer never allocated**
28672	**No board name**
Cause	DASTCDevOpen did not find a board *name* in the specified configuration file.
Solution	Make sure that a name is specified in your configuration file. The legal name is DASTC.
28673	**Bad board name**
Cause	DASTCDevOpen found the board *name* in the specified configuration file to be illegal. The legal name is DASTC.
Solution	Check the keyword following *name* in your configuration file.
28674	**Bad board number**
Cause	DASTCDevOpen found the *board* number in the specified configuration file to be illegal. Legal board numbers are 0 and 1.
Solution	Check the number following *board* in your configuration file.

28675 **Bad base address**

Cause DASTCDevOpen found the board's base address in the specified configuration file to be illegal. Legal addresses are 100h through 3FFh in increments of 8h, inclusive.

Solution Check the number following address in your configuration file. NOTE that to specify a Hex number, the address must be preceded by *&H*.

28676 **Bad interrupt level**

Cause DASTCDevOpen found the interrupt level in the specified configuration file to be illegal. Legal interrupt levels are 2, 3, 4, 5, and 7.

Solution Check the number following *IntLevel* in your configuration file.

28677 **Bad Normal Mode Rejection frequency**

28678 **Bad number type**

28679 **Bad channel configuration**

Cause Channel number is out of range.

Solution Check the channel number; it should be 0 to 15.

28680 **Checksum error**

Cause Checksum test in the communications packet failed.

Solution Retry the KDASDevInit function. If the problem persists, call for technical support.

28681 **Board not initialized**

Cause A function was called before the KDASDevInit call was made. The PC-side board diagnostics performed during KDASDevInit failed. The return of the DAS-TC ID failed. Incorrect base address.

Solution Check the base address and all other boards to be sure there is no conflict. Call for technical support if you cannot resolve the problem.

28682 **Protocol communication error**

32768 **No error**

32769 **Function not supported**

Cause A request is made to a function not supported by the DAS-TC driver. This error should not occur in standard release software.

Solution Contact the Keithley MetraByte applications engineers.

32770 **Function out of bounds**

Cause Illegal function number is specified. This error should not occur in standard release software.

Solution Contact the Keithley MetraByte applications engineers.

32771	**Nonvalid board number**
Cause	The DAS-TC driver supports up to two boards: Board 0 and Board 1.
Solution	Check the board number parameter in your call to DASTCGetDevHandle.

32773	**Board not found at configured address**
Cause	This error is issued during KDASDevInit whenever the board presence test fails. The error is normally caused by a conflict between the specified board I/O address and the actual I/O address the board is configured for. Also, this error is issued when the board is not present in the system.
Solution	Check the board's base I/O address DIP switch and make sure it matches the base address in your configuration file.

32774	**A/D not initialized**
Cause	A function was called before KDASDevInit was complete.
Solution	Retry KDASDevInit.

32794	**Interrupts already active**
Cause	An attempt is made to start an interrupt-based operation while another is already active.
Solution	Stop current interrupt mode first and retry.

D.8 INPUT ERROR CONDITIONS

When a thermocouple input is under or over the range setting of a particular channel, the DAS-TC returns a specific value for each condition and for each DAS-TC External Driver mode (16- or 32-bit). These values are listed in the following table.

Values Returned for Thermocouple Inputs Outside the Range Setting

Thermocouple Input	Value Returned for 16-bit Mode	Value Returned for 32-bit Mode
Under the temperature range setting:	-3,276.8 ° C or ° F	-167,772.16 ° C or ° F
Over the temperature range setting:	+3,276.7 ° C or ° F	+167,772.15 ° C or ° F

■ ■ ■

❐

❐

❐