PCF-8

The

PCF-8
Guide to Using
PASCAL, C, & FORTRAN
Callable Driver Software

for the

DAS-8, DAS-8PGA, & DAS-8/AO

Revision B, - July 1990

Part Number: 24871

KEITHLEY METRABYTE CORPORATION

440 MYLES STANDISH BLVD., Taunton, MA 02780
TEL. 508/880-3000, FAX 508/880-0179

Warranty Information

All products manufactured by Keithley MetraByte are warranted against defective materials and worksmanship for a period of one year from the date of delivery to the original purchaser. Any product that is found to be defective within the warranty period will, at the option of Keithley MetraByte, be repaired or replaced. This warranty does not apply to products damaged by improper use.

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.

Disclaimer

Information furnished by Keithley MetraByte is believed to be accurate and reliable. However, the Keithley MetraByte Corporation assumes no responsibility for the use of such information nor for any infringements of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent rights of Keithley MetraByte Corporation.

Notes

Keithley MetraByte/Asyst/DAC is also referred to here-in as *Keithley MetraByte.*

Basic™ is a trademark of Dartmouth College.

IBM® is a registered trademark of International Business Machines Corporation.

PC, XT, AT, PS/2, and **Micro Channel Architecture**® (MCA) are trademarks of International Business Machines Corporation.

Microsoft® is a registered trademark of Microsoft Corporation.

Turbo C® is a registered trademark of Borland International.

Contents

Chapter 1
INTRODUCTION

1.1 OVERVIEW

MetraByte's PCF-8 package is an aid for Pascal, C, and Fortran programmers who must develop data acquisition and control routines for MetraByte's DAS-8 series and EXP-16 (expander/multiplexer) boards. The package consists of an Assembly language driver (DAS8.OBJ), a simple graphics package (DRAW.LIB), and several example programs for each language.

Note that the DAS-8 series of boards includes the DAS-8, the DAS-8PGA, and the DAS-8/AO models. For the sake of brevity, this manual refers to these boards collectively as the DAS-8/PGA/AO.

1.2 PCF-8 IMPLEMENTATION

The three software drivers supplied in the PCF-8 package are similar to the BASIC drivers provided with the DAS-8/PGA/AO and should therefore be easily implemented by anyone familiar with these boards. If you are unfamiliar with MetraByte's DAS-8/PGA/AO, we suggest you read the manual for these boards -- paying particular attention to the programming section (Chapters 3 and 4).

The differences between the PCF-8 callable drivers and those of the DAS-8 are the way data and computer control are passed to and from the user program and the assembly driver. The various functions are selected using *MODEs* . Currently, there are 25 MODEs (0 thru 24), each performing a specific function. Each of these MODEs is described in Chapter 4 of the DAS-8/PGA/AO manual, which is extensively referenced herein. However, this manual includes a quick reference guide and a description of each MODE along with applicable execution parameters. This document specifically illustrates syntax and calling conventions required to execute any of the 25 MODEs from Pascal, C, and Fortran.

1.3 CURRENT DAS-8 DRIVER

The current DAS-8/PGA/AO driver is Version 4.4. This version is larger than its predecessors and therefore uses more memory. It is important that machines have enough free memory when loading the driver via BASIC.

Chapter 2

MODE DESCRIPTIONS

2.1 OVERVIEW

This chapter details the functions and implemention of each MODE. Arguments are integer variables. Using QuickBASIC, a typical MODE passes and returns values as follows:

```
integer MD                          'Declare all variables as 2-byte
integer D%(6)                       'signed/unsigned integers.
integer FLAG                        'integer corresponding to error number.
MD% = mode number                   'Range is 0 - 24
D%(0) = argument 1                  'Set up arguments
.
D%(5) = argument 6
FLAG% = DAS8(MD%,D%)                'Flag returned from function
```

2.2 MODE TYPES & FUNCTIONS

MODE	FUNCTIONAL DESCRIPTION
0	Initialize DAS-8/PGA/AO (Set Base Address).
1	Set Channel Scan limits: DAS-8/PGA/AO Mux low & high.
2	Set next multiplexer channel.
3	Read current multiplexer channel.
4	Single A/D conversion, return data and increment channel.
5	Multiple A/D conversions. Scan rate set by Counter 2 or external strobe.
6	Enable DAS-8/PGA/AO interrupt handler.
7	Disable DAS-8/PGA/AO interrupt handler.
8	Multiple A/D conversions on interrupts (Background operation).
9	Data transfer to BASIC array.
10	Configure DAS-8/PGA/AO timer/counter.
11	Load timer/counter.
12	Read timer/counter.
13	Read digital inputs IP1-3.
14	Write digital outputs OP1-4.
15	Measure frequency (with timer/counter).
16	Measure period (with timer/counter).
17	Enable/Disable Channel Tag (lower nybble).
18	Multiple A/D conversions using software trigger.
19	Set PGA gain (DAS-8PGA only).
20	Return interrupt status.
21	Perform "N" A/D conversions in X channel bursts with triggers.
22	Multiple A/D conversions w/interrupts using EXP-16.
23	Output Data To DACs (DAS-8/AO Only).
24	Update DACs from Array On Interrupt (DAS-8/AO Only).

2.3 MODE DESCRIPTIONS

2.3.1 MODE 0: Initialize

Prior to using other MODEs in the CALL routine, you must set the BASE ADDRESS and check for the PGA or /AO options. Failure to provide this information causes an error flag (FLAG = 1, Base Address unknown) when other MODEs are implemented. Intialization is required only once and is generally accomplished in the initialization section of your program.

On entry the following parameters should be initialized:

```
MD% = 0                          (MODE #)
BASADR% = &H300                  (I/O address)
FLAG% = X                        (value does not matter)
```

Then

```
CALL DAS8 (MD%, BASADR%, FLAG%)
```

Alternatively, you may use DAS8DI.SYS (if previously installed, see Section 2.1) as follows:

```
MD% = 0                          (MODE 0)
BRDID% = 0, 1, 2, 3, 4, or 5     (ID# for selected µCDAS-8PGA)
FLAG% = X                        (Value does not matter)
```

Then

```
CALL DAS8(MD%, BRDID%, FLAG%)
```

On return, the variables contain data as follows:

```
MD% = 0                          (unchanged)
BASADR% = &H300                  (unchanged)
```

The following Error Codes apply to MODE 0:

If FLAG% = 0 (no error, OK)
= 2 (MODE number out of range, <0 or >24)
= 3 (Base Address out of range <255 or >1016)

Note that Error 3 occurs if you have specified an I/O address that is less than 255 (Hex FF) or greater than 1016 (Hex 3F8). I/O addresses below Hex FF are used internally by devices on the IBM PC system board and will always cause an address conflict with the DAS-8/PGA/AO. Addresses above Hex 3FF are not decoded on the IBM PC.

2.3.2 MODE 1: Set Channel Scan Limits (MUX Low & High)

MODE 1 allows you to assign the channel scan limits prior to performing A/D conversions. The lower and upper limits are passed as arguments in a 2-element array variable LT%(0) and LT%(1). Default Channels are 0 and 7, respectively.

To illustrate the action of MODE 1, assume we set LT%(0) = 3 and LT%(1) = 6. The first A/D conversion (via MODEs 4 or 7) would be performed on Channel 3. Data is returned and the channel incremented to 4. The next conversion is performed on Channel 4, data returned and the channel

incremented to 5, etc.. This continues until a conversion is performed on Channel 6. The multiplexer address is then reset to 3 and the cycle may be repeated. Subsequent channel scanning occurs from Channels 3 through 6. Continuous conversions on a single channel require the low and high channel limits be set equal; for example LT%(0) = 1, LT%(1) = 1 (for Channel 1).

Note that the starting channel for A/D conversion defaults to LT%(0), the lower limit. If you wished to start A/D conversion at some other channel, MODE 2 could be used to assign the starting channel # after setting the channel scan limits.

On entry the following parameters should be initialized:

```
MD% = 1                          (MODE #)
LT%(0) = 0 thru 7                (lower scan limit)
LT%(1) = 0 thru 7                (upper scan limit)
FLAG% = X                        (value does not matter)
```

Then

```
CALL DAS8 (MD%, LT%(0), FLAG%)
```

Specifying the first array element, LT%(0), will pass all other required array parameters.

On return the variables contain:

```
MD% = 1                          (unchanged)
LT%(0) = 0 thru 7                (unchanged)
LT%(1) = 0 thru 7                (unchanged)
```

The following Error Codes apply to MODE 1:

If FLAG% = 0 (no error, OK)
= 1 (Base Address unknown)
= 2 (MODE number out of range, <0 or >24)
= 4 (scan limits out of range)

Error Code 4 indicates that either or both of the scan limits are out of range. That is, <0 or >7 or their order is reversed; for example, lower limit > upper limit.

