



10623 Roselle Street, San Diego, CA 92121 • (858) 550-9559 • FAX (858) 550-7322  
contactus@acesio.com • www.acesio.com

**MODEL 104-DA12-8A**  
**Analog Output Board with Arbitrary**  
**Waveform Generation**  
**USER MANUAL**

FILE: M104-DA12-8A.B11b

## Notice

The information in this document is provided for reference only. ACCES does not assume any liability arising out of the application or use of the information or products described herein. This document may contain or reference information and products protected by copyrights or patents and does not convey any license under the patent rights of ACCES, nor the rights of others.

IBM PC, PC/XT, and PC/AT are registered trademarks of the International Business Machines Corporation.

Printed in USA. Copyright 2001, 2007 by ACCES I/O Products, Inc. 10623 Roselle Street, San Diego, CA 92121. All rights reserved.

## WARNING!!

**ALWAYS CONNECT AND DISCONNECT YOUR FIELD CABLING WITH THE COMPUTER POWER OFF. ALWAYS TURN COMPUTER POWER OFF BEFORE INSTALLING A BOARD. CONNECTING AND DISCONNECTING CABLES, OR INSTALLING BOARDS INTO A SYSTEM WITH THE COMPUTER OR FIELD POWER ON MAY CAUSE DAMAGE TO THE I/O BOARD AND WILL VOID ALL WARRANTIES, IMPLIED OR EXPRESSED.**

## **Warranty**

Prior to shipment, ACCES equipment is thoroughly inspected and tested to applicable specifications. However, should equipment failure occur, ACCES assures its customers that prompt service and support will be available. All equipment originally manufactured by ACCES which is found to be defective will be repaired or replaced subject to the following considerations.

## **Terms and Conditions**

If a unit is suspected of failure, contact ACCES' Customer Service department. Be prepared to give the unit model number, serial number, and a description of the failure symptom(s). We may suggest some simple tests to confirm the failure. We will assign a Return Material Authorization (RMA) number which must appear on the outer label of the return package. All units/components should be properly packed for handling and returned with freight prepaid to the ACCES designated Service Center, and will be returned to the customer's/user's site freight prepaid and invoiced.

## **Coverage**

First Three Years: Returned unit/part will be repaired and/or replaced at ACCES option with no charge for labor or parts not excluded by warranty. Warranty commences with equipment shipment.

Following Years: Throughout your equipment's lifetime, ACCES stands ready to provide on-site or in-plant service at reasonable rates similar to those of other manufacturers in the industry.

## **Equipment Not Manufactured by ACCES**

Equipment provided but not manufactured by ACCES is warranted and will be repaired according to the terms and conditions of the respective equipment manufacturer's warranty.

## **General**

Under this Warranty, liability of ACCES is limited to replacing, repairing or issuing credit (at ACCES discretion) for any products which are proved to be defective during the warranty period. In no case is ACCES liable for consequential or special damage arriving from use or misuse of our product. The customer is responsible for all charges caused by modifications or additions to ACCES equipment not approved in writing by ACCES or, if in ACCES opinion the equipment has been subjected to abnormal use. "Abnormal use" for purposes of this warranty is defined as any use to which the equipment is exposed other than that use specified or intended as evidenced by purchase or sales representation. Other than the above, no other warranty, expressed or implied, shall apply to any and all such equipment furnished or sold by ACCES.

# Table of Contents

<b>Chapter 1: Overview</b> .....	5
<b>Figure 1-1: Block Diagram</b> .....	6
<b>Chapter 2: Installation</b> .....	7
<b>Figure 2-1: PC/104 Key Information</b> .....	8
<b>Chapter 3: Option Selection</b> .....	9
<b>Table 3-1: Base Address Jumpers</b> .....	9
<b>Table 3-2: Standard Address Assignments</b> .....	10
<b>Figure 3-1: Option Selection</b> .....	11
<b>Chapter 4: Programming</b> .....	12
<b>Table 4-1: Register Map</b> .....	12
<b>Chapter 5: Software</b> .....	16
<b>Chapter 6: Connector Pinouts</b> .....	21
<b>Table 6-1: Connector P1, Analog Outputs, 40-Pin Header</b> .....	21
<b>Table 6-2: Connector P2, ARB Status and Control, 10-Pin Header</b> .....	22
<b>Table 6-3: Connector P4, External 12V Input, 8-Pin Header</b> .....	23
<b>Figure 6-1: Signal Connection</b> .....	23
<b>Appendix A: Technical Specifications</b> .....	24
<b>Appendix B: 82C54 Counter Timer Operation</b> .....	25

# Chapter 1: Overview

## Features

- Eight 12-Bit 10uS Digital to Analog Converters
- Broadly Configurable Arbitrary Waveform Generator
- Counter/Timer with a 32-bit Divisor for Conversion Triggers
- Counter/Timer with a 16-bit Divisor for Interrupts
- 4-20mA Current Sink per Channel

## Circuit Description

The board has eight 12-bit digital to analog converters. The circuit uses an Analog Devices AD5348 which is an octal DAC chip with a single 12-bit input port, a 3-bit address port, and a shared 'start conversion' signal. The outputs are buffered by eight op-amps. Conversion values for each DAC may come from either the card's static RAM or from the PC/104 bus. The primary I/O connector (P1) is a 40-pin header that has all analog output voltage and current loop pins as well as ground and fused +5 and +12V pins. A 10 pin header, P2 is used for control including a general purpose counter output (4-pins, inputs) and monitoring (4-pins, outputs) of the ARB function.

Two countdown timers (based on an i8254) are provided. The output of Counter/Timer 0 (16 bit, see appendix B) is available at connector P2 pin 4. The output of chained timers 1 & 2 (32 bit) provide the conversion trigger. Both the 16-bit timer and the 32-bit trigger timer have 10MHz clock inputs, either may generate an interrupt. Note that the ARB's ability to load 8 DACs (sequentially) and each DAC's 10uS conversion rate requires the divisor to be at least 40, resulting in a 50,000Hz maximum output frequency per channel. (Up to 8 D/ACs updated simultaneously at 50kHz.)

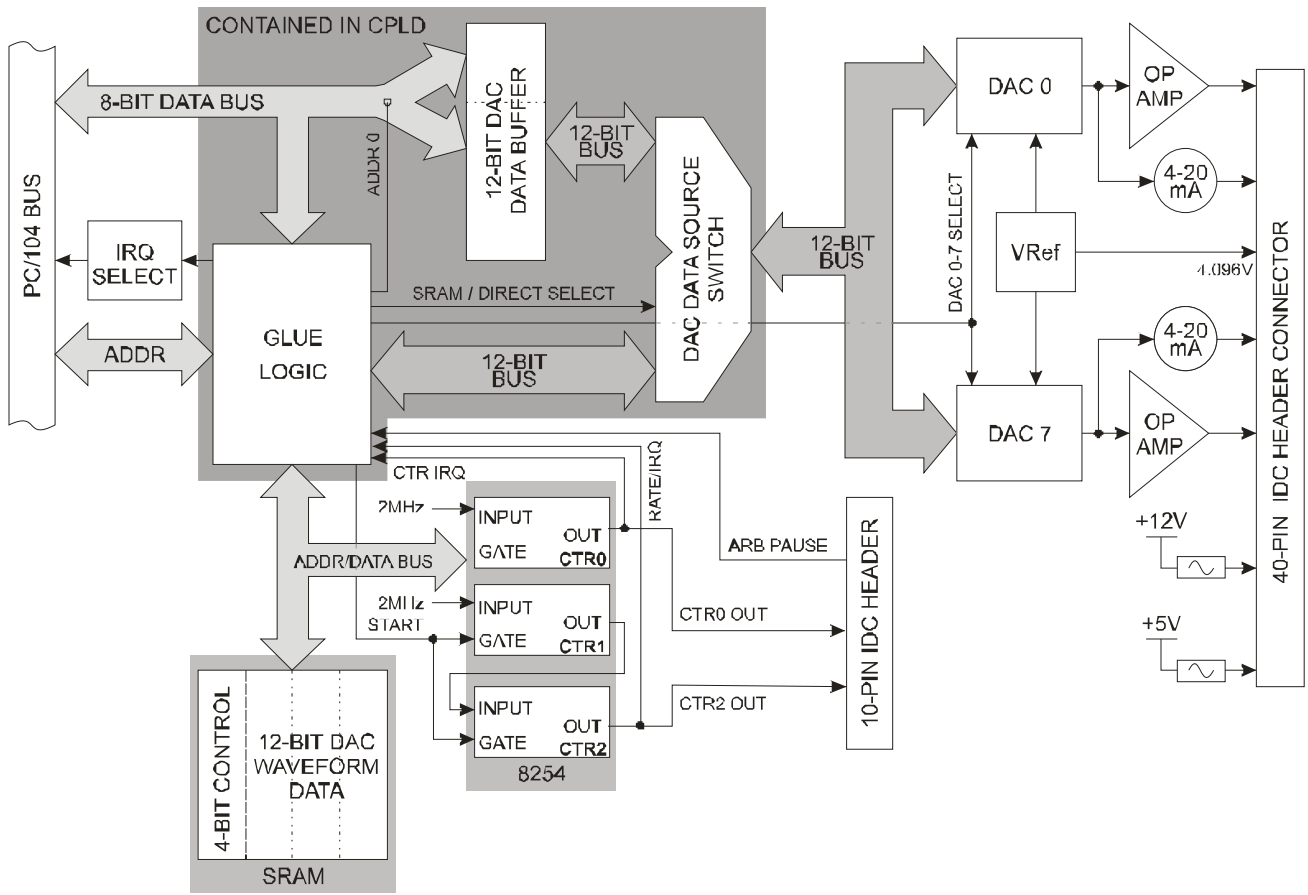
The board has eight 4-20mA current sink outputs. Each DAC may drive a 4-20mA sink with digital values from 0 (4mA) to 4095 (20mA). This is in addition to and in parallel with the op-amp driven voltage outputs. The current is regulated with FETs, each source lead connects to the card's ground plane through a small resistance and each drain lead connects to the user's circuit.

