ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Keithley DAS-HRES* at our website: *Click HERE*

DAS-HRES

User's Guide

KEITHLEY

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

C.E. LAB

DAS-HRES

Keithley Instruments, Inc. Data Acquisition Division

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

User Guide

for the

Keithley MetraByte

DAS-HRES

16-Bit

Analog & Digital

I/O Board

1730

Revision D - February 1992
Copyright © Keithley Instruments, Inc. 1989
Part Number: 24815

Keithley Instruments, Inc. Data Acquisition Division

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

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Warranty Information

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

Warning

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

Disclaimer

Information furnished by Keithley Instruments, Inc. Data Acquisition Division is believed to be accurate and reliable. However, the Keithley Instruments, Inc. Data Acquisition Division 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 Instruments, Inc. Data Acquisition Division.

Note:

Keithley MetraByte™ is a trademark of the Keithley Data Acquisition Division of Keithley Instruments.

BASIC™ is a trademark of Dartmouth College.

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

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

Microsoft® is a registered trademark of Microsoft Corporation.

Turbo C® is a registered trademark of Borland International.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Contents

CHAPTER 1: INTRODUCTION

1.1 Summary Of Functions 1-1
1.2 Additional Software Support 1-3

CHAPTER 2: SETUP & INSTALLATION

2.1 Computer Type 2-1
2.2 Copying Distribution Software 2-1
2.3 Hardware Installation 2-2

CHAPTER 3: REGISTER FUNCTIONS

3.1 Register Structure & Format 3-1
3.2 A/D Registers (Base Address +0 & +1) 3-1
3.3 MUX Scan Register (Base Address +2) 3-2
3.4 Digital I/O (Base Address +3) 3-3
3.5 D/A Output Registers (Base Address +4 to +7) 3-4
3.6 Status Register (Base Address +8) 3-5
3.7 DAS-HRES Control Register (Base Address +9) 3-6
3.8 Timer-Counter Enable Register (Base Address +10) 3-7
3.9 Programmable Gain Register (Base Address +11) 3-7
3.10 8254 Programmable Interval Timer Counter (Base Address +12 To +15) 3-8
3.8 DAS-HRES Autocalibration Software 3-8

CHAPTER 4: PROGRAMMING

4.1 List Of Modes 4-1
4.2 Descriptions Of Modes 4-2
MODE 0: Initialize 4-2
MODE 1: Set Multiplexer Scan Limits 4-6
MODE 2: Read Current Multiplexer Address & Scan Limits 4-7
MODE 3: Do One A/D Conversion & Increment MUX 4-8
MODE 4: Do N A/D Conversions Direct To Array 4-10
MODE 5: Do N A/D Conversions & Transfer To Memory On Interrupt 4-12
MODE 6: Do N A/D Conversions & Transfer To Memory Via DMA 4-15
MODE 7: Disable DMA/Interrupt Operation 4-17
MODE 8: Read DMA/Interrupt Status 4-18
MODE 9: Transfer Data From Memory To Array 4-19
MODE 10: Set Counter 0 Configuration 4-21
MODE 11: Load Counter 0 4-22
MODE 12: Read Counter 0 4-22
MODE 13: Write Digital Output OP0-7 4-23
MODE 14: Read Digital Input IP0-7 4-24
MODE 15: Output Data To One D/A 4-25
MODE 16: Output Data To Both D/As 4-25
MODE 17: Set Programmable Timer Rate 4-26
MODE 18: Analog Output/Input On Interrupt 4-27
MODE 19: Analog Trigger Function 4-29
MODE 20: A/D Scan Of Channels On Interrupt 4-31

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Contents

MODE 21: Set Channel Gains 4-33
MODE 22: Output A Square Wave Frequency 4-35
MODE 23: Allocate Memory For DMA 4-35
MODE 24: Deallocate Memory Segment 4-36
4.3 Summary Of Error Codes 4-37

CHAPTER 5: PROGRAMMING

5.1 Overview 5-1
5.2 Programming Tips For BASIC 5-1
5.3 Loading Machine Language Call Routine HRES.BIN 5-2
5.4 Format Of The Call Statement 5-5
5.5 QuickBasic Applications 5-7
5.6 Making Executable Programs 5-9
5.7 Programs, Examples, & Utilities 5-11
5.8 Assembly Language Programs & Calls In Other Languages 5-12
5.9 A Note On Execution Times 5-12
5.10 Multiple DAS-HRESs In One System 5-13

CHAPTER 6: PROGRAMMBLE INTERVAL TIMER

6.1 The 8254 Programmable Interval Timer 6-1
6.2 Reading & Loading The Counters 6-4
6.3 Timer-Counter Enable Register 6-7
6.4 Generating Square Waves Of Programmed Frequency 6-7
6.5 Measuring Frequency & Period 6-8
6.6 Generating Time Delays 6-8
6.7 Triggering The A/D Periodically 6-9
6.8 Generating Interrupts With The Counter/Timer 6-9

CHAPTER 7: D/A CONVERTERS

7.1 Description 7-1
7.2 Programming 7-2
7.3 Arbitrary Waveform Output 7-2

CHAPTER 8: GENERATING DATA FILES

CHAPTER 9: TEST & CALIBRATION

9.1 Overview 9-1
9.2 Interim Calibration 9-1

CHAPTER 10: APPLICATIONS

10.1 Connecting Analog A/D Inputs Correctly 10-1
10.2 Selecting Input Channel Range & Gain 10-2
10.3 Matching The Input Clock Frequency To The Application 10-3
10.4 Measuring Current & 4-20mA Current Loops 10-4

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Contents

10.5 Adding More Analog Inputs 10-4
10.6 Interface To Transducers, Thermocouples, Etc. 10-4
10.7 Precautions In Use: Noise, Groundloops, & Overloads 10-5

CHAPTER 11: FACTORY RETURNS

APPENDICES

Appendix A: I/O Connections
Appendix B: Specifications
Appendix C: Storage Of Integer Variables
Appendix D: Repairs & Service
Appendix E: Direct Memory Access Hardware

■ ■ ■

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 1

INTRODUCTION

1.1 SUMMARY OF FUNCTIONS

The Model DAS-HRES is a multi-function, high speed gain analog/digital I/O expansion board for the IBM Personal Computer. It is a full length board that installs internally in an expansion slot of an IBM PC, IBM PC/XT, PC/AT or RT PC or PS2 Model 25 or 30. and turn the computer into a fast high precision data acquisition and signal analysis instrument. The DAS-HRES can also be used in bus compatible computers such as Compaq, Tandy, AT & T 6300, Olivetti M24, Zenith, Leading Edge, Televideo, Tandon, Epson Equity, PC's Limited etc. The board is of multilayer construction with integral ground plane to minimize noise and cross talk at high frequencies. The following functions are implemented on the DAS-HRES:

1. DAS-HRES uses a 16-bit sampling A/D converter which provides a maximum throughput rate of 47.6KHz in DMA mode. The throughput does not vary with the gain selected.

GAIN	UNIPOLAR	BIPOLAR	THROUGHPUT
1	0 to +10V	±10V	47.6KHz
2	0 to +5V	±5V	47.6KHz
4	0 to +2.5V	±2.5V	47.6KHz
8	0 to +1.25V	±1.25V	47.6KHz

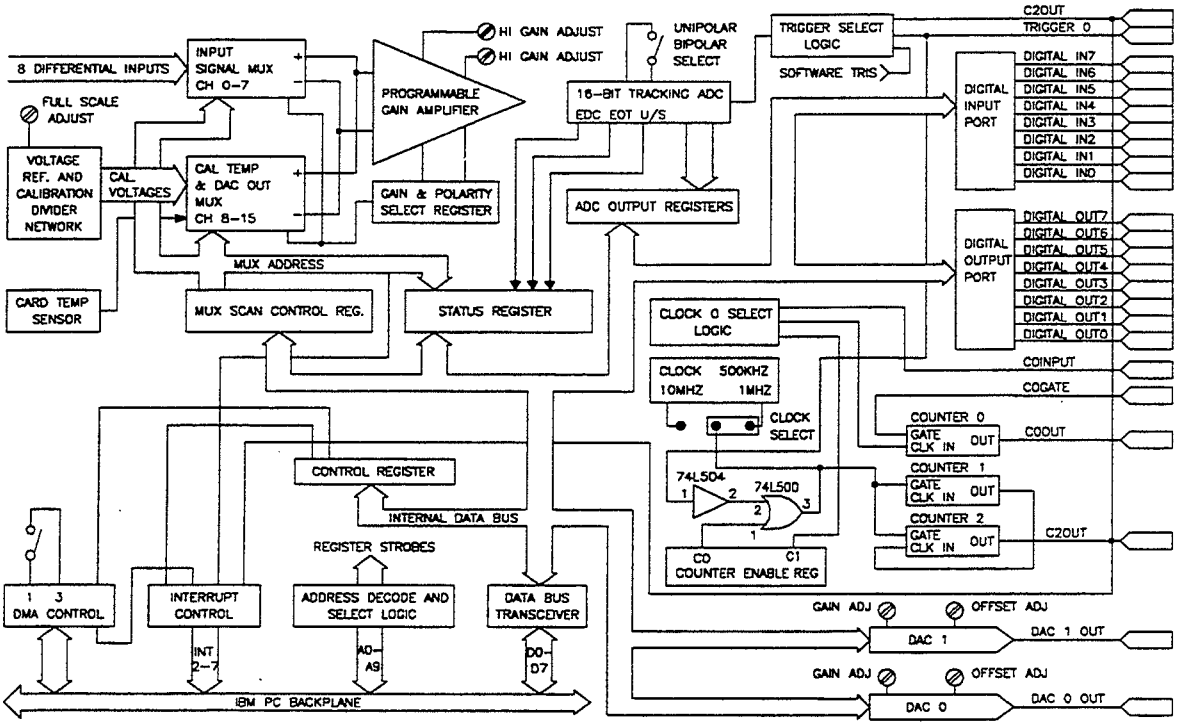
Note that the DAS-HRES gain is set in binary steps of 1, 2, 4 & 8. Each range has 16 bit resolution (65536 steps), so that resolutions of the DAS-HRES1 vary from 305 uV/bit (+/-10v) to 19.1 uV/bit (0 to +1.25v).

A slide switch controls the Unipolar (Zero to +Full Scale) or Bipolar (-Full Scale to +Full Scale) channel configuration. The maximum common-mode input voltage of all channels at any gain is ±10V. The common mode rejection of each channel is a function of gain and frequency. Minimum values are as follows:

GAIN	MINIMUM DC CMRR (DB)
1	90
2	100
4	100
8	110

2. A/D conversions may be initiated in any one of three ways: by software command, by internal programmable interval timer, or by direct external trigger to the A/D. At the end of the A/D conversion, it is possible to transfer the data by any of three ways: by polling a status register and reading data when ready, by generation of a hardware interrupt and an interrupt service routine, or purely by hardware through DMA (Direct Memory Access). All operating modes are selected by a control register on the DAS-HRES and are also supported by the utility software. A block diagram of DAS-HRES follows on the next page.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Block Diagram Of DAS-HRES

3. The Programmable Gain Amplifier is of full differential design, and the gain is selected by a 2-bit gain code through a read/write register at (Base Address +11). The gain setting is global; it affects all channels until changed. In non-DMA modes, the software driver can select different gains for each channel prior to conversion, so it is possible to perform scans with a variety of channel gains at lower speeds up to a few thousand conversions/second. Operation in the high speed DMA mode precludes any intervention from the software; thus, in this case, the gain setting is global and affects all channels. It is possible to design hardware that will select gains and channels in DMA operation, but the DAS-HRES does not offer this feature. If you need this capability, the DAS-20 Data Acquisition Board provides this capability.

4. A 3-channel programmable interval timer (Intel 8254) provides trigger pulses for the A/D at any rate from 50KHz to 1 pulse/hr. Two channels are operated in fixed-divider configuration from an internal 1, 8, or 10MHz crystal clock. The third channel is uncommitted and provides a gated 16-bit binary counter that can be used for event or pulse counting, frequency or pulse generation, delayed triggering, and (in conjunction with the other channels) for frequency and period measurement.

5. Two channels of 16 bit-D/A output. The D/A converters offer a fixed output range of ±10 Volts. D/As are double-buffered to provide 180 microsecond single-step updates. Arbitrary or periodic waveform generation using interrupts is supported by the utility software.

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

7. The DAS-HRES uses 16 consecutive locations in I/O address space. The Base I/O Address can be set on any 16-bit boundary by a 6-position DIP switch. This means that DAS-HRES can always be installed regardless of I/O address usage of other peripherals. No memory address space is used and there is no on-board initializing BIOS or microprocessors.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

8. A slide switch selects between the only two available levels, Level 1 or 3, for DMA operations. Interrupts may be enabled on any of Levels 2-7 and are software selected. Both DMA and interrupts are tristated when disabled and can share levels with other peripherals that have the same capability.

DAS-HRES can be used many of the same accessories as the DAS-20 (the EXP-20 Expansion Board, for example). The STA-HRES Screw Connector Board brings all DAS-HRES I/O connections out to easily accessed screw terminals.]

The utility software for DAS-HRES is provided on a PC-DOS 2.10, 360K format, 5-1/4" floppy disk (upward compatible with all later versions of DOS). 3-1/2" disks are also available. The Distribution Software includes

1. A machine language driver (HRES.BIN) for control of A/D, D/A, and digital I/O channel functions and data transfer modes via BASIC CALL from interpreted or compiled BASIC.
2. BASIC examples of each of the driver modes.
3. Quick Basic libraries and examples of each of the driver modes in Quick Basic.
4 Initial setup and installation aids.
5. Calibration and test program.
6. README.DOC provides recent information about new features not documented in this manual.

1.2 ADDITIONAL SOFTWARE SUPPORT

As a further convenience in application, the DAS-HRES is (or will be soon) supported by several optional higher-level software products available, as follows:

1. STREAMER high speed hard disk data transfer utility. Continuous A/D data transfer rates to hard disk in excess of 50KHz are possible on PC/AT, somewhat slower on PC/XT depending on hard disk and controller type.
2. LABTECH NOTEBOOK menu driven data acquisition and analysis software from Laboratory Technologies Corporation.
3. ASYST programmable data acquisition and analysis software from Asyst Software Technologies Inc.
4. ASYSTANT menu driven data acquisition and analysis software from Asyst Software Technologies Inc.
5. SNAPSHOT STORAGE SCOPE menu driven data acquisition and analysis software from HEM Data Corporation.
6. TTOOLS utilities for the Turbo-Pascal programmer by Quinn-Curtiss Software.

For further information on these products, please contact the manufacturer. Free demonstration disks are available for several of the above software packages.

■ ■ ■

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 2

INSTALLATION

2.1 COMPUTER TYPE

To install a DAS-HRES in an IBM PC/XT/AT and all compatibles running under the PC-MS/DOS operating system, proceed as described below.

2.2 COPYING DISTRIBUTION SOFTWARE

As soon as possible, make a working copy of your Distribution Software. You may put the working copy on diskettes or on the PC Hard Drive. In either case, making a working copy allows you to store your original software in a safe place as a backup.

To make a working copy of your Distribution Software, you will use the DOS *COPY* or *DISKCOPY* function according to one of the instructions in the following two subsections.

To Copy Distribution Software To Another Diskette

In either of these instructions, the *source* diskette will be the diskette containing your Distribution Software; the *target* diskette will be the diskette you will copy to. Before you start, be sure to have one (or more,as needed) formatted diskettes on hand to serve as target diskettes.

First, place your Distribution Software diskette in your PC's A Drive and log to that drive by typing **A:** . Then, use one of the following instructions to copy the diskette files.

- If your PC has just one diskette drive (Drive A), type **COPY *.* B:** (in a single-drive PC, Drive A also serves as Drive B) and follow the instructions on the screen.

  If you prefer to use the DOS *DISKCOPY* function, instead of *COPY* , you will type **DISKCOPY A: A:** and follow instructions on the screen. This alternative is faster, but requires access to DISKCOPY.COM, in your DOS files.

- If your PC has two diskette drives (Drive A and Drive B), type **COPY *.* B:** (the same as above) and follow the instructions on the screen.

  If you prefer to use the DOS *DISKCOPY* function, instead of *COPY* , you will type **DISKCOPY A: B:** and follow instructions on the screen. This alternative is faster, but requires access to DISKCOPY.COM, in your DOS files.

To Copy Distribution Software To The PC Hard Drive

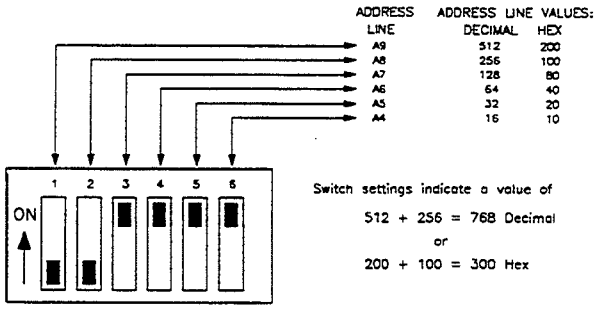
Before copying Distribution Software to a hard drive, make a directory on the hard drive to contain the files. While the directory name is your choice, the following instructions use *HRES* .

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

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

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

2.3 HARDWARE INSTALLATION



DAS-HRES uses 16 consecutive address locations in I/O space. Some I/O addresses will already be used by internal I/O and your other peripheral cards, so to avoid conflict with these devices, DAS-HRES's I/O address can be set by the Base Address DIP switch (see diagram at right) to be on a 16-bit boundary anywhere in the IBM PC decoded I/O space. The PC and PC/XT's expansion I/O address space extends from decimal 512-1023 (Hex 200-3FF) which is much larger than is ever likely to be fully occupied. The PC AT's I/O address space is larger and extends from 256-1023 (Hex 100-3FF), many XT compatibles also follow this expanded I/O space capability. Such a large space also allows use of more than one DAS-HRES in a single computer. The reserved I/O addresses for standard IBM devices are as follows:

HEX RANGE	USAGE	HEX RANGE	USAGE
000 to 1FF	Internal System	387 to 37F	LPT1:
200 to 20F	Game	380 to 38C	SDLC comm.
210 to 217	Expansion unit	380 to 389	Binary comm. 2
220 to 24F	Reserved	3A0 to 3A9	Binary comm. 1
278 to 27F	Reserved	3B0 to 3BF	Mono dsp/LPT1:
2F0 to 2F7	LPT2:	3C0 to 3CF	Reserved
2F8 to 2FF	COM2:	3D0 to 3DF	Color graphics
300 to 31F	Prototype card	3E0 to 3E7	Reserved
320 to 32F	Hard disk	3F0 to 3F7	Floppy disk
		3F8 to 3FF	COM1:

This covers the standard IBM I/O options (most compatibles are identical), but if you have other I/O peripherals such as special hard disk drives, special graphics boards, prototype cards, etc., they may be making use of I/O addresses not listed in the table above. Memory addressing is separate from I/O addressing so there is no possible conflict with any add-on memory that may be in your computer.

Usually, a good choices is to put the DAS-HRES at Base Address Hex &H300 or &H310 (Decimal 768 or 784). (Note if you are using an IBM prototype board, it uses the Hex 300-31F address space and would conflict; &H330 or &H340 would be a good alternative in this case). As an aid to setting the Base Address DIP switch, a graphical switch position display program, CALHRES.EXE, can be run from the DOS prompt. This program also will generate a HRES.CFG file that holds the current DAS-HRES I/O address for the use of other programs. To run the installation aid program, log to the HRES files and type **CALHRES** .

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

When you get the *Enter DAS-HRES base address?* prompt, type in your choice in decimal or &H— format and press return. The program will round your address to the nearest 16-bit boundary, and draw a picture of the correct positions of the toggles on the base address DIP switch. See the Base Address Switch diagram for an explanation of base address switch settings.

CALHRES.EXE performs one further function: it allows you to generate a file named HRES.CFG, which contains the base I/O address that you have selected. If your application programs read this file instead of declaring the address in each program (list EX0.BAS to see how it's done), then should you wish to change the board address in the future, all you have to do is alter the HRES.CFG file instead of altering dozens of application programs. The DAS-HRES disk comes with a HRES.CFG file ready loaded with decimal 768 (Hex 300) which will be overwritten if you choose to generate another address file when you run CALHRES.

The next step is to remove the DAS-HRES board from its protective electrostatic packaging and set the BAse Address Switch located just to the upper-left of the gold-edge connector. It is a good precaution to discharge any electrostatic charge you may have accumulated by touching the metal frame of your computer before inserting the board in the computer. This is also a convenient point to set some of the other switches and jumpers on the DAS-HRES board if you wish (it is not essential at this stage). Their functions and markings are as follows:

1. **A/D..-** This slide switch controls the input range. In the UNI (Unipolar) position, inputs can be positive only (ranges are from zero to some positive full-scale voltage). In the BIP (Bipolar) position, inputs can range from equal negative to positive full-scale limits.
2. **DMA -** This slide switch selects the operating DMA level. If you have floppy disk drives only on your computer (plain IBM P.C.) set it to level 3. If you have a hard disk equipped computer (PC XT or equivalent), Level 1 is preferable, although operation of some hard disk equipped machines may also be possible on Level 3. For a detailed explanation of DMA (Direct Memory Access), see Appendix E.
4. **JUMPER BLOCKS -** There is one jumper block on the DAS-HRES. The jumper marked TIMER selects the timer input frequency 10MHz, 8HHz, or 1MHz. Generally 10MHz is more useful and will give a finer time interval resolution at higher conversion rates. For further infomation regarding reasons to use the 8MHz or 1MHz setting refer to Applications appendix.

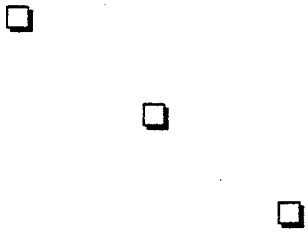
None of the above switch or jumper settings apart from the Base Address is critical to operation of the board. If at this time you are not sure about your requirements, don't waste time worrying about it, install the board and make changes later, no harm can result.

To install the board, TURN OFF THE POWER on your computer, disconnect or turn off any already installed peripherals and remove the case (See the *Guide to Operations Manual* for your computer if you are not already expert at this maneuver). Remove a vacant back plate by undoing the screw at the top; plug the DAS-HRES in and secure the back plate. DAS-HRES will fit in any of the regular full depth slots of the IBM PC or XT or Portable computer. On the PC/AT, DAS-HRES can be plugged into any socket but it will not make use of the extended AT bus interface connector. Installation is now complete. You may plug any of the DAS-HRES accessories or your own cable into the 50-pin connector at the rear.

Remember, TURN OFF THE POWER whenever installing or removing any peripheral board including the DAS-HRES. Never try to install or remove any peripheral board with the power on as it can cause costly damage to the electronics of your computer and/or the DAS-HRES board.

If for any reason you later remove the DAS-HRES board, the manufacturer recommends that you retain the special electrostatic-shield packaging and use it for storage. ■■■

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 3

REGISTER FUNCTIONS

3.1 REGISTER STRUCTURE & FORMAT

The DAS-HRES uses 16 locations in I/O address space as follows:

LOCATION		FUNCTION	TYPE
Base address	+0	A/D Low byte Start A/D	R W
	+1	A/D High byte Trigger A/D calibration	R W
	+2	Mux scan control	R/W
	+3	Digital I/O Out byte Digital I/O In byte	W R
	+4	D/A 0 Low byte	W
	+5	D/A 0 Hi byte, and load trigger	W
	+6	D/A 1 Low byte	W
	+7	D/A 1 Hi byte, and load trigger	W
	+8	DAS-HRES Status	R/W
	+9	DAS-HRES Control	R/W
	+10	Counter enable	R/W
	+11	Programmable gain, and calibration polarity	R/W
	+12	Counter 0	R/W
	+13	Counter 1	R/W
	+14	Counter 2	R/W
	+15	Counter Control	W

3.2 A/D REGISTERS (BASE ADDRESS +0 & +1)

The ADC is read and controlled from the registers at Base Address +0 and +1. Data is in straight binary format. A Read from Base Address +0 yields the least significant byte of the ADC output. Note that the Distribution Software automatically converts data to 2's Compliment format for Bipolar mode.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

BASE ADDRESS +0

BIT:	D7	D6	D5	D4	D3	D2	D1	D0
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
	MSB							LSB

BASE ADDRESS +1

BIT:	D7	D6	D5	D4	D3	D2	D1	D0
	ADC15	ADC14	ADC13	ADC12	ADC11	ADC10	ADC9	ADC8
	MSB							LSB

NOTE: A write to Base Address +0 will always trigger a conversion.

3.3 MUX SCAN REGISTER (BASE ADDRESS +2)

Base Address +2 is the MUX Scan Register, which is Read/Write and controls the channel scan limits of the A/D Multiplexer. The high nybble provides the scan-ending address; the low nybble provides the scan-starting address. Data is true binary with a format as follows:

BIT:	D7	D6	D5	D4	D3	D2	D1	D0
	AH3	AH2	AH1	AH0	AL3	AL2	AL1	AL0
	End Of Scan Address				Start Of Scan Address			

NOTE: Read the contents of the Status Register (Base Address +8) to verify loading of the MUX Scan Counter. The four LSBs loaded into the MUX Scan Register should be the same as the four LSBs read back from the Status Register, as long as no conversions have occurred between the MUS Scan load and the status read. If the Status Register readback does not return the correct value, repeat the MUX Scan load and Status Register readback until a correct value is read back. No more than two or three passes should be required.