2.3.3 MODE 2: Set Next Multiplexer Channel

MODE 2 sets the multiplexer address for the next conversion. Default is zero.

On entry the following parameters should be initialized:

```
MD% = 2                          (MODE)
CH% = 0 thru 7                   (channel number)
FLAG% = X                        (value does not matter)
```

Then

```
CALL DAS8 (MD%, CH%, FLAG%)
```

On return the variables contain data as follows:

```
MD% = 2                          (unchanged)
CH% = 0 thru 7                   (unchanged)
```

The following Error Codes apply to MODE 2:

If FLAG% = 0 (no error, OK)
= 1 (Base Address unknown)
= 2 (MODE number out of range, <0 or >24)
= 5 (channel number out of range)

Note that MODE 1 should be used to explicitly declare the channel scan limits prior to entering MODE 2. Otherwise, they will default to 0 and 7.

2.3.4 MODE 3: Read Current Multiplexer Channel

MODE 3 allows you to determine the current multiplexer (channel) address.

On entry the following parameters should be initialized:

```
MD%     = 3                      (MODE)
CH%     = X                      (value does not matter)
FLAG%   = X                      (value does not matter)
```

Then

```
CALL DAS8 (MD%, CH%, FLAG%)
```

On return the variables contain data as follows:

```
MD%  = 3                         (unchanged)
CH%  = 0 thru 7                  (current channel number)
```

The following Error Codes apply to MODE 3:

If FLAG% = 0 (no error, OK)
= 1 (Base Address unknown)
= 2 (MODE number out of range, <0 or >24)

Note that attempting to read the MUX channel number while an A/D conversion is in progress may have unexpected and generally undesirable effects. Therefore, when using the auto-increment MODEs 4, 5 or 8, it is good practice to check the DAS-8/PGA/AO Status Register EOC bit for inactivity (channel incrementing occurs at the start of the A/D conversion while the sample/hold is holding and the EOC bit is active high).

2.3.5 MODE 4: Single A/D Conversion & Increment Channel MUX

MODE 4 initiates the following sequence of events:

1. Starts 12-bit A/D conversion.
2. Increments channel mux prior to A/D and sample/hold is holding. Checks scan limit; if limit exceeded, reset to lower scan limit.
3. Poll status to determine if A/D finished.

4. Return data to variable.

The A/D performs conversions on channels in accordance with the conditions set in MODEs 1 & 2. If MODEs 1 & 2 have not been entered prior to MODE 4, scan limits default to 0 & 7.

On entry the following parameters should be initialized:

```
MD%     = 4                     (MODE)
D%      = X                     (value does not matter)
FLAG%   = X                     (value does not matter)
```

Then

```
CALL DAS8 (MD%, D%, FLAG%)
```

On return the variables contain data as follows:

```
MD%     = 4                     (unchanged)
D%      = data                  (converted data)
```

Note that if MODE 17 has been enabled prior to entering MODE 4, returned data will be "tagged" with the channel address (see MODE 17).

The following Error Codes apply to MODE 4:

If FLAG% = 0 (no error, OK)
= 1 (Base Address unknown)
= 2 (MODE number out of range, <0 or >24)
= 6 (A/D timeout)

Error Code 6 will occur only if the end of conversion signal (EOC) from the A/D remains high for more than 100 microseconds or stays low after a conversion has been initiated. Either of these conditions is indicative of a hardware fault and it is recommended that users replace the A/D converter I.C. (AD574A) as a first step in attempting correction. A normal 12-bit A/D conversion should not take more than 35 microseconds.

2.3.6 MODE 5: Multiple Conversions On Trigger; Transfer Data To Array

MODE 5 is an expanded version of MODE 4 for multiple data conversions according to the channel scan limits. Mulitple conversions are initiated by a high input on IP1. Once the scan has started, the rate at which conversions are performed is set by Counter 2. The number of conversions (N) specified at entrance to the CALL is performed and data is transferred to a specified integer array ARRAY%(N) previously dimensioned for not less than N elements. Error checking is not performed to determine whether sufficient array space exists.

The interrupt handshake flip-flop on the DAS-8/PGA/AO is used to acknowledge trigger inputs in this MODE although the conversions are not interrupt driven. MODE 5 performs a maximum of about 4000 conversions/sec; although some conversions may be delayed by the timer interrupt of the IBM PC, since the system clock has a very high interrupt priority level. If you wish to trigger at a programmable time interval set by Counter 2 then:

Jumper externally	OUT 2 (Pin 6) to INT.IN (Pin 24)
Start scan	High input to IP1 (Pin 25)

See MODEs 10 & 11 and Chapter 4 on setting Counter 2 to output a periodic pulse (configuration # 2 or 3). For slow scan rates (>27 milliseconds/conversion), counter 1 and/or 0 can be cascaded to counter 2 output and their outputs connected to INT.IN. Alternatively, you may connect an external trigger source to INT.IN, with each positive edge triggering a conversion. After the CALL routine is entered, IP1 is polled. If IP1 is low, polling will continue until it goes high. If IP1 is an open circuit (floats high) or as soon as it is driven high, the scan sequence commences on the first positive transition to INT.IN input. Each time a positive edge is received, an A/D conversion is performed as in MODE 4.

The sequence of events is as follows:

1. Test IP1 until it goes high.
2. Start A/D conversion on positive edge to INT.IN
3. Clear interrupt flip-flop (IRQ) for next trigger.
4. Increment Mux after A/D has started and sample/hold is in hold. Check whether scan limit exceeded; if so, reset to lower scan limit.
5. Poll status to determine if A/D finished.
6. Return data to next element of array.
7. Check for N conversions. If so, exit otherwise loop & do again.

The A/D will perform conversions on channels in accordance with the conditions set in MODEs 1 or 2. If MODEs 1 or 2 have not been entered prior to MODE 5, channel scan limits default to 0 & 7.

On entry the following parameters should be initialized:

```
MD%       = 5                     (MODE)
TRAN%(0)  = VARPTR(ARRAY%(0))     (pointer to data array)
TRAN%(1)  = N                     (number of conversions)
FLAG%     = X                     (value does not matter)

IF TRAN%(0) = -1 THEN
   TRAN%(2) = OFFSET OF ARRAY
   TRAN%(3) = SEGMENT OF ARRAY
```

Then

```
CALL DAS8 (MD%, TRAN%(0), FLAG%)
```

On return the variables contain data as follows:

```
MD%          = 5                        (unchanged)
TRAN%(0)     = VARPTR(ARRAY%(0))        (unchanged)
TRAN%(1)     = N                        (unchanged)
ARRAY%(0)    = Data from 1st. conversion
ARRAY%(1)    =   "    "  2nd.      "
ARRAY%(2)    =   "    "  3rd.      "
.
ARRAY%(N-1)  = Data from Nth. conversion
```

Note that if MODE 17 has been enabled prior to entering MODE 5, returned data will be "tagged" with the channel number (see MODE 17). Also, exit from the routine will not occur until IP1 has been taken high and TRAN%(0) is satisfied (desired number of conversions have been performed). If these conditions are not met, your computer may appear to be hung waiting for the above conditions to be

satisfied. Control-Break will not exit you from the routine once started. Control-Alt-Delete will, but your program will be lost. Be careful!

The following Error Codes apply to MODE 5:

If FLAG%	= 0	(no error, OK)
	= 1	(Base Address unknown)
	= 2	(MODE number out of range, <0 or >24)
	= 6	(A/D timeout on any conversion)

Error Code 6 occurs only if the end of conversion signal (EOC) from the A/D remains high for more than 100 microseconds or stays low after a conversion has been initiated. Either of these conditions is indicative of a hardware fault and it is recommended that users replace the A/D converter I.C. (AD574A) as a first step in attempting correction. A normal 12-bit A/D conversion should not take more than 35 microseconds.

2.3.7 MODE 6: Enable Interrupt Handler

MODE 6 sets up an interrupt service routine for subsequent background data transfer using MODE 8. Before selecting MODE 6/8, you should decide upon the Interrupt Level you wish to assign to the DAS-8/PGA/AO and position the Interrupt Level Jumper on the DAS-8/PGA/AO board (J2) to that level. The IBM PC is provided with 8 levels of interrupts operating through the 8259 Interrupt Controller. Level 0 has the highest priority and Level 7 the lowest. Levels 0 and 1 are not available on the expansion bus connectors since Level 0 is used internally for the TIME & DATE functions, and Level 1 is used to service the keyboard.

Levels 2 - 7 have been preassigned by IBM for use as follows:

Level 2	Reserved (but not used) by Color Graphics adapter
Level 3	Serial I/O - used by COM2: if installed.
Level 4	Serial I/O - used by COM1: if installed.
Level 5	Printer - may be used by LPT2: if installed.
Level 6	Always in use by disk drives
Level 7	Printer - may be used by LPT1: if installed.

