

HI TECH TOOLS
27 RUE VOLTAIRE
72000 LE MANS- FRANCE
Tél. 33 02 43 28 15 04
Fax. 33 02 43 28 59 61
Info@hitechtools.com
http://www.hitechtools.com
<Instrumentation Electronique &Outils de développement>

Table of Contents

CHAPTER 1 INTRODUCTION I
1.1 GENERAL DESCRIPTION 1
1.1.1 KEY FEATURES..- 2
1.2 PROGRAMMING OVERVIEW 3
1.3 ORDERING INFORMATION 3
1.4 HOW TO USE THIS MANUAL 4

CHAPTER 2 INSTALLATION 5
2.1 UNPACKING AND INSPECTION 5
2.2 MAKING A BACK-UP COPY OF THE DISKETTE 5
2.3 SOFTWARE INSTALLATION 6
2.4 SWITCHES, JUMPERS, AND CONNECTORS 6
2.4.1 BASE ADDRESS SWITCH 9
2.4.2 JUMPERS 10
2.4.2.1 TIMER 0 CLOCK JUMPER 10
2.4.2.2 D/A REFERENCE JUMPER 11
2.4.3 CONNECTOR 11
2.5 HARDWARE INSTALLATION 13

CHAPTER 3 15
3.1 PROGRAMMING IN Quick BASIC 15
3.1.1 FORMAT OF THE CALL STATEMENT 15
3.1.2 CREATING EXECUTABLE PROGRAMS 17
3.1.2.1 STAND-ALONE DEVELOPMENT 17
3.1.2.2 Quick BASIC ENVIRONMENT DEVELOPMENT 18
3.2 PROGRAMMING IN C 18
3.2.1 SYNTAX OF DRIVER FUNCTION 19
3.2.2 LINK WITH LIBRARY 20
3.3 DRIVER FUNCTIONS DESCRIPTIONS 20
3.3.1 Function 0 22
3.3.2 Function 1 24
3.3.3 Function 2 26
3.3.4 Function 3 27
3.3.5 Function 4 29
3.3.6 Function 5 32
3.3.7 Function 7 35
3.3.8 Function 8 36
3.3.9 Function 9 38

3.3.10 Function 10 40
3.3.11 Function 11 42
3.3.12 Function 12 43
3.3.13 Function 13 45
3.3.14 Function 14 6
3.3.15 Function 15 47
3.3.16 Function 17 48
3.3.17 Function 18 50
3.3.18 Function 19 52
3.3.19 Function 20 54
3.3.20 Function 21 56
3.4 PROGRAMMING EXAMPLES 62

CHAPTER 4 REGISTER FORMAT AND STRUCTURE 63
4.1 1/0 Port Address Map 63
4.2 A/D DATA REGISTERS (READ BASE+O & +1) 64
4.3 D/A OUTPUT REGISTERS (WRITE BASE+0,+1) 65
4.4 DIGITAL I/O REGISTERS (BASE+2) 65
4.5 A/D STATUS & CLEAR INTERRUPT REGISTER (BASE+3) 66
4.6 GAIN CONTROL REGISTER (BASE+4) 67
4.7 MUX CONTROL REGISTER (BASE+5) 68
4.9 SOFTWARE TRIGGER REGISTER (BASE+7) 69
4.10 PROGRAMMABLE TIMER/COUNTER REGISTER (BASE+8,+9, +10 & +11) 69

CHAPTERS PROGRAMMABLE TIMER/COUNTER 71
5.1 COUNTER CONTROL REGISTER 71
5.2 COUNTER OPERATION MODE 73
5.2.1 Mode 0 - Pulse on Terminal Count 73
5.2.2 Mode 1 - Programmable One-Shot 73
5.2.3 Mode 2 - Rate Generator 73
5.2.4 Mode 3 - Square Wave Generator 74
5.2.5 Mode 4 - Software Triggered Strobe 74
5.2.6 Mode 5 - Hardware Triggered Strobe (Retriggerable) 74
5.3 WRITE OPERATION 74
5.4 READ OPERATION 75
5.4.1 Counter Latch Command 75
5.4.2 Read-Back Command 76
5.5 GENERATING SQUARE WAVES OF PROGRAMMED FREQUENCY 77
5.6 MEASURING FREQUENCY AND PERIOD 78
5.7 TRIGGERING THE A/D PERIODICALLY 78
5.8 GENERATING INTERRUPTS WITH THE TIMER/COUNTER 79

CHAPTER 6 AID CONVERSION 81
6.1 ANALOG INPUT RANGE SELECTION 81
6.2 AID TRIGGER MODE 81
6.3 AID DATA TRANSFER 82

CHAPTER 7 DIA CONVERSION 83
7.1 D/A DATA TRANSFER 83
7.2 WAVEFORM GENERATION 83

CHAPTER 8 SIGNAL CONNECTION 85
8.1 ANALOG INPUT CONNECTION 85
8.2 ANALOG OUTPUT CONNECTION 86
8.3 DIGITAL SIGNAL CONNECTION 86

CHAPTER 9 CALIBRATION 89
9.1 A/D CALIBRATION 89
9.2 D/A CALIBRATION 89

APPENDIX A SPECIFICATIONS 90
APPENDIX B DRIVER ROUTINE RETURN ERROR MESSAGE 92

Copyright

Copyright 1992 by Acrosser Technology. All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system , or translated into any language or computer language, in any form or any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior written permission of Acrosser Technology.

Acrosser, IBM, IBM PC, PC/AT MS-DOS ,Microsoft, TURBO, are registered trademarks.

Disclaimer

Acrosser Technology makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchant ability or fitness for any particular purpose. Further, Acrosser Technology reserves the right to revise this publication and to make changes from time to time in the contents hereof without obligation of Acrosser Technology to notify any person of such revision or changes.

Return your card for service

If your card requires servicing, contact the dealer from whom you purchased the product for service information. If you need to ship your card to us for service, be sure it is packed in a protective carton. We recommend that you keep the original shipping container for this purpose.
You can help assure efficient servicing of your product by following these guidelines:

1. Include your name, address, telephone and facsimile number where you may be reached during the day-time.

2. A description of the system configuration and/or software at the time of malfunction.
3. A brief description of the symptoms.

Technical Support and User Comments

User comments are always welcome as they assist us in improving the usefulness of our products and the understanding of our publications. They form a very important part of the input used for product enhancement and revision.

We may use and distribute-any of the information you supply in any way we believe appropriate without incurring any obligation. You may, of course, continue to use the information you supply.

Please send your comments to Acrosser Technology Co., Ltd. or your local sales representative.

This page intentionally left blank

CHAPTER 1 INTRODUCTION

1.1 GENERAL DESCRIPTION

The AR-B3204 is multi-function, general purpose, programmable, A/D,D/A I/O interface boards for the IBM PC/XT/AT and compatibles. It is an half-length board that install internally in an expansion slot of an IBM Personal Computer to turn the computer into a high-precision data acquisition and signal analysis instrument. AR-B3204 is designed with a 2-layer construction with a wide area ground plane to minimize noise and crosstalk at high frequencies.

The AR-83204 uses an industry standard 16-bit successive approximation A/D converter. Th AR-B3204 is operated with 16 single-ended inputs, and bipolar modes. Maximum AID throughput is up to 30K.

A/D conversion can be triggered in three ways, by software, by pacer, or by an external trigger pulse. The transfer of the data can also be accomplished in 2 ways, by programmable control, or by interrupt service routine. All mode are selectable by software.

The AR-B3204 offers 5 software programmable input range. Selectable range include +/- 0.3125, +/- 0.625V, +/- 1.25V, +/- 2.5V, and +/- 5V volts.

The AR-B3204 also features one channel of multiplying 12-bit D/A output. The DAC can use a fixed -5V or -10V references available from on-board for a 0 to +5V or 0 to +10V output range. DIA is double-buffered to provide instantaneous single-step updates. The data transfer of D/A converter can be accomplished by software, or interrupt service routine.

Digital I/O consists of 4 bits of TTL-compatible digital output and 7 bits of digital input. Apart from being addressed as I/O ports, one of the digital inputs do double duty as A/D external trigger input. The digital outputs are low at their power on state, and stay at the same state(latched) until the next output operation occurs.

The AR-B3204 uses the Intel 8253 programmable interval timer. It consists of three independent 16 bit down counters. Each counter has a clock input, control gate and an output that can be programmed to count in the range of 2 to 65535. The maximum clock input frequency of 8253 is 4 MHz.

Counter 1 and counter 2 on the AR-B3204 are cascaded. Counter 2 input is connected to a precision 2 MHz crystal oscillator. The output of counter 2 is connected to the clock input of counter 1. This makes a 32-bit wide timer/ counter. The output of counter 1 is internally configured to provide AID trigger pulses and also routed to the main I/O connector.

Counter 0 is uncommitted and its input, output, and gate control are available to the user on the main I/O connector. It can be used to count events or measure frequency, period and pulse width.

The AR-B3204 comes complete with a utility software package that greatly simpifies the operation of the board. The utility software includes:

1. BASIC and C language drivers for control of the A/D, DIA , counterttimer and digital interface , and allow the user to completely utilize the power of the board.
2. Calibration routine.
3. Examples, and demostration programs.

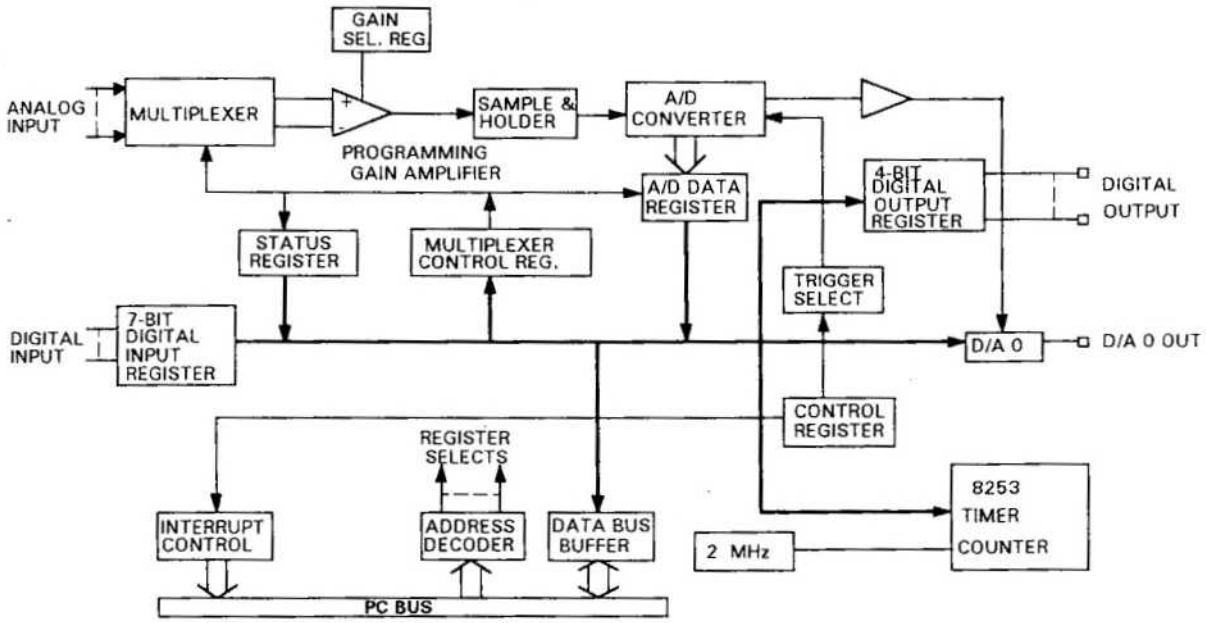


Figure 1-1 AR-B3204 Functional Block Diagram

1.1.1 KEY FEATURES

The key features of AR-B3204 include:

- 16 single-ended analog input channels.
- 12 bit successive approximation AID converter with 25uS conversion time.
- 5 programmable analog input ranges: +/-5V, +/-2.5V, +/-1.25V, +/-0.625V, +/-0.3125

- A/D conversion can be triggered by any of the following: software trigger, programmable pacer trigger or direct external trigger to the AID.
* A 3-channel programmable interval ti mer (Intel 8253) provides trigger pulses for the A/D at any rate from 500 KHz down to 0.00046 Hz. The time base is jumper selectable. Two channels are operated in a fixed divider configuration from an internal crystal dock. The third channel is uncommitted and provides a gated 16-bit counter that can be used for event or pulse counting, delayed triggering, and frequency measurement.
- One channel of 12-bit monolithic multiplying D/A output. The D/A data can be transferred by program control or pacer triggered interrupt.
- The DAC can use the on board +5V (+/-0.05V), or +10V (+/-0_ 1 V) precision reference voltage to have 0 to 5V or 0 to 10V output range.

1.2 PROGRAMMING OVERVIEW

The AR-B3204 can be programmed by two options:

(1) Call Functions
(2) Low-Level Direct Register 1/O

The Cal! Function Interface is designed to use Compiled BASIC or C language. The function calls are accessed from your program. This collection of functions allows a great deal of flexibility and background data acquisition to take place. The Call Functions are described in Chapter 3.

The second method, Low-Level Direct Register I/O, should be used only if you have a high-level of programming expertise and understanding of your PC's architecture and the PC's Peripheral Controllers (i.e. 8259 Interrupt Controller etc.) and the AR-B3204 hardware. This method can use any language. The Register Map and other details needed for Low Level Programming are provided in Chapter 4 through 6.

1.3 ORDERING INFORMATION

Following lists and describes part numbers for the AR-B3204 board and its accessories.

Part Number	Description
AR-B3204	Includes the AR-B3204, a 16-channel, general purpose data acruisition card, as well as software and appropriate documentation.

AR-B9007 Includes a screw Terminal Board and a 60 cm shielded round cable to connect AR-B3204.

1.4 HOW TO USE THIS MANUAL

This manual provides you with the information necessary to install and program the AR-B3204. It assumes you are familiar with your personal computer, DOS and have some programming experience. In addition to this manual, you may also need:

* The documentation for your computer.
* A language reference manual, if you intend to program custom applications for the AR-B3204.

The remainder of this manual is as follows:

Chapter 2, Installation, provides hardware installation instructions. It describes the switch, jumpers, and connector. It also describes how to install AR-B3204.

Chapter 3, Programming the AR-B3204 with Driver Routine, describes how to use the Call Functions from programs written in Quick BASIC and C. It also provides detailed descriptions of each call function.

Chapter 4, Register Format and Structure, describes the registers on the AR-B3204 and how they are used to program the AR-B3204.

Chapter 5, Programmable Timer Counrter, provides details of the 8253 Programmable Interval Timer.

Chapter 6, AID Conversion, describes the programming and use of AR-133204's AID converter.

Chapter 7, D/A Conversion, describes the programming and use of AR-133204's D/A converter.

Chapter 8, Signal Connection, explains the connections between 1/0 signal types, how to use them and how to avoid group loop.

Chapter 9, Calibration, describes how to calibrate the AID and D/A channels.

The Appendices provides other useful information. Appendix A describes the AR-133204's specifications. Appendix B lists the Call Function error messages.

CHAPTER 2 INSTALLATION

This chapter provides instructions for installing the AR-B3204 in an IBM PC/ XT/AT and compatible modes.

2.1 UNPACKING AND INSPECTION

The AR-B3204 was carefully inspected both mechanically and electrically before shipment. It should be free of marks and scratches and in perfect electrical order on receipt.