Between 0.5 and 1 microsecond after the A/D starts a conversion, while the Sample/Hold is holding the previous channel, the multiplexer address is incremented by the board hardware so that it is ready for the next conversion. On finishing the conversion performed with the end of scan address, the mux channel counter is automatically reloaded with the scan start address, ready for the next scan. If you write data to the mux. scan register the mux channel 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.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The DAS-HRES has only eight A/D channels. However the scan multiplexer can select Channels 0-15. Selecting Channels 8 through 15 measures other variables on the DAS-HRES. The entire channel selection table is shown in the following table:

CHANNEL NUMBER	FUNCTION
0	Input Channel 0
1	Input Channel 1
2	Input Channel 2
3	Input Channel 3
4	Input Channel 4
5	Input Channel 5
6	Input Channel 6
7	Input Channel 7
8	Gain of 1 calibration channel
9	Gain of 2 calibration channel
10	Gain of 4 calibration channel
11	Gain of 8 calibration channel
12	Zero volts
13	DAC 0 out
14	DAC 1 out
15	Temperature

3.4 DIGITAL I/O (BASE ADDRESS +3)

Digital I/O consists of two hardware-configured ports, an 8-bit output port (Out 0-7) and an 8-bit input port (IN 0-7). These ports share the same I/O address (Base Address +3) but are independent (data written to the output port is not read back by the input port unless the output lines are externally connected to the input lines). In addition, some of the input port lines do double duty. IP0 is also read by some modes to determine when to begin taking data. Depending on the application, this secondary function of IP0 may or may not be used, but regardless of whether they are used, the state of the inputs can always be determined by reading Base Address +3. Formats of the Output and Input Ports are as follows:

Output Port:

BIT:	D7	D6	D5	D4	D3	D2	D1	D0
	OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Input Port:

BIT:

D7	D6	D5	D4	D3	D2	D1	D0
IN7	IN6	IN5	IN4	IN3	IN2	IN1	IN0

3.5 D/A OUTPUT REGISTERS (BASE ADDRESS +4 TO +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

Base Address +4 (Low Byte):

BIT:

D7	D6	D5	D4	D3	D2	D1	D0
DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0

LSB

Base Address +5 (High Byte):

BIT:

D7	D6	D5	D4	D3	D2	D1	D0
DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8

MSB

D/A #1

Base Address +6 (Low Byte):

BIT:

D7	D6	D5	D4	D3	D2	D1	D0
DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0

LSB

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Base Address +7 (High Byte):

BIT:

D7	D6	D5	D4	D3	D2	D1	D0
DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8

MSB

3.6 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-HRES. Writing to the Status Register address clears the DAS-HRES interrupt request and provides the means of acknowledging the DAS-HRES interrupt and re-enabling it. Theformat is

Base Address +8:

BIT:

D7	D6	D5	D4	D3	D2	D1	D0
EOT	U/B	EOC	INT	MA3	MA2	MA1	MA0

EOT - This bit is used to determine when valid data is available after initiation of a conversion. This bit indicates the status of the converter's internal sampling system: high when the converter is tracking the input signal; low when the input has been acquired. Timing of the EOT signal is as follows:

1. EOT goes high between 50ns and ius from the time a conversion is initiated. It will remain high until the conversion is complete and the signal has been acquired for next conversion. This time is 16.25us for the conversion plus 3.75us for the tracking acquisition and 1us for synchronization uncertainty--for a total of between 20 and 21us. During this time, no conversions should be started. Note that the synchronization uncertainty has nothing to do with the aperture time or jitter specifications.

2. When EOT returns low, it indicates two conditions. One is that data in the ADC Registers contains valid data; the other is that the converter is ready to accept another Convert command.

U/B - When at logic one, U/B indicates that the ADC is operating in Unipolar mode. When at logic zero, U/B indicates that the ADC is in Bipolar mode. This condition is selected by a switch on board at time of installation.

EOC - This bit pulses low for 1us when a conversion is complete. It is included for diagnostic purposes only. Use the EOT signal for ADC status.

INT - logic one if an interrupt has been generated. This bit must be cleared by writing to the status register. The bit will be generated according to the controls dictated in the control register.

MUXA3-0 - These bits indicate which input channel will be selected next . The multiplexer changes channels between 500-1000ns after aconversion is started.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.7 DAS-HRES CONTROL REGISTER (BASE ADDRESS +9)

The DAS-HRES Control Register is a read/write register that controls the operating modes of the board. This register is automatically 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:

Base Address +9:

BIT:

D7	D6	D5	D4	D3	D2	D1	D0
INTE	L2	L1	L0	ISRC	DMA	S1	S0

INTE - This bit enables or disables DAS-HRES 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 - Selects the desired Interrupt Level for processing interrupts, as follows:

L2	L1	L0	INTERRUPT LEVEL
0	0	0	None - disabled
0	0	1	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

Programmers should avoid using an Interrupt Level that is inuse by another I/O device e.g. Level 6 is used by the floppy-disk drive(s).

ISRC & DMA - Control the source of interrupt generation according to thefollowing table:

ISRC	DMA	FUNCTION
0	0	Interrupts are generated at the end of each conversion.
0	1	Interrupts will generated by the terminal count of the system board DMA controller. This signifies that the data block transfer is complete.
1	X	Interrupts will be generated by the Counter/Timer (8254 Counter 2 output).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

S1-0 - These bits control the source of start (or trigger) pulses for the A/D, as follows:

S1	S0	ACTION
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 & S1 bits, an A/D conversion canalways be initiated by a write to the A/D register at Base Address +0.

Note that all triggers except TRIG0 will be disabled if theTRIG0 input is high, S1 is high, and S0 is low.

3.8 TIMER-COUNTER ENABLE REGISTER (BASE ADDRESS +10)

This is an 8-bit read/write register that controls operation of the Timer/Counter (see Figure 5-1). Only the bottom two bits are used; the upper six bits have no affect on the hardware and can be used by the programmer for any other purpose. Note that the Timer-Counter Enable register is not cleared by power-on Reset; it must be explicitly loaded and initialized by the user. DAS-HRES uses an Intel 8254-2 Timer Counter for periodically triggering the A/D. Counters 1 &2 are cascaded and driven by a 1MHz or 10MHz clock so that periods ranging from a few microseconds to in excess of 1 hour may be programmed. TRIG0 and C0 of the Timer/Counter Enable Register are AND'ed, so that C0 becomes an enable for TRIG0 and counting will proceed when both are high. This allows set up of the counter and a DMA or interrupt process and subsequent enabling of operations either by software or a signal on TRIG0.

The function of C1 is somewhat different. When C1 and CTR 0 IN are both high, the input of Counter 0 is internally connected to a 500KHz 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 500KHz 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 (COUNTER 2 OUTPUT, Counter 0 can be used to determine frequency).

Base Address +10:

BIT:	D7	D6	D5	D4	D3	D2	D1	D0
	X	X	X	X	X	X	C1	C0

X = Don't care.

3.9 PROGRAMMABLE GAIN REGISTER (BASE ADDRESS +11)

This is 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 affect on the hardware and can be used by the

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

programmer for any other purpose.

NOTE: 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:

Base Address +11:

BIT:

D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	X	X	X	X

X = Don't care.

The G1 and G0 gain code bits select the gains as follows:

G1	G0	GAIN
0	0	1
0	1	2
1	0	4
1	1	8

3.10 8254 PROGRAMMABLE INTERVAL TIMER/COUNTER (BASE ADDRESS +12 TO +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; a brief description appears in Chapter 5.

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

3.11 DAS-HRES AUTOCALIBRATION SOFTWARE

Introduction

DAS-HRES uses high-quality hardware, combined with a software-correction scheme to obtain true 16-bit A/D accuracy. However, all autocorrection is performed automatically by the DAS-HRES software. Since the autocorrection is transparent to the user, the following section need not be read by most users. However, for the sake of completeness and to help software developers who are not using the DAS-HRES driver software, the following section is included.

To achieve full 16-bit accuracy and stability, the DAS-HRES provides a means of automatically

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

calibrating out gain and offset errors. This is achieved with a combination of on board hardware resources and calibration software.

The software correction/calibration system consists of three basic parts: initial calibration, recalibration, and the runtime software (all run-time corrections/calibrations are handled in the DAS-HRES software driver).

The first step in calibration is handled by the user installation software and requires the use of a precision voltmeter (at least 5 1/2 digits and preferably more). A Keithley Instruments Model 196 6 1/2 digit voltmeter is a good choice for calibration. The procedure is described in more detail in Chapter 9. The board calibration procedure creates a HRES.CFG file containing the board base address, a calibration factor for each range, the values of the on-board calibration voltages, and the board temperature in degrees C.

In addition to the initial calibration, the "RECAL" program included on the DAS-HRES utility disk provides a means of correcting for any drift of the gain setting components. The gain setting components will drift more than the internal reference signals. The RECAL program is a stand-alone program that reads the voltages stored in the HRES.CFG file and recalculates the calibration constants. This program allows an interim of recalibrating the DAS-HRES without using a voltmeter.

Once the HRES.CFG file has been created user, applications must read the calibration constants from it and pass them to the runtime routines. In the DAS-HRES software driver this is done by passing the calibration constants in the MODE 0 call. Three calibration options are available: no calibration, offset correction only, or offset and gain correction only. Only the third mode allows the DAS-HRES to meet its overall accuracy specifications.

The Calibration Factors

DAS-HRES software must make gain and offset corrections of the raw data received from the HRES hardware. This correction is described by the following equation:

Corrected Code = CF * (raw ADC data - offset code) + raw ADC data

Where CF is a calibration factor retrieved from the HRES.CFG file, and"offset code" is a number that must be generated by the run-timesoftware. There are 8 calibration factors stored in the HRES.CFG file. One for each Gain, in both Unipolar and Bipolar modes.

In Foreground mode operations, the above calculation is made real-time. This means the above calculation must be made each time a sample from the A/D is taken. In background modes (DMA or Interrupt Driven) there is not time to perform the above calculation real-time. In these cases, the data stored in memory must be corrected after the actual data acquisition run. The HRES software driver performs all foreground calculations real-time. The driver's Mode 9 is used tomove DMA or Interrupt driven data from memory into an arrayautomatically corrects the data as a post-acquisition function.

CF

The calibration factors (CF) represent the deviation from ideal gain in percent multiplied by a scale factor. The deviation from ideal is calculated and multiplied by 2^22 (4,194,304),and rounded to the nearest integer. Each unit of the calibration constant represents about 238 parts per billion in deviation from ideal. For example if again deviated from ideal by +0.1%, then

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Cal Factor = (0.001 * (4,194,304)
= 4194.3

which would be rounded to 4194.

HRES.CFG File

The format of the HRES.cfg file is as follows:

ENTRY	ITEM	DATA TYPE
1	Base Address	Signed 16-bit integer.
2	Gain of 1 Calibration Factor (Bipolar).	Signed 16-bit integer.
3	Gain of 2 Calibration Factor (Bipolar).	Signed 16-bit integer.
4	Gain of 4 Calibration Factor (Bipolar).	Signed 16-bit integer.
5	Gain of 8 Calibration Factor (Bipolar).	Signed 16-bit integer.
6	Gain of 1 Calibration Factor (Unipolar).	Signed 16-bit integer.
7	Gain of 2 Calibration Factor (Unipolar).	Signed 16-bit integer.
8	Gain of 4 Calibration Factor (Unipolar).	Signed 16-bit integer.
9	Gain of 8 Calibration Factor (Unipolar).	Signed 16-bit integer.
10	On-board calibration voltage for gain of 1 range.	Double precision floating point number.
11	On-board calibration voltage for gain of 2 range.	Double precision floating point number.
12	On board calibration voltage for gain of 4 range.	Double precision floating point number.
13	On-board calibration voltage for gain of 8 range.	Double precision floating point number.
14	Temperature of board at time of calibration in degrees C.	Signed 16-bit integer.
15	Date of calibration.	ASCII
16	Card serial number.	16 bit integer.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Below is a listing of a typical HRES.CFG. Note that the file is allASCII format as stored. The data type listed above are the types that are used once they are brought into a program.

```
 768
-285
-262
-123
-466
-315
-267
-117
-481
7.49783275993439
3.74953740004481
1.87480976242457
0.93733611934756
37
7/26/1989
936
```

Offset Code

The offset code is the value of averaging 5000 conversions from channel 12 (for the gain selected). This code does not need to be recalculated for every conversion, in general is only done during card initialization.

■ ■ ■

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 4

THE CALLABLE MODES

4.1 LIST OF MODES

The following subsections give detail information and examples of the use of the CALL routine in all 22 modes. The modes are selected by the MD% parameter in the CALL as follows:

MODE	FUNCTION
0	Initialize card,input HRES base address,interrupt level, DMA level,and calibration level.
1	Set multiplexor low and high scan limits.
2	Reads the current channel setting and scan limit settings.
3	This MODE performs a single A/D conversion. Data from the conversion is returned, the multiplexor is incremented, and the next channel gain is selected from the gain table set up in MODE 21. This MODE is slow and runs in the foreground. If calibration MODE 1 is selected in MODE 0 then data will be offset corrected. If calibration MODE 2 is selected in MODE 0 then both gain and offset will be corrected.
4	This routine will perform N A/D conversions after receipt of an external trigger. Scan rate can be set by the programmable timer (set up by MODE 17), or by the external trigger. This MODE is faster than MODE 3 but still runs in the foreground. MODE 21 sets gain table. If calibration MODE 1 is selected in MODE 0 then data will be offset corrected. If calibration MODE 2 is selected in MODE 0 then both gain and offset will be corrected.
5	This MODE is similar to MODE 4 except that data transfer is driven by interrupts. Scan rates are set by the programmable timer or the external trigger. Speed is medium and runs in the background. Data is transferred by an interrupt service routine. MODE 21 sets gain table. MODE 9 is used to transfer the data from memory into a BASIC Array (autocalibration is performed in MODE 9).
6	This MODE is the fastest MODE of data transfer. An array of integers is stored directly into memory by a DMA transfer. Since is no interrupt service routine to select the next channel's gain, all channels are at the same gain. MODE 9 is used to transfer data from memory into a BASIC array. Note that MODE 6 puts data into memory without performing any autocalibration. For calibrated results use MODE 9, which not only takes the data from memory, but also performs that autocalibration operation.
7	Disable DMA and interrupt operations started in MODEs 5,6,18, or 20.
8	Reports status of DMA and interrupt operations started by MODEs 5,6,18 or 20.
9	Block move of data acquired in MODEs 5, 6 and 20. If calibration MODE 1 is selected in MODE 0 then data will be offset corrected. If calibration MODE 2 is selected in MODE 0 then both gain and offset will be corrected.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

10	Set Counter 0 configuration. This is the counter whose output is available at the output connector.
11	Load Counter 0 data.
12	Read counter 0.
13	Output to digital outputs OP0-OP7.
14	Read digital inputs IP0-IP7.
15	Output data to a single DAC channel.
16	Outputs data to both DAC channels.
17	Set programmable timer rate.
18	DAC waveform output & ADC input. (Note that no data correction is performed by MODE 18.
19	Analog trigger function.
20	A/D block scan on interrupt. This MODE will do a complete scan of channels using gains set by MODE 21. Runs in background at medium speed. Use MODE 9 to transfer data from memory into a BASIC array. Note that MODE 20 puts data into memory without performing any autocalibration. For calibrated results use MODE 9, which not only takes the data from memory, but also performs that autocalibration operation.
21	Set channel gains. Use this MODE to set up gain table prior to calling MODEs 3,4,5 and 20.
22	Output a square wave frequency from counter 0 out. Frequency from 1.5Hz to 25KHz.
23	Allocate memory Segment.
24	Deallocate Memory Segment.

4.2 DESCRIPTIONS OF MODES

MODE 0: Initialize

Before using any other MODE of the CALL routine, you must provide the I/O location, or Base Address, of the DAS-HRES board and the interrupt and DMA levels that you intend to use. If you fail to provide this information before accessing any other MODE of the CALL, error flag (FLAG% = 1, driver not initialized) will be returned and none of the other MODEs will execute. Calling MODE 0 need only be done once in the initialization section of your program usually straight after loading the HRES.BIN routine. For examples of loading HRES.BIN and initializing, run and list EXH0.BAS.

Entry:

D%(0) = DAS-HRES base address

D%(1) = Interrupt level (2 through 7)

D%(2) = DMA level (1 or 3)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

D%(3) = Calibration level
- 0 = No calibration
- 1 = Offset correction only (no calibration factors need be passed for this MODE.)
- 2 = Gain and offset calibration is to be performed (Calibration factors needed for this level of calibration).

D%(4) = Gain of one calibration factor (from HRES.CFG file)

D%(5) = Gain of two calibration factor (from HRES.CFG file)

D%(6) = Gain of four calibration factor (from HRES.CFG file)

D%(7) = Gain of eight calibration factor (from HRES.CFG file)

Note that Bipolar and Unipolar operations have different calibration factors. Be sure to use the calibration factors which match the Unipolar/Bipolar switch on the board.

then:

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

Note that specifying the first@< element of the array will pass all other required array parameters.

Return:

MD% = 0 (unchanged)

D%(0-15) (unchanged)

Error Codes:

FLAG% = 0 (No Error, OK).
= 2 (MODE number out of range, < 0 or > 22).
= 3 (Base Address out of range < 256 or > 1008).
= 4 (interrupt level < 2 or > 7).
= 5 (DMA level not 1 or 3).
= 22 (Board not present, I/O address wrong).
= 27 (Invalid Auto-Calibration Level).

Base Address

The standard IBM PC or PC/XT allows use of I/O addresses between 512 and 1008 (Hex 200 - 3F0); the PC/AT allows use of addresses between 256 and 1008 (Hex 100 - 3F0); all machines use I/O addresses below 256 (Hex 100) for internal devices. Hence the driver will flag any attempt to assign an I/O address in the range 0 - 255 (Hex 0 - FF) by returning Error Code #3 in the FLAG% variable.

Interrupt Level

Error #4 will occur if you have specified a non-valid interrupt level. The available levels on the PC expansion bus correspond to 2 thru 7. Certain of these levels may be in use by other peripheral devices (especially level 6 used by floppy disk drive). A list of the standard IBM interrupt assignments is:

Level 2 - Reserved (but not used) by Color Graphics adapter
Level 3 - Serial I/O - used if COM2: installed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Level 4 - Serial I/O - used if COM1: 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.

If you do not have a particular device installed, it is safe to assume that level is available for use by DAS-HRES. The lower the level number, the higher the interrupt priority. **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, D%(1) = 2).

DMA Level

Error #5 is obtained if you specify a DMA level other than 1 or 3. There are 4 DMA levels available on the IBM PC (see Appendix E. Level 0, the highest priority is internally used for dynamic memory refresh and is not accessible to the user. The other levels 1 thru 3 are available on the expansion bus, but Level 2 is always used by the floppy disk drive(s) and cannot be shared. In hard disk (XT) computers, Level 3 is assigned to the hard disk controller, depending on the design of the disk controller and fixed disk BIOS, it may in some cases be available (certain Adaptec controllers do not use DMA). In any case Level 1 is usually available so if Level 3 is used by the hard disk set D%(2) = 1. In floppy disk only machines D%(2) can be 1 or 3. Be sure to set the DMA select slide switch on the DAS-HRES board to the level selected. Also, note that DMA requests and the 8237 controller are not enabled until you enter MODE 6. If you are not using MODE 6, it is irrelevant whether you set D%(2) = 1 or 3 or what position the DMA slide switch is in.

Calibration Level

Three levels of calibration are available on the DAS-HRES. These are: No Calibration, Offset correction only, and Gain and Offset correction. Operation in the No Calibration and Offset Only Calibration MODEs will not yield fully calibrated results. For this reason Gain and Offset correction (D%(3) = 2) is recommended. The entire data correction process used on the DAS-HRES is automatically performed, and fully calibrated data available when Gain and Offset correction is selected. For further details on the autocalibrate process please refer to APPENDIX G.

Once an automatic calibration MODE has been selected by MODE 0 , other MODEs will correct the raw data according to the calibration level selected. MODEs that are interrupt or DMA driven collect raw data, but calling MODE 9 (Transfer from memory to array) after data collection will correct the data. All other ADC data collection MODEs correct data as it is acquired.

Gain Factors

One gain factor must be loaded into the MODE 0 call for each gain range. These factors are generated during the DAS-HRES calibration procedure, and are stored in the HRES.CFG file. The format of the HRES.CFG file is shown below.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENTRY	ITEM	DATA TYPE
1	Base Address	Signed 16-bit integer
2	Gain of 1 Cal. Factor (Bipolar).	Signed 16-bit integer
3	Gain of 2 Cal. Factor (Bipolar).	Signed 16-bit integer
4	Gain of 4 Cal. Factor (Bipolar).	Signed 16-bit integer
5	Gain of 8 Cal. Factor (Bipolar).	Signed 16-bit integer
6	Gain of 1 Cal. Factor (Unipolar).	Signed 16-bit integer
7	Gain of 2 Cal. Factor (Unipolar).	Signed 16-bit integer
8	Gain of 4 Cal. Factor (Unipolar).	Signed 16-bit integer
9	Gain of 8 Cal. Factor (Unipolar).	Signed 16-bit integer
10	On-board calibration voltage for gain of 1 range.	Double-precision floating-point number.
11	On board calibration voltage for gain of 2 range.	Double-precision floating point number.
12	On board calibration voltage for gain of 4 range.	Double-precision floating-point number.
13	On board calibration voltage for gain of 8 range.	Double-precision floating-point number.
14	Temperature of board at time of calibration in degrees C.	Signed 16-bit integer.
15	Date of calibration.	ASCII
16	Card serial number.	Signed 16-bit integer.

MODE 0 performs several other initializing functions. Default Channel Scan Limits of 0 and 7 are set in MODE 0. All channel gains are set to unity. The DAS-HRES control and timer counter enable registers are cleared disabling all interrupt, DMA, and external trigger functions, and the programmable interval timer (Counters 1 and 2) are configured to the rate generator MODE and set to produce a 1KHz (10KHz with 10MHz clock) pulse rate (this can be changed to other frequencies by MODE 17). MODE 0 also performs a simple read/write test as a check on the function and presence of the DAS-HRES hardware. If you obtain Error 22, it indicates either a hardware fault in the DAS-HRES or a discrepancy between the Base Address specified in D%(0) and the actual switch setting on the board.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MODE 1: Set Multiplexer Scan Limits

MODE 1 is used to set the scan limits of the multiplexer to other than the default limits provided by initializing MODE 0. This may be done prior to performing conversions in MODEs 3, 4, 5, 6, 18, or 20. Two limits are passed: D%(0) contains the lower limit and D%(1) the higher limit. If the default limits of MODE 0 are OK for you, it is not necessary to enter MODE 1.

To illustrate the action of MODE 1 assume we set D%(0) = 3 and D%(1) = 6. The first conversion (commanded by MODEs 3, 4, 5, 6, 18, or 20) would be performed on Channel 3, data returned, and the channel incremented to 4. The next conversion would be 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 reload to the start of scan limit to repeat the sequence. Scanning of channels would always be stepped between and including Channels 3 and 6 as follows:

3 - 4 - 5 - 6 - 3 - 4 - 5 - 6 - 3 - etc.

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

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

The board contains eight analog input channels. However, selecting readings from Channels 8 through 15 is allowed. The measurement corresponding to each of these channels is shown in the following table:

CHANNEL NUMBER	FUNCTION
0	Input channel 0
1	Input channel 1
2	Input channel 2
3	Input channel 3
4	Input channel 4
5	Input channel 5
6	Input channel 6
7	Input channel 7
8	Gain of 1 calibration channel
9	Gain of 2 calibration channel
10	Gain of 4 calibration channel
11	Gain of 8 calibration channel
12	Zero volts (shorted input)
13	DAC 0 out
14	DAC 1 out
15	Temperature

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 D%(0) = 1 and D%(1) = 1 would perform continuous conversions on channel 1.

Note that after exit from MODE 1, the starting channel will always be D%(0), the lower limit.

Control and incrementing of the multiplexer channel address is performed by high speed hardware on the DAS-HRES board and is independent of the processor. Approximately 1 microsecond after the A/D has been triggered and the sample/hold is holding the data which is being converted by the A/D, the multiplexer is advanced to the next channel. This allows the programmable gain amplifier

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

to settle before the sample/hold returns to the sample state at the end of the 20 microsecond A/D conversion. This pipelining technique helps to improve the overall throughput of the system.]

Entries:

MD%	= 1 (MODE number)
D%(0)	= 0 thru 7 (lower scan limit)
D%(1)	= 0 thru 7 (upper scan limit)
D%(2-15)	(value does not matter)
FLAG%	= X (value does not matter)

then

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

Returns:

MD%	= 1 (unchanged)
D%(0-15)	(unchanged)]

Error Codes:

FLAG%	= 0 (No Error, OK). = 1 (driver not initialized). = 2 (MODE number out of range, < 0 or > 22). = 6 (scan limits out of range).

An example of the use of MODE 1 will be found in EXH3.BAS as well as several of the other examples using the A/D converter.

MODE 2: Read Current Multiplexer Address & Scan Limits

MODE 2 lets you determine the current multiplexer channel setting and scan limits and also returns the current gain setting.

Entries:

MD%	= 2 (MODE)
D%(0-15)	(value does not matter)
FLAG%	= X (value does not matter)

then

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

Returns:

MD%	= 2 (unchanged)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

D%(0) = Channel address for next conversion (0 thru 7)

D%(1) = Lower scan limit (0 thru 7)

D%(2) = Upper scan limit (0 thru 7)

D%(3) = Returns current gain setting as follows:

D%(3)	GAIN
0	1
1	2
2	4
3	8

D%(4-15) (unchanged)

Error Codes:

FLAG% = 0 (No Error, OK).
= 1 (driver not initialized).
= 2 (MODE number out of range, < 0 or > 22).

If MODE 2 is run during the 20 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 and gain code returned always correspond to the next channel on which a conversion will take place.

An example of the use of MODE 2 will be found in EXH3.BAS.

MODE 3: Do One A/D Conversion & Increment MUX

MODE 3 performs one A/D conversion by software command. The mux is automatically incremented after the conversion by the hardware. Data is returned as follows:

D%(0) - A/D data (0 - 65535 or -32768 to +32767)

D%(1) - Channel number

Data is transferred to D%(0) in 2's Compliment form (standard integer). If the Unipolar/Bipolar switch is in the UNIPOLAR position data returned is in the range of 0 - 65535 bits and is transferred directly. If the switch is in the BIPOLAR position, then 32768 is subtracted giving a data range of -32768 to +32767 bits. Hence, zero volts always corresponds to zero bits. This saves further processing of the data after exit from the CALL.

D%(1) contains the channel from which the data is derived. This information can be used or ignored as required.

The A/D will perform conversions on channels in accordance with the scan limit conditions set in MODEs 1. If MODE 1 has not been entered prior to MODE 3, conversions will cycle between Channel 0 and Channel 15. The gain of each channel will correspond to the set up sequence specified in MODE 21. If MODE 21 has not be entered before running MODE 3, all gains will default to a gain of 1. D%(2) returns the gain of the converted channel.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Entries:

MD% = 3 (MODE number).