Since the disk drives are slow devices, using interrupt levels lower than 5 might delay A/D conversion service for long periods. In most systems, at least one or more of Levels 2 - 5 is available making these the best choices. When you have decided, move the Interrupt Level Jumper on the DAS-8/PGA/AO board to the desired Interrupt Level.

MODE 6 initiates the following sequence of events:

1. Loads interrupt vectors into memory for the level selected and stores any old vectors for subsequent restitution by MODE 7.
2. Enables interrupt handler routine.
3. Initializes 8259 interrupt controller and enables 8259 interrupt mask register for level selected. Interrupts are generated by a low to high transition on the INT.IN input (Pin 24).
4. Specifies the type of interrupt buffer according to the "re-cycle" parameter. If D%(1)=0, conversions cease after the buffer is filled. If D%(1)=1, the buffer is circular and begins overwriting data upon buffer full. It makes sense to make the buffer a multiple of the number of scanned channels. Buffer location and size are specified using MODE 8.

Interrupts are not enabled until the INTE bit in the DAS-8/PGA/AO control register is taken high. MODE 8 performs this final step.

On entry the following parameters should be initialized as follows:

```
MD%     = 6  (MODE)
D%(0)   = 2 thru 7              (interrupt level)
D%(1)   = 0 or 1                (0 = NO Recycle, 1 = Recycle))
FLAG%   = X                     (value does not matter)
```

Then

```
CALL DAS8 (MD%, D%, FLAG%)
```

On return the variables contain data as follows:

```
MD%     = 6  (unchanged)
D%(0)   = 2 thru 7              (unchanged)
D%(1)   = Recycle               (unchanged)
```

The following Error Codes apply to MODE 6:

If FLAG% = 0 (no error, OK)
= 1 (Base Address unknown)
= 2 (MODE number out of range, <0 or >24)
= 7 (interrupt level out of range <2 or >7)

2.3.8 MODE 7: Disable Interrupt Handler

MODE 7 disables interrupt processing initiated by MODEs 6 & 8 for the level previously selected. It restores previous interrupt vectors and the 8259 mask register state. Note that the CALL routine should not be reloaded before entering MODE 7 as this may destroy temporary storage of old vectors etc.

On entry the following parameters should be initialized:

```
MD%     = 7                     (MODE #)
D%      = X                     (declare variable, value does not matter)
FLAG%   = X                     (entry value does not matter)
```

On return the variables contain data as follows:

```
MD%     = 7                     (unchanged)
D%      = X                     (unchanged)
```

The following Error Codes apply to MODE 7:

If FLAG% = 0 (no error, OK)
= 1 (Base Address unknown)
= 2 (MODE number out of range, <0 or >24)

2.3.9 MODE 8: Multiple Conversions On Interrupt (Background)

MODE 8 initiates background data acquisition via interrupts. MODE 8 requires two initialization parameters; (1) the length of the data buffer (in words, 1 - 32767) and (2) the destination memory segment for the data buffer. Each data word from the A/D requires 2 bytes, so that 32767 conversions

will occupy an entire memory segment (64K). Each conversion is triggered by an interrupt, and when ready, data is transferred to the next available word in the buffer and the multiplexer incremented within the scan limits in preparation for the next interrupt. Periodic interrupts can be derived from the DAS-8/PGA/AO timer/counter (set via 10) or alternatively by an external trigger input. An external connection is required from the selected counter output or the user's trigger source to the INT. IN (pin 24). Interrupts are triggered by a low to high transition at pin 24.

On entry, MODE 8 enables the INTE bit of the DAS-8/PGA/AO and the CALL is exited. As soon as an interrupt is received, a conversion is performed and data transferred. This continues on each interrupt until the data buffer is filled. If the recycle option of MODE 6 has been disabled, the INTE bit of the DAS-8/PGA/AO control register is set to zero and futher interrupt inputs are ignored. If the recycle option of MODE 6 is enabled, further conversions overwrite the data in the buffer, word by word, starting from the beginning. In this case, the data buffer can be as small as 1 word or as large as 32767 words, but making the buffer a multiple of the number of active channels in the scan makes every word correspond with a particular channel which is convenient for data retrieval. If the interrupt is being derived from the DAS-8/PGA/AO 8254 counter/timer, the counter gate can be used to hold off interrupts until an external system is ready.

Each 12 bit conversion requires 2 bytes of memory for storage. Consecutive conversions are placed in consecutive memory locations starting with the segment address specified. The segment address can be in the range of 0 to -24577 (Hex 0 to 9FFF) * which corresponds to the maximum addressable RAM available in the IBM PC. If MODE 17 is selected, the channel address is contained in the lower nybble of the 1st byte. This "Tagged" data requires 2 bytes for storage (see MODE 17). In either case, the 8 most significant bits of data are in the 2nd byte and the 4 least significant bits in the upper nybble of the 1st byte.

Data is retrieved from memory to a BASIC integer array using MODE 9. You should choose a data storage segment that will not interfere with your program space (that is, BASIC workspace or other programs). If your machine is memory-limited, contraction of your BASIC workspace will be required to open up a data storage area. Also avoid loading data over the CALL routine which contains the interrupt handler. Once the interrupt is running, you may load other programs and other languages and come back later to process collected data. If you do this, avoid resetting the computer as the memory check that BIOS performs will destroy your data. The Ctrl-Alt-Delete sequence will leave your data intact in memory but will discontinue interrupts.

NOTE: The DAS-8 driver adds the number of conversions to the segment address. If the resultant segment address = > A000H, FLAG% = 8 is returned.

Apart from being a background operation, MODE 8 has a number of operational advantages from a performance point of view. Data can be collected much faster than in MODEs 4 & 5 as the execution time of the BASIC interpreter involved in going in and out of the CALL statement is eliminated. Secondly, a greater amount of data can be stored than is possible using a BASIC array since 64K corresponds to the maximum workspace of BASIC including program, stack and temporary storage. Therefore, considerably less than 64K of data storage is available in arrays within the BASIC workspace. Provided you have sufficient memory, a full 64K block can be set aside for data storage. Whenever maximum speed and/or maximum storage is required, MODE 8 should be used.

Up to 4000 conversions/sec are possible using MODE 8. As the interrupt rate rises, less and less time is available for foreground operations. Above about 3000 conversions/sec a noticeable degradation in computer processing speed will occur and certain I/O operations will become significantly slower.

On entry, the following variables should be initialized:

```
MD%     = 8                     (MODE)
D%(0)   = 1 thru 32767          (data buffer length in words)
D%(1)   = 0 thru -24577         (segment address of data storage)
FLAG%   = X                     (entry value does not matter)
```

Then

```
CALL DAS8 (MD%, D%(0), FLAG%)
```

Note that D% must be a 2-element integer array with only the first element referenced in the CALL.

On return all variables are unchanged except FLAG%.

```
MD%     = 8                     (unchanged)
D%(0)   = 1 thru -24577         (unchanged)
D%(1)   = 0 thru 32767          (unchanged)
```

The following Error Codes apply to MODE 8:

If FLAG% = 0 (no error, OK)
= 1 (Base Address unknown)
= 2 (MODE number out of range, <0 or >24)
= 8 (transfer parameters out of range)

Error code 8 will occur if the number of conversions requested is <1 or >32767 and/or the segment address for storage is negative or >32767 (A000H). In these cases, the interrupt will not be enabled.

2.3.10 MODE 9: Data Transfer To BASIC Array

MODE 9 transfers data from a memory storage area into a BASIC integer data array. MODE 9 transfers N conversions from offset M of the memory segment selected in MODE 8. MODE 9 can be entered whether or not conversions in MODE 8 have ceased. MODE 9 simply copies data from memory and will not alter the data. The number of conversions transferred (2 byte data words) can be between 1 and 32767 and the offset can be anywhere from 0 to 32767. This allows transfer of any block length of data from any location in data storage to an appropriately dimensioned integer array. This indirect method of retrieving data may seem complex, but it avoids several problems that would arise if data was transferred directly to an array by MODE 8. After entry to the CALL, the pointers to the array are placed on the stack and used by the routine to locate the array. Once you have initiated MODE 8 interrupt, it is possible to declare further simple variables, alter your program or chain to other programs. Any of these actions would dynamically relocate the array variables so that CALL would lose the location of the array with disasterous results. The solution is to re-enter the CALL so that it is always working with the current location of the array. This is why an indirect method is used, giving you the flexibility to start the interrupt (MODE 8) with one program, then load another program to read the data (MODE 9) and avoid disturbing the array location.

Once background data acquisition has been set up (via the recycle option of MODE 6 and a data buffer area specified by MODE 8), a foreground program may be used to process data retrieved by MODE 9. This is excellent for graphic and "digital oscilloscope" applications. The role of the variables is somewhat different in MODE 9, and they should be initialized as follows:

```
MD%        = 9                    (MODE)
TRAN%(0)   = VARPTR(ARRAY%(N))    (pointer to data array)
TRAN%(1)   = 1 thru 32767         (number of data transfers)
TRAN%(2)   = 0 thru 32767         (offset from segment start)
FLAG%      = X                    (value does not matter)
```

TRAN%(0) points to the desired element in your data array (ARRAY%(N)) via VARPTR (N=0 for element 1). This array should be dimensioned for the maximum number of transfers specified in TRAN%(1).