Unpack the board and place it on its anti-static packaging. Hold the board in one hand and place the other hand firmly on a metal portion of the system chassis. This procedure eliminates static electricity build-up, preventing any possible damage to the board. After allowing a moment for static electricity discharge, carefully unwrap the board from the anti-static material.

After unpacking, check the unit for signs of shipping damage. If there is any damage to the unit or it fails to meet specifications, notify our service department or your local sales representative immediately. We will make arrangements to repair or replace the unit.

2.2 MAKING A BACK-UP COPY OF THE DISKETTE

Distribution software is furnished on a 5.25" diskette. Before installing the software, first create a back-up copy of the provided diskette. Followthe general procedure outlined below.

1. Turn on your computer and display monitor.

2. Switch to the directory containing your DISKCOPY.EXE DOS utility.

3. At the DOS prompt, type:

   DISKCOPY (source drive designator): (copy drive designator):

4. When your copy has been completed, put the original disk in a safe place. Label the back-up copy and use this disk to install the software.

2.3 SOFTWARE INSTALLATION

The AR-B3204 software should be installed on your computer's hard disk. To install the software:

1. Create a special directory for the AR-B3204 in the root directory of your hard drive. Assume this directory_ is called B3204. At the DOS prompt,_ change to the root directory, i.e. type

   cd \

2. Next, create the 63204 directory. Type:

   and \B3204

   Change to the B3204 directory by typing:

   cd \B3204

3. Place the AR-B3204 Disk into the floppy drive(assume this is drive a:) and type:

   xcopy a: ls

Once you have successfully installed the software you should have two sub-directories under the B3204 directory. The C sub-directory contains C language driver for AR-B3204. The BASIC subdirectory contains Quick BASIC driver.

Example programs are also copied in the BASIC sub-directory.

2.4 SWITCHES, JUMPERS, AND CONNECTORS

There are one functional switch and two jumpers on the AR-B3204. This section describes functional use of these switch and jumpers. It also provides a description of the Analog and Digital I/O connector. Figure 2-1 shows the location of each switch, jumper, and connector for AR-B3204. Table 2-1 list the functions of jumpers, switch, variable resistors and connector shown in Figure 2-1.

Table 2-2 lists the factory default configuration of switches and jumpers on the board.

Legend	Description
DSW1	I/O port base address'
JP1	D/A channel reference (+5V/+10V)
JP2	Timer clock 0 input source (2MHz/EXT)
VR1	D/A channel gain adjustment
VR2	A/D Offset adjustment
VR3	A/D gain adjustment
VR4	A/D null adjustment
CN1	Main I/O connector

Table 2-1 List of switch, jumpers and variable resistors.

Table 2-2. AR-B3204 Default Configuration.

Switch/Jumper	Default
Base Address	300H
Timer 0 Clock Source	2 MHz
D/A Reference Voltage	+10V

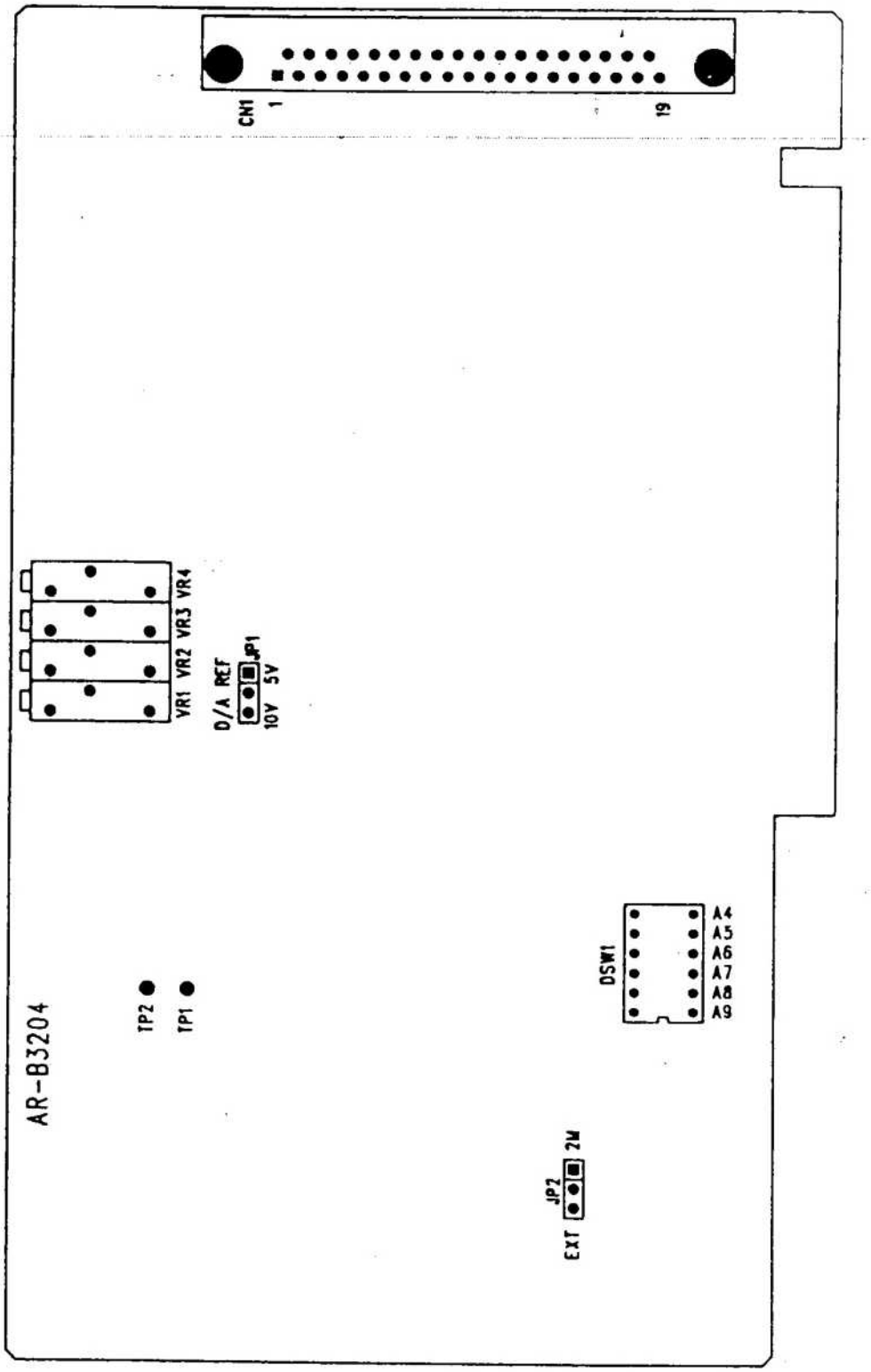
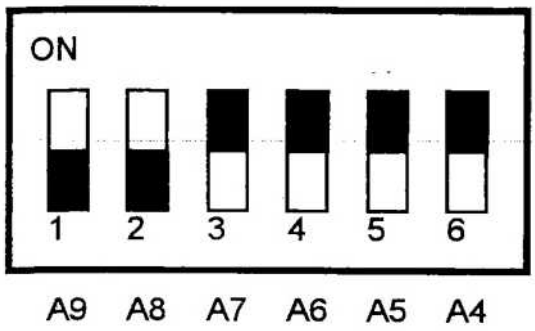


Figure 2-1. AR-B3204 Switches and Jumpers Location.

2.4.1 BASE ADDRESS SWITCH



ADDRESS LINE	ADDRESS VALUE HEX	DECIMAL
A4	10	16
A5	20	32
A6	40	64
A7	80	128
A8	100	256
A9	200	512

ADDRESS=200* A9+100*A8+80*A7+40*A6+20*A5+10*A4 (Hex)

OR

=512*A9+256*A8+128*A7+64*A6+32*A5+16*A4 (Dec)

A9 .... A4 = 0 IF SWITCH IS ON.

A9 .... A4 = 1 IF SWITCH IS OFF.

Figure 2-2. Base Address switch

Switch name: DSW1

The operation of AR-B3204 is controlled through the input and output ports. Sixteen consecutive I/O ports are used by AR-B3204. The Base Address of these I/O ports is selected by using Base Address Switch DSW1. The AR-B3204 is factory-configured for a Base Address of 300 Hex. If this Base Address is already occupied, you will need to change the Base Address switch setting. It is essential that each peripheral device, such as a AR-B3204, be assigned a unique Base Address within the range of 200 to 3f0 Hex and that the address is on a 16-byte boundary. Figure 2-2 shows the Base Address Switch. Use Table 2-3 as an aid to selecting an unoccupied Address.

Table 2-3. IBM PC 1/O Address Map

110 Address (HEX)	Peripheral
200-207	Game Controller
278 - 27F	Parallel Port 2
2F8 - 2FF	Serial Port 2
300 - 31F	Prototype Card
378 - 37F	Parallel Port 1
380 - 38F	SDLC,BISYNC 2
3A0 - 3AF	BISYNC 1
3130 - 3BF	Mono Display / Printer port
3C0 - 3CF	EGA Adapter
3D0 - 3DF	CGA Adapter
3F0 - 3F7	Floppy Disk Controller
3F8 - 3FF	Serial Port 1

2.4.2 JUMPERS

The AR-B3204 has two jumpers to configure timer 0 clock source and D/A reference voltage.

2.4.2.1 TIMER 0 CLOCK JUMPER

Jumper name : JP2

This jumper selects the timer clock input frequency to the channel 0 of 8253 programmable timer/counter. Insertion of the jumper (JP3) to the right position(marked '2M') will select internal 2MHz clock frequency. If the jumper is inserted to the left position (marked 'EXT'), The clock input of counter 0 is entered from external clock through pin 18 of 1/O connector. In this mode counter 0 can be used as an event counter. The counter may be used for waveform generating, period counting, or as a secondary timer.

2.4.2.2 DIA REFERENCE JUMPER

Jumper name : J P1

The AR-B3204 provides a jumper to select the D/A converter reference voltage. If the jumper is inserted at the left position(marked '10V), the reference input of D/A. channel. is connected to the on-board +10V reference voltage and the D/A channel has an output range of 0 to +5V. If JP1 is inserted at the right position(marked ' 5V) , the reference input of D/A channel is connected to the on-board +5V reference voltage and the D/A channel has an output range of 0 to 5V.

2.4.3 CONNECTOR

The AR-B3204 is equipped with a 37-pin, D-type female connector to access analog and digital I/O. This connector and its signal conductor functions are described in Table 2-4 and Table 2-5.

Table 2-4 Main //O Connector (CNI)

Pin	Name	Description
1	ADO	A/D CHO
2	AD1	A/D CH1
3	AD2	A/D CH2
4	AD3	A/D CH3
5	AD4	A/D CH4
6	AD5	A/D CH5
7	AD6	A/D CH6
8	AD7	AID CH7
9	IP4	Digital input CH4
10	IP5	Digital input CH5
11	AGND	Analog ground
12	AGND	Analog ground
13	IPO/TRG	Digital input CHO I External trigger

Table 2-5 Main I/O Connector (CM) (Continued)

Pin	Name	Description
14	IP1	Digital input CH1
15	IP2	Digital input CH2
16	IP3	Digital input CH3
17	COOUT	Timer/Counter 0 OUT
18	COCIN	Timer/Counter 0 external clock input
19	+5V DC	+ 5V DC power
20	AD8	A/D CH8
21	AD9	A/D CH9
22	AD10	A/D CH10
23	AD 11	AID CH 11
24	AD12	A/D CH12
25	AD 13	A/D CH 13
26	AD14	A/D CH14
27	AD15	A/D. CH15
28	DAOOUT	D/A CHO output
29	D16	Digital input CH6
30	CIOUT	Timer/Counter 1 output
31	AGND	Analog ground
32	OPO	Digital output CHO
33	OP1	Digital output CH1
34	OP2	Digital output CH2
35	OP3	Digital output CH3
36	GATED	Counter 0 gate control
37	DGND	Digital ground

2.5 HARDWARE INSTALLATION

This section provides general instructions for installing the AR-B3204 card. For more detailed information regarding installation of peripheral boards, consult the documentation provided with your computer.

WARNING
TURN OFF YOUR PC POWER SUPPLY WHENEVER INSTALLING OR REMOVING ANY ADAPTER BOARD.

Installing the AR-B3204 card in your computer:

1. Set the switches and jumpers as described in section 2.4.
2. Turn off the power of the computer and any peripheral devices.
3. Unplug the power cord from the electrical outlet. Make a note concerning location of all the cables and cords attached to the rear of the system unit and then disconnect.
4. Remove the system unit cover (refer to your computer user's guide if necessary).
5. Choose an available option slot. Remove the screw that secures the expansion slot cover to the system unit.
6. Hold the AR-B3204 in one hand. With the other hand, touch any metallic part of the computer's cabinet. This will safely discharge any static electricity which has built-up in your body.
7. Align the gold edge connector with the edge socket and install the back adapter with the adapter plate screw Gently press the board downward into the socket. Re-install the adapter plate screw
8. Replace the system unit cover.
9. Connect all cables and cords you previously removed in step 3.

This page intentionally left blank

CHAPTER 3
PROGRAMMING THE AR-B3204 WITH DRIVER ROUTINES

The AR-B3204 package provides driver routines for Quick BASIC and C Language. Those functions covered by these driver routines include the application of A/D, D/A, Digital i/O and programmable Counter Timer. The various function numbers of the driver routine select all functions of the AR-B3204, format and error-check data, and perform frequently used sequences of instructions.

3.1 PROGRAMMING IN Quick BASIC

The Quick BASIC drivers of AR-B3204 support Microsoft Quick BASIC version 4.0 and above. There are two files in the BASIC sub-directory of your AR-B3204 software diskette. The "BAS3204. LI B" is used for stand-alone development. To develop your program under the Quick BASIC-Environment, "BAS3204.QLB" allows you to take full advantage of QB's power.

3. 1.1 FORMAT OF THE CALL STATEMENT

The format of the CALL statement for the Quick BASIC driver is:

CALLS ARB3204(F U N %, PARA%(0),ARY1 %(0),ARY2%(0), E R%)

Where

ARB3204 Name of the driver routine.

FUN% An integer variable specifying the function number of the called routine.

PARA%(0) A variable specifying the starting location of the call function parameter array.

ARY1 %(0) A variable specifying the starting location of the data array. This data array is used in some of the call functions. For those functions which do not need a data array, it is a dummy array.

ARY2%(0) A variable specifying the starting location of the data array. This data array is used in some of the call functions. For those functions which do not need a data array, it is a dummy array.

ER% An integer variable containing the return message code.

The driver routine needs your QB program to pass above variables by far reference. The second method to call the driver is : .

CALL ARB3204(SEG FUN%,SEG PARA%(O),SEG ARY1%(0),SEG ARY2%(0),SEG ER%)_ .

Using CALLS is the same as using CALL with the SEG keyword applied to each argument.

Programming note:

1. Declare the prototype of the driver routine in the first line of the program as:

   DECLARE SUB ARB3204(FUN%, PARA%, ARY1%, ARY2%, ER%)

This will invoke the Quick BASIC to check the number and type of arguments.

2. The arguments must always be written in the correct order. The subroutine does not recognize the names of the variables, just their location. For example, if the line

   CALLS ARB3204(PARA%(O),FUN%, ARY1%(O),ARY2%(O), ER%)

is used, the driver would interpret PARA%(O) as function number, FUN% as parameter array, etc.

