DAS-16, DAS-16F, and DAS-16G High Speed Analog I/O Boards User's Manual

New Contact Information

Keithley Instruments, Inc.
28775 Aurora Road
Cleveland, OH 44139

Technical Support: 1-888-KEITHLEY
Monday- Friday 8:00 a.m. to 5:00 p.m. (EST)
Fax: (440) 248-6168

Visit our website at http://www.keithley.com

KEITHLEY

DAS-16, DAS-16F & DAS-16G

High Speed Analog I/O Boards

Part Number: 24840

Revision G: April 1993

WARNING

Keithley MetraByte assumes no liability for damages consequent to the use of this product. This product is not designed with components of a level of reliability suitable for use in life support or critical applications.

(This page intentionally left blank.)

TABLE OF CONTENTS

(This page intentionally left blank.)

LIST OF TABLES

LIST OF FIGURES

(This page intentionally left blank.)

CHAPTER 1 INTRODUCTION

1.1 GENERAL DESCRIPTION

The Model DAS-16, DAS-16F, and DAS-16G (hereinafter referred to as DAS-16) are multi-function, high-speed, programmable, A/D (and D/A) I/O expansion boards for the IBM Personal Computer. They are full length boards that install internally in an expansion slot of an IBM PC/XT/AT and compatibles to turn the computer into a fast, high-precision data acquisition and signal analysis instrument. DAS-16 boards are of multilayer construction with integral ground plane to minimize noise and crosstalk at high frequencies. The DAS-16G includes an additional register at an I/O address location for setting the gain. Block diagrams of the three boards are shown in Figures 1-1 and 1-2.

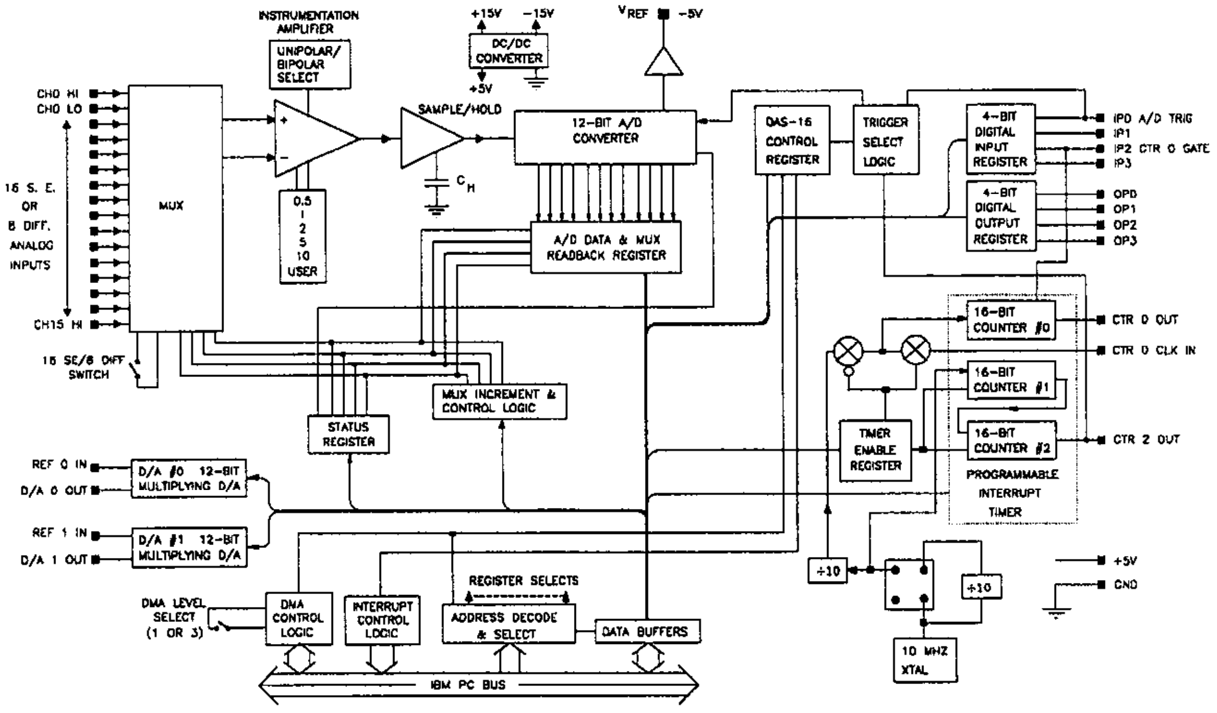


Figure 1-1. DAS-16 and DAS-16F Functional Block Diagram

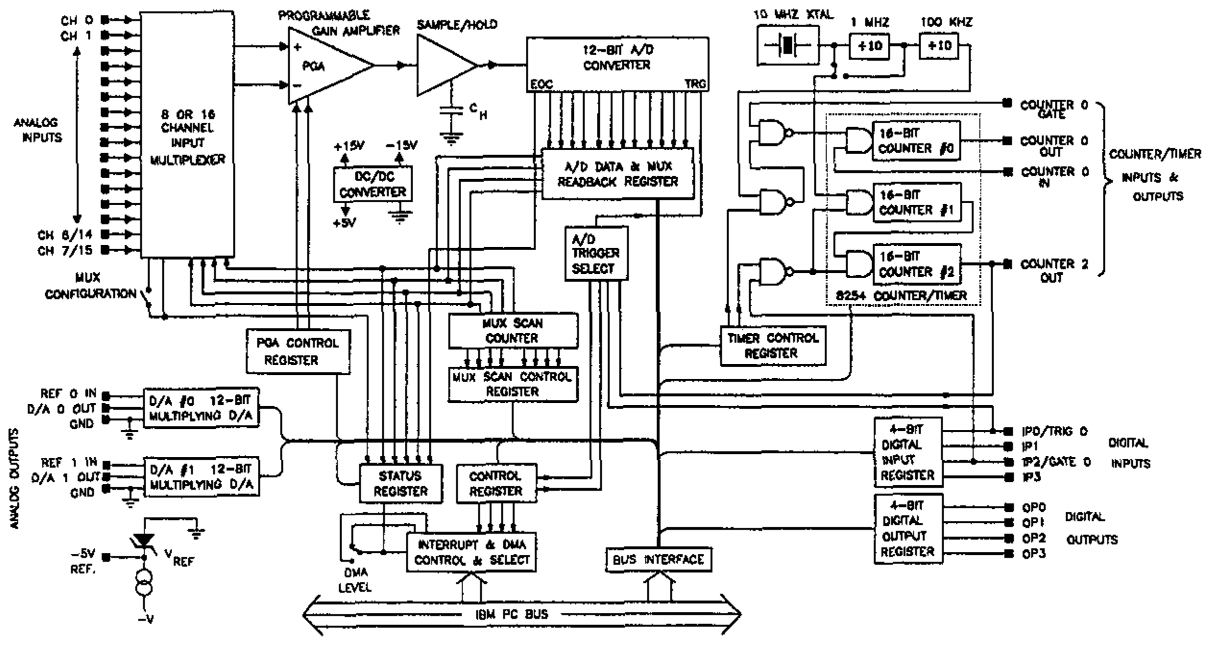


Figure 1-2. DAS-16G Functional Block Diagram

One feature of the DAS-16 is the availability of two channels of multiplying 12-bit D/A output. The DACs may use a fixed -5V reference available from on-board for a 0 to +5V output range. Alternatively, an external AC or DC reference may be used to give different output ranges or programmable attenuator action on an AC signal. D/A's are double-buffered to provide instantaneous single-step updates.

Digital I/O consists of four bits of TTL/DTL-compatible digital output and four bits of digital input. Apart from being addressed as individual I/O ports, some of the digital inputs do double duty in some modes as A/D trigger and counter gate control inputs.

Additional features include:

- Switch selectable Channel Input Configuration, High Impedance Ranges, Base I/O Address, and DMA Level.
- A/D conversions can be triggered by any of the following: software command, internal programmable-interval timer, or direct external trigger to the A/D. Once the A/D conversion has been completed, data transfers are accomplished via program transfer, interrupt, or DMA.
- A 3-channel programmable interval timer (Intel 8254) provides trigger pulses for the A/D at any rate from 250 KHz down to 8 pulses/hr. Two channels are operated in fixed divider configuration from an internal crystal clock. The third channel is uncommitted and provides a gated 16-bit binary counter that can be used for event or pulse counting, delayed triggering, and in conjunction with the other channels for frequency and period measurement.
- A -5V (± 0.05V) precision reference voltage output is derived from the ADC reference. Typical applications are providing a DC reference input for the DACs and providing offsets and bridge excitation to user-supplied input circuits.

1.2 PROGRAMMING OVERVIEW

The DAS-16 can be programmed using any of four options:

- Pop-Up Menu
- File I/O Commands
- Call Modes
- Low-level Direct Register I/O

The **Pop-Up Menu** is a small application program which allows you to directly control the operation of the DAS-16 without writing a single line of code. It allows you to configure the DAS-16 to perform an Analog or Digital operation and to store the resultant data in a disk file. It consists of three "panels" which are manipulated through the use of a Mouse or the PC's keyboard. The Pop-Up Menu provides a quick way to test your DAS-16 as well as being useful for debugging/monitoring your programs written using the File I/O Command Interface. The Pop-Up Menu is described in detail in Chapter 3.

Programmers familiar with using File I/O (Opening files and transferring data between the program and files) and desiring some degree of flexibility may prefer the **File I/O Command** Interface. Programming can be as simple as writing strings to a file opened for output or reading strings from a file opened for input. The File I/O Commands are "English-Language"-like, programming language-independent, commands. These commands perform most of the same tasks as the Mode Calls. In addition, this interface supports transparent memory allocation of large DMA buffers and allows you to save collected the data to a file in Binary or ASCII format. The File I/O Command Interface is described in Chapter 4.

The **Call Mode Interface** is designed to be used from Interpreted BASIC, Compiled BASIC or QUICKBASIC. The Mode calls are a collection of functions called MODEs which are accessed from your BASIC program through a common CALL statement. The Call Modes allow a great deal of flexibility and allow background data acquisition to take place. The Call Modes are described in Chapter 5. Optionally available is the PCF-16G package. This provides mode call support for programs written in Pascal, C, and FORTRAN.

The last method, **Low-Level Direct Register I/O**, should be used only if you have a high-level of programming expertise and understanding of your PC's architecture, the PC's Peripheral Controllers (the 8259 Interrupt Controller, the 8237 DMA Controller, etc.) and the DAS-16 hardware. This method can be used from most languages. However, because BASIC has no interrupt or DMA processing functions, background data acquisition tasks cannot be performed from BASIC programs. Register Maps and other details needed for Low-Level Programming are provided in Chapters 6 through 8.

1.3 ORDERING INFORMATION

Table 1-1 lists and describes part numbers for the DAS-16 boards and their accessories.

Table 1-1. Ordering Information

Part Number	Description
DAS-16	Includes the DAS-16, a 16-channel, high speed A/D interface with DMA,(70,000 samples/sec. max.) as well as software (on 5 1/4" disks) and appropriate documentation.
DAS-16F	Includes the DAS-16F, a 16-channel, high speed A/D interface with DMA,(100,000 samples/sec. max.) as well as software (on 5 1/4" disks) and appropriate documentation.
DAS-16G1	Includes the DAS-16G1, a 16-channel, high speed A/D interface with software selectable input gains (1, 10, 100, and 500), software (on 5 1/4" disks), and appropriate documentation.
DAS-16G2	Includes the DAS-16G2, a 16-channel, high speed A/D interface with software selectable input gains (1, 2, 4, and 8), software (on 5 1/4" disks), and appropriate documentation.
STA-16	Screw Terminal Adapter Board. Can be used with the DAS-16.
C-1800	Cable to connect DAS-16 to a STA-16.
PCF-16G	Pascal, C, and FORTRAN support for all models. This package includes a fully commented assembly language source listing.
STREAMER	Menu-driven Data Acquisition package which allows data to be written to a hard disk at up to 100 Khz.
LABTECH NOTEBOOK	Menu-driven data acquisition and analysis from Laboratory Technologies Corporation.

Table 1-1. Ordering Information

Part Number	Description
ASYST 1, 2, 3, 4	Programmable data acquisition and analysis .
UNKELSCOPE	Menu-driven data acquisition and analysis from Unkel Software Inc. (Copyright MIT).
SNAPSHOT STORAGE SCOPE	Menu-driven data acquisition and analysis from H.E.M. Data Corporation.
VIEWDAC	Data acquisition, analysis, and graphical display software.
EASYEST LX	Menu-driven data acquisition, analysis, and graphical display software.
TTOOLS	Utilities for the Turbo-PASCAL programmer by Quinn-Curtiss Software.

1.4 HOW TO USE THIS MANUAL

This manual provides you with the information necessary to install and program the DAS-16. It assumes you are familiar with your personal computer, DOS, and have some programming experience. In addition to this manual, you may also need:

- The documentation for your computer
- A language reference manual, if you intend to program custom applications for the DAS-16

The remainder of this manual is as follows:

Chapter 2, "Installation", provides hardware installation instructions. It describes the switches, jumpers, and connectors. It also describes how to install the DAS-16.

Chapter 3, "Introduction to the MDAS16 Driver", describes the MDAS-16 driver. It includes directions for installing the driver and describes the Pop-Up Menu.

Chapter 4, "File I/O Commands" provides a detailed description of each File I/O Command. It also describes how to invoke those commands from programs written in each of the supported languages: BASIC, QuickBASIC, Microsoft C, Turbo C, Microsoft PASCAL, and Turbo PASCAL.

Chapter 5, "Programming the DAS-16 via the Call Modes" describes how to use the Call Modes from programs written in BASIC and QuickBASIC. It also provides a detailed description of each call mode.

Chapter 6, "Low Level Programming" describes the registers on the DAS-16 and how they are used to program the DAS-16.

Chapter 7, "Programmable Interval Timer" provides details of the 8254 Programmable Interval Timer.

Chapter 8, "D/A Converters" describes the programming and use of the DAS-16's D/A converters.

Chapter 9, "Applications" describes different applications for the DAS-16.

Chapter 10, "Maintenance and Repair" gives calibration and test procedures for the DAS-16.

Chapter 11, "Instructions for Returns" describes how to contact the factory should you encounter a problem and/or need to return your DAS-16.

The Appendices provides other useful information. Appendix A describes the DAS-16's specifications. Appendix B lists the File I/O Command error messages. Appendix C lists the Call Mode error messages. Appendix D describes DMA operation. Appendix E explains how integers are stored. Appendix F provides detailed information about linearization.

(This page intentionally left blank.)

CHAPTER 2
INSTALLATION

This chapter provides instructions for the installing the DAS-16 in an IBM PC/XT/AT and compatible models. The chapter begins with procedures for unpacking and inspection. It then describes how to make a back-up copy of the Distribution Software. Descriptions of the switches and jumpers, and procedures for installing the DAS-16 are also given.

2.1 UNPACKING AND INSPECTION

Confirm that each item on the packing list has been shipped. It is a good idea to retain the outer packing material in the event that the board must be returned to the factory for repair.

1. Then, if you haven't already done so, unpack the board down to its anti-static packaging.
2. Hold the board in one hand. Place the other hand firmly on a metal portion of the system chassis. This procedure eliminates static electricity build-up, preventing any possible damage to the board.
3. After allowing a moment for static electricity discharge, carefully unwrap the board from the anti-static material.
4. Inspect the board for any possible shipping damage. If any sign of damage is detected, return the board to the factory as described in Chapter 11.

You are now ready to install your DAS-16. Set switches and jumpers as necessary. Consult section 2.4 for more information. Then, install the board as described in section 2.5.

2.2 MAKING A BACK-UP COPY OF THE DISKETTES

Distribution software is furnished on several 5.25", 360K floppy diskettes. It is also optionally available in 3.5" diskette format. Before installing the software, first create a back-up copy of the provided diskettes. Follow the general procedure outlined below:

1. Turn on your computer and display.
2. Switch to the directory containing your *DISKCOPY.EXE* DOS utility.

3. At the DOS prompt, type:

   ```
   DISKCOPY {source drive designator}: {copy drive designator}:
   ```

4. The system will prompt you through the disk copying process. Insert the source diskette into the specified drive. Then, it will ask you to insert the target disk into the other drive. This is a blank, formatted disk which is to be your back-up disk. When the copying process has been completed, the computer will ask `Copy another (Y/N)?`. Enter `N`.

5. When your copy has been completed, put the original disk is a safe place. Label the back-up copy. Use this disk to install the software.

☞ **If for any reason the provided software is unusable, contact the factory for a new copy. See Chapter 11.**

2.3 SWITCHES, JUMPERS, AND CONNECTORS

The DAS-16 has several switches and jumpers which must be configured prior to use. This section describes the switches and jumpers on the board. It also provides a description of the Analog and Digital I/O connector. Figures 2-1 and 2-2 show the location of each switch, jumper, and connector for DAS-16.

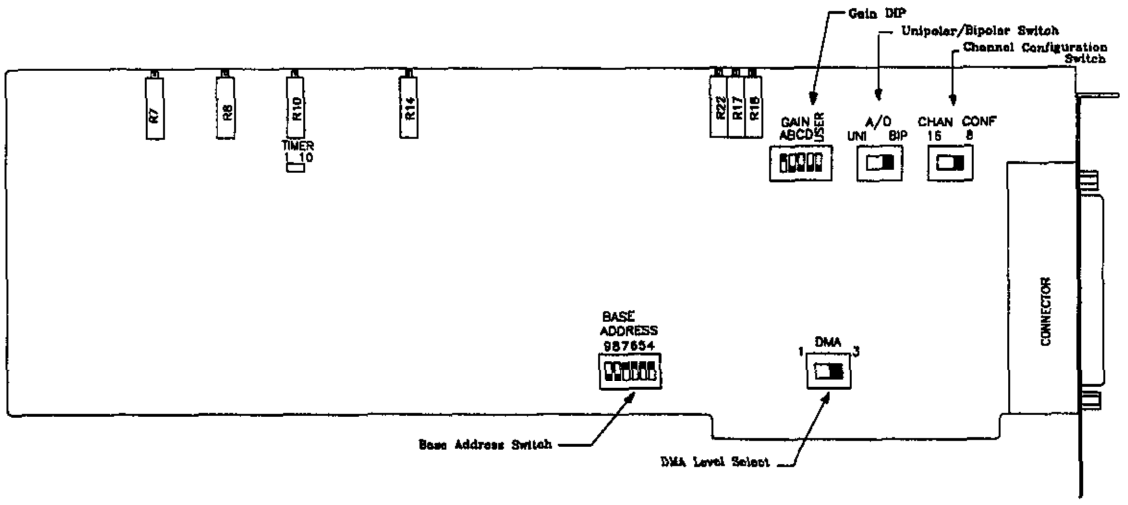


Figure 2-1. DAS-16/16F Switch and Jumper Locations

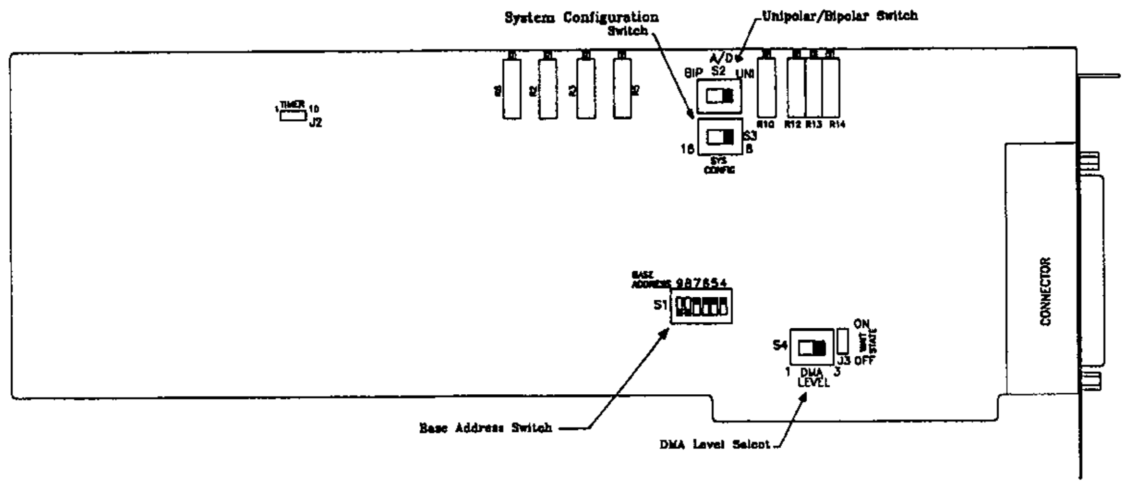


Figure 2-2. DAS-16G Switch and Jumper Locations

Descriptions of the switches and respective functions are described in the following sections. Table 2-1 lists the factory default configuration.

Table 2-1. DAS-16 Default Configuration

Switch/Jumper	Default
Base Address	300H
DMA Channel	3
A/D Mode	16-Channel Single-Ended
A/D Input Range	±10V
A/D Coding	Straight Binary
Timer Source Clock	1 MHz.

2.3.1 Switches

DAS-16 must be configured for operation via several on-board switches, as follows:

- Wait State
- Base Address
- Channel/System Configuration
- Unipolar/Bipolar Modes
- DMA Level Select
- Gain (*DAS-16/16F only*)

Except for Base Address, none of these functions requires immediate attention. If you need additional help in setting the switches, there are two utility programs provided on the distribution diskettes which may help. These are the *CONFIG16* and *INSTALL* programs. They are described further in section 2.6

Base Address Switch

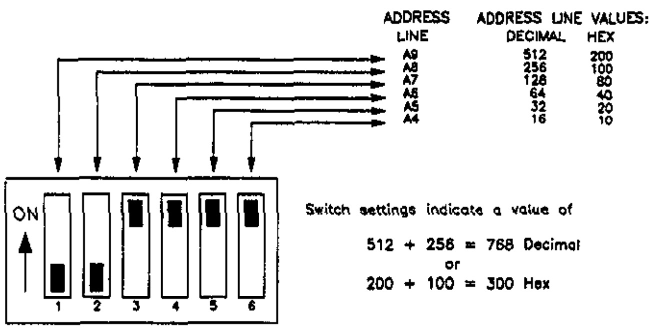


Figure 2-3. Base Address Switch

The DAS-16 is factory-configured for a Base Address of 300 Hex (768 decimal). If this Base Address is already occupied, you will need to change the Base Address switch setting. (Figure 2-3 shows the Base Address Switch.) It is essential that each peripheral device, such as a DAS-16, be assigned a unique Base Address within the range of 200 to 3F0 (512 to 1008 Decimal) and that the address is on an 16-byte boundary. Use Table 2-2 as an aid to selecting an unoccupied Base Address.

Table 2-2. IBM PC I/O Address Map

Hex Range	Usage	Hex Range	Usage
200-20F	Game/Control	310-31F	Prototype Card
210-21F	Expansion Unit (XT)	320-32F	Hard Disk (XT)
238-23B	Bus Mouse	378-37F	Parallel Printer
23C-23F	Alt. Bus Mouse	380-38F	SDLC
278-27F	Parallel Printer	3A0-3AF	SDLC
2B0-2BF	Alt. EGA	3B0-3BB	MDA
2C0-2CF	Alt. EGA	3BC-3BF	Parallel Printer
2D0-2DF	Alt. EGA	3C0-3CF	EGA
2E0-2E7	GPIB(AT)	3D0-3DF	CGA
2E8-2EF	Serial Port 4	3E8-3EF	Serial Port 3
2F8-2FF	Serial Port 2	3F0-3F7	Floppy Disk
300-30F	Prototype Card	3F8-3FF	Serial Port 1

Channel Configuration Switch (System Configuration Switch)

This slide switch is used to select whether 8 differential or 16 single-ended analog input channels are to be used. On the DAS-16/16F, this switch is marked **CHAN CONF**. On the DAS-16G, it is marked **SYS CONFIG**. On either switch, to select eight differential input channels, move the switch to the right (towards the **8** designator). Likewise, to select 16 single-ended analog input channels, move the switch to the left (towards the **16** designator).

DMA Level Switch

This slide switch selects the DMA level. This two-position slide switch selects the DMA as 1 or 3. Some early PC's utilize Channel 3 for their hard drives. If this is the case, you can not choose DMA Channel 3 for your DAS-16. Run the program *LEV3.EXE* supplied with your DAS-16 software to determine if you can use DMA Channel 3. More information regarding DMA is provided in Appendix B.

A/D Switch

This slide switch controls the input range. When moved to the UNI (unipolar) position, inputs can be positive only (ranges are from zero to some positive full-scale voltage). When moved to the **BIP** (bipolar) position, inputs can range from equal negative to positive full scale limits.

GAIN Switch (DAS-16/16F Only)

This switch is a 5-position DIP switch and is found only on the DAS-16/16F boards. The five positions on the switch are marked A, B, C, D, and USER. These are used to set the range as shown in the Table 2-3.

Table 2-3. DAS-16/16F Gain Switch Positions

A	B	C	D	USER	UNIPOLAR	BIPOLAR
OFF	OFF	OFF	OFF	OFF	N/A	± 10V
ON	OFF	OFF	OFF	OFF	0-10V	± 5V
ON	ON	OFF	OFF	OFF	0-5V	± 2.5V
ON	OFF	ON	OFF	OFF	0-2V	± 1 V
ON	OFF	OFF	ON	OFF	0-1V	± 0.5V
ON	OFF	OFF	OFF	ON	User Set	User Set

If you need more help in setting the Gain Switch, run the *QBCAL.EXE, QBCALF.EXE,* or *QBCALG.EXE* calibration programs and then select the A/D Converter Calibration option. Also, the range of gains can be extended by selecting and installing your own input resistor. For more information concerning this, refer to section Chapter 9.

2.3.2 Jumpers

The DAS-16/16F has only one jumper block: the **TIMER** jumper block. The DAS-16G has two jumper blocks: the **TIMER** and **WAIT STATE** jumper blocks.

The TIMER Jumper

This jumper selects a timer input frequency of 10MHz or 1MHz. For most applications, 10MHz is more useful and will give a finer time interval resolution at higher conversion rates.

The WAIT STATE Jumper

This jumper, when set to ON, will cause the DAS-16G to generate a 450ns wait state on I/O accesses. This jumper will only need to be set to ON, if you have built your own computer and know that the I/O read and write pulses will be less than 210nS duration.

Generally, this jumper should be in the OFF position. Most computer manufacturers include internal wait states on all I/O accesses to maintain compatibility with other standard peripherals.

2.3.3 Connectors

Analog and Digital I/O occurs through a 37-pin, D-type connector that projects through the computer case at the rear of the board. The mating connector for DAS-16 is a standard, 37-pin D-type female such as an ITT/Cannon #DC-37S for soldered connections. Insulation displacement (flat cable) types are readily available (for example, Amp #745242-1). Other manufacturers make equivalent parts. This connector and its signal conductor functions are described in Figure 2-4.

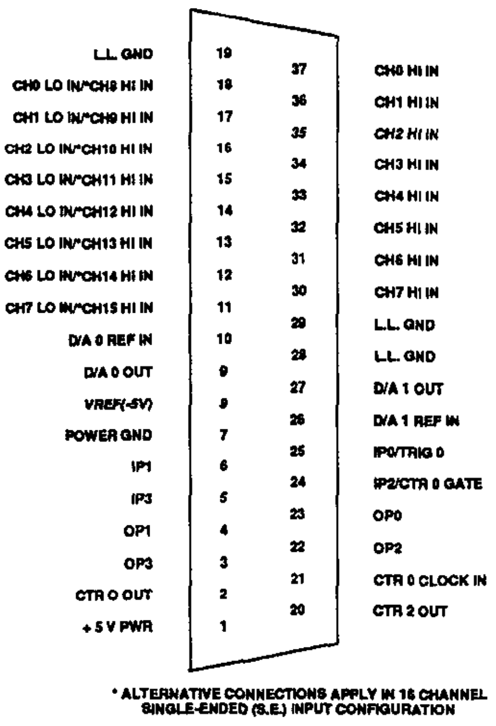


Figure 2-4. Main I/O Connector

☞ ***NOTE:*** Pins 11 thru 18 perform a double function depending on the setting of the channel configuration switch. In 8-channel differential configuration, these pins provide the low inputs of Channels 0 - 7 corresponding to the high inputs of these channels on Pins 30 - 37. In 16 channel single-ended configuration, they provide additional channel high inputs for Channels 8-15.

2.4 BOARD INSTALLATION

This section provides general instructions for installing the DAS-16 Board. For more detailed information regarding installation of peripheral boards, consult the documentation provided with your computer.

WARNING

DO NOT ATTEMPT TO INSERT OR REMOVE ANY ADAPTER BOARD WITH THE COMPUTER POWER ON! THIS COULD CAUSE DAMAGE TO YOUR COMPUTER!

To install the DAS-16 Board:

1. Turn the power to the PC and to all attached options OFF.

2. Unplug the power cords of all attached options from the electrical outlets. Make a note of where all the cables and cords are attached to the rear of the system unit and disconnect.

3. Remove the cover of the PC. To do this, first remove the five cover mounting screws on the rear panel of the computer. Then, slide the cover of the computer about 3/4 of the way forward. Tilt the cover upwards to remove.

4. Choose an available option slot. Loosen and remove the screw at the top of the blank adapter plate. Then slide the plate up and out to remove.

5. Hold the DAS-16 in one hand. With the other hand, touch any metallic part of the PC/AT cabinet. This will safely discharge any static electricity which has built-up in your body.

6. Set the switches and jumpers as described in section 2.3.

7. Align the gold edge connector with the edge socket and the back adapter place with the adapter plate screw. Gently press the board downward into the socket. Re-install the adapter plate screw.

8. Replace the computer's cover. Tilt the cover up and slide it onto the system's base, making sure the front of the cover is under the rail along the front of the frame. Install the mounting screws.

9. Plug in all cords and cables. Turn the power to the computer back on.

2.5 SOFTWARE INSTALLATION

The DAS-16 software should be installed on your computer's hard drive. Installing the software in this manner requires the use of the DOS **COPY** command. To install the software:

1. Turn on your PC and its display. You should see a prompt which indicates you are at the DOS level.

2. The following instructions create a special directory for the DAS-16 Disk files and assume this directory is called **DAS16**. At the DOS prompt, change to the root directory (or other directory you want the directory to be branched off from), i.e. type:

```
cd \
```

3. Next, create the DAS-16 directory. Type:

   mkdir \DAS16 OR **md \DAS16**

   Change to the DAS-16 directory by typing:

```
cd \DAS16
```

☞ ***NOTE:*** You may also want to create separate directories branched from the main DAS-16 directory for the Call Mode software and the Pop-Up/File I/O Command software.

4. Place each DAS-16 Disk into the floppy drive (assume this is drive a:) and type:

```
copy a:*.*
```

Once you have successfully installed the software you will be ready to run the *CONFIG16.EXE* or *INSTALL.EXE* programs.

2.6 BEFORE YOU BEGIN PROGRAMMING

If you have changed any of the jumper or DIP switch settings, before you begin programming, you must run one of two utility programs. These programs can also be run if you have trouble setting the switches or jumpers. The program that you choose is dependent on the programming interface selected. If you are going to program the DAS-16 via the Pop-Up and/or the File I/O Command set, be sure run the *CONFIG16* program. This program generates a file containing all of your DIP switch and jumper configurations which is read in by the *MDAS16* driver. (This is further described in Chapter 3.) Note that the configuration file created by the CONFIG16 utility is in binary format and therefore not readable or suitable for editing using standard text editors. All changes to these files must occur using this utility.

You should run the *INSTALL* program if you intend to program the DAS-16 via the CALL Mode Interface or Low-Level Programming techniques. This program generates the *DAS16.ADR* file which is read in by some of the BASIC example programs. (See Chapter 5 for more information.)

The remainder of this section describes these two programs. It assumes that you have already transferred the software to your hard disk.

CONFIG16.EXE

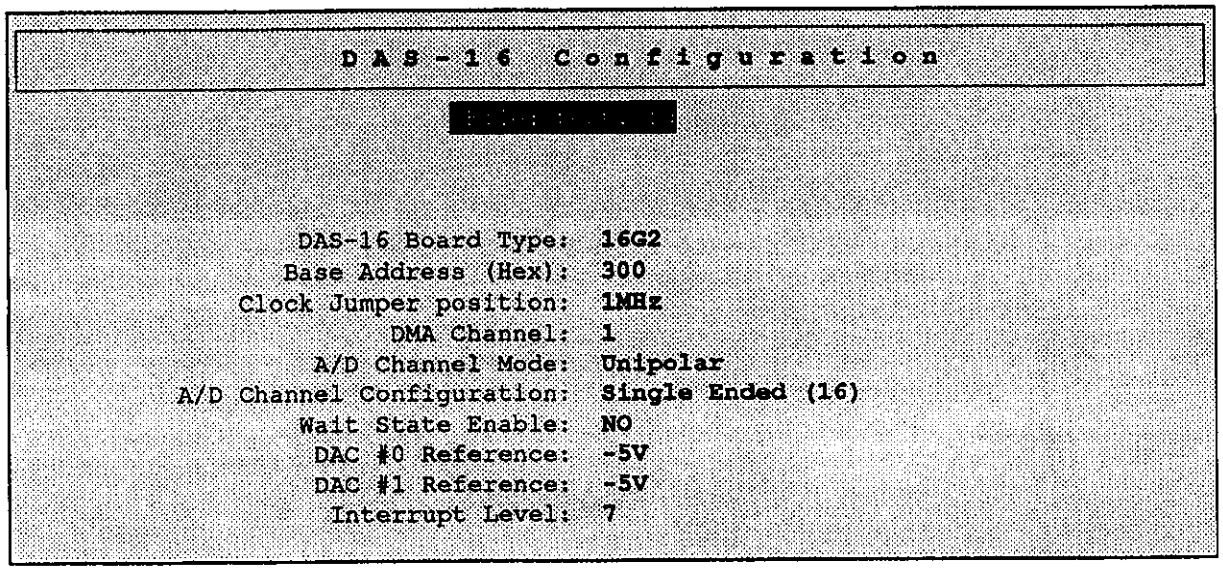
If necessary, switch to the appropriate directory. Then, to run the *CONFIG16* program, at the DOS prompt type:

```
CONFIG16 filename
```

Where:

`filename` specifies the file to which the configuration parameters are to be saved. This can be any valid DOS filename and may include a DOS Drive and path specification where necessary. For example: **`CONFIG16 C:\DAS16\DAS16G2.CFG`**

If the specified file is a new file, the default Factory Configuration is displayed. Otherwise, if the file already exists, the contents of the file are shown. Figure 2-5 shows an example of creating a new configuration file called *TEST.CFG.*

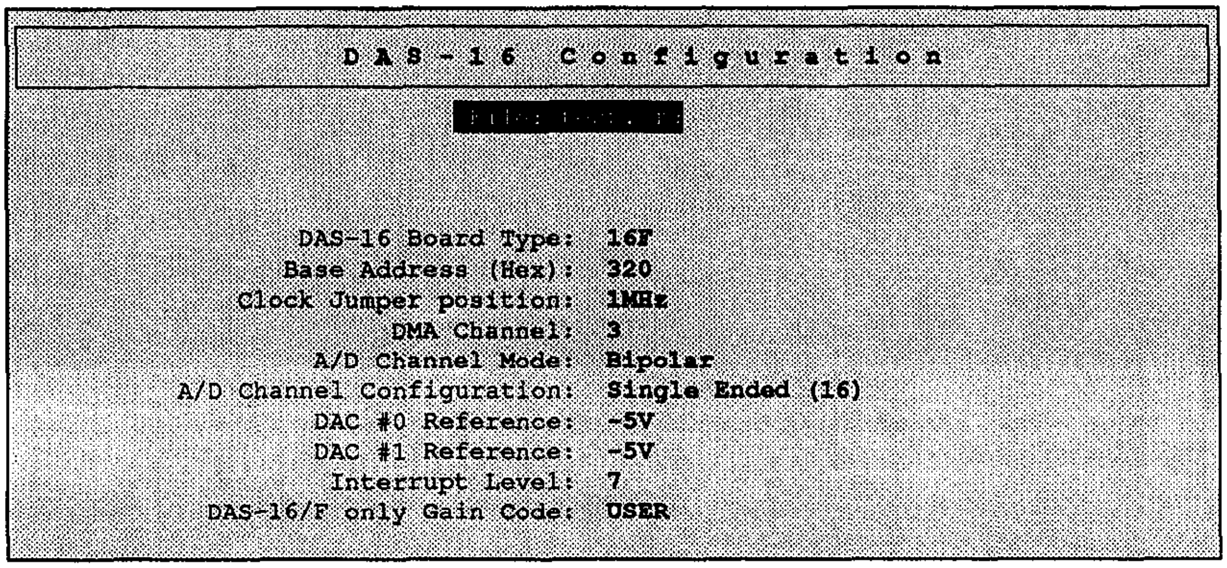


```
                    D A S - 1 6   C o n f i g u r a t i o n

            DAS-16 Board Type:  16G2
           Base Address (Hex):  300
        Clock Jumper position:  1MHz
                  DMA Channel:  1
             A/D Channel Mode:  Unipolar
    A/D Channel Configuration:  Single Ended (16)
            Wait State Enable:  NO
             DAC #0 Reference:  -5V
             DAC #1 Reference:  -5V
              Interrupt Level:  7
```

Figure 2-5. Creating a New Configuration File

The program will then prompt you through each of the settings. Follow the directions to make any changes. When you have completed making your changes, you should finish with a screen similar to the one depicted in Figure 2-6. Save your changes as directed.



```
                     D A S - 1 6   C o n f i g u r a t i o n

                              File: [illegible]


                     DAS-16 Board Type:  16F
                    Base Address (Hex):  320
                 Clock Jumper position:  1MHz
                           DMA Channel:  3
                      A/D Channel Mode:  Bipolar
             A/D Channel Configuration:  Single Ended (16)
                      DAC #0 Reference:  -5V
                      DAC #1 Reference:  -5V
                       Interrupt Level:  7
                DAS-16/F only Gain Code:  USER
```

Figure 2-6. Modified Configuration File

INSTALL.EXE

Unlike the *CONFIG16* program, this program only deals with the Base Address Switch setting. To run the *INSTALL* program, follow the instructions given below.