Then

```
CALL DAS8 (MD%, TRAN%(0), FLAG%)
```

On return the variables are as follows:

```
MD%           = 9                 (unchanged)
TRAN%(0 - 2)                      (unchanged)
ARRAY%(N)     = 1st data word
ARRAY%(N+1)   = 2nd data word
ARRAY%(N+2)   = 3rd data word
```

The following Error Codes apply to MODE 9:

If FLAG% = 0 (no error, OK)
= 1 (Base Address unknown)
= 2 (MODE number out of range, <0 or >24)
= 9 (transfer parameters out of range)

Error 9 is returned if the number of data transfers requested is zero, negative or >32767 and/or the memory offset is negative or >32767. Note that array size is not checked so that more data may be transfered than the array can handle. This is often undesirable and may cause your computer to hang up.

It is reccommended that TRAN%(0) = VARPTR(ARRAY%(0)) be assigned immediately before the CALL statement since declaration of a new simple variable after this assignment will dynamically relocate ARRAY%(0) and upset operation of this MODE. For instance, the following example correctly declares the variable "I" prior to the assigning TRAN%(0)=VARPTR(ARRAY(0)).

```
xxx10  I = 3
xxx20  TRAN%(0) = VARPTR(ARRAY%(0))
xxx30  CALL DAS8 (MD%, TRAN%(0), FLAG%)
```

As an alternative to MODE 9, you may be tempted to retrieve data using BASIC's PEEK function. If interrupt transfers are active, be warned that PEEK retrieves data one byte at a time and interrupts occuring between readings of low and high data bytes will cause the time sequential PEEK to return erroneous data. Using MODE 9 avoids this problem.

2.3.11 MODE 10: Configure Timer/Counter

MODE 10 is used to configure the DAS-8/PGA/AO timer/counter. For a complete discussion of the possible configurations see Chapter 4 on timer/counter operations. Each of the three counters (0, 1 & 2) may be set to one of six configurations:

0	Pulse high on terminal count
1	Programmable one-shot
2	Rate generator
3	Square wave generator
4	Software triggered strobe
5	Hardware triggered strobe

On entry the following parameters should be initialized:

```
MD%     = 10 (MODE)
D%(0)   = 0 thru 2              (counter number)
D%(1)   = 0 thru 5              (configuration number)
FLAG%   = X                     (value does not matter)
```

Then

```
CALL DAS8 (MD%, D%(0), FLAG%)
```

On return the variables are unchanged:

```
MD%     = 10                    (unchanged)
D%(0)   = 0 thru 2              (counter number)
D%(1)   = 0 thru 5              (configuration number)
```

The following Error Codes apply to MODE 10:

If FLAG%	= 0 (no error, OK)
	= 1 (Base Address unknown)
	= 2 (MODE number out of range, <0 or >24)
	= 10 (counter number out of range <0 or >2)
	= 11 (configuration number out of range <0 or >5)

2.3.12 MODE 11: Load Timer/Counter

MODE 11 loads the selected timer counter. Since each counter is a 16-bit device, counts as high as 65,535 are possible. Integer variables are signed 16-bit words; that is, they 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.

On entry the following parameters should be initialized:

```
MD%     = 11                    (MODE)
D%(0)   = 0 thru 2              (counter number)
D%(1)   = -32768 to +32767      (counter load data)
FLAG%   = X                     (value does not matter)
```

Then

```
CALL DAS8 (MD%, D%(0), FLAG%)
```

On return the variables are unchanged:

```
MD%     = 11                    (unchanged)
D%(0)   = 0 thru 2              (unchanged)
D%(1)   = -32768 to +32767      (unchanged)
```

The following Error Codes apply to MODE 11:

If FLAG% = 0 (no error, OK)
= 1 (Base Address unknown)
= 2 (MODE number out of range, <0 or >24)
= 10 (counter number out of range <0 or >2)

2.3.13 MODE 12: Read Timer/Counter

MODE 12 reads the selected timer/counter. Counter data is 16-bits wide and integer variables are returned as signed 16-bit words with 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.

On entry the following parameters should be initialized:

```
MD%     = 12                    (MODE)
D%(0)   = 0 thru 2              (counter number)
D%(1)   = X                     (value does not matter)
FLAG%   = X                     (value does not matter)
```

Then

```
CALL DAS8 (MD%, D%(0), FLAG%)
```

Variables are returned as follows:

```
MD%     = 12                    (unchanged)
D%(0)   = 0 thru 2              (unchanged)
D%(1)   = -32768 to +32767      (counter data)
```

The following Error Codes apply to MODE 12:

If FLAG% = 0 (no error, OK)
= 1 (Base Address unknown)
= 2 (MODE number out of range, <0 or >24)
= 10 (counter number out of range <0 or >2)

2.3.14 MODE 13: Read Digital Inputs IP1-3

MODE 13 allows you to read the state of digital inputs IP1-3. Data returned can range between 0 and 7 corresponding to all combinations of the 3 input bits.

On entry the following parameters should be initialized:

```
MD%     = 13                    (MODE)
IP%     = X                     (value does not matter)
FLAG%   = X                     (value does not matter)
```

Then

```
CALL DAS8 (MD%, IP%, FLAG%)
```

On return the variables contain data as follows:

```
MD%     = 13                    (unchanged)
IP%     = 0 thru 7              (input data)
```

The following Error Codes apply to MODE 13:

If FLAG% = 0 (no error, OK)
= 1 (Base Address unknown)
= 2 (MODE number out of range, <0 or >24)

2.3.15 MODE 14: Write Digital Outputs OP1-4

MODE 14 allows you to output data on the 4 digital output lines OP1-4. Since this data is passed through the DAS-8/PGA/AO control register, MODE 14 performs an OR operation with the current multiplexer address and INTE bits, leaving these unchanged.

On entry the following parameters should be initialized:

```
MD%     = 14                    (MODE)
OP% 0 thru 15                   (output data)
FLAG%   = X                     (value does not matter)
```

Then

```
CALL DAS8 (MD%, OP%, FLAG%)
```

On return the variables contain data as follows:

```
MD%     = 14                    (unchanged)
OP%     = 0 thru 15             (unchanged)
```

The following Error Codes apply to MODE 14:

If FLAG% = 0 (no error, OK)
= 1 (Base Address unknown)
= 2 (MODE number out of range, <0 or >24)
= 12 (output data out of range, <0 or >15)

2.3.16 MODE 15: Measure Frequency

MODE 15 allows you to measure frequency using the 8254 counter/timer and one digital input. In this MODE, Counter 2 is configured to output 1mspulses. This square-wave output signal must be externally connected to Counter 1 input. Setting FRQ%(0) = 100 provides a 100ms gating pulse at the output of Counter 1 (FRQ%(0) can be set to any number < 65,535mS). This output signal (counter 1) must be externally connected to the gate of Counter 0 and IP2. The unknown frequency (a TTL compatible signal) is connected to the clock input of Counter 0. During the low period of the gate signal, Counter 0 is set to 65,535. When the gate signal goes high, Counter 0 counts down, and on the gate signal returning low, the change in the count is returned in FRQ%(1). This count is proportional to the frequency of the unknown signal and if gate intervals of 0.1, 1 or 10 seconds are used, the unknown frequency is calculated by simply multiplying FRQ%(1) by 10, 1 or 0.1 respectively.

The following external connections should be made:

Jumper - Out 2 (Pin 6) to Clk.1 (Pin 4)
Jumper - Out 1 (Pin 5) to Gate 0 (Pin 21) and IP2 (Pin 26)
Unknown frequency (TTL) - wired to Clk.0 (Pin 2) and Common (Pin 11 or 28)

The DAS-8 derives its timing from the system clock (of the computer) and is actually 1/2 of the system clock frequency. The DAS-8PGA has a dedicated 1 MHz clock. The DAS-8 contains a small rounding error that occurs in Counter 2 (actual output pulses are 0.99985 mS period). An absolute accuracy of ±0.1% ±1 count is readily achievable.

MODE 15 sets the correct counter configurations and performs the entire frequency measurement sequence once the external connections are made. Note that you may still use the output of Counter 1 to trigger interrupts and, in fact, this MODE sets the output of Counter 1 in convenient 1ms increments. Note that the execution time of MODE 15 may be as much as 4*FRQ(0)mS. At higher frequencies with 10 or 100ms gating intervals, the measurement time is brief, but at low frequecies (for example, 1,000 or 10,000ms gating intervals) measurement becomes quite slow. A faster and more accurate method of measuring low frequencies (<100Hz.) is to measure the period using MODE 16 (next section) and derive the frequency from the reciprocal.