A static RAM plus CPLD form an arbitrary waveform generator. The user may load up to eight DAC waveforms 12-bits wide with varying depths, the static RAM will hold 64K conversion values. For example, the waveform for DAC 1 may be 32K words deep and the waveform for DAC 2 may be 16K words deep and the waveforms for both DAC 3 and 4 may be 8K words deep. Of course the static RAM can be divided equally between any number of DACs.

The user may cause all DACs to update (perform a conversion) from software or from the ARB. The ARB may be started, stopped, or paused with software or with TTL signals at connector pins. Start-conversion and ARB stop status signals are available at connector pins.

A 4.096 volt reference is available at a connector pin (J1 Pin 35). Fused (resettable) +5 volts from the PC/104 bus and fused (resettable) +12 volts are also available. The +12 volts may come from the user's power supply if the PC/104 bus doesn't have it (some don't).

Note that on power-up the 4.096V reference is in standby mode. This ensures that all D/AC outputs are at zero volts. The user must set bit 6 at the card's base address +10h. Clearing this bit at any time will cause all outputs to go to zero.



**Figure 1-1: Block Diagram**

# Chapter 2: Installation

A printed Quick-Start Guide (QSG) is packed with the board for your convenience. If you've already performed the steps from the QSG, you may find this chapter to be redundant and may skip forward to begin developing your application.

The software provided with this PC/104 Board is on CD and must be installed onto your hard disk prior to use. To do this, perform the following steps as appropriate for your operating system. Substitute the appropriate drive letter for your CD-ROM where you see d: in the examples below.

## CD Installation

The following instructions assume the CD-ROM drive is drive "D". Please substitute the appropriate drive letter for your system as necessary.

### DOS

1. Place the CD into your CD-ROM drive.
2. Type `D:\>Enter` to change the active drive to the CD-ROM drive.
3. Type `I N S T A L L >Enter` to run the install program.
4. Follow the on-screen prompts to install the software for this board.

### WINDOWS

1. Place the CD into your CD-ROM drive.
2. The system should automatically run the install program. If the install program does not run promptly, click START | RUN and type `D:\>I N S T A L L`, click OK or press `Enter`.
3. Follow the on-screen prompts to install the software for this board.

### LINUX

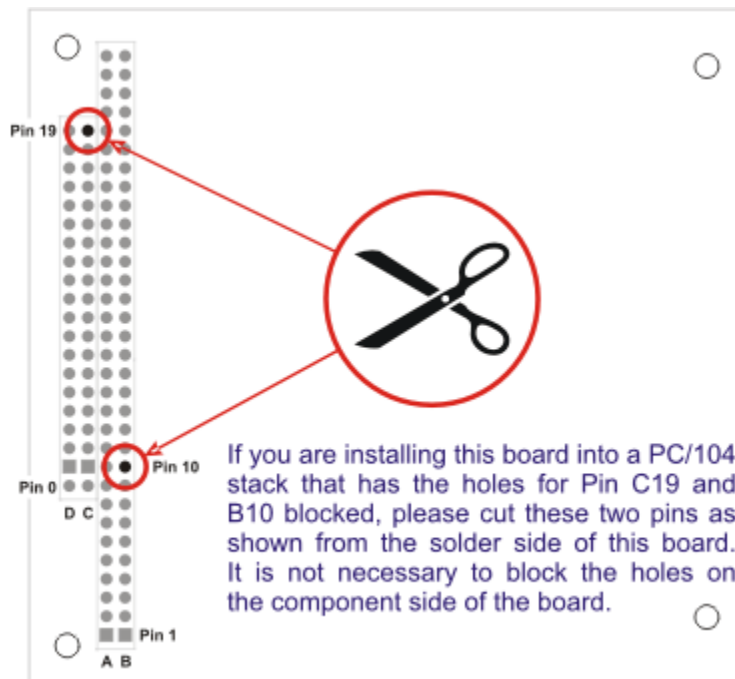
1. Please refer to linux.htm on the CD-ROM for information on installing serial ports under linux.

## Installing the Hardware

Before installing the board, carefully read Chapter 3 and Chapter 4 of this manual and configure the board according to your requirements. The SETUP Program can be used to assist in configuring jumpers on the board. Be especially careful with Address Selection. If the addresses of two installed functions overlap, you will experience unpredictable computer behavior. To help avoid this problem, refer to the FINDBASE.EXE program installed from the CD. The setup program does not set the options on the board, these must be set by jumpers.

### To Install the Board

1. Install jumpers for selected options and base address according to your application requirements, as mentioned above.
2. Remove power from the PC/104 stack.
3. Assemble standoff hardware for stacking and securing the boards.
4. Carefully plug the board onto the PC/104 connector on the CPU or onto the stack, ensuring proper alignment of the pins before completely seating the connectors together.
5. Install I/O cables onto the board's I/O connectors and proceed to secure the stack together or repeat steps 3-5 until all boards are installed using the selected mounting hardware.
6. Check that all connections in your PC/104 stack are correct and secure then power up the system.
7. Run one of the provided sample programs appropriate for your operating system that was installed from the CD to test and validate your installation.



**Figure 2-1: PC/104 Key Information**

# Chapter 3: Option Selection

Jumpers are available on the card to setup the following:

- Base address
- IRQ level
- DAC output voltage ranges

## Address Selection

The Card's Base Address is set by jumpers marked /A5 through /A9, and /A5 is the least significant bit of the address. The base addresses can be selected anywhere within the I/O address range 000-3E0 provided that they do not overlap with other functions. The FINDBASE software utility provided on the CD with your card will help you select a base address that does not conflict with other assignments.

This card requires a block of 32 addresses (20 hex). In order to configure the desired address, the hexadecimal address must be converted to a binary representation which is then selected by installing jumpers on the card.

For example, as illustrated below, switch selection corresponds to hex **2C0 (or binary 10 110x xxxx)**. The "xxxx" represents address lines A4 through A0 used on the card to select individual registers.

Hex Representation	2		C		
Conversion Factors	2	1	8	4	2
Binary Representation	1	0	1	1	0
Jumper Installed	NO	YES	NO	NO	YES
Jumper Label	A9	A8	A7	A6	A5

**Table 3-1: Base Address Jumpers**

Please note that "1" means that no jumper is installed and that "0" means that a jumper must be installed.