D%(0-15) = X (value does not matter)

FLAG% = X (value does not matter)

then

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

Returns:

MD% = 3 (unchanged)

D%(0) = A/D data (0 to 65535 if Unipolar, -32768 to +32767 if Bipolar)

D%(1) = Channel number of data (0 - 7)

D%(2) = Returns channel gain as follows:

D%(2)	GAIN
0	1
1	2
2	4
3	8

D%(3-15) (unchanged)

Error Codes:

FLAG% = 0 (No Error, OK).
= 1 (driver not initialized).
= 2 (MODE number out of range, < 0 or > 22).
= 9 (No EOC from A/D, time out indicates hardware failure).

An example of the use of MODE 3 will be found in EXH3.BAS.

Note that MODE 3 is the only MODE that allows you to perform an A/D conversion initiated by the program (software command). For this reason, it is fairly slow since it is limited by the program execution speed, which for interpreted BASIC is slow (several milliseconds per line of code). A tight loop such as the one below will perform slightly under 200 conversions/sec. on a standard IBM PC (4.77MHz clock):

```
xxx10   MD% = 3
xxx20   FOR I% = 1 TO 10000
xxx30   CALL HRES (MD%, D%(0), FLAG%)
xxx40   ARRAY%(I%) = D%(0)
xxx50   NEXT I%
          etc.
```

If this is compiled using the BASIC compiler, a conversion rate of about 4000 samples/sec. will be obtained.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MODE 4: Do N A/D Conversions Direct To Array

MODE 4 performs N A/D conversions and transfers data directly into a BASIC integer array. Since the CPU is performing the A/D polling and data transfers as a "foreground" operation, exit from the CALL will *not* occur until all conversions have been completed. To provide an escape route, hitting any key on the keyboard while data is being gathered in MODE 4 will abandon further conversions and produce an immediate return to your BASIC program. If you do not want to wait for data to be collected, both MODEs 5 or 6 can be used to gather the data as a "background" operation so that your program is able to process data and collect it at the same time.

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 15. The gain of each channel will be set according to the predefined sequence set up by MODE 21. If MODE 21 has not been entered prior to running MODE 4, all gains will default to 1.

The A/D may be triggered from two sources, the Programmable Interval Timer or an external trigger pulse according to D%(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.

Entries (Original Options):

MD% = 4 (MODE number)

D%(0) = Number of conversions required (Word count). Range 1 to N where N-1 < = array dimension

D%(1) = VARPTR(ARRAY%(M)) - array pointer Conversions may be loaded starting at the M'th position in an array or at the start if M = 0.

D%(2) - Trigger sources (two are possible):

D%(2) = 0 : External trigger input. Conversions take place on positive transitions of the TRG0 input and continue until the word count is reached. **WARNING!** - exit from the routine cannot take place until pulses equal to the word count have been supplied.

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

D%(3-15) = X (value does not matter)

FLAG% = X (value does not matter)

then

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

Entries (New Options):

MD% = 4 (MODE number)

D%(0) = Number of conversions required (Word count). Range 1 to N where N-1 < = array dimension

D%(1) = -1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

D%(2) - Trigger sources (two are possible):

D%(2) = 0 : External trigger input. Conversions take place on positive transitions of the TRG0 input and continue until the word count is reached. **WARNING!** - exit from the routine cannot take place until pulses equal to the word count have been supplied.

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

D%(3) = VARPTR(ARRAY%(M)) - Offset of array pointer. Conversions may be loaded starting at the M'th position in an array or at the start if M = 0.

D%(4) = VARSEG(ARRAY%(M)) - Segment of array pointer conversions may be loaded.

D%(5-15) = X (value does not matter)

FLAG% = X (value does not matter)

then

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

Returns:

MD% = 4 (unchanged)

D%(0-15) (unchanged)

ARRAY%(M) = 1st. data word

ARRAY%(M+1) = 2nd. data word

ARRAY%(M+2) = 3rd. data word

. . . etc.

Error Codes:

FLAG% = 0 (No Error, OK).
= 1 (Driver not initialized).
= 2 (MODE number out of range, < 0 or > 22).
= 11 (Number of conversions 0 or negative).
= 19 (Trigger MODE not 0 or 1).
= 28 (Possible memory index overflow).

An example of the use of MODE 4 will be found in EXH4.BAS.

A number of precautions apply to the reliable use of MODE 4. First, you must dimension a receiving array that has at least as many elements as the number of conversions specified in D%(0). No checks are made by the CALL routine on whether you are performing more conversions than the array will hold. If you do, some of the data area of BASIC will be overwritten which may destroy descriptors and other variable data and cause strange effects as a result. Do not overrun the array limits.

Second, after assigning the pointer to the receiving array, do not introduce any new simple variables before entering the CALL. For example, the following is acceptable:

```
xxx10  DIM D%(15)                    'declare D%(*)
xxx20  MD% = 4                       'MODE #
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
xxx30   D%(0) = 1000                'number of conversions
xxx40   D%(1) = VARPTR(X%(0))       'pointer to array
xxx50   D%(2) = 1                   'trigger MODE
xxx60   CALL HRES (MD%, D%(0), FLAG%)
```

But this will cause a problem:

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

The problem arises because of the way BASIC stores array variables. They are located in memory above the data area for simple (non-array) variables which in turn is located above the program storage area. If you introduce a simple variable that has not been used before, BASIC makes room for this variable by re-locating all the array variables upwards in memory. If the pointer (using VARPTR) is assigned to the receiving array @B[before] a new variable is introduced and then the CALL is entered, the actual location of the array will have changed, and the CALL routine writes data to the old array location causing strange effects.]

ALSO NOTE that exit from MODE 4 cannot occur until IP0 has been taken high if using the programmable timer and a sufficient number of trigger pulses to perform the desired number of conversions has been supplied. If these conditions are not met, your computer may give the erroneous appearance of being hung up. To abandon further conversions, hit any key on the keyboard.

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.]

MODE 5: Do N A/D Conversions & Transfer To Memory On Interrupt

MODE 5 performs N A/D conversions triggered either externally or by the Programmable Interval Timer. At the end of each conversion, an interrupt is generated that invokes an interrupt handler routine installed by this MODE. This routine transfers the data from each conversion to a specified segment of memory and keeps track of the total number of conversions performed. When the number reaches N, as specified by D%(0), interrupts are disabled if in the nonrecycle MODE (D%(3)=0) or the process is repeated continuously to the same segment of memory if D%(3) specifies the re-cycle MODE. Note that once MODE 5 has enabled interrupts, conversions continue regardless of what other programs the user may be running (although they should not interfere either with the location of the HRES.BIN driver or the A/D data area). For this reason it is termed a background operation and in most respects is functionally similar to MODE 6 (DMA) although much slower. About 3000 samples/sec are possible in MODE 5.

To return data to a BASIC array when MODE 5 is operating or has finished operation, use MODE 9. (MODE 9 will automatically correct/calibrate the raw data stored in RAM, and place corrected data in the BASIC array.) To assess the progress of an operation initiated by MODE 5, use MODE 8. To abort an interrupt operation initiated by MODE 5, use MODE 7. For detail descriptions of the features of these other MODEs, see the following sections.

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 5, conversions will start on Channel 0 and

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

the upper scan limit will be Channel 15. The gain of each channel will be set according to the predefined sequence set up by MODE 21. If MODE 21 has not been entered prior to running MODE 4, all gains will default to 1. Note that one advantage of MODE 5 versus MODE 6 is that conversions may be made with channels set to different gains; MODE 6 requires all channel gains to be the same.

The A/D may be triggered from two sources, the Programmable Interval Timer or an external trigger pulse according to D%(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.

Entries (Original parameter options):

MD% = 5 (MODE number)

D%(0) = Number of conversions required (Word count). Range 1 to 32,767, or 0 for 32,768 max.

D%(1) = Segment of memory to receive data. Segments are on paragraph (16 bit) boundaries (for example, D%(1) = &H4000 would start loading A/D data at 256K). Be sure to choose an empty area of memory for a data buffer (always outside BASIC workspace).

D%(2) - Trigger source (Two are possible):

D%(2) = 0 : External trigger input. Conversions take place on positive transitions of the TRG0 input and continue until the word count is reached.

D%(2) = 1 : Programmable Interval Timer. In this case, IP0 should be held low until you want to start conversions. After IP0 goes high, IP0 will have no further effect.

D%(3) - Single cycle/Recycle operation:

D%(3) = 0 : One cycle. After completion of the number of conversions specified, interrupts are disabled, setting the operation status to zero.

D%(3) = 1 : Recycle. In this case data is continuously written to the same memory. D%(0) corresponds to the memory "buffer" length. The status of the operation is 1 = active until stopped by MODE 7.

D%(4-15) = X (value does not matter)

FLAG% = X (value does not matter)

then

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

Entries (New parameter options):

MD% = 5 (MODE number)

D%(0) = Number of conversions required (Word count). Range 1 to 32,767, or 0 for 32,768 max.

D%(1) = -1

D%(2) - Trigger source (Two are possible):

D%(2) = 0 : External trigger input. Conversions take place on positive transitions of the TRG0 input and continue until the word count is reached.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

D%(2) = 1 : Programmable Interval Timer. In this case, IP0 should be held low until you want to start conversions. After IP0 goes high, IP0 will have no further effect.

D%(3) - Single cycle/Recycle operation:

D%(3) = 0 : One cycle. After completion of the number of conversions specified, interrupts are disabled, setting the operation status to zero.

D%(3) = 1 : Recycle. In this case data is continuously written to the same memory. D%(0) corresponds to the memory "buffer" length. The status of the operation is 1 = active until stopped by MODE 7.

D%(4) = Offset of memory to receive data.

D%(5) = Segment of memory to receive data.

D%(6-15) = X (value does not matter)

FLAG% = X (value does not matter)

then

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

Returns:

MD% = 5 (unchanged)

D%(0-15) (unchanged)

Error Codes:

FLAG% = 0 (No Error; OK).
= 1 (Driver Not Initialized).
= 2 (MODE number out of range, < 0 or > 22).
= 11 (Number of conversions 0 or negative).
= 20 (Interrupt or DMA already active).

An example of the use of MODEs 1, 5, 7, 8, 9, and 17 will be found in EXH5.BAS.

Several details apply to the reliable use of MODE 5. First, MODE 5 will not let you reinstall the interrupt handler if the interrupt is still active. Error #20 will result. If you request the recycle MODE which generates continuous interrupts, then MODE 7 which disables interrupts must be run before you can successfully run MODE 5 again. If you are in nonrecycle MODE, then it must have reached the word count which automatically disables interrupts before MODE 5 can be run again. On completion of an interrupt operation, the selected level of the 8259 interrupt mask register is disabled and the tristate interrupt drivers of the DAS-HRES are placed in the high impedance state. This allows more than one hardware device, or multiple DAS-HRES's to use the same interrupt level as long as they do so sequentially. Note that any old interrupt vectors are not restored by the driver after DAS-HRES has finished using the interrupt.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Second, since the segment registers are not incremented by the handler, the maximum data area available is 64K (a page) for 32,767 conversions. Be sure that your data area is not in use by your program or altered by subsequent operations. Data may be retrieved using MODE 9 during or after the operation of MODE 5 and MODE 9 will not alter the memory. It is possible to rewrite the interrupt handler to use several segments of main or extended memory for data storage.

Third, conversion speeds up to 3000 samples/s can be reliably achieved with MODE 5. For higher speeds use MODE 6. MODE 5 is subject to disruptions from higher priority interrupts, notably the system timer, and this is the main limitation on throughput.

ALSO NOTE that exit from MODE 5 cannot occur until IP0 has been taken high if using the Programmable Interval Timer.

MODE 6: Do N A/D Conversions & Transfer To Memory Via DMA

MODE 6 is the fastest A/D conversion MODE and performs N A/D conversions triggered either externally or by the Programmable Interval Timer. At the end of each conversion, a direct memory transfer of the A/D data to memory is performed under the control of the IBM PC 8237 DMA controller. This device transfers the data from each conversion to a specified segment of memory and keeps track of the total number of conversions performed without involving the 80x88 CPU. When the number reaches N, as specified by D%(0), DMA transfers cease and a terminal interrupt is generated if in the nonrecycle MODE (D%(3), or DMA transfers are repeated continuously to the same segment of memory if D%(3) specifies the recycle (autoinitialize) MODE. Note that once MODE 6 has enabled DMA, conversions continue regardless of what other programs the user may be running. Also the HRES.BIN driver may be erased from memory. DMA is performed purely by the system and DAS-HRES hardware, is a background operation, and is extremely fast. Throughput is limited by the speed of the A/D converter and settling of the sample hold and with the A/D used on the DAS-HRES is about a maximum of 47,600 samples/sec. A technical description of the IBM PC DMA arrangements is contained in Appendix E.

To return data to a BASIC array when MODE 6 is operating or has finished operation, use MODE 9. (MODE 9 will automatically correct/calibrate the raw data stored in RAM, and place corrected data in the BASIC array.) To assess the progress of an operation initiated by MODE 6, use MODE 8. To abort a DMA operation initiated by MODE 6, use MODE 7. For detailed descriptions of the features of these other MODEs, see the following sections.

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 15.

The gain of all the channels in DMA MODE must be the same due to the design of the DAS-HRES hardware. On entry, D%(4) selects one of the four possible gains.

The A/D may be triggered from either of two sources: the Programmable Interval Timer or an external trigger pulse according to D%(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.

Entries:

MD% = 6 (MODE number)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

D%(0) = Number of conversions required (Word count). Range 1 to 32,767 or 0 for 32,768.

D%(1) = Segment of memory to receive data. Segments are on paragraph (16 bit) boundaries e.g. D%(1) = &H5000 would start loading A/D data at 320K. Be sure to choose an empty area of memory for a data buffer (always outside BASIC workspace)

D%(2) - Trigger source (Two are possible):

D%(2) = 0 : External trigger input. Conversions take place on positive transitions of the TRG0 input and continue until the word count is reached.

D%(2) = 1 : Programmable Interval Timer. In this case, IP0 should be held low until you want to start conversions. After IP0 goes high, IP0 will have no further effect.

D%(3) - Single cycle/Re-cycle operation:

D%(3) = 0 : One cycle. After completion of the number of conversions specified, DMA ceases, an interrupt is generated and DMA is disabled, setting the DMA status to zero.

D%(3) = 1 : Recycle. In this case data is continuously written to the same memory. D%(0) corresponds to the memory "buffer" length. This corresponds to a DMA auto-initialize operation. To stop DMA use MODE 7.

D%(4) = 0 - 3 : Sets gain for all channels:

D%(2)	GAIN
0	1
1	2
2	4
3	8

D%(5-15) (value does not matter)

FLAG% = X (value does not matter)

then

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

Returns:

MD% = 6 (unchanged)

D%(0-15) (unchanged)

Error Codes:

FLAG% = 0 (No Error; OK).
= 1 (Driver Not Initialized).
= 2 (MODE number out of range, < 0 or > 22).
= 11 (Number of conversions 0 or negative).
= 20 (Interrupt or DMA already active).
= 21 (Page wraparound).
= 26 (Gain code setting < 0 or > 3).

An example of the use of MODEs 1, 6, 7, 8, 9, and 17 will be found in EXH6.BAS.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Several details apply to the reliable use of MODE 6. First, you cannot rerun MODE 6 if a MODE 5 interrupt or a previous MODE 6 DMA operation is still active. Error #20 will result. If you request the recycle MODE which generates continuous DMA transfers, then MODE 7 which disables DMA and interrupts must be run before you can successfully run MODE 6 again. If you are in nonrecycle MODE, then it must have reached the word count which automatically disables DMA before MODE 6 can be run again. On completion of a DMA operation, the tristate DMA request (DRQ) drivers of the DAS-HRES are placed in the high impedance state. This allows more than one hardware device, or multiple DAS-HRES's to use the same DMA level as long as they do so sequentially.

Second, since the DMA page registers cannot be incremented by the controller, the maximum data area available is 64K (a page) for 32,767 conversions. Be sure that your data area is not in use by your program or altered by subsequent operations. Also be careful that your word count added to your segment address will not cross a page (64K) boundary. This condition which would otherwise cause a wrap round to the beginning of the page is detected by the driver and produces Error #21. To avoid this, a conservative rule is to use a segment address that is on a page boundary e.g. &H1000, &H2000 etc. Data may be retrieved using MODE 9 during or after the operation of MODE 6 and MODE 9 will not alter the memory.

ALSO NOTE that exit from MODE 6 cannot occur until IP0 has been taken high if using the Programmable Interval Timer.

Note that once DMA operation is initiated, it may well continue during following execution of the program. This is a "background" operation. Strange effects can occur if you inadvertently transfer data into your program area, be sure to use a free area of memory for your DMA buffer. DEBUG can help you locate free areas, they are usually loaded with zeros on boot-up.

If you wish to transfer more data than can be held in one 64K data segment, the manufacturer has optional data streamer software (STREAMER) which has been specially devised to perform continuous transfer through a DMA buffer to hard disk. With this software it is possible to continuously stream "gap less" data to your hard disk at more than 45KHz on a PC/AT (somewhat slower on an XT). A standard 20MB hard disk will hold about three minutes of data at 50KHz. The maximum speeds attainable are dependent on the type of your disk controller and seek time of your hard disk - contact the manufacturer for further information.

MODE 7: Disable DMA/Interrupt Operation

MODE 7 causes an immediate disable of any running interrupt or DMA operation initiated by MODEs 5, 6, 18, or 20. The operation will be abandoned at the time MODE 7 executes.

Entries:

MD%	= 7 (MODE number)
D%(0-15)	(value does not matter)
FLAG%	= X (value does not matter)

then

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


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returns:

MD% = 7 (unchanged)

D%(0-15) (unchanged)

Error Codes:

FLAG% = 0 (No Error; OK).
= 1 (Driver Not Initialized).
= 2 (MODE number out of range, < 0 or > 22).

For an example of the use of MODE 7, see EXH5.BAS, EXH6.BAS, EXH18.BAS, or EXH20.BAS.

MODE 8: Read DMA/Interrupt Status

MODE 8 allows you to monitor the status of a background operation initiated by MODEs 5, 6, 18, or 20.

Entries:

MD% = 8 (MODE number)

D%(0-15) (value does not matter)

FLAG% = X (value does not matter)

then

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

Returns:

MD% = 8 (unchanged)

D%(0) - Operation type as follows:
0 - None
1 - DMA (MODE 6)
2 - Interrupt (MODE 5)
3 - Interrupt (MODE 18)
4 - Interrupt (MODE 20)

D%(1) - Status of operation:
0 - Done (finished)
1 - Active (in progress)

D%(2) - Current word count (number of conversions, so far).

D%(3-15) - unchanged

Error Codes:

FLAG% = 0 (No Error; OK).
= 1 (Driver Not Initialized).
= 2 (MODE number out of range, < 0 or > 22).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

For an example of the use of MODE 8, see EXH5.BAS, EXH6.BAS, EXH18.BAS, or EXH20.BAS.

MODE 9: Transfer Data From Memory To Array

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 a word (2 bytes) of A/D data + channel number. MODE 9 will automatically correct/calibrate the raw data stored in RAM, and place corrected data in the BASIC array (based on the level of gain/offset correction selected in MODE 0).

MODE 9 can transfer data as a whole block or piece by piece. The number of words (or conversions) that you wish to transfer to the data and channel arrays is set into D%(0). The segment of memory that you wish to transfer data from is set into D%(1). Data can be transferred from the beginning of this segment (D%(2) = 0) or any other point. A/D data is transferred to a dimensioned integer array. The transfer will start at the pointer set into D%(3). To start at the beginning of the array DT%(*) set D%(3) = VARPTR(DT%(0)). If we want to start at the Mth. element then D%(3) = VARPTR(DT%(M)). In this way you can move any number of words (conversions) from any source location to any destination location in an array.

A/D data is shifted and also the MSB complemented and sign extended if the DAS-HRES is operating in Bipolar MODE. In Unipolar MODE, data ranges from 0 to 65535; in Bipolar, -32768 to +32767. In order to determine whether the DAS-HRES 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, then incorrect data may be obtained.

Note that you must be careful about the transfer parameters. In particular:

1. Do not transfer more words than an array will hold or overrun the end of the array. No checking is performed to detect this condition, which will corrupt BASIC workspace and cause strange effects.
2. 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.
3. Due to the reformatting of data that this MODE performs, it is not a general purpose block move utility.

Entries (Original Options):

MD% = 9 (MODE number)

D%(0) - Number of words to transfer (1 - 32767) or 0 for 32,768 max.

D%(1) - Source segment in memory (0 - 32767). Typically this will be set on an even 64K Page Boundry such as @H2000 or @H4000.

D%(2) - Starting conversion number (0 - 32767)

D%(3) - VARPTR (Data array) (e.g. VARPTR(DT%(0)))

D%(4-15) (value does not matter)

FLAG% = X (value does not matter)

then

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

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

Entries (New Options):

MD%	= 9 (MODE number)
D%(0)	= Number of words to transfer (1 - 32767) or 0 for 32,768 max.
D%(1)	= -1
D%(2)	= Starting conversion number (0 - 32767)
D%(3)	= VARPTR (Data array) (e.g. VARPTR(DT%(0)))
D%(4)	= VARSEG (Data array) (e.g. VARSEG(DT%(0)))
D%(5)	= Source segment in memory
D%(6)	= Source offset in memory
D%(7-15)	(value does not matter)
FLAG%	= X (value does not matter)

then

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

Returns:

MD%	= 9
D%(0-15)	(unchanged)

Error Codes:

FLAG%	= 0 (No Error; OK). = 1 (Driver Not Initialized). = 2 (MODE number out of range, < 0 or > 22). = 17 (Start conversion number, D%(2), negative). = 18 (Word count, D%(0), zero or negative).

For an example of the use of MODE 9, see EXH5.BAS, EXH6.BAS, or EXH20.BAS.

Once data acquisition has been set up as a constant background operation using the recycle options of MODE 5 or 6, a foreground program can be processing the data as it is acquired using MODE 9 to retrieve the data. This is excellent for graphic and digital oscilloscope applications.

Note that it is advisable to make the assignments:

```
D%(3) = VARPTR(DT%(0))
```

immediately before the CALL statement, as declaring a new simple variable after making this assignment will dynamically relocate the arrays and upset operation of this MODE. If the Variable I had not been declared prior to line xxx10, then the following sequence would cause a problem (see MODE 4 for a fuller description of this problem):

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
xxx10   D%(3) = VARPTR(DT%(0))
xxx20   I = 3
xxx30   CALL HRES (MD%, D%(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 since PEEK retrieves data a byte at a time, it is possible that an interrupt can occur between reading the low byte and high byte of a data word, thereby changing the halves of a word in midflight and causing your sequential PEEKs to return erroneous data. Using MODE 9 avoids this problem. In addition, using MODE 9 also performs the autocalibration function, which would have to be seperately implemented in a PEEK/POKE operation.

MODE 10: Set Counter 0 Configuration

MODE 10 sets the Counter 0 operating configuration. For a fuller discussion of the possible configurations, see Section 5 on timer/counter operations. Counter 0 may be set to one of six configurations:

0: Output goes high on terminal count. Output remains high until reloaded.

1: Programmable one-shot. Output pulses low for one clock on terminal count.

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

3: Square wave generator. Output high for one half of the count, low for the 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.

It is also possible to set Counters 1 and 2 of the 8254 to any of these configurations, but this is not supported by the HRES.BIN driver. Counters 1 and 2 are set by MODE 0 (initialize) to operate as divide-by-N counters in Configuration #2, as this is the best arrangement for triggering the A/D. To change their configuration use BASIC OUT instructions.

Entries:

MD%	= 10
D%(0)	= Configuration number (0 thru 5)
D%(1-15)	(value does not matter)
FLAG%	= X (value does not matter)

then

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

Returns:

MD%	= 10
D%(0-15)	(unchanged)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Error Codes:

FLAG% = 0 (No Error; OK).
= 1 (Driver Not Initialized).
= 2 (MODE number out of range, < 0 or > 22).
= 12 (config. number < 0 or > 5).

For an example of the use of MODE 10, see EXH10.BAS.

MODE 11: Load Counter 0

MODE 11 loads Counter 0. Since the counter is a 16-bit device, it supports count loads as high as 65,535. 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 C). If MODE 22 or 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 and event counting. If MODE 10 has been entered prior to MODE 11, Counter 0 configuration will remain the same.

Entries:

MD% = 11 (MODE number)

D%(0) = Counter data (-32768 to +32767) (note this corresponds to 0-65535)

D%(1-15) (Value does not matter)

FLAG% = X (value does not matter)

then

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

Returns:

MD% = 11

D%(0-15) Unchanged

Error Codes:

FLAG% = 0 (No error; OK).
= 1 (Driver not initialized).
= 2 (MODE number out of range, < 0 or > 22).

For an example of the use of MODE 11, see EXH10.BAS.

MODE 12: Read Counter 0

MODE 12 reads Counter 0. Counter data is 16 bits and integer variables are returned as signed 16-bit words (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). See Appendix C on integer storage.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Two types of Counter 0 read operation are possible in MODE 12. If D%(0) = 0 a plain read operation is performed. This should only be done if the counter data is known to be static i.e. gate input low or no more clock input pulses. If D%(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" i.e. while it is counting. Neither type of read will affect the counter data or operation.

Entries:

MD%	= 12
D%(0)	= 0 or 1 - type of read operation
D%(1-15)	(Value does not matter)
FLAG%	= X (value does not matter)

then

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

Returns:

MD%	= 12
D%(0)	= (Unchanged)
D%(1)	= Read data (-32768 to +32767)
D%(2-15)	= (Unchanged)

Error Codes:

FLAG%	= 0 (No error; OK). = 1 (Driver not initialized). = 2 (MODE number out of range, < 0 or > 22). = 16 (Read operation number < 0 or > 1).

For an example of the use of MODE 12, see EXH10.BAS.

MODE 13: Write Digital Output OP0-7

MODE 13 is used to write digital data to the 8-bit output port, OP0-7. Output data is checked to be in the range 0 - 255; when it is not, an error exit (Error # 13) occurs.