1. If necessary, switch to the appropriate directory. Then, to run the *INSTALL* program, at the DOS prompt type:

   INSTALL and press Enter.

2. The program will respond with the prompt **DESIRED BASE ADDRESS ----> ?**. Type in a Base Address. You may specify a decimal number or a Hex number preceded by &H, such as &H300. The computer will display the corresponding Base Address switch settings. If the entry is unacceptable, the computer will display an explanatory statement and a request for another entry.

CHAPTER 3
INTRODUCTION TO THE MDAS16 DRIVER

This chapter introduces you to the *MDAS16* Driver. This driver supports two programming interfaces: the Pop-Up Menu and the File I/O Command Set. The *MDAS16* Driver can be loaded and unloaded from the DOS command line (or optionally can be loaded from your *AUTOEXEC.BAT* file) and is completely independent of the *DASG.LIB* driver. (This driver supports the Call Modes and is described in Chapter 5.) The *MDAS16* driver fully supports all models of the DAS-16.

The Pop-Up Menu is a diagnostic tool designed to aid you in the initial set-up and debug of your DAS-16. The Pop-Up is designed around an intuitive user interface which is controlled using a mouse or from the keyboard. The Pop-Up Menu allows you, without having to program a line of code, to perform such tasks as:

- ADC and transfer using Interrupts, DMA, Synchronous, and Block Scan
- DAC to one or both D/A Converters
- Digital I/O and Square Wave Output
- Provisions for saving A/D data to user disk file.

The File I/O Commands allow you to customize and extend the capabilities of the Pop-Up Menu for your DAS-16 application. The File I/O Commands are programming-language independent and easily used from any language supporting standard file I/O. The File I/O Commands are more fully described in Chapter 4.

The remaining sections of this chapter describe how to install the *MDAS16* driver and how to use the Pop-Up Menu.

3.1 THE MDAS16 DRIVER

The *MDAS16* driver consists of two pieces of loadable/unloadable TSR (Terminate Stay Resident) software. These are the *VI.EXE* (or optionally the *VITASK.EXE*) driver and the *MDAS16.EXE* driver.

The *VI.EXE* driver (requires 55K of memory) is a control program which provides the graphics display routines required by the Pop-Up Menu. It also provides other software routines which are required to support the File I/O Command interface. If you are not going to use the Pop-Up Menu or you find that the *VI.EXE* program takes up too much memory for your application, you can alternatively load the *VITASK.EXE* driver. *VITASK.EXE* requires only 21K of memory.

The *MDAS16.EXE* file is the software driver which supports the Pop-Up and File I/O interfaces for controlling the DAS-16. This driver takes up approximately 54K. The *MDAS16.EXE* driver reads in the contents of the software configuration file which was created when the *CONFIG16* program was run (See Chapter 2). This driver is loaded after the *VI.EXE* or *VITASK.EXE* driver. Note that this driver supports only the Pop-Up Menu and File I/O Command interfaces. If you are going to program the DAS-16 via the Call Mode interface or through Low-level programming, you must use another driver as described in Chapter 5.

The *VI* and *MDAS16* drivers are TSR programs which can be loaded manually from DOS or can be loaded from your *AUTOEXEC.BAT* file or another batch file. One advantage to loading the drivers from a batch file (such as *AUTOEXEC.BAT*) is that the software is automatically available once your computer has been powered-up. This is useful if you are frequently going to use the DAS-16. However, the drivers do occupy memory and may limit the size of other programs you use. If this becomes a problem or you are only running DAS-16 programs once in a while, you will probably want to load the drivers from the DOS prompt. You can also elect to unload the drivers, while using memory intensive software, as described in section 3.1.3.

☞ ***Note:*** Normally, the VI and *MDAS16* drivers are loaded into DOS Conventional memory area (the 1st 640K bytes). However, if you are using 80386 class PC with 1M memory or more, you may be able to load these TSR programs into "high memory" (the area between 640KB and 1MB). For more information regarding this procedure, contact the Technical Support Department as described in Chapter 11.

The following subsections describe how to load the DAS-16 driver. They assume that the Pop-Up Menu will be using the default Hot Key and Configuration selections. If, after you have run the Pop-Up Menu, you discover that these conflict with assignments made by other programs, the default selections can be changed as described in section 3.5.

3.1.1 The M16SETUP Program

When you load the driver via the *AUTOEXEC.BAT* file, your files will remain loaded until you shut down the computer or manually unload the drivers from DOS. (See Section 3.1.3.) Note, however, that if you unload the drivers from DOS and need to re-activate them, you will have to re-load the drivers from the DOS command line as described in section 3.1.2. If they are loaded via the *AUTOEXEC.BAT* file, they will automatically be re-loaded each time the computer is powered-up.

If you elect to load the DAS-16 driver from the *AUTOEXEC.BAT* file, you will need to run the *M16SETUP* program. This program will edit the contents of your current *AUTOEX-EC.BAT* file for you. To use the *M16SETUP* program, change to the directory where you loaded the DAS-16 software and type:

```
M16SETUP
```

Then, follow the prompts. Once you have completed the *M16SETUP* program, it will modify your *AUTOEXEC.BAT* file for you. Then, re-boot the computer (Press Ctrl Alt Del.) and you will be ready to use the Pop-Up Menu and/or to develop a program incorporating the File I/O Commands. If you only intend to develop programs that use the File I/O Commands, proceed to Chapter 4.

3.1.2. Loading the DAS-16 Driver from DOS

If you elect to load the drivers manually from DOS, you must remember to load the VI driver first, then the *MDAS16* driver.

To load the VI driver, at the DOS prompt type:

```
[C:] [PATH]VITASK
```

or, in the case of VI

Add the line `[C:] [PATH]VI [MONO] [/HK=] [/MK=] [/SK=]`

Where:

`[C:]`	is an optional parameter which indicates the DOS driver where *VI.EXE* or *VITASK.EXE* is installed. You must only include this if the driver is not installed in the root directory. Do not include the square brackets ([ ]) in your selection.

`[PATH]`	is an optional parameter which represents the path to the directory in which *VI.EXE* or *VITASK.EXE* is installed. You must only include this if the driver is not installed in the root directory. Do not include the square brackets ([ ]) in your selection.
`[MONO]`	is an optional parameter which must be included for computers with monochrome displays. If this parameter is not included, it is assumed that a color monitor is being used. Do not include the square brackets ([ ]) in your selection.
`[HK=]`	is an optional parameter which overrides the default setting of the Help Key (Alt H). Whenever the indicated key combination is pressed, the help screen will be displayed - if the DAS-16 Pop-up menu is displayed. On-line help provides some brief start-up instructions and a listing of all the File I/O commands.

You can use the Ctrl or Alt keys in combination with any other keyboard key as the new Help Key sequence, e.g.:

```
/HK= ALT ?
```

`[MK=]` is an optional parameter which overrides the default setting of the Mode Select Key (Alt M). Whenever the indicated key combination is pressed, the DAS-16 Pop-Up Menu will be placed in keyboard entry mode.

You can use the Ctrl or Alt keys in combination with any other keyboard key as the new Mode Select Key sequence, e.g.:

```
/MK= CTRL L
```

`[SK=]` is an optional parameter which overrides the default setting of the Instrument Select Key (Alt M). If you have multiple instruments which use the *VI.SYS* driver, this key sequence is used to toggle between the various Pop-up Menus.

You can use the Ctrl or Alt keys in combination with any other keyboard key as the new Instrument Select Key sequence, e.g.:

```
/MK= CTRL L
```

For example:

```
C:\DAS16\VITASK
```

or

```
C:\DAS16\VI
```

Then, install the MDAS16 driver. Type the line:

```
[C:][PATH] MDAS16 [/DAS=] [/F=] [/BA=] [/G=]  [/REF=] [/NAME=]
```

The switches do not have to appear in any particular order, unless otherwise specified.

Where:

`[C:]`	is an optional parameter which indicates the DOS driver where *MDAS16.EXE* is installed. You must only include this if the driver is not installed in the root directory. Do not include the square brackets ([ ]) in your selection.
`[PATH]`	is an optional parameter which represents the path to the directory in which *VI.EXE* or *VITASK.EXE* is installed. You must only include this if the driver is not installed in the root directory. Do not include the square brackets ([ ]) in your selection.
`[/DAS=]`	is an optional parameter which indicates whether a DAS-16G1, DAS-16G2, DAS-16, or DAS-16F is used. If this switch is not included, the default of DAS-16G2 is assumed. Do not include the square brackets ([ ]) in your selection. Otherwise, use one of the following:

/DAS=16G1	for the DAS-16G1
/DAS=16G2	for the DAS-16G2
/DAS=16	for the DAS-16
/DAS=16F	for the DAS-16F

`[/F=]`	This switch allows you to specify a configuration file which overrides the DAS-16's default settings. The configuration file must be previously created using the *CONFIG16* utility. Do not include the square brackets ([ ]) in your selection.

[/BA=] The switch allows you to specify a new Base Address for your installed DAS-16. Note that the address specified here must reflect the actual DIP switch positions on the actual board. Do not include the square brackets ([ ]) in your selection. Specify the base address in decimal or in Hexdecimal as follows:

```
/BA=768                    'Decimal Address
/BA=&H310                  'Hexdecimal Address
```

[/G=] This switch is optional when a DAS-16 or DAS-16F are used; it is ignored when a DAS-16G is used. This switch specifies a Gain Code 0 thru 4 of the actual Gain DIP switch positions on the board itself. Note that this switch must come after the **/DAS** or **/F** switches in the command line. If not supplied, Gain Code of 0 is assumed.

```
/G=1                        '0-5V if Unipolar, or
                            '+/-5V if Bipolar.
```

[/REF=] This switch is optional. It specifies one of four DAC voltage reference values (-5, -10, +5 and +10) respectively. If not supplied, DAC reference of -5 Volts is assumed. Note that both DAC's use the same voltage reference!

```
/REF=-10V        'New Reference is -10 Volts
```

[/PK=] This switch is optionally used to override the default Pop-Up Key sequence: Alt F4. You can use the Ctrl or Alt keys in combination with any other keyboard key as the new Pop-Up sequence, e.g.:

```
/PK=CTRL A
```

/NAME= Use this switch if you are loading multiple versions of the *MDAS16* Driver for the purpose of controlling multiple DAS-16's simultaneously. The default device name for the first *MDAS16* driver is $MDAS16. Use this switch to specify a new name for each new version of *MDAS16.*

```
/NAME=MDAS17               'The $ is not required!
```

Note that Device name used must be in accordance with the MS DOS requirements for file name specification.

For example:

```
C:\DAS16\MDAS16  /F=TEST.CFG
```

3.1.3 Unloading the DAS-16 Driver from DOS

To unload the DAS-16 driver software, first unload the *MDAS16* software and then the VI software. For example, make sure your path name is correct, then at the DOS prompt, type:

```
MDAS16 /U
```

Then:

`VI /U` or `VITASK /U`

3.2 ABOUT THE POP-UP MENU

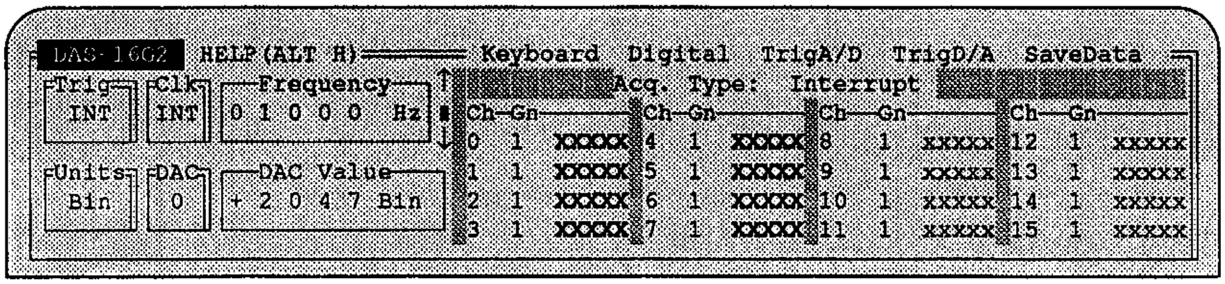
Once you have loaded the driver, you can bring up the Pop-Up Menu by pressing the Pop-Up Key Sequence (Alt F4 by default). This will display the first panel of the Pop-Up Menu. The Pop-Up Menu has a total of three panels. The first, the **Analog I/O Panel**, is used to configure the DAS-16 for A/D (or D/A) conversions. The Analog I/O Panel is described in detail in section 3.3.1. The second panel is the **Digital I/O Panel**. This is used to configure the DAS-16 for digital I/O operations. This panel is described in detail in section 3.3.2. The third screen is the **File Panel**. This screen is used to configure the file to which data is saved.

There are several Hot Keys which are used in the Pop-up Menu. If they conflict with assignments made by existing programs, they can be changed by the methods described in section 3.1.2. These default assignments for these keys are:

Esc	**Escape Key.** This key allows you to exit keyboard mode.
Alt F4	**Pop-Up Key.** This is the key sequence used to display the Pop-Up Menu.
Alt H	**Help Key.** This key sequence can be used to display help at any time during Pop-up Menu programming.
Alt Tab	**Instrument Select Key.** If you have more than one PCIP-type instrument, this key sequence is used to toggle between the Pop-Up Menus.
Alt M	**Mode Select Key.** This key sequence changes user input to the keyboard.

3.3.1 The Analog I/O Panel

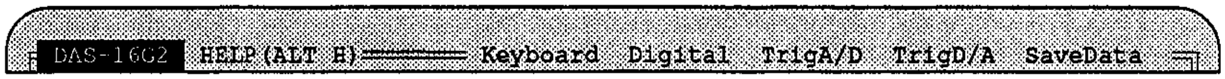
This is the First Panel which appears, it is used to configure the DAS-16 for analog I/O. Figure 3-1 shows the Analog I/O Panel.



```
 DAS-16G2  HELP(ALT H)══════ Keyboard  Digital  TrigA/D  TrigD/A  SaveData
 Trig  Clk  Frequency          Acq. Type:  Interrupt
 INT   INT  0 1 0 0 0  Hz   Ch─Gn         Ch─Gn         Ch─Gn         Ch─Gn
                            0  1  xxxxx   4  1  xxxxx   8  1  xxxxx   12  1  xxxxx
 Units DAC  DAC Value       1  1  xxxxx   5  1  xxxxx   9  1  xxxxx   13  1  xxxxx
 Bin   0    + 2 0 4 7 Bin   2  1  xxxxx   6  1  xxxxx   10 1  xxxxx   14  1  xxxxx
                            3  1  xxxxx   7  1  xxxxx   11 1  xxxxx   15  1  xxxxx
```

Figure 3-1. Analog I/O Panel

The Analog I/O Panel consists of two parts: the Panel Menu (which runs across the top of the panel) and the Analog Panel itself. The Analog Panel itself allows you to configure the DAS-16; whereas the Panel Menu (See Figure 3-2.) lets you switch data entry modes, toggle between the three panels, trigger an A/D or D/A operation, and save the data to a file on disk. These panels are described in more detail below.



```
 DAS-16G2  HELP(ALT H)══════ Keyboard  Digital  TrigA/D  TrigD/A  SaveData
```

Figure 3-2. Panel Menu

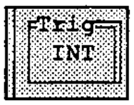
Panel Menu This menu contains five options: **Keyboard, Digital, Trig A/D, Trig D/A,** and **Save Data.** These perform the following functions:

Keyboard -	This option toggles between the Keyboard and Mouse Data Entry mode. Note that this function is also performed by the **Mode Select Key.**
Digital -	This option brings up the **Digital I/O Panel.**
Trig A/D -	This option triggers an A/D operation.

Trig D/A - This option triggers a D/A operation.

Save Data - This option will save data to a disk file. The disk file must have already been configured and selected through the use of the **File Panel.**

Analog Panel This is the section of the panel which is used to configure the DAS-16 for your intended Analog I/O operation. It has a series of "buttons" which can be manipulated as directed at the bottom of the screen. They are:



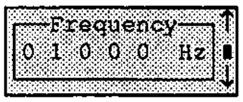
Selects the initial trigger source used to start the next conversion. You can select an internal (**INT**) or external (**EXT**) trigger source. If **INT** is selected, conversions are triggered on command by selecting **TrigA/D** or **TrigD/A** from the Panel Menu. If EXT is selected, conversions are started when a low-to-high transition is detected on Digital Input line IP1. When external triggering is selected, **TrigA/D** and **TrigD/A** are used as enable/disable control for monitoring the external trigger line.

Note that, after a **TrigA/D** (or **TrigD/A**) is initiated, a blinking EXT in this button implies that the DAS-16 is waiting for an external trigger event to occur.

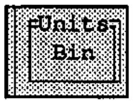
EXT can be selected regardless of whether the external source is analog or digital. (Analog External trigger is available through the File I/O Commands or the Call Modes.)

Defines the A/D conversion clock source as internal (**INT**) or external (**EXT**). When **INT** is selected, conversions are performed according to the value specified in the **Frequency** data field. When **EXT** is selected, conversions occur according to the signal detected on Digital Input line IP0/TRG0.

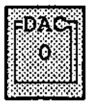
Note that, after a **TrigA/D** is initiated, a blinking **EXT** in this button implies that the DAS-16 is waiting for an external trigger event to occur.



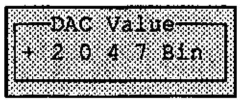
Selects an internal clock frequency. The frequency selected must be within the range 1 to 99999 Hertz. Refer to section 3.3 for operational details.


Units
Bin

Selects the unit of measure to be used. You can select **BIN** (binary), **V** (volts), or **mV** (millivolts).


DAC
0

Defines the DAC channel to be used in the next DAC output (i.e., **TrigD/A**) operation. You can select **0, 1,** or **0&1.**


DAC Value
+ 2 0 4 7 Bin

Specifies the DAC value to output for the next DAC output operation. This value is always specified as a code value between 0 and 4095.

Acq. Type: Interrupt

Sets the type of the next A/D Acquisition. You can select between **Interrupt, DMA, Synchronous,** and **Scan** modes. All methods will work with both Internal and External Clock and Trigger sources; however, the maximum transfer rates differ according to the mode selected. For all modes <u>except scan</u>, the acquisition process continues until one of the following occurs:

1. the desired number of samples is reached
2. an error condition occurs
3. the acquisition is interrupted by the user.

More detailed descriptions of the modes follow:

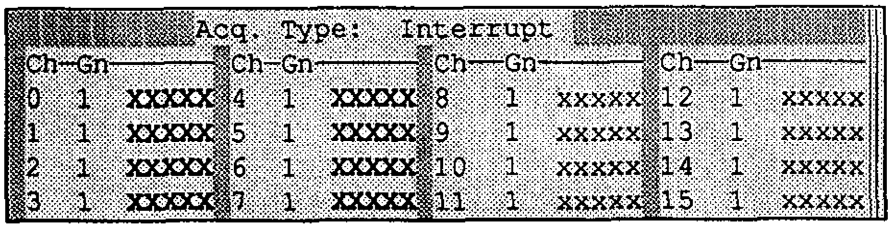
INT This selects the interrupt mode. This mode allows data acquisition and transfer using one of five (5) user programmable interrupt levels. (Selected on the Digital I/O Panel.) When a conversion is completed, the DAS-16 issues an interrupt which the software detects and then reads the data value into memory. Because of the involvement of the software (and the CPU) to read the data into memory, the maximum conversion rate is limited to approximately 5KHz. In this mode only one gain for all channels is used. This mode operates completely in the background, which means you can perform most other functions of the board while this is in progress.

DMA DMA mode allows for data to be transferred from the board directly to PC memory without the involvement of the CPU or any driver software. Because of this, the maximum conversion rates for your DAS-16 board are achievable (70KHz to 100KHz). In this mode, only one gain for all channels is used. As with the INT mode, this mode operates completely in the background, which means you can perform most other functions of the board while this is in progress.

SYNC This select the synchronous mode. This mode, unlike the interrupt and DMA modes, operates in the foreground. That is, once an A/D operation is initiated, no other board function may performed until this operation terminates. The advantage of this mode is that it allows the use of different gains with different channels during acquisition. The maximum conversion throughput for this mode is machine-dependent, i.e. it depends on the PC you are using, its CPU speed, bus speed, etc.

SCAN Scan mode allows for continuous sampling and display of the selected channels. Only the number-of-channels worth of data are acquired per scan and the Pop-Up display is updated with the latest data twice per second.

A/D Data Display and Channel/Gain Select



```
              Acq. Type:  Interrupt
Ch-Gn---------Ch-Gn---------Ch--Gn---------Ch--Gn---------
0  1   XXXXX  4  1   XXXXX  8    1   xxxxx 12   1   xxxxx
1  1   XXXXX  5  1   XXXXX  9    1   xxxxx 13   1   xxxxx
2  1   XXXXX  6  1   XXXXX  10   1   xxxxx 14   1   xxxxx
3  1   XXXXX  7  1   XXXXX  11   1   xxxxx 15   1   xxxxx
```

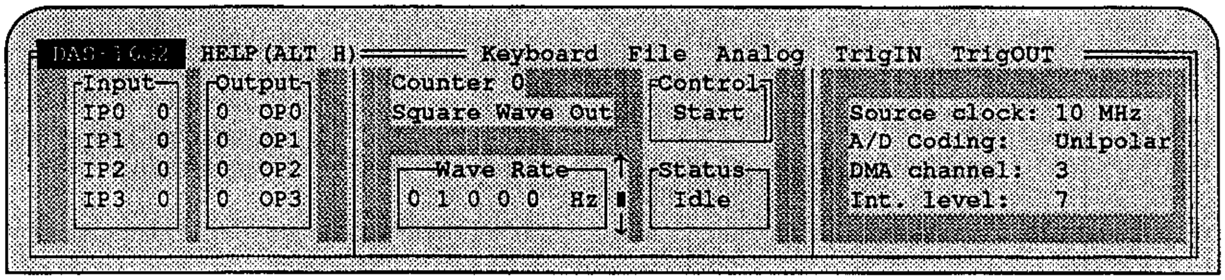
This portion of the panel is used to display the A/D data. The following symbols have special meanings:

xxxxx	Data is irrelevant. Channel has not been selected.
XXXXX	Channel has been selected; however data is invalid.
+++++	Selected channel is over range.
-----	Selected channel is under range.

To select new channel scan limits, click on the desired starting and ending channel numbers (Ch). For the DAS-16G1/G2/AG, change the channel(s) gain by clicking on the desired Gain (Gn) number shown. Note that different gains for different channels are allowed in Synchronous Acquisition mode only.

3.3.2 The Digital I/O Panel

This panel is used to configure the DAS-16 for digital operations. Figure 3-3 shows the Digital I/O Panel.

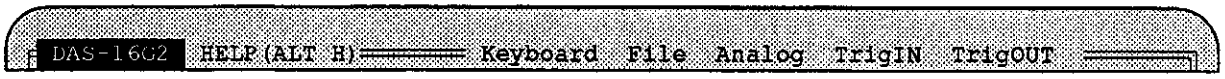


```
DAS-16G2  HELP(ALT H)  Keyboard  File  Analog  TrigIN  TrigOUT
Input      Output      Counter 0              Control
IP0  0     0  OP0      Square Wave Out         Start         Source clock: 10 MHz
IP1  0     0  OP1                                            A/D Coding:   Unipolar
IP2  0     0  OP2       Wave Rate             Status         DMA channel:  3
IP3  0     0  OP3       0 1 0 0 0  Hz          Idle          Int. level:   7
```

Figure 3-3. Digital I/O Panel

Like the Analog I/O Panel, the Digital I/O Panel consists of two parts: the Panel Menu (which runs across the top of the panel) and the Digital Panel. The Analog Panel itself (See Figure 3-4.) allows you to configure the DAS-16; whereas the Panel Menu lets you switch data entry modes, toggle between the three panels, and trigger a digital operation. These panels are described in more detail below.

Panel Menu This menu contains five options: **Keyboard, File, Analog, TrigIN,** and **TrigOUT**. These perform the following functions:

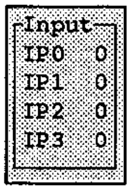


```
DAS-16G2  HELP(ALT H)  Keyboard  File  Analog  TrigIN  TrigOUT
```

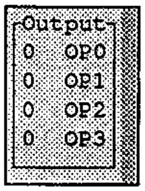
Figure 3-4. Digital Panel Menu

Keyboard -	This option toggles between the Keyboard and Mouse Data Entry mode. Note that this function is also performed by the **Mode Select Key**.
File -	This option brings up the **File Panel.**
Analog -	This option brings up the **Analog I/O Panel.**
TrigIN -	This option triggers an input operation.
TrigOUT -	This option triggers an output operation.

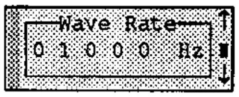
Digital Panel This is the section of the panel which is used to configure the DAS-16 for your intended I/O operation. It has a series of "buttons" which can be manipulated as directed at the bottom of the screen. Any buttons which can be modified are displayed in green. They are:



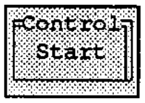
Shows the value present at the DAS-16 input port (pins IP0 through IP3). The default setting is 0.



Specifies the value to be output on pins OP0 to OP3, when triggered. Set each bit to **1** or **0**. Note that there is no range checking performed on this value. The default setting is 0.



Select the rate of the square wave output. This value must be between 2 and 25000 Hertz. The default setting is 1000 Hertz.



STARTS or **STOPS** a square wave output. The default setting is IDLE.

`Int. level: 7`

Selects the Interrupt Level to be used during the next A/D acquisition. Interrupt Levels 2 - 7 are allowed. Interrupt Level 7 is the default.

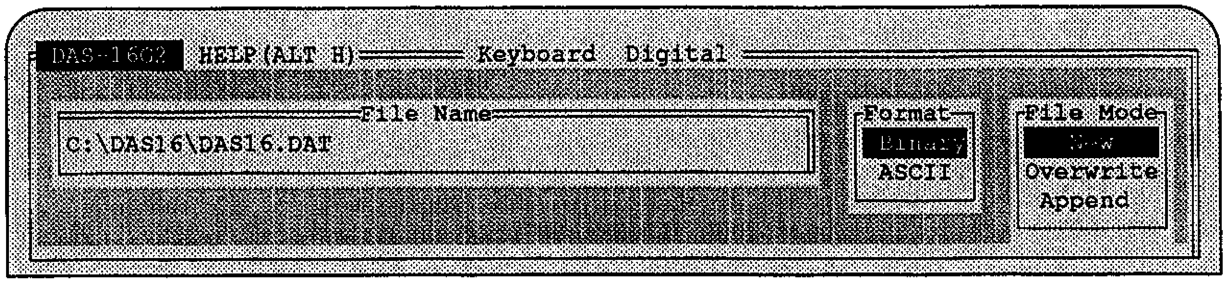
`Status`
`Idle`

This display-only field reflects the Wave Output status as one of the following:

`IDLE`	Wave output is not active. (Default)
`ACTIVE`	Wave output is active. To stop, select **STOP** from the **Control** button or externally bring IP2 low.
`IP2LOW!`	Wave Output has been initiated, but has not yet started, pending IP2 going high.

3.3.3 *The File Panel*

This panel is used to set-up a file on disk to which data can be saved. Figure 3-5 shows the File Panel.



```
File Name: C:\DAS16\DAS16.DAT
Format: Binary / ASCII
File Mode: New / Overwrite / Append
```

Figure 3-5. File Panel

Like the Analog and Digital I/O Panels, the File Panel consists of two parts: the Panel Menu (which runs across the top of the panel) and the File Panel. The File Panel itself allows you set-up the file; whereas the Panel Menu lets you switch data entry modes, and toggle between the three panels. These panels are described in more detail below.

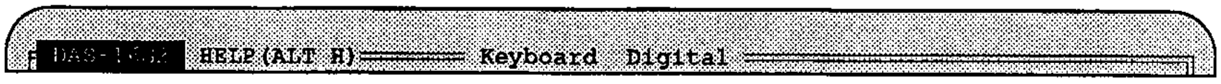
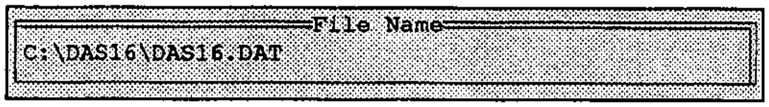


Figure 3-6. File Panel Menu

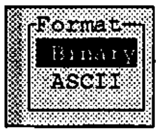
Panel Menu This menu contains two options: **Keyboard** and **Digital.**

Keyboard -	This option toggles between the Keyboard and Mouse Data Entry mode. Note that this function is also performed by the **Mode Select Key.**
Digital -	This option brings up the **Digital I/O Panel.**

File Panel This is the section of the panel which is used to configure a file to which data can be saved. It has a series of "buttons" which can be manipulated as directed at the bottom of the panel. Any buttons which can be modified are displayed in green. They are:



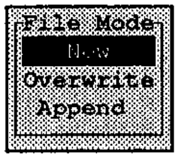
Designates the filename. Valid filenames can include any valid DOS Drive/path combination. The default is the current drive and path followed by the filename *DAS16.DAT*.



Selects the format in which data is to be saved. Choose **ASCII** or **BINARY.** Binary is the default selection.

Data which is saved in ASCII is saved according to channel number. A sample file is shown below:

```
1432        1234        1254        1234
1423        1232        1232        1232
1433        1233        1240        1233
 .           .           .           .
 .           .           .           .
 .           .           .           .
```



Selects the mode in which data is written to the file. Specify one of the following:

NEW - Indicates that the file is a new file and that data will be stored starting at the beginning of the file. This is the default selection.

OVERWRITE - Indicates that the file is an existing file and that the new data will overwrite any existing data.

APPEND - Indicates that the file is an existing file and that new data will be added to the end of the previously stored data.

3.3 TOUR THROUGH THE POP-UP MENU

As described in the previous section, the Pop-Up Menu is comprised of three separate panels, each of which consists of a series of controls and displays of various relevant information. The panels are grouped by function, as follows:

- Panel 0 - Analog I/O
- Panel 1 - Digital I/O
- Panel 2 - File Control

You can use either a mouse or the keyboard to manipulate these controls.

To use the Keyboard, press Alt M. Wait until the **Keyboard** option on the Panel Menu is highlighted, then press Enter. (Or, if you do have a mouse, point to the **Keyboard** option on the Panel Menu and click.) The help screen section of the Pop-Up Panel will display instructions telling you the appropriate keys to be used.

To use a mouse, just point and click on the desired control to activate the next allowed setting for that control.

Example: Clicking on the INT option of **Trig** changes it to **EXT**. To return it to **INT**, click on **EXT**.

In general, clicking the left mouse key displays and activates the next valid setting for that control. Using the right mouse key displays and activates the previous setting.

Example: Clicking the left mouse button on the **Bin** option of the **Units** Control changes it to **V**. Clicking the right mouse button on the **V** enclosed in the **Units** Control changes it to **Bin**. Repeated clicking with the left mouse key cycles through the other allowable settings.

Other controls are described below. These instructions assume that you are using a mouse.

Frequency and Wave Output Rate Selection

The Frequency and Wave Rate buttons are controlled as follows. Click on the digits to "dial-in" the desired frequency. Use the Left Mouse key for the next number and the Right Mouse button for the previous -- notice the blinking ▮ symbol to the right. Once you arrive at the desired frequency, click on the ▮ symbol to lock-in the dialed rate.

☞ ***NOTE:*** The selected frequency may not be achievable by the Timer/Counter arrangement on the DAS-16 hardware. Therefore, the Pop-Up program may adjust your frequency selection to the nearest achievable rate. This effect is only noticeable at higher rates.

Example: Select 1 5 0 0 0 for the Frequency and click the ▮. Notice the Frequency value changes to 1 5 0 1 5. (10Mhz Source Clock assumed!)

For rates greater than 2KHz (200 Hz for the WaveRate) you may use the ↑ and ↓ controls to index to the next achievable frequency.

Channel Scan Limit Selections

The data area on the Analog I/O panel is arranged depending on the position of the A/D Config (8, 16) switch on the actual DAS-16 hardware. Based on this switch, eight or 16 channels are displayed. In either case, the Channel-Gain-Value numbers are displayed as follows:

```
Ch-Gn------
0  1  xxxxx
```

The **Ch** number is used to select the Start and Stop channels. The Gain (**Gn**) can be selected for only the DAS-16G1 and G2 only. The actual A/D value is for display only. The current scan limit is shown as all channels with the A/D values highlighted with actual A/D values or larger X's. Channels that are not part of the current scan are displayed as small x's.

To select new channel scan limits, choose one or two channel numbers. These are now the start and stop channels which establish the new scan limits to be used during the next A/D operation. This results in the current A/D data being displayed as invalid data (large, highlighted X's).

Analog Input Gain Selection

Gn (gain) selection is only relevant for the DAS-16G1/G2. Click on the gain number next to the desired channel to change it. The allowed gain selection are hardware dependant and as follows:

DAS-16G1	1, 10, 100, 500
DAS-16G2	1, 2, 4, 8

The type of board is selected when the *MDAS16* driver is loaded. This is done using the /DAS= or /F= optional command line switches. However, if neither is specified, a DAS-16G2 is assumed. Of the four Acquisition types, only the Synchronous type allows for the selection of different gains for different channels. Selecting a new Gain value makes the current A/D data invalid which is reflected on the display by highlighted X's for A/D values. On the DAS-16 and 16F, the DIP-switch selectable gain is entered into the *MDAS16* driver at load-time using the /G= or /F= command line switches.

Controlling an A/D operation

Once you have setup all the desired parameters, Click on the **TrigAD** menu item to initiate an A/D acquisition of 1000 samples to an internal buffer. Normally a blinking ✻ will appear next to **TrigAD** indicating that data acquisition is in progress. During acquisition, all panel controls are disabled for input except the Main Menu bar, the DAC and DACValue buttons. Click on TrigAD ✻ to cancel data acquisition; otherwise, the blinking ✻ will disappear at the end of data acquisition and all panel controls are re-enabled for input. When errors occur during acquisition, the operation is terminated and an error message is displayed within the panel describing that error.

You can easily generate an INTERRUPT OVERRUN error by proceeding as follows:

- Select Int clock and 10000 Hz frequency.
- Select Interrupt Acq. Type
- Click on TrigAD

The error message is displayed. To proceed, click within the error message box or if you are using Keyboard mode, press Alt M.

When you click on **TrigAD** to start an acquisition with Trigger set for External, the **EXT** label will begin blinking until an external trigger signal (The Digital Input Line 1 - IP1 goes high.) is received. When the external trigger condition is satisfied, **EXT** stops blinking and data acquisition begins (within 55 ms of detecting the trigger.) If this signal never arrives, click on **TrigAD** again to cancel the operation and check your external trigger source; no data is acquired.

When you click on TrigAD to start an acquisition with Conversions Clock source set for External, **EXT** Clk label will begin blinking to indicate that the DAS-16 is waiting for signal on IP0/TRG0 or receiving signal on IP0/TRG0. To cancel the operation, click on **TrigAD**.

(This page intentionally left blank.)

CHAPTER 4
FILE I/O COMMANDS

The File I/O command set allows you to control the DAS-16 from any language that supports standard File I/O. This chapter describes how to implement the commands from other languages and describes each of the commands in detail. Examples for using the File I/O commands from programs written in PASCAL, C, and FORTRAN are given in the optional software package PCF-16G.

4.1 USING THE FILE I/O COMMANDS IN BASIC

Preparing the Environment

Opening the DAS-16 driver's I/O devices uses the majority of BASIC's limited number of file handles (3). If three file handles is insufficient for your program's requirements, invoke the BASIC interpreter with the `BASICA /F:num` option. This will allocate data space for the additional file opens. Consult your BASIC manual for the correct value of `num`.

Opening the Device Driver

One example of opening the device driver is shown below:

```
010  CLS
020  ON ERROR GOTO 330
030  OPEN "$MDAS16" FOR OUTPUT AS #1
040  PRINT #1, "CLEAR"
050  OPEN "$MDAS16" FOR INPUT AS #2
.
.
.
```

I/O Operations

Once the files have been opened properly, the BASIC `PRINT` or `INPUT` command is used to pass through the File I/O Command. The `PRINT` command is used to pass commands to the DAS-16 and the `INPUT` command is used to read data from the DAS-16.