3. All parameters must be defined as integer variables. If you do not declare the prototype of the driver, CALL will not perform any error checks on the variable type. Using the wrong variable type will make the CALL function perform incorrectly.

4. Do not perform any arithmetic functions within the parameter list brackets of the CALL statement. For example: CALLS ARB3204(FUN%, PARA%(0)*3, ARY1 %(0), ARY2%(O), ER%) will not perform correctly.

5. Do not use a constant for any of the parameters.

6. You can assign any name to the variables that you wish.

7. You should declare all variables before executing the CALL. If you do (not, the simple variable will be declared by default on execution, but array

variables cannot be dimensioned by default and must be dimensioned before the CALL. Most functions of the CALL routine require multiple items of data to be passed in an array. For this reason, PARA%(O) is specified as the data variable so that the CALL routine can locate the whole array from the position of its initial element.

Likewise,-any-other-CALL-arguments-may be integer_ array_ variables if.. required, and you can name any number of different integer data arrays for output and input. It is possible to dimension arrays with more elements than those that will be used by the CALL. Unused elements will be unchanged.

3.1.2 CREATING EXECUTABLE PROGRAMS

This section describes how to integrate the AR-B3204 driver with your Quick BASIC program. Supplied with your AR-B3204 are the following linkable libraries:

BAS3204.QLB Load this Quick Library into your Quick BASIC Integrated Environment.
BAS3204.LIB Link this library to your stand-alone Quick BASIC program.

3.1.2.1 STAND-ALONE DEVELOPMENT

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

* From within the Quick BASIC environment
* From the DOS command line

Creating within the QB environment

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

1. Enter the following command at the DOS prompt:
   QB SAMPLE02 /L BAS3204
2. Select the RUN menu item.
3. Select the MAKE EXE FILE... from the RUN menu.
4. Select Produce: Stand-Alone EXE File.
5. Select MAKE EXE and Exit option.

Execute the program by simply typing SAMPLE02.

Creating from DOS command line

- compile Quick BASIC programs from DOS command line, link the driver with your program using the following command.

```
BC 10 SAMPLE02;
LINK SAM PLE02,,, BAS3204;
```

The /0 switch causes the Quick BASIC compiler to create programs that link BCOM45.LIB and therefore, do not require BRUN45.EXE at execution time. The /0 switch will also cause LINK to link with BCOM45.LIB library. If BCOM45.LIB is not located in the current directory and the directory BCOM45.LIB is not appended to the DOS append environment variable, LINK will prompt you to enter the path where BCOM45. LIB is located. Consult your Quick BASIC user's manual for details about compiling and linking.

3.1.2.2 Quick BASIC ENVIRONMENT DEVELOPMENT

- develop programs in QB environment, you have to load the AR-B3204 driver's Quick Library BAS3204.QLB into the QB environment from the DOS command line as follows:

```
QB IL BAS3204
```

- load an application program (such as SAMPLE02.BAS) together with the Quick Library and SAMPLE02.BAS, type:

```
QB SAMPLE02 IL BAS3204
```

Select Start from the RUN menu item to run this program.

3.2 PROGRAMMING IN C

The AR-B3204's driver for C language supports Microsoft C and Turbo C. Four C libraries are provided in the C directory of software diskette packed with your AR--B3204 card:

1. 3204CS.LIB Library for SMALL memory model.

2. 3204CM.LIB Library for MEDIUM memory model.

3. 3204CC. LI B Library for COMPACT memory model.

4. 3204CL.LIB Library for LARGE memory model.

3.2.1 SYNTAX OF DRIVER FUNCTION

The syntax of AR-133204's C driver is:

*void arb3204(int func, int *para, int *aryl, int **ary2,** int *er);*

int func; Number of driver function.

int *para; Pointer to beginning of parameter array.

int *aryl ; Pointer to beginning of data array 1.

int *ary2; Pointer to beginning of data array 2.

int *er; Pointer to returned message code.

Application note

1. **Declare the prototype of the driver function** in your program **as:**

   void arb3204(int func, int *para, int *aryl, int *ary2, int *er);

   This will invoke the C compiler to check the number and type of arguments.

2. **The arguments must always be written in the correct order. Mission** argument or pass argument in incorrect order will cause the to driver perform incorrectly. The following is an example of how to call a driver function.
   arb3204(3, &para[O], &aryl [0], &ary2[0], &er);

   Since the name of the C array also represents the pointer of the first element of that array, you can call the driver function as

   arb3204(3, para, aryl, ary2, &er);

3. **All parameters must be defined as integer variables.** If you do not declare the prototype of the driver, the compiler will not perform any error checkes on the variable type. Using the wrong variable type will make the CALL function perform incorrectly.

4. You can assign any name to the variables that you wish. Likewise, any of the other arguments may be integer array variables if necessary, and you can name any number of different integer data arrays for output and input. It is possible to dimension arrays with more elements than those that will be used by the function.

3.2.2 LINK WITH LIBRARY

This will not describe in detail how to make an executable program in C language. Please consult your C compiler user's manual for further details. The following is a simple example of how to compile and link AR-133204's C Library to your program by using Microsoft C.

To compile your program to.obj file:

```
cl Ic program.c;
```

Link the obj file with a small memory model library and make the executable program

```
link program ,,,3204cs;
```

3.3 DRIVER FUNCTIONS DESCRIPTIONS

This section describes in detail all driver functions in the Quick BASIC CALL statement. All functions provided in BASIC driver are identical with C driver functions. Once you are familiar with BASIC driver, the same method uses the C driver. A brief listing of available functions that are given in Table 3-1.

Table 3-1. Driver functions list

Function	Description
o	nitialize. Initializes the AR-B3204 with appropriate operating parameteys.
1	Sets the start and stop channel of auto-scan limits.
2	Reads auto-scan limit setting and next scan channel.
3	Performs a single software trigger A/D conversion and returns data. Multiplexer address will increase to the next channel. (Low speed, Foreground operations)
4	Performs specified number of A/D conversions after trigger. Transfers data into data array 1 via interrupt. (Medium speed, Foreground Operation)

Table 3-1. Driver functions list (countinued)

Function	Description
5	Performs specified number of AID conversions. Transfers data into a memory buffer via interrupt. (Medium speed, Background Operation)
6	Reserved
7	Terminates interrupt operation of functions 5, 18, or 20.
8	Obtains status of interrupt operation of functions 5,18, or 20.
9	Converts A/D data in memory buffer and transfers to an integer array.
10	Sets Counter 0 operating mode.
11	Writes Counter 0 data.
12	Reads Counter 0 data.
13	Output data to digital outputs OPO-3.
14	Read digital inputs data I PO-6.
15	Output data to both D/A channel.
16	Reserved
17	Sets pacer rate of interrupt operation.
18	Performs AID conversion and D/A waveform output.
19	Waits for a certain level analog input. (Analog trigger function)
20	Performs A/D block channel scan on a single pacer interrupt.
21	Set A/D channel gain.
22	Output square wave on Counter 0.
23	Read AID gain code of specific channel.
100	Read driver version date code.

3.3.1 Function 0 - INITIALIZE

Purpose

This function initializes the AR-B3204 base address, and interrupt level according to the parameters specified by the user. It also initializes the following items:

- Sets default scan limits of Channel 0 to Channel 15.

* Performs the present check of AR-B3204 hardware.
- Configures the Programmable Counter/Timer channel 1 & 2 to the Rate Generator Mode and set the pacer rate to 1 KHz.
* Disables all interrupt, and external trigger functions by clearing the Control register.
- Set gains of all channels to 1.

Input Arguments

The following input arguments have to be initialized before calling the Function 0:

FUNC% = 0

PARA%(O) = The 110 port Base Address. Base address can be located between 200 to 3F0 hex.

PARA%(1) = The Interrupt Level. Valid interrupt levels are 2 through 7.

Returns

ER%	= 0	Good return.
	= 2	Invalid function number.
	= 3	Invalid i/O port base address.
	= 4	Invalid interrupt level.
	22	Hardware failure. Could be invalid base address setting or installation error.

Programming Notes

1. <u>This function must be called prior to any other function.</u> Otherwise, no other function can be executed. Function 0 only needs to be called once in the initialization of your program.

2. The Base Address specified must match the setting of the DSW1.

3. Note that the interrupt will not be enabled unless you enter a function which requires interrupt for operation. If you are not going to use interrupt, any level can be chosen. Function 5, 18, and 20 will enable the interrupt for their operation.

Example for Quick BASIC

```
DIM-PARA%(O TO 9)
DIM ARYI%(0 TO 9)
DIM ARY2%(0 TO 9)
PARA%(0) = &H300 'SET BASE ADDRESS
PARA%(1) = 2          'USE IRQ2
FUNC% = 0             'FUNCTION NUMBER 0
CALLS ARB3204(FUNC%,PARA%(0),ARY1%(0),ARY2%(0),ER%)
IF ER% <> 0 THEN
PRINT "Initialization error" : STOP
END IF
```

Example for C

```
int    para[10],ary1[10],ary2[10];
int    er=0;
main()

       para[0] = 0x300;          /* base address */
       para[ 1 ] = 2;            /* use IRQ2 */
       arb3204(0,para,aryl,ary2,&er);
       if( er != 0)
       {
              printf("Initialization error\n");
              exit(0);

}
```

3.3.2 Function 1 - SET MULTIPLEXER SCAN RANGE

Purpose

This function sets the multiplexer scan range to the range other than the default provided by Function 0.

Input Arguments

FUNC% = 1

PARA%(0) = The Lower Scan Channel. Valid number is 0 to 15.

PARA%(1) = The Upper Scan Channel. Valid number is 0 to 15.

Returns

ER% = 0 Good return.
= 1 Driver is not initialized.
= 2 Invalid function number.
= 7 Invalid scan range.

Programming Notes

1. Function 1 should be called pdorto performing AID conversion by function 3, 4, 5, 18 or 20, unless the default scan range set by Function 0 can, meet your requirements.

2. An action example of this function is as follows:
   Assume we set the lower limit to 1 and upper limit to 6. After every AID conversion, scanning of channels is always stepped sequentially in the order of: 1-2-3-4-5-6-1-2-3-4-5-6-etc.

3. It is possible to have the lower limit greater than the upper limit. For example, if the lower limit is 10 and the upper limit is 3, the scan limit will be: 10-11-12-13-14-15-0-1-2-3-10-etc.

4. If you want to perform AID conversion always on a single channel, set the lower limit and upper limit to the same channel that you desire. For example, setting both the lower limit and upper limit to 5 would perform conversions continuously on channel 5.

Example for Quick BASIC

```
DIM PARA%(0 TO 9)          'DECLARE DIMENSION
DIM ARY1%(0 TO 9)          'DECLARE DIMENSION
DIM ARY2%(O TO 9)          'DECLARE DIMENSION
```

```
PARA%(0) = 2                    'SET LOWER LIMIT
PARA%(1) = 8                    'SET UPPER LIMIT
FUNC% = 1                       'FUNCTION NUMBER 1
CALLS ARB3204(FUNC%, PARA%(0), ARY1 %(0), ARY2%(O),ER%)
IF ER% <> 0 THEN
PRINT "Set scan range fail" : STOP
END-LF.
```

Example for C

```
                para[0] = 2;                    r set lower limit `/
                para[1] = 8;                    /* set upper limit */
                arb3204(1, para,aryl ,ary2,&er);
                if( er != 0)

                   printf("Set scan range fail\n");
                   exit(0);
```

3.3.3 Function 2 - READ CURRENT M UX CHANNEL AND SCAN RANGE

Purpose

This function returns the next scan channel and the scan range setting.

Input Arguments

FUNC% = --2

PARA%() Not Significant.

Returns

PARA%(0) = Channel number for next conversion. This number will be a integer between 0 an 15.

PARA%(1) = Lower scan limit. This number will be a integer between 0 and 15.

PARA%(2) Upper scan limit. This number will be a integer between 0 and 15.

ER% = 0 Good return.
= 1 Driver is not initialized.
= 2 Invalid function number.

Example for Quick BASIC

```
DIM PARA%(0 TO 9)          'DECLARE DIMENSION
DIM ARY1%(0 TO 9)          'DECLARE DIMENSION
DIM ARY2%(0 TO 9)          'DECLARE DIMENSION
FUNC% = 2                  'FUNCTION NUMBER 2
CALLS ARB3204(FUNC%,PARA%(0),ARY1%(0),ARY2%(0),ER%)
IF ER% <> 0 THEN
      PRINT "Read scan range fail" : STOP
END IF
PRINT "NEXT CHANNEL TO BE CONVERTED IS ";
PRINT PARA%(0)
PRINT "THE LOWER SCAN LIMIT IS ";
PRINT PARA%(1)
PRINT "THE UPPER SCAN LIMIT IS";
PRINT PARA%(2)
```

Example for C

```
arb3204(2, para,aryl ,ary2,&er);
if( er != 0)

      printf("Read scan range fail\n");
      exit(O);

printf("next channel to be converted is %d\n", para[O]);
printf("the lower scan limit is %d\n",para(1));
printf("the upper scan limit is %d\n",para(2));
}
```

3.3.4 Function 3 - PERFORM A SINGLE AID CONVERSION

Purpose

This function performs a single AID conversion by software trigger. The channel number of the multiplexer is increased automatically by the driver after conversion.

¡nput Arguments

FUNC% = 3

PARA%() = Not Significant.

Returns

PARA%(0) = Converted data. This value will be the integer between -2048 and 2047.

PARA%(1) = Channel number. This value will be the channel number from which data is acquired. The possible number is between 0 and 15.

ER% = 0 Good return.
= 1 Driver is not initialized.
= 2 Invalid function number.
= 9 AID converter error (no EOC).

Programming Notes

1. The AID conversion will perform on channels according to the scan range set by Function 1. If Function 1 has not been called prior to this function, conversion will be cycled between the default scan range set by Function 0.

2. This function performs the conversion by software trigger. The sampling rate through this function is fairly slow and it is dependent upon the program execution speed.

I

Example for Quick Basic

```
DIM PARA%(0 TO 9)          'DECLARE DIMENSION
DIM ARY1%(0 TO 9)          'DECLARE DIMENSION
DIM ARY2%(0 TO 9)          'DECLARE DIMENSION
FUNC% = 3                  'FUNCTION NUMBER 3
CALLS ARB3204(FUNC %,PARA%~0),ARY1 %(0),ARY2%(0),ER%)
IF ER% <> 0 THEN
      PRINT "Conversion fail" : STO
END IF
PRINT "Channel";
PRINT PARA%(1);
PRINT "data =";
PRINT PARA%(0)
```

Example for C

```
arb3204(3,para,aryl,ary2,&er);
if( er != 0)
t
      pant€("Conversion fail\n");
      exit(O);

printf("Channel %d data = %din",pa~a[1],para[0]);
```

3.3.5 Function 4- PERFORM SPECIFED NUMBER OF CONVERSIONS VIA INTERRUPT AND TRANSFER DATA DIRECTLY TO AN ARRAY

Purpose

This function performs the specified number of A/D conversions via pacer or external triggered interrupt and ttansfers data to an integer array. Note the A/D conversion-performed-by this function -is-a foreground operation.

Input Arguments

FUNC% = 4

PARA%(0) = **Number of** conversions required. The valid number is between 1 and 32767.

PARA%(1) = Not significant.