On entry the following parameters should be initialized:

```
MD%     = 15                    (MODE)
FRQ%(0) = 1000                  (gating interval in mS, 10 - 32767)
FRQ%(1) = X                     (value does not matter)
FLAG%   = X                     (value does not matter)
```

Then

```
CALL DAS8 (MD%, FRQ%(0), FLAG%)
```

On return the variables contain data as follows:

```
MD%     = 15                    (unchanged)
FRQ%(0) = 1000                  (unchanged)
FRQ%(1) = data                  (count proportional to frequency)
```

The following Error Codes apply to MODE 15:

If FLAG% = 0 (no error, OK)
= 1 (Base Address unknown)
= 2 (MODE number out of range, <0 or >24)

The frequency should be derived from the change in count returned by FRQ%(1) and the gating interval FRQ%(0) as follows:

```
xxxxx  FREQ = FRQ%(1) * 1000 / FRQ%(0)
```

Note that a slightly more accurate result is obtained if account is taken of the real output interval of Counter 2 (DAS-8 only) which is 0.99985ms instead of 1ms. This correction error is small and can usually be ignored, but for perfectionists, the correction can be included in the frequency calculation above:

```
xxxxx  FREQ = FRQ%(1) * 1000 / ( FRQ%(0) * 0.99985 )
                                          DAS-8 only
```

2.3.17 MODE 16: Measure Period Or Pulse Width

MODE 16 measures Pulse Width by triggering Counter 2 (positive going edge from input signal) and halting Counter 2 on the negative going edge. This results in determination of the half period of any repetitive waveform or the duration of a positive pulse. DAS-8 timing is derived from the system clock (2.38636 MHz, bus dependent) with each decrement being 0.41905 us while the DAS-8PGA provides its own timing @ 1.000 MHz. Maximum period measurements are 27.46 ms (DAS-8) and 65.54 ms (DAS-8PGA). The signal for period or pulse duration measurement must be externally connected to the gate input of Counter 2 (pin 23) and IP2 (pin 26). MODE 16 performs all the necessary counter initialization and timing operations. Note that Counters 0 & 1 are not used and are free for other operations.

On entry the following parameters should be initialized:

```
MD%     = 16                    (MODE)
PW%     = X                     (value does not matter)
FLAG%   = X                     (value does not matter)
```

Then

```
CALL DAS8 (MD%, PW%, FLAG%)
```

On return the variables contain data as follows:

```
MD%     = 16                    (unchanged)
PW%     = -32768 thru 32767     (counts decremented)
```

The following Error Codes apply to MODE 16:

If FLAG% = 0 (no error, OK)
= 1 (Base Address unknown)
= 2 (MODE number out of range, <0 or >24)

Note that counts greater than 32767 will be returned as negative integers to and stored in 2's complement form (see Appendix C).

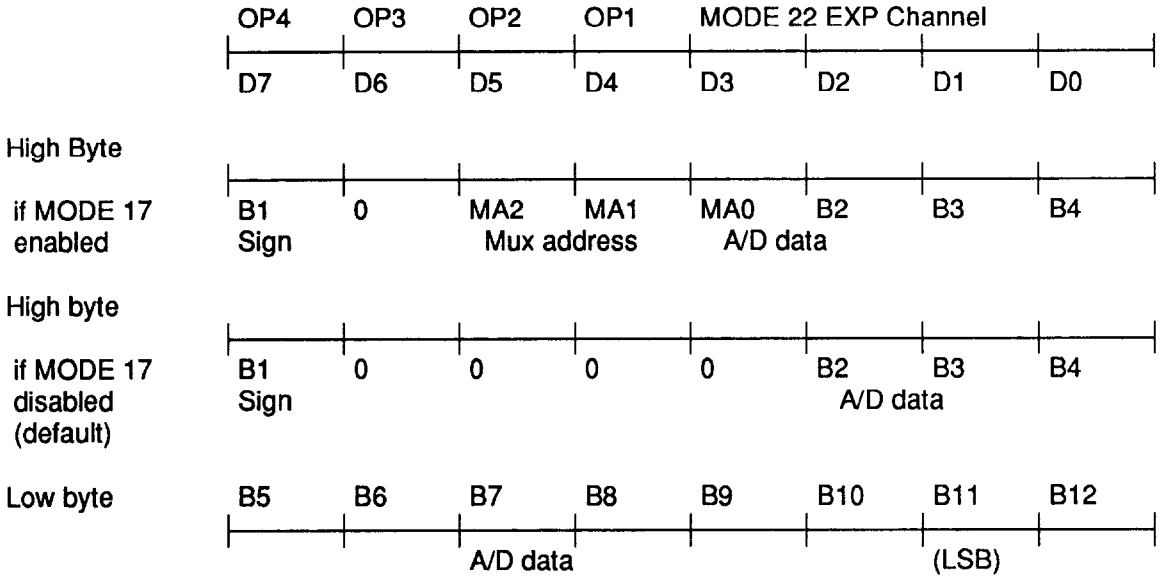
The following short BASIC program returns a true count into a real variable (PULWID) and scales the count to microseconds:

```
xxx10   IF PW% <0 THEN PULWID = 65536 + PW% ELSE PULWID = PW%
xxx20   PULWID = PULWID * 0.41905  'change to microseconds (DAS8)
xxx30   PERIOD = 2 * PULWID        'for symmetric waveform
```

Line xxx20 may be eliminated for DAS-8PGA since the multiplier is 1.

2.3.18 MODE 17: Enable/Disable Channel Tag

MODE 17 "tags" analog data with its respective input channel #. MODE 17 is defaulted to disable tagging. Since the A/D data requires 12 bits and the multiplexer channel number (0-7) uses 3 bits, the two can be combined and stored in a single integer variable (2 bytes). Conversion data is stored as an integer variable (2's complement):



On entry the following parameters should be initialized:

```
MD%     = 17                    (MODE #)
EN%(0)  = 0                     (0=disable; 1=enable)
FLAG%   = X                     (value does not matter)
```

Then

```
CALL DAS8 (MD%, EN%, FLAG%)
```

On return the variables remain unchanged:

```
MD%     = 17                    (unchanged)
EN%(0)  = 0 or 1                (unchanged)
```

The following Error Codes apply to MODE 17:

If FLAG% = 0 (no error, OK)
= 1 (Base Address unknown)
= 2 (MODE number out of range, <0 or >24)
= 13 (trigger channel number out of range <0 or >7)

Packing the channel number & data together maximizes memory and/or disk storage space. If packed data is returned to an integer array (Z%), separation may be accomplished by isolating and right justifying the EXP channel # (D6-D3) as follows:

```
CHANNEL% = (Z%(I) AND &H7800)/2048
```

If data is negative, the sign must be moved to where the channel number was so that it immediately precedes the data. This is accomplished as follows:

```
IF Z%(I) AND &H8000 THEN Z%(I) = (Z%(I) AND &H87FF) OR &H7800
```

Z%(I) is pure data and may be retrieved (see QBM22INT.BAS for usage).

2.3.19 MODE 18: Multiple A/D Conversions Via Software Trigger

MODE 18 is an analog threshold trigger with a user-efinable setpoint (per channel). As such, triggering is controlled by software (comparison of input signal to threshold value). Once this threshold is exceeded, "N" A/D conversions occur.

On entry the following parameters should be initialized:

```
MD%     = 18                          (MODE #)
D%(0)   = Segment address of ARRAY*   (Offset of 0 is assumed)
D%(1)   = Number of conversions       (Max number is 32768)
D%(2)   = Trigger channel             (0-7 independent of MUX Range)
D%(3)   = Center of trigger window    (0-4095 or +/-2047: Uni-Bipolar)
D%(4)   = Upper/lower limit           (0-2047)
D%(5)   = Trigger slope               (0 positive, 1 negative)
```

* FORTRAN, PASCAL, C, and TurboBASIC pass the ARRAY Segment address as **D%(0) = VARSEG(ARRAY%(0))**

whereas, GW BASIC and QuickBASIC pass the Segment address as **D%(0) = VARPTR(ARRAY%(0))**

NOTE The trigger boundary defines a region +/- the trigger level. For a valid trigger to occur the signal must pass through this region with the correct slope.

FLAG may return the following Errors:

If FLAG% = 0 No errors.
= 1 Base address unknown.
= 6 A/D time out.
= 13 Trigger channel out of range.
= 14 Trigger level out of range.
= 15 Invalid slope value.

NOTE: Error 6 indicates a hardware fault (See MODE 4).