Entries:

MD%	= 13
D%(0)	= Output data (range 0-255)
D%(1-15)	= (Value does not matter)
FLAG%	= X (Value does not matter)

then

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

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

Returns:

MD% = 13 (Unchanged)

D%(0-15) (Unchanged)

Error Codes:

FLAG% = 0 (No error; OK).
= 1 (Driver not initialized).
= 2 (MODE number out of range, < 0 or > 22).
= 13 (output data < 0 or > 255).

For an example of digital I/O using MODE 13, see EXH13.BAS.

MODE 14: Read Digital Input IP0-7

MODE 14 allows you to read the state of digital inputs IP0-7. Data returned can range between 0 and 255 corresponding to all combinations of the eight input bits, IP0-7.

Entries:

MD% = 14

D%(0-255) (Value does not matter)

FLAG% = X (Value does not matter)

then

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

Returns:

MD% = 14

D%(0) = Contains input data (range 0 - 255)

D%(1-15) (Unchanged)

Error Codes:

FLAG% = 0 (No error; OK).
= 1 (Driver not initialized).
= 2 (MODE number out of range, < 0 or > 22).

For an example of digital I/O using MODE 14, see EXH13.BAS.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MODE 15: Output Data To One D/A

MODE 15 writes data to one or other of the D/A channels. The data is checked to be in the range of -32768 to 32767; if it is not, an error exit occurs and the D/A is not updated.

Entries:

MD%	= 15
D%(0)	= D/A channel # (0 or 1)
D%(1)	= D/A data (-32768 to 32767)
D%(2-15)	= (Value does not matter)
FLAG%	= X (Value does not matter)

then

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

Returns:

MD%	= 15
D%(0-15)	(Unchanged)

Error Codes:

FLAG% = 0 (No error; OK).
= 1 (Driver not initialized).
= 2 (MODE number out of range, < 0 or > 22).
= 14 (D/A data out of range, < -32768 or > 32767).
= 15 (D/A channel # not 0 or 1).

For an example of D/A output using MODE 15, see EXH15.BAS.

NOTE: The following formula details the relationship between D/A output voltage, and the D/A data.

Vout = 305.18 E-6 * D%(0)

or

Vout = (20/65535) * D%(0)

MODE 16: Output Data To Both D/As

MODE 16 writes data to both D/A channels as simultaneously as possible. D/A #1 is updated 7us after D/A #0 (standard 4.77MHz IBM PC). This MODE is an alternative to entering MODE 15 twice in succession and is useful for driving X/Y plotters, resolvers, etc. where the delays using MODE 15 twice in succession would produce nonideal responses.

Data is checked to be in the range of -32768 to 32767; if it is not, an error exit occurs and neither D/A is updated.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Entries:

MD% = 16

D%(0) = D/A #0 data (-32768 to 32767)

D%(1) = D/A #1 data (-32768 to 32767)

D%(2-15) (Value does not matter)

FLAG% = X (Value does not matter)

then

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

Returns:

MD% = 16

D%(0-15) (Unchanged)

Error Codes:

FLAG% = 0 (No error; OK).
= 1 (Driver not initialized).
= 2 (MODE number out of range, < 0 or > 22).
= 14 (D/A data out of range, < -32768 or > 32767) (either channel).

For an example of D/A output using MODE 16, see EXH15.BAS.

NOTE: The following formula details the relationship between D/A output voltage, and the D/A data.

Vout = 305.18 E-6 * D%(0) For channel 0

or

Vout = 305.18 E-6 * D%(1) For channel 1

MODE 17: Set Programmable Timer Rate

MODE 17 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 Counter input clock frequency may be 1MHz or 10MHz (selected by the "TIMER" jumper block) and is derived from a precision on-board crystal oscillator.

Output frequency = 1,000,000 / (N2 * N1) Hz (TIMER jumper at "1")

or

Output frequency = 8,000,000 / (N2 * N1) Hz (TIMER jumper at "8")

or

Output frequency = 10,000,000 / (N2 * N1) Hz (TIMER jumper at "10")

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

where

D%(0) = N2

D%(1) = N1

The minimum divisor for either counter is 2, the maximum 65,535, giving possible pulse rates ranging from 250KHz to less than 1 pulse/hour with a 1MHz clock, eight times these rates with the 8MHz clock, or 10 times these rates with a 10MHz clock. If you wish to perform conversions at rates in the 10-100KHz region, the 8 or 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. Counters 1 and 2 are set to divide-by-N MODE (8254 MODE 2) and output 1KHz (8KHz if the TIMER block is set to 8MHZ or 10KHZ if TIMER block is set to 10MHZ) by the initialization sequence of MODE 0.

Entries:

MD% = 17

D%(0) = Counter 2 divide data (2 - 65,535)

D%(1) = Counter 1 divide data (2 - 65,535)

D%(2-15) = (Value does not matter)

FLAG% = X (Value does not matter)

then

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

Returns:

MD% = 17

D%(0-15) = (Unchanged)

Error Codes:

FLAG% = 0 (No error; OK).
= 1 (Driver not initialized).
= 2 (MODE number out of range, < 0 or > 22).
= 10 (divisor 0 or 1 either counter).

For an example of the use of MODE 17, see EXH17.BAS, EXH5.BAS, EXH6.BAS, or EXH20.BAS.

MODE 18: Analog Output/Input On Interrupt

The DAS-HRES hardware may generate an interrupt either from the EOC (end of conversion) signal from the A/D or the terminal interrupt signal from the DMA controller. If the A/D converter is triggered by the on-board timer, the resulting EOCs will provide a periodic interrupt, delayed about 12 microseconds (depending on A/D conversion time) from the timer pulse. This interrupt can be used for a variety of purposes, to transfer A/D data as in MODE 5 or initiate a channel block scan as in MODE 20. The action depends on the interrupt service routine.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MODE 18 provides an interrupt service routine that transfers data out of a BASIC integer array to one or other of the D/A converters and optionally if D%(4) is non-zero, 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. The channel scan limits, channel gains and interrupt rate (from timer) are set independently preceding this MODE using MODEs 1, 21 and 17 respectively. 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. This latter MODE is useful for waveform generation etc.

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, 4.77MHz or 8MHz clock etc. but typically with a 4.77MHz clock, interrupt rates approaching 4KHz are possible. 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.

For programming convenience, data is interchanged directly with BASIC integer arrays, **but it is important that these arrays do not change their location in memory after MODE 18 becomes active**. Interpreted BASIC stores arrays in memory above the data space used for simple variables. Introducing a new simple variable will dynamically relocate (or "shuffle up") the arrays in memory. It is therefore essential to assign or declare all variables that are used in your program at the beginning of the program (structured programming style) to avoid this problem. This is a price paid for the convenience of interchanging data directly with the array rather than using a buffer area outside BASIC's workspace (as in MODEs 5 and 6). It is simple to reassemble MODE 18 to use an external buffer if this is preferred.

If you elect to output multiple cycles from the D/A array and have enabled A/D input into an array, the A/D data will overwrite the data in the A/D array for the number of cycles specified.

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. Once interrupts have been initiated by MODE 18 they will run as a background operation until the number of cycles of output specified have been performed. If you wish to terminate operation of MODE 18 before this then use MODE 7. MODE 8 can also be used to determine the status of a MODE 18 operation.

EXH18.BAS provides an example of the use of MODE 18 by outputting a variable frequency sine wave from a D/A and optionally gathering A/D data.

Entries (Original Options):

MD%	= 18
D%(0)	= D/A channel number (0 or 1 valid)
D%(1)	= Number of words to transfer (1 - 32767 valid; 0 for 32768 max)
D%(2)	= Number of cycles to transfer array (0 - 65535) (0 = continuous transfer)
D%(3)	= D/A data array pointer (VARPTR(DA%(0))
D%(4)	= A/D data array pointer (VARPTR(AD%(0)) (If D%(4) = 0 then A/D data is not transferred)
D%(5-15)	= (Value does not matter)
FLAG%	= X (Value does not matter)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

then

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

Entries (New Options):

MD%	= 18
D%(0)	= D/A channel number (0 or 1 valid)
D%(1)	= Number of words to transfer (1 - 32767 valid; 0 for 32768 max)
D%(2)	= Number of cycles to transfer array (0 - 65535) (0 = continuous transfer)
D%(3)	= -1
D%(4)	= -1
D%(5)	= Offset of D/A data array pointer (VARPTR(DA%(0))
D%(6)	= Segment of D/A data array pointer (VARSEG(DA%(0))
D%(7)	= Offset of A/D data array pointer (VARPTR(AD%(0))
D%(8)	= Segment of A/D data array pointer (VARSEG(AD%(0))
D%(9-15)	= X (Value does not matter)
FLAG%	= X (Value does not matter)

then

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

Returns:

MD%	= 18
D%(0-15)	= Unchanged

Error Codes:

FLAG%	= 0 (No error; OK).
	= 1 (Driver not initialized).
	= 2 (MODE number out of range, < 0 or > 22).
	= 15 (D/A channel number out of range).
	= 18 (Word count zero or > 32767).
	= 20 (Interrupt already active).

For an example of the use of MODE 18, see QBEXH18.BAS.

MODE 19: Analog Trigger Function

MODE 19 provides an analog trigger function similar to an oscilloscope trigger. It is sometimes useful to wait for a voltage to reach a certain level before starting to gather data and MODE 19 provides this capability. Any of the analog input channels may be designated as a trigger channel, and you may set the level and slope for triggering.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The main use for MODE 19 is in front of any of the other data acquisition MODEs as a gating or wait loop until the specified analog trigger conditions are met. Since it is possible to get stuck in the wait loop indefinitely if the trigger conditions are not fulfilled, you can also exit MODE 19 by hitting any key which will return you to the calling program. Also as this MODE requires the use of the DAS-HRES mux scan setting register at Base Address + 2 for selecting the trigger channel, the previous value of this register is saved and restored on exit.

Parameters D%(0) thru D%(2) control triggering and select the trigger channel number, the trigger level, and the trigger direction (slope). D%(0) specifies the trigger channel number. It 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. The voltage level at which triggering occurs is set by D%(1) in bits, ranges of -32768 to +32767 bits corresponding to Bipolar input ranges and 0 to 65535 for Unipolar ranges. The direction of triggering or slope is controlled by D%(2); for instance, if D%(1) = 16384 on the +/-10V range the trigger level will be +5V and if D%(2) = 0 (positive slope) triggering will take place when the signal exceeds +5V, alternatively if D%(2) = 1 (negative slope) triggering would take place when the trigger signal becomes less than +5V.

The gain of the trigger channel is set by D%(3).

Entries:

MD% = 19

D%(0) = Channel number (0 - 7)

D%(1) = Trigger level (-32768 to +32767 bits Bipolar or 0 - 65535 Unipolar)

D%(2) = Slope (0 = positive, 1 = negative)

D%(3) = 0 - 3, sets gain of trigger channel:

D%(3)	GAIN
0	1
1	2
2	4
3	8

D%(4-15) = X (Value does not matter)

FLAG% = X (Value does not matter @Newpage)

then

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

Returns:

MD% = 19

D%(0-15) = (Unchanged)

Error Codes:

FLAG% = 0 (No error; OK).
= 1 (driver not initialized).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

= 2 (MODE number out of range, < 0 or > 22).
= 23 (trigger channel out of range, 0-7).
= 24 (trigger data out of range < 32768 or > 32767 if Bipolar set, < 0 or > 65535 if Unipolar).
= 25 (slope data not 0 or 1).
= 26 (gain code data < 0 or > 3).

For an example of the use of MODE 19, see QBEXH19.BAS.

MODE 20: A/D Scan Of Channels On Interrupt

MODE 20 is similar in operation to MODE 5, except that instead of performing a single A/D conversion on each trigger pulse, a complete scan of the channels, as specified by the scan limits set in MODE 1, is performed. MODE 20 installs an interrupt handler to perform this function.

Due to the execution delays of the interrupt handler, block scan rates are limited to about 500 scans/second (see below) on a standard 4.77MHz clock PC. During the block scan, channels are sampled at about 100 microsecond intervals, although other interrupts may produce variable delays in the scan rate. If this is a problem, it is recommended to suppress other interrupts (especially the PC timer interrupt) while data is collected using MODE 20. Note that the scan rate attainable is dependent on the number of channels in the block. The numbers given above apply to 16 channels, if you scanned only four channels, then 2000 scans/second would be possible (Channels x scan rate < 8000). Better performance can be expected on the PC/AT and "turbo" PC's. If you start missing channels or foreground processes slow to a crawl, you will know that you are operating beyond or close to the maximum interrupt rate. The setup parameters are similar to MODEs 5 and 6.

The gain of each channel will be set according to the predefined sequence set up by MODE 21. If MODE 21 has not been entered prior to running MODE 4, all gains will default to 1.

Entries (Original Options):

MD% = 20

D%(0) = Number of conversions required (Word Count). Range 1 to 32,767 or 0 for 32,768 max.

D%(1) = Memory segment to receive data

D%(2) = Trigger source. There are 2 possible:

D%(2) = 0 : External trigger input. Interrupts will begin as soon as MODE 20 has executed. A low to high transition on TRG0 initiate each scan and interrupts remain enabled until the word count is reached in non-recycle MODE, or are continuous in recycle MODE.

D%(2) = 1 : Programmable interval time. In this case, IP0 should be held low until you want to start interrupts. After scanning commences, IP0 will have no further effect. Each scan is initiated by a timer pulse. Note that exit from MODE 20 cannot occur until IP0 goes high.

D%(3) = Cycle/Non-recycle operation.

D%(3) = 0 : Nonrecycle. After completion of the number of conversions specified, interrupts are disabled.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

D%(3) = 1 : Recycle. Data is continuously written and rewritten to the same memory area. In this case D%(0) specifies the length of the memory buffer in words (1 word = 2 bytes).

D%(4-15) = (Value does not matter)

FLAG% = X (Value does not matter)

then

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

Entries (New Options):

MD% = 20

D%(0) = Number of conversions required (Word Count). Range 1 to 32,767 or 32,768 max.

D%(1) = -1

D%(2) = Trigger source. There are 2 possible:

D%(2) = 0 : External trigger input. Interrupts will begin as soon as MODE 20 has executed. A low to high transition on TRG0 initiate each scan and interrupts remain enabled until the word count is reached in non-recycle MODE, or are continuous in recycle MODE.

D%(2) = 1 : Programmable interval time. In this case, IP0 should be held low until you want to start interrupts. After scanning commences, IP0 will have no further effect. Each scan is initiated by a timer pulse. Note that exit from MODE 20 cannot occur until IP0 goes high.

D%(3) = Cycle/Non-recycle operation.

D%(3) = 0 : Nonrecycle. After completion of the number of conversions specified, interrupts are disabled.

D%(3) = 1 : Recycle. Data is continuously written and rewritten to the same memory area. In this case D%(0) specifies the length of the memory buffer in words (1 word = 2 bytes).

D%(4) = Offset of memory to receive data.

D%(5) = Segment of memory to receive data.

D%(6-15) = (Value does not matter)

FLAG% = X (Value does not matter)

then

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

Returns:

MD% = 20

D%(0-15) = (Unchanged)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Error Codes:

FLAG% = 0 (No error; OK).
= 1 (Driver not initialized).
= 2 (MODE number out of range, < 0 or > 22).
= 11 (number of conversions zero or negative).
= 20 (DMA or interrupt already active).

Note that D%(0) provides the total number of conversions required, not the number of scans.

Number of conversions = Number of scans x Channels per scan

For an example of the use of MODE 20, see EXH20.BAS.

MODE 21: Set Channel Gains

MODE 21 sets up a gain/channel table internal to the driver for it to use in MODEs 3, 4, 5, and 20 that support software switching of the gain between conversions.

If after initializing MODE 0 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).

Entries:

MD% = 21

D%(0) = Gain code for Channel 0

D%(1) = Gain code for Channel 1

D%(2) = Gain code for Channel 2 etc.

.. .. ..

.. .. ..

D%(15) = Gain code for Channel 15, as follows:

D%(3)	GAIN
0	1
1	2
2	4
3	8

FLAG% = X (Value does not matter)

then

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

Returns:

MD% = 21

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

D%(0-15) = (Unchanged)

Error Codes:

FLAG% = 0 (No error; OK).
= 1 (Driver not initialized).
= 2 (MODE number out of range, < 0 or > 22).
= 26 (Gain code < 0 or > 3).

Some programming examples for the DAS-HRES are as follows:

```
xx000  MD% = 21              'select MODE 21
xx010  D%(0) = 3             'channel 0 gain = 8
xx020  D%(1) = 1             'channel 1 gain = 2
xx030  FOR I = 2 TO 7        'channels 2-7 gain = 1
xx040  D%(I) = 0 : NEXT I
xx050  D%(8) = 1             'channel 8 gain = 2
xx060  FOR I = 9 TO 13       'set channels 9-13 gains = 4
xx070  D%(I) = 2 : NEXT I
xx080  D%(14) = 1            'channel 14 gain = 2
xx090  D%(15) = 0            'channel 15 gain = 1
xx100  CALL HRES (MD%, D%(0), FLAG%)
xx110  IF FLAG% <  > 0 THEN PRINT "Error #";FLAG%;" in setting gains":STOP
xx120                        '.....  Continue program
```

NOTE: MODE 21 range checks all the channel-gain data in elements D%(0) thru D%(15) whether or not you are using the channels. For instance if you have used MODE 1 to set the scan limits from 2 to 5, only Channels 2, 3, 4, and 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 error #26 if this is not so. This is so that if you change scan limits later, there will be no problems.

You may find that defining another array to store the gain configuration data will avoid confusion with your general data transfer D%(*) array and will keep a permanent record of all the channel gains. For example,

```
yy000  MD% = 1               'set scan limits
yy010  D%(0) = 3             'start on channel 3
yy020  D%(1) = 6             'end on channel 6
yy030  CALL HRES (MD%, D%(0), FLAG%)
yy040  IF FLAG% <  > 0 THEN PRINT "Error #";FLAG%;"in setting scan
           limits":STOP
yy050                        'Now set channel gains (this can also precede
                                  MODE 1 if you prefer)
yy060  MD% = 21
yy070  DIM G%(15)            'dimension a new array for channel gain data
yy080  G%(4) = 3             'set channel 4 to a gain of 8
```

NOTE: BASIC initializes all other array elements to zero automatically selecting a gain of 1 on all other channels, so you only have to define the exceptions.

```
yy090  CALL HRES (MD%, G%(0), FLAG%)    'reference G%(0) in CALL instead of
                                              D%(0).
yy100  IF FLAG% <  > 0 THEN PRINT "Error #";FLAG%;" in setting gains":STOP
yy110                                  '.  .  .  Continue program, G%(*)
                                             retains the gain configuration
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MODE 22: Output A Square Wave Frequency

MODE 22 provides a quick and easy way of outputting a square wave of defined frequency from the DAS-HRES. It performs all the setup necessary to connect Counter 0 to the internal 500KHz crystal clock source (see Section 5) 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 / D%(0) Hertz

Practical output frequencies range from 25KHz down to 1.5 Hz. Since, by nature, the counter divides by an integer (D%(0)), finer intervals of frequency are obtainable at lower frequencies. Due to the operation of the 8254 Counter-Timer, if D%(0) is an odd number, one half of the square wave will be one clock period (10 microseconds) longer than the other half. If D%(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.

Entries:

MD% = 22

D%(0) = Counter divider

D%(1-15) = (Value does not matter)

FLAG% = X (Value does not matter)

then

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

Returns:

MD% = 22

D%(0-15) = (Unchanged)

Error Codes:

FLAG% = 0 (No error; OK).
= 1 (Driver not initialized).
= 2 (MODE number out of range, < 0 or > 22).
= 10 (D%(0) divider data 0 or 1).

An example of the use of MODE 22 is in EXH22.BAS.

MODE 23: Memory Allocate

MODE 23 uses DOS Memory Allocate Function 48 to reserve a good memory buffer for DMA. This MODE also returns a segment address in D%(5), a zero offset in D%(6), and the value of the actual segment allocated by DOS in D%(7) (this value may be different than the segment address in D%(5) but must be passed to MODE 24 to deallocate the memory).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MODE 23 may be used with MODE 5, MODE 6, MODE 18, and MODE 20 in BASICA, QuickBASIC, PASCAL, C, or FORTRAN – but not in the QuickBASIC environment! To use MODE 5 and MODE 20 in the QuickBASIC environment, use ARRAY (instead of MODE 23) to allocate memory. For MODE 5, see Appendices E and F for information on memory allocation for DMA.

Entries:

D%(0) = Number of words or bytes to allocate.

D%(1) = 0 for bytes, 1 for words.

Returns:

D%(5) = Segment Address (also good for DMA).

D%(6) Offset (always 0).

D%(7) Actual allocated segment.

FLAG% = 0 if OK, 30 for Memory Allocate/Deallocate Error

A typical program entry preceding MODE 23 might read

```
210  MD%  = 23                 'Select mode number
220  D%(0)  = 5000             'Allocate 5000 words
230  D%(1)  = 1                'Words (should be the same as MODE 1 D%(1)
240                            'Execute MODE 23
250  CALL HRES(MD%, D%(0),FLAG%)
260  DMASEG=D%(5)              'Segment Address (also good for MODE 5).
270  DMAOFF=D%(6)              'Zero Offset Address.
280  ACTSEG=D%(7)              'Pass to MODE 24 to release memory.
290  IF FLAG% <> 0 THEN PRINT "MODE 23 ERROR #";FLAG%
300                           ,Continue program
```

MODE 24: Deallocate Memory Segment

MODE 24 uses DOS Deallocate Memory Function 49 to release memory allocated by MODE 23. This MODE requires the actual segment to be passed in D%(0); this value is returned by MODE 23 in D%(7). MODE 24 should be called when the program is finished using the memory buffer allocated by MODE 23.

Entry:

D%(0) = Actual segment value returned by D%(7) under MODE 23.

Returns:

FLAG% = 0 if OK.
= 29 for Memory Allocate/Deallocate Error.

A typical program entry preceding MODE 24 might read as follows:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
210  MD%=24                      'Select mode number.
220  D%(0) = ACTSEG              'Segment to release from MODE 23.
230                              'Execute MODE.
240  CALL PDMA(MD%, D%(0), FLAG%).
250  IF FLAG% <> THEN PRINT "MODE 24 ERROR #";FLAG%.
260                              'Continue program
```

4.3 SUMMARY OF ERROR CODES

If for any reason the FLAG% variable is returned nonzero, 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. The only exception to this rule is Error #22 (Hardware Failure or Installation Error) , where an attempt will be made to initialize the hardware even if there appears to be a problem so that other MODEs may possibly be run to diagnose the problem. The following is a list of error codes:

ERROR	FAULT
1	Driver not initialized. Using another MODE before initializing with MODE 0.
2	MODE number out of range. Specifying MD% less than 0 or greater than 22.
3	Base Address out of range. Valid addresses must be within 256-1008 (100h to 3F0h).
4	Interrupt level out of range (< 2 or > 7).
5	DMA level not 1 or 3.
6	Mux scan limits out of range in differential configuration. Either limit < 0 or > 15.
7	Not Used.
8	Not Used.
9	No End Of Conversion (EOC) signal from A/D subsystem.
10	One or both Counter values out of range.
11	Number of conversions out of range (0 or negative in MODEs 4, 5, or 6).
12	Counter 0 configuration number out of range in MODE 10 (< 0 or > 5).
13	Digital output data out of range in MODE 13 (< 0 or > 255).
14	D/A data out of range in MODEs 15 or 16 (-32768 or > 32767).
15	D/A channel number out of range (< 0 or > 1).
16	Counter 0 read operation not 0 or 1.
17	Start convert number out of range.
18	Requested sample count out of range (Word count 0 or negative).
19	Trigger MODE not 0 or 1.
20	DMA/Interrupt operation already active.
21	DMA page wrap around.
22	Hardware failure or installation error (Base Address set wrong)
23	Analog Trigger Channel out of range.
24	Analog Trigger data out of range.
25	Analog Trigger Slope data not 0 or 1.
26	Gain code data out of range.
27	Auto-calibration code out of range.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

28	Possible memory index overflow.
29	Memory allocation/deallocation error.

Error detection after the CALL routine is easily implemented:

```
xxx10  CALL HRES (MD%, D%(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. With a suitable error-handling subroutine, it can be left permanently in the program.

■ ■ ■

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 5

PROGRAMMING DAS-HRES

5.1 OVERVIEW

At the lowest level, DAS-HRES is programmable with I/O input and 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 (IN AL,DX and OUT DX,AL). Use of these functions usually involves formatting data and dealing with absolute I/O addresses. Although not demanding, this can require many lines of code and necessitates an understanding of the devices, data format, and architecture of the DAS-HRES. To simplify program generation, a special I/O driver routine (HRES.BIN) is included in the DAS-HRES Distribution Software. This may be accessed from BASIC by a single-line CALL statement and covers the majority of common operating modes. The various modes of the CALL routine select all the functions of the DAS-HRES, format and error check data, and perform frequently used sequences of instructions. An example is Mode 3, which performs a sequence of operations required to perform an A/D conversion, check A/D status, read data and increment the multiplexer.

Using the HRES.BIN driver saves programming time and has some other benefits as well. The driver also supports data collection on interrupt or DMA from an external source or the DAS-HRES Timer/Counter. Note that BASIC has no interrupt or DMA processing functions, and so-called *background* data collection using these methods is available only by using the CALL routine.

Both methods of programming using INP and OUT functions and the CALL routine achieve the same result, and you are free to choose either although usually the BASIC programmer will find the CALL routine much simpler to implement. If you need to perform specialized scans such as nonsequential channels, special interrupt routines etc., you can modify the HRES.BIN driver to your requirements. The fully commented assembly source file (HRES.ASM) is available by purchasing the PCF-HRES package and can be reassembled using the Microsoft Macro-Assembler. The HRES.ASM code is a good starting point for assembly language programmers who wish to modify the standard driver. For the steps involved in generating a BLOADable HRES.BIN file, follow the instructions in the file HOWTOBIN.DOC file in the Distribution Software (enter TYPE HOWTOBIN.DOC after the DOS prompt).

5.2 PROGRAMMING TIPS FOR BASIC

Some BASIC commands, which you may have rarely used but which may be useful with DAS-HRES are

1. **`WAIT port, n[,m]`**

   Data read at the port is EXCLUSIVE-OR'd (XORed) with integer expression "m" and AND'd with "n". If the result is zero, BASIC loops back and tests the port again until a non-zero result is obtained. This is an excellent way of making your program wait until some desired external condition is attained.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2. **ON TIMER (n) GOSUB line**

This command is available only in DOS 2.0 and above. In effect it provides you with a pseudo-interrupt. After execution of each BASIC statement, BASIC checks the timer to see if the condition >n (1 < n < 86,400 seconds) is satisfied. If so, control passes to the subroutine. Otherwise, the next line is executed. This polling of the timer is called *trapping* and is activated by