**Consult the documentation for your system before selecting a card address.**

The board occupies 32 bytes of I/O space. The board base address can be selected anywhere within the I/O address range 0-3E0 hex. If in doubt of where to assign the base address, refer to the following tables and the FINDBASE program to find an available address for your system.

HEX RANGE	USAGE
000-00F	8237 DMA Controller 1
020-021	8259 Interrupt
040-043	8253 Timer
060-06F	8042 Keyboard Controller
070-07F	CMOS RAM, NMI Mask Reg, RT Clock
080-09F	DMA Page Register
0A0-0BF	8259 Slave Interrupt Controller
0C0-0DF	8237 DMA Controller 2
0F0-0F1	Math Coprocessor
0F8-0FF	Math Coprocessor
170-177	Fixed Disk Controller 2
1F0-1F8	Fixed Disk Controller 1
200-207	Game Port
238-23B	Bus Mouse
23C-23F	Alt. Bus Mouse
278-27F	Parallel Printer
2B0-2BF	EGA
2C0-2CF	EGA
2D0-2DF	EGA
2E0-2E7	GPIB (AT)
2E8-2EF	Serial Port
2F8-2FF	Serial Port
300-30F	reserved
310-31F	reserved
320-32F	Hard Disk (XT)
370-377	Floppy Controller 2
378-37F	Parallel Printer
380-38F	SDLC
3A0-3AF	SDLC
3B0-3BB	MDA
3BC-3BF	Parallel Printer
3C0-3CF	VGA EGA
3D0-3DF	CGA
3E8-3EF	Serial Port
3F0-3F7	Floppy Controller 1
3F8-3FF	Serial Port

**Table 3-2:** Standard Address Assignments

## IRQ Selection

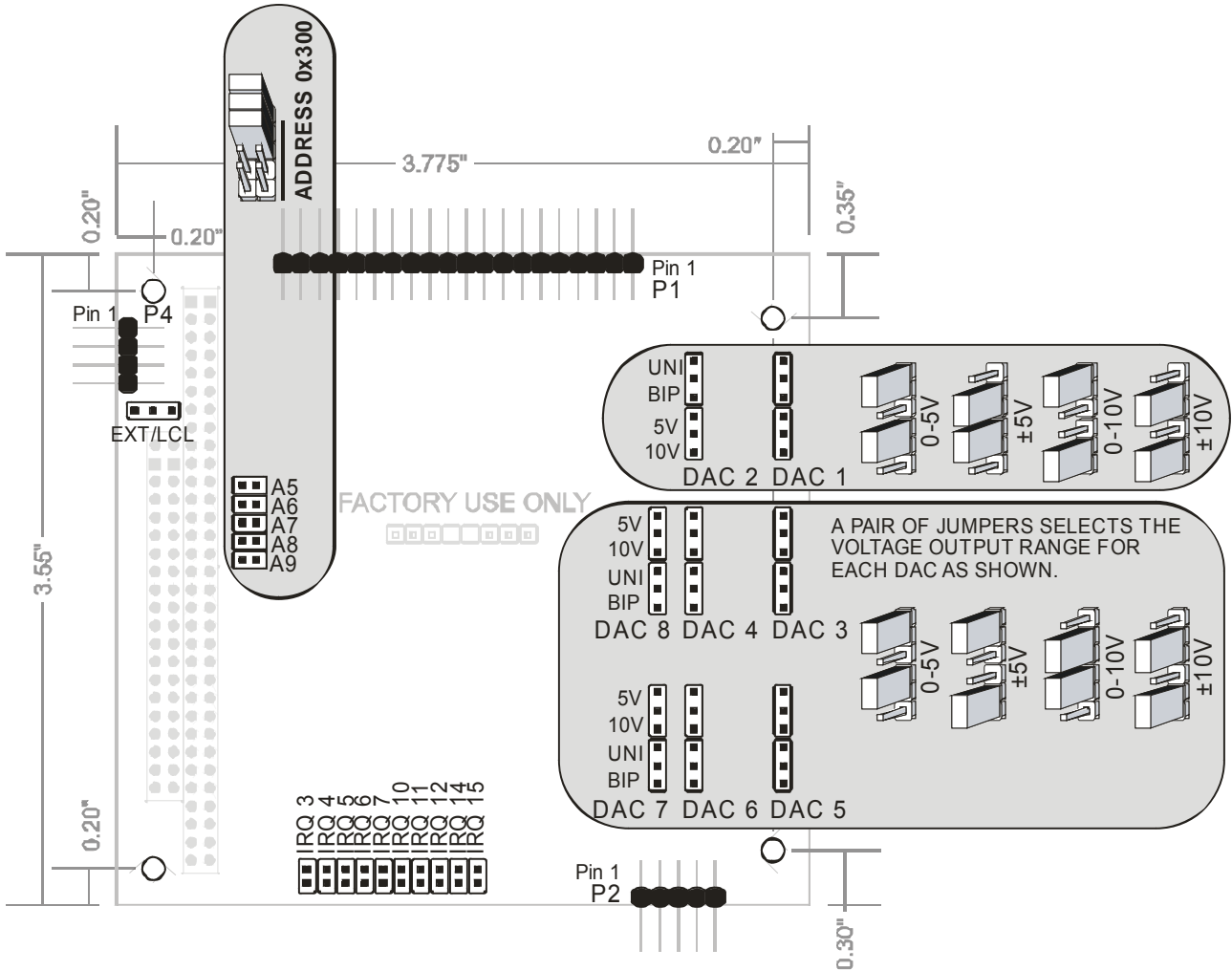
The board can generate an interrupt on the DAC conversion or on the falling edge of Counter/Timer 0. The IRQ # is selected by a jumper placed on one pair of pins from the group labeled 'IRQ3' through 'IRQ15'. If you do not intend to use the IRQ, do not install a jumper on any IRQ pins. The locations of these jumpers is shown in the Option Selection map, as well as in the Setup Program provided with the card.

**This interrupt can not be shared.**

# DAC Option Selections

Each DAC's op-amp circuit has two jumpers which configure its voltage range. The user may select unipolar or bipolar outputs and 5V or 10V ranges. In other words, the user may select four output ranges: 0 to 5V, 0 to 10V, -5V to +5V, -10V to +10V on a channel by channel basis.

**Please note the channel naming of DAC outputs associated with this card is DAC 1 through DAC 8 on the board silk-screen as well as in the following option selection map and in the card setup program. However, in the board's block diagram, register address map and connector pin assignments, they are referred to as DAC 0 through DAC 7.**



**Figure 3-1: Option Selection**

## Chapter 4: Programming

Offset	Write Function	Read Function
0	DAC 0 LSB	
1	DAC 0 MSB	
2	DAC 1 LSB	
3	DAC 1 MSB	
4	DAC 2 LSB	
5	DAC 2 MSB	
6	DAC 3 LSB	
7	DAC 3 MSB	
8	DAC 4 LSB	
9	DAC 4 MSB	
A	DAC 5 LSB	
B	DAC 5 MSB	
C	DAC 6 LSB	
D	DAC 6 MSB	
E	DAC 7 LSB	Disable DAC Outputs
F	DAC 7 MSB	Enable DAC Outputs
10	Output / ARB Control Register	ARB Control Readback
11	Synchronous / IRQ Enable Register	IRQ Enable Readback
12	Synchronous Software Trigger	
13	DAC RESET	
14	Program Counter 0	Read Counter 0
15	Program Counter 1	Read Counter 1
16	Program Counter 2	Read Counter 2
17	Counter Control Register	
18	Static Ram Address Lines 0-7	
19	Static Ram Address Lines 8-15	
1A	Static Ram Address Line 16	
1B	<i>Safe to write</i>	
1C	Static Ram Data LSB	
1D	Static Ram Data MSB	
1E	IRQ Clear	
1F	<i>Safe to write</i>	

**Table 4-1: Register Map**  
All offsets are in hexadecimal.

## Register Descriptions

The card is controlled by reading and writing 8- or 16-bit registers in I/O Memory. The registers are described below. Please note, all offsets are hexadecimal.