2.3.20 MODE 19: Set DAS-8PGA (Programmable Gain Amplifier)

MODE 19 configures the PGA for input range and UNIPOLAR or BIPOLAR operation. The default range is ±5 Volts. The PGA option is configured as follows:

RANGE (VOLTS)	GAIN CODE
±5	0
±10	8
0 to 10	9
±0.5	10
0 to 1	11
±0.05	12
0 to 0.1	13
±0.01	14
0 to 0.02	15

On entry the following parameters should be initialized:

```
MD%     = 19                    (MODE #)
D%(0)   = PGA                   (gain code given above)
```

FLAG may return the following Error Codes:

If FLAG%	=0	No errors.
	=1	Base address unknown.
	=16	Invalid PGA gain code.
	=17	Function unavailable (No DAS-8PGA)

2.3.21 MODE 20: Return Interrupt State/PGA Status

MODE 20 is used for status determination of interrupt state (MODE 8) as well as for DAS-8PGA determination (present or not).

On return, the variables contain:

```
MD%     = 20
D%(0)   = Interrupt status 0=done, 1=active.
D%(1)   = Number of conversions done to this point.
D%(2)   = Current channel number in DAS8-MUX.
D%(3)   = PGA gain code (see above), (-1=PGA not present).
```

FLAG may return the following Error Codes:

If FLAG%	= 0	No errors.
	= 1	Base address unknown.

2.3.22 MODE 21: "N" Conversions In "X" Channel Bursts

MODE 21, converts "X" A/D channels on each interrupt generated by the DAS-8/PGA/AO. The advantage of this MODE is that interchannel skew is small (420 us (max) for eight channels on an IBM-PC). Interchannel skew can be eliminated altogether by using MetraByte's SSH-4 (requires special cable).

MODE 21 uses OUT-2 (Pin 6) to generate a square-wave frequency source. OUT-2 is tied to INT-IN (Pin 24), IP-1 (Pin 25), and the sample/hold line of MetraByte's SSH-4 (optional). Each interrupt is triggered by the rising edge of OUT-2 (this directs an SSH-4 to sample data). Once interrupted, the software waits for the falling edge of OUT-2, by polling IP-1, triggering A/D conversions. This ensures an SSH-4 is holding data. MODE 1 CALL (assign "X" channels) should precede a MODE 21 CALL in order to set up the channels to be scanned.

To control the start of data collection two optional triggers are provided. A hardware trigger, IP-2 (Pin 26), will be polled until it goes high. A software trigger, like MODE 18's, is also provided.

To run this MODE correctly requires three calls to MODE 21. The first call sets up the address of the data array. The call is set up as follows:

```
MD%  = 21                                  (MODE #)
D%(0)  = 0                                 (Setup data array)
D%(1)  = VARPTR(A%(0))[1]                  (BASICA's PTR to data array)
D%(2)  = # of array elements corresponding to the # of conversions
D%(3)  = DAS-8PGA 2 thru 7[2]              (Interrupt level)
```

The second call initiates the interrupt and enables the optional software/hardware triggers. The call is set up as follows:

```
MD%  = 21                                  (MODE #)
D%(0)  = 1                                 (Start data taking with triggers)
D%(1)  = 0                                 (no software trigger)
       = 1                                 (trigger on positive slope)
       = 2                                 (trigger on negative slope)
```

If D%(1) isn't 0 then D%(2-4) contains additional software trigger information

```
D%(2)  = Trigger channel                   (0 - 7)
D%(3)  = Trigger level                     (+/- 2047 Bipolar)
                                           (0 to 4095 Unipolar)
D%(4)  = Trigger boundary                  (0 to 2047)[3]
D%(5)  = 0 disables IP-2 trigger
```

To complete the conversion, the user must perform additional calls to MODE 21 with D%(0)=2. On return from this call, D%(3) contains the number of conversions performed. The user must call this MODE until D%(3) returns the total number of conversions requested. The call is set up as follows:

```
MD%    = 21                                (MODE #)
D%(0)  = 2                                 (Monitor conversions)
D%(1)  = 1                                 (to terminate MODE 21 immediately)
D%(2)  = 0
```

On return:

```
D%(3)  = Number of conversions done
```

Possible return errors in FLAG%

0 (No errors)
1 (Base address unknown)
7 (Interrupt out of range)
13 (Trigger channel out of range)
14 (Trigger level out of range)
15 (Invalid slope value)
19 (MODE 21 D%(0) out of range 0-2)

1 Note: FORTRAN, C, PASCAL, & TurboBASIC pass the ARRAY segment address as **D%(0) = VARSEG(ARRAY%(0))**

whereas, GW BASIC & QuickBASIC pass the segment address as **D%(0) = VARPTR(ARRAY%(0))**

2 On the UCDAS8-PGA the interrupt level isn't required. It is derived directly from the boards registers.

3 The trigger boundary defines a region +/- the trigger level. For a valid trigger to occur the signal must pass through this region with the correct slope.

2.3.23 MODE 22: Multiple A/D Conversions With Interrupts Using EXP-16

MODE 22 is an interrupt driven data acquisition routine for use with one or more EXP-16s (8, max). The interrupt level is set via MODE 6 and the interrupt itself may be generated by the DAS-8/PGA/AO counter or by an external TTL compatible signal. The scan limits on the DAS-8/PGA/AO must first be set using MODE 1. Upon interrupt, a single EXP-16 channel will be read for each DAS-8/PGA/AO channel within the scan limit. Upon channel scan completion, the EXP-16 will be incremented for the next interrupt. Subsequent interrupts trigger the process on the new EXP-16 channel. MODE 22, like MODE 8 allows background data acquisition while other tasks/programs are operating in the foreground. If the channel tag (MODE 17) is used, the 4 high order bits appended to the data will reflect the EXP-16 channel read (0-15) and not the DAS-8/PGA/AO channel. Data in memory will appear as follows:

Interrupt #1	1xx	EXP-16 Ch1 for lowest DAS-8/PGA/AO channel selected.
	1xx	EXP-16 Ch1 for highest DAS-8/PGA/AO channel.
Interrupt 2	2xx	EXP-16 Ch2 for lowest DAS-8/PGA/AO channel selected.
	2xx	EXP-16 Ch2 for highest DAS-8/PGA/AO channel.

Upon completion of MODE 22 (via MODE 20 or otherwise), MODE 22 should be called again with D%(0)=0 to disable the interrupt routine.

On entry the following parameters should be initialized:

```
MD %    = 22                              (MODE #)
D%(0)   = Interrupt vector (+ recycle) Level 2-7 (+ 8 for recycle)
D%(1)   = Segment Address of data array (OFFSET 0 required)
D%(2)   = Number of conversion            (1 thru 32766)
D%(3)   = EXP-16 Low scan limit
D%(4)   = EXP-16 High scan limit
D%(5)   = # of readings to average **  (0 thru 100)
```

If FLAG% = 0 No errors
= 1 Base address Unknown
= 7 Interrupt level out of range
= 8 Transfer parameters out of range
= 20 EXP-16 channel out of range

** Input signals higher than 15Hz may require removal of the 1uF capacitor from each EXP-16 input channel. Removal of this Cap will increase input signal noise. Signal averaging will substantially reduce noise.

2.3.24 MODE 23: Output Data To DACs (DAS-8/AO Only)

MODE 23 Writes data to DAC Channels 0 or 1 when configured for normal updates. If the DACs are set for simultaneous update, MODE 23 updates both channels at the same time.

On entry, the following parameters should be initialized:

```
MD%     ->  23
D%(0)   ->  DAC SELECT
            0 = DAC 0
            1 = DAC 1
            2 = Both (Simultaneous update)

D%(1)   ->  DAC 0 Data (0 - 4095)
D%(2)   ->  DAC 1 Data (0 - 4095) (used only if D%(0) = 2)
```

FLAG may return the following Error Codes:

0 = No errors.
9 = Invalid DAC selected.
10 = DAC data out of range.

2.3.25 MODE 24: Update DACs From Array On Interrupt (DAS-8/AO Only)

MODE 24 updates DAC 0 or 1 on Interrupt when configured for normal updates. If the DACs are set for simultaneous update, MODE 24 updates both DAC channels.

On entry, the following parameters should be initialized:

```
MD%     ->  24
D%(0)   ->  DAC SELECT
            0 = DAC 0
            1 = DAC 1
            2 = Both (Simultaneous update)

D%(1)   ->  INTERRUPT LEVEL (2 - 7)
D%(2)   ->  NUMBER OF SAMPLES (0 = Continuous or 1 - 32767)
D%(3)   ->  NUMBER OF SAMPLES
```

If GWBASIC:

```
D%(4)   - >  SEGMENT = -1
```

Else:

```
D%(4)   ->   SEGMENT OF DATA ARRAY
D%(5)   ->   OFFSET OF DATA ARRAY
```

Flag may return the following error codes:

0 = No Error.
11 = Interrupts Already Enabled.
12 = Invalid DAC Channel Selected.
13 = Interrupt Level Out Of Range.