```
TIMER ON
```

and disabled by

```
TIMER OFF
```

Note that trapping occurs only while your BASIC program is executing (unlike a true interrupt) and can be slightly delayed by statements that require a lot of execution time. If you need to sample at long intervals, use the ON TIMER command is an alternative to using the internal DAS-HRES counter timer.

5.3 LOADING MACHINE LANGUAGE CALL ROUTINE *HRES.BIN*

HRES.BIN is a general-purpose BIN file that provides access to the DAS-HRES callable modes when used with GWBASIC or BASICA. The file may be loaded using the BASIC BLOAD command. The call structure for this file is

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

Direct I/O using BASIC INP and OUT can be somewhat tedious to implement, although much of the required programming can be handled in subroutines. Use of the CALL routine described in the this section avoids these problems, circumvents the execution time delays of interpreted or compiled BASIC, and also permits high speed DMA (Direct Memory Access - see Appendix E) and interrupt-driven operations that BASIC does not support.

In order to make use of the CALL routine, HRES.BIN, it must first be loaded into memory. You must avoid loading it over any part of memory being used by another program (BASIC - watch out for BASICA.EXE if you are using GW BASIC), print spoolers, or RAM-Disk. If you do interfere with another program's use of memory, the CALL routine will not work and most likely your PC will do strange things for a second or two and then hang up (no damage will result, but to get things going again you will have to turn off power and wait a few seconds before turning on again). Here is another word of advice: when you write programs with CALLs in them, **SAVE them before you run them!** Note that the information given in this section is general and would apply to loading any CALL or USR routine and supplements the limited information in Appendix C of the *IBM BASIC MANUAL* .

When you load the HRES.BIN CALL routine, you have two options depending on the size of your available memory. Due to the design of Microsoft BASIC, the maximum memory segment that BASIC is able to use is 64K bytes. If BASIC is using its maximum 64K you will get the following message on power up or from DOS by entering A> BASIC(A) :-

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


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The exact number of *Bytes free* varies with the version of BASIC(A) and DOS but is usually greater than 60000 bytes if an excess of memory over and above what BASIC can use is available. When the number of memory bytes free is less than 60000, this is an indication that your PC's memory is already fully utilized and BASIC is adjusting to this condition by using less than its possible 64K maximum workspace. If this is the case, you will have to load the CALL routine by further forced contraction of the BASIC workspace and loading the routine at the end of the newly defined workspace. HRES.BIN will occupy about 45947 bytes but to keep things simple, let's clear a 5K (5120 byte) space for it.

Step 1 is to work out how much memory BASIC is able to use. First load BASIC(A) from DOS with the following:

```
BASIC(A)
```

Note the number of bytes free in BASIC's greeting messages. Next do a SYSTEM command to return to DOS and this time load BASIC(A) with the optional /M parameter, as follows:

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

where WS is a number (not a variable).

Try setting the WS (workspace) parameter to 30000 or 40000 and then note the number of bytes free in the greeting message. The objective is to determine the workspace that must be specified to reduce the bytes free by at least 5120 bytes. Once this is determined 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
```

Next, we need to know what segment BASIC is occupying in memory. In all versions of Microsoft derived BASIC, this can be found from the contents of absolute memory locations &H511 and &H510 which hold the current BASIC segment which we can call SG. SG can be determined 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 we can now load the CALL routine will simply be at the end of the working space For example,

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

The routine can now be loaded as follows:

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

A BLOAD must be used as we 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%, D%, FLAG% prior to the CALL sequence as follows:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
xxx70   DEF SEG = SG
xxx80   HRES = 0
xxx90   CALL HRES (MD%, D%(0), FLAG%)
```

Note that in interpreted Microsoft BASIC, HRES 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. We have chosen HRES as a name as it makes CALL HRES easy to associate with the device and would distinguish it from any other CALLs to other routines that might be in the same program. In compiled BASIC, Quick Basic or other compiled languages (FORTRAN, PASCAL, etc.) the significance of HRES in the CALL is different. In this case, it is the name of the external routine that the linker will be looking for when linking after compilation. HRES.OBJ is supplied on the software disk for this purpose (see Sections 4.7 and 4.8).

Returning to interpreted BASIC, HRES is the offset (actually zero) from the current segment as defined by the last DEF SEG statement that tells your BASIC interpreter where the CALL routine is located. Be careful that you do not inadvertently redefine the current segment somewhere in a program before entering the CALL. If you are using DEF SEG in other parts of your program, it is good practice to immediately precede the CALL statement by the appropriate DEF SEG statement (the same one you preceded your BLOAD with) even at the cost of duplication. This precaution can save a lot of wasted time and frustration from crashing your computer!

Another important detail is that CLEAR sets working space from the bottom of the BASIC working area up whereas we must set aside space for our subroutine from the top of available memory down. If we attempt to CLEAR more space than is actually available, we will end up loading our routine over the end of the BASIC program, data space, and stack and will hang up the computer. Be careful this does not happen inadvertently if you are memory limited and later load BASIC with DEBUG or some other memory resident program without making a compensating reduction in the workspace (WS) declaration in the CLEAR statement. When memory is limited, setting up a workspace that is a considerable amount less than what is available is a simple precaution. Another precaution in this regard occurs on computers (other than the IBM PC) that do not contain most of the BASIC in ROM. These machines load BASIC completely into RAM in the form of a BASICA.COM and a BASICA.EXE file. BASICA.COM, DOS and the BASIC program space load from the bottom of memory up and BASICA.EXE loads from the top of memory down. The free memory ends up in the middle and some contraction of the BASIC program space will automatically take place as soon as the total memory minus that used by DOS and BASICA.EXE becomes less than 64K. This can easily happen on a machine with less than 128K of memory in which case further forced contraction of the workspace is the only way of loading the CALL routine.

These considerations all sound a little on the complex side, but like all things they are not once you understand what you are doing. To further assist you, run and list the file EXH0.BAS. This gives you an examples of loading and using the CALL routine and ready made loading and initializing code that you can merge into your own programs.

The second option is somewhat simpler and applies when you have plenty of memory (generally 256K or more) and you have the luxury of loading the CALL routine outside the BASIC workspace. In this case, choose a segment that has 4K bytes clear at its beginning. For example we might choose &H6000 which is at 384K. Then proceed as follows:

```
xxx10   DEF SEG = &H6000      'Sets up load segment
xxx20   BLOAD "HRES.BIN",0    'Loads at 6000:0000
xxx30   HRES = 0
xxx40   DIM D%(15)
   ""
   ""
   ""
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
xxxxx   DEF SEG = &H2800
xxxyy   CALL HRES (MD%, D%(0), FLAG%)
xxxzz   etc.
```

An example of this approach is also contained in file EXH0.BAS. Before you try loading outside the workspace, be sure you really do have an unused 5K of memory at 384K. You can change the DEF SEG statements in EXH0.BAS and experiment with loading the CALL routine at other locations. Usually any clash with another program's use of the same memory results in obliteration of some of the routine code and a failure to exit and return from the routine. The computer hangs up, and the only cure is to switch off, wait a few seconds and turn on the power again. Note that memory resident programs such as Borland's Sidekick will raise the loading floor in memory for additional programs such as BASIC(A). In this case even machines with large amounts of memory may run out of space or require loading the HRES.BIN much higher in memory than the example above. Also higher revisions of DOS with additional features use more resident space in memory than earlier versions and may require loading the CALL routine higher up.

5.4 FORMAT OF THE CALL STATEMENT

This explanation may assist you in understanding how the CALL transfers execution to the machine language (binary) driver routine. 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 HRES.BIN driver must be of the form

```
xxxxx   CALL HRES (MD%, D%(0), FLAG%)
```

As explained in the previous section, HRES is the address offset from the current segment of memory as defined in the last DEF SEG statement. In all of our examples, we have chosen to define the current segment to correspond with the starting address of the CALL routine, therefore this offset is always zero and HRES = 0.

The three variables within brackets are known as the CALL parameters. On 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. Three important format requirements must be met, as follows:

1. The CALL parameters are positional. The subroutine knows nothing of the names of the variables, just their locations from the order of their pointers on the stack. If you write:

   ```
   xxxxx CALL HRES (D%(0), MD%, FLAG%)                    'Incorrect use
   ```

   you will mix up the CALL routine, since it will interpret D%(0) as the mode number, the mode MD% as the data etc. **The parameters must always be written in the correct order, as follows:**

   (mode, data, errors)

2. The CALL routine expects its parameters to be integer type variables and will write and read to the variables on this assumption. If you slip up and use a noninteger (real single or double precision) variable in the CALL parameters, the routine will not function correctly. No error checking is done in the CALL on the variable type, so be careful since you may crash the computer!

3. You **cannot** perform any arithmetic functions within the parameter list brackets of the CALL statement. For example.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
CALL HRES (MD% + 2, D%(0) * 8, FLAG%)          'Incorrect use
```

is illegal and will produce a syntax error.

4. You cannot use constants for any of the parameters in the CALL statement. The following is illegal:

```
CALL HRES (7, 2, FLAG%)              'Incorrect Use
```

This must be programmed as

```
xxx10  MD% = 7
xxx20  D%(0) = 2
xxx30  CALL HRES (MD%, D%(0), FLAG%)
```

Apart from these restrictions, you can name the integer variables what you want, the names in the examples are just convenient mnemonics. Strictly, you should declare the 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 HRES.BIN CALL routine require multiple items of data to be passed in an array. For this reason, D%(0) is specified as the data variable so that the CALL routine can locate the whole array from the position of its initial element.

You can use unusual techniques with the CALL parameters. Let's say we wanted to record or output a whole series of data in a FOR . . . NEXT loop. You can dimension your D% array as a two-or-more dimension array. For example,

```
xxx00  DIM D%(4,100)
xxx10  DEF SEG = SG
xxx20  FOR I = 0 to 100
xxx30  CALL HRES (MD%, D%(0,I), FLAG%)
xxx40  NEXT I]
```

Similarly, 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 OK 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.

Use the same CALL structure for selecting any of the functions of the board. This makes the CALL structure easy to remember and helps to avoid errors compared to CALLs with variable offsets and number of parameters. One consequence of this is that not all functions or modes use all of the data array D%(*). In practice it is dimensioned to D%(15) to allow for the needs of Mode 21 (which sets the channel gains), but most of the other modes use only a few elements. In the interests of clarity, our example programs use in-line code with each CALL written out separately, but you can just as readily put the CALL into one subroutine for the whole program and save a lot of lines of code. For example,

After BLOADing the HRES.BIN driver (see EXH0.BAS) proceed as follows (this program logs channels 2 - 5 to disk every 10 seconds)

```
xx100  DIM D%(15)
xx110  MD% = 0
xx120  D%(0) = &H300 : D%(1) = 2 : D%(2) = 3
xx130  GOSUB 10000                    'initialize
xx140  MD% = 1
xx150  D%(0) = 2 : D%(1) = 5
xx160  GOSUB 10000                    'set scan limits
xx170  OPEN "MYFILE.DAT" FOR OUTPUT AS #1
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
xx180   TNOW = TIMER                          'get system time (in seconds)
xx190   MD% = 3
xx200   GOSUB 10000                           'do 1 A/D conversion
xx210   PRINT #1, D%(0)*10/32768              'save data to disk scaled in volts
xx220 IF INKEY$<>"" THEN CLOSE #1:END         'finish if a key has been pressed
xx230 IF TIMER > TNOW + 10 THEN GOTO xx180 ELSE GOTO xx220
                                              'get next channel at 10 second
                                                   intervals
10000   CALL HRES (MD%, D%(0), FLAG%)
10010   IF FLAG%<>0 THEN ?"Error # ";FLAG%;" in mode ";MD%:STOP
10020   RETURN
```

5.5 QUICKBASIC APPLICATIONS

The following section is to aide users of the DAS-HRES in writing programs in QuickBASIC 4.00 and 4.50. The various operating "Modes" in QUICKBASIC are identical to those described earlier for BASICA. In addition, there are example programs in the DAS-HRES Distribution Software that should clearly show how the QuickBASIC language can be used in programming the DAS-HRES. However, for those non-QuickBASIC experts, the following tutorial may prove helpful.

The QuickBASIC program will need to make calls to an external driver/library. A "Quick library" is provided in a file called HRES.QLB. This is a general-purpose library for use in the QuickBASIC environment (up to Version 4.5 - qb /l HRES.QLB) to provide access to the DAS-HRES operating modes. HRES.QLB may be loaded into the QB environment when QB is invoked from the DOS command line, as follows:

```
CALL QBHRES (MD%,VARPTR(D%(0)),FLAG%)
```

Optionally, an application program such as "MYPROG.BAS" could also be loaded as follows:

```
QB /L HRES MYPROG          (For QB Version 4.5)

QB /L HRESX MYPROG         (For QBX Version 7.0)
```

NOTE: HRESX.QLB is identical to HRES.QLB but is for use in the QuickBASIC Extended Environment under Version 7.0 (QBX).

Next, the driver has to be declared so it will be known to QB by the statement

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

This is unlike the method used in BASICA where the driver was made into a special .BIN file and then loaded into BASICA with the BLOAD statement.

One of the major differences between BASICA and QuickBASIC programs relates to declaring variable arrays. In QB, the first and third parameters are passed as variables, while the second parameter is passed as a pointer. This is necessary since this parameter must represent the offset of the command integer array. To pass the actual offset we will use the VARPTR function as:

```
CALL HRES(DM%, VARPTR(D%(0)), FLAG%)
```

Where just as in the BASICA case, DM% is the Mode, D%(N) is the data array, and FLAG% returns the

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

errors detected. The VARPTR function has returned the address of D%(0) which we will pass as a value to the driver that will use that value as a pointer to the first element of our command integer array D%.

D% is declared as a $STATIC array since it must reside in DGROUP in order to be relative to QB data segment as required by the driver. Since parameters are passed to and returned by the external driver, we will declare those parameters as inter-module global variables as:

```
COMMON SHARED D%(), A%()
```

The COMMON makes these variables GLOBAL between modules and the SHARED statement at the module level makes them known globally in this module. Now that we have named these variables in a COMMON statement, the position of their dimension statement determines whether the array is a $STATIC or a $DYNAMIC array. If the array is dimensioned BEFORE the COMMON statement,the array is $STATIC. If the array is dimensioned AFTER the COMMON statement,the array is $DYNAMIC. If the array is NOT GIVEN in the COMMON statement, it is $DYNAMIC, but not intermodule global.

```
DIM D%(6)                          '$STATIC
DIM A%(5000)                       '$STATIC

  COMMON SHARED D%(), A%()         '$DYNAMIC

  dat%(30000)                      '$DYNAMIC, but not intermodule global
```

The $ Metacommands are necessary for proper compilation using BC in DOS.

SEE the Section 2.5 ($STATIC and $DYNAMIC Arrays) and (COMMON statement) in the QuickBasic Language Reference Manual for a more detailed discussion of what factors determine array types.

The dat% array is used to pass the segment to MODES that collect data and store directly to an intermediate memory buffer as:

```
D%(1) = VARSEG(dat%(0))

CALL HRES(MD%, VARPTR(D%(0)), FLAG%)
```

Here, QB is managing the location of the buffer rather than the more hazardous practice of guessing where to put the data as:

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

At this point the declaration header should be as follows:

```
DIM D%(6)
DIM A%(5000)
COMMON SHARED D%(), A%()
DECLARE SUB DIRVER(MODE %, BYVAL dummy%, FLAG%)

'$DYNAMIC

dat%(30000)

'$STATIC
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A final note on arrays is that if it is desirable to erase and redimension an array during program execution, they should be declared as $DYNAMIC. The REDIM statement can now change the size, but not the dimensional structure. The ERRASE statement is not necessary.

In general, the differences between BASICA and QB should be studied in Appendix A of the Learning and Using QB manual, but a few common errors we have encountered can be eliminated by making note of the following points:

1. Watch out for reserved words (Appendix B Language Reference). Some of our example programs use an array sub%, which will cause an error. A good substitute is mux%.
2. Rounding Rules. Remember, a compiler will round integers differently than an interpreter. 0.5 will be rounded to 1 in BASICA, but to 0 in QB.
3. OVERFLOW errors. QB will use default or declared variable types for intermediate calculations where BASICA will make adjustments.

   X%=A%(I) * 1000 / 10000 may cause an overflow since

   A%(I) could = 20000, then 20000 * 1000 = 20000000 ,

   which is to large to fit an integer. The problem is not the assignment, but the evaluation of the multiplication. Using 1000! solves this problem.

5.6 MAKING EXECUTABLE PROGRAMS

To compile programs in BASIC, use the HRES.LIB library (supplied in the DAS-HRES Distribution Software) as shown in the following steps:

1. Compile the source file and generate an object file (.OBJ file) with the following command:

   ```
   BASCOM /O filename.BAS
   ```

2. Link the .OBJ file with HRES.LIB and BASCOM.LIB to create the .EXE file with the following command:

   LINK filename.OBJ,,,HRES.LIB+BASCOM.LIB

To compile QuickBASIC programs from the environment when the HRES.QLB has been loaded, it is necessary to have the HRES.LIB supplied on the DAS-HRES utility disk.

It is a good idea to make this library at the same time that the Quick Library is made. To make an executable from the environment, use the following sequence:

First invoke the environment by:

```
QB /L HRES MYPROG
```

then

Select the RUN option

Select the Make Exe option

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Select the Stand alone .exe option

Select the Make and Exit option

These commands will produce a stand alone HRES.EXE that does not require Run Time Support.

Alternately,

Select the RUN option

Select the Make Exe option

Select the Make .exe requiring BRUN40.EXE option

Select the Make and Exit option

These commands will produce a HRES.EXE that requires the Run Time support program BRUN40.EXE

To execute either type of program type the program name followed by < Enter >

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

```
BC /o MYPROG.BAS;

LINK MYPROG.OBJ+HRES.OBJ+HRESPCF.OBJ;
```

or

```
LINK MYPROG.OBJ,,,HRES.LIB;
```

The /o option given in the compiler line, causes references to the BCOM40.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.

Alternately:

```
BC  MYPROG.BAS;

LINK MYPROG.OBJ+HRES.OBJ+HRESPCF.OBJ;

LINK MYPROG.OBJ,,,HRES.LIB;
```

The absence of the /o option in the compiler line, causes references to the BRUN40.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 an executable the requires the presence of BRUN40.EXE in the root directory named \BIN or in the current directory at the time the program is executed. Refer to 6.2 in the Learning and Using MicroSoft QuickBASIC Manual for a discussion of the advantages of both types of executable programs.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Note that the Compiler and Linker expect to find executables in the root directory named \BIN, and Libraries in the directory named by the environment variable LIB.

To run either type of executable program, type **MYPROG**.

References:

Microsoft QuickBASIC Manuals

Advanced QuickBASIC 4.0, Murray Lesser, Bantam Computer Books

5.7 PROGRAMS, EXAMPLES, & UTILITIES

Refer to FILES.DOC in the Distribution Software for listings and descriptions of programs, utilities, and utilities available for DAS-HRES. To view FILES.DOC, enter **TYPE README.DOC**.

5.8 ASSEMBLY LANGUAGE PROGRAMS & CALLS IN OTHER LANGUAGES

To facilitate use of the I/O driver CALL routines in other languages (C, etc.), the assembly object code file HRES.OBJ is provided. This was assembled using the IBM or Microsoft Macro Assembler and may be linked to other object modules from compilers, etc. When using the linker, the routine's public name is HRES (see next section concerning use of the BASIC COMPILER). The fully commented source code and listings of the CALL routines will be found on the utility disk in file HRES.ASM. Simply run this through the Macro Assembler (MASM HRES) to generate a full listing (HRES.LST). This material is mainly of interest to assembly language programmers who will find the source listing an excellent starting point for adapting or customizing routines to special requirements. The process for generating a BLOADable HRES.BIN file is described in the PCF-HRES Manual.

5.9 A NOTE ON EXECUTION TIMES - COMPILED BASIC

The throughput of the DAS-HRES A/D converter is a function of many interactions but the two fundamental limitations are the speed of the A/D and the settling time of the programmable gain amplifier. These limitations will usually be evident only when using high speeds in DMA MODE 6. Typical throughputs that can be obtained using MODE 6 are

GAIN	UNIPOLAR	BIPOLAR	THROUGHPUT
1	0 to +10V	±10v	47.6KHz
2	0 to +5V	±5v	47.6KHz
4	0 to +2.5V	±2.5V	47.6KHz
8	0 to +1.25V	±1.25V	47.6KHz

NOTE: Actual maximum speeds will vary with computer type and processor clock rate. For guidance, speed slow = up to a few 100 conversions/sec., medium = up to 3000 - 5000 conversions/sec., fast = up to full rated speed of DAS-HRES (47.6 KHz). Fast speeds are obtained using DMA which is not significantly influenced by processor timing and is mainly dependent on the DAS-HRES hardware.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

All other MODEs are software execution time limited to such an extent that the hardware will not impose any limitations. DMA MODE 6 will run at rated speeds regardless of the programming language as it is purely hardware dependent. Once the CALL is entered, all the functions run at assembly language rates and are extremely fast, however other operations in your program processing the data between CALLs may produce bottlenecks and reduce your overall throughput.

One quick fix to improve the speed of an interpreted BASIC program is to compile it using the IBM Basic Compiler, Microsoft Basic Compiler or Microsoft QuickBASIC (registered trademarks of the Microsoft Corporation). When you compile a BASIC program the significance of the HRES in the CALL statement is no longer the same.

```
xxx10  CALL HRES (MD%, D%(0), FLAG%)
```

HRES is not interpreted by the compiler as a variable. It becomes the public name of the subroutine that you wish to call. Before compiling your program, remove lines that BLOAD the HRES.BIN routine and all DEF SEG statements that control the location of the routine. These are not required as the linker will locate the HRES routine in memory automatically. After compiling your program, run the linking session as follows:

```
LINK yourprog.obj + HRES.obj
```

HRES.OBJ is on your distribution disk for this purpose. An example of the effect on performance of compiling a program is supplied in the programs STRIP.BAS and STRIP.EXE. This is a real time "strip chart" emulating program. The BASIC interpreter form, STRIP.BAS, is extremely slow, but when compiled (STRIP.EXE) using the Microsoft Quick Basic compiler, the speed improves to the point of being able to display about 30 readings/second and produces an effective real time display.

5.10 MULTIPLE DAS-HRESs IN ONE SYSTEM

What if you wish to operate more than one DAS-HRES in a system? To avoid conflicts, each DAS-HRES must have a different Base Address and, if interrupts are used, be connected to a different interrupt/DMA level, or if on a common level, each board's interrupt/DMA can only be enabled one at a time. Each board must also be assigned its own CALL routine. To do this start by loading the HRES.BIN routine at different locations in memory, as follows:

```
xxx10  DEF SEG = SG1
xxx20  BLOAD "HRES.BIN",0
xxx30  SG2 = SG1 + 6144/16      'allow 6K for each routine
xxx40  DEF SEG = SG2
xxx50  BLOAD "HRES.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, as follows:

```
yyy10  DEF SEG = SG1
yyy20  CALL HRES1 (MD%, X%, FLAG%)
yyy30  DEF SEG = SG2
yyy40  CALL HRES2 (MD%, X%, FLAG%)   'Etc.
```

■ ■ ■

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 6

PROGRAMMBLE INTERVAL TIMER

6.1 THE 8254 PROGRAMMABLE INTERVAL TIMER

The Intel 8254 programmable interval timer is used in the DAS-HRES. This is a flexible but complex device consisting of three independent 16-bit pre-settable down counters. Each counter can be programmed to divide by any integer in the range 2 - 65,536. As configured in the DAS-HRES, 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 16). The other counter, Counter 0, is uncommitted and its input, output and gate control are available to the user on COUNTER 0 IN (Pin 38), COUNTER 0 OUT (Pin 39) and Counter 0 Gate (Pin 14). A block diagram of the DAS-HRES counter arrangement is shown in the following diagram.

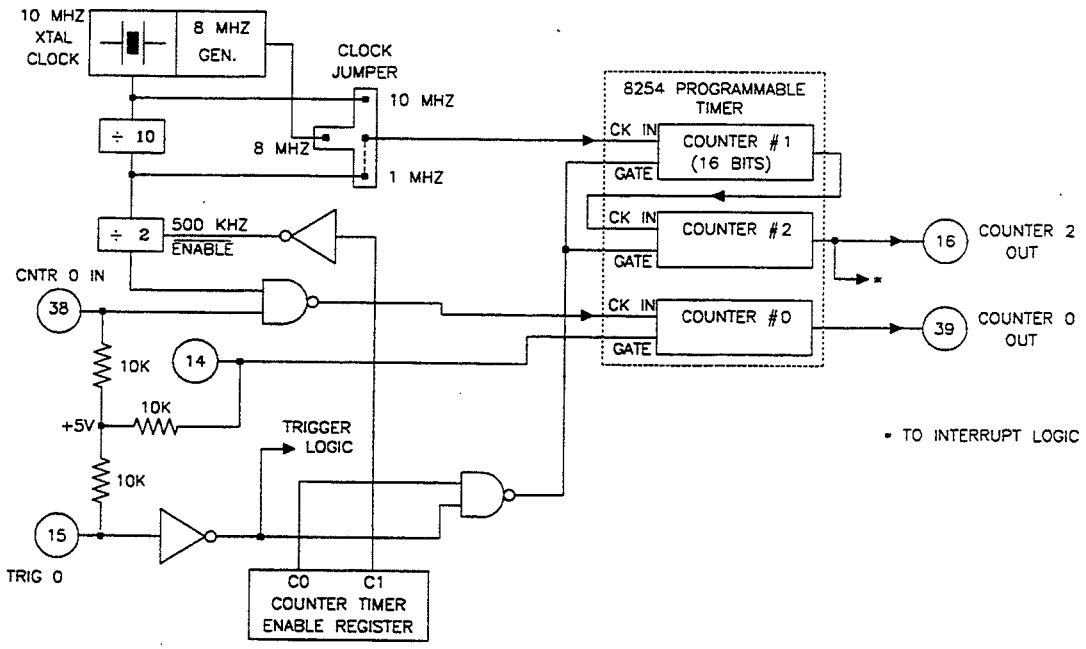


Figure 6-1. Programmable timer configuration.