Trapping Errors

Refer to *BASPOPUP.BAS* for an example of how to write an error-handler.

4.2 USING THE FILE I/O COMMANDS IN QUICKBASIC

Versions Supported

QuickBASIC 4.0 and higher

Opening the Device Driver

```
010  CLS
020  ON ERROR GOTO 330
030  OPEN "$MDAS16" FOR OUTPUT AS #1
040  PRINT #1, "CLEAR"
050  OPEN "$MDAS16" FOR INPUT AS #2
.
.
.
```

I/O Operations

Once the files have been opened properly, the QuickBASIC `PRINT` or `INPUT` command is used to pass through the File I/O Command. The `PRINT` command is used to pass commands to the DAS-16 and the `INPUT` command is used to read data from the DAS-16.

Trapping Errors

Refer to *QB_POPUP.BAS* for an example of how to write an error-handler.

4.3 USING THE FILE I/O COMMANDS IN MICROSOFT C

Versions Supported

4.0 and higher

Opening the Device Driver

```
FILE *DAS16;
DAS16 = fopen( "$MDAS16", "r+");
```

Remember to execute a `rewind` command between an input and output or an output and input command to the DAS-16. In addition, remember to issue a `fflush ()` command after the `fprint()` to ensure that the command is flushed from DOS' buffer.

Use the `fopen()` function with the `r+` file mode parameter.

The following shows an example of opening the driver:

```
FILE *DAS16;
char ErrMess1[200], ErrMess2[200]
char ErrNum[20];
```

```
main();
{
if ((DAS16 = fopen("$MDAS16","r+"))==NULL)
     printf("Could not find device DAS-16");
if (!(fputs("Set FREQ 500",DAS16)GetError();
if (fflush(DAS16 == EOF) Get Error();
...
...
...
}
/* retrieve error number and error message from
device*/
void  GetError()
{

rewind(VI);

fgets(ErrNum,10,VI):    /* get error number */

fgets(ErrMess1,80,VI); /* get error message 1 */

fgets(ErrMess2,80,VI); /* get error message 2 */

rewind(VI);

printf("ERROR Number %s", ErrNum);

printf("ERROR - %s", ErrMess2);

printf("On command line of ...%s", ErrMess1);
exit(0);
}
```

I/O Operations

1. In Versions 5.0 and higher, call the `rewind()` function before switching between read and write operations.
2. Strings are OUTPUT using the `fprintf()` function.
3. Use `fgets()` to INPUT strings.
4. After each `fprintf()` be sure to call `fflush()`. This will flush the output buffer and write an output string immediately. Otherwise, the output strings will be buffered and the device may fail due to overflow.

Trapping Errors

1. When `fflush()` is called, an EOF will be returned if a write error occurred.

2. The driver can return any of three diagnostic error strings. These are terminated by a CR,LF or Newline sequence.

3. Note that the `fscanf()` function will not execute correctly due to the whitespace (blanks, line-feeds) contained in the returned diagnostic error strings. Use `fgets()` instead. This function treats the CR,LF sequence as string delimiters.

4.4 USING THE FILE I/O COMMANDS IN TURBO C

Versions Supported All

Opening the Device Driver

1. Use the `fopen()` function with the `r+` file mode parameter.

I/O Operations

1. Don't use `rewind()` to switch from Input to Output mode. Instead, use `fclose()` and then re-open the file; otherwise, use Microsoft C.

2. Strings are OUTPUT using the `fprintf()` function.

3. Use the `fgets()` to INPUT strings.

4. After each `fprintf()` be sure to call `fflush()`. This will flush the output buffer and write an output string immediately. Otherwise, the output strings will be buffered and the device may fail due to overflow.

Trapping Errors

1. When `fflush()` is called, an EOF will be returned if a write error occurred.

2. The driver can return any of three diagnostic error strings. These are terminated by a CR,LF or a newline sequence.

3. Note that the `fscanf()` function will not execute correctly due to the whitespace (blanks, line-feeds) contained in the returned diagnostic error strings. Use `fgets()` instead. This function treats the CR,LF sequence as string delimiters.

4.5 USING THE FILE I/O COMMANDS IN MICROSOFT PASCAL

Versions Supported

Versions 3.0., 3.3, and 4.0

Opening the Device Driver

1. File I/O to and from the driver is accomplished via the intrinsic PASCAL type TEXT. This type of file handle allows files to be opened for input and output at the same time. Thus, only one file variable, `x` should be ASSIGNed for both input and output.
2. Set the MODE field for file `x` to DIRECT.
3. Declare file `x` globally as `VAR X : Text(127);`.

I/O Operations

1. Use `Writeln` to write commands to the device rather than `write`.
2. If an error occurs after an I/O operation has been executed, the ERRS field for file `x` will be non-zero.
3. After an error occurs, the ERRS field for file `x` must be cleared.
4. To detect runtime errors, the TRAP field for file `x` must be set.
5. You must use a `SEEK(X,1)` to rewind the file before any read or write is performed.

Trapping Errors

1. The driver can return any of three diagnostic error strings. These are terminated by a CR,LF.
2. Use `Readln` to read diagnostic strings, not read.
3. You must use a `SEEK(X,1)` to rewind the file before any read or write!is performed.

4.6 USING THE FILE I/O COMMANDS IN TURBO PASCAL

Versions Supported

Version 4.0 and higher

Opening the Device Driver

1. One file variable should be ASSIGNed for input and another for output.
2. `REWRITE` the OUTPUT file variable.

I/O Operations

1. Use `Writeln` to write commands to the device rather than write.
2. If an error occurs after an I/O operation has been executed, the `IOResult` will be non-zero.

Trapping Errors

1. The driver can return any of three diagnostic error strings. These are terminated by a CR,LF.
2. Use `Readln` to read diagnostic strings, not read.

Examples

4.7 NOTE REGARDING POP-UP MENU SUPPORT

In its support of all the *MDAS16.EXE* graphical characteristics, the *VI.EXE* driver prevents any screen writes by non-DAS-16 programs to the DAS-16 screen area. *VI.EXE* performs this function by intercepting all INT 10H calls and blocking those that would effect unwanted screen writes to the DAS-16 screen area. Thus, any program that uses DOS or INT 10Hs to display information can work, but it cannot disturb the DAS-16 screen area.

However, *VI.EXE* cannot assure total immunity to the DAS-16 screen area. If a non-DAS-16 program writes directly to display memory, it can overwrite the screen. In this situation, the DAS-16 screen area can be recovered by doing a pop-down followed by a pop-up (Press the Pop-Up Menu hot key(s) twice). Functionally, the instrument itself will be unaffected by any such overwrite.

Most BASIC programs will present no problems, although some BASICs write directly to display memory when scrolling the screen up. In this case, a user could do a menu pop-down before the scroll.

Microsoft QuickC will not change to the user's screen unless there has been a write to the screen. If the user programs the instrument to pop-up (using the SHOW command) before any normal screen I/O occurs, the instrument will pop-up over Quick C's screen and not the user's.

4.8 THE FILE I/O COMMANDS

This section describes the File I/O Commands and their usage.

4.8.1 File I/O Command Usage

The File I/O Commands must be used according to the following rules:

1. Multiple commands may appear on one command line.

2. The commands are not case-sensitive. They may be in upper case, lower case, or mixed.

3. Incorrect spellings are unacceptable and can lead to no action, wrong action, and/or an error condition.

4.8.2 File I/O Command Description Format

Each command description is divided into several sections as described below.

Description Provides a brief description of the command.

Syntax Describes the syntax required by each command. The commands are implemented as described in sections 4.2 through 4.7. The following rules are used in this section:

Command names and input parameters which are mandatory appear in **`this typeface`**.

All of the command names and some of the input parameters have abbreviated versions. These abbreviations (or keywords) are specified in `UPPER CASE`. The remaining optional letters appear in `lower case`.

Parameters which are optional appear in *`this typeface`*.

Square Brackets ([ ]) enclose a list of optional parameters. You may select any combination of these options as described in the parameters section. Note that each option is separated by a comma (,). **DO NOT include these commas or the Square Brackets.**

Parameters Describes the input parameters.

Returns Describes any data which will be returned after execution of the command has been completed. Your application must read these responses. An Error Code will be returned if an error has occurred.

Programming Notes Describes any special programming considerations for the File I/O command.

Example Provides an example of valid File I/O Command Syntax. This is independent of the programming language.

4.8.3 The File I/O Command Set

This section describes all of the File I/O Commands.

CLEAR

Description	This command clears all data currently prepared by the driver for input to your program during the next input operation. It is recommended that this command be the first File I/O command sent to the driver.
Syntax	`CLEAR`
Parameters	None.
Returns	An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	None.
Example	`CL` Clears all data.

HIDE

Description	Hides the Pop-Up Panel.
Syntax	`HIde`
Parameters	None.
Returns	An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	1. This command will be ignored if the *VITASK.EXE* driver is loaded. See Chapter 3.
Example	`hi` — Removes the Pop-Up Panel from the display.

LOCK

Description	"Locks" in the operating parameters specified on the Pop-Up panel and prevents them from being edited by the mouse or keyboard. This command is useful when the Pop-Up is used for File I/O program monitoring.
Syntax	`LOck`
Parameters	None.
Returns	An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	None.
Example	`lo` Locks the Pop-Up Panel.

READ AD

Description Returns specified information about an A/D operation. You may specify a status, count, address, total number of samples to be acquired, or the value of a specific conversion.

Syntax `REad AD [STatus, COunt, SEgment, OFfset, SAmples, VAlue I]`

Parameters

`STATUS`	Returns the status of the A/D conversion.
`COUNT`	Returns the current conversion count. This command is useful for monitoring the progress of the A/D sample transfers, and should be used for Interrupt and DMA data acquisition modes only.
`SEGMENT`	Returns the segment value in decimal of the internal data acquisition buffer. This command is useful for languages such as C that allow easy and direct access to absolute memory locations.
`OFFSET`	Returns the offset value in decimal of the internal data acquisition buffer. This command is useful for languages such as C that allow easy and direct access to absolute memory locations.
`SAMPLES`	Returns the current number of samples that were acquired. This number is 1000 by default and may overridden by using the `SET AD` command.
`VALUE I`	Returns the specified A/D value. This value is located at position `I` within the internal data buffer. `I` is the sample index within this buffer, starting at 0. If `I` is not specified, the first data sample is returned. This command is useful for languages without convenient methods for accessing direct memory location.

Returns Returns will vary according to the type of data requested as follows:

`STATUS`	Returns a 0 if Idle, a 1 if Active, a 2 if waiting for an external event to occur (e.g. a trigger or clock), or other if an error has occurred during acquisition. Reading the status when an error has occurred clears that error.

READ AD (cont'd)

`COUNT`	Returns the current conversion count while an A/D operation is in progress; otherwise, a zero is returned.
`SEGMENT`	Returns the segment value in decimal of the internal data acquisition buffer.
`OFFSET`	Returns the offset value in decimal of the internal data acquisition buffer.
`SAMPLES`	Returns the current number of samples to be acquired.
`VALUE I`	Returns the specified A/D value. This value is located at position `I` within the internal data buffer or the first sample.

An Error Code if an error has occurred. The error codes are further described in Appendix B.

Programming Notes None.

Example

`read ad value`	Returns the value of the first sample.
`RE AD VAL 7`	Returns the value of the sample located at position 7 in the buffer.
`re ad co`	Returns the number of conversions which have been performed.
`read ad status`	Returns the conversion status and/or clears an acquisition error.
`re ad segment`	Returns the segment address of the buffer.
`READ AD SAMPles`	Returns the total number of samples to be acquired.

READ ADTYPE

Description	Returns the current A/D operation type.
Syntax	`READ ADType`
Parameters	None.
Returns	Returns a number reflecting the current A/D operation type as follows:

0	Interrupt mode
1	DMA mode
2	Synchronous mode
3	Scan mode

An Error Code if an error has occurred. The error codes are further described in Appendix B.

Programming Notes The interrupt, scan, and DMA data acquisition modes require that a valid interrupt level has been set via the `SET INTERRUPT` command.

Example

`read adT`	Reads the current type of A/D operation.
`READ ADTYPE`	Reads the current type of A/D operation.

READ BASE

Description	Returns the base address of the DAS-16.
Syntax	`REad BAse`
Parameters	None.
Returns	A string representing the board's base address in decimal. This number will be within the range of 512 to 1008. An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	None.
Example	`re ba` Returns the base address.

READ CLOCK

Description	This command returns the current clock source.
Syntax	`REad CLOck`
Parameters	None.
Returns	Will return a 0 (Internal Clock) or 1 (External Clock). An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	None.
Example	`RE CLO` Reads the clock source.

READ CONFIG

Description	Returns an integer representing the input range configuration.
Syntax	`REad COnfig`
Parameters	None.
Returns	Returns a 0 for Bipolar Configuration or 1 for Unipolar Configuration. This value reflects the A/D switch position at load-time of the driver. An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	The slide switch position is read by the MDAS16 driver at load-time. If this switch is changed, the driver must be unloaded and then re-loaded.
Example	`re config` Returns an ASCII 1 or 0.

READ DAC

Description	Returns a value indicating the channel to be used for DAC output.
Syntax	`REad DAC`
Parameters	None.
Returns	A number indicating which channel is to be used as follows: 0 = Channel 0 1 = Channel 1 2 = Both Channel 0 and 1 An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	None.
Example	`read DAC` — Returns the current channel used for DAC output operations.

READ DACVALUE

Description	Returns the current DAC value in binary code (0-4095), regardless of the current Pop-Up display unit setting.
Syntax	`REad DACValue`
Parameters	None.
Returns	An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	The Driver's DMA Level is changed at driver load-time through the board's configuration file (`/F=`). Note that the level specified through this file must match the actual DMA level switch position on the board.
Example	`re dacv` Returns the current DAC value.

READ DMA

Description	Returns an integer representing the DMA level configuration.
Syntax	`REad DMA`
Parameters	None.
Returns	Returns a 1 or 3 representing the DMA Level. An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	None.
Example	`read dma` Will return a 1 or 3.

READ FILE

Description	Returns a string describing the filename to which data is being saved, the format in which data is being saved, or the mode in which data is written to the file.
Syntax	`REad FIle [NAme, FOrmat, MOde]`
Parameters	`NAME` Returns the name of the file to which data is being saved.
	`FORMAT` Returns the format in which data is being saved. If a 0 is returned, the data is being saved in Binary format. If a 1 is returned, the data is being saved in ASCII format.
	`MODE` Returns the mode in which data is written to the file, where: 0 = NEW mode 1 = OVERWRITE mode 2 = APPEND mode See the SET FILE command for a description of each of these modes.
Returns	An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	None.
Example	`re fi na` Returns the name of the file to which data is currently being saved.

READ FREQUENCY

Description	This command returns the actual frequency of the internal clock.
Syntax	`REad FRequency`
Parameters	None.
Returns	Returns the actual frequency in Hertz of the internal clock. An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	None.
Example	`Read Freq` — Returns the current internal frequency in Hertz.

READ GAIN

Description Returns the gain setting for the specified channel.

Syntax `REad GAin C`

Parameters C Specifies the channel. It must fall within the range of 0 and 7 for Differential input or 0 and 15 for Single Ended input configuration.

Returns The gain code of the selected channel, where:

GAIN CODE	DAS-16G1 GAIN	DAS-16G2 GAIN
0	1	1
1	10	2
2	100	4
3	500	8

An Error Code if an error has occurred. The error codes are further described in Appendix B.

Programming Notes None.

Example `Read Gain 7` Returns the current gain code (0-3) of channel 7.

READ INPUT

Description	Returns the value present at the DAS-16 input port.
Syntax	`REad INput`
Parameters	None.
Returns	This command returns a decimal number between 0 and 15 in ASCII format representing the value currently present at the DAS-16 input port. The value returned must be converted from string to integer and AND'ed with the value 15 (F hex) to arrive at the actual 4-bit input value. An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	None.
Example	`RE inp` — This will read the input from the DAS-16 input port.

READ INTERRUPT

Description	This command returns the current interrupt level setting.
Syntax	`REad INTerrupt`
Parameters	None.
Returns	Returns an ASCII number representing the current interrupt level setting. This number will range from 2 to 7. An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	None.
Example	`read interrupt`

READ MAXCHANNELS

Description	Returns the maximum number of channels available.
Syntax	`REad MAxchannels`
Parameters	None.
Returns	Returns an 8 (for differential configurations) or 16 (for single-ended configurations). This value reflects the slide switch position at load-time of the driver. An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	The slide switch position is read by the MDAS16 driver at load-time. If this switch is changed, the driver must be unloaded and then re-loaded.
Example	`re ma` Will return an 8 or 16.

READ START/STOP

Description	Returns the beginning/ending analog input channels.	
Syntax	`REad [STArt, STOp]`	
Parameters	`START`	Designates that the "start" (beginning) channel is to be read back.
	`STOP`	Designates that the "stop" (ending) channel is being read back.
Returns	An ASCII number representing the channel. This is a number between 0 and 7 for differential mode or 0 and 15 for single-ended mode.	
	An Error Code if an error has occurred. The error codes are further described in Appendix B.	
Programming Notes	None.	
Example	`Read Start`	Returns the start channel number.

READ TIMER

Description	Returns the timer jumper configuration.	
Syntax	`REad TImer`	
Parameters	None.	
Returns	Returns a 0 if the timer jumper is set for 1 MHz or 1 if set for 10 MHz. An Error Code if an error has occurred. The error codes are further described in Appendix B.	
Programming Notes	None.	
Example	`read ti`	Will return a 0 or 1 depending on the timer jumper setting.

READ TRIGGER

Description Returns the trigger mode, channel, slope, or gain selected by the last SET TRIGGER.

Syntax `REad TRigger [CHannel, LEvel, SLope, GAin]`

Parameters

`CHANNEL`	<u>This is an optional parameter which can be specified only if the board is in external analog trigger mode.</u> If it is included, the trigger channel setting will be returned, as was set by the last SET TRIG command. See SET TRIGGER for more information.
`LEVEL`	<u>This is an optional parameter which can be specified only if the board is in external analog trigger mode.</u> If it is included, the trigger level will be returned, as was set by the last SET TRIG command. See SET TRIGGER for more information.
`SLOPE`	<u>This is an optional parameter which can be specified only if the board is in external analog trigger mode.</u> If it is included, the trigger slope setting will be returned, as was set by the last SET TRIG command. See SET TRIGGER for more information.
`GAIN`	<u>This is an optional parameter which can be specified only if the board is in external analog trigger mode.</u> If it is included, the trigger gain setting will be returned, as was set by the last SET TRIG command. See SET TRIGGER for more information.

Returns If `READ TRIGGER` is specified, one of the following ASCII values will be returned:

0 =	Internal trigger mode
1 =	External digital trigger mode
2 =	External analog trigger mode

If `READ TRIGGER` is followed by one of the optional parameters, the value of that parameter is returned.

READ TRIGGER (cont'd)

An Error Code if an error has occurred. The error codes are further described in Appendix B.

Programming Notes

None.

Example

`read trigger`	Returns 0, 1 or 2.
`trig ext 0 1024 + 1`	Sets for external analog trigger.
`read trig level`	Returns 1024, the Analog level.
`re tr ch`	Returns 0, the Analog input channel number.

READ UNITS

Description	This command returns the current Pop-Up Unit setting.
Syntax	`REad UNits`
Parameters	None.
Returns	Returns a value representing the current Pop-Up unit setting as follows: 0 = for Binary 1 = for Volts 2 = for mVolts An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	None.
Example	`Read units` Returns the current units setting.

READ WAVE

Description This command returns the square wave output status or rate.

Syntax `REad WAve [STAtus, RAte]`

Parameters

`STATUS`	Specifies that the square wave output status is to be returned.
`RATE`	Specifies that the rate of square wave output is to be returned.

Returns Returns will vary according to the type of data requested as follows:

`STATUS` Returns an ASCII 0, 1 or 2 where:

0 = Square wave output is idle
1 = Square wave output is active
2 = Square wave output is pending IP2 being high.

`RATE` Returns an ASCII string representing the actual wave rate in KHertz.

An Error Code if an error has occurred. The error codes are further described in Appendix B.

Programming Notes None.

Example

`re wave status`	Returns the current square wave output status.
`READ WAVE RATE`	Returns the current square wave output rate.

SAVE

Description	If the current A/D data stored in the internal buffer is good, then this data is written to the user specified file in the selected format (See the `SET FILE` command).
Syntax	`SAve`
Parameters	None.
Returns	An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	None.
Example	`SA` — Writes data from internal buffer to the designated file.

SET AD

Description

Start/Stop an A/D operation. Optionally, you can specify the number of conversions to be performed and the destination buffer to be used for storage. If these options are not specified, the default 1000 conversions will be performed and the data will be stored in the internal buffer.

Syntax

SEt **AD** *X SEG OFFSET*

Parameters

X	is an optional variable which represents the number of conversions. **X** can range from 1 to 32767.
SEG	is an optional variable representing the segment portion of the destination buffer's address. Note that in some languages such as BASICA, the segment portion of the address is not available. In this case, specify -1. If **SEG** is specified, you must also specify **OFFSET**
OFFSET	is an optional variable representing the offset portion of the destination buffer's address. You must specify this parameter if **SEG** is specified.

Returns

An Error Code if an error has occurred. The error codes are further described in Appendix B.

Programming Notes

1. Before starting an A/D operation, make sure that the current A/D status is "Idle". (Use the **REad AD STatus** command.) Otherwise, issue an AD command to stop the A/D operation already in progress.

2. The address of the internal buffer is determined using the **READ AD [seg, offset]** commands.

SET AD (cont'd)

Example

`AD`	Stop A/D operation if already in progress, or Start A/D operation for 1000 conversions to internal buffer
`AD 10000`	Stop A/D operation if already in progress, or Start A/D operation for 10000 conversions to internal buffer.
`AD 10000 25545 0`	Stop A/D operation if already in progress, or Start A/D operation for 10000 conversions to user buffer at address 25545:0. (63C9:0 hex)
`AD 10000 -1 VARPTR(A%)`	Stop A/D operation if already in progress, or Start A/D operation for 10000 conversions to user buffer.

SET ADTYPE

Description

Sets the type of the next A/D acquisition. You may select between interrupt, DMA, synchronous, and scan modes. All methods will work with both Internal and External Clock and Trigger sources; however, the maximum transfer rates differ according to the mode selected. For all modes except scan, the acquisition process continues until one of the following occurs:

1. the desired number of samples is reached
2. an error condition occurs
3. the acquisition is interrupted by the user.

Syntax

`SEt ADType [INT, DMA, SYNC, SCAN]`

Parameters

`INT`	This selects the interrupt mode. This mode allows data acquisition and transfer using one of five (5) user programmable interrupt levels. When a conversion is completed, the DAS-16 issues an interrupt which the software detects and then reads the data value into memory. Because of the involvement of the software (and the CPU) to read the data into memory, the maximum conversion rate is limited to approximately 5KHz. Only one gain for all of the channels is used in this mode. This mode operates completely in the background, which means you can perform most other functions of the board while it is in progress.
`DMA`	This selects DMA mode. DMA mode allows for data to be transferred from the board directly to PC memory without the involvement of the CPU or any driver software. Because of this, the maximum conversion rates as rated for your DAS-16 board are achievable (70KHz to 100KHz). Only one gain for all channels is used in this mode. Like the INT mode, DMA operates completely in the background.
`SYNC`	This selects the synchronous mode. This mode, unlike the interrupt and DMA modes, operates in the foreground. That is, once an A/D operation is initiated (via the SET AD Command), no other board function may be performed until the

SET ADTYPE

		operation terminates. The advantage of this mode is that it allows the use of different gains with different channels during acquisition. The maximum conversion throughput for this mode is machine-dependent (i.e., it depends on the PC you are using, its CPU speed, bus speed, etc.).
	`SCAN`	Scan mode allows for continuous sampling and display of the selected channels. Only the number of channels worth of data are acquired per scan and the Pop-Up display is updated with the latest data twice per second.

Returns An Error Code if an error has occurred. The error codes are further described in Appendix B.

Programming Notes None.

Example `Set ADType DMA` Sets the acquisition type to DMA for the next A/D operation.

SET CLOCK

Description Sets the A/D conversion clock source to internal or external.

Syntax `SEt CLOck [INTernal, EXternal]`

Parameters

`INTERNAL`	This parameter specifies that the on-board 8254 Programmable Interval Timer is to be the clock source. The Programmable Timer is set to the value specified by the last **SET FREQUENCY** command or the POP-UP's *FREQUENCY* button.
`EXTERNAL`	This parameter specifies that the internal frequency value is ignored and that the external signal detected on IP0/TRG0 (pin 25) is used to cause A/D and D/A conversions.

Returns An Error Code if an error has occurred. The error codes are further described in Appendix B.

Programming Notes None.

Example

`SEt CLOck INT`	Sets the conversion clock source as internal.
`CLO EXT`	Sets the conversion clock as external.

SET DAC

Description	Selects the DAC channel to use for the next DAC output operation.	
Syntax	`SEt DAC x`	
Parameters	**x**	is a variable which indicates the channel is to be used. **x** can be 0, 1, or 2, where: 0 = Channel 0 1 = Channel 1 2 = Both Channel 0 and 1
Returns	An Error Code if an error has occurred. The error codes are further described in Appendix B.	
Programming Notes	None.	
Example	`SET DAC 1`	Selects DAC output channel 1.
	`DAC 2`	Selects DAC output channel 2.

SET DACVALUE

Description	Specifies the DAC value to output for the next DAC output operation.	
Syntax	`SEt DACValue V`	
Parameters	`v`	is a variable which represents the DAC value. This value is always specified as a value between 0 and 4095.
Returns	An Error Code if an error has occurred. The error codes are further described in Appendix B.	
Programming Notes	None.	
Example	`Set DACV 2048`	Sets the DAC output value to 2048.
	`DACV 1024`	Sets the DAC output value to 1024.

SET FILE

Description

This command is used to select the data file to which the A/D data is to be saved (via the **SAVE** command). You can also specify the format in which the data is saved.

Syntax

```
SEt FIle NAme F
SEt FIle FOrmat [ASci1, BInary]
SEt FIle MOde [NEw, OVerwrite, APpend]
```

Parameters

NAME	Indicates that the filename is to be selected. **NAME** is followed by the **F** parameter described below. **F** designates the filename. Valid filenames can include any valid DOS Drive/path combination.
FORMAT	Indicates that the file format is to be selected. **FORMAT** is followed by either the **ASCII** or **BINARY** parameters. The parameters are described below: **ASCII** - Saves the data in ASCII format. (Section 3.2 gives you an example of what an ASCII file looks like.) **BINARY** - Saves the data in binary format.
MODE	Selects the mode in which data is written to the file. **MODE** is followed by the **NEW, OVERWRITE**, or **APPEND** parameters. The parameters are described below: **NEW** - Indicates that the file is a new file and that data will be stored starting at the beginning of the file. **OVERWRITE** - Indicates that the file is an existing file and that the new data will overwrite any existing data. **APPEND** - Indicates that the file is an existing file and that new data will be added to the end of the previously stored data.

Returns

An Error Code if an error has occurred. The error codes are further described in Appendix B.

SET FILE

Programming Notes None.

Example

`SE FI FORMAT ASCII`	Specifies that data is to be stored in ASCII format.
`fi mode new`	Specifies that the file is a new file and data is to be stored at the beginning of the file.
`fi name c:\das16g\data1.dat`	Specifies that data is to be stored to the *data1.dat* file located in the `das16g` directory on the `c` drive.

SET FREQUENCY

Description When Internal Conversion clock source is selected, this command sets the internal clock frequency.

Syntax `SEt FRequency R`

Parameters

`R`	is a variable which selects the frequency. The value associated with the variable must be within the range 1 to 99999 Hertz.

Returns An Error Code if an error has occurred. The error codes are further described in Appendix B.

Programming Notes

1. The internal clock is generated by cascading two 16-bit counters of the Intel 8254 Timer chip; and the frequency specified here is used to generate the two counter dividers for programming the timer chip. Note that because of the limitations of the 8254, the desired frequency may not be attainable; however the driver selects the counter divider values to obtain the nearest possible value.

2. This command should be followed by a `READ FREQUENCY` command to determine the actual frequency programmed by the driver.

Example

`Set Frequency 2000`	Sets the internal clock frequency to 2KHz.
`Freq 15000`	Sets internal clock frequency to 15015 Hz.
`Re Freq`	Returns 15015

SET GAIN

Description Sets the specified analog input channel to the given gain code. This command can only be used with the DAS-16G1 and DAS-16G2.

Syntax `SEt GAin C G`

Parameters

C — Specifies the channel. It must fall within the range of 0 and 7 for differential input or 0 and 15 for single-ended input configuration.

G — Sets the gain for the specified channel on the DAS-16G1/G2. Valid gain codes are 0 to 3, where:

GAIN CODE	DAS-16G1 GAIN	DAS-16G2 GAIN
0	1	1
1	10	2
2	100	4
3	500	8

Returns An Error Code if an error has occurred. The error codes are further described in Appendix B.

Programming Notes

1. You can only assign different gains to different channels if Synchronous Data Acquisition is used. (See the Set AD command description.) Otherwise, all selected channels will be assigned the specified gain; i.e., the channel number is ignored.

Example

`Set Gain 0 1` — Sets channel 0 for gain code 1 for Sync. Acq. Type; otherwise, sets gain code 1 for all selected channels.

SET INPUT

Description	Triggers a digital input operation from IP0 thru IP3.
Syntax	`SEt INPut`
Parameters	None.
Returns	An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	1. This command must be followed by a `READ INPUT` command to determine the value currently latched on the DAS-16 input port.
Example	`Input`

SET INITIALIZE

Description	Initializes the DAS-16 hardware and software driver.
Syntax	`SEt INItialize`
Parameters	None.
Returns	An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	None.
Example	`Se initialize` Initializes the DAS-16 hardware.

SET INTERRUPT

Description	Sets the interrupt level to be used for the next A/D acquisition. This interrupt level is required by the interrupt, scan, and DMA data acquisition modes.
Syntax	`SEt INTerrupt [2, 3, 4, 5, 6, 7]`
Parameters	`2 3 4 5 6 7` Designate the interrupt level to be used. Valid interrupt levels are 2 through 7. Select one interrupt level.
Returns	An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	None.
Example	`Set Int 5` Selects to use interrupt level 5.

SET OUTPUT

Description	Triggers the output of the lower nibble (least-significant 4 bits) of the specified value on OP0 thru OP3.
Syntax	`SEt OUtput X`
Parameters	**X** — is the value. The value to output may be specified in decimal, binary (preceded by &B), octal (preceded by &O), or hexadecimal (preceded by &H). The value must fall within the range 0 to 15 decimal (0 to F hex).
Returns	An Error Code if an error has occurred. The error codes are further described in Appendix B.
Programming Notes	None.
Example	`Out 5` — Output the binary value 0101.
	`ou &B1010` — Output the binary value 1010.

SET START/STOP CHANNEL

Description Sets the beginning/ending analog input channels to sample during the next A/D operation.

Syntax `SEt [STArt, STOp] C`

Parameters

`START`	Designates that the "start" (beginning) channel is being specified.
`STOP`	Designates that the "stop" (ending) channel is being specified.
`C`	Represents the channel number. It must fall within the range of 0 and 7 for differential input or 0 and 15 for single-ended input configuration.

☞ ***NOTE:*** If you are unsure of the configuration, use the `REad MAxchannels` command or view the number of channels displayed on the Pop-Up Analog I/O panel.

Returns An Error Code if an error has occurred. The error codes are further described in Appendix B.

Programming Notes The A/D operation is initiated using the AD command or using the TrigA/D button from the Pop-Up Analog I/O panel .

Example

`Start 0 Stop 10`	Sets the scan limit from channel 0 to 10 inclusive.

SET TRIGGER

Description Sets the start-conversions (A/D and D/A) trigger source. It can be set for internal software trigger, or external digital or analog triggers.

Syntax `SEt TRigger [INTernal, EXternal CHANNEL LEVEL SLOPE GAIN]`

Parameters

INTERNAL	Indicates that an internal software trigger is to be used.
EXTERNAL	Indicates that an External Trigger is to be used. If specifying an external analog trigger, you must also provide the **CH**, **LEVEL**, and **SLOPE** parameters. If you are programming a DAS-16G1/G2, you must also provide the **GAIN** parameter.

CH - This parameter is only required if an External Analog Trigger is specified. It specifies the channel number to be used. It may be one of the scanned channels within the scan limits or a separate channel, outside of those limits, used only for triggering. Valid channel number range from 0-7 (differential configuration) or 0-15 (single-ended configuration).

LEVEL - This parameter is only required if an External Analog Trigger is specified. It selects the voltage level at which triggering occurs. Valid ranges are -2047 to +2048 for bipolar input ranges and 0 to 4095 for unipolar ranges.

SLOPE - This parameter is only required if an External Analog Trigger is specified. It sets the direction of triggering. Use + to indicate that triggering should occur on a positive slope (when the signal exceeds the trigger level) and - to indicate that triggering should occur on a negative slope (when the signal becomes less than the trigger level).

SET TRIGGER (cont'd)

GAIN - This parameter is only required if an External Analog Trigger is specified and you are programming a DAS-16G1/G2. It sets the gain for the DAS-16G1/G2. Valid gain codes are 0 to 3, where:

GAIN CODE	DAS-16G1 GAIN	DAS-16G2 GAIN
0	1	1
1	10	2
2	100	4
3	500	8

Returns An Error Code if an error has occurred. The error codes are further described in Appendix B.

Programming Notes None.

Example

`Set Trigger Internal`	Setup for internal trigger
`Set Trig External`	Setup for external digital trigger. Conversions begin when a low-to-high transition is detected on digital input line 0 (IP0/TRG0 - pin 25)
`Trig ext 1 1024 + 0`	Setup for external analog trigger. Conversions begin when A/D channel 1 reaches level 1024, positive slope, and gain code of 0.

SET UNITS

Description	Sets the A/D and D/A value(s) units of the Pop-Up Display.	
Syntax	`SEt UNits [BInary, VOlt, MVolt]`	
Parameters	`BINARY`	Selects binary code as the unit of measure. Valid ranges are -2047 to +2048 for bipolar input ranges and 0 to 4095 for unipolar ranges.
	`VOLT`	Selects volts as the unit of measure.
	`MVOLT`	Selects millivolts as the unit of measure.
Returns	An Error Code if an error has occurred. The error codes are further described in Appendix B.	
Programming Notes	None.	
Example	`Set Units VO`	Sets the units to volts.
	`UN BI`	Sets the units to binary.

SET WAVE

Description	This command controls the square wave output on counter 0.
Syntax	`SEt WAve [STArt, STOp, RAte r]`

Parameters

`START`	Begins square wave output.
`STOP`	Ends square wave output.
`RATE R`	Selects the rate of square wave output. `R` is the rate and must be a value between 2 and 25000 Hertz.

Returns

An Error Code if an error has occurred. The error codes are further described in Appendix B.

Programming Notes

1. Note that the square wave output is Gated by Digital Input line 2 (IP2 - PIN 2) being high.
2. Note that the desired rate may not be attainable. The driver selects a counter divider to obtain the nearest possible value.

Example

`wave start`	Start square wave output.
`wa sto`	Stop square wave output.

SHOW

Description Displays the specified portion of the Pop-Up Panel. If no argument is specified, the Analog I/O panel is shown.

Syntax `SHow [0, 1, 2, AIo, DIo, FIle]`

Parameters

`0`	is an optional parameter, which specifies that the Analog I/O Panel is to be displayed.
`1`	is an optional parameter, which specifies that the Digital I/O Panel is to be displayed.
`2`	is an optional parameter, which specifies that the File Panel is to be displayed.
`AIO`	is an optional parameter, which specifies that the Analog I/O Panel is to be displayed.
`DIO`	is an optional parameter, which specifies that the Digital I/O Panel is to be displayed.
`FILE`	is an optional parameter, which specifies that the File Panel is to be displayed.

Returns An Error Code if an error has occurred. The error codes are further described in Appendix B.

Programming Notes This command will be ignored if the *VITASK.EXE* driver is loaded. See Chapter 3.

Example

`SHOW 1`	Displays the Analog I/O Panel.
`sh fi`	Displays the File Panel.

TRANSFER

Description Transfers the specified number of A/D samples from the internal buffer to an array.

Syntax `TRAnsfer SEG OFFSET COUNT INDEX`

Parameters

`SEG`	Represents the segment portion of the destination buffer's address. Note that in some languages such as BASICA, the segment portion of the address is not available. In this case, specify -1.
`OFFSET`	Represents the offset portion of the destination buffer's address.
`COUNT`	Represents the number of samples to be transferred.
`INDEX`	Represents the position of first sample to be transferred. Positions start at 0.

Returns An Error Code if an error has occurred. The error codes are further described in Appendix B.

Programming Notes Note that no range checking on any of the supplied parameters is performed by the driver. You must be sure that the data transferred with this command is within the bounds of the buffer.

Example Refer to the QuickBASIC example on the distribution disks.

UNLOCK

Description	"Unlocks" in the operating parameters specified on the Pop-Up panel and allows them to be edited.	
Syntax	`UNLock`	
Parameters	None.	
Returns	An Error Code if an error has occurred. The error codes are further described in Appendix B.	
Programming Notes	None.	
Example	`unl`	Unlocks the Pop-Up Panel.