* * * * *

Chapter 3
ERROR CODES

SUMMARY OF ERROR CODES

If for any reason the FLAG% variable is returned non-zero, then an error has occurred in the input of data to the CALL routine. Checking of data occurs first in the routine and no action will be taken if an error condition exists. An immediate return will take place with the error specified in the FLAG% variable.

ERROR CODE	FAULT
0	No error, function completed successfully.
1	Base address unknown. (I/O location not initialized)
2	Mode number <0 or >24.
3	Invalid Base Address, <256 or >1016.
4	Mux scan limits out of range or reversed order, <0 or >7.
5	Channel number out of range, <0 or >7.
6	A/D timeout. Incorrect busy signal from A/D.
7	Interrupt level out of range, <2 or >7.
8	Mode 8 data transfer parameters out of range.
9	Mode 9 data transfer parameters out of range.
10	Counter number out of range, <0 or >2.
11	Counter configuration number out of range, <0 or >5.
12	Digital output data out of range, <0 or >15.
13	Mode 18 Trigger channel number out of range <0 or >7.
14	Mode 18 Trigger level out of range, <2048 or >2047.
15	Mode 18 Trigger slope not 0 or 1.
16	PGA gain codes illegal <0 or >3.
17	PGA function not available (no DAS-8PGA).
18	Mode # unavailable for this language.
19	Mode 21 not initialized.

Error detection after the CALL routine is easily implemented:

```
xxx10   CALL DAS8 (MD%, D%, FLAG%)
xxx20   IF FLAG%  < > 0 THEN GOSUB yyyyy
.
.
yyyyy   REM: Error handling subroutine
.
.
zzzzz   RETURN
```

This is useful while debugging a new program or within a program containing a suitable error handling subroutine.

* * * * *

Chapter 4

PROGRAMMING EXAMPLES

4.1 LANGUAGE INTERFACING

To simplify programming, the DAS-8 basic driver uses a single set of routines callable by MicroSoft PASCAL, C, FORTRAN, and other compilers supporting the MS-FORTRAN calling conventions. To avoid conflicts with DAS-8 CALL routines, a set of "include" files has been written for each language. The routines are contained in DAS8.OBJ and are linked by the linker LINK.EXE. The following entry points (*) are provided in DAS8.OBJ:

ALLMEM()	Allocates array memory (**).
FREMEM()	Frees any memory allocated by ALLMEM (**).
RETSEG()	Returns a segment value for DAS-8 MODEs 8, 18, and 21.
INPCHR()	Returns a character from the keyboard (if any).
DAS-8()	Entry point into DAS-8 driver.

** Unsupported in FORTRAN because of language limitations.

A simple graphics library, DRAW.LIB, is also included for plotting points and lines in both CGA and EGA modes. DRAW.LIB uses the same calling conventions as DAS8.OBJ. The screen coordinates used in DRAW.LIB assign the CRT's center as point (0,0). In addition, DRAW.LIB has scaled the screen for approximately equal X and Y (Horiz/Vert) pixel spacing. The following entry points are provided in DRAW.LIB:

SETUP()	Used to switch modes between text/graphics.
CLEAR()	Used to set background color.
COLOR()	Used to set foreground color.
POINT()	Point plotting routine (graphic modes).
LINE()	Line plotting routine (graphic modes).

The example programs cover DAS-8 Modes 18 and 20. Also, a program for DAS-8PGA (Mode 19) set-up has been included to illustrate the interface between the DAS-8 and each particular language.

* All entry points are declared as FAR references.

4.2 PROGRAMMING IN PASCAL

PASCAL programs on the PCF-8 disk begin with the letter P and should be copied (P*.*) to a working disk (or hard drive) along with DAS8.OBJ and DRAW.LIB. PASCAL programs are set up as follows:

```
Program MODE 19(INPUT,OUTPUT);
{ $include: 'pdas8.inc' }          <---Contains references to DAS8.OBJ
                                       and Creates PUBLIC variables
                                       MD, DIO, & FLAG.
```

```
{ $include: 'pdraw.inc' }           <---Contains all references to
                                        DRAW.LIB.

{ $include: 'psubs.inc' }           <---Contains user modifiable PASCAL
                                        subroutines.
VAR
  r:integer;                        { holds range value }
begin
  MD := 0;                          { Initialize base address }
  DIO[0] := 16#300;                 { user setable base address }
  FLAG := DAS8( MD, DIO[0] );       <---Sample call set up to the DAS-8

  if FLAG <> 0 then
    begin
      writeln('Error in initialization'); ABORT(NULL,0,0);
    end;

  writeln(chr(16#1B),'[2J');        { clear screen (ANSI.SYS must be
                                        loaded) }

  writeln('          Selection          Range    ');
  writeln('          ----------         -----    ');
  writeln('          +/-5V              0      ');
  writeln('          +/-10V             1      ');
  writeln('          0 to +10V          2      ');
  writeln('          +/-0.5V            3      ');
  writeln('          0 to +1V           4      ');
  writeln('          +/-50mV            5      ');
  writeln('          0 to +100mV        6      ');
  writeln('          +/-10mV            7      ');
  writeln('          0 to +20mV         8      ');
  writeln;
  write('Enter input range for DAS-8PGA:  ');
  readln(r);

  if r<>0 then r:=r+7;              { set up proper das8-PGA range }
  MD := 19;                         { SET DAS-8PGA }
  DIO[0] :=r;                       { range number 0,8-15 }
  FLAG := DAS8( MD, DIO[0] );

  if FLAG <> 0 then
  begin
    writeln('Error in setting PGA'); ABORT(NULL,0,0);
  end;
end.

-------------------------------------------------------------------------
Programs may be compiled and linked as follows:
  PAS1 PMODE19,,;
  PAS2
  LINK/map PMODE19+DAS8,,,math+pascal+DRAW
```

4.3 PROGRAMMING IN TURBO PASCAL

Turbo PASCAL supports a compact and a large memory model. The compact model supports one code segment and multiple data segments. In this model, code segment is limited to 64K with Assembly routine calls being near calls. The data segment is unlimited. The large model permits unlimited code and data segments with Assembly calls and data access calls being far calls.

The program TINST.EXE, shipped with Turbo PASCAL, can change the calling convention so that the user may not know which convention is in use. The Defaultstate is in Off or Compact mode. To determine which mode is in use, run TINST.EXE.

Model:	Compact (Forces far call Off in TINST.EXE).
Passes:	dword size pointers (offset and segment).
Sequence:	Arguments passed left to right.
Default Calling Convention:	Arguments passed by value.

Example:

PASCAL Call: Result =	tp_dasg (Var1, Var2);
PASCAL Declaration:	FUNCTION tp_dasg (VAR Var1:integer; VAR Var2);
.ASM Subroutine:	(Either Model)

The following assembly code shows how the driver handles user arguments.

```
tp_das8         proc near       ;near call(single word return address)
                push bp         ;save base pointer
                mov bp,sp       ;save stack pointer
                .               ;[bp+10] offset of VAR1
                .               ;[bp+8]
                .               ;[bp+6] offset of VAR2
                .               ;[bp+4]
                .               ;
                .               ;
                mov ax,n        ;return Value for Function in ax register
                pop bp
                ret 8           ;return & pop values prior to exit
tp_das8         endp
```

Other:

This information is offered to those wishing to create their own drivers.

- Use the $L 'Metacommand' to link the object file containing external function tpdas8; that is, use **$L tpdas8** to link to file tpdas8.
- The VAR declarative forces pass by reference (address of variable) in the function declaration. Default is pass by value (pushing the actual integer valu onto thee stack.
- tp_das8 is declared external in the calling program along with the type of return value (integer). Remember that in PASCAL, functions return a value whereas procedures never do.
- The .ASM file contains an explicit declaration of the code segment containing tp-das8. Turbo PASCAL handles segments in a primitive manner not compatible with the '.model' statements

available in MASM or TASM. The function tp_das8 must reside in a segment called "CODE." Turbo PASCAL will not accept any other segment name. If tp_das8 is not in segment "CODE," the linker returns an "unresolved external" error. The Segment Declaration for "CODE" in the .ASM file must appear as:

```
CODE SEGMENT WORD PUBLIC
ASSUME CS:CODE
.
.                    ;CODE GOES HERE
.
CODE ENDS
```

Turbo PASCAL Example:

```
{$R-}
{$I-}
{$B+}
{$S+}
{$N-}
{$L tpdas8}
{$M 65500,16384,655360}

{
*******************************************************
*                                                     *
*    Demonstration program using MODE 0 *
*    Compiled with Borland Turbo PASCAL *
*                                                     *
*******************************************************
}

TYPE
PArray = array [1..7] of word;
Var
  DIO       : PArray;
  MD,FLAG   : integer;

(* Define the Turbo Pascal Function Call *)

FUNCTION TP_DAS8(VAR MD:integer;VAR DIO:PArray):integer;external;

begin

  Print_INTRO          {print introductory message }

  MD := 0              {Initialize Base Address }
  DIO[1] := 768;       {User set able Base Address }

  FLAG := tpdas8( MD, DIO ); {Call MODE 0 }

  if(FLAG <> 0) then
  ReportError;

end.
```

4.4 PROGRAMMING IN C

C programs on the PCF8 disk begin with the letter C. These should be copied to your working drive (C*.*) along with DAS8.OBJ and DRAW.LIB.