PARA%(2) = Trigger source. IF PARA%(2) = 0, the external trigger source will be used. Conversion will take place on every positive edge of inppt pulse from I PO/ T'RGO. Exit from the call will not take placo until the specified pulses are reached. If PARA%(2) = 1, thO Programmable Timer will be used as a trigger source. Tho IPO/TRGO should be high to start the AID conversion. If IPO1TRGO is opened or connected to high, conversion wül $tart immediately. After IPO/TRGO goes high, IPO/TRGO will have no further effect. Exit occurs when the number of conversions are reached.

Returns

ARY1 %(0) = First converted datai word.

ARY1%(1) = Second converted data word.

ARY 1 %(2) = Third converted data word.

etc.

ER% = 0 Good return.
= 1 Driver is not initialized.
= 2 Invalid function mumber.
= 19 Invalid trigger source code(not 0 or 1).
= 20 Interrupt is active.
= 27 Conversion is terminated by keyboard hit.

Programming Notes

1. This function will exit only when **all** conversions have been completed. If the computer is hung up due to erroneous operation parameter settings like absence of trigger pulses, hit any key on the keyboard to terminate the operation.

2. This function performs A/D conversion in the foreground.- -If-you--do not wish to wait for data collection, use function 5 or 6 to collect data in the background.

3. Note that only data is stored in the array not the channel number. If you want to keep the channel number, use function 5 or 6.

4. The A/D will perform conversions on channels in accordance with the scan range set by Function 1. If Function 1 has not been executed before Function 4, conversions will scan from channel 0 to 15.

5. If the programmable interval timer is used as the trigger source, IPO acts as a starting gate to the operation. If an external trigger source is used, trigger pulses are applied to IPO, positive edges start conversions. By connecting Counter 0 output to IPO, it is possible to produce a programmed delay before conversions are performed using the programmable interval timer.

6. The size of ARY1 % should have a! least as many elements as the number of conversions specified in PARE%(0). If fewer elements of array are specified than the number of conversions, some of the data area will be overwritten, causing strange effects.

7. Use function 17 to set the conversion rate. The interrupt in the computer (mainly the timer interrupt) may divert the CPU's attention in order to transfer data from the A/D for several hundred microseconds.

Example for Quick Basic

```
DIM PARA%(0 TO 9)          'DECLARE DIMENSION
DIM ARY1%(O TO 9)          'DECLARE DIMENSION
DIM ARY2%(0 TO 599)        'DECLARE DIMENSION
FUNC% = 4                  'FUNC'T'ION NUMBER 4
PARA%(0) = 600             'NUMBER OF CONVERSION
PARA%(2)-= 1          _--  -'SELECT -PACER TRIGGER
CALLS ARB3204(FUNC%, PARA%(t),ARY1%(0),ARY2%(0),ER%)
IF ER% <> 0 THEN
      PRINT "Function 4 conversion fail" : STOP
END IF
FOR 1=0 TO 599
      PRINT ARY1%(I)
NEXT
```

Example for C

```
para[0] = 600;          /*number of conversion*/
para[2] = 1;            /*trigger from pacer*/
arb3204(4,para,aryl,ary2,&er);
if( er != 0)

      printf("Function 4 conversion fail1n");
      exit(0);

for(i=0; i<600;i++)
{
      printf("data = %d\n",ary1[i]);
}
```

3.3.6 Function 5-PERFORM SPECIFIED NUMBER OF CONVERSIONS VIA INTERRUPT AND TRANSFER DATA TO MEMORY

Purpose

This function performs the specified number of A/D conversions in background operation. At the end of each conversion, the interrupt service routine is invoked to trahsfer data to a specified memory segment. Conversions are repeated continuously under the re-cycle mode or terminated under the single cycle mode.

Input Arguments

FUNC% = 5

PARA%(O) = Number of conversions required. The valid number is between 0 and 32767. Specifiy 0 when 32768 conversions are required.

PARA%(1) = Segment of memory to receive data.

PARA%(2) = Trigger source. IF' PARA%(2) = 0, the external trigger source will be used. Conversion will take place on every positive edge of input pulse from IPO/TRGO and continue until the specified pulses are reached.

If PARA%(2) = 1, the Programmable Timer will be used as trigger source. The IPO/TRGO should be high to start the AID conversion. If IPO/TRGO is opened or connected to high, conversion will start immediately. After IP0/TRGO goes high, IPO/TRGO will have no further effect. Exit will not occur until IPO has been taken high.

PARA%(3) = Single Cycle/Re-cycle Mode Selection. IF PARA%(3) _ 0, than only one cycle will be performed. A cycle is reached when the specified number of conversions are completed. The interrupt service routine will remain active until this cycle has been complete.

If PARA%(3) = 1, recycle mode is selected. The operation will continuously wf to data to the specified memory area until Function 7 is called.

Returns

ER%	= 0	Good return.
	= 1	Driver is not initialized.
	= 2	Invalid function number.
	= 8	Invalid single/re-cycle mode.
	= 11	Invalid number of conversion.
	-=_19	Invalid trigger source code(not 0 or 1)..
	= 20	Interrupt is active.

Programming Notes

1. Since the Segment Registers are not increased by the interrupt service routine, the maximum data area available is 64K byte for 32768 conversions. Be sure that this data area is not used by your program or altered by subsequent operations.

2. The AID will perform conversions on channels according to the scan range set by Function 1. If Function 1 has not been executed before function 5, conversions will scan from channel 0 to 15

3. Use Function 8 to monitor the status and transfer progress.

4. Use Function 7 to terminate the re-cycle operation mode.

5. Use Function 9 to transfer data to user arrays.

6. Conversion rate is set by Function 17.

Example for Quick Basic

```
DIM PARA%(O TO 9)          'DECLARE DIMENSION
DIM ARY1%(0 TO 9)          'DECLARE DIMENSION
DIM ARY2%(0 TO 9)          'DECLARE DIMENSION
FUNC% = 5                  'FUNCTION NUMBER 5
PARA%(0) = 600             'NUMBIER OF CONVERSION
PARA%(1) = &H4000          'DATA MEMORY SEGMENT
PARA%(2) = 1               'SELECT PACER TRIGGER
PARA%(3) = 0               'SINGLE CYCLE MODE
CALLS ARB3204(FUNC%,PARA%(O),ARY1%(0),ARY2%(0),ER%)
IF ER% <> 0 THEN
      PRINT "Function 5 conversion tail" : STOP
END IF
'USE FUNCTION 9 TO TRANSFER DATA TO ARRAY
```

Example for C

```
    para[0] = 600;          /`number of conversion*/
    para[l] = 0x4000        /*data memory segment */
    para[2] = 1;            /*trigger from pacer*/
    para[3] = 0; _ _        /*single cycle mode*/
    arb3204(5,para,aryl,ary2,&er);
. - if( --er != 0)

          printf("Function 5 conversion failln");
          exit(0);

    /* Use Function 9 to transfer data to array */
```

3.3.7 Function 7 -TERMINATE INTERRUPT ACTIVITY

Purpose

This function terminate any running interrupt activity. This activity can be initiated by Function 5,18,or 20.

Input Arguments

FUNC% = 7

PARA%() = Not Significant.

Returns

ER% = 0 Good return.
= 1 Driver is not initialized.
= 2 Invalid function number.

Programming Notes

None.

Example for Quick Basic

```
DIM PARA%(0 TO 9)          'DECLARE DIMENSION
DIM ARY1%(0 TO 9)          'DECLARE DIMENSION
DIM ARY2%(0 TO 9)          'DECLARE DIMENSION
FUNC% = 7                  'FUNCTION NUMBER 7
CALLS ARB3204(FUNC %,PARA%(0),ARY1 %(0),ARY2%(0),ER%)
IF ER% <> 0 THEN
    PRINT "Function 7 termination fail" : STOP
END IF
```

Example for C

```
arb3204(7, para,aryl ,ary2,&er);
if( er != 0)

    printf("Function 7 termination fail\n");
    exit(0);
```

3.3.8 Function 8 - READ INTERRUPT STATUS

Purpose

This function allows user to checkthe status of interrupt operation initialized by Function 5,18 or 20.

Input Arguments

FUNC% 8

PARA%() = Not significant.

Returns

PARA%(0) = **Operation Type.** This will be an integer between 0 and 3 and is interpreted as follows:
0 None.
1 Function 5 interrupt.
2 Function 18 interrupt.
3 Function 20 interrupt.

PARA%(1) = **Operation Status**
0 Completed.
1 In progress.

PARA%(2) = **Number of Current Conversions.** The number returned states the number of conversions performed up to that point.

ER% = 0 Good return.
= 1 Driver is not initialized.
= 2 Invalid function number.

Programming Notes

None.

Example for Quick Basic

```
DIM PARA%(0 TO 9)          'DECLARE DIMENSION
DIM ARY1%(0 TO 9)          'DECLARE DIMENSION
DIM ARY2%(0 TO 9)          'DECLARE DIMENSION
FUNC% = 8                  'FUNCTION NUMBER 8
CALLS ARB3204(FUNC%,PARA%o(0),ARY1 %(0),ARY2%(0),ER%)
IF ER%-<> 0-THEN
      PRINT "Function 8 status read fail" : STOP
END IF
PRINT "FUNCTION :";PARA%(O)
PRINT "OPERATION STATUS :";PARA%(1)
PRINT "CONVERSION NUMBER :";PARA%(2)
```

Example for C

```
arb3204(8,para,aryl,ary2,&er);
if( er != 0)

      printf("Function 8 status read fail\n");
      exit(O);

printf("Function : %d\n",para[O]);
printf("Status of operation : %odln",para[1]);
printf("Conversion number : %d\n",para[2]);
```

3.3.9 Function 9 - Transfer Data from Memory to Array

Purpose

This function transfers data derived from Function 5 or 20 to integer array variables. Data is unpacked from packed form consisting of a word (2 bytes) of AID data plus the channel number.

Input Arguments

FUNC% = 9

PARA%(0) = Number of conversions required. The valid number is between G and 32767. Specify 0 when 32768 conversions are required.

PARA%(1) = Source Segment of memory to be transferred. This is the segment specified in Function 5 or 20.

PARA%(2) = Starting conversion number. This is an integer number between 0 and 32767.

Returns

ARY 1 %() = Converted data array.

ARY2%() = Channel array.

ER% = 0 Good return.
= I Driver is not initialized.
= 2 Invalid function number.
= 17 Invalid start conversion number.
= 18 Invalid number of conversion.

Programming Notes

1. The AID data is converted from offset binary to 2's complement. The range of returned data is between -2048 to 2047.

2. Be sure to allocate enough array variables to accommodate the transferred data.

Example for Quick Basic

```
DIM  PARA%(0 TO 9)          'DECLARE DIMENSION
DIM ARY1 %(0 TO 599)        'DECLARE DIMENSION
DIM ARY2%(0 TO 599)         'DECLARE DIMENSION
FUNC% = 9                   'FUNCTION NUMBER
PARA%(0)    600             'NUMBER OF CONVERSION
PARA%(1) = &H4000 .         'DATA_ MEMORY SEGMENT
PARA%(2) = 0                'STARTING CONVERSION NUMBER
CALLS ARB3204(FUNC%, PARA%(0),ARY1%(0),ARY2%(0),ER%)
IF ER% <> 0 THEN
      PRINT "Function 9 transfer fail" : STOP
END IF
FOR I = 0 TO 599
      PRINT USING "CHANNEL## =     #i#";ARY2%(I),ARY1%(I)
NEXT I
```

Example for C

```
para[0] = 600;                  /*number of conversion */
para[1] = 0x4000;               /*data memory segment */
para[2] = 0;            /*start conversion number*/
arb3204(9, para,aryl,ary2,&er);
if( er != 0)

      printf("Function 9 transfer fail\n");
      exit(0);

for( i=0; i<600; i++)
{
      printf("Channel %d = %d",aryl [i],ary2[i]);
```

3.3.10 Function 10 - SET COUNTER 0 OPERATION MODE

Purpose

This function established the Counter 0 operation mode. There are six operation mode available to Counter 0. These operation mode are discussed in detail in chapter 5.

Input Arguments

FUNC% = 10

PARA%(0) = Operation Mode Number. The valid number is between 0 to 5, where:

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

1 Programmable one-shot. Output will be initially high. Output will go low on the clock pulse following a trigger to begin the one-shot pulse, and will remain low until the counter reaches zero. Output will then go high and remain high until the clock pulse after the next trigger.

2. Rate generator. This mode function like a divide-by-N counter. Output pulses low on every terminal count.

3 Square wave mode. Output high for one half count, low for another.

4 Software triggered strobe. Output will be initially high. When the initial count expires, output will go low for one clock pulse and then go high again. The counting sequence is riggered by writing the initial count.

5 Hardware triggered strobe. Output will initially be high. Counting is triggered by a rising edge of GATE. When the initial count has expired, output will go low for one clock pulse and then go high again.

Returns

ER% = 0 Good return.
= 1 Driver is not initialized.
= 2 Invalid function number.
= 12 Invalid counter operation mode.

Programming Notes

1. The clock input source of Counter 0 is selected by JP2. Refer Section 2.4.2 for details of JP2.

Example for Quick Basic

```
FUNC% = 10' "          'FUNCTION NUMBER
PARA%(0)_= 2 _. ....'RATE_ GENERATOR MODE
CALLS ARB3204(FUNC%,PARA%(O),ARY1 %(O),ARY2%(O),ER%)
IF ER% <> 0 THEN
      PRINT "Function 10 set counter mode fail" : STOP
END IF
'USE FUNCTION 11 TO LOAD COUNT
```

Example for C

```
para[O] = 2;              /*rate generator mode */
arb3204(10,para,aryl,ary2,&er);
if( er != 0)

      printf("Function 10 set counter mode failln");
      exit(0);

ruse Function 11 to load count*/
```

3.3.11 FUNCTION 11 - WRITE COUNTER 0 DATA

Purpose

This function loads the counter 0 data.

Input Arguments

FUNC% = 11

PARA%(0) = **Counter 0 data.** The valid value that can be loaded ranges from -32768 to +32768 for actual values of 0 to 65535.

Returns

ER% = 0 **Good return.**
= 1 Driver is not initialized.
= 2 Invalid function number.

Programming Notes

1. Integer variables are signed 16-bit words (ranging between -32768 to +32767). To load a number above 32767, the integer data should be set to X-65536; for example, 40,000 would be entered as -25,536.

Example for Quick Basic

```
FUNC% = 11            'FUNCTION NUMBER
PARA%(0) = 600        'COUNTER DATA
CALLS ARB3204(FUNC%,PARA%(0),ARYI%(0),ARY2%(0),ER%)
IF ER% <> 0 THEN
      PRINT "Function 11 counter data load fail" : STOP
END IF
```

Example for C

```
para[0] = 600;                    /*counter data*/
arb3204(11,para,ary1,ary2,&er);
if( er != 0)

      printf("Function 12 counter data load faii\n");
      exit(0);
```

3.3.12 FUNCTION 12 - READ COUNTER 0 DATA

purpose

This function reads the counter 0 data.

Input Arguments

FUNC% = 12

PARA%(O) = Mode of read operation. Two modes of read operation are available.

If PARA%(O) = 0, the straight read operation for static data is performed. This can only be used when counter data is known as static (while gate input is low or there is no more clock input).

If PARA%(0) = 1, then a latch read operation is performed. This type of read can be used if you want to read the counter on the fly. Neither type of read will affect the counter or operation.

Returns

PARA%(0) = Unchanged.