### Base+ 0 through Base + E, DACs 0-7 Direct Control Registers

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

These 8 word-sized registers allow you to update the DACs from software without involving the ARB functionality. If you're interested in simple DC voltage or current outputs you can ignore all the rest of the registers, and just use these to control the outputs of the DACs.

Write a word to an even address in the range of (base +) 0 through 0xE to update the DAC with a new count value. Data is offset Binary: 0x0000 is the minimum value (in unipolar modes, 0Volts, in bipolar modes, the most-negative value possible). 0x0800 is the "middle" of the range (0V when in Bipolar modes), and 0x0FFF is the maximum positive voltage.

Bits D15 through D12 are unused and should be set to zero for future compatibility.

To calculate the counts for any given Desired Output Voltage use the following equation:

Given: SpanVolts is equal to the highest positive voltage minus the most negative voltage the range supports (ex: +5V is "5" - "-5" which is "5+5" or "10")

Unipolar: Counts = (DesiredVolts / SpanVolts) \* 4096

Bipolar: Counts = (DesiredVolts / SpanVolts) \* 2048 + 2048

### Base + 10, ARB Control

D7	D6	D5	D4	D3	D2	D1	D0
ARB Busy	VREF				H/WSTART	PAUSE ARB	START

**Bit D0 START:** Set/clear to start/stop the Arbitrary Waveform Generator. This bit will be set if the ARB is triggered by a low-going pulse at pin 9 on connector P2.

**Bit D1 PAUSE ARB:** Set this bit to pause the generator, the DAC outputs controlled by the ARB will be frozen at DC values. When this bit is cleared the waveforms continue.

**Bit D2 H/W START ENABLE:** This bit must be set before a low-going pulse at pin 9 on connector P2 will start the ARB. Once the waveform generator is running, clearing this bit will have no effect. Use this bit to prevent H/W from starting the ARB unexpectedly. Hold pin 9 at connector P2 low to prevent S/W from starting the ARB, the waveforms start on a rising edge.

**Bit D6 VREF:** Set/clear to enable/disable the 4.096V DAC REFERENCE. While this bit is set DACs can generate signals, and P1 Pin 35 will have 4.096V output. At power-up this bit is cleared and all outputs are 0V.

**Bit D7 ARB Busy:** This status bit (active-HIGH) indicates when the ARB is writing to the D/ACs.

Bits D3, D4 and D5 are unused.

Reading this register will return the current state of these bits.

All bits in this register power-on or reset to "zero"

### Base + 11 Write, Interrupt Enable Register

D7	D6	D5	D4	D3	D2	D1	D0
H/W & S/W SIMUL						EICTR0	EITICK

Set/clear bits D0 and D1 to enable/disable an irq from each bit's IRQ source.

**Bit D0 EITICK:** Enables IRQs from the DAC Scan **Tick** (output of Counter 2). Each time the output of counter 2 goes low, the ARB will perform one DAC Scan, and if this bit is set, will generate an IRQ.

**Bit D1 EICTR0:** Enables IRQs from the output of **CounTeR 0**. Each time the output of Counter 0 goes low and this bit is set an IRQ will be generated. This counter, and its IRQ and output on the P2 connector (pin 4), are solely for use by the customer. They are unused by the ARB.

**Bit D7 H/W & S/W Simultaneous Update:** Setting this bit will inhibit the function of the hardware External DAC Conversion Trigger on pin 8 of P2. It will also prevent conversions triggered by writing new values to the D/ACs' buffers.

The D/ACs' conversion of their buffers can be triggered simultaneously by an external event in the following manner.

1. Set bit 7 at base address + 11h high.
2. Connect a tick generator (a low-going pulse or level for at least 1uS) to P2 pin 8.
3. Write values to the D/ACs' buffers.
4. Clear bit 7 at base+11h, the next tick will trigger conversions at all D/ACs.
5. Set bit 7, update the D/ACs' buffers, clear bit 7, wait for the tick, etc.

Alternately, loop the output of Counter 0 (P2 pin 4) onto the external trigger pin (P2 pin 8) and program the counter for the tick frequency. The card can generate an interrupt on the tick (see register base+11h) to signal that the D/ACs have converted their 12 bit buffers to analog. Use bit 7 to prevent conversions while the D/AC buffers are being loaded.

Reading this register will return the current state of these bits.  
All bits in this register power-on or reset to "zero."

### Base + 12, Synchronous Software Trigger

Write any value to this address to trigger all DACs' conversion of their buffer contents.

### Base + 13, DAC Reset

Write any value to this register to clear the DACs and set all outputs to their most negative value. For example, if the range is -10V to +10V on DAC 1 it will go to -10V. If the range on channel 2 is 0V to 5V it will go to zero volts.

### Base + 14, Program Counter/Timer 0 - (see Appendix B)

This is a 16-bit timer used as a timer-tick for external hardware. Its 10MHz input clock is divided by a load 16-bit value and output on P2 pin 4. This counter output can also generate an IRQ, if enabled (see base + 10.)

### Base + 15 and 16, Program Counter/Timers 1 & 2 - (see Appendix B)

These two counters are concatenated on the board. The input to Counter 1 is a 10MHz clock, with the output of Counter 1 tied to the input of Counter 2. The output of Counter 2 provides a Tick, on which the ARB will perform a "scan" if it is running. The total divisor of the two counters is equal to the product of the divisors loaded into each. For example, if you load Counter 1 with 100 and load Counter 2 with 50 it will provide a total divisor of 5000 (which divides the 10MHz input by 5000 down to a 400Hz tick rate) Place both counters in Mode 2 (divide-by-N pulse train), and make sure the total divisor in the two concatenated counters is not less than 40 i.e, you can load 4 into Counter 1, and 10 into Counter 2, but never 4 and 9.

### Base + 17, Counter/Timer Control Register - (see Appendix B)

Use this register to configure the 8254.

### Base + 18, Static Ram Address Pointer (16-bit or 32-bit)

This is a 32-bit pointer into the static RAM. The static RAM holds 128K bytes which equal 64K words of conversion data. The register at 'base + 18' holds static RAM address lines 0-7, 'base + 19' holds lines 8-15, and the LSbit of the register at 'base + 1A' holds address line 16. You can write a 32-bit value to Base + 18, or you can write a 1-bit value to Base + 1A followed by a 16-bit value to Base + 18, whichever is more convenient to your software.

### Base + 1A, Static Ram Address Pointer Most Significant Bit (Bank Select)

The 16-bit pointer at Base + 18 can only address 64K worth of data in the SRAM, so to reach the upper 64K of the total 128K SRAM on the card you must set Bit D0 at Base + 1A. This can be done with a single byte-write to Base + 1A, or, if its more convenient to your software, you can write a single 32-bit long SRAM address to Base + 18, and the card will parse it all out into the needed 17-bit address.

### Base + 1C, Static Ram Data Write

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
END	F	EODS	LOOP	12-Bit DAC data as described for Base + 0 through + E											

After setting up the SRAM address pointer as described above in Base + 18's description, write a 16-bit data value to this register at Base + 1C. The data written here will be copied into the SRAM at the location pointed to by Base + 18. The top four bits of the 16-bit value written will be interpreted as instructions to the ARB. The least significant 12-bits of the data are identical to the 12-bits written to Base + 0 through + E. Read Chapter 5, Software, for a detailed description on using the ARB and SRAM.

**Bit D15, END:** When the ARB encounters an END instruction in the SRAM it will complete the current output scan then stop.

**Bit D14, F:** The F bit is used as a Flag bit. Your software can use this bit to indicate anything you would like. If so enabled the ARB will generate an IRQ each time this bit is set. This could be used to synchronize your software to the playback of a waveform, for example.

**Bit D13, EODS:** The EODS instruction tells the ARB that the next 16-bit value in the SRAM is intended for DAC0. EODS stands for End-Of-DAC-Scan. By setting this bit you can change the number of channels the DAC output Scan consists of on a per-scan basis. If every word in the SRAM has EODS set, you will only be outputting on DAC0 (a 1-DAC Scan). If every other data point in the SRAM had EODS set, you would be outputting on DAC0 and DAC1(a 2-DAC Scan). To output on all 8 DACs (and 8-DAC Scan), set EODS on every eighth data point. Note that any combination is possible; you can mix any size DAC Scans you'd like freely, and in doing so achieve a variety of DAC pattern output rates, and maximize SRAM size by reducing redundant data loaded.