CHAPTER 5
PROGRAMMING THE DAS-16 VIA THE CALL MODES

In addition to the Pop-Up and the File I/O command, there is another programming alternative - the call modes. These may be accessed from BASIC by a single line call statement. The various modes of the call routine select all of the functions of the DAS-16, format and error-check data, and perform frequently used sequences of instructions.

There are two drivers provided to support the CALL modes. If you have previously purchased a DAS-16/16F (software version 4.01 or less) and written programs for it, the *DASG.BIN* driver is provided. Otherwise, use the *DAS16G.BIN* driver.

5.1 PROGRAMMING IN BASICA

5.1.1 Loading The Machine Language Call Routine DASG.BIN

There are two methods for loading the *DASG.BIN* driver. The option you choose will depend on the amount of RAM memory installed in your machine. For most applications, Method 1 will suffice. This procedure requires a minimum of 256K RAM. If your machine does not have this much RAM, use Method 2.

Method 1

This method involves loading the driver outside of the BASIC workspace using the BLOAD command. To do this, you must select a segment of memory which has 6 KBytes clear at the beginning and that does not conflict with any other program or data area. For example, you could choose &H6000. Then proceed as follows:

```
xxx10   DEF SEG = &H6000          'Sets up load segment
xxx20   BLOAD "DASG.BIN",0        'Loads at 6000:0000
xxx30   DAS16 = 0
xxx40   DIM PARAM%(15)
...
xxxxx   DEF SEG = &H6000
xxxyy   CALL DASG (MD%, PARAM%(0), FLAG%)
xxxzz   etc.
```

An example of this method in given in *EX3.BAS*.

Method 2

This method involves loading the driver within the BASIC workspace using the BLOAD command. To do this, you must select a segment of memory which has 6 KBytes clear at the beginning and that does not conflict with any other program or data area. (If you do interfere with another program's use of memory, the CALL routine will not work and your PC will most likely "hang-up". (If this happens, reboot your computer by pressing Ctrl Alt Del.) To determine a safe area to load it, proceed as follows.

1. Determine the size of the BASIC workspace. By nature of its design, the maximum memory segment that BASIC is able to use is 64K. To determine the size of the workspace, at the DOS prompt type:

   ```
   BASIC(A)
   ```

   The computer will respond with something similar to :

   ```
   BASIC
   The IBM Personal Computer Basic
   Version D1.10 Copyright IBM Corp.  1981, 1982
   61807 Bytes Free
   ```

   The exact number of "Bytes Free" varies with the version of BASIC(A) and DOS but is usually greater than 60KB. A number less than 60K indicates that your PC's memory is already heavily used. If this is true, you will have to load the CALL routine by further contraction of the BASIC workspace and by loading the routine at the end of the newly defined workspace.

   You will need to 6K (6144 byte) space for the *DASG.BIN* driver. To do this, first determine how much memory BASIC is able to use. Then, load BASIC(A) from DOS with the command:

   ```
   BASIC(A)
   ```

   Note the number of Bytes Free in BASIC's greeting message. Now, use a SYSTEM command to return to DOS and reload BASIC(A) with the optional /M parameter, as follows:

   ```
   BASIC(A) /M:WS
   ```

   Try setting the WS (workspace) parameter to 30000 or 40000 and then note the number of *Bytes Free*. Continue this process, increasing the workspace parameter until the

Bytes Free number is reduced by at least 6144 bytes. Then, you can either load BASIC(A) by specifying this workspace or include a `CLEAR` command right at the beginning of your program, as follows:

```
xxx10 CLEAR, WS
```

2. Identify the segment BASIC occupies in memory. In all versions of Microsoft derived BASIC, you can determine the segment from the contents of absolute memory locations &H511 and &H510. These locations hold the current BASIC segment, which we can call SG. Determine SG as follows:

```
xxx20     DEF SEG = 0
'define current segment = 0000 before reading
'absolute addresses 0000:0510 & 0000:0511
xxx30     SG = 256*PEEK(&H511) + PEEK(&H510)
```

The segment address at which the CALL routine can be loaded will be at the end of the working space. For example:

```
xxx40    SG = WS/16 + SG             'remember segment addresses
                                     'are on 16-byte boundaries
```

Load the routine as follows:

```
xxx50    DEF SEG = SG
xxx60    BLOAD "DASG.BIN",0          'loads routine at SG:0000
```

A BLOAD must be used as you are loading a binary (machine language) program. Once loaded, the CALL can be entered as many times as needed in the program after initializing the call parameters MD%, PARAM%, FLAG%. Enter these parameters prior to the CALL sequence as follows:

```
xxx70    DEF SEG = SG
xxx80    DAS16 = 0
xxx90    CALL DASG (MD%, PARAM%(0), FLAG%)
```

Note that `DASG` is a variable that specifies the memory offset of the starting address of the CALL routine from the current segment, as defined in the most recent preceding `DEF SEG` statement.

Notes

1. DASG is the offset (actually zero) from the current segment, as defined by the last DEF SEG statement (DEF SEG tells your BASIC interpreter where the CALL routine is located). Avoid inadvertently redefining the current segment somewhere in a program before entering the CALL. It is good practice to immediately precede the CALL statement by the appropriate DEF SEG statement (the same one you preceded your BLOAD with).

2. CLEAR sets working space from the bottom of the BASIC working area up, whereas you must set aside space for your subroutine from the top of memory down. Any attempt to CLEAR more space than is actually available will load your routine over the end of the BASIC program, data space, and stack and will hang up the computer. Be especially careful this does not happen if you are memory-limited and later load BASIC with DEBUG or some coresident program without declaring a corresponding reduction in workspace (WS) in your CLEAR statement.

5.1.2 Format Of The Call Statement

If you are new at using CALL statements, this section will help you to understand the use of CALL statements. Prior to entering the CALL, the DEF SEG = SG statement sets the segment address at which the CALL subroutine is located. The CALL statement for the DASG driver is of the format:

```
xxx     CALL DASG (MD%, PARAM%(0), FLAG%)
```

Where:

DASG	= The address offset from the current segment of memory as defined in the last DEF SEG statement.
MD%	= Call parameter representing the Mode Number.
PARAM%	= Call parameter(s) representing the Data Variable(s).
FLAG%	= Call parameter representing Errors.

In executing the CALL, the addresses of the variables (pointers) are passed in the sequence written to BASIC's stack. The CALL routine unloads these pointers from the stack and uses them to locate the variables in BASIC's data space so data can be exchanged with them. There are several important rules to remember when using the CALL statement:

1. **The CALL parameters must always be written in the correct order.** The subroutine does not recognize the names of the variables, just their locations . For example, if the line `xxxxx CALL DASG (PARAM%(0), MD%, FLAG%)` is used, the CALL routine would interpret `PARAM%(0)` as the MODE number, `MD%` as the data, etc.

2. **All parameters must be defined as integer type variables.** The CALL does not perform any error checking on the variable type. If you use the wrong variable type, the CALL function will not perform correctly.

3. **Do not perform any arithmetic functions within the parameter list brackets of the CALL statement.** For example, `CALL DASG (MD% + 2, PARAM%(0) * 8, FLAG%)` is illegal and will produce a syntax error.

4. **Do not use constants for any of the parameters.** For example, this is illegal: `CALL DASG (7, 2, FLAG%)`.

5. **You can assign any name to the variables that you want.**

6. **You should declare all variables before executing the CALL.** If you do not, the simple variables will be declared by default on execution, but array variables cannot be dimensioned by default and must be dimensioned before the CALL to pass data correctly if used as a CALL parameter. Most MODEs of the CALL routine require multiple items of data to be passed in an array. For this reason, `PARAM%(0)` is specified as the data variable so that the CALL routine can locate the whole array from the position of its initial element.

   Likewise, any of the other CALL parameters may be integer array variables if required, and you can name any number of different integer data arrays for output and input. It is permissible to dimension arrays with more elements than will be used by the CALL. Unused elements will be unchanged and for example could be used for tagging data with time, date, or other information.

5.1.3 Execution Times - Compiled BASIC

The execution times of most modes of the DAS-16 are limited by the software. Additionally, other operations in your program which process the data may also delay your overall throughput. One solution which would improve the speed of your program is to use compiled BASIC or QuickBASIC. See section 5.2 for more information.

5.2 PROGRAMMING IN QUICKBASIC

This section contains information useful to users wishing to write data acquisition programs in QuickBASIC (QB). In addition to the information provided in this section, you may want to consult the QuickBASIC example programs on the Distribution Disks.

5.2.1 Loading The Program

A QuickBASIC program will have to make calls to an external driver/library. Supplied with your DAS-16 are the following linkable driver/libraries:

DASG.QLB	Load this Quick Library into your QuickBASIC Integrated Environment Version 4.5 or lower.
DASGx.QLB	Load this Quick Library into your QuickBASIC Extended Environment Version 7.0 .
DASG.LIB	LINK this library to your stand-alone QuickBASIC program.

Load the *DASG.QLB* Quick Library into the QB environment from the DOS command line using the `/L` switch:

```
QB /L DASG
```

To load an application program (such as *EXG3.BAS*) along with the Quick Library, you load the Quick Library and *EXG3.BAS* together by typing:

```
QB /L DASG EXG3
```

Use *DASGx.QLB* in an identical manner when using the QuickBASIC Extended Environment (QBX) Version 7.0.

5.2.2 Declaring The Driver

Before you can use the driver/library, you must declare the CALL label to make it known to your application. You make this declaration by inserting the following at the beginning of your program:

```
DECLARE SUB BASDASG (MODE%, BYVAL dummy%, FLAG%)
```

where, BASDASG is the common entry point to the driver/library for driver modes.

☞ ***NOTE:*** All subroutine DECLARE's in your program MUST be before any $DYNAMIC arrays are allocated. $DYNAMIC data is data that is allocated space in the FAR heap, outside the default data segment. All arrays used for data acquisition must be declared as $DYNAMIC; QuickBASIC assumes $STATIC data (Default data segment) unless otherwise specified.

5.2.3 Format of the Call Statement

Unlike in BASICA, in QB the first and third parameters are passed as variables while the second parameter is passed as a pointer. This arrangement is necessary because the second parameter must represent the offset of the command integer array. To pass the actual offset, use the VARPTR function as follows:

```
CALL BASDASG(MD%, VARPTR(PARAM%(0)), FLAG%)
```

where

MD% is the Mode number
PARAM%(N) is the parameter array, and FLAG% returns detected errors.

The VARPTR function returns the address of PARAM%(0), which you pass as a value to the driver. The driver uses that value as a pointer to the first element of our command integer array PARAM%.

PARAM% is declared as a $STATIC array since it must reside in the default data segment in order to be relative to the QB data segment, as required by the driver. Since PARAM% is passed to and returned by the external driver, declare this parameter as an inter-module global variable, as follows:

```
...
DIM PARAM%(16)
COMMON SHARED PARAM%()
...
```

The COMMON makes this variable visible between modules and the SHARED statement at the module level makes it known globally in this module. After declaring all your $STATIC variable you can declare any large $DYNAMIC arrays for DMA data acquisition.

```
...
REM $DYNAMIC
DIM DMA%(10000)
...
```

☞ ***NOTE:*** All $DYNAMIC data declaration must occur AFTER all COMMON and DECLARE statements in your program. If you get the QB error `COMMON and DECLARE must precede all executable statements` double check the order of DECLARE's COMMON's and $DYNAMIC declarations.

See Section 2.5 ($STATIC and $DYNAMIC Arrays) and (COMMON statement) in the QuickBASIC Language Reference Manual for a detailed discussion of the factors that determine array types.

To summarize, your program header should look as follows:

```
...
DECLARE SUB BASDASG (MODE%, BYVAL dummy%, FLAG%)
...
DIM PARAM%(16)
COMMON SHARED PARAM%()       ' Parameter Array for mode CALLs
...
REM $DYNAMIC                ' 
DIM DMA%(10000)             ' DMA array in FAR Heap
...
```

Refer to the supplied QB example programs for more detail. Note that the address of the array `DMA%()` is a "FAR" address (32-bits)and may be determined by using the built-in operators VARSEG and VARPTR. Your DAS-16 driver/library (.LIB or .QLB) is designed with provisions to accept 32-bit addresses. With these provisions, you can avoid specifying absolute addresses as in BASIC:

```
PARAM%(1) = 5000H
```

Instead, when passing FAR pointers to QB, specify -1 where you normally specify the "memory segment" and supply the FAR pointer's segment and offset in unused `PARAM%()` parameter array elements. The following is an example of how this is accomplished for the DMA mode 6. Assume that the array `DMA%(10000)` is previously declared in the $DY-NAMIC area as described above:

```
...
PARAM%(0)  = 10000             'Sample count
PARAM%(1)  = -1                'flag to look for Seg:ofs below!
PARAM%(2)  = 1                 'internal clock
PARAM%(3)  = 0                 'single cycle
PARAM%(4)  = VARPTR(DMA%(0))   'Offset of DMA%(0)
PARAM%(5)  = VARSEG(DMA%(0))   'Segment of DMA%(0)
...
```

A final note on arrays. If it is desirable to erase and redimension an array during program execution, it should be declared as `$DYNAMIC`. The `REDIM`statement can now change the size, but not the dimensional structure. The `ERRASE` statement is not necessary. Note that if a page wrap occurs, you should use MODE 23 to transfer the data.

5.2.4 Making Executable Programs

There are two ways to create a stand-alone QuickBASIC program that is executable from the DOS command line:

- From within the QB (QBX) environment
- From the DOS command line.

When making a stand-alone executable, you need the *DASG.LIB* driver/library.

From within the QB Environment

To make an executable from the QB Environment, use the following sequence:

1. Invoke the environment, as follows:

   ```
   QB /L DASG QBEXG3           ' for QB up Ver 4.5
   ```

   or

   ```
   QBX /L DASGx QBEXG3         ' for QB Ver 7.0
   ```

2. Select the **Run** menu item.

3. Select the **Make EXE File...** from the **Run** menu.

4. Select **Produce: (•) Stand-Alone EXE File.**

5. Select the **<Make EXE and Exit>** option.

This sequence produces a stand-alone *QBEXG3.EXE* that does not require run-time support. As an alternative, you can use this procedure:

1. Select the **Run** menu item.

2. Select the **Make EXE File...** from the **Run** menu.

3. Select **Produce: (•) EXE Requiring BRUN45.EXE.**

4. Select the **<Make EXE and Exit>** option.

This sequence produces a program *QBEXG3.EXE* that requires the run-time support program *BRUN45.EXE*. To run either type of executable program, type: `QBEXG3`.

From the DOS Command Line

To compile QB programs from the DOS command line, use the following command sequence from DOS:

```
BC /e /o QBEXG3.BAS;
LINK QBEXG3,,,DASG.LIB;
```

The `/o` option given in the compiler line, causes references to the *BCOM45.LIB* library to be placed in the object module, so the library response need not be given in the LINK line. This sequence will produce a stand-alone executable.

As an alternative, type:

```
BC QBEXG3.BAS;
LINK QBEXG3,,,DASG.LIB;
```

The absence of the `/o` option in the compiler line causes references to the *BRUN45.LIB* library to be placed in the object module, so the library response need not be given in the LINK line. This sequence produces an executable program that requires *BRUN45.EXE* to be in a subdirectory named \BIN or in the current directory at the time the program is executed.

Note that the Compiler and Linker expect to find the necessary executables (*BC.EXE, LINK.EXE*, etc..) in a subdirectory named \BIN, and to find Libraries in the directory named by the environment variable `LIB` (`SET LIB=` in your *AUTOEXEC.BAT*.)

To run either type of executable program, type `QBEXG3`.

5.3 CALL MODE DESCRIPTIONS

This section provides descriptions of all the CALL MODES. A brief listing of the available modes is given in Table 5-1.

Table 5-1. Call Modes

Mode	Description
0	Initialize. Initializes the DAS-16 with appropriate operating parameters.
1	Sets the multiplexer scan limits to values other than the defaults.
2	Read multiplexer current channel and scan limit setting.
3	Perform a single A/D conversion. Return data and increment multiplexer address. (Programmed conversion). Speed - slow, Operation - foreground.
4	Perform specified number of conversions after trigger and transfer data via interrupt into an integer array. Speed - medium, Operation - foreground.
5	Perform specified number of conversions and transfer data via interrupt into a memory buffer. Speed - medium, Operation - background.
6	Perform specified number of conversions and transfer data via DMA into a memory buffer. Speed - fast, Operation - background.
7	Terminate DMA/Interrupt operation of MODEs 5, 6, 18, or 20.
8	Report status of DMA/Interrupt operation initiated by MODE 5, 6, 18, or 20.
9	Transfer data from buffer memory area to BASIC integer array.
10	Set Counter 0 operating configuration.

Table 5-1. Call Modes

Mode	Description
11	Load Counter 0 data.
12	Read Counter 0.
13	Output to digital outputs OP0-3.
14	Read digital inputs IP0-3.
15	Output data to a single D/A channel.
16	Output data to both D/A channels.
17	Set programmable timer rate.
18	D/A waveform output & A/D input.
19	Analog trigger function.
20	A/D block channel scan on interrupt.
21	Set channel gains (DAS-16G only).
22	Output a square wave.
23	Allocate DMA buffer.
24	Free DMA buffer.

MODE 0 - INITIALIZE

Description

This mode initializes the DAS-16 by:

- Designating the Base Address, Interrupt Level, and DMA channel to be used.
- Setting default scan limits of Channels 0 & 7 if the DAS-16 is in 8-channel differential-input configuration, or limits of 0 & 15 are set if in 16-channel single-ended configuration.
- Clearing the Control and Timer-Counter-Enable registers. This disables all interrupt, DMA, and external trigger functions.
- Configures the Programmable Interval Timer and Counters 1 and 2 for Rate Generator MODE and set to produce a 1KHz (10KHz with 10MHz clock) pulse rate.
- Performs a simple read/write test to check for the presence of the DAS-16 hardware.

Input Variables

The following input variables must be initialized prior to calling MODE 0:

`MD% =`	0
`PARAM%(0) =`	**The Base Address of the DAS-16.** Base Addresses can range from 200 to 3F0 hex. Be sure when specifying a hex base address to precede the address with `&H`, i.e.: &H300 is 300 hex (768 decimal). The Base Address you provided must match the setting of the Base Address Switch. (See Chapter 2.)
`PARAM%(1) =`	**The Interrupt Level.** Valid interrupt levels are 2 thru 7.
`PARAM%(2) =`	**The DMA Level.** This value may be 1 or 3. The default DMA level is 3. The DMA Level specified must match the setting of the DMA Level Switch. (See Chapter 2.)

MODE 0 - INITIALIZE (cont'd)

Returns

`MD%`	=	0
`PARAM%(0-15)`	=	Unchanged.
`FLAG%`	=	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

Programming Notes

1. <u>This mode must be called prior to any other mode</u>; otherwise no other MODEs can execute. MODE 0 needs to only be called once in the initialization of your program--usually immediately after loading *DASG.BIN*. See *EXG3.BAS* for an example.

2. Note that the interrupt will not be enabled unless you enter a MODE which requires interrupts for operation. If you are not going to make use of interrupts, any level can be chosen; for example, `PARAM%(1) = 2`.

Example

An example of the use of MODE 0 is provided in *EXG3.BAS*.

MODE 1 - SET SCAN LIMITS

Description

This mode sets the scan limits of the multiplexer to other than the default limits provided by MODE 0.

Input Variables

The following input variables must be initialized prior to calling MODE 1:

`MD% =`	1
`PARAM%(0) =`	**The Lower Scan Limit.** For a differential configuration, this value can be from 0 to 7. For single-ended configurations, this value can range from 0 to 15.
`PARAM%(1) =`	**The Upper Scan Limit.** For a differential configuration, this value can be from 0 to 7. For single-ended configurations, this value can range from 0 to 15.

Returns

`MD% =`	1
`PARAM%(0-15)=`	Unchanged.
`FLAG% =`	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

Programming Notes

1. MODE 1 should be called prior to performing conversions in MODES 3, 4, 5, 6, 18 or 20 – unless the default limits of MODE 0 (0-7 for differential, 0-15 for singled-ended) are correct for your application.

2. To illustrate the action of MODE 1, assume we set PARAM%(0) = 3 and PARAM%(1) = 6. The first conversion (commanded by MODEs 3, 4, 5, 6, 18, or 20) is performed on Channel 3; data is returned and the channel is incremented to 4. The next conversion is performed on Channel 4, data returned, and the channel incremented to 5, etc. After the conversion on Channel 6 has been performed, the counter controlling the mux will automatically re-load to the start of scan limit to repeat the sequence. Scanning of channels is always stepped between and including Channels 3 and 6, for example: 3-4-5-6-3-4-5-6-3- etc.

MODE 1 - SET SCAN LIMITS (cont'd)

3. It is possible to have the lower limit greater than the higher limit. For example, if PARAM%(0) = 13 and PARAM%(1) = 2, the scan sequence is:

   13-14-15-0-1-2-13-14-15-0-1-2-13-etc.

☞ ***NOTE:*** If the board is operating in the 8-channel differential mode, setting the lower limit greater than the upper limit should be avoided, as the counter hardware will attempt to cycle through Channels 8-15.

4. If you wish to perform continuous conversions on one channel, then set the low and high limits equal to each other and the desired channel number. For example, setting PARAM%(0) = 1 and PARAM%(1) = 1 would perform continuous conversions on Channel 1. Note that after exit from MODE 1, the starting channel will always be PARAM%(0), the lower limit.

Example An example of the use of MODE 1 is provided in *EXG3.BAS*.

MODE 2 - READ CURRENT MULTIPLEXER ADDRESS AND SCAN LIMITS

Description This mode lets you determine what the current multiplexer channel setting and scan limits are.

Input Variables The following input variables must be initialized prior to calling MODE 2:

`MD%` = 2.

Returns

`MD%` = 2.

`PARAM%(0)` = **Channel address for next conversion.** This will be an integer from 0 to 15.

`PARAM%(1)` = **Lower scan limit.** This will be an integer from 0 to 15.

`PARAM%(2)` = **Upper scan limit.** This will be an integer from 0 to 15.

`PARAM%(3)` = **Current gain register setting.** For the DAS-16/16F, a -1 will be returned (indicating that the value is not available). For the DAS-16G1/G2, any of the following may be returned:

GAIN PARAM%(3)	DAS-16G1	DAS-16G2
0	1	1
1	10	2
2	100	4
3	500	8

`FLAG%` = If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

MODE 2 - READ CURRENT MULTIPLEXER ADDRESS AND SCAN LIMITS (cont'd)

Programming Notes

1. If MODE 2 is run during the short 8-12 microsecond interval when the A/D is busy and the multiplexer address is possibly being incremented, it will wait until the A/D has finished converting. In this way, the multiplexer address returned always corresponds to the next channel on which a conversion will take place.

Example An example of MODE 2 usage is found in *EXG3.BAS.*

MODE 3 - DO ONE A/D CONVERSION AND INCREMENT THE MULTIPLEXER

Description This mode performs one A/D conversion by software command. The multiplexer is automatically incremented after the conversion by the hardware.

Input Variables The following input variables must be initialized prior to calling MODE 3:

`MD% =` 3.

Returns

`MD% =` **The MODE number.** Unchanged

`PARAM%(0) =` **A/D data.** This value will range between 0 and 4095 if Unipolar or between -2048 and +2047 if Bipolar.

`PARAM%(1) =` **Channel Number.** This variable will contain an integer representing the channel number from which data is acquired. This can range from 0 to 7 , in a differential configuration or 0 to 15 in a single-ended configuration.

`PARAM%(2) =` **Current gain register setting.** For the DAS-16/16F, a -1 will be returned (indicating the value is not available). For the DAS-16G1/G2, any of the following may be returned:

GAIN PARAM%(2)	DAS-16G1	DAS-16G2
0	1	1
1	10	2
2	100	4
3	500	8

MODE 3 - DO ONE A/D CONVERSION AND INCREMENT THE MULTIPLEXER (cont'd)

`PARAM%(3-15) =`	Unchanged.
`FLAG% =`	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

Programming Notes

1. Data is transferred to PARAM%(0) in standard integer form (2's complement form). If the Unipolar/Bipolar switch is in the UNIPOLAR position, data returned is in the range 0 - 4095 bits and is directly transferred. If the switch is in the BIPOLAR position, then 2048 is subtracted giving a data range of -2048 to +2047 bits. Hence, zero volts always corresponds to zero bits. This saves further processing of the data after exit from the CALL.

2. The A/D will perform conversions on channels in accordance with the scan limit conditions set in MODE 1. If MODE 1 has not been entered prior to MODE 3, conversions will cycle between Channel 0 and Channel 7 (differential) or between Channel 0 and Channel 15 (single-ended).

3. Note that MODE 3 is the only MODE that allows you to perform an A/D conversion initiated by the program (i.e. software command). It is fairly slow since it is limited by the program execution speed.

Example An example of the use of MODE 3 will be found in *EXG3.BAS*.

MODE 4 - PERFORM SPECIFIED NUMBER OF A/D CONVERSIONS & TRANSFER DIRECTLY TO AN ARRAY

Description This mode performs the specified number of A/D conversions and transfers data directly into a BASIC integer array.

Input Variables The following input variables must be initialized prior to calling MODE 4:

`MD% =`	4.
`PARAM%(0) =`	**Number of conversions required (Word count).** This is an integer in the range 1 to *N* where *N-1* is less than or equal to the array dimension (up to a maximum of 32767).
`PARAM%(1) = VARPTR(ARRAY%(M)`	**Defines the array pointer.** *M* represents the start position in the array (*ARRAY%*) where conversions to be loaded. If *M* = 0, conversion will be loaded starting at the beginning of the array.
`PARAM%(2) =`	**Trigger source.** If `PARAM%(2) = 0`, then an external trigger input is used. Conversions will take place on positive transitions on the IP0/TRG0 input and continue until the word count is reached.

☞ ***NOTE:*** Exit from the CALL cannot take place until pulses equal to word count have been supplied.

If `PARAM%(2) = 1`, then a Programmable interval timer will be used. In this case, IP0/TRG0 should be held low until you want to start conversions. After IP0/TRG0 goes high, IP0/TRG0 will have no further effect. Exit occurs when the word count is reached.

MODE 4 - PERFORM SPECIFIED NUMBER OF A/D CONVERSIONS & TRANSFER DIRECTLY TO AN ARRAY (cont'd)

Returns

`MD% =`	The MODE number.
`PARAM%(0-15) =`	Unchanged.
`ARRAY%(M) =`	First data word.
`ARRAY%(M+1) =`	Second data word.
`ARRAY%(M+2) =`	Third data word.
`...`	
`etc.`	

> ***NOTE:*** Note that only the data is stored in the array not the channel number. If you want to keep the channel number, use MODE 5 or 6.

`FLAG% =` If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

QuickBASIC Programming Notes

1. Larger arrays can be specified from QuickBASIC than from BASIC(A) by declaring them as $DYNAMIC type. This type of array is placed in the FAR data heap and located via Segment and Offset address values. (Unlike BASIC which just requires an offset). MODE 4 allows you to specify a FAR Address (32-bits) to the destination user data array. To specify a FAR address, the following modifications apply to the normal BASIC Mode Call setup:

```
PARAM%(1) = -1      'i.e. Seg:Offset are in PARAM%(3)
                    'and (4)
PARAM%(2) = 0 or 1    'Usual trigger source
PARAM%(3) = VARPTR (ARRAY%(M))     'Address Offset
PARAM%(4) = VARSEG (ARRAY%(M))     'Address Segment
```

MODE 4 - PERFORM SPECIFIED NUMBER OF A/D CONVERSIONS & TRANSFER DIRECTLY TO AN ARRAY (cont'd)

Where, ARRAY%() is earlier declared as follows:

```
REM $DYNAMIC
DIM ARRAY%(10000)
'DIM of 10000 used for demonstration only.
```

Refer to the program *QBEXG4.BAS* for usage.

☞ ***NOTE:*** When programming from the QuickBASIC environment, you cannot use MODE 23 for allocation of DMA buffer. This is because in this case, QB reserves all of the FAR Heap space for its internal use. This makes all calls to MODE 23 fail.

General Programming Notes

1. An exit from the CALL will not occur until all conversions have been completed. For an immediate return to your BASIC program, press any key on the keyboard while data is being gathered. If you do not want to wait for data to be collected, consider using MODE 5 or 6. Each of these gathers the data as a "background" operation, thus allowing your program to process and collect data simultaneously.

2. The A/D will perform conversions on channels in accordance with the scan limit conditions set in MODE 1. If MODE 1 has not been entered prior to MODE 4, conversions will start on channel 0 and the upper scan limit will be channel 7 (differential) or 15 (single-ended).

MODE 4 - PERFORM SPECIFIED NUMBER OF A/D CONVERSIONS & TRANSFER DIRECTLY TO AN ARRAY (cont'd)

3. The A/D may be triggered from two sources: the programmable interval timer or an external trigger pulse according to PARAM%(2). If the programmable interval timer is used, IP0 acts as a start gate to the operation. If an external trigger is used, trigger pulses are applied to IP0, positive edges start conversions. By connecting Counter 0 output to IP0, it is possible to produce a programmed delay before conversions are performed using the programmable interval timer.

4. Be sure to dimension a receiving array that has at least as many elements as the number of conversions specified in PARAM%(0). If more conversions are performed than the array will hold, some of the data area of BASIC will be overwritten. This may destroy descriptors and other variable data and cause strange effects as a result.

5. After assigning the pointer to the receiving array (by using VARPTR), do not introduce any new simple variables before entering the CALL. For example, the following is correct:

```
xxx10   DIM PARAM%(15)            'declare PARAM%(*)
xxx20   MD% = 4                   'MODE #
xxx30   PARAM%(0) = 1000   'number of conversions
xxx40   PARAM%(1) = VARPTR(X%(0))'pointer to array
xxx50   PARAM%(2) = 1             'trigger MODE
xxx60   CALL BASDASG (MD%, PARAM%(0), FLAG%)
...
```

6. Conversion rates in excess of 2000 samples/sec are attainable in this MODE. As interrupts in the computer (mainly the timer interrupt) may divert the CPU away from attending to transferring data from the A/D for several hundred microseconds, data may be lost above 3000 samples/sec.

7. The conversion rate is set by Mode 17.

Example An example of MODE 4's usage is found in *EX4.BAS.*

MODE 5 - PERFORM SPECIFIED NUMBER OF A/D CONVERSIONS & TRANSFER TO MEMORY ON INTERRUPT

Description

This mode will perform the specified number of conversions in background mode and returns control to the calling program. At the end of each conversion, an interrupt occurs which invokes an interrupt handler routine that transfers the data to a specified area of memory. The routine also keeps track of the total number of conversions performed, and will either terminate the transfer or continually transfer data to the same area of memory.

Input Variables

The following input variables must be initialized prior to calling MODE 5:

`MD% =`	5.
`PARAM%(0)=`	**Number of conversions required (Word count).** This is an integer in the range 0 to 32767. Specify 0 when the maximum number of conversions (32768) is desired.
`PARAM%(1)=`	**Segment of memory to receive data.** Use Mode 23 to obtain a suitable memory segment.
`PARAM%(2) =`	**Trigger source.** If `PARAM%(2) = 0`, then an external trigger input is used. Conversions will take place on positive transitions on the IP0/TRG0 input and continue until the word count is reached. If `PARAM%(2) = 1`, then the programmable interval timer will be used. In this case, IP0/TRG0 should be held low until you want to start conversions. After IP0/TRG0 goes high, IP0/TRG0 will have no further effect. Exit occurs when the word count is reached.

MODE 5 - PERFORM SPECIFIED NUMBER OF A/D CONVERSIONS & TRANSFER TO MEMORY ON INTERRUPT (cont'd)

`PARAM%(3) =` **Single Cycle/Continuous Operation.** If `PARAM%(3)=0`, then one cycle will be performed. This means that after the specified number of conversions are completed, interrupts will be disabled; thus, setting the operation status to zero. The interrupt handler will remain "active" until the specified number of conversions has been completed.

If `PARAM%(3)=1`, continuous operation is selected. This means that data will be continuously written to the same area of memory, where `PARAM%(0)` represents the memory "buffer" length. Data will continue to be written until MODE 7 is called.

Returns

`MD% =` 5.

`PARAM%(0-15) =` Unchanged.

`FLAG% =` If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

QuickBASIC Programming Notes

1. When using QuickBASIC, you may elect not to use MODE 23 to allocate memory for the acquisition buffer, especially when in the QB Integrated Environment where MODE 23 will not work. Instead, you can declare a $DYNAMIC array of any size and pass its Segment and Offset to Mode 5.

   To specify a FAR address to Mode 5, the following modifications apply to the normal BASIC Mode Call setup:

```
PARAM%(1) = -1
'i.e. Seg:Offset are in PARAM%(4) and (5)
.
.
.
PARAM%(4) = VARPTR (ARRAY%(0))    'Address Offset
PARAM%(5) = VARSEG (ARRAY%(0))    'Address Segment
.
.
.
```

MODE 5 - PERFORM SPECIFIED NUMBER OF A/D CONVERSIONS & TRANSFER TO MEMORY ON INTERRUPT (cont'd)

Where, ARRAY%() is earlier declared as follows:

```
REM $DYNAMIC
DIM ARRAY%(10000)
REM $STATIC
'DIM of 10000 used for demonstration only.
```

Programming Notes

1. Because the segment registers are not incremented by the interrupt handler, the maximum data area available is 64Kbyte (a page) for 32,768 conversions. Be certain that this data area is not used by your program or altered by subsequent operations.

2. Use Mode 8 to monitor the status and transfer progress.

3. Use Mode 9 to transfer data from memory to user array.

4. If a programmable timer is used to trigger the interrupt, exit from MODE 5 will not occur until IP0 has been taken high.

5. The scan limits set by the mode determine the number of conversions performed. Otherwise, if no scan limits are set, the conversions will begin at Channel 0 and continue to Channel 7 (for differential) or 15 (for single-ended).

6. The conversion rate is set by Mode 17.

Example An example of MODE 5 can be found in *EX5.BAS* or *QBEXG5.BAS*.

MODE 6 - PERFORM THE SPECIFIED NUMBER OF A/D CONVERSIONS AND TRANSFER DATA TO MEMORY VIA DMA

Description

This mode performs the specified number of A/D conversions and, at the end of each conversion, a direct memory transfer from the A/D to memory via DMA. (Conversions can be triggered either externally or by the programmable timer.)

Input Variables

The following input variables must be initialized prior to calling MODE 6:

`MD% =`	6
`PARAM%(0)=`	**Number of conversions required (Word count).** This is an integer in the range 0 to 32767. Specify 0 when the maximum number of conversions (32768) is desired.
`PARAM%(1)=`	**Segment of memory to receive data.** Use Mode 23 to obtain a suitable memory segment.
`PARAM%(2) =`	**Trigger source.** If `PARAM%(2) = 0`, then an external trigger input is used. Conversions will take place on positive transitions on the IP0/TRG0 input and continue until the word count is reached. If `PARAM%(2) = 1`, then a programmable interval timer will be used. In this case, IP0/TRG0 should be held low until you want to start conversions. After IP0/TRG0 goes high, IP0/TRG0 will have no further effect. Exit occurs when the word count is reached.
`PARAM%(3) =`	**Single Cycle/Continuous Operation.** If `PARAM%(3)=0`, then one cycle will be performed. This means that after the specified number of conversions are completed, DMA will cease and an interrupt will be generated. The DMA status will be set to zero. The interrupt handler will remain "active" until the specified number of conversions has been completed.

MODE 6 - PERFORM THE SPECIFIED NUMBER OF A/D CONVERSIONS AND TRANSFER DATA TO MEMORY VIA DMA (cont'd)

If `PARAM%(3)=1`, continuous operation is selected. This means that data will be continuously written to the same area of memory, where `PARAM%(0)` represents the memory "buffer" length. This corresponds to a DMA auto-initialize operation. To stop DMA, call MODE 7.

Returns	`MD%` =	6
	`PARAM%(0-15)` =	Unchanged.
	`FLAG%` =	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

QuickBASIC Programming Notes

1. In order for QuickBASIC users to utilize the new features supplied as Modes 23 and 24 (DAS-16 software Ver 4.20 or higher) you must develop your programs to run as Stand-alone EXE's. Refer to Mode 23 for more detail.

   Refer to the program *QBEXG23.BAS* for usage.

Programming Notes

1. DMA is performed by the system and DAS-16 hardware. It is a background operation and extremely fast. Throughput is limited by the speed of the A/D converter and settling of the sample hold. A technical description of the IBM PC DMA arrangement is contained in Appendix C.

2. There are other MODES which are associated with DMA. Use MODE 9 to return the data to an array. Use MODE 8 to assess the progress of DMA operation initiated by MODE 6. To abort a DMA operation initiated by MODE 6, use MODE 7.

3. The A/D will perform conversions on channels in accordance with the scan limit conditions set in MODE 1. If MODE 1 has not been entered prior to MODE 6, conversions will start on Channel 0 and the upper scan limit will be Channel 7 (differential) or 15 (single-ended).