PARA%(1) = Read data. The read data is in a 16-bit integer format and ranges from -32768 to +32767. This corresponds to actual values from 0 to 65535.

ER% = 0 Good return.
= 1 Driver is not initialized.
= 2 Invalid function number.
= 16 Invalid read operation mode.

Programming Notes

1. If the returned counter data is less than 0, the actual counter data is PARA%(1)+65536. For example, if -9,000 is returned, it corresponds to 65,536-9,000=56536.

Example for Quick Basic

```
FUNC% = 12                    'FUNCTION NUMBER
PARA%(0) = 1                  'LATCH READ OPERATION
CALLS ARB3204(FUNC%,PARA%(0),ARY1%(0),ARY2%(0),ER%)
IF ER% <> 0 THEN
      PRINT "Function 12 counter data read fail" : STOP
END IF
PRINT "COUNTER 0 DATA= ";PARA%(1)
```

Example for C

```
para[O] = 1;              /Match read mode*/
arb3204(12,para,ary1,ary2,&er);
if(er != 0)

      printf("Function 12 counter data read fail\n");
      exit(0);

printf("Counter 0 data = %d\n",para[1]);
```

3.3.13 FUNCTION 13 - WRITE DIGITAL OUTPUT OPO-OP3

Purpose

This function writes data to the digital output channel OPO to OP3.

Input Arguments

FUNC% = 13

PARA%(1) = Output data. Output data must be in the range of 0 to 15.

Returns

PARA%() = Unchanged.

ER% = 0 Good return.
= 1 Driver is not initialized
= 2 Invalid function number.
= 13 Invalid output data.

Programming Notes

None.

Example for Quick Basic

```
FUNC% = 13                    'FUNCTION NUMBER
PARA%(0) = 6                  'OUTPUT DATA
CALLS ARB3204(FUNC %,PARA%(O),ARY1%(O),ARY2%(O),ER%)
IF ER% <> 0 THEN
      PRINT "Function 13 digital output fail" : STOP
END IF
```

Example for C

```
para[0] = 6;              /* Output data */
arb3204(13, para,ary1,ary2, &er);
if(6er != 0)

      printf("Function 13 digital output faihn");
      exit(O);
```

3.3.14 FUNCTION 14 - READ DIGITAL INPUT 1P0-1P6

Purpose

This function reads the status of digital input channel IPO to IP6.

Input Arguments

FUNC% = 14

Returns

PARA%(0) = Input data. Range is between 0 and 128.

ER% = 0 Good return.
1 Driver is not initialized
2 Invalid function number.

Programming Notes

None.

Example for Quick Basic

```
FUNC% = 14                    'FUNCTION NUMBER
CALLS ARB3204(FUNC%, PARA%(0),ARY1 %(0),ARY2%(0),ER%)
IF ER% <> 0 THEN
      PRINT "Function 14 digital input fail" : STOP
END IF
PRINT "Digital input -- ";PARA%(O)
```

Example for C

```
arb3204(14, para,aryl ,ary2,&er);
if( er != 0)

      printf("Function 14 digital input fail\n");
      exit(O);

printf("Digital input -- %d",para[O]);
```

3.3.15 FUNCTION 15 - OUTPUT DATA TO DIA CHANNEL

Purpose

This function writes data to the D/A channel.

Input Arguments

FUNC% 15
PARA%(0) = **D/A data.** Valid value ranges from 0 to 4095.

Returns

ER% = 0 **Good return.**
= 1 Driver is not initialized
= 2 Invalid function number.
= 14 Invalid D/A data.
= 15 Invalid D/A channel number.

Programming Notes

None.

Example for Quick Basic

```
FUNC% = 15          'FUNCTION NUMBER
PARA%(0) = 1024 'D/A data
CALLS ARB3204(FUNC%, PARA%(0),ARY1%(0),ARY2%(0),ER%)
IF ER% <> 0 THEN
      PRINT "Function 15 D/A data write fail" : STOP
END IF
```

Example for C

```
para[0] = 1024;                  /* D/A data */
arb3204(15,para,aryl,ary2,&er);
if( er != 0)

      printf("Function 15 D/A data write fail\n");
      exit(0);
```

3.3.16 FUNCTION 17 - SET PACER TRIGGER RATE

Purpose

This function sets the AID pacer rate by specifying the divisor for Counter 1 and Counter 2 of the 8253 counter timer. The default divisors (set by function 0) are 2000.

Input Arguments

FUNC% = 17

PARA%(0) = - Counter 2 divisor. Valid divisor may range between 2 and 65535.

PARA%(1) = Counter 1 divisor. Valid divisor may range between 2 and 65535.

Returns

ER%	= 0	Good return.
	= 1	Driver is not initialized.
	= 2	Invalid function number.
	= 10	Invalid counter devisor.

Programming Notes

1. The possible pacer rates range from 500KHz to 0.000466 Hz with the internal 2MHz clock.

The counter input clock frequency is 2MHz.

Pacer rate = 2MHz / (D1 * D2)

Where:

D1 = Counter 1 divisor
D2 = Counter 2 divisor

The timer output can be viewed on an oscilloscope between Pins 30(+) and 37(-) of CN 1.

Example for Quick Basic

```
FUNC% = 17                  'FUNCTION NUMBER
PARA%(0) = 50               'COUNTER 1 DIVISOR
PARA%(1) = 20               'COUNTER 2 DIVISOR
CALLS ARB3204(FUNC%,PARA%(0),ARY1%(0),ARY2%(0),ER%)
IF ER% <> 0 THEN
    PRINT "Function 17counter divisor --write fail"---
STOP
END IF
```

Example for C

```
para[0] = 50;           /*counter 1 divisor*/
para[ 1 ] = 20;         /*counter 2 divisor*/
arb3204(17,para,ary1,ary2,&er);
if( er != 0)
(
    printf("Function 17 counter divisor write fail\n");
    exit(0);
```

3.3.17 Function 98 - Perform Analog Output on Interrupt

Purpose

This function transfers data from a previous prepared integer array to the D/A converter upon interrupt. Optionally, it transfers AID data generated along with each interrupt to another array. This enables user to perform stimulus/response- type- of testing where D/A outputs a signal and AID measures its result.

Input Arguments

FUNC% = 18

PARA%(0) = Number of conversions. Valid value may range between 0 and 32767. Specify 0 when 32768 conversions is desired.

PARA%(1) = Number of cycle to transfer. Valid value may range between 0 and 65535. Specify 0 when continuous operation is desired. Continuous transfer is useful for waveform generation.

ARY1 %() = D/A data array.

Returns

ARY2%() = A/D data array.

ER% = 0 Good return.
= 1 Driver is not initialized.
= 2 Invalid function number.
= 18 Invalid number of conversions.
= 20 Interrupt is active.

Programming Notes

1. The following functions perform Function 18 related activities.

   Function 7 Terminate interrupt.
   Function 8 Check interrupt status.
   Function 17 Set pacer rate.

2. The transfer rate is dependent on the rate at which interrupt can be processed by your computer, and varies according to the type of hardware and CPU. Due to the use of interrupts, D/A update has some variable latency delays of several microseconds. These can be improved by suppressing other higher priority internal interrupts such as the PC timer interrupt on level 0.

3. The IPO/TRGO should be high to start the conversion. If IPO/TRGO is opened or connected to high, conversion will start immediately. After IPO/TRGO goes high, IPO/TRGO will have no further effect. Exit will not occur until IPO has been taken high.

Example for Quick Basic

```
'Create a sine wave data array
DIM ARY1 %(0 TO 49),ARY2%(0 TO 49)
FOR 1% =0T049
      ARY1 %(I) = CINT(2048+2047*SIN(3.14*1/25))
NEXT I
FUNC% = 18                    'FUNCTION NUMBER
PARA%(0) = 50                 'NUMBER OF CONVERSION
PARA%(1) = 0                  'CONTINUOUS OPERATION
CALLS ARB3204(FUNC%,PARA%(O),ARY1%(0),ARY2%(0),ER%)
IF ER% <> 0 THEN
      PRINT "Function 18 analog 1/O fail":STOP
END IF
```

Example for C

```
/*Create a sine wave data array*/
for (i=0; i<50;i++)
      aryl[i] = (2048+2047*sin(3.14*i/25));
para[0] = 50;          /*number of conversion*/
para[1] = 0;           /*continuous operation*/
arb3204(18,para,ary1,ary2,&er);
if( er != 0)

      pnntf("Function 18 analog I/O failln");
      exit(0);
```

3.3.18 ***FUNCTION 19 - WAIT ANALOG TRIGGER FUNCTION***

Purpose

This function performs a waiting operation for the analog input to reach a certain level with a specified slope. User may choose any input channel as the trigger channel and select the trigger level and slope. It is useful as a gate or -wait loop in front-of-any of the other data acquisition. functions.

Input Arguments

FUNC% = 19

PARA%(0) = Trigger channel number. Valid values are within the range 0 to 15.

PARA%(1) = Trigger level. This integer value selects the voltage level at which triggering occurs. The valid values range from -2048 to 2047.

PARA%(2) = Slope. This selects the direction of triggering, 0 for positive slope or 1 for negative. For example, suppose PARA%(1) = 1024 when input range is +/-10v and slope of trigger is positive (PARA%(2) = 0). Triggering would take place when the trigger signal exceeds +2.5v.

Returns

ER% = 0 Good return.
= 1 Driver is not initialized.
= 2 Invalid function number.
= 20 Interrupt is active.
= 23 Invalid trigger channel number.
= 24 Invalid trigger level.
= 25 Invalid trigger direction.
= 27 Terminate by keyboard hit.

Programming Notes

1. The trigger channel 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.

2. If the trigger conditions are not fulfilled, it is possible to get stuck in the wait loop indefinitely. If this happens, pressing any key of your keyboard will return to the calling program and generates an error code.

Example for Quick Basic

```
FUNC% = 19          'FUNCTION NUMBER
PARA%(0) = 3        'AID CHANNEL 3 TO BE SAMPLED
PARA%(1) = 1024 'TRIGGER LEVEL
PARA%(2) = 0        'POSITIVE TRIGGER
CALLS ARB3204(FUNC%,PARA%(0),ARY1%(0),ARY2%(0),ER%)
IF ER% <> 0 THEN_
    PRINT "Function 19 analog trigger fail":STOP
END IF
```

Example for C

```
para[0] = 3;            /*channel 3 to be sampled*/
para[1] = 1024;         /*trigger level*/
para([2] = 0            /*positive trigger*/
arb3204(19, para,ary1,ary2,&er);
if( er != 0)

    printf("Function 19 analog trigger fail\n");
    exit(0);
```

3.3.19 FUNCTION 20 - SCAN AID CHANNELS ON INTERRUPT

Purpose

This function performs a complete scan of analog input channels as defined by the last call to Function 1 for each conversion clock signal.

Input **Arguments**

FUNC% = 19

PARA%(O) = Number of conversions required. Valid values are within the range 0 to 32767. Specify 0 if you want 32768 conversions.

PARA%(1) = Segment address of data array. The interrupts load AID data from the beginning of this segment.

PARA%(2) = Trigger source. This selects the source of triggering, 0 for external trigger or 1 for pacer trigger.

If external trigger is selected, every high pulse on IPO/TRGO will start a scan and interrupt continue until the word count is reached in single cycle mode or indefinitely in continuous mode.

If internal trigger is selected, IPO/TRGO must be high to start the interrupts. After the first scan is triggered, level of IPO/TRGO will have no further effect. Exit will not occur until I PO/TRGO goes high.

PARA%(3) = Single Cycle/Re-cycle Mode Selection. IF PARA%(3) _ 0, then only one cycle will be performed. A cycle is reached when the specified number of conversions are completed. The interrupt service routine will remain active until this cycle is complete.

If PARA%(3) = 1, re-cycle mode is selected. The operation will continuously write data to the specified memory area until Function 7 is called.

Returns

ER% = 0 Good return.
= 1 Driver is not initialized.
= 2 Invalid function number.
= 8 Invalid single/recycle mode
= 11 Invalid number of conversion

= 19 Invalid trigger source code(not 0 or 1)
= 20 Interrupt is active.

Programming Notes

1. Block scan rates are limited and may produce variable delays in the scan rate due to the execution delays of the interrupt service routine. Try to suppress other interrupts _(especially the PC timer interrupt) if delays are critical to your application.

2. The maximum scan rate also depends on the number of channels scanned. If you begin to miss channels of foreground process slow, you are operating beyond or close to the maximum interrupt rate.

3. Use Function 8 to monitor the status and transfer progress.

4. Use Function 9 to transfer data from memory to user array.

Example for Quick Basic

```
DIM PARA%(0 TO 9)          'DECLARE DIMENSION
DIM ARY1%(0 TO 9)          'DECLARE DIMENSION
DIM ARY2%(0 TO 9)          'DECLARE DIMENSION
FUNC% = 20                 'FUNCTION NUMBER
PARA%(0) = 600             'NUMBER OF CONVERSION
PARA%(1) _ &H4000          'DATA MEMORY SEGMENT
PARA%(2) = I               'SELECT PACER TRIGGER
PARA%(3) = 0               'SINGLE CYCLE MODE
CALLS ARB3204(FUNC%,PARA%(0),ARY1%(0),ARY2%(0),ER%)
IF ER% <> 0 THEN
      PRINT "Function 20 block scan fail" : STOP
END IF
'USE FUNCTION 9 TO TRANSFER DATA TO ARRAY
```

Example for C

```
para[0] = 600;       /`number of conversion*/
para[1] = 0x4000 /*data memory segment */
para[2] = 1;         /*trigger from pacer*/
para[3] = 0;         /*single cycle mode*/
arb3204(20,para,ary1,ary2,&er);
if( er != 0)

   printf("Function 20 block scan failln");
   exit(0);

/* Use Function 9 to transfer data to array */
```

3.3.20 Function 21 - SET GAIN OF AN AID CHANNEL

Purpose

This function set the gain of of a specified AID channel.

Input Arguments

FUNC% --. _ 21

PARA%(0) = Channel number: This data specifies the channel number whose gain is going to be set. It can be any number between 0 and 15.

PARA%(1) = Gain code. Gain code are as follows:

G2	GI	GO	Gain	Input Range
0	0	0	1	-5V to +5V
0	0	1	2	-2.5V to +2.5V
0	1	0	4	-1.25V to +1.25V
0	1	1	8	-625mV to +625mV
1	0	0	16	-312.5mV to 312.5mV

Returns

ER% = 0 Good return.
= 1 Driver is not initialized.
= 2 Invalid function number.
= 28 Invalid A/D channel number.
= 29 Invalid gain code.

Programming Notes

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

Example for Quick Basic

```
FUNC% = 21                  'FUNCTION NUMBER
PARA%(1) = 2                'GAIN = 4.
FOR i% = 3 TO 5             'Channel 3 to 5
    PARA%(0) = 1%           'Channel number
    CALLS ARB3204(FUNC%,PARA%(O),ARY1%(O), ARY2%(0),ER%)
NEXT I
PARA%(1) = 4                'GAIN = 16.
FOR I% = 6 TO 15            'Channel 6 to 15
    PARA%(O) = 1%           'Channel number
    CALLS ARB3204(FUNC%,PARA%(0),ARYI%(0), ARY2%(O),ER%)
NEXT 1%
IF ER% <> 0 THEN
    PRINT "Function 21 gain setting fail" : STOP
END IF
```

Example for C

```
para[1] = 2;            /`gain = 4*/
for(i=3;i<=5;i++)       /* channel 3 to 5 */
{
    para[0] = i;        /* channel number */
    arb3204(21,para,aryl,ary2,&er);


para[ 1 ] = 4;          /*gain = 16*/
for(i=6;i<=15;i++)      /* channel 6 to 15 */

    para[O] = i;        /* channel number */
    arb3204(21,para,aryl,ary2,&er);
}

if( er != 0)

    printf("Function 21 gain setting fail\n");
    exit(O);
```

3.3.21 Function 22 - SQUARE WAVE OUTPUT ON COUNTER 0

Purpose

This function provides an easy way to output TTL level square wave on AR-B3204- This is achieved by programming the Counter 0 to connect its clock input to -the internal 2MHz clock source and to operate in square wave output mode.