C programs are set up as follows:

```
#include <stdio.h>                    /*Required for basic I/O*/
#include <cdef.h>                     /*defines WORD as "unsigned integer"*/
#include <cdraw.h>                    /*Contains all references to DRAW.LIB*/
#include <cdas8.h>                    /*Contains all references to DAS8.OBJ
                                        and Creates PUBLIC variables MD,
                                        DIO, & FLAG*/


main ()

{
  int r;                              /* holds a range value */
  MD = 0;                             /* Initialize base address */
  DIO(0) = 0x300;                     /* user setable base address */
  FLAG = DAS8(&MD, DIO);              /*sample call setup for C*/

  if (FLAG != 0)
    { printf("\\nError in initialization\\n"); return(0); };

  printf("\\x1B[2J");                 /* clear screen (ANSI.SYS must be
                                        loaded)*/


  printf("  Selection Range           \\n");
  printf("  --------- -----           \\n");
  printf("  +/-5V     0               \\n");
  printf("  +/-10V    1               \\n");
  printf("  0 to +10V 2               \\n");
  printf("  +/-0.5V   3               \\n");
  printf("  0 to +1V  4               \\n");
  printf("  +/-50mV   5               \\n");
  printf("  0 to +100mV               6   \\n");
  printf("  +/-10mV   7               \\n");
  printf("  0 to +20mV                8   \\n\\n");
  printf("Enter range you would like to set das8-PGA to:  ");
  scanf("%d",r);

  if (r!=0) r=r+7;                    /* set up proper das8-PGA range */

  MD = 19;                            /* SET DAS-8PGA */
  DIO(0) = r;                         /* range number 0,8-15 */
  FLAG = DAS8( &MD, DIO );

  if (FLAG != 0)
    { printf("\\nError in setting PGA\\n"); return(0); };

  return(0);                          /* normal completion */
}

                                      /* Include additional routines used by
```

```
                                        C programs*/
  #include <csubs.h>                 /*Contains user modifiable C
                                        subroutines*/

-------------------------------------------------------------------------
C programs may be compiled and linked as follows:

MSC CMODE19 /AS;      <- small MODEl chosen here ...
LINK/Se:255/CP:1 CMODE19+DAS8,,,DRAW;
```

4.5 PROGRAMMING IN TURBO C

Medium Model:

Model:	Medium ("-mm") switch on command line.
Passes:	word size pointers (offset, no DS register)
Sequence:	Arguments passed right to left.
Default Calling Convention:	Arguments passed by value.

Example

'C' Call:	tcm_das8 (&Mode, Params);
'C' Declaration:	extern void tcm_das8(int*,int*);
.ASM Subroutine:	

The following assembly code shows how the driver handles user arguments.

```
_tcm_das8       proc far        ;dword pointer return address
                push bp         ;save base pointer
                mov bp,sp       ;save stack pointer
                .               ;[bp+6] holds offset of Mode
                .               ;[bp+8] holds offset of Params
                .               ;Program execution here
                .               ;
                .               ;
                pop bp          ;restore bp & sp prior to exit
                ret             ;return
_tcm_das8 endp
```

Other:

This information is for those wishing to create their own drivers.

- _tcm_das8 is declared "PUBLIC" in the .ASM file.
- tcm_das8 is declared "extern" in the C file.
- The .ASM file contains the ".model medium" directive (MASM & TASM only).
- Add leading underscore "_" to all tcm_das8 occurences in the .ASM file.

- tcm_das8 must be in a segment fname_TEXT (where *fname* is the name of the file in which tcm_das8 resides); else Linker returns an error.

Large Model

Model:	Large (-ml) switch on command line
Passes:	dword size pointers (offset and DS register)
Sequence:	Arguments passed right to left
Default Calling Convention:	Arguments passed by value

Example

'C' Call:	tcl_das8 (&Mode, Params);
'C' Declaration:	extern void tcl_das8 (int*,int*);
.ASM Subroutine:	

The following Assembly code shows how the driver handles user arguments.

```
_tcl_das8       proc far        ;dword pointer return address
                push bp         ;save base pointer
                mov bp,sp       ;save stack pointer
                .               ;[bp+8] holds Mode
                .               ;[bp+6]
                .               ;[bp+12] holds Params
                .               ;[bp+10]
                .               ;Program execution here
                .               ;
                .               ;
                pop bp          ;restore bp & sp prior to exit
                ret             ;return
_tcl_das8 endp
```

Other:

This information is for those wishing to create their own drivers:

- _tcl_das8 is declared "PUBLIC" in the .ASM file.
- tcl_das8 is declared "extern" in the C file.
- The .ASM file contains the ".model large" directive (MASM & TASM only).
- Add leading underscore "_" to all tcl_das8 occurences in .ASM file.
- Both code and data use dword (segment/offset) pointers.
- tcl_das8 must be in a segment fname_TEXT (where *fname* is the name of the file in which tcl_das8 resides); else Linker returns an error.

Turbo C Example:

```
#include <stdio.h>
#include <math.h>
#include <dos.h>

/* DECLARE THE PROPER FUNCTION CALL */

extern int far tcm_das8(int near *, int near *);

main()
{

   int A[1000];                        /* date array */
   int MD, DIO[6], FLAG;
   char far *temp;                     /* temp char buffer */
   int ns, s;                          /* holds number of samples */
   int x;
   unsigned seg, offs;

   Print_INTRO();                      /* print introductory message */

/***** use MODE 0 to initialize the DAS8/AO *****/

   MD = 0;                             /* Initialize Base Address */
   DIO[0] = 0x30;                      /* Base Address */

   FLAG = tcm_das8( &MD, DIO );        /* Medium Model Call */

   if (FLAG != 0)
      {
        printf("\nError in initialization\n");
        return(0);
      }

}
```

4.6 PROGRAMMING IN FORTRAN

FORTRAN programs on the PCF8 disk begin with the letter F. These should be copied to the working drive along with DAS8.OBJ and DRAW.LIB. The following is an example of setting up a program in FORTRAN:

```
$include: 'fdas8.inc'                  <--- File contains references to
                                            DAS8.OBJ and Creates PUBLIC
                                            variables MD, DIO, & FLAG
$include: 'fdraw.inc'                  <--- File contains all references
                                            to DRAW.LIB ...

C*                                     ** holds range number **
        integer*2 r
C*                                     *** Initialize via DIO(1) ***
        MD = 0
        DIO(1) = 16#300
```

```
        FLAG = DAS8(MD, DIO)
        if (FLAG.ne.0) then
          write(*,*) 'Error in initialization'
          stop
        endif
C*                                      clear screen ANSI.SYS must be loaded
        write(*,*) chr(16#1B),'[2J'
        write(*,*) '                    Selection  Range'
        write(*,*) '                    ------------  ---------'
        write(*,*) '                    +/-5V  0'
        write(*,*) '                    +/-10V 1'
        write(*,*) '                    0 to +10V  2'
        write(*,*) '                    +/-0.5V3'
        write(*,*) '                    0 to +1V   4'
        write(*,*) '                    +/-50mV5'
        write(*,*) '                    0 to +100mV    6'
        write(*,*) '                    +/-10mV7'
        write(*,*) '                    0 to +20mV 8'
        write(*,*) write(*,*) 'Enter range you would like to set das8-PGA to:
'
        read(*,*) r
C*                                      set up proper das8-PGA range
        if (r.ne.0) then
          r=r+7
        endif
C*                                      set DAS-8PGA
        MD = 19
C*                                      **range number 0,8-15
        DIO(1) = r
        FLAG = DAS8( MD, DIO )
        if (FLAG.ne.0) then
        write(*,*)'Error in setting PGA'
          stop
        endif
end
$include: 'c:fsubs.inc'                 <--- Contains user modifiable FORTRAN
                                            subroutines

-------------------------------------------------------------------------
FORTRAN programs may be compiled and linked as follows (modifications may be
required for certain machine comfigurations):
        FOR1 FMODE19;
        PAS2
        LINK/map PMODE19+DAS8,,,math+fortran+DRAW
```

4.7 GENERAL NOTES

The PASCAL compiler used is MicroSoft(MS) MS-PASCAL V3.31.

The TurboPASCAL compiler is Borland (Version 3.0 - 4.0).

The C compiler used is MS-C V4.0.

The Turbo C compiler is Borland

The Fortran compiler used is MS-FORTRAN V3.31. (Earlier versions may not be compatible)

Some machines may not have enough memory to compile these examples. In this case, the examples may be segmented and compiled separately.