MODE 6 - PERFORM THE SPECIFIED NUMBER OF A/D CONVERSIONS AND TRANSFER DATA TO MEMORY VIA DMA (cont'd)

4. The A/D may be triggered from two sources: the programmable interval timer or an external trigger pulse according to PARAM%(2). If the programmable interval timer is used, then IPO acts as a start gate to the operation. If an external trigger is used, trigger pulses are applied to IP0, positive edges start conversions. By connecting Counter 0 output to IP0, it is possible to produce a programmed delay before conversions are performed using the programmable interval timer.

Example For an example of MODE 6 usage, see *EXG6.BAS*.

MODE 7 - TERMINATE DMA/INTERRUPT OPERATION

Description	This mode causes an immediate termination of any running interrupt or DMA operation. These can be initiated by MODES 5, 6, 18, or 20. Note that MODE 7 is not required for the normal termination of MODES 5, 6, 18, and 20.
Input Variables	The following input variables must be initialized prior to calling MODE 7:

	`MD% =`	7.
Returns	`MD% =`	7.
	`PARAM%(0-15) =`	Unchanged.
	`FLAG% =`	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

Programming Notes	None.
Example	For an example of MODE 7 usage, see *EXG5.BAS, EXG6.BAS, EXG18.BAS,* or *EXG20.BAS.*

MODE 8 - READ DMA/INTERRUPT STATUS

Description

This mode allows you to monitor the status of a background operation initiated by MODEs 5, 6, 18, or 20.

Input Variables

The following input variables must be initialized prior to calling MODE 8:

`MD% =`	8.

Returns

`MD% =`	8.
`PARAM%(0) =`	**Operation type.** This will be an integer from 0 to 4 and is interpreted as follows:
`PARAM%(1) =`	**Status of Operation.** If a 0 is returned, the operation has been completed. If a 1 is returned the operation is still in progress (active).
`PARAM%(2) =`	**Current Word Count.** The integer returned defines the number of conversions performed up to that point.
`PARAM%(3-15) =`	Unchanged.
`FLAG% =`	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

Operation type values (PARAM%(0)):

0	None.
1	DMA (Mode 6)
2	Interrupt (Mode 5)
3	Interrupt (Mode 18)
4	Interrupt (Mode 20)

Programming Notes

None.

Example

EX5.BAS, EX6.BAS, EX18.BAS, or *EX20.BAS* illustrate the use of MODE 8.

MODE 9 - TRANSFER DATA FROM MEMORY TO ARRAY

Description

MODE 9 transfers data from any segment of memory to integer array variables in BASIC workspace. Data in memory derived from MODES 5 or 6 is in packed form consisting of a word (2 bytes) of A/D data plus the channel number.

Input Variables

The following input variables must be initialized prior to calling MODE 9:

`MD% =`	9.
`PARAM%(0)=`	**Number of conversions required (Word count).** This is an integer in the range 0 to 32767. Specify 0 when the maximum number of conversions (32768) is desired.
`PARAM%(1)=`	**Source segment of memory to receive data.** This must be the same value as specified in Mode 5, 6, or 20.
`PARAM%(2) =`	**Starting conversion number.** This is an integer in the range 0 - 32767.
`PARAM%(3) = VARPTR(DATA%(M))`	**Start Position in the data array.** This defines where the conversion data is to be loaded within the array. *M* represents the element within the data array (*DATA%*) at which the data loading is to begin. If *M* = 0, conversion will be loaded starting at the beginning of the array.
`PARAM%(4) = VARPTR(CH%(M))`	**Start Position in the channel array.** This defines where the channel data is to be loaded. *M* represents the element within the channel array (*CH%*) where conversions are to be loaded. If *M* = 0, conversion will be loaded starting at the beginning of the array. If `PARAM%(4) = 0`, then no channel data will be stored.

MODE 9 - TRANSFER DATA FROM MEMORY TO ARRAY (cont'd)

Returns

`MD% = 0`		9.
`PARAM%(0-15)`	=	Unchanged.
`FLAG% =`		If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

QuickBASIC Programming Notes

1. Mode 9 for DAS-16 Software Version 4.10 or higher optionally accepts FAR addresses (segment and offset) for the data and/or channel arrays.

   To specify FAR addresses, the following modifications apply to the normal BASIC Mode 9 Call Set-up:

```
PARAM%(1) = -1      'All addresses are FAR and specified in
                    'PARAM%(3) thru (8).

PARAM%(2) =         'Starting Conversion #
PARAM%(3) = VARPTR(DATA%(M))  'Data array Address Offset
PARAM%(4) = VARSEG(DATA%(M))  'Data array Address Offset
PARAM%(5) = VARPTR(CH%(M))  'Channel array Address Offset
PARAM%(6) = VARSEG(CH%(M))  'Channel array Address Offset
PARAM%(7) =    ' "Raw Data" Address Offset  (usually 0)
               ' (usually 0)
PARAM%(8) =    ' "Raw Data" Address Segment
               ' the same as the PARAM%(1) previously
               ' passed to mode 5, 6, or 20
```

Programming Notes

1. To keep track of what channel the data came from, dimension identical channel and data arrays. This will ensure that for each element of the data array there will be a corresponding element of the channel array containing the channel data.

2. A/D data is shifted and also the MSB complimented if the DAS-16/16F is operating in bipolar MODE. In Unipolar MODE, data ranges from 0 to 4095; in Bipolar, -2048 to +2047. The channel data is masked out and ranges from 0 - 15. In order to determine whether the DAS-16/16F is operating in Unipolar or Bipolar MODE, the U/B bit of the status register is read. If the board is removed from the computer or the UNIPOLAR/BIPOLAR switch is changed before operating MODE 9, incorrect data may be obtained.

MODE 9 - TRANSFER DATA FROM MEMORY TO ARRAY (cont'd)

3. Be sure to dimension a receiving array that has at least as many elements as the number of conversions specified in PARAM%(0). If more conversions are performed than the array will hold, some of the data area of BASIC will be overwritten. This may destroy descriptors and other variable data and cause strange effects as a result.

4. After assigning the pointer to the receiving array (by using VARPTR), do not introduce any new simple variables before entering the CALL. For example, the following is correct:

5. There is nothing to prevent you transferring garbage from a source segment that does not contain A/D data or from overrunning the end of A/D data. No checking is performed to detect this condition.

6. Note that due to the reformatting of data that this MODE performs, it is not a general purpose block move utility.

Example See *EXG5.BAS, EXG6.BAS*, or *EXG20.BAS* for an example of MODE 9 usage.

MODE 10 - SET COUNTER 0 CONFIGURATION

Description This mode sets the counter 0 operating configuration. Counter 0 may be set to one of six configurations. These configurations are discussed in detail in Chapter 7.

Input Variables The following input variables must be initialized prior to calling MODE 10:

`MD% =`	10.
`PARAM%(0)=`	**Configuration Number.** This can be an integer from 0 to 5, where:

0 **Output goes high on terminal count.** Output remains high until re-loaded.

1 **Programmable one-shot.** The output goes low after a rising edge of the gate input and goes high when the counter passes thru zero. The period that the output is low is set by the loaded count. If the gate input goes high again before the one shot has timed out, a new timing cycle is initiated. (The one shot is re-triggerable and, if a new count is loaded, it will not become effective until any cycle in progress has terminated.)

2 **Rate generator or divide-by-N counter.** Output pulses low every terminal count.

3 **Square wave generator.** Output high for one half of count, low for other half.

4 **Software triggered strobe.** Output pulses low on terminal count after loading.

5 **Hardware triggered strobe.** Output pulses low on terminal count. Count started by rising edge of gate input.

MODE 10 - SET COUNTER 0 CONFIGURATION (cont'd)

Returns	`MD%` =	10.
	`PARAM%(0-15)`=	Value does not matter.
	`FLAG%` =	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

Programming Notes None.

Example See *EX10.BAS* for an example of MODE 10's use.

MODE 11 - LOAD COUNTER 0

Description	This mode loads Counter 0.	
Input Variables	The following input variables must be initialized prior to calling MODE 11:	
	`MD% =`	11.
	`PARAM%(0) =`	**Counter data.** Values can range from -32768 to +32767 (corresponding to 0 - 65535).
Returns	`MD% =`	11.
	`PARAM%(0-15)=`	Unchanged.
	`FLAG% =`	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

Programming Notes

1. Since the counter is a 16-bit device, count loads as high as 65,535 are possible. Integer variables are signed 16-bit words (can have values between -32768 and +32767). To load a number above 32767, the integer data variable should be set to *X - 65,536*; for example, 40,000 would be entered as -25,536 (See Appendix E.). If MODE 10 has not been entered (set Counter 0 configuration) prior to MODE 11, the configuration will default to #0 which is OK for pulse & event counting.

Example For an example of MODE 11 usage, see *EXG10.BAS*.

MODE 12 - READ COUNTER 0

Description This mode reads Counter 0.

Input Variables The following input variables must be initialized prior to calling MODE 12:

`MD% =`	12.
`PARAM%(0) =`	**Type of Read Operation.** Two types of Counter 0 read operation are possible. If `PARAM%(0) = 0`, a *plain read operation* is performed. This should only be done if the counter data is known to be static (gate input low or no more clock input pulses). If `PARAM%(0) = 1`, then a *counter latching operation* is performed. This type of read should be done if you want to read the counter on the "fly" (while it is counting). Neither type of read will affect the counter data or operation.

Returns

`MD% =`	12.
`PARAM%(0) =`	Unchanged.
`PARAM%(1) =`	**Read data.** Counter data is 16 bits and integer variables are returned as signed 16-bit words. These can have values between -32768 and +32767. Numbers above 32767 are returned as negative integers; for example, -8,000 would correspond to 65,536 - 8,000 = 57,536. For more information regarding integer storage, refer to Appendix E.
`PARAM%(2-15) =`	Unchanged.
`FLAG% =`	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

MODE 12 - READ COUNTER 0 (cont'd)

Programming Notes None.

Example For an example of MODE 12, see *EXG10.BAS*.

MODE 13 - WRITE DIGITAL OUTPUT OP0-3

Description	This mode is used to write digital data to the 4-bit output port, OP0-3.	
Input Variables	The following input variables must be initialized prior to calling MODE 13:	
	`MD% =`	13.
	`PARAM%(0) =`	**Output data.** Output data must be in the range 0 to 15.
Returns	`MD% =`	**The MODE number.**
	`PARAM%(0-15) =`	Unchanged.
	`FLAG% =`	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.
Programming Notes	None.	
Example	For an example of digital I/O using MODE 13, see *EXG13.BAS*.	

MODE 14 - READ DIGITAL INPUT IP0-3

Description	This mode reads the state of digital inputs IP0-3.	
Input Variables	The following input variables must be initialized prior to calling MODE 14:	
	`MD% =`	14.
Returns	`MD% =`	14.
	`PARAM%(0) =`	**Input data.** Data returned can range between 0 and 15 corresponding to all combinations of the 4 input bits, IP0-3.
	`PARAM%(1-15)=`	Unchanged.
	`FLAG% =`	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.
Programming Notes	None.	
Example	For an example of digital I/O using MODE 14, see *EX13.BAS*.	

MODE 15 - OUTPUT DATA TO ONE DAC

Description	This mode writes data to one or the other of the D/A channels.	
Input Variables	The following input variables must be initialized prior to calling MODE 15:	
	`MD% =`	15.
	`PARAM%(0) =`	**D/A channel number.** This can be 0 or 1.
	`PARAM%(1) =`	**D/A data.** The data must be in the range 0 - 4095. If it isn't, an error exit occurs and the D/A is not updated.
Returns	`MD% =`	15.
	`PARAM%(0-15) =`	Unchanged.
	`FLAG% =`	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.
Programming Notes	None.	
Example	For an example of D/A output using MODE 15, see *EXG15.BAS*.	

MODE 16 - OUTPUT DATA TO BOTH DAC's

Description	This mode writes data to both of the D/A channels as simultaneously as possible. D/A #1 is updated 7 microseconds after D/A #0. This MODE is an alternative to entering MODE 15 twice in succession and is useful for driving X/Y plotters, resolvers, etc.	
Input Variables	The following input variables must be initialized prior to calling MODE 16:	
	`MD% =`	16.
	`PARAM%(0) =`	**D/A #0 data.** Data is checked to be in the range of 0 - 4095. If it isn't, an error exit occurs and neither D/A is updated.
	`PARAM%(1) =`	**D/A #1 data.** Data is checked to be in the range of 0 - 4095. If it isn't, an error exit occurs and neither D/A is updated.
Returns	`MD% =`	16.
	`FLAG% =`	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.
Programming Notes	None.	
Example	For an example of D/A output using MODE 16, see *EXG15.BAS.*	

MODE 17 - SET PROGRAMMABLE TIMER RATE

Description

This mode sets the division ratios of Counter 2 and Counter 1 of the 8254 counter timer to produce a programmable output pulse rate for triggering the A/D. The default divisors (set by MODE 0) for Counters 1 & 2 are N (8254 MODE 2). Chapter 7 provides more information regarding the programmable timer.

Input Variables

The following input variables must be initialized prior to calling MODE 17:

`MD% =`	17.
`PARAM%(0) =`	**Counter 2 divisor.** Divisors may range between 2 and 65,535. Use of this divisor is explained in Programming Note 1.
`PARAM%(1) =`	**Counter 1 divisor.** Divisors may range between 2 and 65,535. Use of this divisor is explained in Programming Note 1.

Returns

`MD% =`	17.
`FLAG% =`	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

Programming Notes

1. The minimum divisor for either counter is 2; the maximum is 65,535. Thus, possible pulse rates range from 250KHz to less than 8 pulses/hour with a 1MHz clock or 10 times these rates with a 10MHz clock. If you wish to perform conversions at rates in the 10-100KHz region, the 10MHz clock will give a greater choice of settable rates since you are always limited to dividing the clock frequency by the product of two integers.

MODE 17 - SET PROGRAMMABLE TIMER RATE (cont'd)

The counter input clock frequency may be 1MHz or 10MHz. This depends on the setting of the TIMER Jumper. (See Chapter 2.) and is derived from a precision, on-board, crystal oscillator. The Output frequency is derived as follows:

<u>1 Mhz. Clock</u>
*Output frequency = 1,000,000 / (N2 * N1) Hz*

<u>10 Mhz. Clock</u>
*Output frequency = 10,000,000 / (N2 * N1) Hz*

Where:

N2 = the value of `PARAM%(0)`
N1 = the value of `PARAM%(1)`

The timer rate output can be viewed on an oscilloscope between Pins 20 (+) and 7 (-) of Connector J1.

2. If using DMA MODE 6 with the internal clock to trigger conversions, the pulse rate programmed here (MODE 17) must not exceed the maximum throughput of the system. There are no provisions in the DAS-16 driver to detect this error condition.

Example For an example of MODE 17 usage, see *EXG17.BAS, EXG5.BAS, EXG6.BAS, EXG18.BAS,* or *EXG20.BAS.*

MODE 18 - ANALOG OUTPUT/INPUT ON INTERRUPT

Description

This mode transfers data from a BASIC integer array to one or other of the D/A converters, upon interrupt. Optionally, it transfers the A/D data generated along with each interrupt to another BASIC array. This lets you perform stimulus/response type of testing where the D/A outputs a signal and the A/D measures its result.

Input Variables

The following input variables must be initialized prior to calling MODE 18:

`MD% =`	18.
`PARAM%(0) =`	**D/A channel number.** This can be 0 or 1.
`PARAM%(1) =`	**Number of words to transfer.** This is the number of words (or conversions) that you wish to transfer. This is an integer in the range 0 to 32767. Specify 0 when the maximum number of conversions (32768) is desired.
`PARAM%(2) =`	**Number of cycles to transfer to the array.** The output and input can be set either to transfer a series of values from an array as a single shot operation, transfer the whole array for any number of cycles up to 65,535 or continuously transfer. Continuous transfer is useful for waveform generation, etc. Select 0 to 65535 cycles, with 0 selecting the Continuous transfer mode.
`PARAM%(3) = VARPTR(DA%(0))`	**Assigns a pointer to the D/A data array.**
`PARAM%(4) = VARPTR(AD%(0))`	**Assigns a pointer to the A/D data array.** If you do not want the A/D data to be transferred, set `DIO(4) = 0`.

MODE 18 - ANALOG OUTPUT/INPUT ON INTERRUPT (cont'd)

Returns

`MD% =`	18.
`FLAG% =`	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

QuickBASIC Programming Notes

1. Mode 18 for DAS-16 Ver 4.20 or higher optionally accepts FAR addresses (Segment and Offset) for the D/A and A/D arrays.

   To specify a FAR address, the following modifications apply to the normal BASIC Mode 18 Call setup:

```
PARAM%(3) = -1            'Seg:Off in PARAM%(5) and (6)
PARAM%(4) = -1            'Seg:Off in PARAM%(7) and (8)
PARAM%(5) = VARPTR(DA%(0))     'Offset of DA%() array
PARAM%(6) = VARSEG(DA%(0))     'Segment of DA%() array
PARAM%(7) = VARPTR(AD%(0))     'Offset of AD%() array
PARAM%(8) = VARSEG(AD%(0))     'Segment of AD%() array
```

   Where, DT%() and CG%() are declared earlier as follows:

```
REM $DYNAMIC
DIM DA%(1000)          'DIM of 1000 used or
DIM AD%(1000)          'demonstration only.
REM $STATIC
```

   Refer to the program *QBEXG18.BAS* for usage.

Programming Notes

1. The channel scan limits and interrupt rate (from timer) are set independently preceding this MODE using MODEs 1 and 17.

2. Transfer speed is dependent on the rate at which interrupts can be processed, and varies according to the type of hardware and CPU clock rate, PC/XT or PC/AT. Due to the use of interrupts, D/A update has some variable latency delays of several microseconds which can be improved by suppressing other higher priority internal interrupts such as the PC's timer interrupt on Level 0.

3. Digital input IP0 is polled before enabling interrupts. If IP0 is held low by external hardware, this will hold off the start of D/A output and A/D input until IP0 is taken high. This provides a means of hardware gating the operation of MODE 18.

MODE 18 - ANALOG OUTPUT/INPUT ON INTERRUPT

4. Once interrupts have been initiated by MODE 18, they will run as a background operation until the specified number of cycles of output have been performed. To terminate operation of MODE 18 before this, call MODE 7. Additionally, you can call MODE 8 to determine the status of a MODE 18 operation.

5. After assigning the pointer to the receiving array (by using VARPTR), do not introduce any new simple variables before entering the CALL.

Example For an example of MODE 18's use, see *EX18.BAS*.

MODE 19 - ANALOG TRIGGER FUNCTION

Description

This mode provides an analog trigger function similar to an oscilloscope trigger. It allows you to wait for a voltage to reach a certain level before starting to gather data and thus, is useful as a gating or wait loop in front of any of the other data acquisition MODEs.

Input Variables

The following input variables must be initialized prior to calling MODE 19:

`MD% =` 19.

`PARAM%(0) =` **Select Trigger Channel Number.** This may be one of the scanned channels within the scan limits and carrying one of the measured signals, or a separate channel outside the scanned channels used only for triggering. Valid integers are within the range 0 to 7 for differential configuration, and 0 to 15 for single-ended configuration.

`PARAM%(1) =` **Trigger Level.** This selects the voltage level at which triggering occurs. This ranges from -2048 to +2047 bits for bipolar input ranges and 0 to 4095 for unipolar ranges.

This variable can also be used in conjunction with `PARAM%(4)` and `PARAM%(5)` to specify a "noise band" around the desired trigger level. To specify a noise band, set `PARAM%(1)=-1`.

`PARAM%(2) =` **Slope.** This sets the direction of triggering, where 0 = positive and 1 = negative. For example, suppose `PARAM%(1) = 1024` on the +/-5v range and the trigger level is +2.5 V. If a positive slope were selected (i.e., `PARAM%(2) = 0`, triggering would take place when the signal exceeds +2.5v. Conversely, if a negative slope were specified, (i.e., `PARAM%(2) = 1`, triggering would take place when the trigger signal becomes less than +2.5V.

MODE 19 - ANALOG TRIGGER FUNCTION (cont'd)

`PARAM%(3) =` **Sets the DAS-16G gain register setting.** For the DAS-16/16F, this value does not matter. For the DAS-16G1/G2, this value can range from 0 to 3, as follows:

GAIN PARAM%(3)	DAS-16G1	DAS-16G2
0	1	1
1	10	2
2	100	4
3	500	8

Returns

`MD% =` 19.

`FLAG% =` If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

Programming Notes

1. You can optionally specify a "noise band" around the trigger level. To specify this band, set `PARAM%(1) = -1` and set `PARAM%(4) = Trigger Level (in bits)` followed by `PARAM%(5) = Noise Band (in bits)`. For example, if `PARAM%(4) = 1024` on the ±5V range and `PARAM%(5) = 100`, the trigger level will be +2.5V with a qualification band of 0.25V about the +2.5V level.

2. It is possible to get stuck in the wait loop indefinitely if the trigger conditions are not fulfilled. If this happens, exit MODE 19 by pressing any key. This will return you to the calling program.

Example

For an example of MODE 19 usage, see *EX19.BAS*.

MODE 20 - A/D SCAN OF CHANNELS ON INTERRUPT

Description

For each conversion clock signal (internal or external) this mode performs a complete scan of the Analog Input channels as defined by the last call to mode 1. The actual sampling of inputs is performed in an Interrupt Service Routine as fast as possible. Note that in this mode, as in mode 5, different Gains may be specified for different channels by first calling mode 21 (DAS-16G/AG only).

Input Variables

The following input variables must be initialized prior to calling MODE 20:

`MD% =`	20.
`PARAM%(0)=`	**Number of conversions required (Word count).** This is an integer in the range 0 to 32767. Specify 0 when the maximum number of conversions (32768) is desired.
`PARAM%(1)=`	**Segment of memory to receive data.** Use Mode 23 to obtain a suitable memory segment.
`PARAM%(2) =`	**Trigger source.** If `PARAM%(2) = 0`, then an external trigger input is used. Interrupts will begin as soon as Mode 20 has executed. High pulses on IP0/TRG0 start scans and interrupts continue until the word count is reached in single cycle mode or indefinitely in continuous mode. If `PARAM%(2) = 1`, then the programmable interval timer will be used. In this case, IP0/TRG0 should be held low until you want to start interrupts. After the sequence is triggered, IP0/TRG0 will have no further effect. Exit will not occur until IP0 goes high.

MODE 20 - A/D SCAN OF CHANNELS ON INTERRUPT (cont'd)

`PARAM%(3) =` **Single Cycle/Continuous Operation.** If `PARAM%(3)=0`, then one cycle will be performed. This means that after the specified number of conversions are completed, interrupts will be disabled and the operation status set to zero. The interrupt handler will remain "active" until the specified number of conversions has been completed.

If `PARAM%(3)=1`, continuous operation is selected. This means that data will be continuously written to the same area of memory, where `PARAM%(0)` represents the memory "buffer" length. Data will continue to be written until MODE 7 is called.

Returns

Returns	`MD% =`	20
	`PARAM%(0-15) =`	Unchanged.
	`FLAG% =`	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

QuickBASIC Programming Notes

1. QuickBASIC users may choose not to use MODE 23 to allocate memory for the acquisition buffer especially when in the QB Integrated Environment where MODE 23 will not work. Instead, users can declare a $DYNAMIC array of any size and pass its Segment and Offset to Mode 21.

   To specify a FAR address to Mode 21, the following modifications apply to the normal BASIC Mode Call setup:

```
PARAM%(1) = -1     'i.e. Seg:Offset are in PARAM%(4)
and (5)
...
PARAM%(4) = VARPTR (ARRAY%(0))    'Address Offset
PARAM%(5) = VARSEG (ARRAY%(0))    'Address Segment
...
```

MODE 20 - A/D SCAN OF CHANNELS ON INTERRUPT (cont'd)

Where, ARRAY%() is earlier declared as follows:

```
REM $DYNAMIC
DIM ARRAY%(10000)          'DIM of 10000 used or
REM $STATIC                'demonstration only.
```

Programming Notes

1. Due to the execution delays of the interrupt handler, block scan rates are limited and may produce variable delays in the scan rate. If this is a problem, try suppressing other interrupts (especially the PC timer interrupt) while data is collected in this MODE.

   The scan rate attainable is also dependent on the number of channels in the block. If you begin to miss channels or foreground processes slow, you are operating beyond or close to the maximum interrupt rate.

Example *EX20.BAS* shows an example of MODE 20.

MODE 21 - SET CHANNEL GAINS

Description

<u>THIS MODE CAN BE USED WITH THE DAS-16G/DAS-16AG ONLY.</u> It sets up a gain/channel table internal to the driver to be used in MODEs 3, 4, 5 and 20. (These modes support software switching of the gain between conversions.)

Input Variables

The following input variables must be initialized prior to calling MODE 21:

`MD% =`	21.
`PARAM%(0)`	Gain code for Channel 0.
`PARAM%(1)`	Gain code for Channel 1.
`PARAM%(2)`	Gain code for Channel 2, etc.
`...`	
`PARAM%(15)`	Gain code for channel 15.

Gain codes are as follows:

GAIN CODE	DAS-16G1	DAS-16G2
0	1	1
1	10	2
2	100	4
3	500	8

Returns

`MD% =`	21.
`FLAG% =`	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.
`PARAM%(0-15)`	Unchanged.

MODE 21 - SET CHANNEL GAINS

Programming Notes

1. If MODE 0 has executed and you do not run MODE 21, all channel gains will default to 1. MODE 21 would normally be run as part of the initializing code to set the channel gain configuration (See programming examples that follow).

2. MODE 21 range checks all the channel gain data in elements PARAM%(0) thru PARAM%(15) whether or not you are using the channels. For instance, if the DAS-16G has been set up to operate in 8 channel differential MODE, and you have used MODE 1 to set the scan limits from 2 to 5, only channels 2, 3, 4 & 5 will be active. The gains of the unused channels will not matter. However, MODE 21 still requires all channels, including the unused ones, to be set with a valid gain code in the range 0-3 and will give an error if this is not done.

Example

```
XX000  MD%=21                          'Select MODE 21
XX010  PARAM%(0)=3                     'Channel 0 gain = 500
XX020  PARAM%(1)=1                     'Channel 1 gain = 10
XX030  PARAM%(I) = 0 : NEXT I          'Channels 2-7 gain = 1
XX040  FOR I = 2 TO 7
XX050  PARAM%(8) = 1                   'Channel 8 gain = 10
XX060  FOR I = 9 TO 13         'Set channels 9-13 gain = 100
XX070  FOR PARAM%(I) = 2 : NEXT I
XX080  PARAM%(14) = 1                  'Channel 14 gain = 10
XX090  PARAM%(15) = 0                  'Channel 15 gain = 1
XX100  CALL DASG(MD%, PARAM%(0), FLAG%0
XX110  IF FLAG%<>0 THEN PRINT "Error #";FLAG%;" in setting
       gains" :STOP
XX120                                  '...Continue Progress
```

MODE 22 - OUTPUT A SQUARE WAVE

Description

This mode provides a quick and easy way of outputting a square wave of defined frequency from the DAS-16. It performs all the setup necessary to connect counter 0 to the internal 100KHz xtal clock source and set its configuration to the square wave output MODE. For this MODE of operation inputs Gate 0/IP2 (pin 24) and Counter 0 In (pin 21) should be held high or left open circuit.

The frequency of the square wave is set by:

Frequency = 100,000 / PARAM%(0) Hertz

Practical output frequencies range from 25KHz down to 1.5 Hz. Since the counter by nature divides by an integer (PARAM%(0)) finer intervals of frequency are obtainable at lower frequencies. Due to the operation of the 8254 counter-timer, if PARAM%(0) is an odd number, one half of the square wave will be one clock period (10 microseconds) longer than the other half. If PARAM%(0) is an even number, both halves will be identical in duration and a perfect 50% duty cycle will be obtained. The output appears on Counter 0 Out (pin 2). Since this is an uncommitted counter, this output can be obtained regardless of other operations A/D, D/A, digital I/O etc. on the board.

Input Variables

The following input variables must be initialized prior to calling MODE 22:

`MD% =`	22.
`PARAM%(0) =`	Counter divider (2 to 65535).

Returns

`MD% =`	22.
`PARAM%(0 - 15) =`	Unchanged.
`FLAG% =`	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

MODE 22 - OUTPUT A SQUARE WAVE

Programming Notes None.

Example An example of the use of MODE 22 is in *EXG22.BAS*.

MODE 23 - MEMORY ALLOCATION

Description This mode uses the DOS Allocate Function 48H to reserve a good memory buffer and returns a its address segment and offset and the value of the actual segment allocated by DOS. This mode can be used with Mode 5, 6, 18, or 20 in any of the supported languages except Quick-BASIC Integrated Environment. See these mode call descriptions for more information on how to allocate memory without using Mode 23.

Input Variables The following input variables must be initialized prior to calling MODE 23:

`MD% =`	23.
`PARAM%(0) =`	**Number of Words/Bytes to Allocate.**
`PARAM%(1) =`	**Allocate Bytes or Words.** Set this variable to 0 if specifying a number of bytes or 1 if specifying a number of words in `PARAM%(0)`.

Returns

`MD% =`	23.
`PARAM%(0-4) =`	Unchanged.
`PARAM%(5) =`	**Address Segment** used by modes 5, 6, and 20.
`PARAM%(6)=`	**Address Offset.** This will always be 0.
`PARAM%(7)=`	**Actual Allocated Segment** used by mode 24.
`FLAG% =`	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

Programming Notes None.

Example An example of the use of MODE 23 is in *EXG23.BAS*.

MODE 24 - DEALLOCATE MEMORY SEGMENT

Description This mode uses the DOS Allocate Function 49H to release memory allocated by Mode 23. This mode requires that the actual segment be passed in. (This was returned by `PARAM%(7)` in mode 23.)

Input Variables The following input variables must be initialized prior to calling MODE 24:

`MD% =`	24.
`PARAM%(0) =`	**Actual Allocated Segment.** This was returned by `PARAM%(7)` in mode 23.

Returns

`MD% =`	24.
`FLAG% =`	If an error has occurred, `FLAG%` will contain the error code(s). These are described further in Appendix C.

Programming Notes None.

Example An example of the use of MODE 23 is in *EXG23.BAS.*

5.4 USING MULTIPLE DAS-16s IN A SINGLE SYSTEM

If you want to install more than one DAS-16 in a single system, be sure that each board is assigned a unique base address, interrupt level, and DMA level. (Note that common interrupt and DMA levels can be assigned as long as only one board's interrupt/DMA is enabled at a time.) Each board must also be assigned its own CALL routine as shown below:

```
xxx10    DEF SEG = SG1
xxx20    BLOAD "DASG.BIN",0
xxx30    SG2 = SG1 + 6144/16       'allow 6K for each routine
xxx40    DEF SEG = SG2
xxx50    BLOAD "DASG.BIN",0
xxx60    SG3 = SG2 + 6144/16        'etc.  for other boards
```

Now the CALL appropriate to each board can be entered as required. Note that each CALL is preceded by a DEF SEG appropriate to that board:

```
yyy10    DEF SEG = SG1
yyy20    CALL DASG1 (MD%, X%, FLAG%)
yyy30    DEF SEG = SG2
yyy40    CALL DASG2 (MD%, X%, FLAG%)  'etc.
```

CHAPTER 6
LOW-LEVEL PROGRAMMING

At the lowest level, the DAS-16 is programmable via I/O (Input/Output) instructions. In BASIC, these are the `INP (X)` and `OUT X,Y` functions. Assembly language and most other high- level languages have equivalent instructions (for example, `IN AL,DX` and `OUT DX,AL` in Assembly). Use of these functions usually involves formatting data and dealing with absolute I/O addresses. Although not demanding, this type of programming requires that you have a full understanding of the devices, data format, and architecture of the DAS-16.

This chapter discusses the DAS-16 registers and their functions. The registers are grouped and discussed according to function, as follows:

- A/D Registers
- Mux Scan Register
- Digital I/O
- D/A Output Registers
- Status Register
- Control Registers
- Timer-Counter Enable Registers
- 8254 Programmable Interval Timer-Counter Registers

More information regarding the Programmable Interval Timer and the D/A Converters are provided in the following chapters.

6.1 I/O ADDRESS MAP

DAS-16 boards use 16 consecutive addresses starting at the base address in the computer's I/O space as shown in Table 6-1.

Table 6-1. I/O Address Map

Location	Function	Type
Base Address + 0	A/D Low byte Convert A/D	Read Write
Base Address + 1	A/D High byte	Read
Base Address + 2	Mux scan control	Read/Write
Base Address + 3	Digital I/O Out Nybble Digital I/O In Nybble	Write Read
Base Address + 4	D/A 0 Low byte	Write
Base Address + 5	D/A 0 Hi byte	Write
Base Address + 6	D/A 1 Low byte	Write
Base Address + 7	D/A 1 Hi byte	Write
Base Address + 8	Board Status	Read
Base Address + 9	Board Control	Read/Write
Base Address + 10	Counter Enable (2 bit) *	Read/Write
Base Address + 11	Programmable Gain (DAS-16G only) *	Read/Write
Base Address + 12	Counter 0	Read/Write
Base Address + 13	Counter 1	Read/Write
Base Address + 14	Counter 2	Read/Write
Base Address + 15	Counter Control	Write

* Users writing drivers for the DAS-16G who have already developed code for the fixed gain DAS-16/16F should note the following differences:

1. *Base Address + 10 (Counter Enable) Read/Write:* On DAS-16/16F this is a 2-bit write-only register. On DAS-16G, it is an 8-bit read/write register, although the same two bits (D0 & D1) are used for counter control. The remaining six bits are unused on DAS-16G and non-existent on DAS-16/16F.

2. *Base Address + 11 (Programmable Gain) Read/Write:* This address was unused on DAS-16/16F. On DAS-16G it is an 8-bit read/write register, the upper six bits are unused, and the lower two bits select the gain range.

3. All software developed for the fixed gain DAS-16/16F operates with the DAS-16G, although it will need modification to use the programmable gain feature. If you want to set DAS-16G gain before running application software designed for the older fixed gain DAS-16/16F, run *SETGAIN.EXE* before starting your application. This sets the programmable gain register to BASE ADDRESS + 11.

6.2 A/D REGISTERS (Base Address + 0 & + 1)

The A/D registers read only at Base Address + 0 and Base Address + 1 in standard Intel 8088 Lo/Hi-byte sequence. Data is true binary and the mux channel address is contained in the least significant nybble so that the data is "tagged" with the channel from which it was derived. A/D data is latched so that data from the previous conversion is available during and up until the end of the following conversion. Channel information can be masked out or discarded by the programmer, as required. Data format is as follows:

	D7	D6	D5	D4	D3	D2	D1	D0
Base + 0	B3	B2	B1	B0	A3	A2	A1	A0
				LSB				
	DATA				CHANNEL NO.			

	D7	D6	D5	D4	D3	D2	D1	D0
Base + 1	B11	B10	B9	B8	B7	B6	B5	B4
	DATA							

The A/D registers are read only. A write to Base Address + 0 acts as an A/D trigger and will always initiate a conversion. Any data written to these address is lost.

6.3 MUX SCAN REGISTER (Base Address + 2)

The MUX scan register is a read/write register that controls the channel scan limits of the A/D multiplexer. The high nybble provides the scan ending address and the low nybble provides the scan starting address. Data is true binary and the format as follows:

	D7	D6	D5	D4	D3	D2	D1	D0
Base + 2	AH3	AH2	AH1	AH0	AL3	AL2	AL1	AL0
	END OF SCAN ADDRESS				START OF SCAN ADDRESS			

About two microseconds after the A/D starts a conversion, while the sample/hold is holding the previous channel, the multiplexer address is incremented ready for the next conversion. On finishing the conversion performed with the end-of-scan address, the cycle repeats starting with the scan start address. On writing to the mux. scan register the counter is always automatically initialized to the scan start address.

To perform conversions on a single channel, the end-of-scan and start-of-scan addresses should be made equal to the desired channel number.

If the DAS-16 is operated in the 8 channel differential mode, the programmer should insure that the AH3 and AL3 bits are zero. The operating mode can always be determined by reading Bit D5 in the DAS-16/16F Status Register.

6.4 DIGITAL I/O (Base Address + 3)

Digital I/O consists of two hardware-configured ports, a 4-bit output port (OP0-3) and a 4-bit input port (IP0-3). These ports share the same I/O address but are independent (i.e. data written to the output port is not readable by the input port unless the OP0-3 lines are externally connected to the IP0-3 lines). In addition some of the input port lines do double duty. IP0 is also TRIG0 and provides an external trigger to initiate A/D conversions, and IP2 is also Counter Gate 0 and controls operation of Counter 0 in the 8254 Counter/Timer. Depending on application, these secondary functions may or may not be used; either way, the state of the inputs can always be determined by reading Base Address + 3.