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 HRES.BIN driver, but can be utilized by the more adventurous programmer. For those interested in detailed information, a full description of the 8254 programmable interval timer can be found in the Intel data sheet (or equivalent manufacturer's

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

literature).

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). It is possible to drive Counters 1 and 2 from a 1 or 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 HRES.BIN 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 follows:

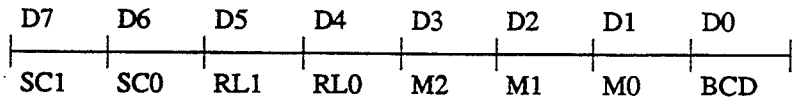
CONFIGURATION	DESCRIPTION
0	PULSE ON TERMINAL COUNT. The output is initially low for this configuration. After the count loads and the counter decrements thru zero, the output goes high and remains high until the counter is reloaded. The counter continues to decrement after passing thru zero and counting 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.
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). This provides a hardware triggered delay or one-shot.
2	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.
3	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 assymmetry). This configuration can be used in the same way as Configuration 2 for periodic triggering or for frequency synthesis.
4	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.
5	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.

The 8254 programmable interval counter occupies four I/O address locations in the DAS-HRES I/O address map:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ADDRESS	REGISTER TYPE	DESCRIPTION
Base Address +12	Read/write	Counter 0
Base Address +13	Read/write	Counter 1
Base Address +14	Read/write	Counter 2
Base Address +15	Write only	Control

Before loading or reading any of the individual counters, the 8254 conrol 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:



SC1-0 - Control which counter is selected.

SC1	SC0	COUNTER
0	0	0
0	1	1
1	0	2
1	1	Readback Command (see next section)

RL1-0 - Control the type of read/load operation.

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 (MSB).

M2-0 - Control counter configuration as above.

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.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

BCD - Controls binary/decimal counting.

BCD	COUNTER TYPE
0	Binary 16 bits
1	Decimal 4 decades

The counters may be programmed to count in binary (modulus 2) or binary coded decimal (modulus 10) modes by the BCD bit. 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.

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 the RL0/1 bits of the control register 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). Users of the IBM PC/AT programming in assembly language are cautioned to put delays (e.g. a short jump) between back-to-back I/O instructions for correct operation. This is due to the design of the IBM PC/AT - see the "IBM PC/AT Technical Reference Manual". This precaution does not apply to the 8088 based PC and PC/XT.

6.2 READING & LOADING THE COUNTERS

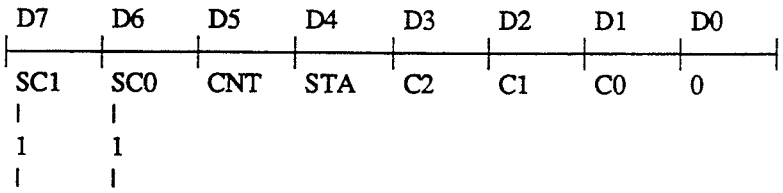
If you attempt to read the counters on the fly with a high input frequency, you will most likely obtain erroneous data. This is partly caused by the rippling of the counter during the read operation and also by the fact that the low and high bytes are read sequentially rather than simultaneously, making 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 - see 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 RL 1/0 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 (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.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Readback command byte format:



Read-back Command

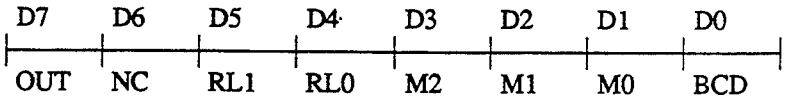
CNT: When 0, latches counters selected by C0 - C2.

STA: When 0, returns status byte of counters selected by C0 - C2.

C0 - C2: When high, select a particular counter for a readback operation:

C0 = 1 selects Counter 0
C1 = 1 selects Counter 1
C2 = 1 selects Counter 2

The status byte returned if STA = 0, consists of:



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.

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 will yield as follows:

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

Some programming examples in BASIC:

```
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 1
xxx80  OUT BASE + 14, XL%
xxx90  OUT BASE + 14, XH%       'load Counter 2
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

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).

The following lines would determine the status of Counter 1:

```
yyy10  OUT BASE + 15, &HE4      'control byte to read status
yyy20  PRINT HEX$(INP(BASE + 13))    'read status
```

The following example shows how to use the uncommitted Counter 0 as a pulse counter. Note that the counters are all "down" counters, so that 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 HRES.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%)
```

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:

```
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

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

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)
```

6.3 TIMER-COUNTER ENABLE REGISTER

The Timer Counter Enable register is a 2-bit Write Only register located at BASE ADDRESS +10. The register format is described in Section 3.8. 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 beginning 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.

6.4 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 HRES.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 2^{32} (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 1KHz. The input frequency to the Counter 0 is 500Khz so you must divide this by 500 to obtain 1Khz. Using MODE 22 of the driver:

```
xxx10  MD% = 22                    'Select MODE 22
xxx20  D%(0) = 500                 'Divider data
xxx30  CALL HRES (MD%, D%(0), FLAG%)
```

Counter 0 output will now be outputting a 1KHz square wave.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

6.5 MEASURING FREQUENCY & 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-HRES 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.

6.6 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. This is the programmable equivalent of a "one shot" or monostable, hence the name.
- *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 the diagram at the end of the chapter.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

6.7 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.

6.8 GENERATING INTERRUPTS WITH THE COUNTER/TIMER

DAS-HRES 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-HRES control register. Indirectly this accomplishes the desired result of generating a periodic interrupt from the timer, and the user can then install any desired interrupt routine to service the interrupt. This is what the HRES.BIN driver does in MODEs 5, 18, and 20; and it provides examples of using the interrupt for servicing the A/D, outputting periodic waveforms on the D/As, and block scanning channels. Since interrupt service routines cannot be programmed directly in BASIC, users interested in customized routines are advised to obtain the DAS-HRES driver source code listing (P/N PCF-16G from the manufacturer) containing commented examples of installing interrupt routines and the routines themselves. It is then a simple task to re-assemble the driver with the Macro-Assembler to create a customized version.

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. ■■■

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



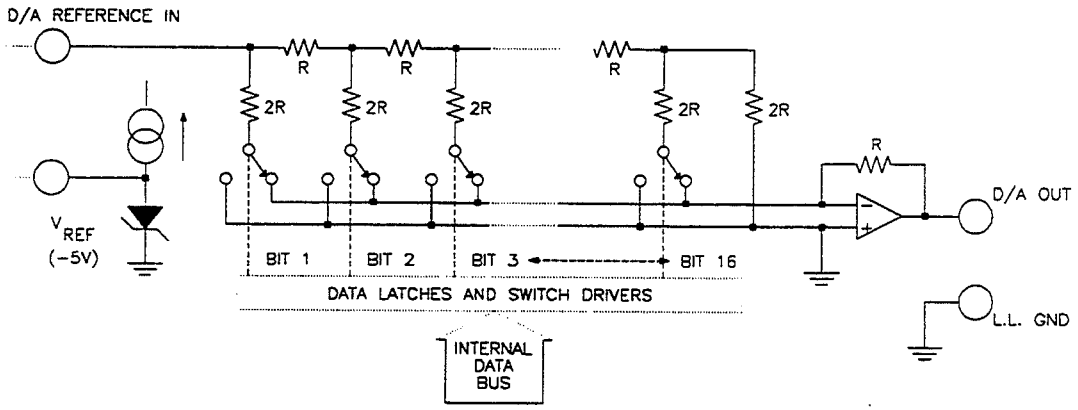
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 7

D/A CONVERTERS

7.1 DESCRIPTION

The D/A channels consist of separate double-buffered bit D/A converters. Each D/A uses an internal reference and provides a ±10 Volt output ranger. A simplified diagram of each D/A channel is shown in the following diagram.



Configuration and connections.

Since the data comprises 16 bits, it has to be written to each D/A in two consecutive bytes. The first byte is the least significant and contains the eight least significant bits of data (see below or Section 3.5 for format). 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* .

The locations of the D/A registers are as follows:

Base Address +4:	D/A #0 - Low byte
Base Address +5:	D/A #0 - High byte
Base Address +6:	D/A #1 - Low byte
Base Address +7:	D/A #1 - High byte

Data format is

Low Byte:

BIT:	D7	D6	D5	D4	D3	D2	D1	D0
Base +4 or +6:	B7	B6	B5	B4	B3	B2	B1	B0
								LSB

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

High Byte:

BIT:	D7	D6	D5	D4	D3	D2	D1	D0
Base +5 or +7:	B15	B14	B13	B12	B11	B10	B9	B8

MSB

7.2 PROGRAMMING

The following example shows how to output data in BASIC. It is readily "translatable" to other languages. Since the D/As have 12-bit resolution, Data D should be in the range -32768 to 32767 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
```

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
```

An assembly language routine is just as simple. Assume AX contains the D/A data and DX has the board I/O address. To write to D/A #0,

```
MOV DX,0304H ;load DX with I/O address
OUT DX,AL    ;write low byte to D/A #0
INC DX
MOV AL,AH
OUT DX,AL    ;write high byte
```

The lengthy routine using BASIC OUTs 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/As as simultaneously as possible (on a standard PC/XT, within 7 microseconds of each other). A typical output routine using the CALL would be

```
xxx10   MD% = 16               'output to both D/A's
xxx20   D%(0) = 1234           'D/A #0 data
xxx30   D%(1) = 22345          'D/A #1 data
xxx40   CALL HRES (MD%, D%(0), FLAG%)
xxx50   IF FLAG%<>0 THEN PRINT "D/A load error"
```

7.3 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

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

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 from interrupt to interrupt 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. EX18.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 detail description of the MODE 18 setup is in Section 4.4 and an example of the operation is provided in EX18.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 sine wave 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 hardware limitation here due to the rate at which the computer can process interrupts:

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 ratio 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.

Using MODE 17 to set the timer,

```
xxx00   MD% = 17                'set timer rate
xxx10   D%(0) = 3               'counter 2 divider
xxx20   D%(1) = 1111            'counter 1 divider
xxx30   CALL HRES (MD%, D%(0), FLAG%)
```

Next we need to load an integer array with the table of output points. Since the array is outputting a 16-bit word to the D/A, we also have to do the scaling in this step. With the D/A output range of ±10 Volts, 0 corresponds to -10.00 volts and 65536 corresponds to the full scale of 9.99998 volts. The following code will dimension a 50 element integer array and load it with a sine wave centered around 0 Volts, with a peak to peak amplitude of 10 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(32767 + 16384 * 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.

```
xxx90MD% = 18                   'MODE number
xx100D%(0) = 0                  'D/A channel, 0 or 1
xx110D%(1) = N                  'number of data points
xx120D%(2) = 0                  'number of waveform cycles, 0 = continuous
xx130D%(3) = VARPTR(DA%(0))     'provide location of data array
xx140D%(4) = 0                  'do not collect A/D data
xx150CALL HRES (MD%, D%(0), FLAG%)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

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 on this important detail see Section 4.4.

■ ■ ■

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 8

GENERATING DATA FILES

After you have performed a data acquisition run, your data will normally be residing in an integer array or memory buffer area. Each data item is a number of bits of A/D output (in the range -32766 to +32767), and the first decision you must make is whether to file the data in this form or turn into real units (for example, volts) and store it in ASCII form.

Filing your data in raw integer form has the advantage of minimizing the amount of disk space required, only two bytes will be needed for each data item. On the other hand, packing data up like this will require a special program to read it and unpack it (you cannot use the DOS TYPE command) and will not make it easily transportable into popular analysis and graphing programs such as Lotus 1-2-3. Most spreadsheet and database programs can import ASCII files. Let's look at some BASIC examples of generating files and storing data in various ways. Assume you have been using some of the example programs and the HRES.BIN driver and you now have your A/D data in an integer array X%(N).

If minimizing disk storage space is your goal, the best way to file the array is to turn it into a random access file. This lets us retrieve any item of data at random. Note the convention for numbering array elements is from 0 to N, whereas file records run from 1 to N (0 is illegal). You get round this with some minor fix-up in the program, as shown, and this is what you would need to do in BASIC:

```
xxx00 OPEN "myfile.dat" AS #1 LEN = 2    'open random access file
xxx10 FIELD #1, 2 AS D$                  '2 bytes per integer
xxx20 FOR I = 1 TO N+1                   'file records start at 1
xxx30 LSET D$ = MKI$(X%(I-1))            'convert & move to buffer
xxx40 PUT #1,I                           'write to file
xxx50 NEXT I
xxx60 CLOSE #1                           'write last buffer
```

The data file created can be read back into an integer array by

```
yyy00 OPEN "myfile.dat" AS #1 LEN = 2    'open same file
yyy10 FIELD #1, 2 AS D$                  'same field definition
yyy20 N = 999                            'array size to suit
yyy30 DIM X%(N)                          'declare array
yyy40 FOR I = 1 TO N+1
yyy50 GET #1,I                           'fetch a record
yyy60 X%(I-1) = CVI(D$)                  'convert 2 bytes to integer
yyy70 NEXT I
yyy80 CLOSE #1                           'close file]
```

This program looks after storing and retrieving data in its most compact form. What if you want to generate a file that can be "imported" into Lotus? Most spreadsheet and database files can read an ASCII comma or space delimited file. Lotus 1-2-3 will do this by using the File/Import/Numbers function from the worksheet menu. From BASIC's point of view, this is a sequential file with each data item separated by a comma to delimit the cell rows and a carriage return to delimit the columns. Data in this form takes a lot more space (-2.174), will take 7 bytes, one byte for each character. For instance the data

1.23, -2.45, 4.86, 5.11
0.24, -1.00, -2.00, 5.99

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

would be imported into Lotus as:

	A	B	C	D	E	etc....
1	1.23	-2.45	4.86	5.11		
2	0.24	-1.00	-2.00	5.99		
etc.						

Starting with a BASIC integer array X%(N) of data, this is how you would generate an ASCII sequential file for the above example. Note that Lotus and many other spreadsheets use a .PRN extension as standard for this type of file:

```
zzz00  OPEN "myfile.PRN" FOR OUTPUT AS #1     'open sequential file
zzz10  SCAN = 4                               'scan length
zzz20  FOR I = 0 TO N
zzz30  X = 5 * X%(I)/32768 'turn into real units (volts on the ±5V scale).
zzz40 IF (I<>0 AND I/SCAN = INT(I/SCAN)) THEN PRINT #1, CHR$(13)
zzz50 PRINT #1, USING "##.####,";X%(I);
zzz60 NEXT I
zzz70 CLOSE #1                                'close file]
```

Line zzz10 specifies the scan length or number of columns in the spreadsheet or entries per line of myfile.PRN. In practice you might devote each column to a channel and the rows would correspond to time so that the intervals that you placed carriage returns in the data would depend on the DAS-HRES scan length, the SCAN variable controls this.

The next step in line zzz30 is to turn the data into real units. You can pass it as bits if you want, but there is no significant space saving in an ASCII file as each character will take 1 byte so that -2048 will take 5 bytes and -5.000 will take 6 bytes. You might just as well have your data in easily understandable units in this case.]

Line zzz40 inserts a carriage return/line feed each time the number of data items becomes an exact multiple of the scan length. Line zzz50 adds the data. Note that the PRINT USING statement is used which provides a fixed format template for the data. This has 2 benefits, one is that each data item will be of fixed length and use a fixed number of bytes for storage, and secondly it makes the data file more readable, especially when using DOS's TYPE command. Incidentally spaces work as well as commas in delimiting data and may be a neater looking alternative if you just want to view your files using the DOS type command or feed them into a word processor. Just change the comma in the PRINT USING template of line zzz50 to a space. It is easy to embellish on the above program to add the time etc. to the beginning of each line if you know the scanning interval e.g. change line zzz40 to:

```
zzz40  IF (I<>0 AND I/SCAN=INT(I/SCAN)) THEN PRINT #1,CHR$(13):PRINT #1,
I*SI;",";
```

where SI = scan interval]

If you want to read this file back in a BASIC program, open the file FOR INPUT and use the INPUT #1, A,B,C,D etc. function. Your BASIC Reference Manual will provide further information on random and sequential files.

■ ■ ■

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 9

TEST & CALIBRATION

9.1 OVERVIEW

Periodic recalibration of DAS-HRES 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 3-month recalibration interval is recommended. For laboratory or office conditions, six months to one year is acceptable.

A 5 1/2 digit digital multimeter is required as a minimum equipment to perform a satisfactory calibration. 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, such as Vector part # 3690-22 (Available from Vector dealers or Vector Electronic Co. Inc., Box 4336, Sylmar, CA 91342. Phone: (213)-365-9661) is an inexpensive device that you will find greatly improves accessibility to the board during calibration and will probably be useful with other boards too.

Calibration is performed using the CALHRES.BAS or CALHRES.EXE programs (simply type BASICA CALHRES or CALHRES after the DOS prompt). This program will lead you through the calibration and set up procedure with a variety of prompts and graphic displays showing you settings and adjustment trimpots.

The calibration program also serves as a useful test of the DAS-HRES A/D & D/A functions and can aid in trouble shooting if problems arise. Simple tests are included for the digital I/O and programmable timer.

9.2 INTERIM CALIBRATION

The hardware consists of a +10 volt nominal reference whose output is divided down by a bulk metal foil resistor network. The voltages produced by this divider network are highly stable. In fact these reference voltages are more stable than the gain setting resistors used in the DAS-HRES front ent. The "RECAL" program provides a means of correcting for drift of gain setting components (based on the extreme stability of the reference circuitry. This is a stand alone program that reads the voltages stored in the HRES.CFG file and re-calculates the calibration constants. This program provides a means of recalibrating the DAS-HRES without using a voltmeter. The assumption used is that gain setting components will drift more than the internal reference signals.

■ ■ ■

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 10

APPLICATIONS

10.1 CONNECTING ANALOG A/D INPUTS CORRECTLY

All DAS-HRES channels operate in differential input configurations. Unlike many lower resolution A/D boards (for example, the DAS-16) the DAS-HRES has no provision for single ended operation. The high accuracy of the DAS-HRES would be all but wasted in a single-ended (common-ground) configuration.

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-HRES 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 10-1) forms a common-mode voltage (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-HRES, the inputs are designed to handle a common-mode limit of ±10 volts. Figure 10-1 shows the correct way to connect a grounded signal source to the DAS-HRES. Figure 10-2 shows the incorrect way to connect the DAS-HRES to a grounded signal source. Figure 10-3 shows the correct connection scheme for an isolated signal source.

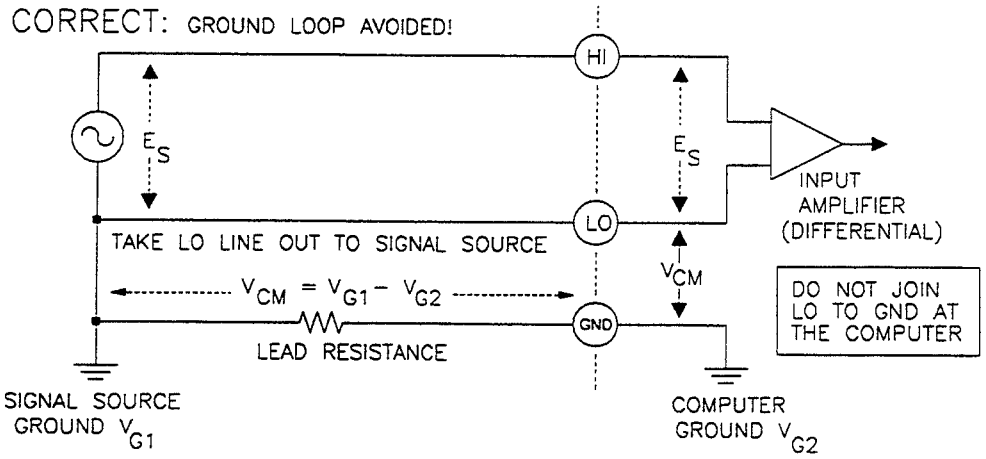


Figure 10-1. Correct method of connecting a grounded signal source.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

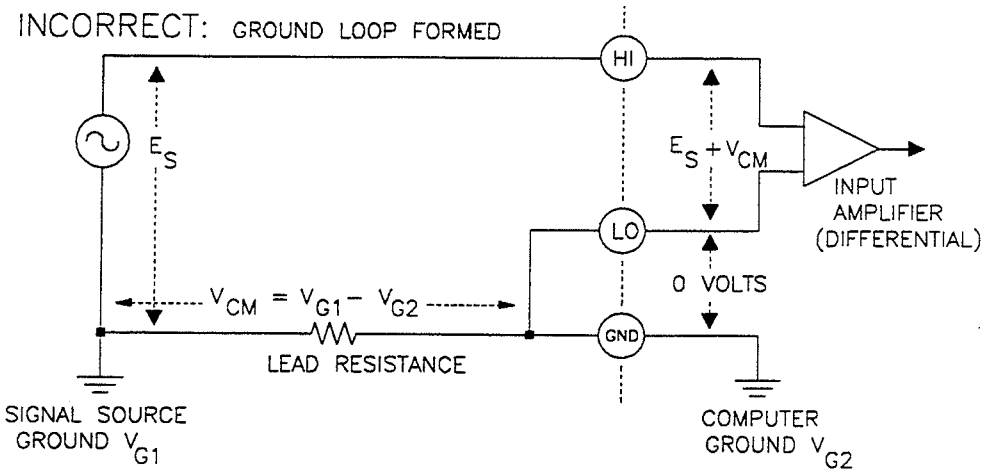


Figure 10-2. Incorrect connection to a grounded signal source.

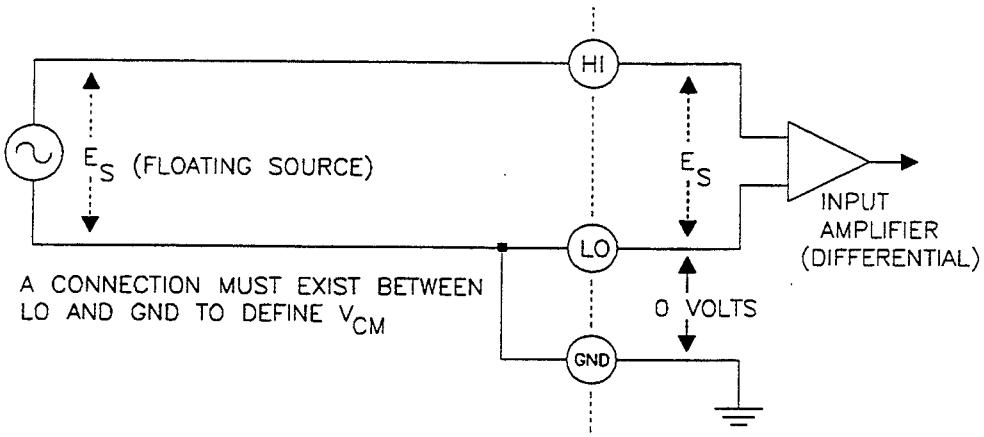


Figure 10-3. Correct connection for an isolated signal source.

If you have a combination of floating and ground referred signal sources, the differential input configuration should be used and the ground referred signals connected as shown in Figure 10-1 and the floating signals connected as shown in Figure 10-3. Note that the jumper between LO and LL GND effectively turns the differential input into a single-ended input.

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 commonest difficulty that users experience in applying and obtaining the best performance from data acquisition systems.

10.2 SELECTING INPUT CHANNEL RANGE & GAIN

The slide switch marked "A/D" has two positions: *UNI* for Unipolar and *BIP* for Bipolar. This switch controls whether the range is Unipolar (zero to some positive full scale) or Bipolar (from a negative to a positive full scale).

The HRES.BIN driver is able to detect the position of the UNI/BIP Switch through the DAS-HRES Status Register, and adjusts the data for Unipolar and Bipolar ranges accordingly. Unipolar scalings correspond to 0 to 65535 bits of output from the A/D, whereas Bipolar scalings correspond to -32768

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

to +32767 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 (e.g. on the ±5V or 0-10V range, multiply the integer data returned by the CALL by the actual bit weight, 305 microvolts/bit, to obtain volts.

10.3 MATCHING THE INPUT CLOCK FREQUENCY TO THE APPLICATION

The DAS-HRES ADC section may be optimized for a particular application by choice of the proper sampling rate and of the correct clock frequency for the internal timer. The DAS-HRES uses a sampling ADC (Crystal Semiconductor CS5016) whose logic operates from a 4MHz clock. This clock will couple into the analog input and show up as single tone in the output data. **The magnitude of this signal is low (RMS value of < 0.5 LSB,peak to peak of 2-3 counts).** This "clock feedthrough" is simple to filter out since it is very easy to predict its frequency from the following relationship.

Tone Frequency = (N * (Sampler Frequency) - (4 Mhz)

where

N = (4 Mhz / (Sampler Frequency)) Rounded to the nearest integer.

Note that this relationship would predict that if integer sub multiples of the clock were used as sampler clock then the tone would end up at 0Hz (DC). So it would seem that the clock feedthough would show up as DC offset in this case and merely be calibrated out with all the other offsets ,but this not the case. It is true that the feedthough ends up at DC but the magnitude of the feedthough varies from sample to sample and thus looks like very low frequency noise (drift). This would actually be desirable in the case of dynamic signal analysis since the only very low frequency portion of the spectrum (<3-4 Hz) would be corrupted by clock feedthrough errors, but would be highly undesirable in the case of DC measurement.

The DAS-HRES addresses both of these concerns by providing a choice of 10, 8, or 1Mhz clock to run the counter timer. The 4Mhz ADC clock is derived directly from the 8Mhz clock. If the 8Mhz clock is chosen therefore, virtually all timer driven ADC sampling frequencies will be integer sub multiples of the ADC clock, producing data useful for dynamic signal analysis. The 10 and 1Mhz clock are not directly related to the ADC clock and can be used to produce sampling frequencies that will shift the tone to high frequency that is easily removed by averaging. Note that the 8254 can still produce rates that will put the tone at low frequency even with the 10 or 1 Mhz clock so one must be careful to chose sampling rates that maximize the tone. A good choice of sampling rates for DC measurement is one that produces a tone at about 9/10ths of the nyquist rate. Note that if you choose to near the nyquist rate you may actually end up with a tone at low frequency since there is some error in the internal clocks.

For best results, several sampling/timing considerations can be made in regards to setup. The choice of 8254 clock frequency plays a role as mentioned above, as does gain selection and sampling rate.

1. When using the DAS-HRES for frequency domain analysis.
   A. Choose a sampling frequency which is an even submultiple of the 4 MHz clock used by the A/D converter. This can by achieved by using the 8Mhz clock or by careful choice of sampling frequency.
   B. Do not use Gain Of 1 range.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2. For DC measurement, use the 10MHz clock and choose a frequency that puts the clock feedthrough at some high frequency. In this way the feedthrough will average out to zero when the DC value is calculated.
3. For general purpose work,
   A. Use the 10Mhz clock.
   B. Remember, even though much has been said about clock feedthrough error, its magnitude is very small and can be ignored in many cases.

10.4 MEASURING CURRENT & 4-20MA CURRENT LOOPS

Process-control current-loop transducers are easily interfaced to DAS-HRES 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 wire wound 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, over 49,000 bits for Unipolar ranges and 24,500 bits for Bipolar ranges.

Ground referred currents can also be measured with a suitable shunt. Nonground referred currents can be measured through an isolation amplifier or DC or Hall Effect current transformer.

10.5 ADDING MORE ANALOG INPUTS

You may add submultiplexers to any or all of the 8 analog inputs. The EXP-20 provides 16 channels per input plus a common high-gain instrumentation amplifier. Up to 8 EXP-20's can be added to one DAS-HRES providing a total of 256 channels. The submultiplexer 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 submultiplex channels, etc.

NOTE: The standard EXP-20 boards and most expansion boards do not provide accuracy comparable to the DAS-HRES's 16 bits. They should only be used when the accuracy can be traded for additional input channels.

When a submultiplexer is added, it is not possible to operate the DAS-HRES in the fast DMA sampling mode due to having to drive the submultiplexer 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-20. The EXP-20 is best suited to handling high gain low rate of change inputs such as thermocouples, pressure transducers, etc. EXP-20 includes cold junction compensating circuitry for thermocouples, switch selectable gain from 0.5 to 1000 and 16 differential inputs.

10.6 INTERFACE TO TRANSDUCERS, THERMOCOUPLES, ETC.

Low level transducers such as thermocouples and strain gage bridges (load cells, pressure, and force transducers) output very small signals (for example, 40 microvolts/°C for a thermocouple). The DAS-HRES can handle these directly although the results will not be as satisfactory as amplifying these signals at the source.

Although it is a convenience and an industry standard practice to provide high gains and resolutions

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

internal to A/D boards, it is always a compromise and never leads to as good results as amplifying your signal at its source. The laws of physics take over as far as noise performance is concerned and although the subject of designing low-noise systems is complex, you should be aware in a general way of these limitations, and that all manufacturers of high gain A/D boards face the same difficulties which arise from several sources:

1. Induced noise in cabling of low level signals (magnetic and capacitative coupling). Shielded and twisted pair cable can help.
2. Thermal noise from the on resistance of the multiplexer and signal source resistance ahead of the programmable gain amplifier. Keep signal source resistances as low as possible.
3. Offsets and leakages in the multiplexer which vary from channel to channel and with temperature. Avoid operation at high temperatures if possible, also keeping signal source resistances to a minimum also helps.
4. System noise internal to the computer from switching power supplies, rf, and digital signal crosstalk. A working compromise is made with all plug in boards that include analog circuitry, usually at the 12-bit level the effects from the computer are minimal unless high gains are used.
5. Ground loop and common mode noise errors generally become of greater magnitude the further you get away from the signal source. Try to set up the system so as to minimize cable length of the signal sources.
6. Noise performance of the amplifier. Wide bandwidth amplifiers with short settling times that have to be used in a multiplexed system always have a greater noise bandwidth than limited bandwidth amplifiers that usually can be used to amplify signals individually at source.

You can often obtain adequate performance depending on your application in which case providing gain on the A/D board is cost effective and satisfactory. If you find that performance does not fit your needs, many other signal conditioning solutions are available from the manufacturer.

For inexpensive temperature measurement@Index{Semiconductor temperature sensors} in the -50 to +125 deg. 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 273uA at 0° C. and a scaling of 1uA/° 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, 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-20 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 only required at the highest levels of accuracy, since in most cases connector temperature (usually room temperature) varies little. EXP-20 provides the sensing hardware needed to perform this correction.

10.7 PRECAUTIONS IN USE: NOISE, GROUNDLOOPS, & 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

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

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 A.C. 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-HRES board.

■ ■ ■

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 11

FACTORY RETURNS

Before returning any equipment for repair, please call 508/880-3000 to notify MetraByte's technical service personnel. If possible, a technical representative will diagnose and resolve your problem by telephone. If a telephone resolution is not possible, the technical representative will issue you a Return Material Authorization (RMA) number and ask you to return the equipment. Please reference the RMA number in any documentation regarding the equipment and on the outside of the shipping container.

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

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

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

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

Ship the equipment 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 the RMA number on the outside of the package!

■ ■ ■

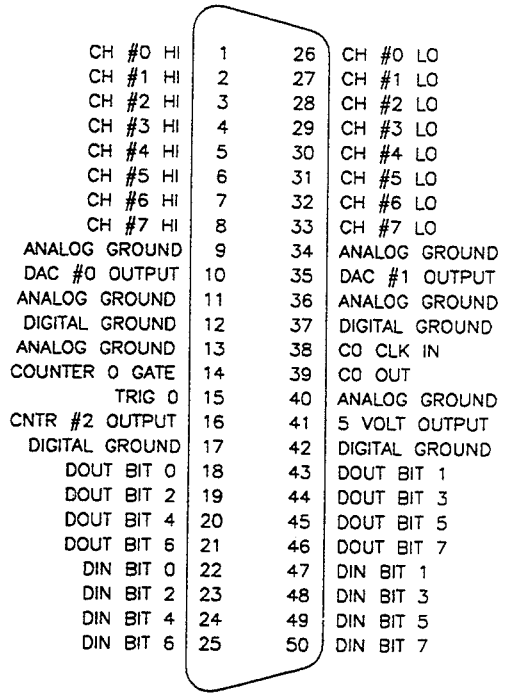
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

APPENDIX A

I/O CONNECTIONS

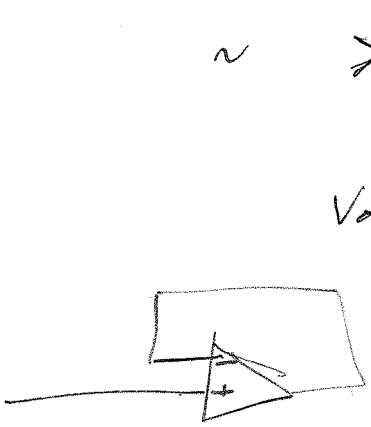
The main analog and digital I/O is via a 50-pin connector at the rear of the Board. The connector is accessible at the rear panel of the computer. Connector pin functions are as shown in the diagram.



Rear view of I/O connector.

■ ■ ■

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

APPENDIX B

SPECIFICATIONS

B.1 POWER CONSUMPTION

+5v supply	1.4 A typ; 1.9A max.

B.2 ANALOG INPUT SPECIFICATIONS

General

Channels	Eight Differential (high/low/ground)
Resolution	16 Bits.
Relative Accuracy	0.003 % of full scale range ±1 bit.
Absolute Accuracy	Gains 1, 2, 4: 0.003 % of full scale range ±1 bit; Gain of 8: 0.0045% of full scale range ±1 bit.
Accuracy with Software Autocalibration Disabled	±0.01% of FSR, ±1bit.
Differential Nonlinearity	±1 LSB.
Warm-up Time	±45 minutes.

Noise Performance

RMS LEVEL OF NOISE

	Sampling Rate Integer Submultiple Of Clock		Sampling Rate Noninteger Submultiple Of Clock	
GAIN	**BIPOLAR**	**UNIPOLAR**	**BIPOLAR**	**UNIPOLAR**
1	0.8 LSB	1.0 LSB	1.0 LSB	1.5 LSB
2	0.8 LSB	1.0 LSB	1.0 LSB	1.5 LSB
4	0.85 LSB	1.1 LSB	1.0 LSB	1.6 LSB
8	0.90 LSB	1.20 LSB	1.0 LSB	1.6 LSB

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PEAK-TO-PEAK LEVEL OF NOISE

	Sampling Rate Integer Submultiple Of Clock		Sampling Rate Noninteger Submultiple Of Clock	
GAIN	**BIPOLAR**	**UNIPOLAR**	**BIPOLAR**	**UNIPOLAR**
1	±2.5 LSB	±3.0 LSB	±3.0 LSB	±4.5 LSB
2	±2.5 LSB	±3.5 LSB	±3.5 LSB	±4.5 LSB
4	±3.0 LSB	±3.5 LSB	±3.5 LSB	±5.0 LSB
8	±3.0 LSB	±4.5 LSB	±3.5 LSB	±6.0 LSB

Aperture jitter	100 picoseconds.
Programmable amplifier settling time	20 microseconds to .003 % for full scale step input.
Input Current	±10 nA at 25° Celsius.
Overvoltage protection	Continuous single-channel to ±35 volts.
Temperature coefficient of input amplifier	Gain ±15 ppm of full scale per degree Celsius.
Offset	±3 ppm of full scale per degree Celsius.
Temperature coefficient of internal references	±5 ppm of full scale per degree Celsius.

B.3 DIGITAL-TO-ANALOG CONVERTERS

Type	16-bit double buffered.
Integral linearity (relative accuracy)	±0.006 % of range.
Monotonacity	14 bits.
Output range	±10 Volts.
Coding	2's Complement.
Output drive current	±5.0 mA.
Output resistance	0.15 ohm
Short circuit to ground	Indefinite duration.
Settling time	180 microseconds to 0.006 %.

B.4 DIGITAL I/O

Output Port	8 TTL lines.
OP0-7 Output (Low Voltage)	0.5V max at Isink = 8.0mA.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

OP0-7 Output (High Voltage)	2.4V min at Isource = -0.4mA.
Input Port	8 TTL input lines,Pulled up to +5 VDC via 10K ohm resistors
IP0-3 Input (Low Voltage)	0.8V max.
IP0-3 Input (Low Current)	-0.4mA max.
IP0-3 Input (High Voltage)	2.0V min.
IP1-3 Input (High Current)	20uA max. @ 2.7V.

B.5 INTERRUPT CHANNEL

Level	2 - 7 software selectable.
Enable	Via INTE of CONTROL register.

Interrupts are latched in an internal flip-flop on the DAS-HRES board. The state of this flip-flop corresponds to the INT bit in the Status Register. The flip-flop is cleared by a write to the Status Register. Service routines should acknowledge and re-enable the interrupt flop.

B.6 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	Hardware capable of 150,000 conversions/sec.

Users are responsible for initialization of the 8237 DMA controller on the system board. With the 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 the end of the following conversion. Transfer sequence is low byte/high byte.

B.7 PROGRAMMABLE TIMER

Type	8254-2 programmable interval timer.
Counters	Three down counters, 16 bit. Two permanently connected to 1 or 10MHz clock as programmable timer. One counter free.
Output Drive Capability	2.2mA @ 0.45V (5 LSTTL loads).
Input, Gate	TTL/DTL/CMOS compatible.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Clock Input Frequency	DC to 10MHz.
Active Count	Negative edge.
Min. Clock Pulse Widths	30nS high / 50nS low.
Timer Range	2.5MHz to <1 pulse/hr.

For additional information on programming, see Chapter 5.

B.8 GENERAL ENVIRONMENTAL

Operating Temperatures	0 to 50° C.
Storage Temperatures	-20 to +70° C.
Humidity	0 to 90% noncondensing.
Weight	14 oz. (270 gm).

■ ■ ■

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

APPENDIX C

STORAGE OF INTEGER VARIABLES

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
-32,768	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 8254 interval timer and so 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
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

From integer variable N% to real variable N:

```
xxx20 IF N% >= 0 THEN N=N% ELSE N = N% + 65536
```

■ ■ ■

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

APPENDIX D

REPAIRS & SERVICE

The components that can possibly be damaged by external shorts, transients, overloads etc. are in sockets. These parts are readily serviceable simply by unplugging and replacing with a new part. The manufacturer can service your board quickly and usually recommends that all repairs be made at the factory. However, if you wish to attempt simple repairs for yourself, one of the manufacturer's applications engineers may be able to assist you.

If a Board is located at wrong I/O address, Error Code 22 appears on initialization and usually indicates that the Base Address switch setting does not correspond to the address used in your program. Run CALHRES, check that the HRES.CFG file is correct. If there is no discrepancy, the board has a serious hardware fault and should be returned to the manufacturer for repair.

■ ■ ■

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

APPENDIX E

DIRECT MEMORY ACCESS HARDWARE

E.1 THE IBM PC DMA STRUCTURE

The IBM PC uses an Intel 8237A-5 DMA controller which provides four prioritized direct memory access (DMA) channels. Each channel has two control signals associated with it, a DMA request (DRQ) and a DMA acknowledge (DACK).

DMA provides a direct transfer of data to/from an I/O device from/to memory without passing the data through the C.P.U. (8088) on the way. The advantage of this is a massive improvement in the speed of data transfer. A normal program transfer example in assembly language might be:

```
LOOP:       IN AX,DX          ;read data from I/O port        ( 8)
            MOV [DI],AX       ;transfer to memory at DS:DI    (10)
            CMP DI,LIMIT      ;end of transfer?               ( 4)
            JZ EXIT           ;yes, exit                      ( 4)
            INC DI            ;no, continue                   ( 2)
            INC DI            ;increment memory pointer       ( 2)
            JMP LOOP          ;repeat                         (15)

EXIT:       . . . . . .       ;continue

                                                              -------
                              Total clock periods    =        43
          Loop execution time with 4.77MHz clock     =         9 uS
```

In practice, an A/D service routine would be more involved as it would require checking the status or EOC of the A/D to see if the data was ready etc. and its execution time makes it difficult to handle data conversion rates above 10KHz. Another serious shortcoming of a programmed transfer through the CPU is that is liable to disruption from interrupts, the timer interrupt being a major culprit on the IBM PC.

DMA avoids these problems. The A/D on DAS-HRES is triggered by the programmable timer or an external start pulse, performs a conversion and issues 2 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 there is a minimum of 20 microseconds (A/D conversion time) for the transfer to take place. This is more than enough time, even allowing for 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).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The 8237 handles a total of 4 DMA channels, 3 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 retention and a computer crash.

DMA Level 2 is always used by the floppy disk(s) to read/write data and 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 in its own ROM. A few manufacturers make use of block moves (MOVS), most 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 leaves it free. Running LEV3.EXE performs a check. If you find it is unused, then the DAS-HRES can safely be set to this level.

The DAS-HRES may be operated on either of Levels 1 or 3. The DMA level is selected by the slide switch marked "DMA" above the gold edge connector P1. Since the DAS-HRES 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-HRES, the DMA request (DREQ) line from the board is disabled (tristate). You can therefore share it with other devices on the same level as long as they are not enabled at the same time.

E.2 PAGE REGISTER & DMA CONTROLLER FUNCTIONS

The 8237 controller on the system board was designed in the days of 8 bit CPUs and 64K memories and 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_ADDRESS
0 & 1	1	&H83
2	2	&H81
3	3	&H82

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The 8237 DMA controller contains 4 registers that must also be initialized before a DMA transfer. These are:

Mode Register	Controls type of operation
Base & Current Address	Sets start memory address
Word Count	Sets number of transfers
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 HRES.BIN call routine, but if you intend to write your own driver, it may help to buy the HRES.BIN commented source listing (Part of PCF-HRES) for guidance. In addition to setting up the 8237 and page registers, the DAS-HRES control register must also be loaded to enable the DAS-HRES DMA hardware. Many of the 8237 registers use low byte/high byte I/O operations to the same I/O address, the 8237 contains a first/last flip flop that is toggled on each 16 bit double sequential byte I/O access. To make your routines reliable, be sure to take the following precautions:

1. Insert a delay between sequential accesses to the same I/O address. This is required on the PC/AT and all 286/386 based machines to allow sufficient I/O recovery time. A short jump inserted between instructions is adequate (see PC AT Technical Reference Manual).
2. Always reset the 8237's internal byte pointer flip flop before reading/writing double byte registers. Another program or interrupt may have left the byte pointer flip flop in the wrong state.
3. Before resetting the byte pointer flip-flop and reading or writing to the double byte registers, clear interrupts. Do not re-enable interrupts until the double byte read or write operation is done. This is to prevent interrupts, particularly the DOS disk interrupts and function calls from changing the byte pointer in the middle of a read or write. Putting these precautions together:

```
        CLI             ;clear interrupts
        OUT 0CH,AL      ;clear byte pointer flip-flop
        OUT DX,AL       ;first I/O access - low byte
        MOV AL,AH       ;set up high byte
        JMP LABEL       ;jump for PC/AT I/O delay
LABEL:  OUT DX,AL       ;second I/O access - high byte
        STI             ;re-enable interrupts
```

There are two types of DMA operation provided for in the HRES.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 nonrecycle 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, and data can be transferred from a memory buffer area to a BASIC array using Mode 9.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E.3 USE OF DMA ON OLDER 64K IBM PCs

If you use DMA Level 1 on the very earliest 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 system board hardware that was corrected on later 256K machines. If this error occurs, you will have to turn off the computer and reboot, very unpleasant! 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     & return
```

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 and additionally there is no way of avoiding using DMA Level 1 instead of Level 3, then before running Mode 6 of the DAS-HRES.BIN driver add a line in your program as follows:

```
xxxxx  OUT &HA0,0          'disables NMI
```

This disables the nonmaskable 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.

■ ■ ■

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

APPENDIX F

MEMORY ALLOCATION & DMA BUFFERS

F.1 INTRODUCTION

A method of transferring data directly between an I/O device and memory without passing the data through the CPU is called *Direct Memory Access* (DMA). DMA in the IBM PC family is performed by one or more INTEL 8237 DMA Controllers (or similar circuits for MicroChannel systems) and can occur in the background while the CPU is servicing other instructions. This ability to run independently at high-transfer rates makes DMA an attractive alternative for moving data in data acquisition systems. Many of the manufacturer's Data Acquisition boards support high-speed DMA (including the DAS-16, DAS-HRES, DAS-20, DAS-40, PDMA-16, PDMA-32, and MicroChannel products CDAS-16G and CPDMA-32).

The 8237 DMA Controller uses a different memory addressing scheme than that of the 80x86 CPU. This difference makes DMA applications software a little more complex, especially when allocating memory from DOS for DMA buffer space. Normally the software driver supplied with your data-acquisition board handles all programming of the board and the PC's DMA Controller via a user-supplied 80x86 Address SEGMENT or SEGMENT-OFFSET Pair and desired number of transfers. Because the DMA Controller addresses memory differently than the 80x86, care must be taken when supplying this address to make sure that it is usable by the DMA Controller.

The sections that follow address the problems associated with memory allocation for DMA. Also provided is information about what has been done to simplify the process of reserving memory for DMA, as well as alternative methods that can be used in cases where other methods fail.

F.2 80X86 (REAL-MODE) ADDRESSING

The 80x86 addresses memory with two 16-bit quantities: a SEGMENT and an OFFSET. A SEGMENT and OFFSET pair is combined internally by the CPU to create a 20-bit ABSOLUTE ADDRESS that references a byte somewhere within the 1 million byte (1MB) address space of the 80x86. To obtain an ABSOLUTE ADDRESS in memory, the SEGMENT value is multiplied by 16 (which is the same as shifting left by 4 binary bits), and the OFFSET is added to this number yielding a 20-bit ABSOLUTE ADDRESS. For example if the SEGMENT is 5B30h and the OFFSET is 2605h (usually represented as 5B30:2605) the ABSOLUTE ADDRESS is determined as follows:

```
5 B 3 0  (SEGMENT shifted left by 4)
+ 2 6 0 5 (OFFSET from the SEGMENT base)
---------
5 D 9 0 5 (ABSOLUTE ADDRESS)
```

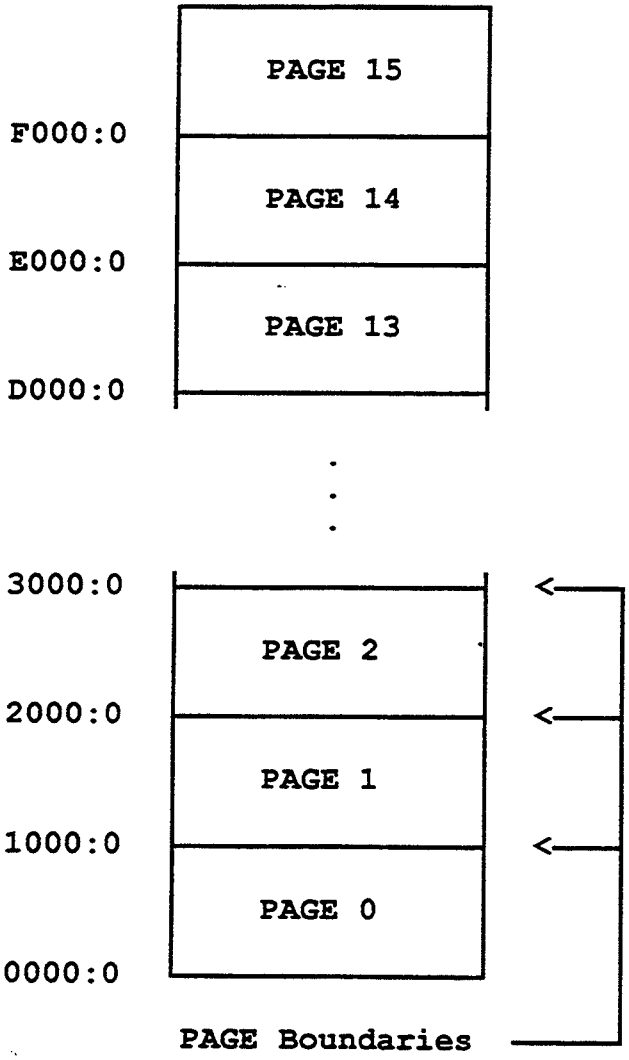
Note that all addresses are in Hexadecimal (h) format unless otherwise denoted.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

F.3 DMA ADDRESSING

In the PC, the 8237 DMA Controller accesses memory on a particular DMA channel via two registers: the PAGE Register and the OFFSET Register. Each DMA channel is assigned one PAGE Register and one OFFSET Register. Although both the PAGE and the OFFSET are held in 16-bit registers, only the least-significant four bits of the PAGE Register are used. Because the PAGE is a 4-bit quantity, there are 16 PAGES of memory (at most) for DMA transfers; these PAGES fall on the first 1MB of PC RAM.

The following memory diagram illustrates division of the first 1MB of RAM into DMA PAGES. As stated below, there are 16 PAGES with PAGE 0 starting at the Address 0:0 and PAGE 15 starting at Address F000:0. Each PAGE is 64KB in length and ends one byte below the next PAGE boundary.



Valid values for the PAGE Register are 0-15. Because the PAGE value represents bits 16-19 in the address for DMA transfers, each possible PAGE value represents a 64KB block in the first 1MB of memory. For example a value of 8 in the PAGE Register selects the ninth DMA PAGE in RAM. The ninth DMA PAGE is the memory ranging from 8000:0 to 8000:FFFF. The value of the PAGE Register is meaningful only when used in conjunction with the OFFSET Register, as explained below.