**Bit D12, LOOP:** The LOOP instruction tells the ARB to reset the internal SRAM pointer, such that the next data point read from the SRAM will come from address 0x00000000, the beginning of the SRAM. If the SRAM contains a LOOP instruction, the waveform will be output repeatedly until stopped.

Again, please consult the Software chapter, for a detailed description on using the ARB and the data in the SRAM.

### Base + 1E, IRQ Clear

Write any value to this address to clear a pending interrupt. Once the card generates an IRQ, no further IRQs will be requested until the pending IRQ has been cleared by writing to this address.

# Chapter 5: Software

This board can be used in two basic modes of operation, direct DC DAC outputs, or as an arbitrary waveform generator (ARB). You can think of the board as a simple 8-channel DAC to output voltage levels. You can also utilize the board's SRAM and ARB to generate complex, precisely timed and repeatable waveforms, all without consuming valuable CPU overhead. A plethora of control and status signals, bits, and interrupts are provided to ensure a flexible solution for any situation.

This chapter will discuss both conceptual modes of operation, starting with using the board as a simple DC DAC; a full discussion of the ARB mode will follow, and the chapter will finish with a note about mixing the two modes.

## DC DAC Modes:

The DACs can be used in either: Immediate Update, or Simultaneous Update Mode.

In Immediate Update Mode any DAC written to will immediately update its output voltage to reflect the newly written value. All other DACs will retain their previously set output voltages.

In Simultaneous Update Mode no DAC will update its output voltage, regardless of it being written to, until a special "Update" command is issued to the board. When this special Update command is issued all 8 DACs will update simultaneously. This simultaneous mode eliminates time skew otherwise experienced between channels.

The board powers up in Immediate Update Mode.

To enter Simultaneous Mode, set bit D7 at base+11, the Interrupt Enable Register.

To leave Simultaneous Mode and re-enter Immediate Update Mode, clear bit D7 in this register.

To issue an "Update" command, write any value to the register at Base+12, "DAC Conversion Trigger.

## Hardware Driven Waveform (ARB) Mode:

The board is far more powerful than a simple DC level eight channel DAC board is. By using the onboard 128KByte SRAM buffer, our onboard ARB logic, and some very flexible status and control signals, very complex waveforms can be generated.

It is possible to generate a precisely timed pulse with no CPU involvement, and have it be re-generated as often as desired, still with no CPU involved, on any number of the DACs. Or you can create a seamlessly repeating waveform that will be continuously generated until explicitly stopped, such as a sine wave. The following sections provide an overview of several possible waveform generation scenarios of increasing complexity; the steps outlined in each section are described in detail after the last scenario.

Please note, the steps are numbered using the most complex example's step numbers; the skipped step numbers in simple scenarios are filled in as you reach more complex examples.

### Generating A Pulse Waveform on One DAC:

To generate a pulse on a single DAC you must perform a sequence of steps.

- 1) Create a waveform.
- 2a) Add EODS commands to each data point in the waveform.
- 5a) Add the ARB END command to the last value in the waveform buffer.
- 6) Write the waveform buffer to the onboard SRAM.
- 7) Configure the pacer TICK (CTR1+CTR2) (or enable external TICK).
- 8) Start the ARB (or configure the ARB for hardware START).

Once a START has occurred, whether from software or the pin at the connector, the ARB will begin executing one DAC Scan per TICK. Each time a TICK occurs (whether from the output of counter 2, or the external signal at P2 pin 1) the ARB will again read data from the SRAM and write it to sequential DACs until EODS stops the process. Because we set EODS on every data point, each DAC Scan consists of only one data point, so we're only writing to a single DAC (the first.) This process of issuing DAC Scans per TICK will continue until the END command is detected in the SRAM data. That DAC Scan will be allowed to complete, and the card will STOP (and while stopped will ignore TICKs.)

#### **Generating A Repeating Waveform on One DAC:**

- 1) Create a waveform.
- 2a) Add EODS commands to each data point in the waveform.
- 5b) Add the ARB LOOP command to the last value in the waveform buffer.
- 6) Write the waveform buffer to the onboard SRAM.
- 7) Configure the pacer TICK (CTR1+CTR2) (or enable external TICK).
- 8) Start the ARB (or configure the ARB for hardware START).

You'll note that only step 5 changes. By using the LOOP command instead of the END command we're instructing the ARB to start over in the SRAM instead of stopping. When enough TICKs have occurred that the entry in SRAM containing the LOOP bit is read, the ARB will reset its internal SRAM pointers so the next DAC Scan data will be read starting from address 0x00000, the beginning of the SRAM. This results in a waveform that will repeat itself until stopped.

#### **Generating a Repeating Waveform on Multiple DACs:**

- 1) Create a waveform for each DAC.
- 2b) Add EODS commands to every point in the waveform for the last DAC you're using. Please note, although this step looks different from 2a), it is essentially the same.
- 4) Shuffle the waveforms for each DAC into a single buffer containing all waveforms.
- 5b) Add the ARB LOOP command to the last value in the waveform buffer.
- 6) Write the waveform buffer to the onboard SRAM.
- 7) Configure the pacer TICK (CTR1+CTR2) (or enable external TICK).
- 8) Start the ARB (or configure the ARB for hardware START).

As noted in Step 2b, nothing really changes in this step. The phrasing was changed in 2a to simplify the example, nothing more.

Also, to generate data for more than one DAC, Step 4 is added. The SRAM on the board requires the data to be in "DAC Scan Order", where all the data for a single DAC Scan is sequential in the SRAM. Because of this, it is useful to shuffle the data before (or as you) write the data to the card. See the detailed step descriptions below for more information on the "shuffle" required. In single-DAC samples Step 4 is implied, as the DAC Scan buffer only contains one waveform..

#### **Generating a Repeating Waveform on Multiple DACs (While Optimizing SRAM size):**

Because the ARB uses the EODS bit to indicate "The DAC Scan is done," it is possible to have DAC Scans in the onboard SRAM with varying numbers of DACs per scan. This allows you to use the SRAM to its fullest and avoid redundant data. Imagine a case where you'd like to have a high fidelity 44KHz modulated sine wave playing on DAC 0, but you also need a simple 16-level step function output synchronously on DAC 1. For the entire 64K Sample SRAM buffer you only have 16 values that DAC 1 needs to output. Instead of wasting half the SRAM for DAC 1 merely because you have two DAC channels in use, our board only requires 16 entries be used.

The steps are very similar:

- 1) Create a waveform.
- 2b) Add EODS commands to every point in the waveform for the last DAC you're using.
- 3) Calculate how often you need to shuffle each waveform into the Scan buffer.
- 4) Shuffle the waveforms for each DAC into a single buffer containing all waveforms.
- 5b) Add the ARB LOOP command to the last value in the waveform buffer.
- 6) Write the waveform buffer to the onboard SRAM.
- 7) Configure the pacer TICK (CTR1+CTR2) (or enable external TICK).
- 8) Start the ARB (or configure the ARB for hardware START).

You'll note we added Step 3. In our prior scenarios we were shuffling the DACs evenly, using 1/2 of the SRAM for each DAC in a 2-channel DAC Scan, 1/4 of the SRAM for each DAC in a 4 channel DAC Scan, etc. In order to only consume 16 entries (32 bytes) you must only shuffle the data into the DAC Scan buffer a total of 16 times.

The DAC Scan buffer (also called the pattern buffer or just the SRAM) must start with the initial value for DAC0 and DAC 1 (with an EODS command attached to DAC 1's data). Following the initial values will be one set of sine wave data points from 0 to 2PI radians, with every data point containing an EODS. This will cause one sine wave to be output on DAC 0, without disturbing the initial data in DAC 1.

Repeat this pattern of writing one sine worth of data for DAC 0 (with EODS set on all points but the first) and adding the DAC 1 value on the last data point (setting EODS on DAC 1's data instead of DAC 0's data) until you've written all of the data points. (Remember to add the LOOP instruction to the last value in the buffer.)

### **The Steps, Detailed:**

This section will describe how to perform each of the steps outlined above in detail. To explain the steps we're going to formulate an example application: A five DAC waveform, using ILDA (laser) signals X, Y, Red, Green and Blue, to be output at 40,000 points per second. The waveform, if played through an ILDA laser projector, would trace a color picture, and we want it to loop, so the picture stays onscreen until stopped.

#### **1) Create a waveform.**

The ARB's onboard SRAM can store 65536 points of waveform data total. This must be divided up among the number of DACs you're using. This board has a very flexible system using the EODS bit, allowing you to use non-identical length buffers for each DAC.

We need to create a waveform for each DAC: In our example each X,Y position in the waveform also has an RGB color set associated with it, for a total of five DACs being output per DAC Scan. Our exact X,Y data and RGB data isn't important to the example, imagine perhaps a circle in rainbow hues. DAC0's waveform contains the X data, DAC1's will have the Y data, with DACs' 2-4 containing the R, G, and B data respectively.

To build a waveform for a circle we will use  $x = \cos(\theta)$  and  $y = \sin(\theta)$ . Theta is the angle in radians for our circle. We want to have as smooth a circle as possible, but we also want to update the entire circle image at around 40 times per second so it looks like a solid circle instead of a moving dot. With our selected 40,000Hz output rate, we can use 1000 points for our circle and achieve both goals. Theta should vary from 0 to  $2\pi$  to complete a circle, and with 1000 steps per circle our X equation becomes  $X = \cos(\theta * 2\pi / 1000)$ , and  $Y = \sin(\theta * 2\pi / 1000)$ , where theta varies from 0 to 999.

This equation will yield 1000 values for our waveform, but they will be vary between -1 and +1. We need to scale the data into 12-bit offset binary (so it varies between 0 and 4095). To do so we can add +1 and multiply by 2047.5, resulting in  $X = (\cos(\theta * 2\pi / 1000) + 1) * 2047.5$ , and a similar equation for Y. If we run each of these equations in a loop with theta varying from 0 to 999, and put the 1000 results into a buffer, we have our X and Y waveforms.

Perform a similar operation to generate waveforms for the R, G, and B DAC channels.

**2b) Add EODS commands to every point in the waveform for the last DAC you're using.**  
We're conceptually using all five DACs every DAC Scan, so the last DAC is always DAC 4, the "Blue" DAC. We can therefore add the EODS command to every Blue data point. To do so, simply add 0x2000 to every point in the Blue buffer.

**3) Calculate how often you need to shuffle each waveform into the Scan buffer.**  
To create a DAC Scan buffer, the data needs to be put into XYRGB order. Right now we have five individual DAC buffers, each containing an entire waveform for the respective DAC. Our example always uses all five DACs on each data point, so our data needs to be interleaved together on a one-to-one basis: the answer to "how often we need to shuffle" is "1" or "every time"

**4) Shuffle the waveforms for each DAC into a single buffer containing all waveforms.**  
This step may be easiest to understand with a code snippet; call the DAC Scan buffer ScanBuf, and each Waveform Buffer XBuf, YBuf, RBuf, GBuf, and BBuf as appropriate.

```
for (i = 0; i < 1000; i++)
{
ScanBuf[i * 5 ] = XBuf[i];
ScanBuf[i * 5 + 1] = YBuf[i];
ScanBuf[i * 5 + 2] = RBuf[i];
ScanBuf[i * 5 + 3] = GBuf[i];
ScanBuf[i * 5 + 4] = BBuf[i];
}
```

If, on the other hand, our example had only 2 channels, DAC 0 playing a 44KHz modulated sine, DAC 1 a simple rising staircase with the DAC 1 data occurring only 16 times in the entirety of the DAC0 buffer:

```
index = 0;
for (i = 0; i < 16; i++)
{
for (j=0; j<2048; i++)
{
ScanBuf[index++] = DAC0Buf[i * j];
}
ScanBuf[index++] = DAC1Buf[j]
}
}
```

**5a) Add the ARB END command to the last value in the waveform buffer.**  
If we wanted to play the circle once we would add an END command to the last data value in the DAC Scan Buffer by adding 0x8000.

**5b) Add the ARB LOOP command to the last value in the waveform buffer.**  
Because we want the circle to play repeatedly, we will add a LOOP command by adding 0x1000 to the last data point in the DAC Scan buffer: `ScanBuf[i*5+4]=ScanBuf[i*5+4]+0x1000;`  
or  
`ScanBuf[index]+=0x1000; //if we used the index method`

**6) Write the waveform buffer to the onboard SRAM.**  
Now that we have a DAC Scan Buffer full of data points to be output we have to write it to the onboard SRAM. The SRAM is accessed via a 32-bit SRAM address pointer register, and a 16-bit data register. Only 17 bits of the SRAM address pointer register are significant, so it can conveniently be treated as a 16-bit pointer and a 1-bit Bank Select if desired. Any simple loop can be used to output the data, but its important to remember you're writing into only the "even" locations in the SRAM (the odd locations are filled with half the 16-bit data you write automatically); that is to say, the pointer is a byte-pointer, but we are treating the buffer as holding words.

```

for (index = 0L; index < 65536; index++)
{
    OutPortL(Base + 0x18, index * 2); //configure address
    OutPort(Base + 0x1C, ScanBuff[index]); //write data
}

```

Please note the above snippet assumes you're writing the full 128Kbyte buffer - in our example circle drawing application, index should remain < 5000, the number of points in our DAC Scan Buffer.

**7) Configure the pacer TICK (CTR1+CTR2) (or enable external TICK).**

The pacer clock needs to be configured to TICK at the correct rate. In our example of 40,000 Hz, this will require dividing down the 10MHz input clock by a factor of 50. Its convenient to use both counters 1 and 2 in mode 2 (divide-by-N), and to get a total division of 50 out of the two counters, we have to put 5 in one counter and 10 in the other. Appendix B shows the technical details of how to program the onboard 8254 counter timers, but our sample programs wrap up all the details into several simple function calls which we recommend you utilize. CtrMode and CtrLoad are used to put the counters into mode 2 and load the divisors into the counters, respectively.

**8) Start the ARB (or configure the ARB for hardware START).**

Now that the counters are moded-and-loaded, and the full waveform to be output has been written to the SRAM, the only remaining step is to start the ARB running. If we want to start the ARB on software command, set START at Base + 10 by writing a 41h to this register. If we wanted to have the ARB start performing DAC Scans synchronous with an external signal on P2 Pin 9 being asserted, write 44h instead. Consult the description of Base +10 above for more details.

In either case, once started the ARB will wait for a TICK to occur. On each tick the ARB will perform a DAC Scan: it will read one 16-bit value from the SRAM and write it to the current DAC.

If the 16-bit value read did not have an EODS command bit set, the current DAC will increment, and the DAC Scan will continue. If EODS was set, the current DAC will be reset to DAC 0 and the DAC Scan ends. (If the current DAC increments past DAC7 it behaves as an automatic EODS).

If any data read had END set, the current DAC Scan will finish, then the ARB will stop (as if the START bit at Base + 10 was cleared.)

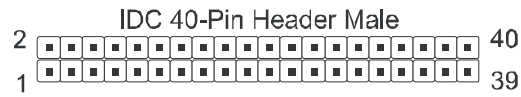
If any data read had LOOP set, the next data read will come from the beginning of the onboard SRAM.

## Mixing the Modes:

It is possible to write to the DACs while the ARB is running. To avoid data corruption, avoid writing to DACs that have data in the onboard SRAM DAC Scan buffer.

It is also possible to use the external trigger signal, simultaneous mode, and the external trigger interrupt to write waveforms to the DAC without using the onboard ARB. This eliminates the 64K-sample buffer limitation as your waveform is stored in the computer's RAM - however the CPU utilization is directly proportional to the output sample rate. Also, because the PC/104 bus is slow, it is not possible to achieve high-data rates on many DACs this way.

# Chapter 6: Connector Pinouts

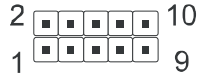


**Table 6-1: Connector P1, Analog Outputs, 40-Pin Header**

Pin	Function	Pin	Function
1	Ground	2	DAC 0 Voltage Output
3		4	DAC 0 4-20mA Output
5		6	DAC 1 Voltage Output
7		8	DAC 1 4-20mA Output
9		10	DAC 2 Voltage Output
11		12	DAC 2 4-20mA Output
13		14	DAC 3 Voltage Output
15		16	DAC 3 4-20mA Output
17		18	DAC 4 Voltage Output
19		20	DAC 4 4-20mA Output
21		22	DAC 5 Voltage Output
23		24	DAC 5 4-20mA Output
25		26	DAC 6 Voltage Output
27		28	DAC 6 4-20mA Output
29		30	DAC 7 Voltage Output
31		32	DAC 7 4-20mA Output
33	34	+12 Volts Fused	
35	4.096V	36	+5 Volts Fused
37	Ground	38	n.c.
39	n.c.	40	n.c.

Note that pin 35 will be 0V if bit 6 of the Control Register at Base+10 is LOW.  
 Note also that pins 38, 39, and 40 are unused, such that connection via a standard 37-Pin DSub connector is simplified.

IDC 10-Pin Header Male



**Table 6-2:** Connector P2, ARB Status and Control, 10-Pin Header

Pin	Function	Pin	Function
1	Load DAC (output)	2	ARB EODS (output)
3	ARB Running (output)	4	Counter 0 (output)
5	Ground	6	Ground
7	Ext ARB Stop (input)	8	Ext DAC Trigger (input)
9	Ext ARB Start (input)	10	Ext ARB Pause (input)

**Pin 1, Conversion Trigger, output**

Every time the DACs are updated a very short low-going pulse will be generated here. (<100nS).

**Pin 2, ARB EODS, output**

A low-going pulse here indicates that the ARB has encountered an EODS instruction in static RAM.

**Pin 3, ARB Running, output**

If the ARB is running this pin will be high.

**Pin 4, Clock Tick, output**

This is the output of counter/timer 0. The user may program this 16-bit count-down timer to generate a tick or a square wave etc. (see appendix B).

Pins 5 and 6 are connected to ground.

**Pin 7, External ARB Stop, input**

This TTL signal (active low) will stop the ARB. If the ARB is re-started it will begin at SRAM address zero.

**Pin 8, External DAC Conversion Trigger, input**

This active low TTL signal will cause all DACs to convert their buffer contents. This hardware signal may be inhibited (masked) by software by setting bit D7 SIMUL at base+11.

**Pin 9, External ARB Start, input**

This pin has a 10K pull-up resistor to +5V. Apply a low voltage to this pin to stop the waveform generator. A rising edge will start the generator if bit 2 in the ARB Control Register is set.

**Pin 10, ARB Pause, input**

A low TTL level here will either cause the ARB to pause or will inhibit it's start. Note that if this signal is asserted and then a start-pulse is applied to pin 9 then the ARB will start when the pause is released. Also, FYI, if the ARB is in the process of updating DACs it will load all of the DAC buffers before pausing. When the pause is released a conversion trigger will be sent to the DACs on the timer tick and the ARB will resume moving data from the SRAM to the DACs.

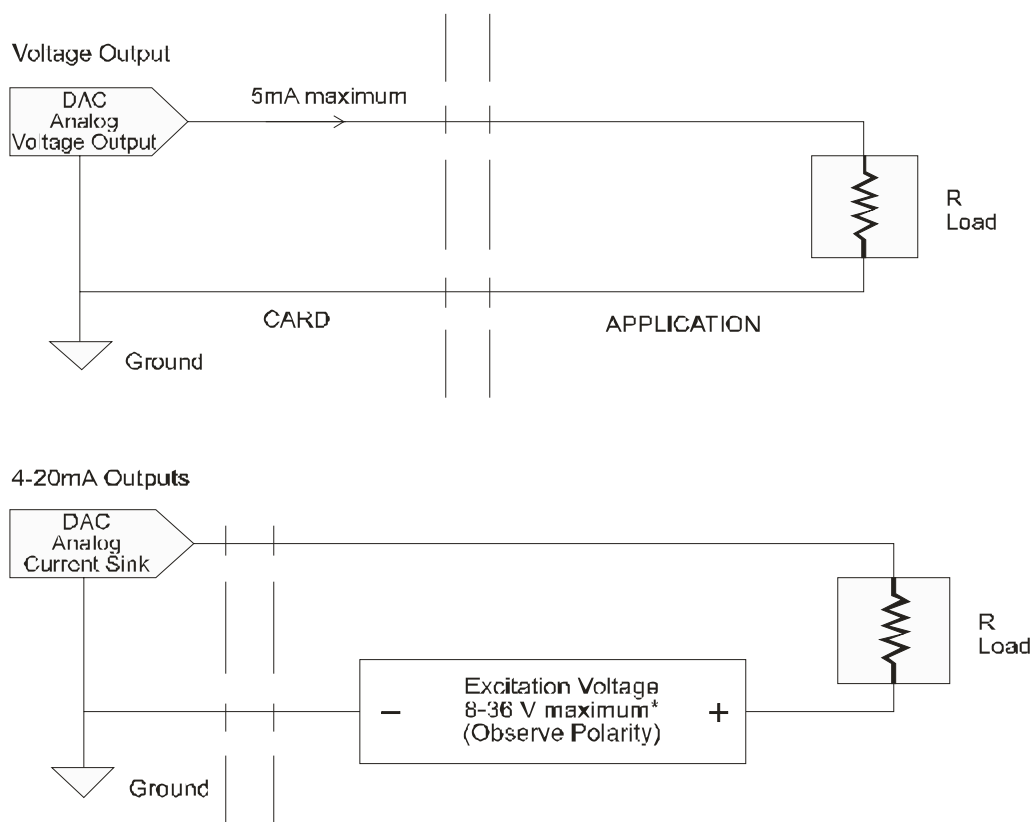
IDC 8-Pin Header Male



**Table 6-3:** Connector P4, External 12V Input, 8-Pin Header

Pin	Signal	Pin	Signal
1	Ground	2	n.c.
3		4	+12V
5		6	n.c.
7		8	n.c.

The board does not require +12V but the user's 4-20mA circuit might. +12V is brought to pin 34 of connector P1 through a resettable fuse. This +12V may come from the PC/104 bus or from an external source. If the voltage comes from the PC/104 bus then a jumper should be placed on the pins of JP32 (next to P4) labeled LCL (local). If the voltage comes from a source connected to P4 then a jumper should be placed on the pins of JP32 labeled EXT. Note that if the voltage for the user's 4-20mA circuit comes from an external source that it isn't limited to 12V.



**Figure 6-1:** Signal Connection

# Appendix A: Technical Specifications

## ANALOG OUTPUTS

- 8 Channels
- 100,000 Conversions per Second, all channels simultaneously
- 12 Bit Resolution
- ARB Onboard memory storage of 128K bytes
- Output Ranges, 0-5V 0-10V +5V +10V
- Eight 4-20mA current sink outputs (external 8 to 36VDC excitation required)
- +2 counts D/AC Relative Accuracy (typical)
- 8uS D/AC Settling Time (typical, to 3/4 scale)
- +0.4 % of Full Scale D/AC Offset Error (typical)
- +0.1 % of Full Scale D/AC Gain Error (typical)
- Drive Capability of 5mA per channel, Outputs are Short-Circuit Protected
- 30mA cumulative total drive from all D/ACs
- 4.096V Voltage Reference

## Counter/Timer

- Type 82C54
- 3 x 16-Bit Down-Counters
- Counter 0 is an IRQ source (clock-tick interrupt) and frequency source (counter 0 output is available at P2 connector pin 4)
- Counters 1 & 2 are chained (32-bit resolution) and dedicated as the ARB clock (an interrupt is available based on the DAC update from the ARB)
- 10MHz Input Clock Frequency

## General

- +5V @ 210mA Power Consumption (typical, no load on the outputs)
- On-board DC/DC Converter allows operation on +5V Power
- Interrupt requests may be generated on channels 3-7, 10-12 and 14-15
- Environment Tolerance: 0-70C, 5% to 95% Humidity (non-condensing)  
(-40 to +85C available with special order)

# Appendix B: 82C54 Counter Timer Operation

The board contains one type 8254 programmable counter/timer. The 8254 consists of three independent, 16-bit, presettable down-counters. Each counter can be programmed to any count between 2 and 65,535 in binary format, depending on the mode chosen. The programmed value is a divisor, the output frequency equals the input frequency divided by the programmed value.

In this manual these three counter/timers are designated Counter/Timer 0, Counter/Timer 1, and Counter/Timer 2.

Counter/Timer 0 is a 16-bit counter with a 10MHz input clock. The output of Counter/Timer 0 is available at connector P2 pin 4. The gate signal for counters 1 & 2 is controlled via software at bit 1 (HIGH = off = paused, LOW = count enabled = power-on default) of the register at Base + 10h. This bit is used to pause the ARB. Counter/Timers 1 and 2 are concatenated by the card to form a single 32-bit counter. The input of the counter is fixed at 10MHz. The output of is available at connector P2 pin 1.

## OPERATIONAL MODES

The 8254 modes of operation are described in the following paragraphs to familiarize you with the versatility and power of this device. For those interested in more detailed information, a full description of the 8254 programmable interval timer can be found in the Intel (or equivalent manufacturers) data sheets. The following conventions apply for use in describing operation of the 8254 :

Clock:	A positive pulse into the counter's clock input.
Trigger:	A rising edge input to the counter's gate input.
Counter Loading:	Programming of a binary count into the counter.

### Mode 0: Pulse on Terminal Count

After the counter is loaded, the output is set low and will remain low until the counter decrements to zero. The output then goes high and remains high until a new count is loaded into the counter. A trigger enables the counter to start decrementing.

### Mode 1: Retriggerable One-Shot

The output goes low on the clock pulse following a trigger to begin the one-shot pulse and goes high when the counter reaches zero. Additional triggers result in reloading the count and starting the cycle over. If a trigger occurs before the counter decrements to zero, a new count is loaded. Thus, this forms a re-triggerable one-shot. In mode 1, a low output pulse is provided with a period equal to the counter count-down time.

### Mode 2: Rate Generator

This mode provides a divide-by-N capability where N is the count loaded into the counter. When triggered, the counter output goes low for one clock period after N counts, reloads the initial count, and the cycle starts over. This mode is periodic, the same sequence is repeated indefinitely until the gate input is brought low.

### Mode 3: Square Wave Generator

This mode operates periodically like mode 2. The output is high for half of the count and low for the other half. If the count is even, then the output is a symmetrical square wave. If the count is odd, then the output is high for  $(N+1)/2$  counts and low for  $(N-1)/2$  counts. Periodic triggering or frequency synthesis are two possible applications for this mode. Note that in this mode, to achieve the square wave, the counter decrements by two for the total loaded count, then reloads and decrements by two for the second part of the wave form.

### Mode 4: Software Triggered Strobe

This mode sets the output high and, when the count is loaded, the counter begins to count down. When the counter reaches zero, the output will go low for one input period. The counter must be reloaded to repeat the cycle. A low gate input will inhibit the counter. This mode can be used to provide a delayed software trigger for initiating A/D conversions.

### Mode 5: Hardware Triggered Strobe

In this mode, the counter will start counting after the rising edge of the trigger input and will go low for one clock period when the terminal count is reached. The counter is retriggerable. The output will not go low until the full count after the rising edge of the trigger.

## PROGRAMMING

On this card the 8254 counters occupy the following addresses (hex):

Base Address + 14: Read/Write Counter 0  
Base Address + 15: Read/Write Counter 1  
Base Address + 16: Read/Write Counter 2  
Base Address + 17: Write to Counter Control register

The counters are programmed by writing a control byte into a counter control register. The control byte specifies the counter to be programmed, the counter mode, the type of read/write operation, and the modulus. The control byte format is as follows:

B7	B6	B5	B4	B3	B2	B1	B0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC0-SC1: These bits select the counter that the control byte is destined for.

SC1	SC0	Function
0	0	Program Counter 0
0	1	Program Counter 1
1	0	Program Counter 2
1	1	Read/Write Cmd.*

\* See Chapter on READING AND LOADING THE COUNTERS.

RW0-RW1: These bits select the read/write mode of the selected counter.

RW1	RW0	Counter Read/Write Function
0	0	Counter Latch Command
0	1	Read/Write LS Byte
1	0	Read/Write MS Byte
1	1	Read/Write LS Byte, then MS Byte

M0-M2: These bits set the operational mode of the selected counter.

MODE	M2	M1	M0
0	0	0	0
1	0	0	1
2	X	1	0
3	X	1	1
4	1	0	0
5	1	0	1

BCD: Set the selected counter to count in binary (BCD = 0) or BCD (BCD = 1).

## READING AND LOADING THE COUNTERS

If you attempt to read the counters on the fly when there is a high input frequency, you will most likely get erroneous data. This is partly caused by carries rippling through the counter during the read operation. Also, the low and high bytes are read sequentially rather than simultaneously and, thus, it is possible that carries will be propagated from the low to the high byte during the read cycle.

To circumvent these problems, you can perform a counter-latch operation in advance of the read cycle. To do this, load the RW1 and RW2 bits with zeroes. This instantly latches the count of the selected counter (selected via the SC1 and SC0 bits) in a 16-bit hold register. (An alternative method of latching counter(s) which has an additional advantage of operating simultaneously on several counters is by use of a readback command to be discussed later.) A subsequent read operation on the selected counter returns the held value. Latching is the best way to read a counter on the fly without disturbing the counting process. You can only rely on directly read counter data if the counting process is suspended while reading, by bringing the gate low, or by halting the input pulses.

For each counter you must specify in advance the type of read or write operation that you intend to perform. You have a choice of loading/reading (a) the high byte of the count, or (b) the low byte of the count, or (c) the low byte followed by the high byte. This last is of the most general use and is selected for each counter by setting the RW1 and RW0 bits to ones. Of course, subsequent read/load operations must be performed in pairs in this sequence or the sequencing flip-flop in the 8254 chip will get out of step.

The readback command byte format is:

B7	B6	B5	B4	B3	B2	B1	B0
1	1	CNT	STA	C2	C1	C0	0

- CNT: When is 0, latches the counters selected by bits C0-C2.  
 STA: When is 0, returns the status byte of counters selected by C0-C2.  
 C0, C1, C2: When high, select a particular counter for readback. C0 selects Counter 0, C1 selects Counter 1, and C2 selects Counter 2.

You can perform two types of operations with the readback command. When CNT=0, the counters selected by C0 through C2 are latched simultaneously. When STA=0, the counter status byte is read when the counter I/O location is accessed. The counter status byte provides information about the current output state of the selected counter and its configuration. The status byte returned if STA=0 is:

B7	B6	B5	B4	B3	B2	B1	B0
OUT	NC	RW1	RW2	M2	M1	M0	BCD

- OUT: Current state of counter 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 load depends on the configuration selected. Until the count is loaded into the counter itself, it cannot be read.  
 RW1, RW0: Read/Write command.  
 M2, M1, M0: Counter mode.  
 BCD: BCD = 0 is binary mode, otherwise counter is in BCD mode.

If both STA and CNT bits in the readback command byte are set low and the RW1 and RW0 bits have both been previously set high in the counter control register (thus selecting two-byte reads), then reading a selected counter address location will yield:

- 1st Read:       Status byte
- 2nd Read:       Low byte of latched data
- 3rd Read:       High byte of latched data

After any latching operation of a counter, the contents of its hold register must be read before any subsequent latches of that counter will have any effect. If a status latch command is issued before the hold register is read, then the first read will read the status, not the latched value.

## Customer Comments

If you experience any problems with this manual or just want to give us some feedback, please email us at: ***manuals@acesio.com***. Please detail any errors you find and include your mailing address so that we can send you any manual updates.



10623 Roselle Street, San Diego CA 92121  
Tel. (858)550-9559 FAX (858)550-7322  
[www.acesio.com](http://www.acesio.com)