Data format of the output port:

	D7	D6	D5	D4	D3	D2	D1	D0
Base + 3	X	X	X	X	O3	O2	O1	O0

X = DON'T CARE

Data format of the input port:

	D7	D6	D5	D4	D3	D2	D1	D0
Base + 3	0	0	0	0	I3	I2	I1	I0

X = DON'T CARE

6.5 D/A OUTPUT REGISTERS (Base Address + 4, + 5, + 6, & +7)

D/A output registers are write-only and in standard low/high byte sequence. Note that the D/A registers are "double buffered" so that the D/A's are not updated until the second (high) byte is written. This insures a single step transition on the analog output of the D/A. The D/A data format is true binary, left-justified as follows:

D/A 0

	D7	D6	D5	D4	D3	D2	D1	D0
Base + 4	B4	B3	B2	B0	X	X	X	X
				LSB				

WHERE X = DON'T CARE

	D7	D6	D5	D4	D3	D2	D1	D0
Base + 5	B11	B10	B9	B8	B7	B6	B5	B4
	MSB							

D/A 1

	D7	D6	D5	D4	D3	D2	D1	D0
Base + 6	B3	B2	B1	B0	X	X	X	X
				LSB				

WHERE X = DON'T CARE

	D7	D6	D5	D4	D3	D2	D1	D0
Base + 7	B11	B10	B9	B8	B7	B6	B5	B4
	MSB							

6.6 STATUS REGISTER (Base Address + 8)

The status register is a read-only register that provides information on the operation and configuration of the A/D in the DAS-16/16F. Writing to the status register address clears the DAS-16/16F interrupt request and provides the means of acknowledging the DAS-16/16F interrupt and re-enabling it. The format is as follows:

	D7	D6	D5	D4	D3	D2	D1	D0
Base + 8	EOC	U/B	MUX	INT	MA3	MA2	MA1	MA0

Where:

EOC	**End of conversion.** If EOC = 1, the A/D is busy. EOC = 0 implies that the A/D is ready to perform the next conversion and the A/D data registers contain valid data from the previous conversion.
U/B	**Unipolar/Bipolar.** This provides information on whether the A/D input is operating in unipolar or bipolar mode (as set by UNI/BIP switch). Unipolar = 1, Bipolar = 0. This status bit can be useful in scaling routines.
MUX	**Single-Ended/Differential.** Provides information on single-ended or differential operations. It is controlled by the position of the Channel Configuration Switch. 16 channel s/e = 1, 8 channel diff. = 0. This status bit is useful in mux control.
INT	**Interrupt Level.** Once INTE (Control Register - BASE +9, Bit 8) is enabled (set to 1), INT is the interrupt generated by the DAS-16 Boards. It is directed to the PC's Interrupt Level 2-7, as specified by the user (see MODE 0 on the DAS-16 initialization, in Chapter 4). The DAS-16 sets this register bit to 1, as required. Your program's Interrupt Service Routine (ISR) that services this interrupt must clear this bit so that subsequent interrupts (as generated by the DAS-16) may be detected. The INT bit must therefore be cleared in the ISR by writing (any data) to the Status Register (BASE + 8).

MA3-0 **Multiplexer Address.** This is the mux address (channel number) of the next channel to be converted if EOC = 0. The channel address changes shortly after EOC goes high and when EOC = 1 may be indeterminate. These status bits provide information on setup for the next conversion.

6.7 CONTROL REGISTER (Base Address + 9)

The Control Register is a read/write register that controls the operating modes of the board. The register is cleared on power up (reset), thereby disabling DMA and interrupt and external A/D start modes. The various bits of the control register function as follows:

	D7	D6	D5	D4	D3	D2	D1	D0
Base + 9	INTE	L2	L1	L0	X	DMA	S1	S0

Where:

INTE **Interrupt Enable.** This bit enables or disables DAS-16 generated interrupts: INTE = 1 - Enabled INTE = 0 - Disabled.

If INTE=1 and DMA=0, an interrupt is generated at the end of each A/D conversion when valid data is available. If INTE=1 and DMA=1, an interrupt is generated when a terminal count (T/C) is received from the 8237 DMA controller to signify completion of the DMA transfer.

L2-L0 **Interrupt Level Select.** Selects the desired interrupt level for processing interrupts as shown below:

L2	L1	L0	Interrupt Level
0	0	None	Disabled
0	0	None	Disabled
0	1	0	Level 2
0	1	1	Level 3
1	0	0	Level 4
1	0	1	Level 5
1	1	0	Level 6
1	1	1	Level 7

Avoid using a level already assigned to another I/O device (for example, Level 6 is used by the floppy disk drive(s)).

X This bit has no assigned function.

DMA **Enable/Disable DMA.** Direct memory access transfers are enabled when DMA = 1 and disabled when DMA = 0. It is the users responsibility to set up the 8237 DMA controller on the IBM PC system board and the page registers before enabling DMA on DAS-16.

S1-0 **Source of Start Pulses.** These bits control the source of start pulses for the A/D as shown below:

S1	S0	Interrupt Level
0	X	Software start only (*)
1	0	Start on rising Trig 0.
1	1	Start on timer output

* Regardless of the state of the S0,1 bits, an A/D conversion can always be initiated by a write to A/D register 0.

6.8 TIMER-COUNTER ENABLE REGISTER (Base Address + 10)

This is an 8-bit read/write register that controls operation of the timer counter. Only the least significant two bits are used; the remaining bits have no affect on the hardware and may be used by the programmer for any other purpose. Note that the Timer-Counter Register is not cleared by power-on/reset; it must be explicitly initialized by the user.

The DAS-16 uses an Intel 8254-2 Programmable Interval Timer (PIT) for periodically triggering the A/D converter. Counters 1 and 2 of the PIT are cascaded and driven by a 1 MHz or 10MHz clock (board-jumper-selectable), so that periods ranging from a few microseconds to 8 pulses per hour may be programmed.

When the board is programmed for A/D conversions using the internal timer, Bit 0 (C0) of this register and the TRG0/IP0 input signal are used to enable Counters 1 and 2 of the 8254 chip, thereby initiating the A/D conversions. To accomplish this, Bits C0 and IP0 are NANDed and the output routed to Counter Gates G1 and G2 of the 8254.

The function of C1 is somewhat different. When C1 and CTR 0 IN are both high, Counter 0 input of the 8254 is internally connected to a 100kHz internal crystal clock source. The CLOCK IN, GATE, and COUNTER OUT of Counter 0 are all brought out to external connections, so that the counter may be used for waveform generation, as a secondary timer or period counting using the gate and internal clock. By disabling the internal 100KHz clock with C1 = 0, external clock pulses can be fed in using the CLOCK IN 0. In this mode Counter 0 can be used as an event counter or if the COUNTER 0 GATE input is connected to a timebase (e.g. COUNTER 2 OUTPUT, Counter 0 can be used to determine frequency).

	D7	D6	D5	D4	D3	D2	D1	D0
Base + 10	X	X	X	X	X	X	C1	C0

Where:

X	Don't Care.

6.9 PROGRAMMABLE GAIN REGISTER (Base Address +11; DAS-16G Only)

An 8-bit read/write register that sets the gain of the programmable gain amplifier. Only the bottom two bits are used, the upper six bits have no effect on the hardware and can be used by the programmer for any other purpose. Note that the Programmable Gain register is not cleared by power on reset, it must be explicitly loaded and initialized by the user.

The functions of the bits in the programmable gain register are as follows:

	D7	D6	D5	D4	D3	D2	D1	D0
Base + 10	X	X	X	X	X	X	C1	C0

Where:

X	Don't Care.
G1-G0	**Gain Code Select Bits.** These are used to select the gain as follows:

GAIN CODE	DAS-16G1	DAS-16G2
0	1	1
1	10	2
2	100	4
3	500	8

6.10 8254 PROGRAMMABLE INTERVAL TIMER-COUNTER (Base Address +12, +13, +14, & +15)

The four registers, from Base Address +12 onward, correspond to the four registers of the 8254 programmable timer-counter. Users are referred to the Intel 8254 data sheet for a full description of features. Condensed information is available in Chapter 7.

Base Address + 12	Counter 0 register	Read/write
Base Address + 13	Counter 1 register	Read/write
Base Address + 14	Counter 2 register	Read/write
Base Address + 15	8254 control register	Write only

CHAPTER 7 PROGRAMMABLE INTERVAL TIMER

The Intel 8254 programmable interval timer is used in the DAS-16. This is a flexible, but complex device consisting of three independent 16-bit presettable down counters. Each counter can be programmed to divide by any integer in the range 2 - 65,536. In the DAS-16, Counters 1 and 2 are cascaded with the input of Counter 1 connected to a precision 1 or 10MHz crystal oscillator. The output of Counter 1 is connected to the input of Counter 2, and Counter 2 output may be selected internally as well as being available to the user at the COUNTER 2 OUT (Pin 20). The other counter, Counter 0, is uncommitted and its input, output and gate control are available to the user on COUNTER 0 IN (Pin 21), COUNTER 0 OUT (Pin 2) and IP2 (Pin 24). A block diagram of the DAS-16 counter arrangement is shown in Figure 7-1.

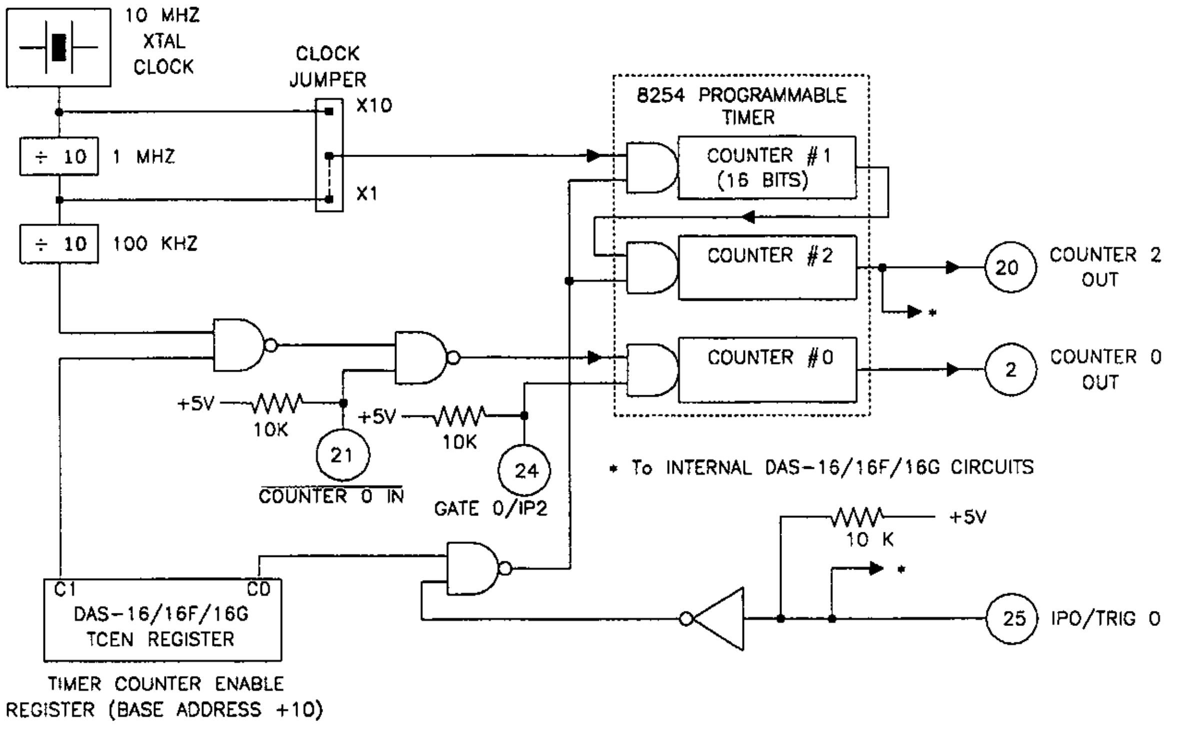


Figure 7-1. Programmable Timer Configuration

Principal uses of the 8254 are as follows:

1. A programmable timer for generating interrupts and triggering periodic A/D conversions.

2. A variable-frequency square wave generator for testing and for frequency synthesis.

3. An event counter for external pulse inputs.

4. A time-delay generator.

In addition, it is possible to accurately measure frequency and pulse width by interconnecting some of the counters. This capability is not supported by the DAS-16 driver; but can be utilized by the more adventurous programmer. Those persons interested in more detailed information on the 8254 may request a complete data sheet from Intel by calling 800/628-8686.

Each counter has a clock input, a gate input that controls counting and triggering and an output. The maximum clock input frequency on any counter is 10MHz with minimum clock duty cycles of 30ns high and 50ns low (note that this specification applies only to the -2 version of the 8254). On later models of the DAS-16, it is also possible to drive Counters 1 and 2 from a 10MHz clock. This selection is made via a jumper block on the board marked TIMER, the 1 position corresponding to a 1MHz clock and the 10 position to 10MHz. The usual function of these two counters is to provide programmable pulse rates to trigger the A/D. Counter 0 is uncommitted and can be used as a secondary pulse-rate generator, a square-wave generator, a programmable monostable delay or an event counter. Counters 1 and 2 are initialized by the DAS-16 driver to operate in the Rate Generator Configuration (#2) and output 1KHz (10KHz with 10MHz clock) after running initializing MODE 0; but there are actually six possible operating configurations for each counter, as described in the next section.

The 8254 programmable interval counter occupies four I/O address locations in the DAS-16 I/O address map as shown in Table 7-2.

Table 7-2. Programmable Interval Counter Register Locations

Location	Function	Type
Base Address + 12	Counter 0	R/W
Base Address + 13	Counter 1	R/W
Base Address + 14	Counter 2	R/W
Base Address + 15	Counter Control	W

7.1 THE COUNTER CONTROL REGISTER

Before loading or reading any of the individual counters, the 8254 Control Register must be loaded with data setting the counter operating configuration as above, the type of read or write operation that will be performed (see following), and the modulus or binary (0-65,535) or BCD (Binary Coded Decimal 0-9,999). The format of the control byte is:

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

Where:

SC1-0 **Select Counter.** These bits select the counter to be used. Valid combinations include:

SC1	SC0	Counter
0	0	0
0	1	1
1	0	2
1	1	Readback Command (See section 7.2).

RL1-0 **Select Read/Load Operation.** These bits select the type of read to be used. Valid combinations include:

RL1	RL0	Operation
0	0	Counter Latch (See Following)
0	1	Read/Load Most Significant Byte
1	0	Read/Load Least Significant Byte
1	1	Read/Load Least Significant Byte, followed by Most Significant Byte.

☞ ***NOTE:*** To minimize I/O port usage the 16 bit counters of the 8254 are designed so that they may be loaded and read by successive 8-bit writes or reads to the same I/O address if RL0 and RL1 are both set high. In the IBM PC 8088/80286 architecture, this is usually the most convenient setting. Data is written in least significant/most significant order and the 8254 uses an internal flip-flop to keep track of the byte order. It is important not to get this flip-flop out of step when programming in this way (make sure your read/load instructions are in pairs). If you have an IBM PC/AT and are programming in assembly language, be sure to put delays (e.g. a short jump) between back-to-back I/O instructions for correct operation.

M2-0 **Select Counter Configuration.** These bits select the counter configuration to be used. Valid combinations include:

M2	M1	M0	Configuration
0	0	0	0 - Pulse on Terminal Count.
0	0	1	1 - Programmable One Shot
X	1	0	2 - Rate Generator.
X	1	1	3 - Square Wave Generator.
1	0	0	4 - Software Triggered Strobe.
1	0	1	5 - Hardware Triggered Strobe.

NOTE: X = Don't Care

The Configuration Modes are described below:

PULSE ON TERMINAL COUNT. The output is initially low for this configuration. After the count loads and the counter decrements through zero, the output goes high and remains high until the counter is reloaded. The counter continues to decrement after passing thru zero and counting can be inhibited by a low gate input. This mode produces a single positive going output transition such as may be required in a time delay initiated by the program.

PROGRAMMABLE ONE SHOT. The output goes low after a rising edge of the gate input and goes high when the counter passes thru zero. The period that the output is low is set by the loaded count. If the gate input goes high again before the one shot has timed out, a new timing cycle is initiated (the one shot is re-triggerable and, if a new count is loaded, it will not become effective until any cycle in progress has terminated). This provides a hardware triggered delay or one-shot.

RATE GENERATOR (or Divide-By-N counter). The output goes low for one input clock period every N counts, where N is the count loaded. The gate input when low, forces the output high, and on going high, reloads the counter. Thus, the gate input can be used to synchronize the counter. This configuration is useful for generating periodic interrupts to trigger A/D conversions.

SQUARE WAVE GENERATOR. Similar to Configuration 2 except that the output is high for half of the count and low for the other half. If N is even, a symmetrical square wave output is obtained. If N is odd, the output is high for (N+1)/2 counts and low for (N-1)/2 counts (has a 1-count asymmetry). This configuration can be used in the same way as Configuration 2 for periodic triggering or for frequency synthesis.

SOFTWARE TRIGGERED STROBE. After the mode is set the output is high. When a count of N is loaded the counter begins counting and the output will go low for one input clock period as it passes thru zero. The cycle is repeated on loading another count. The gate input may be used to inhibit counting.

HARDWARE TRIGGERED STROBE. This is essentially the same as Configuration 1, except that the output will go low for one clock period at the end of the cycle and return high again. The start of the cycle is triggered by the rising edge of the gate input, and as in Configuration 1, is retriggerable.

BCD **Binary/Decimal Coding.** These bits select whether counts will be taken in binary (modulus 2) or Binary Coded Decimal (modulus 10). The binary mode with a full count of 65,535 has the obvious advantage of providing a greater count range than the BCD mode which has a 9,999 full scale and is also more compatible with data transfer to and from integer (binary) variables.

Valid combinations include:

BCD	Counter Type
0	Binary 16 bits
1	Decimal 4 decades

7.2 READING AND LOADING THE COUNTERS

If you attempt to read the counters on the fly with a high input frequency, you will most likely obtain erroneous data. This is partly caused by the rippling of the counter during the read operation and also by the fact that the low and high bytes are read sequentially rather than simultaneously, making it highly probable that carries will be propagated from the low to high byte during the read cycle. To circumvent these problems, you can perform a counter latch operation in advance of the read cycle. To do this you load the RL 1/0 bits of the control byte with "0 0" which instantaneously latches the count of the selected counter in a 16 bit hold register. An alternative method of latching counter(s) which has an additional advantage of operating simultaneously on several counters is by use of the **Readback Command.** (This is described below.) A subsequent read operation on the selected counter returns the held value. Latching is the only way of correctly reading a counter "on the fly" without disturbing the counting process. If you do not specify a latching operation, then the counter itself will be read. You can only rely on directly read counter data if the counting operation is suspended while reading e.g by removing the clock input or taking the counter gate low.

For each counter you are required to specify in advance the type of read or load operation that you intend to perform. You have a choice of loading/reading the high byte of the count or the low byte of the count, or the low byte followed by the high byte. This last mode is of the most general use and is selected for each counter by setting the RL1 and RL0 bits to "1 1". Subsequent read/load operations must be performed in pairs in this sequence, otherwise the internal sequencing flip-flop of the 8254 will get out of step.

If the SC0 and SC1 bits are both set to 1, you can perform two types of operations. When CNT=0 (In the Readback Command Byte, See below.), counters selected by C0 thru C2 are latched simultaneously. When STA=0, the counter status byte will be read on accessing the counter I/O location. The status byte provides information on the current output state of the counter, and its operating configuration.

Readback Command

The Readback Command byte is of the following format:

D7	D6	D5	D4	D3	D2	D1	D0
SC1 = 1	SC0 = 1	CNT	STA	C2	C1	C0	0

Where:

SC1-0 **Select Counter.** These bits both = 1.

CNT **Counter Latch.** When this bit is 0, the counter(s) selected by bits C2 - C0 are latched.

STA **Status.** If this bit is 0, then the status of the counter(s) selected by bit C2 - C0 is returned. The status byte returned is of the format:

D7	D6	D5	D4	D3	D2	D1	D0
OUT	NC	RL1	RL0	M2	M1	M0	BCD

Where:

OUT Current State of Counter Output.

NC **Null count.** This indicates when the last count loaded into the counter register has actually been loaded into the counter itself. The exact time of load depends on the configuration selected. Until the count is loaded into the counter itself, it cannot be read from the counter.

C2 - C0 **Select a Counter.** When these bits = 1, they select the counter(s) to be used in the Readback Operation, where:

C0 = Counter 0
C1 = Counter 1
C2 = Counter 2

For example, if both the STA and CNT bits are set low and RL0 and RL1 have both been previously set high in the counter control byte (selecting 2 byte reads), then reading a selected counter location should yield the following:

1st. read :	Status byte
2nd. read :	Low byte of latched data
3rd. read :	High byte of latched data
4th. read :	Low byte of counter direct
5th. read :	High byte of counter direct

Programming Examples in BASIC

Example 1.

```
xxx10     OUT BASE + 15, &H76       'set Counter 1 to squarewave mode
xxx20     OUT BASE + 15, &HB6       'set Counter 2 to squarewave mode
xxx30     X = 1000                  'number to load both counters
xxx40     XH% = INT(X/256)          'high byte
xxx50     XL% = X - 256 * XH%       'low byte
xxx60     OUT BASE + 13, XL%
xxx70     OUT BASE + 13, XH%        'load counter 0
xxx80     OUT BASE + 14, XL%
xxx90     OUT BASE + 14, XH%        'load counter 1
```

The two counters will now be dividing the 1 or 10 MHz XTAL clock by 1,000,000 (1,000 x 1,000) so that COUNTER 2 OUT will now be outputting a 1 or 10Hz. squarewave (depending on crystal input). Then, to determine the status of Counter 1, you could program:

```
yyy10     OUT BASE + 15, &HE4              'control byte to read status
yyy20     PRINT HEX$(INP(BASE + 13))       'read status
```

Example 2. The following example shows how to use the uncommitted Counter 0 as a pulse counter. Note that the counters are all "down" counters. Thus, when resetting them, it is better to load them with a full count value of 65,535 rather than zero.

```
xxx10     OUT BASADR%+15, &H30        'setup Counter 0 to pulse on
                                      'terminal count, load with
                                      'double byte transfer,
                                      'binary count.
```

```
xxx20      OUT BASADR%+12, &HFF          'load low byte with Hex FF
xxx30      OUT BASADR%+12, &HFF          'load high byte with Hex FF
```

Counter 0 is now loaded with Hex FFFF (decimal 65,535)

```
xxx40      OUT BASADR%+15, &H00           'latch counter 0
xxx50      XL% = INP(BASADR%+12)          'read low byte
xxx60      XH% = INP(BASADR%+12)          'read high byte
xxx70      CHANGE = 65535 - 256*XH% - XL%  'calculate change in count
```

In practice, MODEs 10 thru 12 of the *DASG.BIN* driver can be used to perform an equivalent operation to the above example with fewer programming steps. An example is provided in *M10.BAS* and the steps explained below.

For counting pulses the counter configuration (one shot, square wave generator, etc.) is not of great importance since you will only be using the count down capabilities of the counter. Configuration 0 is as good as any other choice for pulse counting:

```
xxx10      MD% = 10 : DIO%(0) = 0       'sets configuration 0
xxx20      CALL DASG (MD%, DIO%(0), FLAG%)
```

Example 3. As in the previous example, we now load counter 0 with a full scale count of 65,535 (Hex FFFF) using MODE 11 of the driver. Note that in 2s complement form, 65,535 corresponds to -1 (see Appendix C). While we are loading the counter, counting can be inhibited if required by holding the gate input low.

```
xxx30      MD% = 11 : DIO%(0) = -1
xxx40      CALL DASG (MD%, DIO%(0), FLAG%)
```

Next, input the pulses to be counted (the gate input must now be high or can be taken high for some fixed time interval to control the number of pulses counted). The new count can be read using MODE 12 of the driver, i.e.:

```
xxx50      MD% = 12 : DIO%(0) = 0 : DIO%(1) = 0
xxx60      CALL DASG (MD%, DIO%(0), FLAG%)
```

Note that in MODE 12 (counter read), DIO%(0) specifies the type of read operation that you wish to perform. If DIO%(0) = 1, a counter latching operation is automatically performed. This method of reading the counter must be used if you want to read the counter while it is still counting. The contents of the counter are transferred to a latch register in the 8254 after the input clock has rippled through the counter. The latch register is then read. If DIO%(0)

= 0 a direct read of the counter itself is performed. In this case the data in the counter must be static (counting ceased or inhibited by the gate) as it is possible to read erroneous data if the counter is counting. On return DIO%(1) contains the counter contents. Due to the 2s compliment storage, this will be a negative number if it is greater than 32,767 (see Appendix C). Finally, the change in count or number of pulses counted can be derived as follows:

```
xxx70  IF DIO%(1)<0 THEN CHANGE = -1-DIO%(1) ELSE CHANGE = 65535-DIO%(1)
```

7.3 USING THE TIMER-COUNTER ENABLE REGISTER IN CONJUNCTION WITH THE COUNTERS

The Timer Counter Enable register is a 2-bit Write Only register located at BASE ADDRESS +10. The register is described in more detail in Chapter 6. If the least significant bit, C0, is high, it allows IP0/TRIG0 to control the gates of Counters 2 and 3 (see the diagram at the end of this chapter). This provides a means of holding off trigger pulses to the A/D from the programmable timer until IP0 is taken high. If C0 is low, then IP0 has no control over the programmable interval timer.

The second bit, C1, controls the source of the clock input for Counter 0. If C1 = 0, then the external clock input, COUNTER 0 IN, is enabled. If C1 = 1 then Counter 0 is connected to a stable 100KHz internal crystal clock source. This is useful if Counter 0 is used for pulse width measurement, delay generation, frequency synthesis, or a secondary timer.

7.4 GENERATING SQUARE WAVES OF PROGRAMMED FREQUENCY

Both the Counter 1 and 2 combination and Counter 0 may be used to generate square waves of programmable frequency. With the C1 bit of the Timer Counter Enable register set high and Counter 0 clock input open or high, Counter 0 is internally connected to a 100KHz crystal signal source. Counter 0 can be operated in Configuration 3 (square wave generator) with a maximum divisor of 65,536. The lowest output frequency obtainable from Counter 0 directly will be about 1.5Hz (100000 / 65535). The minimum divisor can be as low as 2 to obtain a maximum output frequency of 50KHz.

The *DASG.BIN* driver will always set Counters 1 and 2 to the rate generator configuration on running initializing MODE 0, but you can use BASIC **OUT** statements to alter their operating configuration to the square wave configuration. The frequency range obtainable is identical to that obtained in the rate generator configuration. Counters 1 and 2 provide considerable flexibility in frequency range, a minimum division ratio of 4 (2 x 2) with a 10Mhz clock providing a 2.5MHz output at one extreme, and a division ratio of 232 (65,535 x 65,535) with a 1MHz clock providing an output of about 1 pulse/hour at the other extreme.

In practice, to obtain a symmetrical square wave, the divisor loaded into the counter should be an even number. If it is an odd number, one half of the square wave will be 1 input clock pulse period longer than the other half.

Calculating the divisor is straightforward. Assume you desire an output frequency of 1 KHz. The input frequency to the Counter 0 is 100Khz so you must divide this by 100 to obtain 1 Khz. Counter 0 should be set in configuration 3 and loaded with 100 as follows:

```
xxx10     OUT BASADR%+10, 2                 'enable 100KHz clock to ctr. 0
xxx20     MD% = 10                          'set timer counter configuration
xxx30     DIO%(0) = 3                       'select square wave configura-
                                            tion
xxx40     CALL DASG (MD%, DIO%(0), FLAG%)
xxx50     MD% = 11                          'setup to load counter 0
xxx70     DIO%(0) = 100                     'divisor = 100
xxx80     CALL DASG (MD%, DIO%(0), FLAG%)
```

Counter 0 output will now be outputting a 1KHz square wave.

7.5 MEASURING FREQUENCY AND PERIOD

The two previous sections show how to count pulses and output frequencies. It is possible to use the 8254 to measure frequency by raising the gate input of Counter 0 for some known interval of time, say 10, 100 or 1000ms and counting the number of pulses clocked into the counter for that interval. The gating signal can be derived from Counters 1 and 2 operating in square-wave mode. Also, the computer has to be informed about the start and finish of the measurement cycle, so one of the DAS-16 digital inputs can be used to monitor the gate input to achieve this requirement.

Counter 0 can also be used to measure pulse width or half period of a periodic signal. The signal should be applied to the gate input of Counter 0 and a known frequency applied to Counter 0 clock input (100 KHz) from the internal crystal. During the interval when the gate input is low, Counter 0 is loaded with a full count, 65,536. The gate input then goes high at the beginning of the measurement, and the counter decrements until the gate input goes low at the end of the pulse. The counter is then read and the change in the count is the duration of the gate input signal. If Counter 0 is fed with 10 microsecond duration clock pulses (100KHz), the maximum pulse duration that can be measured is 65,535*10 = 655 milliseconds. Longer pulse durations can be measured using Counters 1 and 2 as an input clock source for Counter 0. One of the digital inputs should also be connected to the gate of Counter 0 to synchronize the loading and reading operations.

7.6 GENERATING TIME DELAYS

Another use for Counter 0 is generating programmable time delays. There are several "one shot" modes that can be used. The counter configurations have the following characteristics when used for time delay generation:

Configuration 0 - Pulse On Terminal Count. After loading the counter the output goes low. Counting is enabled when the gate input is high and continues until the count reaches zero when the output goes high. The output will remain high until the counter is reloaded by a programmed command. Taking the gate input low during the count down will disable counting as long as it is low.

Configuration 1 - Programmable One-Shot. The counter need only be loaded once. The timing delay is initiated by the gate input going high. At this point the output goes low. If the gate input goes low, counting continues but a new cycle will be initiated if the gate input goes high again before the time out delay has expired (is re-triggerable). At the end of the time out, as the counter reaches zero, the output goes high and will remain high until re-triggered by the gate input.

Configuration 4 - Software-Triggered Strobe. This is similar to Configuration 0, except Software that after loading the output goes high and only goes low for one clock period on timing out. This produces a negative strobe pulse a programmed duration after loading the counter.

Configuration 5 - Hardware-Triggered Strobe. This is similar to Configuration 1 except that the time out is triggered by the gate input going high and the output is normally high, going low for one clock period on timeout and producing a negative going strobe pulse. Like configuration 1, the time out is re-triggerable (a new cycle will commence if the gate input is taken high before a current cycle has timed out).

Counter 0 is good for directly producing delays up to 650ms. If the output of Counter 0 operated in Configuration 0 is connected to IP0/TRIG0 with Counters 1 and 2 used as a programmable trigger source for the A/D (C0=1, C1=1), then after loading Counter 0 and taking the gate input high, a programmable delay occurs before the A/D receives trigger pulses. This is a way to "hold off" conversions for a period after receipt of an external trigger signal. Connections are shown in Figure 7-2.

7.7 TRIGGERING THE A/D PERIODICALLY

When you are using the A/D converter, one of the key uses for the 8254 programmable interval timer is in providing trigger pulses for starting the A/D to perform periodic samples.

You can set up any given output frequency by using MODE 17 to load Counters 1 and 2 with the required divisors. As an example, let us set up a trigger rate of 8.3KHz. First, work out the overall division ratio from 1MHz or 10MHz (depending on setting of TIMER jumper block):

1,000,000 / 8300 = 120.48 (for 1MHz clock)

The closest frequency obtainable will be:

1,000,000 / 120 = 8.333 KHz

Next, apportion the divisor between the 2 counters:

```
xxx10      DIO%(0) = 3                 'Counter 2 divider
xxx20      DIO%(1) = 40                'Counter 1 divider
xxx30      MD%=17
xxx40      CALL DASG (MD%, DIO%(0),
           FLAG%)
```

If we had used the 10MHz clock source we could get closer to the desired frequency of 8.300KHz with the limitation of integer division ratios of the counters:

10,000,000 / 8300 = 1204.8 (for 10MHz clock)

The closest frequency obtainable will be:

10,000,000 / 1205 = 8.299 KHz

Note that a counter cannot divide by one; the minimum divisor is two, and the maximum 65,536. Initialization MODE 0 sets Counters 1 and 2 to the Rate Generator configuration with a division ratio of 1,000. MODE 17 allows you to change the frequency (division ration) but does not alter the counter configuration.

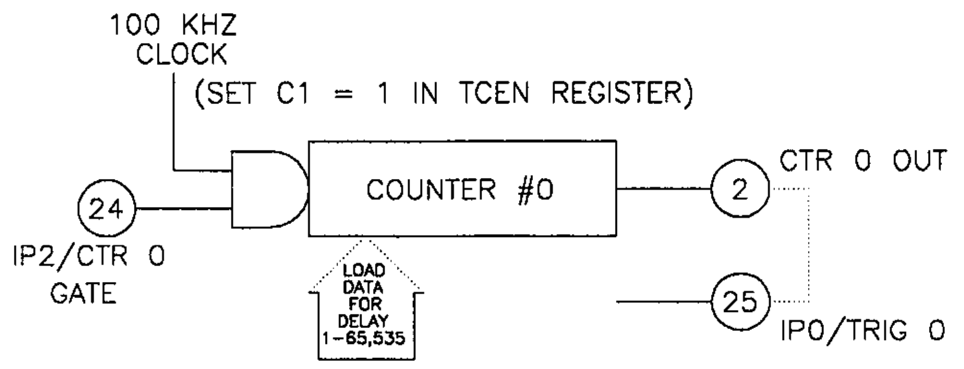


Figure 7-2. Using Counter 0 to Generate a Delayed Start

7.8 GENERATING INTERRUPTS WITH THE COUNTER/TIMER

DAS-16 architecture does not allow direct generation of an interrupt from the Counter/Timer. However, it is easy to set up the A/D to be triggered by the Counter/Timer and in turn have the A/D generate an interrupt at the end of its conversion cycle (a constant delay of 8-12 microseconds). This setup is performed through the DAS-16 control register. Indirectly, this accomplishes the desired result of generating a periodic interrupt from the timer, and you can then install any desired interrupt routine to service the interrupt. However, since interrupt service routines cannot be programmed directly in BASIC, you will need the DAS-16 driver source code listing. (Call Technical Support as instructed in Chapter 11.)

Note also that it is possible to trigger the A/D any other way (externally or by a programmed write to an I/O port) and invoke an interrupt at the end of A/D conversion in the same way.

CHAPTER 8
D/A CONVERTERS

The D/A channels consist of two separate double-buffered, 12-bit multiplying D/A converters. Each D/A may be used with the fixed -5V DC reference as a conventional 0 to +5V output D/A. Alternatively, the D/As may be operated with a variable, or AC, reference signal as multiplying D/As, the output is the product of reference and digital inputs. Accuracy remains at 12-bits up to 1KHz. The maximum output swing of the D/A's is ± 10V. A simplified diagram of each D/A channel is shown in the following diagram.

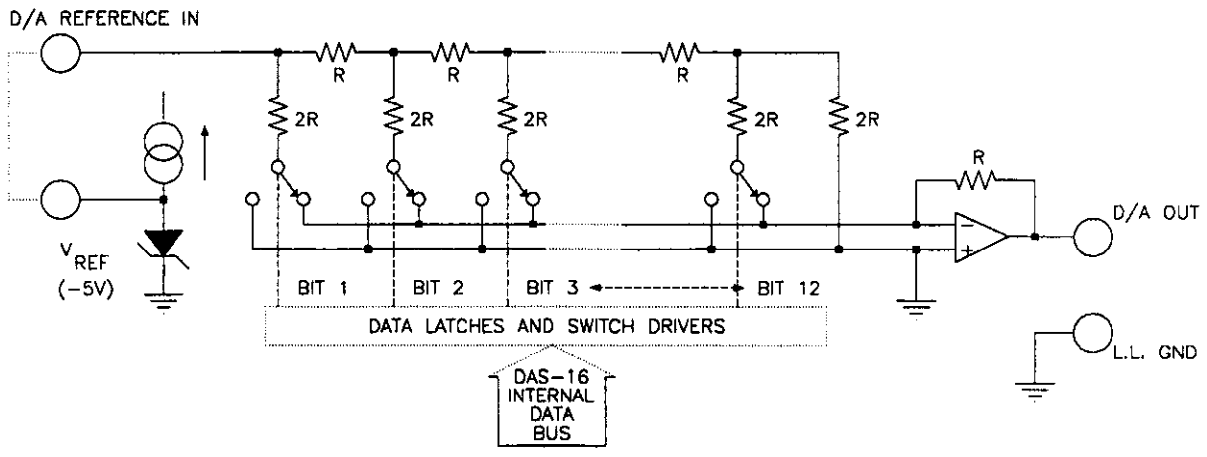


Figure 8-1. D/A Configuration and Connections

Since data is 12 bits, it must be written to each D/A in two consecutive bytes. The first byte is the least significant and contains the four least significant bits of data (See Chapter 6). The second byte is the most significant and contains the most significant eight bits of data. The least significant byte should be written first and is stored in an intermediate register in the D/A, having no effect on the output. When the most significant byte is written, its data is added to the stored least significant data and presented "broadside" to the D/A converter thus assuring a single step update. This process is known as double buffering. D/A register locations and data formats are as follows:

Table 8-1. I/O Address Map

Location	Function	Type
Base Address + 4	D/A 0 Low byte	W
Base Address + 5	D/A 0 Hi byte	W
Base Address + 6	D/A 1 Low byte	W
Base Address + 7	D/A 1 Hi byte	W

	D7	D6	D5	D4	D3	D2	D1	D0
Base + 4 or + 6	B3	B2	B1	B0	X	X	X	X
				LSB				

WHERE X = DON'T CARE

	D7	D6	D5	D4	D3	D2	D1	D0
Base + 5 or + 7	B11	B10	B9	B8	B7	B6	B5	B4
	MSB							

For 8-bit operations, first write zero to the low byte; all subsequent operations may then be performed with the high byte only.

Assembly language programmers should note that if data is left justified, 16-bit output operations may be used (for example, `OUT DX, AX`) as the data sequence is conventional Intel Low/High byte.

8.1 PROGRAMMING

Example 1. The following example shows how to output data in BASIC. It is readily "translatable" to other languages. Since the D/A's have 12-bit resolution, Data D should be in the range 0 - 4095 decimal. First, split the data into two bytes DL% (low) and DH%(high), as follows:

```
xxx00     DH% = INT(D/16)              'generate high byte
xxx10     DL% = D - 16*DH%             'derive remainder in low byte
```

```
xxx20       DL% = 16 * DL%                          'shift low nybble 4 places
                                                    left
```

Next write the data to the D/A. The example assumes D/A #0 with a board Base Address of Hex 300:

```
xxx30       OUT &H304, DL%                          'low byte
xxx40       OUT &H305, DH%                          'high byte and load
```

Example 2. An assembly language routine is even simpler. Assume AX contains the data and DX has the board I/O address. To write to D/A #0:

```
            MOV CL,4                                ;set up for 4 left shifts
            SAL AX,CL                               ;left justify data
            OUT DX,AX                               ;write to D/A #0
```

Example 3. A lengthy routine using BASIC OUT's can be avoided by using MODEs 15 or 16 of the CALL. MODE 15 outputs data to one D/A and MODE 16 is optimized to update both D/A's as simultaneously as possible (within 7 microseconds of each other). A typical output routine using the CALL would be as follows:

```
xxx10       MD% = 16                                'output to both D/A's
xxx20       DIO%(0) = 1234                          'D/A #0 data
xxx30       DIO%(1) = 2345                          'D/A #1 data
xxx40       CALL DASG (MD%, DIO%(0), FLAG%)
xxx50       IF FLAG%<>0 THEN PRINT "D/A load error"
```

8.2 USED WITH AN AC REFERENCE (DIGITAL ATTENUATOR)

Apart from its use as a standard DC output D/A, the D/As can be used with variable bipolar, AC, or DC reference signals. In this mode, they behave as a digitally programmed gain control or attenuator. The voltage output *Vout* is as follows:

$$Vout = -(Digital\ input) * Vref / 4096$$

Two additional parameters are of interest in AC operation. The first is feedthrough, the amount of residual signal at digital zero. The second parameter is the accuracy/frequency characteristic--it is a limit at a lower frequency. Feedthrough which is mainly a function of stray capacitance, rises with frequency; at 10KHz, it is typically 5mV peak-peak with a ±5V reference. Due to distributed capacitance in the R-2R ladder network of the D/A, the full 12 bit performance falls off as the frequency rises. Above about 1KHz the dynamic performance of the D/A will have less than 12-bit accuracy.

The D/A's will perform well in synchro-digital and resolver applications for sine/cosine generation with 400 Hz reference.

8.3 ARBITRARY WAVEFORM OUTPUT

One common requirement is to output a waveform from a D/A converter. At slow speeds this can be done with a timing loop in your program, but it is usually difficult to control the timing with any degree of precision especially when operating at more than a few points per second.

MODE 18 provides a method of outputting arbitrary data from an array using interrupts. The maximum rate at which data can be transferred is mainly dependent on the execution time of the interrupt service routine and the speed of the processor. The MODE 18 service routine can output in excess of 4000 points per second on a standard 4.77MHz 8088 processor. After an interrupt is asserted, there is a delay in starting to process the interrupt and this delay varies a little depending on the instruction that is being processed at the time or if a higher priority interrupt is being serviced or requesting service. In the PC, both the internal DOS timer on level 0 and the keyboard on level 1 have higher priority than any of the expansion bus interrupts (level 2-7). The latency or variation in the delay which can amount to several tens of microseconds can be reduced by suppressing other interrupts while outputting data from the D/A. *EXG18.BAS* shows how to do this for the timer interrupt which is the main offender. The latency variation causes a time jitter in the steps which is more noticeable at higher output frequencies.

A detailed description of the MODE 18 setup is in Section 4.4.19 and an example of the operation is provided in *EXG18.BAS*. The following description gives some idea of the steps involved in using MODE 18. For a typical example, let's choose to generate a 60Hz sinewave with 50 points per cycle. The first step is to set up the counter-timer to output a frequency of 60 * 50 = 3KHz to generate interrupts at the desired rate. Note that there is a limitation here due to the rate at which interrupts can be processed:

*Frequency * Number of steps <4,000*

The lower the frequency the more steps or points we can put in the waveform. Using a clock frequency of 10MHz, we can set Counters 1 and 2 to output a frequency of 3000.3Hz with a division Cratio of 3333. In turn with 50 points per cycle this would give us an output of 60.006 Hz fairly close to the desired 60Hz. Use MODE 17 to set the timer as follows:

```
xxx00     MD% = 17                        'set timer rate
xxx10     DIO%(0) = 3                     'counter 2 divider
xxx20     DIO%(1) = 1111                  'counter 1 divider
```

```
xxx30      CALL DASG (MD%, DIO%(0), FLAG%)
```

Next, we need to load an integer array with the table of output points. Since the array is outputting a 12-bit word to the D/A, we also have to do the scaling in this step. With the D/A input connected to the -5V internal reference, 0 corresponds to 0 volts and 4095 corresponds to the full scale of 4.9988 volts. The following code will dimension a 50-element integer array and load it with a sine wave centered around 2048 bits or +2.5 volts.

```
xxx40      N = 50                   'number of output points in waveform
xxx50      DIM DA%(N-1)             'dimension data array

xxx60      FOR I = 0 TO N-1
xxx70      DA%(I) = CINT(2048 + 2047 * SIN(2*3.14159*I/N))
xxx80      NEXT I
```

Note that in this case, we have loaded the array with a sinusoidal function, but we could just as easily have loaded it with any other set of values to generate any arbitrary wave shape. The next step is to output the array through one of the D/A channels using MODE 18, as follows:

```
xxx90      MD% = 18                          'mode number
xx100      DIO%(0) = 0                       'D/A channel, 0 or 1
xx110      DIO%(1) = N                       'number of data points
xx120      DIO%(2) = 0                       'number of waveform cycles
xx130      DIO%(3) = VARPTR(DA%(0))          'provide location of data array
xx140      DIO%(4) = 0                       'do not collect A/D data
xx150      CALL DASG (MD%, DIO%(0), FLAG%)
```

MODE 18 provides another example of data exchange directly with a BASIC array and the usual precaution of not introducing any additional simple variables in your program after the line that defines the array location (VARPTR in line xx130) applies. For more information, see Chapter 5.

(This page intentionally left blank.)

CHAPTER 9
APPLICATIONS

9.1 CONNECTING ANALOG A/D INPUTS CORRECTLY

DAS-16 offers optional switch-selected 8-channel differential or 16-channel single-ended input configurations. To select either option, set the CHANNEL CONFIG slide switch to 16 or 9.

A single-ended configuration is so-named because it has only one input relative to ground. A differential input has two inputs relative to ground; this configuration responds to the difference between them. Obviously, operating 16-channel, single-ended provides more input channels; but in this configuration, the inputs are suitable only for what are known as "floating" sources. A floating signal source does not have any connection to ground at the signal source and can be wired to the inputs as shown in Figure 9-1.

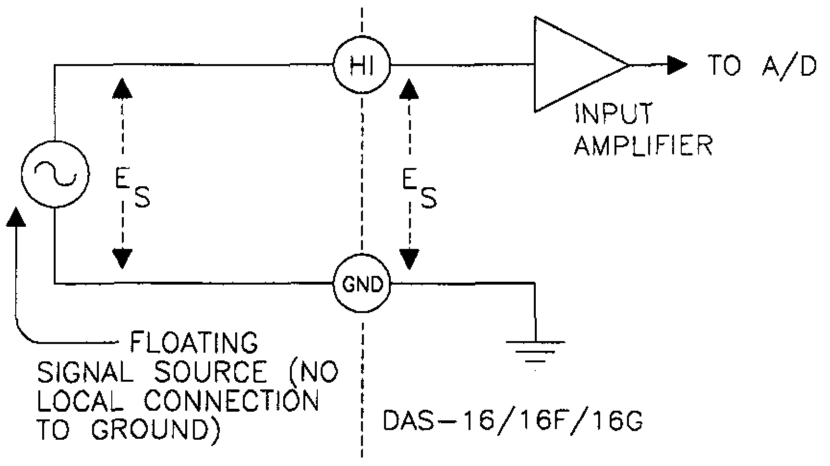


Figure 9-1. Connecting a Floating Signal Source to a Single-Ended Input

If the signal source has one side connected to a local ground, the 8-channel differential configuration should be used. A differential input responds only to difference signals between the high (HI) and low (LO) inputs. In practice, the signal source ground and the computer ground (where the DAS-16 is) will not be at exactly the same voltage as they are connected through the ground returns of the equipment and the building wiring. The difference between the ground voltages (See Figure 9-2) forms a common mode voltage; for example, a voltage common to both inputs (Vcm) that a differential input is designed to reject up to a certain limit. In the case of DAS-16, the inputs are designed to handle a common mode limit of ± 10 volts. If a ground referred signal source is wired to a single ended input (as shown in Figure 9-3), the difference in voltage between grounds ends up appearing in series with the input signal causing errors and noise and unreliable readings and forming what is known as a "ground loop". For this reason, you should avoid connecting a ground referred signal to a single ended input unless its ground is one and the same ground as the computer's.

If you have a combination of floating and ground referred signal sources, the 8-channel differential configuration should be used; the ground referred signals should be connected as shown in Figure 9-2 and the floating signals connected as shown in Figure 9-4. Note that the jumper between LO and LL GND effectively turns the differential input into a single-ended input.

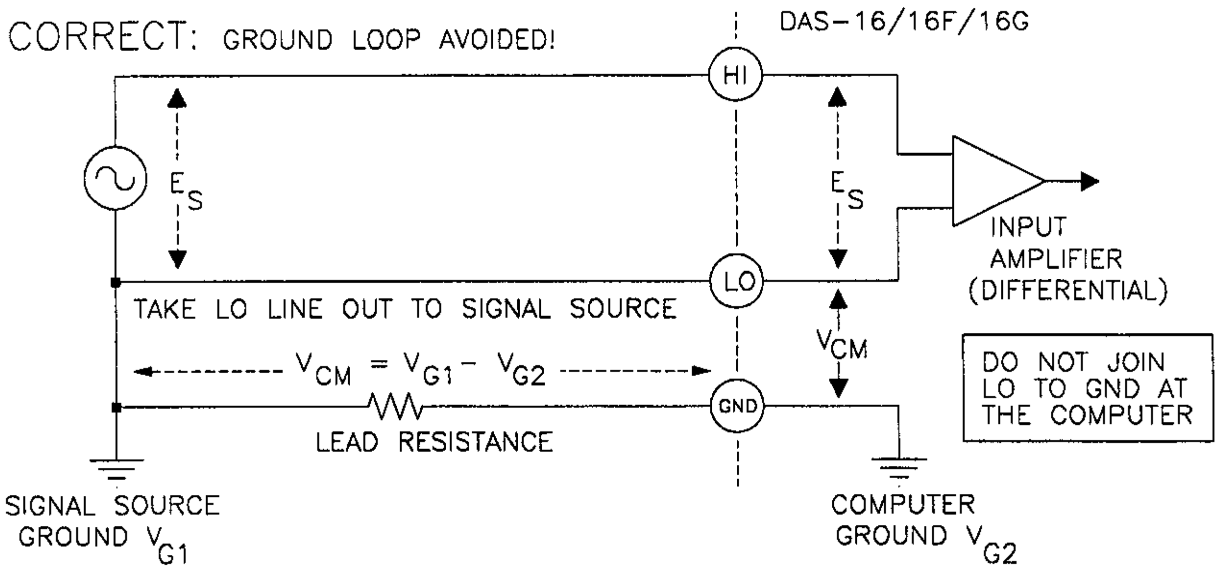

CORRECT: GROUND LOOP AVOIDED!
DAS-16/16F/16G
HI
E_S
E_S
INPUT AMPLIFIER (DIFFERENTIAL)
LO
TAKE LO LINE OUT TO SIGNAL SOURCE
$V_{CM} = V_{G1} - V_{G2}$
V_{CM}
GND
LEAD RESISTANCE
DO NOT JOIN LO TO GND AT THE COMPUTER
SIGNAL SOURCE GROUND V_{G1}
COMPUTER GROUND V_{G2}

Figure 9-2. Correct Method for Connecting a Grounded Signal Source

It is important to understand these distinctions between the input types, how to use them effectively, and how to avoid ground loops. Misuse of the inputs is the most common difficulty users experience in applying and obtaining the best performance from data acquisition systems. Figure 9-3 shows you how to not connect a grounded signal source.

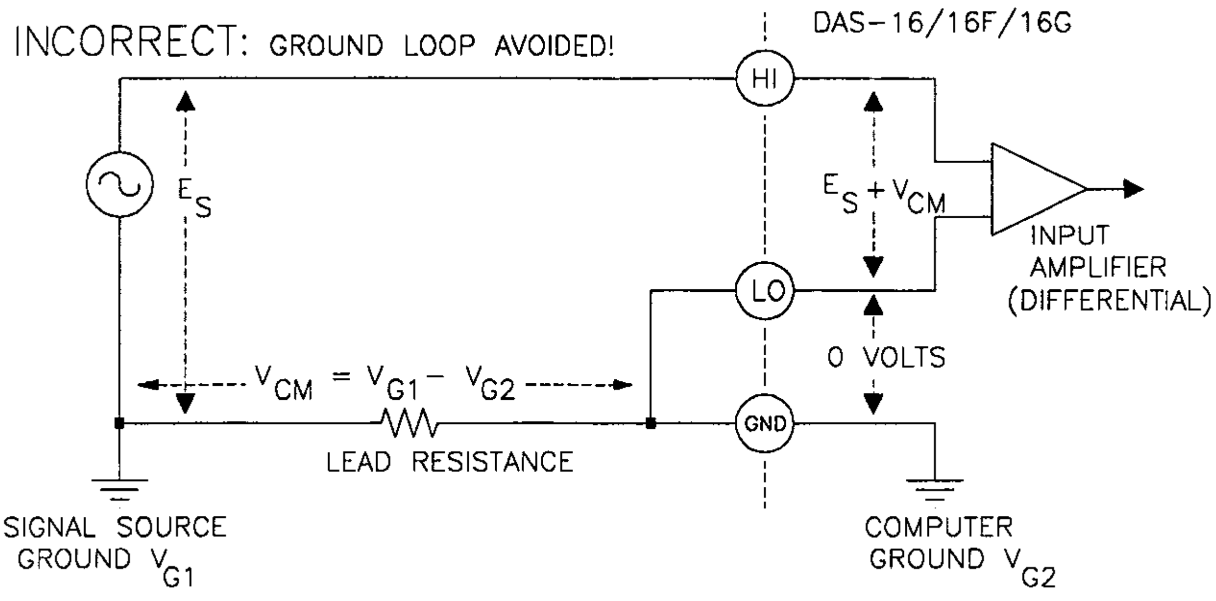

INCORRECT: GROUND LOOP AVOIDED!
DAS-16/16F/16G
HI
LO
GND
E_S
$E_S + V_{CM}$
INPUT AMPLIFIER (DIFFERENTIAL)
0 VOLTS
$V_{CM} = V_{G1} - V_{G2}$
LEAD RESISTANCE
SIGNAL SOURCE GROUND V_{G1}
COMPUTER GROUND V_{G2}

Figure 9-3. Incorrect Method for Connecting a Grounded Signal Source

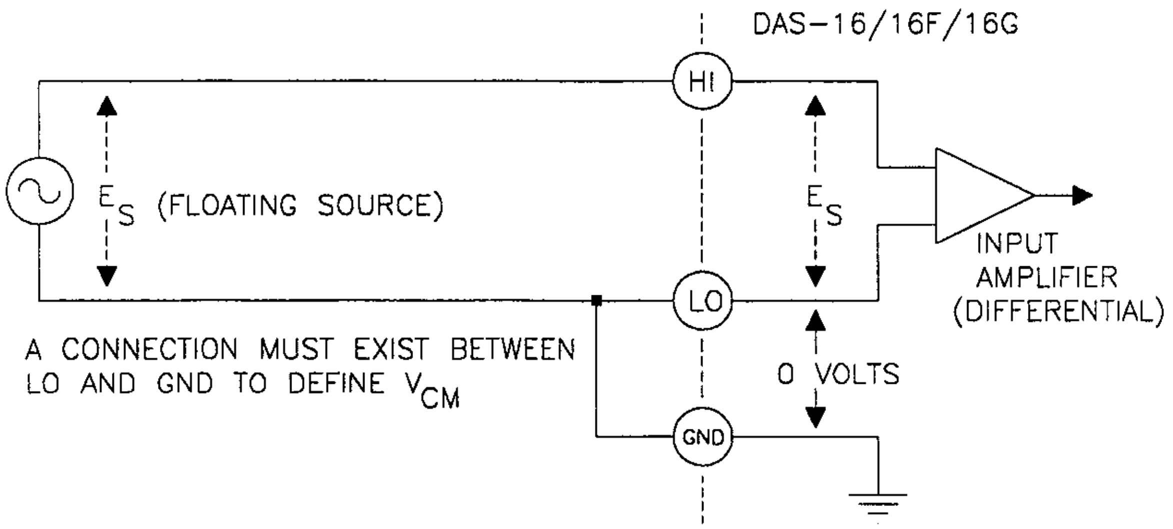

DAS-16/16F/16G
HI
LO
GND
E_S (FLOATING SOURCE)
E_S
INPUT AMPLIFIER (DIFFERENTIAL)
0 VOLTS
A CONNECTION MUST EXIST BETWEEN LO AND GND TO DEFINE V_{CM}

Figure 9-4. Connecting A Floating Source To A Differential Input

9.2 SELECTING INPUT CHANNEL RANGE AND GAIN

The DIP switch labelled GAIN controls the full-scale range common to all the channels. Instructions on setting this switch are provided in Section 2.2. The slide switch marked A/D has two positions: UNI (Unipolar) and BIP (Bipolar). These two switches determine the input scaling and whether the range is unipolar (zero to some positive full scale) or bipolar (from a negative to a positive full scale). The easiest method of selecting the correct settings is running the calibration program *CALF.BAS*. Selecting the A/D Converter option of the calibration program presents a menu of range options followed by a diagram of the switch showing your selection of positions.

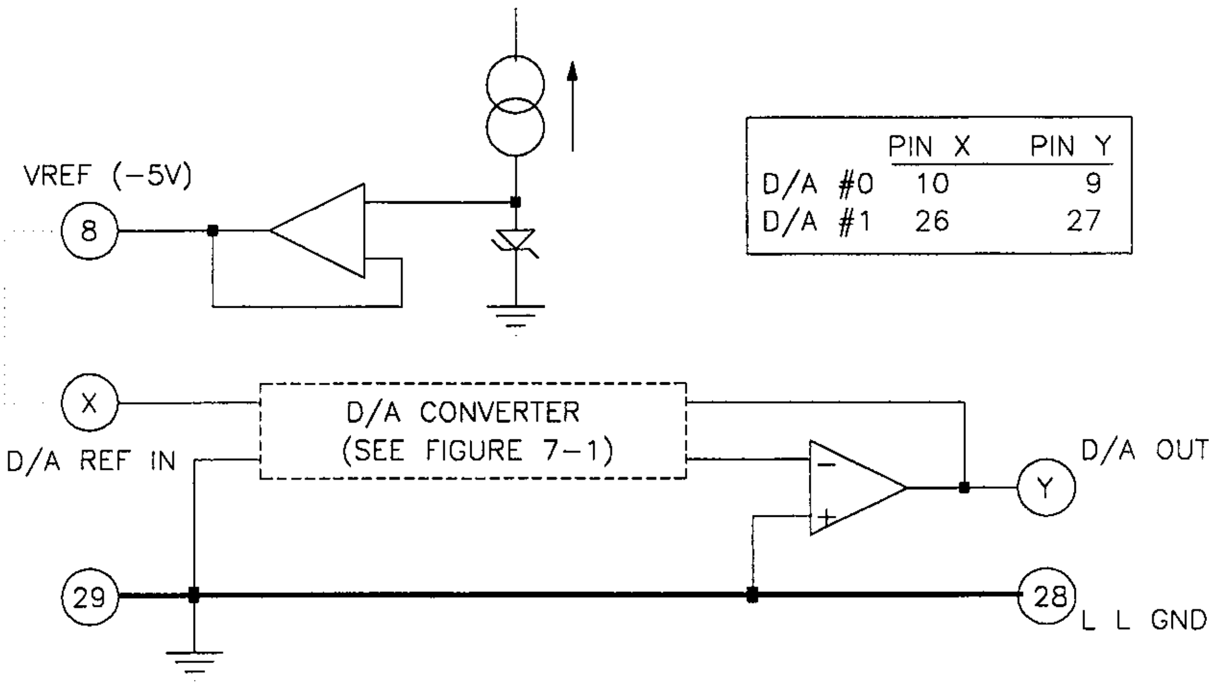
The *DASG.BIN* driver detects the position of the UNI/BIP switch through the DAS-16 status register, and it adjusts the data for unipolar and bipolar ranges accordingly. Unipolar scalings correspond to 0 to 4095 bits of output from the A/D, whereas bipolar scalings correspond to -2048 to +2047 bits. In this way 0 bits always corresponds to 0 volts, so the only operation usually required is a simple multiplication to scale the reading to real units; for example, on the ±5V or 0-10V range, multiply the integer data returned by the CALL by the actual bit weight (2.44 millivolts/bit) to obtain volts.

The DAS-16/16F has a 5-position DIP switch (marked A, B, C, D, USER) which is set according to the instructions given in Chapter 2. You can have a non-standard input scale by soldering a precision resistor into the USER location and selecting the USER position on the GAIN switch. Scale span (the difference between full-scale limits) is related to resistor value as follows: *Span (in volts) = 10 / (1 + 20,000/Ruser).* For example, Ruser = 1053 ohms gives 0.5V span.

Signals below about 0.5V volt are subject to system noise and should therefore be preamplified using an EXP-16 (or equivalent) before applying them to DAS-16 inputs).

9.3 D/A CONVERTER CONNECTIONS

Make the D/A output connections from the appropriate D/A output and LL GND, as shown in Figure 9-5.



	PIN X	PIN Y
D/A #0	10	9
D/A #1	26	27

Figure 9-5. D/A Converter Connections

The D/A reference inputs can be connected to the -5V reference signal on the rear connector. In this case, the D/A output scaling will be 0 - +5V. It is also possible to connect the reference inputs to an external reference signal whether it be AC or DC.

9.4 MEASURING CURRENT AND 4-20 mA CURRENT LOOPS

Process control current loop transducers are easily interfaced to DAS-16 by adding a suitable shunt resistor across the input. Since the maximum current will be 20mA, the precision shunt resistor should have a resistance in ohms of 50 * *Vfs*. The resistor should be of low-temperature-coefficient-metal-film or wirewound construction for stability with time and temperature. Using this interface, the 4-20mA working range of the current loop corresponds to 80% of the normal resolution, about 3,277 bits for unipolar ranges and 1,638 bits for bipolar ranges.

Non-ground referred currents outside the common-mode range can also be measured with a suitable shunt. Non-ground referred currents can be measured through an isolation amplifier, DC, or Hall Effect current transformer.

9.5 THE D/A VOLTAGE REFERENCE

A -5V stable reference voltage (Vref) is brought out for users. This voltage is derived from the A/D converter reference. Apart from its use as a D/A reference, it can also be used for offsetting signals, powering bridge transducers, etc. The maximum available output source/sink current is 5mA.

9.6 USING EXP-16s TO ADD MORE ANALOG INPUTS

You may add sub-multiplexers to any or all of the 8/16 analog inputs. An EXP-16 provides 16 channels per input plus a common high gain instrumentation amplifier. Up to 8 EXP-16s can be added to one DAS-16, providing a total of 128 channels. The sub-multiplexer address can be driven by digital outputs OP0-3 so that a typical scan would use Mode 3, 4, 5, or 6 and then increment the sub-multiplexer address using Mode 13 and repeat the scan on the next set of sub-multiplex channels, etc. An S-1600 adapter cable can be used to connect the first EXP-16 to the DAS-16. Additional EXP-16s can be added with C-1800 or C-1000 cables. When using EXP-16s with the DAS-16, the optional PG-408 DC/DC converters are required. Refer to Figure 9-6 for a diagram showing the use of additional analog inputs.

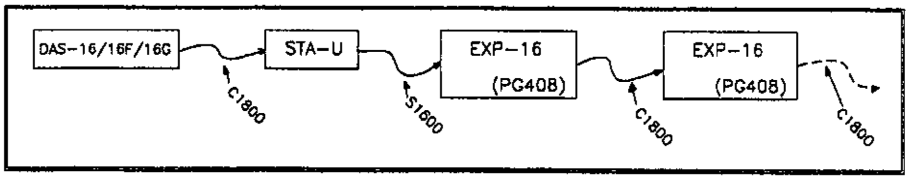


Figure 9-6. Adding Analog Inputs Using EXP-16s

When a sub-multiplexer is added, it is not possible to operate the DAS-16 in the fast DMA sampling mode due to having to drive the sub-multiplexer board address through the digital output port. However, it is often possible to partition your channels into some high speed direct inputs and some lower speed indirect inputs through the EXP-16. The EXP-16 is best suited to handling high gain low rate of change inputs such as thermocouples, pressure transducers etc. EXP-16 includes cold junction compensating circuitry for thermocouples, switch selectable gain from 0.5 to 1000 and 16 differential inputs.

EXP-16 cards are designed to cascade with flat cable and connectors. All analog channel connections are made by screw connectors, and each EXP-16 (group of 16 channels) can be operated at a different gain. In this way a system can be configured with a variety of different channel functions and gains, single-ended and differential.

9.7 INTERFACE TO TRANSDUCERS, THERMOCOUPLES, ETC.

Low level transducers such as thermocouples and strain gage bridges (load cells, pressure & force transducers) require amplification before applying to the high level DAS-16 inputs. The EXP-16 expansion multiplexer incorporates an instrumentation amplifier that can provide stable amplification and also includes circuitry that allows cold junction compensation of thermocouples. EXP-16 will handle most interfacing requirements to DC output transducers and also includes spaces for filters, shunts and attenuators.

For inexpensive temperature measurement in the -50 to +125° C temperature range, semiconductor temperature transducers are a good choice. The most popular types are the AD590/592 (Analog Devices) which behaves like a constant current source with an output of 273µA at 0 °C and a scaling of 1µA/°C and the LM335 (National Semiconductor) that has an output of 2.73 volts at 0 °C and a temperature coefficient of 10mV/°C.

For measuring high temperatures, up to 1800 °C or more, thermocouples are the most satisfactory solution. The base metal thermocouples (types J, K, T, and E) have outputs around 40 microvolts/°C while the platinum and tungsten types used for the highest temperature measurement. Thermocouple types S, B, and R tend to have lower outputs--in the 6-12 microvolt/°C range. A further complication encountered in the use of thermocouples is the "cold-junction" compensation. Where the thermocouple wire is terminated to the copper EXP-16 connections, an unwanted thermocouple junction is formed. As the connector temperature varies, this introduces an error. The error can be bucked out by sensing the connector temperature using a semiconductor sensor on another channel, and correcting the thermocouple readings in software. This is required only at the highest levels of accuracy, since in most cases connector temperature (usually room temperature) varies little. EXP-16 provides the sensing hardware needed to perform this correction.

9.8 PRECAUTIONS: NOISE, GROUNDLOOPS, AND OVERLOADS

Unavoidably, data acquisition systems give users access to inputs to the computer. Do NOT, whatever else you do, get these inputs mixed up with the AC line. An inadvertent short can in an instant can cause extensive and costly damage to your computer. The manufacturer can accept no liability for this type of accident. As an aid to avoiding this problem:

1. Avoid direct connections to the AC line.

2. Make sure that all connections are tight and sound so that signal wires are not likely to come loose and short to high voltages.

3. Use isolation amplifiers and transformers where necessary. There are two ground connections on the rear connector called POWER GND and LL GND. Power ground is the noisy or "dirty" ground that is meant to carry all digital signal and heavy current (power supply) currents. Low level ground is the signal ground for all analog input functions. It is meant to carry only signal currents (less than a few mA) and is the ground reference for the A/D channels. Due to connector contact resistance and cable resistance, there may be many millivolts difference between the two grounds although they are connected to each other and the computer and power line grounds on the DAS-16 board.

9.9 USING THE DAS-16G1 AT HIGH GAINS

Operating the DAS-16G1 at gains of 100 or 500 can lead to problems if you are unable to cope with noise. At a gain of 500, each bit of A/D output corresponds to 10 microvolts of analog input. Thus, with the high speed and bandwidth of this board, analog noise and performance degradation come easily, unless you take precautions to avoid them. The following collection of ideas and suggestions is aimed at avoiding these problems.

1. Operate the DAS-16G1 in 8-channel differential mode. Using the board in 16-channel, single-ended mode at high gains introduces enough ground-loop noise to produce extremely large fluctuations in reading (10s to 100s of counts).

2. Shorting Channel LO, Channel HI, and L.L. GND on the back connector should cause fluctuations of no more than 2-3 bits at a gain of 500 (fluctuation is almost non-existent at a gain of 100). This is the "best" performance level that can be obtained directly from the DAS-16G1; as soon as you begin to add cabling, connect to signal sources, etc., you begin to degrade this "best" performance capability.

3. The normally furnished flat cables and screw-connector boards (for example, C-1800 and STA-16) tend to degrade noise performance through crosstalk and induced-voltage pickup. To obtain best performance at high gains, wire directly from the rear connector using a solder-type D connector (such as Part No. SFC-37) and shielded cable. Connect the shield to L.L. GND and the inner conductors to Channel LO and HI. Channel LO and L.L. GND should have a DC return (or connection) at some point; this return should be as close to the signal source as possible. Induced noise from RF and magnetic fields can easily exceed many 10s of microvolts, even on one- or two-foot cables; shielded cable eliminates this problem.

4. The common-mode rejection of the AD-625 programmable gain instrumentation amplifier is greater at high gains; typical values are 80dB at gain = 1, 100dB at gain = 10, and about 120dB at gains = 100 and 500. However, these values are measured at DC and decline rapidly as the frequency of common-mode voltage increases. If you apply a ±10V common-mode signal to a DAS-16G1 operating at a gain of 500, it produces a reading change of barely one or two counts; increasing frequency to 60 Hz is likely to cause a reading change of 10 counts or more, while increasing to 1 KHz produces 100s of counts of noise.

5. Avoid bimetallic junctions in the signal-input circuitry. For example, the kovar leads used on reed relays typically have a thermal emf to copper of 40 microvolts/°C. Thermals can introduce strange random variations due to air currents, etc.

6. When all else fails, consider filtering. This approach can use hardware (resistors, capacitors, etc.) but is often accomplished more easily with software. Instead of reading the channel once, read it 100 or more times in quick succession, and average the readings. If the noise is random and gaussian, it will be reduced by the square-root of the number of readings. Do this with the DAS-16G1 utility routines, as follows:

```
100   MD% = 1: D%(0) = 3 : D%(1) = 3  'select a channel (such as 3).
110   CALL DASG (MD%, D%(0), FLAG%)
120   X = 0                           'initialize averaging variable.
130   N% = 100                        'number of readings to average.
140   MD% = 3                         'do one A/D conversion mode.
150   FOR I% = 1 TO N%
160   CALL DASG (MD%, D%(0), FLAG%)   'do an A/D conversion.
170   X = D%(0)                       'running total (note X is real
to
                                      'exceed 32767).
180   NEXT I%
190   X = X/N%                        'X = average of N% readings.
```

This technique is used in the "filter" function of the CALG and CALGA calibration routines; its effect is easy to examine by running these programs.

(This page intentionally left blank.)

CHAPTER 10
MAINTENANCE AND REPAIR

10.1 GENERAL

This chapter describes calibration and other maintenance procedures. User serviceable parts are also described. Should your board require factory repair, return to factory and warranty information is provided in Chapter 11.

10.2 CALIBRATION AND TEST

Periodic recalibration of the DAS-16 is recommended to retain full accuracy. The recalibration interval depends to a large extent on the type of service that the board is subjected to. For an environment with frequent large changes of temperature and/or vibration, a three-month recalibration interval is recommended. For laboratory or office conditions, six months to one year is acceptable.

Satisfactory calibration requires a digital multimeter with a display of at least 4 1/2 digits. In addition, a voltage calibrator or a stable noise free DC voltage source that can be used in conjunction with the digital multimeter is required. A card extender is an inexpensive device that will greatly improve accessibility to the DAS-16 for maintenance and calibration (and could be useful with other boards, as well). The DAS-16 distribution software contains calibration programs in BASIC and QuickBASIC (refer to *FILES.DOC* to determine the appropriate calibration program for any particular model). These programs lead you through the calibration and set up procedure with a variety of prompts and graphic displays showing you settings and adjustment trimpots. The explanatory material in this section is brief and is intended for use in conjunction with the calibration programs.

The calibration programs include tests of the A/D and D/A functions and can aid in troubleshooting if problems arise. These programs also include simple tests for the digital I/O and programmable timer.

Figures 10-1 and 10-2 show the location of various switches and controls which are used in the calibration process.

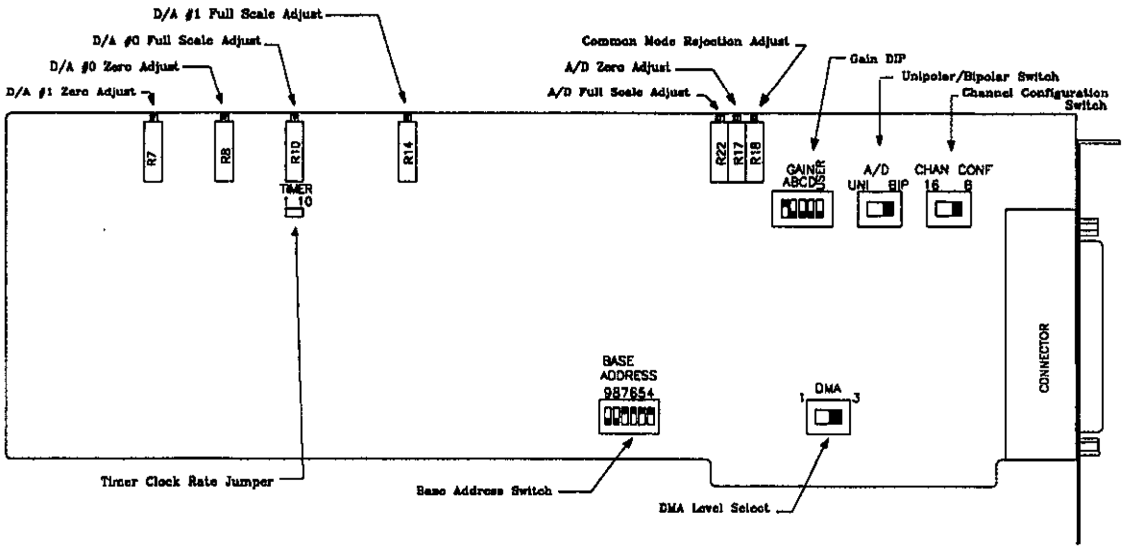


Figure 10-1. DAS-16/16F Switch and Control Locations

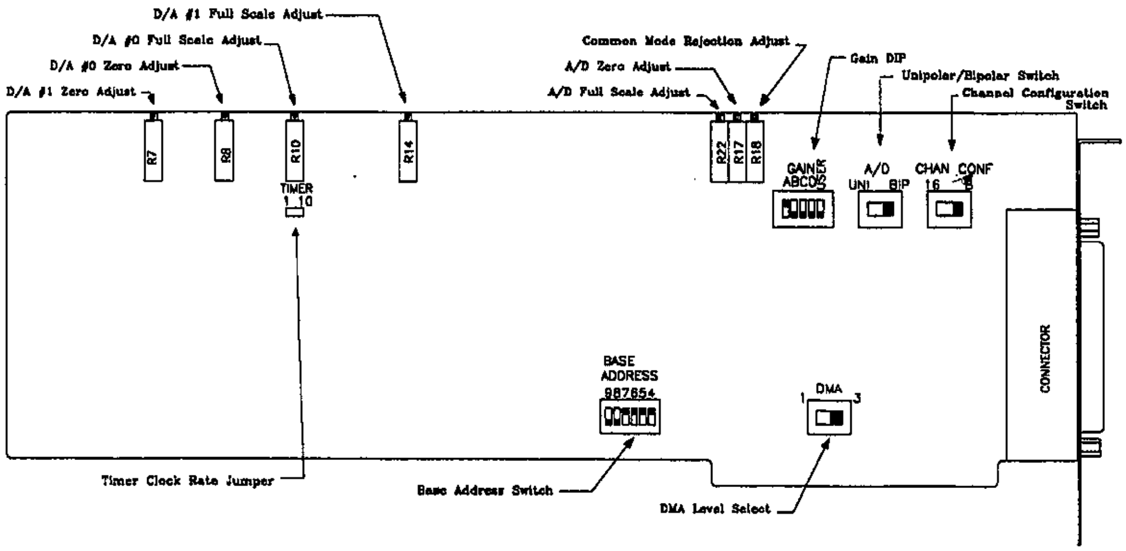


Figure 10-2. DAS-16/16G Switch and Control Locations

10.3 CALIBRATING THE A/D

The A/D output should be observed (the calibration programs do this) while applying a known calibration voltage to any or all analog input channels. The A/D should be calibrated on the range you are going to use since calibration on one range does not insure correct calibration on any other range. Briefly, the adjustment sequence is as follows:

1. Apply an analog input of 1/2 bit above -F.S. and adjust the A/D zero pot so that the output flickers between -2048 and -2047 (bipolar) or 0 and 1 (unipolar).

2. Apply an analog input of +0.0v (bipolar) or 1/2 of F.S. unipolar and adjust the A/D full-scale pot for a reading of 0 or 2048, respectively. Other inputs such as 1/4 and 3/4 full scale can be applied to check the linearity.

All DAS-16 models contain an adjustment to optimize the common-mode rejection of the input amplifier. Make this adjustment only in the 8-channel differential mode, and connect the "Hi" and "Lo" of a channel as shown in Figure 10-2. The voltage calibrator should be varied from zero to + or - 10 volts and the pot adjusted for no change in the reading of the selected channel. Any channel may be used for this test; the calibration programs use Channel 0.

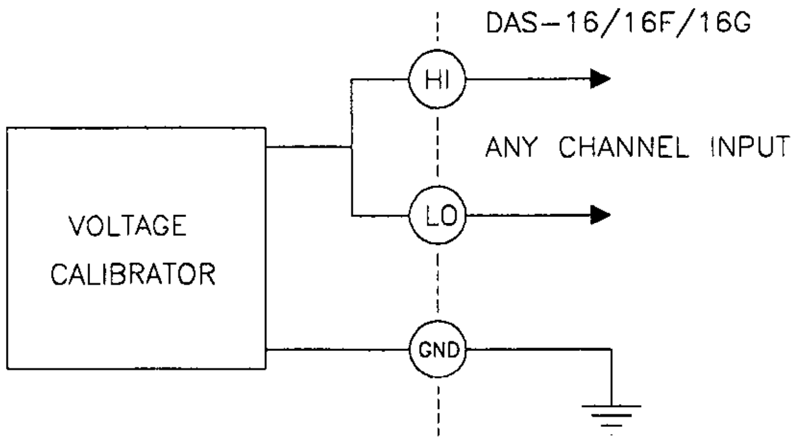


Figure 10-2. Common Mode Adjustment Test Configuration

10.4 CALIBRATING THE D/As

Connect the D/A reference inputs to the -5V reference voltage and then, using the calibration program, make the following adjustments to each D/A. Connecting the digital voltmeter to each D/A output in turn, set the voltages to within ±0.0005V (1/2 millivolt) of the recommended values, which are shown in Table 10-1.

Table 10-1. D/A Recommended Calibration Values

D/A #	DIGITAL INPUT	ADJUST POT	ADJUST OUTPUT TO
0	0	D/A 0 zero	0.0000V
0	4095	D/A 0 F.S.	+4.9988V
1	0	D/A 1 zero	0.0000V
1	4095	D/A 1 F.S.	+4.9988V

10.5 PROGRAMMABLE TIMER AND DIGITAL I/O

No special calibration for the counter/timer or digital I/O is required, but the calibration program makes a check of the functionality of Counters 1 and 2 and the internal crystal clock by performing an accumulated-count test against the system clock.

The calibration program also lets you read and write to the digital I/O. The functionality of each output can be checked by grounding inputs and using a voltmeter or a logic probe on outputs.

10.6 USER SERVICEABLE PARTS

Most of the components vulnerable to damage by external shorts, transients, overloads, etc. are in sockets. These parts are serviceable simply by unplugging and replacing with a new part. The manufacturer can service your board; but, if you wish to make simple repairs for yourself, Table 10-2 provides a guide to locating and fixing faults.

Table 10-2. Troubleshooting Guide

Symptom	Action to be Taken
Board located at wrong I/O address	Error code received on initialization usually indicates that the BASE ADDRESS switch setting does not correspond to the address used in your program. Run *INSTALL.EXE,* check that the *DAS16.ADR* file is correct. If there is no discrepancy, try another address. As a final resort, assume the board has a serious hardware fault and is to be returned to the manufacturer for repair.
A/D Converter non-operational, Non-linear, Missing Bits, Excessive Noise	All channels dead, Error Code 9 in Mode 3. First check that other functions on the board are working (digital I/O, control register, etc. If other functions are OK, replace: •U37, with HI-674AJD on the DAS-16 •U37, with HI-774JD on the DAS-16F •U21, with HI-674AJD on the DAS-16G1/G2
Missing Channels, Opens, and Shorts	Suspect multiplexer chips. Replace: •U40 and U46 with HI-508A-5 on the DAS-16/16F. •U23 and U15 with HI-508A-5 on the DAS-16G1/G2.
Gain Wrong, Drift, Excessive Noise, Unable to Zero	All these symptoms can be due to a faulty input instrumentation amplifier. Replace: •U31, U48, or U49 with HA3-2505-5 on the DAS-16/16F. •U14 with HA3-2505-5 on the DAS-16G1/G2.

Table 10-2. Troubleshooting Guide

Symptom	Action to be Taken
D/A converters Missing Bits	D/A #0, replace D/A U18, AD7548KN and/or output amplifier U13, OP-07DN. •D/A U20, AD7548KN and/or output amplifier U11, OP-07DN on the DAS-16/16F. •D/A U19, AD7548KN and/or output amplifier U10, OP-07DN on the DAS-16G1/G2.
Non-linearity, No Output, Missing Bits	D/A #1, replace: •D/A U19, AD7548KN and/or output amplifier U12, OP-07DN for the DAS-16/16F. •D/A U19, AD7548KN and/or output amplifier U10, OP-07DN for the DAS-16G1/16G2.
Digital Inputs	Faulty or blown inputs. Replace: •U43, Part #74LS244N and/or (for counter and triggering control) U27, #74LS132N on the DAS-16/16F. •U29, Part #74LS244N and/or (for counter and triggering control) U24, #74LS132N on the DAS-16G1/G2.
Digital Outputs	Faulty or blown outputs. Replace: •U44, Part #74LS197N on the DAS-16/16F. •U37, Part #74LS197N on the DAS-16G/16G1.

Table 10-2. Troubleshooting Guide

Symptom	Action to be Taken
Counter/Timer	Replace: • U20, Intel 82C54-2 or equivalent on the DAS-16/16F. • U18, Intel 82C54-2 or equivalent on the DAS-16G1/16G2.
Voltage Reference	Replace:. • U30, LF444CN on the DAS-16/16F. • U12, LF442CN on the DAS-16G1/16G2.

(This page intentionally left blank.)

CHAPTER 11
INSTRUCTIONS FOR RETURNS

Before returning any equipment to the factory for repair, you must first call the Technical Support Department at (508) 880-3000. They will try to diagnose and solve your problem over the phone. If they ascertain that the unit has to be returned to the factory for repair, they will issue a Return Material Authorization (RMA) number. Note that if the board is to be repaired under warranty, the Technical Support Department will need your invoice number and the date the board was purchased. Please reference the RMA number on any correspondence regarding the board.

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

1. A brief description of the problem.

2. Your name, address, and telephone number.

3. The invoice number and the date when the board was purchased.

4. Repackage the board in its original anti-static wrapping (Handle it with ground protection.) and ship it back to:

Repair Department
Keithley MetraByte Corporation
440 Myles Standish Boulevard
Taunton, Massachusetts 02780

Telephone:(508)880-3000
Telex:503989
FAX:(508)880-0179

Be sure to reference your RMA number on the outside of the package!

This page intentionally left blank.

APPENDIX A SPECIFICATIONS

Power Consumption

+5V Supply:	800mA typical, 1A max.
+12V Supply:	13.5mA typical, 18mA max.
-12V Supply:	23.5mA typical, 31mA max.

Analog Input Specifications

Channels:	8 differential (HI/LO/GND) or 16 single-ended (HI/GND) switch-selectable.
Resolution:	12 bits.
Accuracy:	0.01% of reading ± 1 bit.
Input Range:	*DAS-16/16F*

±10V, ±5V, ±2.5V, ±1V, and ±0.5V; or 0-10V, 0-5V, 0-2V, and 0-1V switch-selected.

DAS-16G1

Gain	Unipolar	Bipolar	Throughput
1	0 to +10 V	±10V	70KHz
10	0 to +1 V	±1V	60KHz
100	0 to +100 mV	±100mV	50KHz
500	0 to +20 mV	±20	30KHz

DAS-16G2

Gain	Unipolar	Bipolar	Throughput
1	0 to +10 V	±10V	70KHz
2	0 to +5 V	±5V	60KHz
4	0 to +2.5 mV	±2.5V	60KHz
8	0 to +1.25 mV	±1.25V	60KHz

Note that the GAIN ranges of both DAS-16G models have a 12-bit resolution (4096 steps), so that resolutions of the DAS-16G1 vary from 4.8mV/bit (±10V) to 4.8µV/bit (0 to +20mV), and resolutions of the DAS-16G2 vary from 4.8mV/bit (±10V) to 305µV/bit (0 to +1.25V). The programmable gain amplifier is a fixed-gain bandwidth product so that its settling time increases at higher gain. This causes the maximum throughput rate to decrease as the gain increases and is especially noticeable on the high-gain Model DAS-16G1.

Overvoltage	Continuous, single-channel to ±35V.
Input Current	*DAS-16/16F:* 250nA max (125nA typ.) at 25°C. *DAS-16G:* ±50nA max. at 25°C.
Temperature Coefficient	*DAS-16/16F* Gain or Full Scale: ±25ppm/°C max. Zero: ±12ppm/°C max. *DAS-16G* Gain or Full Scale: ±15ppm/C max. Zero: ±3ppm/°C max.

A/D Specifications

Type:	Successive approximation.
Resolution:	12 bits.
Coding:	Offset binary (bipolar ± inputs). True binary (unipolar ± inputs).
Conv. Time:	DAS-16/16G: 15ms max. (12 typ.). DAS-16F: 8.5ms max.
Type:	DAS-16/16G: Harris HI-674A or equivalent. DAS-16F: Harris HI-774.
Monotonicity:	Guaranteed over operating temperature range.

Linearity:	± 1 bit.
Zero drift:	± 10ppm/deg. C max.
Gain drift:	± 30 ppm/deg. C max.
Trigger Source:	By external trigger, programmable timer source, or program command; software selectable.

Sample Hold Amplifier

Acq. Time:	1ms to 0.01% typical for full scale step input.
Aperture Uncertainty:	0.3ns typical.
Type:	Harris HI-5320 or Signetics NE5060.

Reference Voltage Output

Ref. Voltage:	-5.0V ± 0.05V.
Temp. Coef.:	± 30 ppm/°C max.
Load current:	± 5mA max.

D/A Converters

Channels:	2 independent.
Type:	12-bit, multiplying, double-buffered. Analog Devices AD-7548.
Linearity:	± 1/2 bit.
Monotonicity:	± 1/2 bit.
Output Range:	0 to +5V with fixed -5V reference. May also be used with other ac or dc reference input. Maximum output limit ±10V.
Output Drive:	5mA min.
Output Resistance:	<0.1 ohm.
Ref. Input Range:	10V.

Gain:	-1.000 adjustable 1%
Settling time:	30ms to 0.01% for full-scale step.

Digital I/O

Output Port:	4 bits.
OP0-3 Out, Low V:	0.5V max at Isink = 8.0mA.
OP0-3 Out, High V:	2.4V min at Isource = -0.4mA.
Input Port & Trig:	4 bits.
IP0-3 Input, Low V:	0.8V max.
IP0-3 In, Low Curr.:	-0.4mA max.
IP0-3 In, High V:	2.0V min.
IP1-3 In, High Curr.:	20mA max @ 2.7V.

Interrupt Channel

Level:	2 - 7 software-selectable.
Enable:	Via INTE of CONTROL register. Interrupts on the DAS-16/16F/16G are latched in an internal flip-flop whose state corresponds to the INT bit in the STATUS register. A write to the STATUS register clears the flip-flop. Service routines should acknowledge and re-enable the interrupt flop.

Direct Memory Access Channel

Levels:	1 or 3 switch-selected.
Enable:	Via DMA bit of CONTROL register.
Termination:	By interrupt on T/C (or auto-initialize).
Transfer:	DAS-16: Hardware capable of up to 70,000 conversions/sec. DAS-/16F: Hardware capable of up to 100,000 conversions/sec. DAS-16G: Hardware capable of up to 70,000 conversions/sec. Users are responsible for initialization of the 8237 DMA controller on system board. With DMA bit set, double-byte requests are generated at the end of each A/D conversion. Data is latched and available for DMA transfer until end of following conversion. Transfer sequence is low byte/high byte.

Programmable Timer

Type:	82C54-2 programmable interval timer.
Counters:	Three down counters, 16-bit. Two permanently connected to 1 or 10MHz clock as programmable timer. One counter free.
Output Dr. Capability:	2.2mA @ 0.45V (5 LSTTL loads).
Input Gate:	TTL/DTL/CMOS compatible.
Clock Input Freq.:	DC to 10MHz.
Active Count (Edge):	Negative.
Min Clk Pulse Widths:	30ns high, 50ns low.
Timer Range:	2.5MHz to <1 pulse/hr.
	For additional information on programming, see Chapter 5.

Power Outputs

IBM PC Bus Sup:	+5V.
Tolerance:	+5V±5%.
Loading:	Dependent on other peripherals and on computer and computer type.

General Environmental

Op. Temperature:	0 to 50°C.
Storage Temp.:	-20 to +70°C.
Humidity:	0 to 90% non-condensing.
Weight:	9 oz. (255 gm).

APPENDIX B
FILE I/O COMMAND ERROR MESSAGES

Error No.	Description	Corrective Action
750	Unknown command encountered...	Check command syntax and/or spelling.
751	Unknown keyword following REad command	Keyword encountered following a **REad** command is not allowed. Check command syntax and/or spelling.
752	Unknown keyword following command.	Keyword encountered following a **REad** or **SEt** command is not allowed. Check command syntax and/or spelling.
753	Square Wave Output active.	Attempt is made to change the Square Wave Rate while the Square Wave Output is already active. Stop the Square Wave Output by issuing a **SEt WAve STOp** command.
755	Illegal keyword following SHow command; use AIo, DIo, FIle.	The **SHow** command may be followed by one of the three listed keywords, or by the numbers 0, 1 or 2; where 0 = AI, 1 = DIo, and 2 = FIle. Note that when used without an argument, SHow 0 is assumed. Check the SHow command.

Error No.	Description	Corrective Action
756	Expecting channel number following command.	The channel number is omitted following a `SEt [STArt, STOp]`,`SEt GAin`, or `REad GAin` commands. Specify a valid channel number. Valid channel numbers are 0 - 7 for Differential input, and 0 - 15 for Single Ended.
757	Illegal channel number specified.	The channel number specified following a `SEt [STArt, STOp]`, `SEt GAin`, or `REad GAin` command is out of range. Specify a valid channel number. Valid channel numbers are 0 - 7 for Differential input, and 0 - 15 for Single Ended.
758	Missing or illegal Gain Code; use 0, 1, 2 or 3.	The gain code specified following a `SEt GAin` command is missing or illegal. Note that this command is valid for the DAS-16G1 and G2 boards only. Check the Gain Code used.

Error No.	Description	Corrective Action
760	D/A or A/D conversions active!	An attempt is made to change a setup condition while an A/D or D/A operation is in progress. Data acquisition must terminate before this command is allowed! Stop operation using `SEt [AD, DA]` command; or wait for normal terminations.
761	Value out of range; use 0 - 4095 inclusive.	DAC value specified is out of range. Specify DAC value within 0 - 4095.
762	Output byte is out of range; use 0 through 15.	Digital value specified is out of range; only four digital output lines are available. Specify value between 0 and 15.
763	Illegal keyword following WAve command; use STArt, STOp or RAte.	The `SEt WAve` command must be followed with one of the keywords shown. Specify: `STArt`, `STOp` or `RAte` following command.
764	Illegal Number following INTerrupt command; use 2, 3, 4, 5, 6 or 7.	Specify: 2, 3, 4, 5, 6 or 7 following command.
765	Frequency/Rate value out of range.	The specified Internal clock frequency or square wave rate is out of range. Specify a conversion frequency within the range 1 to 99999 Hertz or a wave output rate between 1 and 25000 Hertz.
766	Illegal keyword following REad AD.	Specify: STatus, COunt, SAmples, SEgment or OFfset following `REad AD` command.

Error No.	Description	Corrective Action
767	Board Re-initialization failed...	Attempt to re-initialize the board using the `SEt INIt` command failed. Check board installation. Check the board's base address.
768	Analog-Level trigger requires: Chan Code +/-	Command to specify an analog external trigger is incomplete. Specify in the form: `{SEt} TRigger [INt, EXt {Ch Co +/- {gn}}]`.
769	Error in specifying the TRAnsfer command.	Command to transfer A/D data from the internal buffer to a user array is incomplete. Check and re-specify the `TRAnsfer` command in the form: `SEt TRAnsfer [-1, Seg] Offset Count`.
770	Illegal keyword following REad TRigger.	Command to read an external analog trigger parameter is incomplete. Specify: CHannel, LEvel, SLope or GAin following command.
771	External Analog Trigger not active!	Attempt to read an external analog trigger parameter while a Digital Trigger is in effect. Set external analog trigger, and repeat.

Error No.	Description	Corrective Action
772	Can not transfer A/D value(s); current data is bad.	Attempt to **REad AD VAlue** or TRansfer data is aborted due to the current A/D data being "bad". A/D data goes "bad" for various reasons, including: 1. Channel gain is changed 2. Scan limits changed 3. A/D aquisition type changed, etc. Acquire new data.
773	Can not set frequency; clock is external!	Attempt to set the internal clock frequency is aborted due to External conversion clock source is in effect. Set the Clock source for internal first.
774	Gain not programmable on DAS-16/F boards!	Attempt to change the A/D channel gain(s) is aborted for DAS-16 and 16F boards. This command is allowed on DAS-16G1 and G2 only. Unload driver. Set user selectable gain thru DIP switches on board. Reload the driver and specify selected gain using the command line switch /G= or thru the CONFIG16 utility and the /F= switch.

Error No.	Description	Corrective Action
775	FIle command must be followed by: NAme, MOde or FOrmat.	Specify: NAme, MOde or FOrmat.
776	FIle MOde command must be followed by: NEw, OVerwrite or APpend.	Specify: NEw, OVerwrite or APpend.
777	FIle FOrmat command must be followed by: BInary or AScii.	Specify: BInary or AScii.
778	File Name expected!	Specify file name following: SEt FIle Name command.

APPENDIX C
CALL MODE ERROR MESSAGES

If the FLAG% variable is returned non-zero, an error has occurred in the input of data to the CALL routine. Data checking occurs first in the routine; no action will be taken if an error exists. An immediate return takes place with the error specified in the FLAG% variable. The only exception to this rule is Error #22 (hardware failure or installation error), where an attempt is made to initialize the hardware so that other MODEs may be used to diagnose the problem. Error detection after the CALL routine is easily implemented. For example:

```
xxx10   CALL DASG (MD%, PARAM%(0), FLAG%)
xxx20   IF FLAG%  < > 0 THEN GOSUB yyyyy
or:
xxx20   IF FLAG%  < > 0 THEN PRINT "Error ";FLAG% : STOP)
.   .   .
.   .   .
yyyyy   REM: Error handling subroutine
.   .   .
.   .   .
zzzzz   RETURN
```

This is useful while debugging a new program; or with a suitable error handling subroutine, it can be left permanently in the program.

The following is a list of error codes:

Error 1 - Driver not initialized

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

`Error 2 - Mode number out of range`

Definition	This error may be issued from any MODE. It indicates that the mode number specified is not allowed.
Action	Check the mode number to be within the range 0 thru 25 inclusive.

`Error 3 - Board's base address out of range`

Definition	This error is generated by MODE 0 whenever the base address specified is not allowed.
Action	Specify a base address within the range 256 (100 Hex) and 1008 (3F0 Hex) inclusive.

`Error 4 - Interrupt Level used is out of range`

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

`Error 5 - DMA Channel not 1 or 3`

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

`Error 6 - Differential scan limits out of range`

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

`Error 7 - Single Ended scan limits out of range`

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

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

Definition	This error is generated by MODE 3 whenever an expected EOC signal from the DAS-16 is not received. This indicates a hardware failure.
Action	Check the board's base address.

`Error 10 - One or both counter values out of range`

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

`Error 11 - Number of conversions out of range`

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

`Error 12 - Counter 0 Configuration number out of range`

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

Error 13 - Digital Output data out of range

Definition	This error is generated by MODE 13 whenever the output data value is not within 0 thru 15 inclusive. There are four (4) Digital Output lines available on the DAS-16, 16F, 16G1 and 16G2.
Action	Check the Digital output value.

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

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

Error 15 - DAC Channel number not 0 or 1

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

Error 16 - Counter 0 'Read' operation number out of range

Definition	This error is generated by MODE 12 whenever an illegal Counter 'Read' operation is attempted. Two types of 'Read's are allowed: 0 = Normal Read and 1 = Latched Read.
Action	Check Read operation number.

Error 17 - Starting Sample number out of range

Definition	This error is generated by MODE 9 whenever the Starting sample number is out of range. The Starting Sample number must within 0 thru 32767 inclusive.
Action	Check the Stating Sample number.

Error 18 - Requested sample count out of range

Definition	This error is generated by MODEs 9 and 18 whenever the number of samples is out of range. The Sample count for these MODEs must within 1 thru 32767 inclusive.
Action	Check the Sample Count.

Error 19 - Trigger mode must be 0 or 1

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

Error 20 - Interrupt or DMA mode(s) already active

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

Error 21 - DMA Page Wrap

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

Error 22 - Board not present

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

`Error 23 - Analog Trigger channel out of range`

Definition This error is generated by MODE 19 whenever the Analog Input channel specified is out of range. This channel number must between 0 and 7 (Differential) or 0 and 15 (Single Ended) inclusive.

Action Check the channel number.

`Error 24 - Analog Trigger data out of range`

Definition This error is generated by MODE 19 whenever trigger value is out of range.

Action Specify a value between 0 and 4095 (Unipolar) or -2048 and 2047 (Bipolar) inclusive.

`Error 25 - Analog Trigger slope not 0 or 1`

Definition For MODE 19, the user is allowed to specify 0 to trigger on a Positive slope or 1 for a Negative slope.

Action Specify 0 or 1 for the Trigger Slope.

`Error 26 - Gain code out of range`

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

Action Specify a valid Gain Code.

`Error 27 - Old BASIC program compatibility problem`

Definition You will get this error when you use a new DAS-16 driver/library feature (Rev 4.10 or newer) from a BASIC program written for a previous Revision of the driver.

Action Replace all: **`CALL DAS16`** with **`CALL DASG`**, and increase your parameter array PARAM%(4) dimension to PARAM%(15).

`Error 29 - Possible memory index overflow`

Definition This is a rare error which is generated by MODEs 4,5, or 20 . Reduce your number of samples by 16.

Action Reduce your number of samples by 16, and retry.

`Error 30 - Memory allocation/deallocation error`

Definition	This error is generated by Mode 23 or Mode 24 whenever memory can not be allocated through DOS.
Action	If an error is generated from Mode 23, reduce the size of allocating memory. If the error is generated from Mode 24, make sure the right memory segment to be deallocated has been specified.

(This page intentionally left blank.)

APPENDIX D
DIRECT MEMORY ACCESS (DMA)

D.1 THE IBM PC DMA STRUCTURE

To obtain a higher speed of data transfer, the IBM PC uses an Intel 8237A-5 DMA controller. The DMA controller improves transfer speed by moving data directly between an I/O device and memory, bypassing the 8088 CPU.

The 8237A-5 provides four prioritized, direct-memory-access (DMA) channels. Each channel responds to two control signals: a DMA request (DRQ) and a DMA acknowledge (DACK).

A normal programmed-transfer example (without DMA) might be as follows:

```
LOOP:     IN AX,DX          ;read data from I/O port  ( 8)
          MOV [DI],AX       ;transfer to memory at    (10)
          CMP DI,LIMIT      DS:DI                     ( 4)
          JZ EXIT           ;end of transfer?         ( 4)
          INC DI            ;yes,exit                 ( 2)
          INC DI            ;no, continue             ( 2)
          JMP LOOP          ;increment memory         (15)
                            pointer
                            ;repeat
EXIT:     . . . . .         ;continue
                                                      -------
                                 Total clock periods = 43

          Loop execution time with 4.77MHz clock = 9µS
```

Actually, an A/D service routine would be even more involved as it would require checking the status or EOC of the A/D to see if the data was ready, etc. As a result, execution time of a typical routine tends to keep data conversion rates below 10KHz. Another shortcoming of a programmed transfer through the CPU is disruption from interrupts, the timer interrupt being a major culprit on the IBM PC.

DMA avoids these problems. The A/D on DAS-16/16F/16G is triggered by the programmable timer or an external start pulse; it performs a conversion and issues two sequential DRQ's at the end of conversion. On receipt of each DACK, the two bytes of data corresponding to each conversion are transferred to the memory location put on the address bus by the 8237 DMA controller. This transfer usually takes place in 5-9 microseconds and is undisturbed by interrupts, etc. Since the A/D data is latched and held until the end of the next conversion, at maximum sample rate is a minimum of 10 microseconds (A/D conversion time) for the transfer to take place. This is more than enough time, even with activity on higher-priority DMA levels.

When the 8237 receives a DRQ, it issues a hold request (HRQ) to the 8088 CPU asking it to release the address and data bus to the DMA controller chip (8237). As soon as the CPU is able to do this (within one machine cycle), it responds by returning a hold acknowledge (HLDA) to the DMA controller to tell it that it has the bus. The DMA controller then supplies the memory address on the address bus, issues a DACK to the I/O device to tell it to place its data on the data bus, and also provides simultaneous I/O read (IOR) and memory write pulses (MEMW) to effect the data transfer. Control is then returned to the CPU for at least one machine cycle before another DMA cycle is possible (this prevents the DMA controller from hogging the bus).

The 8237 handles a total of four DMA channels (three of which are available on the expansion bus), as follows:

DMA Level	Function	Signals	Priority
0	Memory refresh	Not on bus	Highest
1	Not used	DRQ1/DACK1	...
2	Floppy disk(s)	DRQ2/DACK2	...
3	Hard disk (XT) (see below)	DRQ3/DACK3	Lowest

Note that DMA Channel 0 is used by the system to perform a dummy read of each memory location every 15 microseconds and thereby refresh the dynamic memory. It is important not to interfere with the setup or operation of Channel 0 as this may lead to loss of memory and a computer crash.

DMA Level 2 is always used by the floppy disk(s) to read/write data; it cannot be shared with other devices.

DMA Level 1 is not committed to any internal device and is generally available on all versions of PCs; although, if you have them installed, some local area network interfaces may use this level.

On floppy-disk-only machines, DMA Level 3 is also free. If the PC has a hard disk, Level 3 may be used by the hard disk, but depending on the type of hard disk controller used it may be free. The hard-disk controller card contains the fixed disk BIOS. Some manufacturers make use of block moves (MOVS), others use hardware DMA to transfer data between the disk controller and DOS disk buffers in memory. This is transparent to the user as the BIOS calls are functionally identical. However, when you come to install another peripheral that uses DMA, it is useful to know whether your particular controller uses Level 3 or is free. Running *LEV3.EXE* performs a check. If you find it is unused, then the DAS-16/16F/16G can be safely set to this level.

The DAS-16/16F/16G may be operated on either of Levels 1 or 3. The DMA level is selected by the slide switch marked DMA above gold edge connector P1. Since the DAS-16/16F/16G is a relatively slow device in terms of DMA service, the higher priority Level 1 offers little real performance advantage over Level 3. Note also that until a DMA operation is enabled on the DAS-16/16F/16G, the DMA request (DREQ) line from the board is disabled (tri-state). You can therefore share it with other devices on the same level as long as they are not enabled at the same time.

D.2 PAGE REGISTER AND DMA CONTROLLER FUNCTIONS

The 8237 controller on the system board was designed in the days of 8-bit CPUs and 64K memories; it can handle only a 16-bit address (A0-A15). Since the 8088 uses a 20-bit address bus, the high-order bits (A16-A19) are provided by a set of supplemental registers for each DMA level. These are known as the DMA page registers, and although there are four DMA levels, there are only three page registers. DMA Level 0, used internally for memory refresh, does not really require a page register since it refreshes all pages regardless of the A16-A19 address bits. To economize, it shares the same page register as Level 1. The page register I/O locations are as follows:

DMA Level	Page Register	I/O Location
0 & 1	1	&H83
2	2	&H81
3	3	&H82

The 8237 DMA controller contains four registers that must also be initialized before a DMA transfer. These are as follows:

1. Mode register: Controls type of operation.
2. Base & current address: Sets start memory address.
3. Word count: Sets number of transfers.
4. Mask: Enables/disables DMA levels.

In addition, a fifth register, the Command register, is set by the BIOS on boot up and should not be altered.

Before a DMA operation can be started, all these registers must be initialized. This is taken care of by Mode 6 of the *DAS16.BIN* call routine; but, if you intend to write your own driver, it may help to buy the *DAS16.BIN* commented source listing (Call the Technical Support Department for more information. See Chapter 11.) for guidance. In addition to setting up the 8237 and page registers, the DAS-16/16F/16G control register must also be loaded to enable the DAS-16/16F/16G DMA hardware.

There are two types of DMA operation provided for in the *DAS16.BIN* driver. The first is a straight one shot data transfer. From 1 to 32767 conversions can be transferred to any 64K page of memory. At the end of the transfer, an interrupt is generated that shuts down the DMA hardware and signals completion of the operation. This is the non-recycle mode. The second type of operation is an auto-initialize or recycle mode where on reaching the final word count, the DMA controller automatically resets to the first memory location. In this case the DMA is continuous, and the word count specifies the length of the "circular" buffer that is created.

Information on the progress of a DMA operation can be obtained using Mode 8 of the CALL. Also, data can be transferred from a memory buffer area to a BASIC array using Mode 9.

D.3 USE OF DMA ON 64K IBM PCs

If you use DMA Level 1 on the early IBM PCs (the type with a maximum of 64K of memory installable on the system board), writing anything other than 0 to Page Register 1 will produce a memory PARITY CHECK error and a shut down of the machine. This is a shortcoming of the memory-error checking hardware that was remedied on later 256K machines. If this error occurs, you will have to turn off the computer and re-boot. There is a simple software fix for this problem. If you are not sure whether your machine has this problem, do this test: load BASICA and then type in

```
OUT &H83,1  (Enter)
```

If you get the usual BASIC "OK" prompt, then you have no problem. If you get "PARITY CHECK 2" and lose control of the computer (there is no way of avoiding using DMA Level 1 instead of Level 3), then before running Mode 6 of the *DAS16.BIN* driver add a line in your program as follows:

```
xxxxx OUT &HA0,0    'disables NMI
```

This disables the non-maskable interrupt (NMI) which is triggered by the memory error checking circuitry. Be aware that your computer is now operating without the memory parity checking. In view of the infrequency of memory errors, this is usually a minor loss. Once this is done the Page 1 register may be changed as many times as needed. Note that this fix may interfere with the 8087 co-processor operation if you have one installed, as it also uses the NMI.

(This page intentionally left blank.)

APPENDIX E
STORAGE OF INTEGER VARIABLES (BASICA)

When using BASIC, data is stored in integer variables (% type) in 2s complement form. Each integer variable uses 16 bits or 2 bytes of memory. 16 bits of data is equivalent to values from 0 to 65,535 decimal, but the 2s complement convention interprets the most significant bit as a sign bit. The actual range therefore becomes -32,768 to +32,767 (a span of 65,535). Numbers are represented as follows:

	HIGH BYTE								LOW BYTE							
	D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
+32,767	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
+10,000	0	0	1	0	0	1	1	1	0	0	0	1	0	0	0	0
+1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
-10,000	1	1	0	1	1	0	0	0	1	1	1	1	0	0	0	0
-32,767	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Sign bit: 1 if negative, 0 if positive.

Integer variables are the most compact form of storage for 12-bit data from the A/D converter and 16-bit data from the 8253 interval timer. Thus, to conserve memory and disk space and optimize execution speed, all data exchange via the CALL is through integer-type variables. This poses a programming problem when handling unsigned numbers in the range 32,768 to 65,535.

If you wish to input or output an unsigned integer greater than 32,767, you must work out its 2s-compliment signed equivalent. As an example, assume we want to load a 16-bit counter with 50,000 decimal. An easy way of turning this to binary is to enter BASIC and execute PRINT HEX$(50000). This returns C350 or binary:

50,000 (Hex C350) Binary 1100 0011 0101 0000

Since the most significant bit is 1, this would be stored as a negative integer and in fact the correct integer variable value would be 50,000 - 65,536 = -15,536. The programming steps for switching between integer and real variables for representation of unsigned numbers between 0 and 65,535 is therefore:

From real variable N (0 <= N <= 65,535) to integer variable N%:

```
xxx10 IF N<=32767 THEN N% = N ELSE N% = N - 65536
```

From integer variable N% to real variable N:

```
xxx20 IF N% >= 0 THEN N=N% ELSE N = N% + 65536
```

APPENDIX F
LINEARIZATION

POLYNOMIAL APPROXIMATION FOR USE WITH NON-LINEAR TRANSDUCERS

A common problem encountered in data acquisition is linearizing or compensating for non-linear transducers; for example, thermocouples, flowmeters, tachogenerators, etc. With a powerful computer behind the data acquisition system, it is simple to perform the linearizing function.

The starting point for any linearizing algorithm is a knowledge of the calibration curve (or input/output behavior) of the transducer. This may be derived experimentally or be available in manufacturer's data or standard tables; for example, NBS tables for thermocouples. There are several approaches to linearizing, two of the most common being (1) the piece-wise linearization using look up tables and (2) the use of a mathematical function to approximate the non-linearity. Amongst the mathematical methods, the polynomial expansion is one of the easiest to implement. The utility program *POLYNOM.BAS* allows you to generate a fifth-order polynomial approximation which is usually very adequate for linearizing most practical transducers. The transducer signal is digitized by the A/D and becomes the input (X) to the linearizing polynomial function, as follows:

$$Y = f(X) = C1 + C2*X + C3*X\char`^2 + C4*X\char`^3 + C5*X\char`^4 + C6*X\char`^5$$

Before you start the program, have in hand a table with your desired output/input data (or calibration data). To run the program from DOS, type **`BASICA POLYNOM`** and answer the first prompt which will ask you how many calibration points you intend to enter. The program will perform a least-squares fit regression analysis to insure the approximation is as close as possible over the whole range of input data. A question arises as to how many data points you have to enter to get a good approximation. This is something of a trial and error process, too few points, say less than six, may be insufficient for an accurate approximation whereas too many, say more than 40, is just tedious to enter and will not significantly improve the accuracy. When you have provided the number of data points, the program will then request you to enter the data as Y,X pairs (or output vs input) points. The two values for each data point should be entered with a comma as a delimiter between the values and you may edit with the backspace before you press return. The line on the bottom of the screen will tell you where you are up to. As soon as you have entered the final pair, you will go into a review and edit mode that allows you to change any values that you may have mis-entered. When all is OK, type **0** in response to the prompt. This is the point of no return and will either present more data for review or lead into the regression analysis. You will then be requested for the order of the polynomial that you wish to use; a 5 is usually the best choice although flexibility is provided in case you are interested in examining the effects of a lower order (Note: There is not really much advantage in using an order lower than 5 from the point of view of subsequent execution time for the polynomial evaluation).

The regression analysis terminates by displaying the coefficients, COEF(1 thru 6), which provide the best least squares approximation to your data. Just how good this approximation is, can be tested by entering any value you like in the TEST CONFORMANCE section and seeing what output results. When you tire of this, type **Q** (for quit) and you will then be asked if you want to try the regression analysis again with a different order (this just saves re-entering your data if you do). Type **Y** (yes) or **N** (no) as required. If you intend to save the values of the coefficients either copy them down now or if you have a printer, press PrtScrn before they are scrolled off into oblivion. This is strictly a utility program and does not save anything for you. As with any BASIC program, you can abort execution at any point by pressing Ctrl Break.

When you want to include a linearization routine in your own programs and evaluate the polynomial, you can do it in a neat little subroutine (this one is in BASIC) as follows:

```
xxx00    COEF(1) = Value
xxx10    COEF(2) = Value
xxx20    COEF(3) = Value
xxx30    COEF(4) = Value
xxx40    COEF(5) = Value
xxx50    COEF(6) = Value
xxx60    Y = COEF(1)
xxx70    FOR CNT% = 5 TO 1 STEP -1
xxx80    Y = Y + COEF(CNT%+1) * X ^ CNT%
xxx90    NEXT CNT%
xx100    RETURN
```

As a further aid to understanding this section, list *POLYNOM.BAS*.