Input Arguments

FUNC% = 22

PARA%(0) = Counter divider. This is an integer in the range of 2 to 65536.

Returns

ER% = 0 Good return.
= 1 Driver is not initialized.
= 2 Invalid function number.

Programming Notes

1. The GATEO(pin 36) input must be high or left open to output square wave on counter 0 out (pin 17).

2. The frequency of square wave is set as:

Frequency = 2,000,000IPARA%(0) Hz

Since this function is performed on Counter 0, this output can be obtained regardless of other operations such as AID, D/A, or digital I/O.

Example for Quick Basic

```
FUNC% = 22          'FUNCTION NUMBER
PARA%(0) = 500      'COUNTER DIVISOR
CALLS ARB3204(FUNC%,PARA%(0),ARY1%(0),ARY2%(O),ER%)
IF ER% <> 0 THEN
      PRINT "Function 22 square wave output fail" : STOP
END IF
```

Example for C

```
para[0] = 500;          /*counter divisor*/
arb3204(22,para,aryl,ary2,&er);
if( er != 0)

      printf("Function 22 square wave output fail\n");
      exit(0);
```

3.3.22 Function 23- READ AID CHANNEL GAIN CODE

Purpose

This function returns the gain code of the specified A/D channel.

Input Arguments

FUNC% 23

PARA%(0) = Channel number. The valid channel number is in the range of 0 to 15.

Returns

PARA%(1) = Gain code. The possible gain code is between 0 and 4. Gain codes and their respective gains and input range are as follows:

Gain Code	Gain	Input Range
0	1	-5V to +5V
1	2	-2.5V to +2.5V
2	4	-1.25V to +1.25V
3	8	-625mV to +625mV
	16	-312.5mV to 312.5mV

ER% = 0 Good return.
= 1 Driver is not initialized.
= 2 Invalid function number.
= 28 Invalid A/D channel number.

Programming Notes

None.

Example for Quick Basic

```
FUNC% = 23                    'FUNCTION NUMBER
FOR 1=0 TO 15                 'CHANNEL 0 TO 15
    PARA%(O) = I              'CHANNEL NUMBER
    CALLS ARB3204(FUNC%,PARA%(0),ARYI%(0), ARY2%(0),ER%)
    PRINT USING "CHANNEL## GAIN CODE =#";I,PARA%(1)
NEXT I
IF ER% <> 0 THEN
    PRINT "Function 23 gain code read fail" : STOP
END IF
```

Example for C

```
for(I=O;i<=15;i++)      ¡channel 0 to 15

        para[O] = ¡;        /*channel number*/
        arb3204(23, para,ary 1,ary2,&er);
        printf("channel %d gain code = %din",i,para[1]);

¡f(er != 0)

        printf("Function 23 gain code read fail\n");
        exit(O);
}
```

3.3.23 Function 100 - READ DRIVER VERSION DATE CODE

Purpose

This function returns the version date code of the AR-B3204 driver.

Input Arguments

FUNC% 100

PARA%() = Not Significant.

Returns

PARA%(0) = Date code. The date code of the AR-B3204 driver. It is a four digit integer. The first 2 digits represent the year and the last two digit represent the month. For example, 9308 represents 1993 August.

ER% = 0 Good return.
= 1 Driver is not initialized.

Programming Notes

None.

Example ***for Quick Basic***

```
FUNC% = 100                    'FUNCTION NUMBER
CALLS ARB3204(FUNC%, PARA%(0),ARY1%(0),ARY2%(0),ER%)
PRINT USING "DRIVER DATE CODE IS ####";PARA%(0)
IF ER% <> 0 THEN
      PRINT "Function 100 READ DATE CODE FAIL" : STOP
END IF
```

Example for C

```
arb3204(100,para,aryl,ary2,&er);
printf("Driver date code is %d\n",para[0]);
if( er != 0)
f
      printf("Function 100 read date code fail\n");
      exit(0);
```

3.4 PROGRAMMING EXAMPLES

To assist the user familiar with programming the AR-B3204 with provided driver routines, some example programs are offered. Each program covers an important application area and comes with complete program remarks. Following is a list of example programs provided in the AR-B3204 software diskette.

Table 3-2 Example program list

File name	Description
SAMPLE01. BAS	Performs software trigger AID by dirrect I/O access.
SAMPLE02.BAS	Performs software trigger AID by using Function 3.
SAMPLE03.BAS	Performs AID conversions by pacer trigger and transfer data via interrupt. Uses Function 4.
SAMPLE04.BAS	Performs A/D conversions by pacer trigger and transfer data via interrupt. Uses Function 5.
SAMPLE05.BAS	Performs A/D block channel scan on pacer trigger and transfer data via interrupt. Uses Function 20.
SAMPLE06.BAS	Waits AID trigger level then performs AID conversions via interrupt. Use function 19 and 5.
SAMPLE07.BAS	Waits AID trigger level then performs A/D conversions via interrupt. Use function 19 and 5.
SAMPLE08.BAS	Performs outputs to D/A channel by using Function 15.
SAMPLE09.BAS	Generate sine waves by transfer D/A data via interrupt. Uses Function 18.
SAMPLE10.BAS	Performs digital I/O by using Function 13 and Function 14.

CHAPTER 4 REGISTER FORMAT AND STRUCTURE

This chapter describes the register level programming of AR-B3204. To program AR-133204's lowlevel register directly, it is important to understand the register format and architecture of AR-B3204.

The following registers are discussed in detail in this chapter.

- AID Data Registers
- D/A Output Registers
- Digital 1/O Registers
- AID Status Register
- A/D Gain Control Register
- Multiplexer Register
- Control Register
- Software Trigger Register
- Programmable Timer/Counter Registers

4.11/0 PORT ADDRESS MAP

The AR-B3204 uses a 16 continuous I/O address space. Table 4-1 shows the location of each register relative to the base address.

Table 4-1- 1/O Port Address Map

Location	Read Function	Write Function
Base + 0	A/D Low byte	D/A low byte
Base + 1	A/D High byte	D/A high byte
Base + 2	Digital input	Digital output
Base + 3	Status	Clear interrupt
Base + 4	N/A	Gain control
Base + 5	N/A	Multiplexer control
Base + 6	N/A	Control register
Base + 7	N/A	A/D software trigger
Base + 8	Counter 0	Counter 0
Base + 9	Counter 1	Counter 1
Base + 10	Counter 2	Counter 2
Base + 11	N/A	Counter control
Base + 12	NIA	N/A
Base + 13	N/A	N/A
Base + 14	N/A	N/A
Base + 15	N/A	N/A

4.2 AID DATA REGISTERS (READ BASE+O & +1)

The A/D data registers use addresses of Base+O and Base+1. Previous converted data is latched and available until the end of the next conversion. The data format is as follows:

Baas+O

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

Base+I

D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	AD11	AD10	AD9	AD8

AD11 to ADO Analog to digital data. ADO is LSB and AD11 is MSB. Data is stôred as offsetbinary." Since AR-B3204 is operated in bipolar mode, for the convenience of the computer's calculation, offset binary should be converted to strait binary by minus 2048.

X Don't care.

4.3 DIA OUTPUT REGISTERS (WRITE BASE+O,+1)

D/A output registers are output only registers. They are double buffered so that they are not updated until the second(high) byte is written. The D/A data is in binary format as follows:

BASE+0

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

BASE+1

D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	DA11	DA10	DA9	DA8

DA11 TO DAO DIA output data. DAO is LSB and DA11 is MSB.
X Don't care.

4.4 DIGITAL 1/O REGISTERS (BASE+2)

The digital 1/O consists of a 7-bit input port(IPO-6) and a 4-bit output port(OPO-3). They use the same I/O address but are independent. Data written to the digital output will not affect the digital input data and can not be read back unless they are connected externally.

Digital input IPO also acts as TRIGO which is the external trigger of A/D conversion.

Data format of digital input register:

BASE+2

D7	D6	D5	D4	D3	D2	D1	D0
0	DI6	DI5	DI4	DI3	DI2	DI1	DI0

D16 to DIO **Digital input data.**

Data format of digital output register:

BASE+2

D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	DO3	DO2	DO1	DO0

D03 to **DOO** **Digital output data.**

X Don't care.

4.5 AID STATUS & CLEAR INTERRUPT REGISTER (BASE+3)

Reading from the AID status register provides information on the operation of AID in the AR-B3204. Write operation to the status register address with any data value clears the interrupt request and acknowledging the interrupt, and re-enabling it. The format of this register is described as follows:

BASE+3

D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	0	0	INT	EOC

EOC End of conversion.

EOC = 0 means the A/D converter is in the idle state and is ready for the next conversion. The AID data registers contain valid data of previous conversion.

EOC =1 means the A/D converter is busy and the A/D conversion is in progress.

INT — Interrupt request flag. This bit will beset to high wï yen the occurence of EOC for AID converter. If the INTE bit of control register is set, an interrupt signal will be sent to the PC bus interrupt level IRQn where n is specified by bit 12, 11 and 10 of control register. Your program interrupt service routine that services this interrupt must clear this bit by writing any data to this status register so that subsequent interrupts may be detected.

X — **Don't care.**

4.6 GAIN CONTROL REGISTER (BASE+4)

An 8-bit write only register that sets the gain of the programmable gain amplifier. Only the bottom three bits are used, the upper five bits have no effect on the hardware. The Programmable Gain register is cleared by power and hardware reset.

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

BASE+4

D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	X	G2	G1	G0

G2 to GO — Gain Code Select Bits. These are used to select the gain as follows:

G2	G1	GO	Gain	Input Range
0	0	0	1	-5V to +5V
0	0	1	2	-2.5V to +2.5V
0		0	4	-1.25V to + 1.25V
0	1	1	8	-625mV to +625mV
1	0	0	16	-312.5mV to 312.5mV

4.7 MUX CONTROL REGISTER (BASE+5)

The MUX Control register is a write only register located at BASE+5. It controls the AID channel input multiplexer. The low nibble selects the analog input channel and the high nibble has no effect to the AR-B3204 hardware. Data is stored in binary format as follows:

BASE+5

D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	MX3	MX2	MX1	MX0

MX3 to MXO Analog input channel.

4.8 CONTROL REGISTER (BASE+6)

This Control Register is a read/write register that controls the operation mode of the board. The format of this register is described as follows:

BASE+6

D7	D6	D5	D4	D3	D2	D1	D0
INTE	I2	I1	I0	X	X	ST1	ST0

INTE Interrupt enable. This bit enables or disables the AR-B3204 generated interrupt.:

INTE = 0 disables the generated interrupt request signal to be sent to the PC bus.

If INTE = 1, an interrupt is generated at the end of each conversion when data is available. This is used when data is transferred by interrupt. .

12 to 10 Interrupt level selection. These three bits select the desired interrupt level that will be used.

Possible levels are as follows:

12	11	10	Innterrupt Level
0	0	0	Disable
0		1	Disable
0	1	0	2
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Do not use a level which is already used by another interface card.

X These bits are not significant.

ST1 & STO Source of trigger. These two bits select the trigger clock source of AID conversion. as described below.

The control register is cleared on power up or hardware reset of your PC, thereby disabling interrupt.

4.9 SOFTWARE TRIGGER REGISTER (BASE+7)

When S 1 of control register is 0, a write to Base+7 acts as an A/D trigger and initiate a conversion. Any data written to this address is lost.

4.10 PROGRAMMABLE TIMER/COUNTER REGISTER (BASE+8,+9, +10 & +11)

These four registers correspond to the four registers of the 8253 programmable timer/counter. Please refer to the Intel 8253 data sheet or Chapter 5 for detail descriptions.

BASE+12	Counter 0 data register	Read/write
BASE+13	Counter 1 data register	Read/write
BASE+14	Counter 2 data register	Read/mite
BASE+15	8253 control register	Write only

This page intentionally left blank

CHAPTER 5 PROGRAMMABLE TIMERICOUNTER

The AR-B3204 uses the Intel 8253 programmable interval timer. It is a very flexible timer/counter device consisting of three independent 16 bit down counters. Each counter has a clock input, control gate and an output that can be programmed to count in the range of 2 to 65535. The maximum clock input frequency of 8253 is 4 MHz..

Counter 1 and counter 2 on the AR-B3204 are cascaded. Counter 2 input is connected to a precision 2 MHz crystal oscillator. The output of counter 2 is connected to the clock input of counter 1. This makes a 32-bit wide timer/ counter. The output of counter 1 is internally configured to provide A/D trigger pulses and also routed to C 1 OUT (pin 30).

Counter 0 is uncommitted and its input, output, and gate control are available to the user on CLKO(pin 18), OUTO(pin 17) and GATEO(pin 36).

5.1 COUNTER CONTROL REGISTER

The 8253 programmable interval timer occupies four I/O locations in the AR-B3204 110 address map as shown below.

BASE+8	Counter 0 data register	Read/write
BASE+9	Counter 1 data register	Read/write
BASE+10	Counter 2 data register	Read/write
BASE+11	Counter Control register	Write only

As power-up, the state of 8253 is undefined. The mode, count value, and output of all counters are undefined.

The method of operation for each counter is determined when it is programmed. Each counter must be programmed before it can be used. Unused counters need not be programmed.

Counters are programmed by writing a control word to the control register and then writing initial count to counter data register.

The control word format is:

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

Where

SC Select Counter. Select the counter to be used.

SCI	SCO	Selection
0		Counter 0
		Counter 1
1	0	Counter 2
1	1	Read-Back Command (see sec

RW ReadIV kite. Select the Read/Write operation.

RW	RW	Operation
0	0	Counter Latch Command (see
0	1	Read/Write LSB only
1	0	Read/Write MSB only
1	1	Read/Write LSB first, than MSE

M Mode. Select the Operation Mode.

M2	M1	MO	Operation
0	0	0	0 - Pulse on Terminal Count
	0	1	1 - Programmable One Shot
X	1	0	2 - Rate Generator
X	1	1	3 - Square Wave Generator
1	0	0	4 - Software Triggeed Strobe
1	0	1	5 - Hardware Triggered Strobe

BCD BCD/Binary. Select binary or BCD counting.

BCD	Operation
0	Binary Counter 16-bit
1	Binary Code Decimal (BCD)

If the modules are set to be binary, the count can be any number from 0 up to 65535. If the modules are set to be BCD, the count can be any number from 0 to 9999.

5.2 COUNTER OPERATION MODE

The Operation Mode of 8253 is described below

5.2.1 MODE 0 - PULSE ON TERMINAL COUNT.

After the control word is written, output is initially low, and will remain low until the counter reaches zero. Output then goes high and remains high until a new count or a newcontrol word is written into the counter. If a newcount is written to the counter, it will be loaded on the next clock pulse and counting will continue from the new count. If a two-byte count is written, the following will happen:

1) Writing the first byte disables counting. Output is set low immediately.
2) Writing the second byte allows the new count to be loaded on the next clock pulse.

This allows the counting sequence to be synchronized by the software.

5.2.2 MODE 1 - PROGRAMMABLE ONE-SHOT.

Output will be initially high. Output will go low on the clock pulse following a trigger to begin the one-shot pulse, and will remain low until the counter reaches zero. Output will then go high and remain high until the clock pulse after the next trigger. After writing the control word and initial count, the counter is armed. A trigger results in loading the counter and setting output lowon the next clock pulse, thus starting the one-shot pulse. A initial count of N will result in a one-shot pulse N clock cycles in duration. The one-shot is retriggerable, hence output will remain lowfor N clock pulses after any trigger. The one-shot pulse can be repeated without rewriting the same count into the counter.

5.2.3 MODE 2 - RATE GENERATOR

The output goes low for one input. clock period on every N counts, where N is the count loaded. When the gate input is 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 pulses to trigger AID conversions.

5.2.4 MODE 3 - SQUARE WAVE GENERATOR

This mode is similar to mode 2 except for the duty cycle of output. Output will initially be high. When half the initial count has expired, output goes low for the remainder of the count. Mode 3 is periodic; the sequence above is repeated indefinitely. An initial count of N results in a square wave with a period of N clock cycles.

5.2.5 MODE 4 - SOFTWARE TRIGGERED STROBE

Output will be initially high. When the initial count expires, output will go low for one clock pulse and then go high again. The counting sequence is "triggered" by writing the initial count. GATE = 1 enables counting; Gate = 0 disables counting. After writing a control word and initial count, the counter will be loaded on the next clock pulse. This clock pulse does not decrease the count, so for an initial count of N, output does not strobe low until N + 1 clock pulses after the initial count is written. If a two-byte count is written, the following will happen:

1) Writing the first byte has no effect on counting.
2) Writing the second byte allows the new count to be loaded on the next clock pulse.

This allows the sequence to be "retriggered" by the software. Output strobes low N + 1 clock pulses after the new count of N is written.

5.2.6 MODE 5 - HARDWARE TRIGGERED STROBE (RETRIGGERABLE)

Out will initially be high. Counting is triggered by a rising edge of GATE. When the initial count has expired, output will go low for one clock pulse and then go high again. After writing the control word and initial count, the counter will not be loaded until the clock pulse after a trigger. This dock pulse does not decrease the count, so for an initial count of N, output does not strobe low until N+1 clock pulses after a trigger. A trigger results in the counter being loaded with the initial count on the next dock pulse. The counting sequence is retriggerable. Output will not strobe low for N + 1 clock pulses after any trigger.

5.3 WRITE OPERATION

The programming procedure for the 8253 is very flexible. Only two conventions need to be remembered:

(1) For each counter, the control word must be written before the initial count is written.

(2) The initial count must follow the count format specified in the control word (least significant byte only, most significant byte only, or least significant byte and then most significant byte).

Since the control word register and three counters have separate addresses, and each control word specifies the counter it applies to, no special instruction sequence is required. Any programming sequence that follows the conventions in Section 5.1 is acceptable.

A new initial count may be written to a counter at any time without affecting the counter's programmed mode in any manner. Counting will be affected as described in the mode definition. The new count must followthe programmed count format.

If a counter is programmed to read/write two-byte counts, the following precaution applies: A program must not transfer control between writing the first and second byte to another routine which also writes into the same counter. Otherwise, the counter will be loaded with an incorrect count.

0.4 READ OPERATION

There are three possible methods for reading the counters:

1) A simple read operation
2) Counter Latch command
3) Read-Back Command.

The first method performs a simple read operation. To read the counter with this method, the clock input of the selected counter must be inhibited by using either the GATE input or external logic. Otherwise, the count may be in the process of changing when it is read, thus giving an undefined result.

5.4.1 COUNTER LATCH COMMAND

The Second method uses the "Counter Latch Command". Like a control word, this command is written to the control word register and the SCO, SCI bits select one of the three counters. However two other bits, D5 and D4, distinguish this command from a control word. The following is the Counter Latch Command format:

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	0	0	X	X	X	X

Where:

SC1,SC0 specify counter to be latched ·

D5,D4 00 designate Counter Latch Command

X Ignore

The selected counter's output latch (OL) latches the count at the time the counter latch command is received. This count is held in the latch until it is read by the computer. The count is then unlatched automatically and the OL returns to "following" the counting element. This allows reading the contents of the counters "on the fly" without affecting the counting in progress. Multiple Counter Latch Commands can be used to latch more than one counter. Each latched counters OL holds its count until it is read. Counter Latch Commands do not affect the programmed mode of the counter in anyway.

If a counter is latched and then, sometime later, latch again before the count is read, the second Counter Latch Command is ignored. The count read will be the count at the time the first Counter Latch Command was issued.

5.4.2 READ-BACK COMMAND

The third method uses the Read-Back Command. This command allows the user to check the count value, programmed mode, and current state of the output pin and null count flag of the selected counter(s).

The command is written into the control word register and has the format shown below

D7	D6	D5	D4	D3	D2	D1	D0
1	1	CNT	STA	CN2	CN1	CN0	0

Where:

D7 & D6 These bits are 1.

CNT — Counter Latch. When this bit is 0, the counter(s) selected by bits CN2 - CNO are latched.

STA — Status. If this bit is 0, the status of the counter(s) selected by bit CN2 - CNO is returned. The status byte returned is of the format:

D7	D6	D5	D4	D3	D2	D1	D0
OUT	NC	RW1	RW0	M2	M1	M0	BCD

Where:

OUT — Current status of the output pin.

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 loading depends on the configuration selected. Until the count is loaded into the counter itself, it cannot be read from -the counter.
NC = 0 Null Count.
NC = 1 Count available for reading.

D5 to DO — are the same as the control word.

CN2-CNO — Select Counter. When these bits=1, they select the counter(s) to be used in the Read-Back Operation, where

CN2 Counter 2
CN 1 Counter 1
CNO Counter 0

d

5.5 GENERATING SQUARE WAVES OF PROGRAMMED FREQUENCY

User can generate square waves of programmable frequency by using Counter 0 and a combination of Counter 1 and Counter 2. Counter 0 can be connected to an internal 2MHz oscillator signal source or external clock source input from CN 1 pin 18. To generate square wave output from Counter 0, set the operation mode of Counter 0 to Mode 3. The output frequency can be programmed and have a maximum frequency of 1MHz(divisor = 2) and a minimum frequency of 30.52 Hz (divisor = 65535).

The driver of AR-B3204 always sets Counter 1 and 2 to the rate generator mode on running Function 0, but you can use the 1/O port statement of any programming language to alter their operation mode to square wave generator mode. Counter 1 and Counter 2 provide flexible frequency ranges, a minimum division ratio of 4(2x2) with a 2 MHz clock providing a 500KHz output at one extreme, and a division 65535x65535 provides an output of about 1.68 pulse/ hour at the other extreme.

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.

5.6 MEASURING FREQUENCY AND PERIOD

It is possible to use 8253 to measure frequency by raising the gate input of Counter 0 for some known interval of time and counting the number of pulses clocked into the counter for that interval. The gate signal can be derived from Counter 1 and Counter 2 operating in square wave mode. One of the AR-B3204 digital inputs can be used to monitor the gate input by the computer in order to know the measurement interval.

User can also use Counter 0 to measure pulse width. The measured signal should be connected to the gate input of Counter 0 and applied the clock input of Counter 0 with the internal clock(2MHz). Counter 0 should be loaded with a full initial count(65535) in the beginning. After the measured signal goes high,, the counter decreases until the gate input goes low at the end of the pulse. The change in the count is the duration of the gate input signal. If Counter 0 is fed with a 2MHz clock, the maximum pulse duration that can be measured is 65536*(1/2000)=32.77 miniseconds. The gate input of Counter 0 should also be connected to one of the digital inputs to synchronize loading and reading operation.

5.7 TRIGGERING THE AID PERIODICALLY

The key uses of the 8253 programmable interval timer in AR-B3204 provide pulses for starting the A/D to performs periodical sample. You can set up any possible output frequency by using Function 17 to load Counter 1 and 2 with the required divisors. For example, let us set up a sampling rate of 24.6KHz with internal 2MHz clock input. The overall division ratio is

2,000,000/24600 = 81.3

The closest frequency obtained will be:

2,000,000/81 = 24.691 KHz

Then, use Function 17 to perform this sampling rate:

```
PARA%(0)   = 3
PARA%(1)   = 27
FUNC%        17
ARB3204(FUNC%,PARA%(0),ARYI %(0),ARY2%(0),ER%)
```

Note that a counter can not divide by one; the minimum divisor is two, and the maximum is 65535. Driver Function 0 sets Counter 1 and 2 to the rate generator mode with a division ratio of 2,000. Function 17 allows you to change the division ratio but does not change the counter operation mode.

5.8 GENERATING INTERRUPTS WITH THE TIMER/COUNTER

The AR-B3204 hardware does not allow direct generation of an interrupt from the timer/counter. However, it is easy to set up the A/D to be triggered by the timer/counter and to have the AID generate an interrupt at the end of its conversion cycle(a constant delay of 20 to 25 microseconds). This setup is performed through the AR-83204 Control Register. Indirectly, this accomplishes the desired result of generating a periodic interrupt from the timer, and you can then install any desired interrupt service routine to service the interrupt.

This page intentionally left blank

CHAPTER 6 AID CONVERSION

This chapter explain how to use the AR-B3204 AID conversion functions. It covers input range selection, trigger modes, and data transfer for different A/D operations.

6.1 ANALOG INPUT RANGE SELECTION

The analog input range is determined by the gain code written into the Programmable Gain register. The gain code will be latched until next gain code is written. Table 6-1 list the gain code and their respective input range.

Table 6-1 Gain Code and Their Input Range

Gain Code	Gain	Input Range
0	1	-5V to +5V
1	2	-2.5V to +2.5V
2	4	-1.25V to + 1.25V
3	8	-625mV to +625mV
4	16	-312.5mV to 312.5mV

6.2 AID TRIGGER MODE

The AR-B3204 AID conversion can be triggered in any one of the following three ways:

Software Trigger. The software trigger is performed by writing any data to I/O address BASE+7. Software trigger normally is not used for high speed AID conversion due to the limitation of program execution time.

Pacer Trigger. The AR-B3204 uses Intel 8253 programmable interval timer/counter to be a pacer to generate AID converter trigger pulses with precise period. The pacer trigger mode is ideal for interrupt data transferwhich normally uses in AID applications requiring a higher sampling rate.

External Trigger. The external trigger pulses of AR-B3204 are input from POTRG (pin 13). This type of trigger mode is usually used in AID applications that require conditional and non-periodical conversions. The AID conversions start at the rising edge of external trigger pulses.

6.3 AID DATA TRANSFER

The AR-83204 A/D data can be transferred in any one of the following two ways:

Program Control: The program control transfer is performed by polling the End Of. Conversion(EOC) flag of the Status. Register. After. the A/D conversion is triggered, the application checks the EOC flag continuously. If a true is detected, the application program reads the AID data register to obtain the converted data.

Interrupt Transfer. In the interrupt transfer, data is transferred from A/D data registers to pre-defined data memory by the interrupt service routine. At the end of each conversion, an interrupt signal is generated to activate an interrupt service routine to perform the transfer. The interrupt control bit and interrupt level selection bits should be specified to enable AR-B3204 to generate interrupt request to the computer. The 8259 interrupt controller would also be initiated to enable the interrupt input in your PC. At the end of the service routine, an operation of writing any data to I/O address BASE+3 (status register) must be performed to enable the next interrupt procedure.

CHAPTER 7 DIA CONVERSION

'The D/A channels consist of a double-buffered, 12-bit multiplying D/A converters. It may be used with the internal fixed +5V or +10V DC reference to have 0 to +5V or 0 to +10V output range.

The D/A data register is write only register and uses addresses of BASE+O and +1 for. The first byte is the least significant and contains the eight least significant bits of data. The second byte is the most significant and contains the most significant four bits of data. The least significant byte should be written first which is temporarily held by a register in the D/A converter and not released to the output. After the most significant byte is written, the low byte and high byte are added and passed to the D/A converter. This double buffering process insures that D/A output has a single step update.

7.1 DIA DATA TRANSFER

There are two ways to perform the AR-B3204 D/A data transfer: program control, or interrupt routine.

Program Control. To transfer D/A data by program control, simply write data to the D/A data register at any time required. This is the most convenient way for applications control the D/A DC output level.

Interrupt Transfer. In interrupt routine transfer, previously prepared D/A data is stored in a block of system memory. Application program enables interrupt, selects interrupt level and selects the trigger source by programming the Control Register(BASE+6). The interrupt request signal for D/A conversions is generated by the EOC signal of the A/D converter. The interrupt service routine transfers two bytes of the D/A data from memory to the D/A converter for each interrupt request. At the end of the service routine, an operation of writing any data to I/O address BASE+3 (status register) must be performed to enable the next interrupt procedure.

7.2 WAVEFORM GENERATION

D/A converter is often required to be a waveform generator. It is difficult, however, to control the timing by using a timing loop in the program to generate waveform precisely.

Function 18 of AR-B3204 driver enable the user to output arbitrary waveforms through D/A channel by transfer D/A data to the D/A converters by interrupts.

The maximum transfer rate by using interrupt depends on the execution time of the interrupt and CPU speed. There is a time delay between the time that interrupt is requested and the time that data is actually processed. This delay mainly depends on whether ornot a higher priority interrupt is being processed. This delay can be reduced by disabling other interrupts while D/A data is being served. When higher output frequency is being performed, this latency delay will cause a more noticeable time jitter in the steps.

CHAPTER 8 SIGNAL CONNECTION

At is important to understand the connections between the type of input signals, how to use them effectively, and how to avoid ground loop. Misuse of the inputs is the most common difficulty users experience in applying and obtaining the best performance from data acquisition systems. This chapter provides useful information concerning how to connect digital and analog signals with AR-B3204 correctly in different types of data acquisition applications.

8.1 ANALOG INPUT CONNECTION

The AR-B3204 supports 16 channels of single-ended analog input configuration. A single-ended signal has only one signal wire for each channel. This type of signal does not have a ground connection at the signal source. This type of signal is also called a "floating source". A standard connection of a floating signal is shown in figure 8-1.

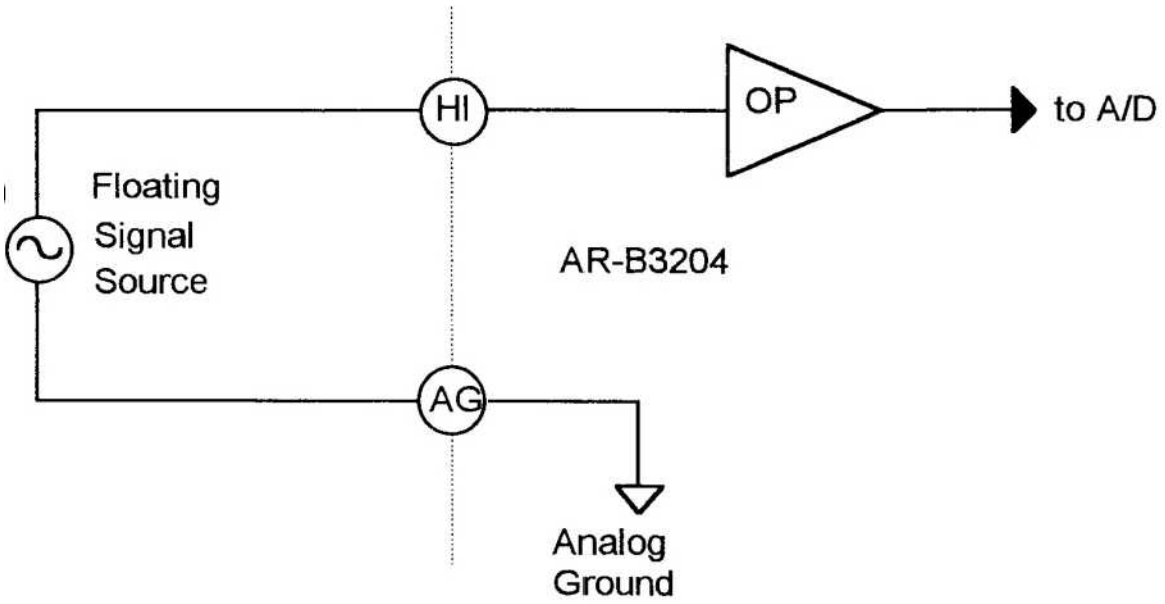


Figure 8-1 Connecting a floating signal to a single-ended input

8.2 ANALOG OUTPUT CONNECTION

The AR-B3204 provides one D/A output channel. The user may use intemaa precision +5 or +10 VDC reference to generate 0 to +5 VDC or 0 to +10 VDC output range. Note that the maximum available output source/sink current of D/A channel is 5 mA. Figure 8-2 shows how to connect D/A output.

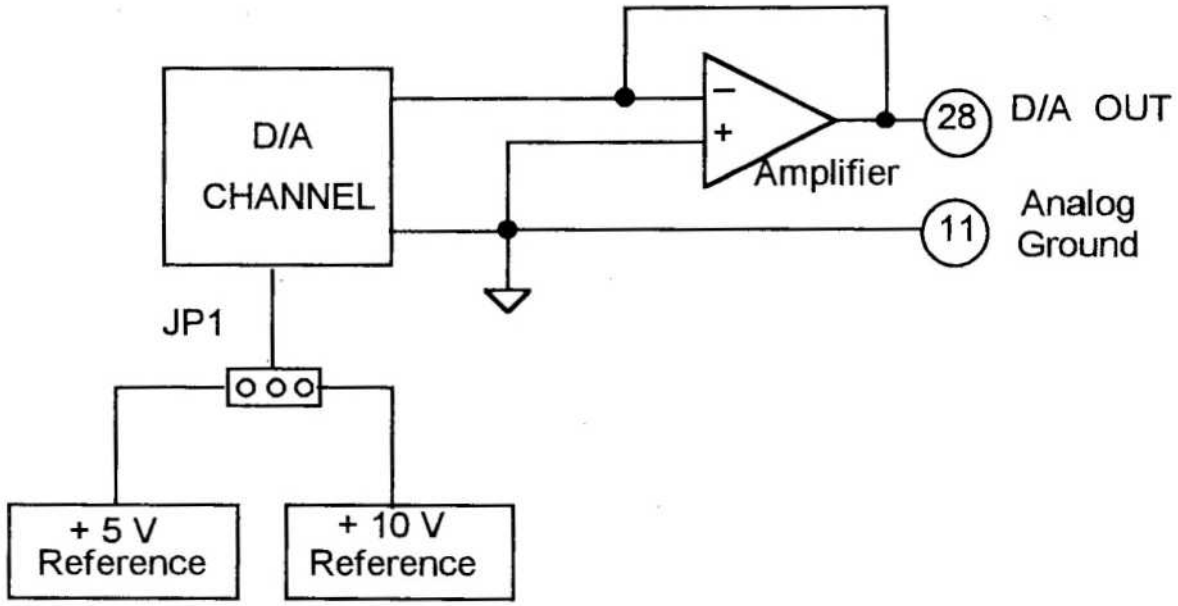


Figure 8-2 Connection of DIA channel

8.3 DIGITAL SIGNAL CONNECTION

AR-B3204 provides 7 TTL compatible digital inputs and 4 TTL compatible digital outputs. Figure 8-3 shows the connection of digital I/O.

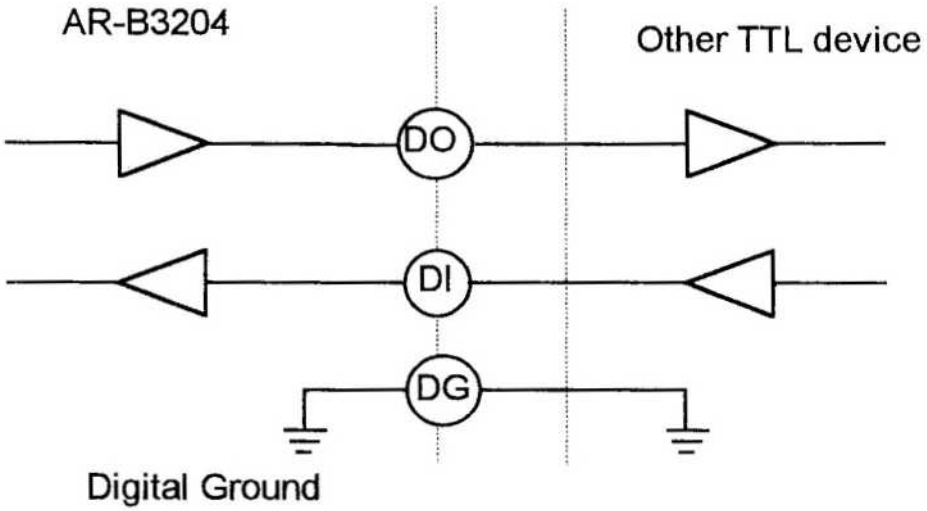


Figure 8-3 Digital !/O connection

The digital input channel 0 is internally pulled up with a 10K Ohm resistor. You may connect channel 0 to discrete signals from relay or switch directly. Using other DI channels to detect discrete signals needs a 10K pull up resister to be added externally.

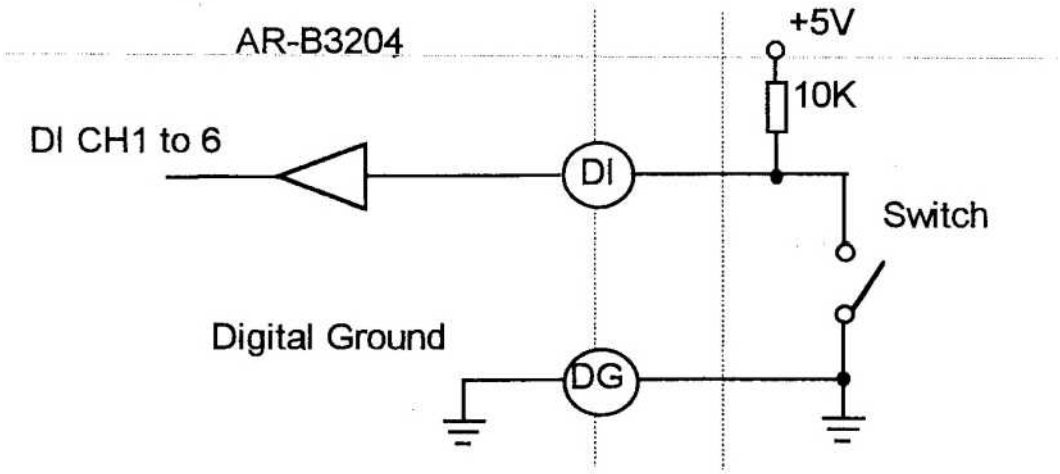


Figure 8-4 Connecting DI to switch/relay contacts

This page intentionally left blank

CHAPTER 9 CALIBRATION

To retain the best accuracy, periodically recalibrating your AR-B3204 is recommended. If you use AR-B3204 in a environment with a frequent large changes of temperature and vibration, we recommend you to calibrate every three months. If AR-B3204 is used in an office or laboratory, a six months to one year re-calibration is recommended.

A calibration program CAL3204.EXE is provided in the AR-B3204 software diskette to assist your calibration work_ This program leads you through the calibration and setup procedure with a variety of prompts and graphic displays showing you settings and adjustment tripots. In addition to the calibration program, a 4 1/2 digit digital multimeter and voltage calibrator or a stable DC voltage source are required. The explanatory material in this section is brief and is intended for use in conjunction with the calibration program.

9.1 AID CALIBRATION

Since the calibration of one A/D input range does not insure accuracy of any other range, it is suggested that you to calibrate the A/D whenever you change the input range. The following is a brief description of the adjustment procedure.

1. Connect the analog ground to analog input channel 0. Select analog input channel 0 by writing 0 to the multiplexer channel 0. Write 4 to gain control register so as to set gain of 16. Adjust the Null potential meter (VR4) until the voltage between TP 1 and TP2 is 0 +/- 0.05mV.

2. Connect a voltage with a value of 0.5 LSB above -FS (-Full Scale) to the analog input channel 0. Adjust the A/D offset potential meter(VR2) until the reading of the A/D flickers between -2048 and -2047.

3. Connect the analog ground to the analog input channel 0. Adjust the AID full-scale potential meter(VR3) until the reading of A/D is 0. Other inputs such as 114 and 3/4 full scale can be applied to check the linearity.

9.2 DIA CALIBRATION

Calibrate the D/A according to the following steps.

1. Connect the reference input of D/A channel to on board -5V or-10V reference by setting J P 1.

2. Output 4095 to D/A data registers and use digital multimeter to measure the analog output. Then, adjust D/A full scale pot (VR1) until the output voltage within the range of 4.9988+/-0.0005V (0 to 5V) or 9.9976+1-0.0005V

APPENDIX A SPECIFICATIONS

Analog Input

Channels:	16 single-ended inputs
Resolution:	12 bits
Conversion Type:	Successive approximation
Input Ranges:	+I-5V,+I-2.5V,+/-1.25V,+/-0.625V +/-0.3125, software programmable
Over voltage:	70V(peak to peak)
Accuracy:	0.01% of reading +/-1 bit
Linearity	+/-1 bit
Coding:	Offset binary
Conversion Time:	20 - 25 microsecond
Trigger Source:	External trigger, programmable pacer, or software trigger. Software selectable
Data Transfer:	Program control, or interrupt

Analog Output

Channel:	1 channel
Resolution:	12 bits
Type:	Multiplying, double buffered
Linearity:	+/- 1/2 bit
Output Range:	0 to +5V or 0 to +10V, jumper selectable
Output Drive:	+/-5mA (max.)
Settling Time:	1 microsecond(max.)
Data Transfer:	Program control, or interrupt
Slew Rate:	11 volt/microsecond (min.)

Digital Output

Channels:	4 bits
High level:	3.4V @ maximum sink current = 0.4 mA
Low level:	0.2V @ maximum source current = 8 mA

Digital Input

Channels:	7 bits
High level V:	2.OV(min.)
Low level V:	0.8V(min.)
High level I:	20 uA(max.) @ Vi=2.7V
Low level I:	-0.4mA(max.) @ Vi=0.4V

Programmable Timer/Counter

Device :	D8253C-2
Timer/Counter:	3 channels 16-bit down counter. Two channels permanently configured as programmable pacer. 1 counter free for user applications.

Input Gate:	TTUCMOS/DTL compatible
Time Base:	Pacer: 2 MHz External clock (4 MHz max.). Counter 0: Internal 2 MHz or external clock (4 MHz max.)
Pacer Output:	0.000466 Hz to 1 MHz

Interrupt Channel

Level:	IRQ2 to IRQ7, software selectable

General

Power Consumption: +5V: 200 mA typical, 0.25 A max.
+ 12V: 15 mA typical, 20 mA max.
-12V: 10 mA typical, 15 mA max.

Operating Temp.:	0 to 50 degrees C
Storage Temp.:	-20 to 70 degrees C
Humidity:	0% to 90% nôn-condensing
Dimension:	17.5 cm x 10.6 cm
Weight:	170 g

APPENDIX B DRIVER ROUTINE RETURN ERROR MESSAGE

Code = **0** **Good return, no error.**

Code 1 **Driver is not initialized.** It indicates that the driver must be initialized prior to this step.

Code = **2** **Invalid function number.** It indicates that *the* function_ number specified is not. allowed.

Code = 3 Invalid base address. This error is generated by Function 0 whenever the base address specified is not in the range of 100 HEX to 3F0 HEX.

Code = 4 Invalid Interrupt Level. This error is generated whenever the Interrupt Level specified is not in the range of 2 to 7.

Code = 5 Reserved.

Code = 6 Reserved.

Code = 7 Out of limits. This error is generated whenever the specified scan range is not between 0 and 15.

Code = 8 Invalid non-recycle/recycle mode. Valid number should be either 0or1.

Code = 9 No End of Conversion (EOC). This error is generated whenever a expected EOC signal from A/D is not received. This could be a hardware failure. Check that the base address set by SW1 matches the base address specified in Function 0.

Code = 10 Invalid Counter divisor. This error is generated whenever the counter divisor specified to one or both counters is 1 or 0. The valid divisor value is within the range of 2 to 65535.

Code = 11 Invalid number of conversions. The valid number of conversions must be in the range of 0 to 32767; where 0 is used to specify the maximum 32768 conversions.

Code = 12 Invalid counter operation mode. This error is generated whenever the counter operation mode specified to Function 10 is not within the allowed range of 0 to 5.

Code = 13 Invalid digital output data. This error is generated whenever the digital output data is not within the allowed range of 0 to 15.

Code = 14 Invalid D/A output data. This error is generated whenever the D/A data specified is out of the valid range 0 through 4095.

Code = 15 Reserved.

Code = 16 Invalid counter read mode. This error is generate whenever the counter read mode is neither 0(straight) nor 1(latched).

Code = 17 Invalid start sample number. This error is generated by Function 9 whenever the starting sample number is not within the valid range of 0 to 32767.

Code = 18 Invalid number of conversions. This error is generated whenever the number of conversions specified to Function 9 and 18 is out of the valid range 1 through 32767.

Code = 19 Invalid trigger source code. This error is generated whenever the AID or D/A trigger source is neither 0(pacer) nor 1 (external).

Code = 20 Interrupt operation is active. This error is generated whenever an interruptmode is attempting- to be -active-while--another is already active. Interrupt acquisition functions share the same resources and can not be used simultaneously.

Code = 21 Reserved.

Code = 22 Hardware failure. The board present test performed by Function 0 has failed. The base address specified to Function 0 must match the actual address selected on the board.

Code = 23 Invalid analog trigger channel number. This error is generated whenever the specified trigger channel in Function 19 is not within the valid range 0 to 15.

Code = 24 Invalid analog trigger level. This error is generated whenever the analog trigger value is out of the valid range -2048 to 2047.

Code = 25 Invalid analog trigger slope. The trigger slope specified in Function 19 is not 0(positive) or 1(negative).

Code = 26 Reserved.

Code = 27 Terminated by keyboard hit. This error is generated by Function 4 and 19 whenever these Functions are terminated by a keyboard hit.

Code = 28 Invalid AID channel number. This error is generated by Function 21 and 23 whenever the specified AID channel is out of the valid range 0 to 16.

Code = 29 Invalid AID gain code. This error is generated by function 21 whenever the specified gain code is not in the range of 0 to 4.