The OFFSET Register is a 16-bit Read/Write register whose value can range from 0 to FFFF (65535 decimal). This register is added to the PAGE Register to form a 20-bit address as follows:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
 PAGE * 2^16  (Shift Left By 16 Places)
+ OFFSET
-------
= 20 BIT ADDRESS
```

Example:

The user supplies the address 1234:5678

This address is first converted to the 20-bit ABSOLUTE ADDRESS 179B8 as follows:

```
  1234
+ 5678
-------
  179B8
```

From this 20-bit ABSOLUTE ADDRESS, bits 16-19 (1) are written to the 8237 PAGE Register for the current DMA channel, and Bits 0-15 (79B8) are written to the OFFSET Register for that channel.

In addition to the PAGE and OFFSET Registers, a BYTE COUNT Register (also internal to the 8237 Controller) is used. This register is programmed with the total number of transfers to the address as represented by the PAGE and OFFSET Registers.

Once DMA is started and as each byte transfer occurs, the Controller automatically updates the OFFSET (incremented) and BYTE COUNT (decremented) Registers untill the Byte Count value has decremented past 0 to FFFFh.

Because the 8237 Controller can not update its PAGE Register, the maximum number of possible byte transfers is limited to 65,536. This maximum may be achieved in any of 16 possible pages – only when the OFFSET Register is loaded with 0 and the BYTE COUNT Register is loaded with FFFFh. All other combinations of values for the OFFSET and COUNT Registers will cause the transfer of less than the maximum possible and/or cause a possible "page wrap" condition. This condition occurs when (during normal transfer) the OFFSET Register reaches FFFFh and the BYTE COUNT has not reached 0! In such a situation, program and/or data areas may be corrupted.

Consider the following example where a user wants to DMA 12288 (3000h) bytes into a buffer at CPU Address 7000:DCBA.

Example:

The DMA Controller is programmed with PAGE 7, OFFSET DCBAh, and TRANSFER COUNT 3000h. Notice, however, that only 2345h bytes remain between the buffer address (7000:DCBA) and the start of the next DMA PAGE (8000:0). The following steps outline the DMA Controller Registers during such an operation:

The TARGET ADDRESS column denotes the actual CPU Address where the DMA Controller is to write the current byte.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

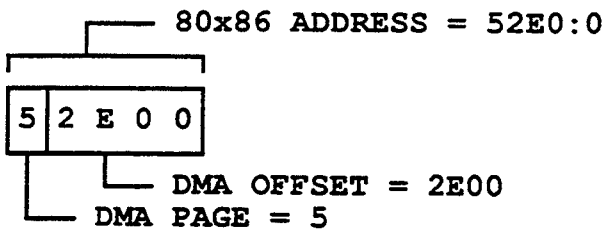
TRANSFER COUNT	PAGE (Hex)	OFFSET (Hex)	TARGET ADDRESS	
3000	7	DCBA	7000:DCBA	
2FFF	7	DCBB	7000:DCBB	
2FFE	7	DCBC	7000:DCBC	
	.			
	.			
	.			
CBA	7	FFFF	7000:FFFF	
CB9	7	0000	7000:0000	<-- *
CB8	7	0001	7000:0001	
	.			
	.			
	.			
0000	7	0CB8	7000:0CB8	

* "Wrap-Around" Point.

Notice that in the above example, all DMA transfers past the "Wrap-Around" point occur in an area of memory outside the user-specified buffer. Destroying the contents of this memory via an incorrect DMA transfer may or may not cause erratic system operation, depending on what code or data is overwritten. For this reason it is imperative to correctly calculate how many bytes remain in the current DMA PAGE prior to programming the 8237 controller.

F.4 ALLOCATING MEMORY FROM DOS

The Software Driver provided with your board performs the actual programming of the DMA Controller and handles the conversion of an 80x86 ADDRESS into a DMA PAGE and OFFSET. Typically, the user passes a SEGMENT value to the DMA MODE (The MODE normally assumes an OFFSET of 0 to complete the 80x86 Address); e.g. 52E0h. The Driver first converts the SEGMENT into the 20-bit ABSOLUTE ADDRESS 52E00 and then extracts the DMA PAGE 5 and OFFSET 2E00. These values are programmed into the DMA Controller by the Driver. The following diagram illustrates the breakdown of an 80x86 SEGMENT and OFFSET into a DMA PAGE and OFFSET.



To prevent a "Wrap Around," the manufacturer's Software Drivers calculate the ending OFFSET corresponding to the BYTE COUNT and starting OFFSET to be sent to the controller. If the required BYTE COUNT causes the ending OFFSET within the DMA Controller to exceed FFFF, a WRAP AROUND ERROR is returned and the Driver does not program the controller.

When DOS is called upon to allocate a block of memory to an application, it will allocate the first block of memory large enough to fill the request. Because of this "first-fit" strategy there is no easy way to

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

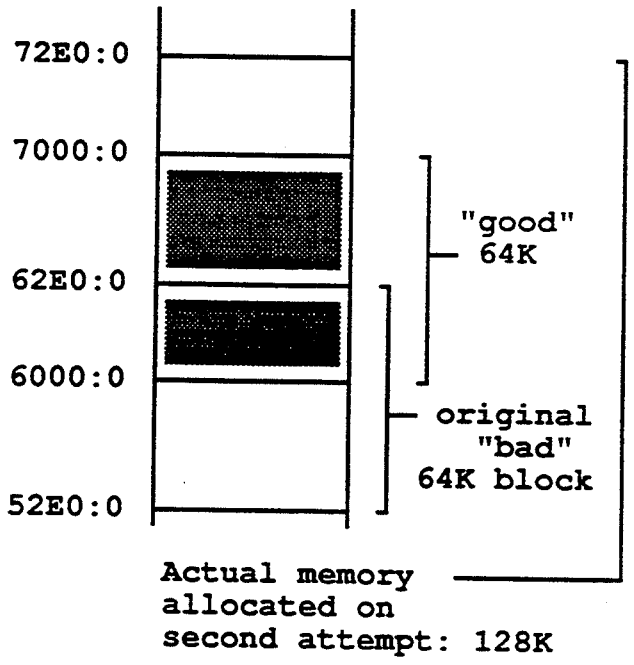
insure that all of an allocated block of memory will be in the same DMA PAGE. The process of transferring a given number of bytes to or from an allocated block of memory will be successful only if the required number of bytes will all fit in the same DMA PAGE.

The latest versions of Software Drivers for the manufacturer's boards that perform DMA transfers support a MODE Call that allocates a "good" block of memory for DMA transfer (when sufficient memory exists). This MODE works from most languages and can be used to allocate a DMA buffer before calling the DMA Transfer MODE. A table is provided at the end of the following section to indicate which versions of drivers contain this memory allocation MODE.

F.5 SOLUTION

The objective of this section is to explain the method used in the manufacturer's software drivers to allocate a suitable block of memory for DMA transfer. The "MEMORY ALLOCATION FOR DMA TRANSFER" MODES, supplied in the software drivers, will work in the majority of programming languages (including C, Pascal, FORTRAN, and QuickBASIC) that are compiled from the DOS command line. The same method used in the driver may work for other languages according to the following guidelines.

The first step in allocating memory suitable for DMA transfer is to allocate the number of bytes that are actually needed. In a DOS environment, allocation and deallocation of memory is handled through INTERRUPT 21h FUNCTION 48h (Allocate Memory Block) and FUNCTION 49h (Release Memory Block). After allocating this initial block of memory a check is performed to see if the entire block falls in the same DMA PAGE. If the block is completely in one PAGE (ie: the sum of the BYTE COUNT and the DMA OFFSET is less than or equal to FFFFh) then no further action is necessary and this block's address can be passed to the DMA MODE of the Driver. If, however, the block of memory is not all on the same DMA PAGE, extra steps are necessary. The first of these steps is to release the previously allocated memory. Once this memory is released, twice as much memory as actually needed is allocated. Upon successful completion of this second allocation, a suitable block of memory has been obtained and all that is left to do is to determine the OFFSET within this block to be sent to the DMA MODE.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The above diagram illustrates the memory usage for this allocation scheme using a desired BYTE COUNT of 64KB. It was determined that the first allocated block from 52E0:0 to 62E0:0 crossed a PAGE boundary at 6000:0. After the initial block was released, twice as much was allocated and the resulting block of memory started at 52E0:0 and ended at 72E0:0. The new memory block crosses two PAGE boundaries (6000:0 and 7000:0) and there is now a complete PAGE of memory that can be accessed by the DMA Controller. The SEGMENT 6000 can be sent to the DMA MODE of the Driver and up to 64KB can be transferred.

Using this method to allocate DMA buffers will work for buffer sizes up to 64KB as long as there is a sufficiently large block of memory available in your system.

The sections that follow demonstrate how to set up for and make the calls to the appropriate memory allocation MODES from different programming languages. For simplicity the DAS-HRES has been chosen for illustration but other boards from the manufacturer are programmed similarly. When programming language specifics make other modifications necessary, they will be included in the example code and/or description. Some programming languages supported by the manufacturer are not compatible with the memory allocation MODES for various reasons; alternative solutions will be provided for these cases as well.

The following table has been provided to indicate the versions of the manufacturer's software drivers that include MODES for allocating and deallocating DMA Buffers.

PRODUCT	VERSION WITH ALLOCATE & DEALLOCATE	ALLOCATE MODE	DEALLOCATE MODE
PDMA-16	2.00	9	10
PDMA-32	2.00	9	10
DAS-16 (all)	4.20	23	24
DAS-20	*	*	*
DAS-40	*	*	*
DAS-HRES	1.10	23	24
uCDAS-16G	1.20	23	24
uCPDMA-32	1.00	9	10

* The Drivers for these products do not contain the memory allocation MODES at the release of this document, the MODES will be incorporated in future versions.

For the DAS-HRES, the MODE descriptions are as follows:

MODE 23: Allocate Memory For DMA

This MODE requires entry of the number of bytes or words to allocate for DMA transfers. MODE 23 calls the DOS Interrupt 21H Function 48H to allocate the desired number of bytes. If there is sufficient memory, MODE 23 returns the SEGMENT of a suitable block of memory for DMA transfer. If there is not enough memory, an error code is returned.

In addition to the SEGMENT returned for DMA transfer, two other parameters are returned; the actual SEGMENT address of the allocated memory block (for use when releasing the memory), and the DMA OFFSET, which is always 0.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MODE 23 also requires the calling routine to specify whether to allocate memory for byte or word transfers of the count specified.

Entry Parameters: Params[0] = number of transfers.

Params[1] = 0 (bytes) or 1 (words).

Return Parameters: Params[5] = "good" SEGMENT for DMA.

Params[6] = DMA Offset (always 0).

Params[7] = actual SEGMENT allocated by DOS (should be passed to MODE 24).

NOTE: Although the SEGMENT Address of the allocated memory may not be the ADDRESS sent to the DMA MODE it should be saved, as it is needed when DMA is complete and the block is released back to DOS.

MODE 24: Deallocate Memory

MODE 24 uses the DOS Interrupt 21h Function 49h to release a memory block previously allocated using MODE 23. This MODE should be called after the program is finished with the DMA buffer. When using this MODE, be sure to pass the actual SEGMENT that was allocated and not the DMA SEGMENT.

Entry Parameters: Params[0] = SEGMENT to be deallocated.

Return Parameters: None.

F.6 LANGUAGE SPECIFIC SOLUTIONS & EXAMPLES

The examples in this section use the MODE calls for the DAS-HRES Driver. The MODES used are, MODE 23: ALLOCATE MEMORY FOR DMA, MODE 6: SET UP AND PERFORM DMA, and MODE 24: DEALLOCATE MEMORY. If you are using a card other than the DAS-HRES slight changes in MODE calls will be necessary as the MODES are not the same for all the manufacturer's boards.

Microsoft C

When using Version 4.x, 5.x, or 6.0 of the Microsoft C Compiler, use DAS-HRES MODE 23 to allocate memory for DMA and DAS-HRES; use MODE 24 to deallocate memory after DMA is finished and the memory is no longer needed.

An example of Microsoft C follows.

Example:

```
/* Allocate Suitable Block of Memory for DMA */

mode = 23                                   /* allocate memory for DMA */
params[0] = 40000                           /* allocate 40000 words */
params[1] = 1                                  /* allocate words */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
mscm_hres(&mode,params,&flag)        /* MICROSOFT C medium model MODE call
*/

if (flag !=0)                           /* Error if flag is not zero */
{
   printf("\n\nMODE 23 Error Flag = %d\n",flag);
   exit(1);
}

actual_seg = params[7]                  /* Save actual Segment for later
                                        deallocation with MODE 24 */

dma_seg = params[5]                  /* "good" segment for DMA MODE 6 */

/* Set Up for Call to MODE 6: Set Up and Perform DMA Transfer */
mode = 6                             /* Set Up and Perform DMA */
params[0] = 40000                    /* number of words */
params[1] = dma_seg
params[2] = 1                           /* trigger source */
params[3] = 0                           /* single cycle/recycle */
params[4] = 0                           /* gain */
mscm_hres(&mode,params,&flag)        /* MICROSOFT C medium model MODE call
*/

if (flag !=0)                           /* Error if flag is not zero */
{
   printf("\n\nMODE 6 Error Flag = %d\n",flag);
   exit(1);
}
     .
     .
     .

/* Release Memory Back to DOS when Finished */
mode = 24                            /* deallocate memory */
params[0] = actual_seg                  /* Segment to release */
mscm_hres(&mode,params,&flag)        /* call driver */

if (flag !=0)                           /* Error if flag is not zero */
{
   printf("\n\nMODE 24 Error Flag = %d\n",Flag);
   exit(1);
}
```

Borland C / Turbo C

With Version 1.x or 2.0 of the Borland / Turbo C Compiler, use DAS-HRES MODE 23 to allocate memory for DMA and DAS-HRES MODE 24 to deallocate the memory when DMA is finished and the memory is no longer needed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example:

```
/* Allocate Suitable Block of Memory for DMA */

mode = 23                                   /* allocate memory for DMA */
params[0] = 40000                              /* allocate 40000 words */
params[1] = 1                                  /* allocate words */
tcm_hres(&mode,params,&flag)                   /* TURBO C medium model MODE call */

if (flag !=0)                                  /* Error if flag is not zero */
{
   printf("\n\nMODE 23 Error Flag = %d\n",flag);
   exit(1);
}

actual_seg = params[7]                         /* Save actual Segment for later
                                                  deallocation with MODE 24 */
dma_seg = params[5]                         /* "good" segment for DMA MODE 6 */


/* Set Up for Call to MODE 6: Set Up and Perform DMA Transfer */

mode = 6                                    /* Set Up and Perform DMA */
params[0] = 40000                              /* number of bytes */
params[1] = dma_seg
params[2] = 1                                  /* Trigger source */
params[3] = 0                                  /* Single-cycle/recycle */
params[4] = 1                                  /* Gain */
tcm_hres(&mode,params,&flag)                   /* TURBO C medium model MODE call */

if (flag !=0)                                  /* Error if flag is not zero */
{
   printf("\n\nMODE 6 Error Flag = %d\n",flag);
   exit(1);
}
    .
    .
    .
/* Release Memory Back to DOS when Finished */
mode = 24                                   /* deallocate memory */
params[0] = actual_seg                         /* Segment to release */
tcm_hres(&mode,params,&flag)                   /* call driver */

if (flag !=0)                                  /* Error if flag is not zero */
{
   printf("\n\nMODE 24 Error Flag = %d\n",flag);
   exit(1);
}
```

Microsoft Quick C / Borland Turbo C Environments

When executing a program from within Borland's TURBO C or Microsoft's Quick C environments, the environment itself uses a large portion of your PC's free memory. This memory usage can limit the amount of memory available for DMA buffer space. If a Memory Allocation Error is returned after a call to the "Memory Allocation for DMA" MODE, alternative action must be taken.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

One option for the case of limited memory is to reduce the number of samples taken and therefore the size of the memory block required. If the number of samples is already at a minimum, and there is still insufficient memory for allocation, another option may be necessary.

A second option is to compile and link the program into a stand-alone executable file and execute it from the DOS command line. If you normally program within an environment and find that your application must be compiled and executed outside of the environment, the following steps may be useful.

1. Test the program from within the environment by taking fewer samples and therefore not exhausting all of your PC's available memory.
2. After you have successfully tested the program from within the environment, increase the number of samples as necessary then compile and link the application into an executable file.

Microsoft QuickBASIC

When executing a QuickBASIC application from the command line, use DAS-HRES MODE 23 to allocate memory for DMA and DAS-HRES MODE 24 to deallocate the memory when DMA is finished and the memory is no longer needed.

When running an application from within the QuickBASIC environment any call to MODE 23 will return an Insufficient Memory error because the QuickBASIC Environment uses all available memory. The Subroutine *Address* in the second example allocates a suitable memory location for DMA transfer and returns the address in aptr%.

Examples of two types of QuickBASIC applications follow.

Example 1:

For QuickBASIC stand-alone executable applications executed from the DOS command line)

```
'----- Allocate memory for DMA (Mode 23) ----------------------------
' *** NOTE : THIS MODE CAN ONLY BE USED FROM AN EXECUTABLE FILE.
'

D%(0) = 5000                                    'NUMBER OF TRANSFERS :
D%(1) = 1                                       'USE WORD MODE
MD% = 23
CALL QBHRES(MD%, VARPTR(D%(0)), FLAG%)  'Allocate memory
IF FLAG% <> 0 THEN PRINT "Error in Memory Allocate # "; FLAG%: STOP

DSEG = D%(5)
ASEG = D%(7)                                    '*** THIS VALUE MUST BE USED WITH
MODE 24
' *** TO DEALLOCATE MEMORY.....

'----- Setup DMA  transfer (MODE 6) ---------------------------------

MD% = 6                                         'Select mode
D%(0) = 5000                                    'NUMBER OF TRANSFERS(MUST BE SAME AS
                                                'ALLOCATED IN MODE 23) :
D%(1) = DSEG
D%(2) = 1                                       'Trigger Source
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
D%(3) = 0                                   'Single cycle/recycle
D%(4) = 0                                   'Gain

CALL QBHRES(MD%, VARPTR(D%(0)), FLAG%)  'set DMA running
IF FLAG% <> 0 THEN PRINT "Error in DMA setup # "; FLAG%: STOP

            .
            .
            .

'----- Deallocate Memory Back To DOS (MODE 24) ----------------------

D%(0) = ASEG                                'actual segment allocated using MODE
23
MD% = 24
CALL QBHRES(MD%, VARPTR(D%(0)), FLAG%)  'set timer
IF FLAG% <> 0 THEN PRINT "Error in releasing Memory # "; FLAG%: STOP
```

Example 2:

For QuickBASIC applications executed from within the QuickBASIC environment)

```
'Declare the Subroutine to Calculate the useable Address for DMA
'DECLARE SUB ADDRESS (DMA() AS LONG, samp AS LONG, addr%)

REM $DYNAMIC
DIM DMA(32000) AS LONG                      'Make array twice as needed
DIM samp AS LONG
samp = 16000
CALL ADDRESS(DMA(), samp, aptr%) 'CALL SUB TO CALC.  DMA ADDRESS
addr = &H300                            'DAS-HRES I/O address

'----- Setup DMA  transfer (MODE 6) ---------------------------------

MD% = 6                                     'Select mode
D%(0) = samp                                      'NUMBER OF TRANSFERS
D%(1) = aptr%
D%(2) = 1                                   'Trigger source
D%(3) = 0                                   'Single-cycle/recycle
D%(4) = 1                                   'Gain

CALL QBHRES(MD%, VARPTR(D%(0)), FLAG%)  'Set DMA running
IF FLAG% <> 0 THEN PRINT "Error in DMA setup # "; FLAG%: STOP
      .
      .
      .

SUB ADDRESS (DMA() AS LONG, samp AS LONG, addr%)
'
'SUBROUTINE TO ALLOCATE SPACE FOR DMA TRANSFERS...
'
DIM a AS LONG
DIM b AS LONG
  CLS

     s = VARSEG(DMA(0))                     'GET SEGMENT ADDRESS
     o = VARPTR(DMA(0))                     'GET SEGMENT OFFSET

     IF o < 0 THEN o = o + 65536
     IF s < 0 THEN s = s + 65536
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        a = s * 16 + o
        PAGE = INT(a / 65536)

      'See if there will be a DMA "WRAP-AROUND"
        b = a - (PAGE * 65536)
        b = b + samp * 2

      IF b > 65535 THEN
        a = (PAGE + 1) * 4096: 'here if PAGE WRAP would occur
        IF a > 32767 THEN
          addr% = a - 65536
        ELSE
          addr% = a
        END IF
      ELSE
        a = INT(a / 16)
        IF a > 32767 THEN
          addr% = a - 65536
      ELSE
        addr% = a
      END IF
    END IF
END SUB
```

Turbo PASCAL

Using versions up to and including 6.0 of the Turbo PASCAL compiler, use DAS-HRES MODE 23 to allocate memory for DMA and DAS-HRES MODE 24 to deallocate the memory when DMA is finished and the memory is no longer needed.

Turbo PASCAL allows the user to specify the memory requirements for a program with the "Memory Allocation Sizes" Parameter Directive {$M}. The {$M} Directive allows specification of the stacksize and the minimum and maximum heap size using the syntax **{$M stacksize,heapmin,heapmax}**.

When specifying a minimum and maximum heapsize for a program that calls the Allocation MODE within the Software Driver, it must be insured that Turbo Pascal does not "use up" all available memory. If the maximum heapsize is not set by the user, Turbo Pascal uses all of remaining memory for its internal heap. The Memory Allocation MODE of the Software Driver can only acquire memory from DOS if the $M directive is used to force Turbo Pascal not to use up all available memory.

The limits for the parameters stacksize, heapmin, and heapmax are as follows (in bytes):

PARAMETER	MINIMUM	MAXIMUM
stacksize	1024	65520
heapmin	0	655360
heapmax	heapmin	655360

For futher information about the "Memory Allocation Sizes" Parameter Directive refer to the Turbo Pascal Manual. Specifically, for Turbo Pascal 4.0 see page 159; for Turbo Pascal 5.0 see page 416 of the Reference Guide; for Turbo Pascal 6.0 see page 209 of the Programmer's Guide.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example:

```
{$M 16384,0,131072}                    (*Reserve 128KB for the TP Heap*)
        .                              (*Manager; reduce this amount*)
        .                              (*if you get "Not enough memory"*)
        .                              (*errors from Mode 9*)

BEGIN
  (* Allocate DMA Buffer Using DAS-HRES MODE 23 *)
  mode := 23;                          (* allocate memory for DMA *)
  params[0] := 40000;                  (* allocate 40000 words *)
  params[1] := 1;                      (* allocate words *)
  tp_hres(&mode,params,&flag);         (* TURBO Pascal MODE call *)

  if (flag <> 0) then                  (* Error if flag is not zero *)
  begin
     write('MODE 23 Error Flag = ');
     writeln(flag);
     halt;                             (* HALT program execution *)
  end;
  dma_seg := params[6];                (* save DMA "good" Segment *)
  actual_seg := params[7];             (* Save actual Segment for later *)

  (* Set Up and Perform DMA Transfer Using DAS-HRES MODE 6 *)
  mode := 6;                           (* Set Up and Perform DMA *)
  params[0] := 40000;                  (* number of words *)
  params[1] := dma_seg;                (* segment allocate *)
  params[2] := 1;                      (* trigger source *)
  params[3] := 0;                      (* single cycle/rerecycle *)
  params[4] := 0;                      (* gain *)

  tp_hres(&mode,params,&flag);(* TURBO Pascal MODE call *)

  if (flag <> 0) then
  begin
      write('MODE 6 Error Flag = ');
      writeln(flag);
      halt                             (* exit to DOS *)
  end;
          .
          .
          .

  (* Releases Memory Back to DOS Using DAS-HRES MODE 24 *)
  mode := 24;                          (* deallocate memory *)
  params[0] := actual_seg;             (* Segment to release *)

  tp_hres(&mode,params,&flag);         (* call driver *)

  if (flag <> 0) then
  begin
      write('MODE 24 Error Flag = ');
      writeln(flag);
      halt                            (* exit to DOS *)
  end;
          .
          .
          .
END.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Microsoft PASCAL

Using versions up to and including 4.0 of the Microsoft Pascal Compiler, use DAS-HRES MODE 23 to allocate memory for DMA and DAS-HRES MODE 24 to deallocate the memory when DMA is finished and the memory is no longer needed.

Example:

```
mode := 23;                          (* allocate memory for DMA *)
params[0] := 40000;                  (* allocate 40000 words *)
params[1] := 1;                      (* allocate words *)
msp_hres(&mode,params,&flag);        (* MICROSOFT Pascal MODE call *)
if (flag <> 0) then
begin
    write('MODE 23 Error Flag = ');
    writeln(flag);
    abort(0)                         (* exit to DOS *)
end;

dma_seg := params[6];                (* save DMA "good" Segment *)
actual_seg := params[7];             (* Save actual Segment for later *)
                                     (*  deallocation with MODE 24 *)

(* Set Up for Call to MODE 6: Set Up and Perform DMA Transfer *)
mode := 6;                           (* Set Up and Perform DMA *)
params[0] := 40000;                  (* number of words *)
params[1] := dma_seg;                (* segment address *)
params[2] := 1;                      (* trigger source *)
params[3] := 0;                      (* single cycle/recycle *)
params[4] := 0;                      (* gain *)
msp_hres(&mode,params,&flag);        (* MS Pascal MODE call *)
begin
    write('MODE 6 Error Flag = ');
    writeln(flag);
    abort(0)                         (* exit to DOS *)
end;

    .
    .
    .
(* Release Memory Back to DOS when Finished *)
mode := 24;                          (* deallocate memory *)
params[0] := actual_seg;             (* Segment to release *)
msp_hres(&mode,params,&flag);        (* call driver *)
begin
    write('MODE 24 Error Flag = ');
    writeln(flag);
    abort(0)                         (* exit to DOS *)
end;
```

Microsoft FORTRAN

Using versions up to and including 5.0 of the Microsoft FORTRAN Cmpiler, use DAS-HRES MODE 23 to allocate memory for DMA and DAS-HRES MODE 24 to deallocate the memory when DMA is finished and the memory is no longer needed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example:

```
        mode = 23
       i(1) = 40000
       i(2) = 1

C      Allocate DMA Buffer Using Mode 9

         call fhres(mode, i(1), Flag)

       if (flag .NE.  0) then
       print *,'Mode = ',mode,'     Error # ',flag
       goto 35
       endif
       dmaseg = i(7)
       actseg = i(8)

C    Set Up and Perform DMA Transfers

        mode = 6
       i(1) = 40000
       i(2) = dmaseg
       i(3) = 1
       i(4) = 0
       i(5) = 0
       flag = 0

        call hres(mode, i(1), Flag)

       if (flag .NE.  0) then
       print *,'Mode = ',mode,'     Error # ',flag
       goto 35
       endif

C    Free DMA Buffer using MODE 24

        mode = 24
        flag = 0
       i(1) = actseg


        call hres(mode, i(1), Flag)

       if (flag .NE.  0) then
       print *,'Mode = ',mode,'     Error # ',flag
       goto 35
       endif

35     end
```

BASIC(A)

The addition of the memory Allocate/Deallocate modes to your DAS driver eliminates the old problems of having to estimate a free-memory area for use as the DMA buffer. Foe all versions of interpreted BASIC, use the DAS-HRES MODEs 9 and 10 for all DMA requirements.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
580 '----- ALLOCATE DMA MEMORY BUFFER -------------------------------------
-----------------
590 D%(0) = 1000                        'Number of samples to allocate for
595 D%(1) = 0                           'each sample is a byte.
600 MD% = 9
610 CALL PDMA (MD%, D%(0), FLAG%)
620 IF FLAG%<>0 THEN PRINT "Error Allocating Memory ";FLAG%:STOP
630 TSEG = D%(5)                        'Save Memory segment to transfer to/from
640 TOFF = D%(6)                        'Save Memory offset to start transfer
                                        ' to/from
650 ACTSEG = D%(7)                      'Save actual memory segment
660 '
670 '----- Setup DMA Transfer (MODE 6) -----------------------------------
-----------------
680 MD% = 1                             'Select mode
690 D%(0) = 1000                        'Number of transfers
700 D%(1) = 0                           'Byte transfers
710 D%(2) = 0                           'Direction
720 D%(3) = 0                           'Single cycle
730 D%(4) = 1                           'Transfer source - internal timer
740 D%(5) = TSEG                        'Transfer segment
750 D%(6) = TOFF                        'Transfer offset
```

■ ■ ■

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP