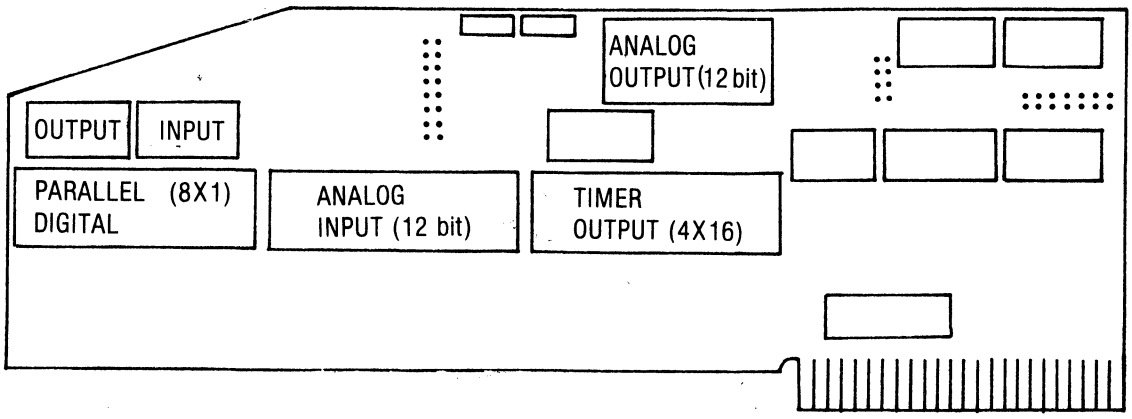
Interactive Microware, Inc.
P.O. Box 771
State College, Pa 16801
(814) 238-8294



ADALAB™ HARDWARE MANUAL

By Paul K. Warme

INTERACTIVE MICROWARE, INC.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.Ø AN OVERVIEW OF THE ADALAB DATA ACQUISITION SYSTEM

ADALAB(tm)[1] is a microcomputer system that is specifically designed for applications in the laboratory. We call ADALAB a system because it includes both the hardware (the APPLE (tm) computer and ADALAB interface board) and extensive software to make this computer useful in every lab. Any instrument that can output voltages to a recorder may be connected to ADALAB. Some examples of such instruments are spectrophotometers, fluorometers, flame photometers, pH meters, chromatography detectors (GC or HPLC), conductivity meters and oxygen monitors. Instruments that are controlled by a voltage may also be connected to ADALAB. This category includes strip chart recorders, proportional control valves and pumps, temperature or flow controllers and electrochemical instruments. ADALAB is also useful for controlling any instrument that has multiple switch inputs or contact closure outputs. In addition, ADALAB can communicate with "intelligent" instruments and other computers having parallel or serial (optional) input/output capabilities.

[1] ADALAB is a trademark of Interactive Microware, Inc. APPLE is a trademark of Apple Computer, Inc.

2.0 ADALAB HARDWARE SUMMARY

>>> Analog to Digital (A/D) Converter Subsystem
* Reads voltages from your instruments with a precision of 0.025% (12 bits) and overall accuracy better than 0.1%
* Jumper-selectable voltage ranges ±4V, ±2V, ±1V or ±0.5V
* Dual slope integrating A/D converter smooths out noisy signals
* True differential input and automatic internal zeroing enhance accuracy
* Up to 20 voltage readings per second; faster response than most recorders

>>> Digital to Analog (D/A) Converter Subsystem
* Sends control voltages to your instruments with 0.025% precision (12 bits) and overall accuracy better than 0.1%
* Jumper-selectable voltage ranges ±4V, ±2V, ±1V or ±0.5V
* D/A conversion rate up to 50,000 conversions per second

>>> Digital and Parallel Input/Output Subsystem
* 8 digital input bits and 8 digital output bits or 16 bidirectional bits individually selectable as inputs or outputs
* TTL-compatible signal levels (one TTL load or drive)
* Versatile handshaking signals, interrupt and enable circuitry
* Latching registers store I/O information on cue

>>> Real-Time Clock/Timer Subsystem
* 32 bit countdown timer may be set for any time interval from 10 microseconds to 100 minutes. May be used as a time of day clock reading in hours, minutes and seconds
* Two 16 bit timer/counters may be configured as an interval timer, event counter, pulse generator, square wave generator or shift register

3.Ø INSTALLING YOUR ADALAB INTERFACE CARD

3.1 Unpacking

The following items are included with each ADALAB interface card:

3	16 pin DIP cables (36")
1	Self-Test Adapter Module or optional Analog Junction Box
1	ADALAB Hardware Manual
1	QUICKI/O Software Manual
1	QUICKI/O Software Diskette
1	Warranty Card and Registration Form
1	Evaluation Form

STATIC ELECTRICITY WARNING: The large (24 and 4Ø pin) integrated circuits on the ADALAB card may be damaged by static electricity. Before removing the protective wrapping or handling the ADALAB card, you should ground yourself by touching a water faucet or the metal case on the APPLE computer's power supply.

3.2 Selecting Jumper Options

Looking at the ADALAB card on the component side with the gold edge connector at the bottom right, you will see two groups of metal pins on the upper half of the card (see Figure 1). Group 1 consists of 2 vertical columns of 8 pins near the center of the board; Group 2 consists of 2 vertical columns of 4 pins. On some of these pins, you will see black plastic jumpers, which connect pairs of pins together. As shipped, the jumpers are set for the ±1V range.

3.3 GROUP 1 Jumper Options (A/D Converter)

The two jumpers on the Group 1 pins are used to select the range of the Analog to Digital (A/D) converter. For the ±4 Volt range, one jumper should be on the pair of pins closest to the top edge of the card and the other jumper should be on the fifth pair down from the top edge. For the ±2V range, move each jumper down one position. If you move each jumper down one more position, you will select the ±1V range. Likewise, moving down one more position selects the ±Ø.5V range for the A/D converter. Note that there should always be three vacant pairs of pins between the two jumpers on the Group 1 pins. It will help if you remember that the voltage range decreases as you move the jumpers downward.

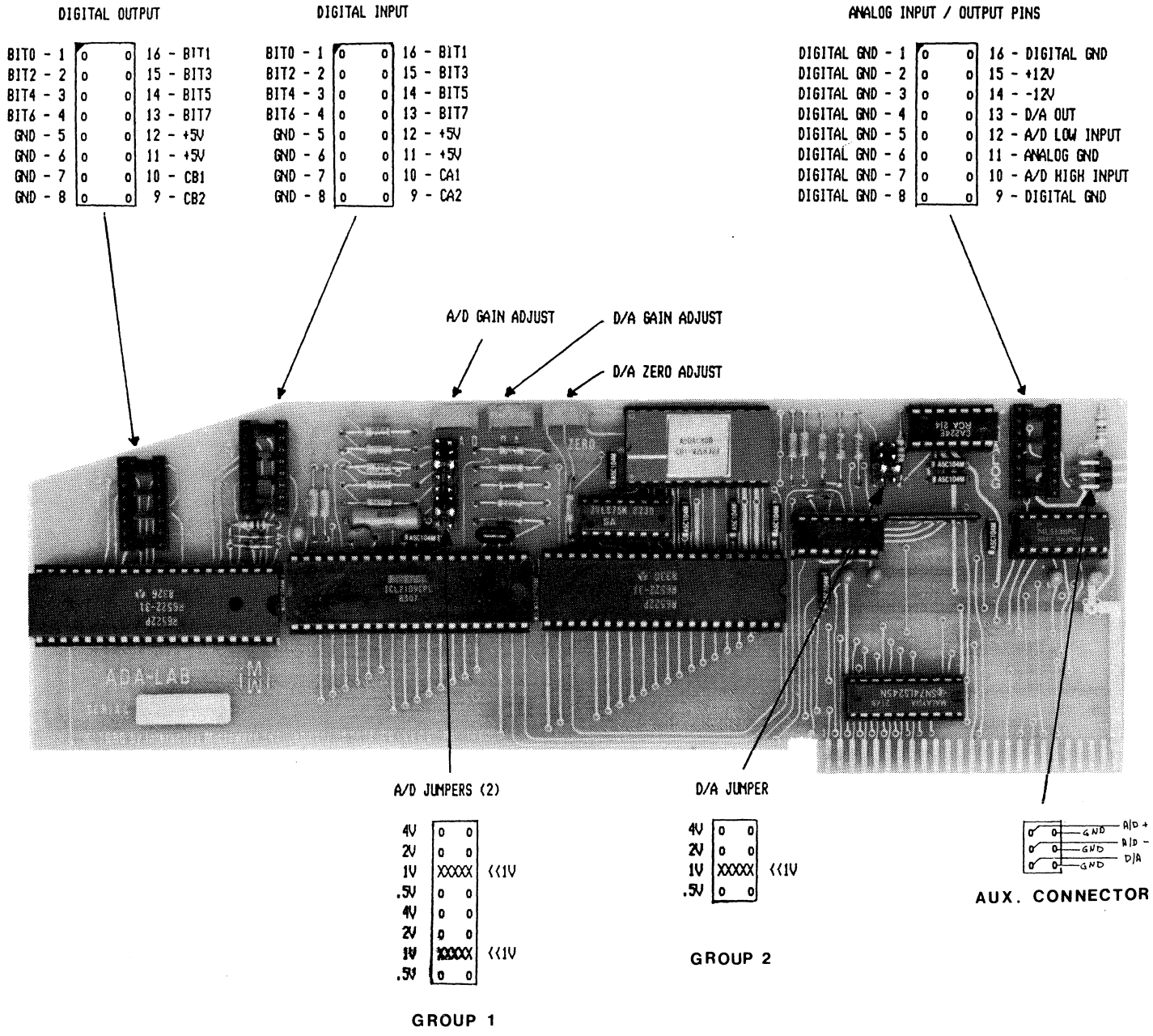


FIGURE 1. The ADALAB Interface Card.

3.4 Group 2 Jumper Options (D/A Converter)

The second group of pins, running vertically on the right (see Figure 1) selects the voltage range for the D/A converter. As was the case for GROUP 1, the voltage range decreases from ±4V (top pair) to ±Ø.5V (bottom pair) as you move the single jumper downward.

3.5 Attaching the Cables to ADALAB

Looking at the ADALAB interface card or referring to Figure 1, you will note that there are three empty 16-pin sockets which are used for attaching the ribbon cables. The Digital (or Parallel) Output socket (labelled OUT) is at the upper left corner, the Digital Input socket (labelled IN) is directly to the right of it, and the Analog Input/Output socket (labelled ANALOG) is at the upper right corner of the ADALAB card. The signals on each pin of these sockets are summarized in Figure 1 and in Table I and II. Pin 1 is in the upper left corner of each socket. It is IMPORTANT to connect the cables in such a way that pin 1 of the cable plug (the pin marked with a white dot) is inserted in the upper left corner of the socket (the corner that is marked white.) Since the plugs on the two ends of the cable are opposite in orientation, it is possible to plug in the cables so that they extend either to the left or to the right. As shown in Figure 2, you should plug in the end of the cable that makes the cable extend to the right (toward the back of the computer.)

3.6 Removing Cables from Sockets

Please be careful when removing the cables from the sockets on ADALAB; the pins are quite fragile and tend to break off when they are bent more than once. The best method for removing cables is to use an integrated circuit extractor (a U-shaped tool with hooks on each end; available in most electronics stores.) A more risky approach is to lever up one end of the cable plug slightly with a small screwdriver and then, lever up the other end slightly. Alternate between the two ends, prying up a little bit at a time until the cable is free.

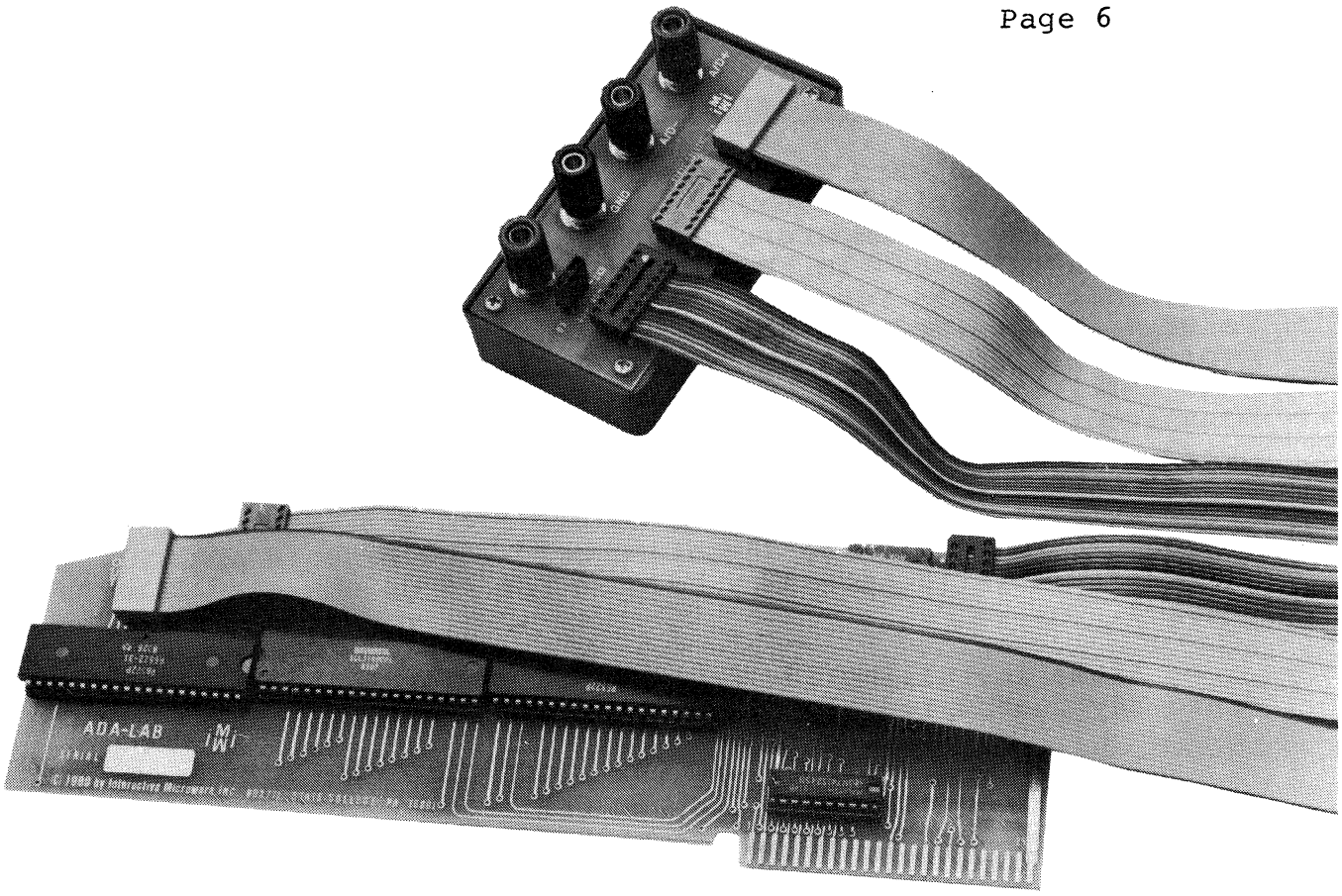


FIGURE 2. Attaching the Cables to ADALAB.

TABLE I. The ANALOG INPUT/OUTPUT Socket.

Pin #	Signal Description	Self-Test Adapter
1-9	Digital Ground	
10	A/D high input	←
11	Analog Ground	←
12	A/D low input	←
13	D/A high output	←
14	-12 V (50mA) out	
15	+12 V (50mA) out	
16	Digital Ground	

TABLE II. The DIGITAL INPUT/OUTPUT Sockets.

Pin #	INPUT Port A	Self-Test Adapter	OUTPUT Port B
1	Bit 0	——	Bit 0
16	Bit 1	——	Bit 1
2	Bit 2	——	Bit 2
15	Bit 3	——	Bit 3
3	Bit 4	——	Bit 4
14	Bit 5	——	Bit 5
4	Bit 6	——	Bit 6
13	Bit 7	——	Bit 7
5,6,7,8	GND		GND
9	CA2	⟋⟍	CB2
10	CA1	⟍⟋	CB1
11,12	+5V		+5V

3.7 Selecting the Slot and Plugging in the ADALAB Card

The ADALAB card will work in any of slots 1 through 7 of any Apple II+ or Apple IIe computer. Up to four ADALAB cards may be used in a single computer, and they need not be plugged into consecutive slots. In the Apple II+, the slots are numbered from Ø (on the left as you face the keyboard) to 7 (on the right); slot Ø is reserved for RAM memory cards and should not be used for ADALAB. In the Apple IIe, the slots are numbered from 1 (on the left) to 7 and slot Ø is replaced by a different type of connector (towards the front) that is reserved for an 8Ø-column card. If you plan to use an Apple IIe with an 8Ø-column card you should avoid using slot 3 for an ADALAB card. If you must, then POKE 49163,Ø to disable the 8Ø-column card and enable the ADALAB card. To re-enable the 8Ø-column card (and disable ADALAB), POKE 49162,Ø. This is necessary because the 8Ø-column card "shares" slot 3.

After installing the cables, you may plug the ADALAB card into any unoccupied slot from 1 to 7 in the computer, as follows:

1. TURN OFF THE POWER TO THE APPLE COMPUTER; but leave the 3-prong power plug in the wall socket, because this ensures that the power supply case is grounded.

2. Remove the top cover by lifting it at the back of the Apple computer.

3. Touch the metal case of the power supply (the large rectangular metal box on the left side) to discharge any static electricity from your body.

4. Plug the gold-plated bottom edge of the ADALAB card into one of the unoccupied slot connectors in the Apple computer. Make sure that the card is pushed down all the way into the connector.

5. For the Apple II+, route the three cables through one of the grooves in the back of your computer. For the Apple IIe, the cables can be folded over the top edge of the back side for convenience of removal during testing phases, or routed through one of the holes in the back side for a more permanent set-up.

6. Replace the cover on the Apple computer.

3.8 Connecting Cables to Adapter Modules
(see Figure 2)

If this is the first time you have used your ADALAB, we recommend that you connect the three cables to the self-test adapter, in preparation for the self-test procedure, which will be described later. When you connect the cables from ADALAB to the self-test adapter or signal conditioner modules, be sure that

the white mark near pin 1 is inserted in pin 1 of the socket, labelled with white. Also, be sure to connect each cable only to the proper socket, as described in this manual or the signal conditioner manual. The analog I/O cable should only be plugged into a socket marked ANALOG. The digital input cable should only be plugged into a socket marked INPUT or DIGITAL INPUT, and the digital output cable should go only in a socket marked OUTPUT, DIGITAL OUTPUT, or HEADER (on an ADA-MUX adapter.)

WARNING! Improper insertion of the cables could cause permanent damage and void your warranty!

4.Ø CALIBRATION PROCEDURES

The ADALAB interface card is calibrated at the factory for ±1V operation. To determine if this calibration is satisfactory, (assuming you have not changed the voltage range jumpers) try the QUICKSAMPLE demonstration program on the QUICKI/O software disk.

NOTE: The A/D converter reading is affected by temperature changes and the reading tends to change as the computer warms up. Therefore, it is a good idea to let the computer run for about 15 minutes before calibrating the A/D converter. For most applications, the actual value returned by the A/D converter is not as important as its linearity, its short-term stability, and its ability to measure changes in voltage precisely.

4.1 QUICKSAMPLE Self-Test Procedure
(Menu Option Nine)

First, plug the three cables from ADALAB into the self-test adapter as described earlier. (Plastic jumpers should be in place for this test. See Figure 3.) Then, insert the QUICKI/O disk in drive 1 and turn on the computer or, if it is already on, type RUN QUICKSAMPLE. The QUICKSAMPLE program determines which slots contain ADALAB cards, initializes the hardware and then runs the self-test routines on Channel Ø (the first ADALAB card.) You may consult the QUICKI/O Software Manual for additional information about the QUICKSAMPLE program and the self-test procedure.

For our present purposes, it is sufficient to know that the ADALAB card is working properly if the Real Time Clock Test, the Timer 1 Test, the Digital I/O Test and the Parallel I/O Test print OK (and don't print ERROR) on the display screen. The self-test procedure ends with the Analog I/O Test and if the MAX ERROR values printed on the screen are greater than ±1Ø, then calibration is necessary. At the bottom of the screen, it will say "PRESS ANY OTHER KEY TO QUIT." At this point, press the RETURN key; a menu (list of options) will then appear at the top of the screen.

4.2 Voltmeter Calibration Procedure

If calibration is shown to be necessary through QUICKSAMPLE self-testing or after changing the voltage range jumper options, it is necessary to recalibrate the D/A and A/D converters. Take a look at Figure 1 and notice that there are 3 plastic potentiometers labelled A/D, D/A and ZERO, at the top near the center of the ADALAB board. Remove the cover of your Apple computer and find these potentiometers on the ADALAB card; note that the adjustment screws face upward so that you can adjust

them without removing the ADALAB card. For calibration, you should use a good-quality digital voltmeter with at least 3.5 digit accuracy. A less accurate voltmeter may be used, but the overall accuracy of the calibration will only be as accurate as your voltmeter. If you are using a voltmeter that also measures resistance and current, set it up to measure voltage and select a voltage range appropriate for the D/A voltage range (usually ±1 Volt). Attach the + lead of your voltmeter to the connector on the self-test adapter marked DA and attach the - lead (ground) of your voltmeter to the GND lead on the self-test adapter. Also, make sure that the black plastic jumpers are connecting DA to AD+ and AD- to GND.

Assuming you are still in the QUICKSAMPLE program, press 8 to select the STABILITY TEST option. Next, the program will ask:

OUTPUT VOLTAGE (-2047 TO 2047)?

This selects the voltage of the D/A converter, which must be between -2047 (full scale negative) and +2047 (full scale positive). The actual voltage corresponding to full scale depends on the D/A range jumpers.

1. The first step of the calibration procedure involves adjusting the zero offset of the D/A converter. Therefore, enter 0 as the OUTPUT VOLTAGE and press RETURN. The screen will display a histogram showing the deviations of successive A/D values from the initial A/D value. During the calibration procedure, you should ignore everything except the value labelled V1 at the bottom of the screen; this is the current A/D value. Adjust the ZERO pot (nearest to the back of the computer, see Figure 1) to yield a value V1=0. Then, press RETURN to return to the QUICKSAMPLE menu.

2. Next, we shall calibrate the D/A full scale voltage. Select option 8 (STABILITY TEST) again and enter 2047 as the OUTPUT VOLTAGE. The voltage reading on your voltmeter should now measure close to the maximum voltage that you have selected by means of the D/A jumper. To adjust the maximum D/A voltage, use a small screwdriver to turn the screw on the middle potentiometer (see Figure 1) marked D/A. Turning the screw clockwise decreases the voltage. For example, if the D/A range jumper is set for ±1 Volt, turn the D/A pot until your voltmeter reads 1.000 Volt.

3. After you have adjusted the D/A converter voltage, you should adjust the range of the A/D converter. This is done by turning the screw on the potentiometer marked A/D, which is closest to the front of the computer (see Figure 1). The A/D converter reading (V1=) is printed on the screen, and you should turn the screw until V1=2045 (slightly less than the maximum value of 2047). Turning the screw clockwise decreases the value of V1. If

Vl=4095, this means that the input voltage is beyond the maximum, so you should turn the screw back a little (clockwise). You will note that a very small change in the screw position changes the voltage reading, so you must turn the screw in very small increments as you approach the final value.

This completes the calibration procedure. However, you can re-select option 8 and verify that other OUTPUT VOLTAGE values give Vl values that are proportional and that the D/A output and A/D input are linear.

5.Ø HOW TO CONNECT YOUR INSTRUMENTS TO ADALAB

CAUTION: Before you connect any external signals to ADALAB, make sure that their voltage range is compatible. Consult the instrument manufacturer's literature or seek help from an electronics technician if you are uncertain. Analog (continuously variable) voltages must be within the range of ±4 Volts (or less depending on the range jumper) in order to be measurable by ADALAB. Digital (discrete on or off) signals must be between -Ø.3 and +Ø.4 Volts for the "off" state and between 2.4 and 5.Ø Volts for the "on" state.

WARNING: Analog voltages greater than ±12 Volts or Digital hardware voltages outside the range of -Ø.3 to 5.Ø Volts can cause hardware damage to ADALAB and/or to your Apple computer. NEVER connect 12Ø Volt AC signals directly to ADALAB. Failure to observe these precautions will void the warranty.

The following section describes some of the many ways to connect external signals to ADALAB. Consult Figure 1 and Tables I and II for information about which signals are on which pins of the cables and connectors. Pin 1 is generally denoted by a white dot or a 1.

5.1 Connecting Through a Self-Test Adapter

The self-test adapter included with the ADALAB system is used primarily for running the self-test procedure in the QUICKSAMPLE program on the QUICKI/O disk. In addition there is a set of connector pins (shipped with plastic jumpers over them) marked DA, AD+, AD- and GND. These are used to connect external signals.

Remove the black plastic jumpers connecting the posts. See Figure 3. Attach the positive (high) wire from your instrument to the AD+ post and attach the negative (low) wire to the AD- post. Small alligator clips work fairly well, but be careful to avoid touching any of the other posts. Another approach involves inserting the wire from your instrument into one half of the plastic jumper and then sliding the other half of the jumper over the post. For a more permanent connection, you should consider soldering wires to the posts on the back side of the self-test adapter card; these wires may then be soldered to banana jacks, BNC jacks or any other style of connector.

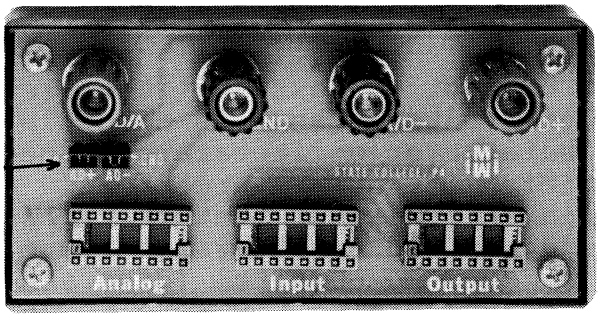


FIGURE 3. The Analog Junction Box (AJB).

The self-test adapter (STA) is the same box without the four binding posts. Both can be used for the QUICKSAMPLE self-testing procedure. To connect external signals, jumpers must be removed.

5.2 Connecting Analog Signals to the Analog Junction Box

This new variety of self-test adapter includes four banana jacks for the D/A, A/D+, A/D- and GND signals. In addition to banana jacks you can connect stripped wires, spade lugs and other connectors to these terminals. Be sure to remove the black plastic self-test jumpers before you connect any external signals to the banana jacks.

5.3 Connecting ADALAB to a Prototyping Breadboard

ADALAB's Analog, Digital Input and Digital Output cables can be plugged into a solderless-terminal breadboard, which is available from IMI or can be purchased in many electronics stores. Each of ADALAB's signals is then readily accessible for wiring to other electronic components or other types of connectors.

5.4 Configuring Special ADALAB Cables

It is possible to eliminate ADALAB's ribbon cables entirely and build special cables from your instruments that plug directly into the 16-pin sockets on the ADALAB card. 16-pin "headers" for this purpose are available from IMI or from electronics supply stores.

For analog signals, a special 6-pin connector is available on the edge of the ADALAB card closest to the back of the computer. The signal appearing on each of these six pins is indicated in Figure 1. Shielded cables that mate with these pins are available from IMI.

5.5 Soldering Wires to 16-pin Sockets

It is not a good idea to solder wires directly to the pins of the 16 conductor cables supplied with ADALAB. However, a 16-pin socket that mates with the cable can be purchased for about 50 cents from any electronics store. You can solder wires from your instrument to a 16-pin socket and plug an ADALAB cable into this socket. This approach offers the advantage that several alternative instruments can be connected or disconnected without opening the case of the computer.

5.6 Connecting Instruments to ADA-BUFF, ADA-MUX, ADA-AMP and Other IMI Accessories

Many signal conditioners are available from IMI for use with ADALAB. Generally, these accessories include sockets for particular ADALAB cables and convenient screw terminals for attaching wires from your instruments. For instance, the ADA-BUFF is particularly convenient for attaching digital signals; the ADA-MUX is an 8 channel multiplexer for connecting multiple analog devices; and the ADA-AMP is used to amplify signals below the ADALAB minimum input voltage. ADA-AMP is also available with an integral multiplexer option. For more details, consult the manual that comes with each accessory.

6.Ø ATTENUATING, AMPLIFYING AND FILTERING ANALOG SIGNALS

6.1 Attenuating High Voltage Signals

Direct current (DC) voltages larger than ±4 Volts can easily be attenuated by means of fixed or variable resistors (potentiometers). For example, a 1Ø Volt signal could be attenuated to 4 volts by the circuits in Figure 4:

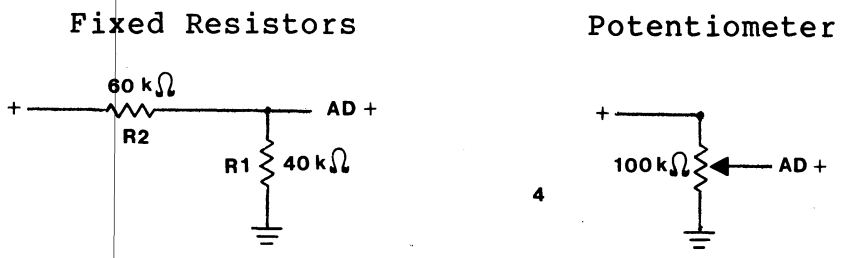


Figure 4: Attenuation of a High Voltage Signal

Generally, the total resistance (R1+R2) should be at least 1ØØØØ ohms. In a fixed resistance divider circuit, the output voltage may be calculated as R1/(R1+R2).

6.2 Amplifying Low Voltage Signals

If you need to measure low voltages, you should first consider using the ±Ø.5 Volt range on ADALAB. However, if your voltage is less than about ±Ø.1 Volt, we suggest using ADA-AMP to amplify the voltage. ADA-AMP contains a high-quality instrumentation amplifier that amplifies the difference between the high and low input signals; thus, noise that is common to both the high and low inputs is effectively ignored.

6.3 How to Deal With Noisy Signals

Induced electrical interference (or "noise") can be a problem when you are transmitting analog voltage signals between certain instruments and the ADALAB interface card. Here are some suggestions for minimizing noise:

1. Keep the signal cables from your instrument to ADALAB as short as possible. Cables pick up induced noise from their surroundings.

2. Avoid use of ribbon cables to carry analog signals. Twist the two signal wires around each other so that any induced noise will be the same for both wires; ADALAB's differential amplifier rejects noise which is equal on

both the high and low signals.

3. Better yet, use a shielded two-conductor cable to carry signals from your instrument to ADALAB. The outer shield should be connected either to the signal ground of your instrument or to the ground terminal on ADALAB (the shield should be connected to one end or the other, but not to both ends).

4. If your instrument has a 3-prong AC line cord, connect a wire from the chassis of your instrument to the case of the Apple computer's power supply. This equalizes the ground potential. Another way to accomplish this is to plug your Apple computer into the same electrical outlet that is used for your instrument.

5. The input resistance (impedance) of ADALAB is very high (8.2 megohms). High input resistance makes signal wires act as antennas that can readily pick up noise. If the output resistance of your instrument is lower than 10 Kohms (as is the case for most instruments), it is desirable to decrease the input resistance of ADALAB. You can do this by installing input resistors to ground from both the A/D+ and A/D- terminals of ADALAB. The appropraite value for input resistance depends entirely upon the output impedance (drive capability) of your external equipment; if the applied voltage decreases significantly when grounded through the input resistor, then you should increase its value. Normally, the input resistance should be at least 10 Kohms; some instruments may be damaged if the input resistance is too low or if their signal leads are shorted together.

6. Filter the output voltage from your instrument. A low pass filter can reduce noise very effectively, at the expense of lengthening the response time. For example, a 20 Kohm (20 E3) resistor in series with a 0.5 uF (0.5 E-6) bipolar capacitor to ground will yield a time constant of 10 milliseconds (20E3 * 0.5 E-6). Since ADALAB has an input resistance of 8.2 megohms to ground, the voltage reading will be attenuated by a factor of 20E3 / 8.2E6 = 2.44E-3. Of course, larger resistors will result in longer time constants and greater attenuation. A low pass filter looks like this:

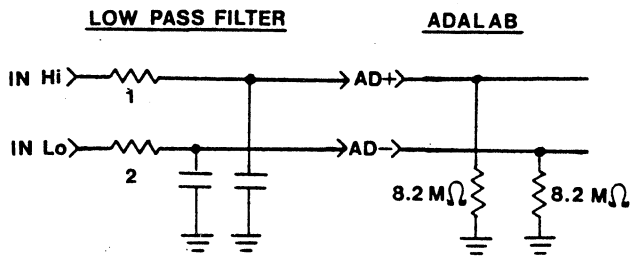


Figure 5: A Typical Low-Pass Filter

The two resistors R1 and R2 should be matched as closely as possible for highest accuracy. The capacitors C1 and C2 should be a bipolar type (mylar, polyester or polystyrene, but <u>not</u> electrolytic or tantalum) to allow for both positive and negative voltages.

7.Ø THE ANALOG TO DIGITAL CONVERTER

7.1 Theory of Operation

ADALAB's A/D converter is called a dual-slope (or integrating) A/D converter. This means that the input voltage is measured by charging a capacitor over a precisely determined interval of time. Very little current is drawn from the external equipment because of the high input impedance of the reference voltage. The time required to discharge the capacitor is also proportional to the input voltage. Since the discharge time is a number that can be read by the computer, we obtain a number that is proportional to the input voltage. If we plot the charge on the capacitor as a function of time, we will see a straight line with a certain slope as the capacitor charges and another straight line with a different slope as the capacitor discharges. This is the origin of the term "dual-slope."

There is another common type of A/D converter which is called a "successive approximation" A/D converter. It uses a D/A converter and a voltage comparator to measure voltages. First, the most significant bit of the digital value sent to the D/A converter is turned on and the resulting D/A output voltage is compared to the A/D input voltage that is to be evaluated. If the D/A output voltage is larger than the A/D input voltage, the most significant bit of the digital value sent to the D/A converter is turned off again. Then, the second significant bit of the D/A digital value is turned on and the D/A output voltage is once more compared to the A/D input voltage. If the D/A voltage is less than the A/D voltage in this case, then this bit of the D/A digital value is left on. This process of turning bits on or off and comparing voltages continues until each bit of the D/A digital value has been tested by this successive approximation technique. At this point, the D/A output voltage should be exactly the same as the A/D input voltage; therefore, the digital value sent to the D/A converter is an accurate measure of the A/D input voltage.

Both the dual-slope (DS) method and the successive approximation (SA) method have certain advantages and disadvantages. The SA method is fast; typically, a measurement is completed within 2Ø microseconds. The DS method is considerably slower; typically a conversion takes about 5Ø milliseconds.

The designers of ADALAB chose a dual-slope converter because the successive approximation method runs into problems when the input signal is "noisy", as is the case with the signals created by most laboratory instruments. Consider what would happen if a voltage fluctuation occurred during the period when a SA converter is trying to measure a voltage. Suppose that the

voltage is momentarily higher than average. Then one or more bits of the D/A output value will be left on which should have been turned off. As the subsequent bits are tested, the voltage may go back down to the average value, so all subsequent bits will be turned off, because the D/A output voltage is already higher than the A/D input voltage. To counteract this problem, it is usually necessary to use a sample and hold amplifier with a SA converter. The sample and hold amplifier samples the input voltage for a very brief period (aperture time less than one microsecond) and then holds that voltage while the SA converter measures it. But now, the measured voltage contains information only about the instantaneous voltage during that very short aperture time of the sample and hold amplifier. If the fluctuations of the input voltage are significantly large, it will be necessary to average many SA conversion values to obtain an accurate indication of the average input voltage. Because many conversions must be averaged, the effective rate of a SA converter for noisy signals is much slower than the specified maximum rate. Also, extra memory space is needed to store the values to be averaged and the program is more complicated because of the need to average multiple values.

How does the dual-slope method avoid this problem? A DS converter automatically averages (integrates) the input signal during a major portion of the conversion time (one-fourth of the time in the case of ADALAB). Thus, although fluctuations of the input voltage will affect the rate of charging of the capacitor, the positive fluctuations will be cancelled by negative fluctuations. The resulting value is a true average, equivalent to the average of thousands of measurements using a SA converter with a sample and hold amplifier. As you can see, the DS converter "damps out" voltage fluctuations similar to the way that a strip-chart recorder damps out noise. Since many laboratory instruments are designed to be connected to strip chart recorders, their output circuitry doesn't attempt to filter out noise that will not normally show up because the recorder is heavily damped. Many scientists are surprised to learn how much noise occurs in the signals they are measuring. To observe this noise, attach an oscilloscope to the recorder output of your instrument.

Another feature of ADALAB's A/D converter that contributes to accuracy is that it automatically zeroes itself between each measurement. This compensates for internal offset errors generated by the buffer amplifier, integrator and comparator.

The ADALAB A/D converter also has true differential inputs. In other words, it measures the voltage difference between the high (+) input and the low (-) input. Normally, the low input is close to ground potential, but other electrical equipment in the vicinity can induce voltages in the wires connecting your instrument to ADALAB. Electrical noise is prevalent in most labs, and noise can affect the accuracy of the input voltage, especially when the wires connecting your instrument to the computer are quite long. Normally, induced voltages in the

signal lines will affect both the high and low inputs nearly equally; thus, ADALAB's differential inputs tend to counteract the effects of induced noise.

The A/D converter chip in ADALAB is actually a 13 bit A/D converter, but the QUICKI/O software throws away the least significant bit, in order to make the A/D converter readings directly comparable to the 12-bit D/A converter output values. Throwing away the least significant bit also increases the stability of the A/D values, because noise affects primarily the low bits of the value. When comparing the specifications of ADALAB with other A/D converters on the market, you should be aware that most other products available for the APPLE computer are only 8 bit A/D converters, they allow only positive voltage inputs, and they have only a single voltage range. An 8 bit converter returns values in the range 0 to 255, whereas ADALAB's 12 bit converter returns values from -2047 to 2047, using any of 4 different voltage scales. Thus, ADALAB is 16 times more accurate than 8 bit A/D converters and gives you the added advantages of measuring both positive and negative voltages. In addition, jumper-selectable voltage scales enable you to use ADALAB with a wide variety of instruments having different full scale voltage ranges.

7.2 A/D Converter Specifications

Integrated Circuit:	Intersil 7109 dual-slope A/D converter
Resolution:	12 bits plus sign bit and over-range bit
Full Scale Voltage:	±0.5V, ±1.0V, ±2.0V, or ±4.0V, jumper selectable
Maximum Conversion Time:	50 milliseconds
Minimum Conversion Rate:	20 samples per second
Maximum Input Voltage:	±12V without damage
Input Impedance:	minimum 8 megohms
Input Current:	maximum 0.5 microamperes
Temperature Coeffient:	100 ppm/degree C
Overall Accuracy:	Adjustable to better than 0.1% of full scale range

Differential Nonlinearity (maximum deviation from ideal step size): ±2 counts (0.05%)

Integral Nonlinearity (maximum deviation from ideal straight line): ±4 counts (0.10%)

True Differential Input (dual floating inputs)

Autozeroing compensates for internal offset voltages

Common Mode Rejection Ratio (common mode voltage ±1V, input voltage ØV, full scale voltage ±Ø.5V): 5Ø microvolts per volt

Software Interface: via Initiate Conversion command, Conversion Completed signal, and Interrupt Enable register. Data are read as two 8-bit bytes. The first byte includes the sign and over-range indicators and the most significant 4 bits. The second byte includes the least significant 8 bits of data.

7.3 A/D Converter Programming Considerations

This section discusses programming of the A/D converter at the assembler language level. For most applications, you will find it much easier to use the QUICKI/O approach described in its software manual. However, if you wish to use interrupts or polled sampling of the Conversion Completed signal, you should study this section. Here, you will also learn how to obtain the full 13-bit precision that is permitted by the Intersil 71Ø9 chip. All addresses in this section are given in hexadecimal notation.

The A/D converter is controlled by the "dedicated" 6522 chip on the ADALAB interface card. As we shall see later, this "dedicated" 6522 is also used for the real-time clock and the D/A converter. We will call the other 6522 chip the "user" 6522 because its functions are completely under the control of the user. Each of these two 6522 chips has 16 successive memory addresses which control its functions. The addresses for the dedicated 6522 start at address BASE1 = $CØØØ + N * $1ØØ, where N is the slot number of the ADALAB card. The addresses for the user 6522 begin at address BASE2 = $CØ3Ø + N * $1ØØ. For example, if the ADALAB card is in slot 2, BASE1 is $C2ØØ (decimal 49664) and BASE2 is $C23Ø (decimal 49712). Table III tells the function of each address in the set of 16 addresses associated with the dedicated 6522 chip.

Table III: Dedicated 6522 Addresses and Functions*

Address†	Function
BASE1+Ø	High byte of D/A digital value; triggers A/D converter on write
BASE1+1	Low byte of D/A digital value; triggers latching of D/A high byte on read or write
BASE1+2	Data Direction Register B; initialized to $8F
BASE1+3	Data Direction Register A; initialized to $FF
BASE1+4	Timer Ø Low Byte data ; initialized to $BE
BASE1+5	Timer Ø High Byte data; initialized to $C7
BASE1+6	Timer Ø Low Byte latched preset value
BASE1+7	Timer Ø High Byte latched preset value
BASE1+8	Timer 1 Low Byte*
BASE1+9	Timer 1 High Byte*
BASE1+A	Shift Register (unused)
BASE1+B	Auxiliary Control Register; initialized to $EØ
BASE1+C	Peripheral Control Register; initialized to $8A
BASE1+D	Interrupt Flag Register
BASE1+E	Interrupt Enable Register
BASE1+F	Low byte of D/A digital value; doesn't trigger high byte latch

*The initialized values referred to in this table are the result of BRUNning QUICKI/O.

†BASE1=$CØØØ+N*1ØØ, where N is the slot number.

*Timers Ø and 1 here correspond to timers 1 and 2, respectively, in the Versatile Interface Adapter data sheets.

To initialize the dedicated 6522 chip, use this program segment:

```
LDA #$8F            ;high byte data direction
STA BASE1+$Ø2       ;DDRB
LDA #$FF            ;low byte data direction
STA BASE1+$Ø3       ;DDRA
LDA #$BE            ;low byte of timer Ø
STA BASE1+$Ø4       ;T1L-L
LDA #$C7            ;high byte of timer Ø
STA BASE1+$Ø5       ;T1L-H & T1C-H
LDA #$EØ
STA BASE1+$ØB       ;Auxiliary Control Register
LDA #$8A
STA BASE1+$ØC       ;Peripheral Control Register
```

This initialization program sets up the D/A converter and timers, as well as the A/D converter. Note that the BASE1 address is calculated as described in the previous paragraph.

An A/D conversion sequence begins when a number is stored in the high byte of the digital to analog (D/A) converter output register. However, this has no effect on the D/A converter voltage, because the D/A converter is triggered by storing a number in the low byte of its output register. When the A/D converter finishes the conversion process, it sets a flag which can be read by your program. Thus, your program can readily determine whether the A/D value is ready. Also, it is possible to set up the A/D converter so that it will produce an interrupt when the conversion is completed.

To start an A/D conversion, you must write any value into address BASE1. As noted in Table III, this is the same as the address of the D/A high byte. However, the D/A high byte does not take effect until the D/A low byte is written into address BASE1+1. Thus, starting the A/D converter doesn't interfere with operation of the D/A converter.

To find out whether the A/D conversion is completed, you must test bit 4 at address BASE1+$ØD. This bit is low (Ø) during an A/D conversion and goes high (1) after completion. The following program will start the A/D and wait for the conversion done signal:

```
      STA BASE1          ;start A/D
WAIT  LDA BASE1+$ØD      ;read interrupt flags
      AND#$1Ø            ;check bit 4
      BEQ WAIT           ;loop if not done
DONE  (continue)
```

To enable the A/D converter to interrupt your program after conversion is done, you should set bit 4 of the IER interrupt

enable register. Storing $90 at address BASE1+$0E enables interrupts, while storing $10 disables interrupts. Of course, you must provide an interrupt handler that catches the interrupts and services the A/D converter. If it was the A/D converter that caused the interrupt, bit 4 of BASE1+$0D should be on ("1"). The following subroutine sets up for interrupts by the A/D converter and allows for interrupts by other devices:

```
SETUP    SEI              ;disable IRQ interrupts
         LDA #<IRQINT     ;low byte address of IRQ handler
         STA $03FE        ;IRQ jump vector for Apple
         LDA #>IRQINT     ;high byte address
         STA $03FF        ;high byte of IRQ jump vector
         CLI              ;reenable interrupts
         RTS

IRQINT   LDA BASE1+0D     ;is it an A/D interrupt?
         AND #$10
         BEQ OTHER        ;if bit 4=0, then not A/D
         AND BASE1+$0E    ;are A/D interrupts enabled?
         BNE A/DINT       ;yes, if bit 4 is 1
OTHER    (ANY CODE)       ;other interrupts handled here
A/DINT   LDA BASE1+$10    ;read A/D low byte
         STA ADVALUE      ;store in memory
         LDA BASE1+$20    ;A/D high byte
         STA ADVALUE+$01  ;store it
         STA BASE1        ;start A/D converter again
         LDA $45          ;recover A register value from
                            before interrupt
         RTI              ;return from interrupt
ADVALUE  0000             ;two bytes to store A/D value
```

The above routine will continuously run the A/D converter at maximum rate. It stores the most recent value in ADVALUE.

Your program should read the A/D converter low byte value at address BASE1+$10 and the high byte at BASE1+$20. The high byte value contains the four most significant bits of the answer (in bits 0-3), the overrange indicator (bit 4) and the sign bit (bit 5). Bits 6 and 7 of the high byte are unused and unspecified, but generally they read as 1's (on). Table IV shows the binary and hexadecimal codes that correspond to various input voltages.

Table IV: A/D CONVERTER CODES.

Binary	Hexadecimal	Voltage
1110 1111 1111 1111	$EFFF	Full Scale Positive
1110 0000 0000 0001	$E001	+1 Least Sign. Bit*
1110 0000 0000 0000	$E000	Zero Volts
1100 0000 0000 0001	$C001	-1 Least Sign. Bit
1100 1111 1111 1111	$CFFF	Full Scale Negative

*One least Significant Bit is the smallest detectable change in voltage at a given resolution.

The following subroutine converts the raw A/D value in ADVALUE into a ones complement number ranging from -8192 to 8191 (decimal) or $E000 to $1FFF (hex).

```
CONVERT  LDA ADVALUE+$01   ;high byte
         AND #$20          ;check for negative value
         BNE PLUS          ;bit 5 on means plus
MINUS    LDA ADVALUE       ;complement low byte if minus
         EOR #$FF
         STA ADVALUE
         LDA ADVALUE+$01   ;complement high byte
         EOR #$FF
         ORA #$E0          ;turn on all sign bits
         STA ADVALUE+$01
         RTS
PLUS     LDA ADVALUE+$01    ;load high byte
         AND #$1F           ;zero out high bits
         STA ADVALUE+$01
         RTS
```

7.4 A/D Connections and Interfacing

Table I lists the pin assignments on the analog I/O socket and cable. The analog I/O socket is in the upper right corner of the ADALAB card, as shown in Fig. 1. Normally, you should connect the ground wire of your instruments to pin 12 (A/D low) and connect the varying voltage signal to pin 10 (A/D high). Since the A/D converter can measure both positive and negative voltages with equal ease, you will not cause any damage if you reverse the wires.

WARNING: Do not connect any device which may exceed ±12 Volts because permanent damage may result.

8.Ø THE DIGITAL TO ANALOG CONVERTER

8.1 Theory of Operation

How does a D/A converter produce an analog output voltage when its given a particular digital input value? Fig. 6, shows a simple 4-bit D/A circuit consisting of a battery reference voltage (VR), a set of switches (one for each digital input bit), a set of carefully matched resistors (R,sR) and a summing operational amplifier (S). Each digital input bit controls one of the switches. Any switch that is connected to the reference voltage will cause current to flow through the resistor network (sometimes called a "ladder") to the summing junction of the op amp. The clever thing about this circuit is that the output voltage is the binary weighted sum of the digital input bits, multiplied by the reference voltage. In other words, the output analog voltage is proportional to the digital input value.

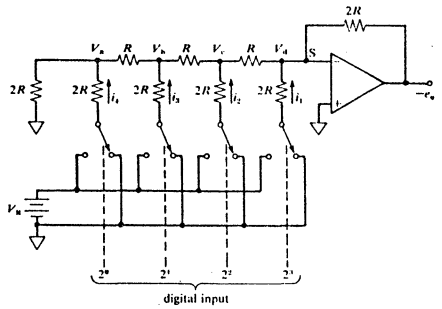


FIGURE6. Circuit Diagram for a 4-bit Digital to Analog Converter. For more details and a circuit analysis, refer to H. V. Malmstadt and C. G. Enke, Digital Electronics for Scientists (W. A. Benjamin, Inc., New York, 1969), pp. 333-335.

ADALAB's D/A converter has 12 bit precision, but it is set up so that you can output the data in two separate 8-bit data units. First, the most significant 4 bits of data are stored in memory location BASE1($CØØØ + SLOT * $1ØØ), but this does not change the D/A output voltage yet. When the remaining 8 bits of data are stored in the memory location BASE1+1, the most significant 4 bits of data are transferred (latched) into an output register. Thus, all 12 bits of the new data are presented simultaneously to the D/A converter. The response time of the D/A converter is almost instantaneous; the settling time for a full scale voltage change is only about 3 microseconds. Thus, no status bit or interrupt is needed to tell us when the conversion is completed. However, the speed of the D/A converter is limited by the software because, as we shall soon see, a simple program loop to output 12 bits of data takes about 2Ø microseconds. It is possible to operate the D/A converter at a faster rate if you don't change the high 4 bits and only update the low 8 bits.

Although the ADALAB's converter chip is configured for ±5V operation, its output voltage is divided down by a chain of resistors. This provides jumper-selectable ranges of ±4V, ±2V,

±1V and ±Ø.5V. The output from this voltage divider is buffered by an operational amplifier operating as a voltage follower with a gain of one. Thus, the D/A output voltage can deliver considerable current to drive external equipment. In other words, the D/A converter has a low output impedance.

8.2 D/A Converter Specifications

Integrated Circuit:	Analog Devices DAC8Ø
Resolution:	12 bits
Full Scale Voltage:	±Ø.5V, ±1.ØV, ±2.ØV or ±4.ØV, jumper selectable
Maximum Conversion Time:	2Ø microseconds
Minimum Conversion Rate:	up to 5Ø,ØØØ conversions per second, limited only by software speed.
Output Current:	sources or sinks 1ØmA
Nonlinearity:	±1 least significant bit
Accuracy:	Adjustable to better than Ø.2% of full scale range
Monotonic:	over entire Ø to 7Ø degree C range
Temperature Coefficient:	1ØØ ppm/degree C
Software Interface:	via output of two data bytes; the most significant 4 bits are stored until the least significant 8 bits are output and then the 12 bits of data are presented simultaneously to the D/A converter.

8.3 D/A Converter Programming Considerations

The easiest way to use the D/A converter is to program it with QUICKI/O, as described in its software manual. However, this section explains how to program the D/A converter in assembler language, which enables the D/A converter to run at rates of up to 5Ø,ØØØ conversions per second.

The D/A converter is controlled by the dedicated 6522 chip. (see Table III). You will recall that BASE1=$CØØØ+SLOT*$1ØØ. In order to initialize the D/A, you must store $8F in location BASE1+$Ø2, store $FF in location BASE1+$Ø3 and store $8A in location BASE1+$ØC. This sets up the 6522 chip for output of 12 bits, with triggering of the high byte latch as soon as the low byte is stored. A program to initialize the dedicated 6522 chip

was presented earlier (see A/D Converter Programming Considerations).

To output a voltage on the D/A converter, first store the most significant 4 bits in location BASE1 and then store the least significant 8 bits in location BASE1+$01. That's all there is to it. It doesn't matter what you place in the high order 4 bits of location BASE1 because only the low 4 bits are actually used as the most significant 4 bits of the output voltage.

The D/A converter uses a complementary offset binary format. Table V shows the digital codes for various output voltages. As you can see, QUICKI/O has to perform some mathematical manipulations in order to make digital codes -2047 to 2047 correspond to minus full scale through plus full scale voltage. The easiest way to transform a signed 16 bit binary value from -2047 ($F801) to 2047 ($07FF) into the appropriate form for the D/A converter is as follows:

```
LDA DALOW       ;D/A data low byte value
EOR #$FF        ;two's complement
CLC
ADC #$01
PHA             ;save on stack
LDA DAHIGH      ;D/A data high byte value
EOR #$F7        ;two's complement; reverse bit 3
ADC #$00        ;add carry from low byte
STA BASE1       ;slot dependent address
PLA             ;unstack low byte
STA BASE1+01    ;low byte triggers high byte latch
```

Bear in mind that each time you output a value to BASE1, it automatically triggers the A/D converter. If the A/D converter is in the middle of a conversion, an extra trigger will have no effect on it.

Table V: D/A CONVERTER CODES

Binary	Hexadecimal	Voltage
0000 0000 0000 0000	$0000	Full Scale Positive
0000 0111 1111 1110	$07FE	+1 Least Sign. Bit*
0000 0111 1111 1111	$07FF	Zero Volts
0000 1000 0000 0000	$0800	-1 Least Sign. Bit
0000 1111 1111 1111	$0FFF	Full Scale Negative

*One Least Significant Bit is the smallest detectable change in voltage at a given resolution.

If you are interested in running the D/A converter at a very fast rate, here are some programming tips. In all of these cases, we will use the X (or Y) register to index an array of data values. Faster rates are attainable if the data are stored in page Ø, but usually this is not feasible. First, let's convert samples from a table of 256 8-bit values, holding the high byte constant (12 machine cycles per loop or 83.3 KHz rate for 1MHz clock).

```
            LDA DAHIGH      ;D/A high byte
            STA BASE1
            LDX #$ØØ        ;point to first data byte
DAOUT       LDA TABLE,X     ;table of D/A values
            STA BASE1+$Ø1
            INX             ;increment pointer
            BNE DAOUT       ;loop for 256 values
            BEQ DAOUT       ;include this for continuous output
```

To refresh both the high byte and the low byte at maximum rate, store the high byte and low byte data as two separate tables and use this program (2Ø clock periods per loop or 5Ø KHz output rate):

```
            LDX #$ØØ        ;point to first data byte
DAOUT       LDA DATAHI,X    ;high byte data table
            STA BASE1
            LDA DATALO,X    ;low byte data table
            STA BASE1+$Ø1
            INX             ;increment table pointer
            BNE DAOUT
            BEQ DAOUT       ;include this for continuous output
```

For each of the previous two programs, the BEQ DAOUT instruction at the end makes a continuous waveform output. To change the frequency of the output waveform, we can advance X by a SKIP interval different from one. Replace the INX instruction above with the following code (this adds 4 to 6 clock periods per loop, depending on the addressing mode of the ADC command):

```
     TXA            ;current position in data table
     ADC SKIP       ;change SKIP for different frequency
     TAX            ;ignore overflow
```

8.4 D/A Connections and Interfacing

As indicated in Table I, the D/A output voltage (high) is on pin 13 of the analog I/O socket, which is in the upper right corner of the ADALAB interface card (see Fig. 1). The D/A reference voltage (low) is the analog ground on pin 11. Note that the D/A output voltage may be either positive or negative, relative to analog ground.

WARNING: When the computer is first turned on, the D/A output voltage is unpredictable, until it is initialized by some program. If you are connecting the D/A converter to an instrument that cannot tolerate negative voltages, be sure to disconnect the cable before turning the computer on.

9.Ø DIGITAL (PARALLEL) INPUT AND OUTPUT

9.1 Theory of Operation

You will recall that the QUICKI/O software distinguishes between digital (bitwise) I/O and parallel (bytewise) I/O. At the hardware level, there is no difference between digital and parallel I/O. The ADALAB parallel I/O is implemented as part of the user 6522 chip, which is totally configurable for your needs. Since there are no parallel I/O buffers on the ADALAB card, each of the 16 parallel I/O bits may be selected for either input or output. Table VI lists the addresses that control each function of the user 6522 chip. In the data direction registers for port A and port B, each bit that is on (1) is selected for output, while each bit that is off (Ø) is selected for input. Normally, port B is used for output because it has greater current drive capability than port A. When timer 2 is used as a frequency generator, bit 7 of port B must be selected for output. When timer 3 is used as a pulse counter, bit 6 of port B must be selected for input.

Four handshaking lines are available to facilitate and synchronize communications between devices. The CA1 and CB1 lines may be set up to recognize either positive or negative input transitions and each can set a bit in the interrupt flag register when such a transition occurs. If the corresponding bit in the interrupt enable register is also set, a transition on CA1 or CB1 will generate an interrupt. Transitions on CA1 or CB1 can latch the input or output data if the Auxiliary Control Register is set up appropriately.

Handshaking lines CA2 and CB2 have the same input capabilities as lines CA1 and CB1, but they also can output signals in four different modes. In mode 1, CA2 and CB2 will change state when data is read from or written into port A or port B, respectively. Their state reverses again when an active transition occurs on CA1 or CB1, respectively. This is exactly what we need for automatic handshaking. In mode 2, a short pulse (one microsecond) is sent out on CA2 or CB2 when data is read from or written to port A or port B. In mode 3, CA2 and CB2 are held low, whereas these outputs are held high in mode 4.

Table VI: User 6522 Addresses and Functions*

Address**	Function
BASE2+Ø	Port B (Output) Data Register
BASE2+1	Port A (Input) Data Register
BASE2+2	Port B Data Direction Register; initialized to $FF
BASE2+3	Port A Data Direction Register; initialized to $ØØ
BASE2+4	Timer 2 Low Byte data***
BASE2+5	Timer 2 High Byte data***
BASE2+6	Timer 2 Low Byte latched preset value***
BASE2+7	Timer 2 High Byte latched preset value***
BASE2+8	Timer 3 Low Byte latched preset value ***
BASE2+9	Timer 3 High Byte
BASE2+A	Shift register
BASE2+B	Auxiliary Control Register; initialized to $ØØ
BASE2+C	Peripheral Control Register; initialized to $88
BASE2+D	Interrupt Flag Register
BASE2+E	Interrupt Enable Register
BASE2+F	Alternate Port A Data Register--no effect on handshake

*The initialized values referred to in this table are the result of BRUNning QUICKI/O.

**BASE2=$CØ3Ø+N*$1ØØ, where N is the slot number.

***Timers 2 and 3 here correspond to timers 1 and 2, respectively, in the Versatile Interface Adapter data sheets.

9.2 Digital I/O Specifications*

Integrated Circuit: MOS Technology 6522 Versatile Interface Adapter

16 bidirectional lines (usually used as 8 bits in and 8 bits out)

Latching capability on input or output

Four handshaking signals accommodate positive or negative logic

Interrupt register and interrupt enable registers are available for each handshake signal.

Input Characteristics:

High Voltage:	2.4V to 5.0V
Current:	-100 to -250 microamperes
Low Voltage:	-0.3V to +0.4V
Current:	-1.0 to -1.6 milliamperes
Leakage Current:	±1.0 to ±2.5 microamperes
Off-state Current:	±2.0 to ±10 microamperes
Capacitance:	10 pF

Output Characteristics:

High Voltage:	2.4V minimum
Current:	-0.1 to -1.0 milliamperes (PA0-PA7, CA2)
	-3.0 to -5.0 milliamperes (PB0-PB7, CB1, CB2)
Low Voltage:	0.4V maximum
Current:	1.6 milliamperes
Leakage Current:	1.0-10 microamperes
Capacitance:	10 pF

* See also the Versatile Interface Adapter Data Sheet in the back of this manual.

9.3 Digital I/O Programming Considerations

If QUICKI/O fails to meet your requirements, this part of the manual will tell you how to program the digital I/O using assembler language. The assembler language approach is necessary if you want to use some configuration other than 8 bits input on Port A and 8 bits output on Port B.

The direction of data flow is controlled by writing a one (1) for each output bit or a zero (Ø) for each input bit into address BASE2+2 (port B) or address BASE2+3 (port A). BASE2 is defined as $CØ3Ø + SLOT * $1ØØ. For example, if the ADALAB card is in slot 2, the port B data direction register is $C232 (decimal 49714) and the port A data direction register is $C233 (decimal 49715). In this case, you could set up port B as 8 output bits by storing $FF at location $C232 (or POKE 49714,255 in BASIC); port A could be set up for 8 input bits by storing Ø at location $C233 (or POKE 49715,Ø). When the computer is first turned on and after you press RESET, the data direction registers are initialized to Ø (all inputs) for both ports A and B. This is intended as a safety device, because two input signals can be connected together safely, whereas two output signals should not be connected together. Therefore, it is safest to assume initially that all data lines will be used as input signals.

After setting up the data direction registers, you may read or write data to address BASE2 (port B) or BASE2+1 (port A). In general, you can read or write to a parallel port, regardless of whether it is selected for input or output. If you read a bit selected for output, you will obtain the last value stored for that bit. If you write to a bit selected for input, it will have no effect. As indicated in Table VII, you may enable latching of data for Ports A and/or B by setting the bits Ø and 1 of the auxiliary control register. When latching is selected, a handshaking transition on CA1 or CB1 will cause the current data to be latched (stored) until it is changed again. This feature allows ADALAB to capture momentary data on cue from some external device. If latching is not enabled, the values read from a port will reflect the current input data.

The Peripheral Control Register (see Table VIII) is very important for selecting the type of handshaking to be used for digital I/O. The low order 4 bits control the handshaking for Port A, while the high order 4 bits pertain to Port B. CA1 and CB1 are always input lines, capable of detecting either positive or negative transitions produced by your external equipment. CA2 and CB2 are more versatile; they can be used as either inputs or outputs. In the input modes, you have a choice of either positive or negative transitions, as well as two different ways of clearing the interrupt flag. In the output modes, you have a choice of constant voltage level outputs (handshake or manual modes) or pulse outputs. The handshake mode and pulse mode are particularly convenient to use, because they coordinate the operation of the CA1+CA2 or CB1+CB2 handshaking pairs.

Table VII: User 6522 Auxiliary Control Register (BASE2+$ØB).

Bit	State	Result
Ø	Ø	Port A; latch is disabled
Ø	1	Port A latches data when CA1 interrupt flag is set
1	Ø	Port B latch is disabled
1	1	Port B latches data when CB1 interrupt flag is set
4,3,2	Ø,Ø,Ø	Shift register is disabled
4,3,2	Ø,Ø,1	Shift in under control of timer 2
4,3,2	Ø,1,Ø	Shift in under control of processor clock
4,3,2	Ø,1,1	Shift in under control of external clock
4,3,2	1,Ø,Ø	Free-running output at rate of timer 2
4,3,2	1,Ø,1	Shift out under control of timer 2
4,3,2	1,1,Ø	Shift out under control of processor clock
4,3,2	1,1,1	Shift out under control of external clock
5	Ø	Timer 3* acts as a one-shot interval timer
5	1	Timer 3 counts pulses on PB6
7,6	Ø,Ø	Timer 2* generates a single interrupt after countdown to Ø
7,6	Ø,1	Timer 2 generates continuous interrupts at free-running rate
7,6	1,Ø	Timer 2 single interrupt mode; also pulses PB7
7,6	1,1	Timer 2 free running mode; also outputs a square wave on PB7.

*Timers 2 and 3 here correspond to timers 1 and 2 in the Versatile Interface Adapter data sheets.

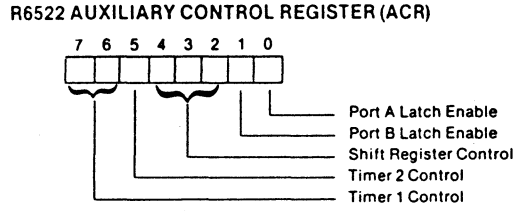
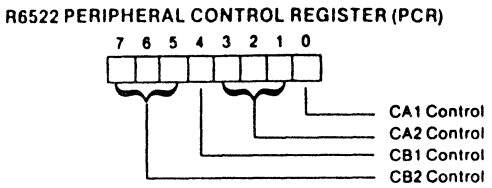


Table VIII: User 6522 Peripheral Control Register(BASE2+$ØC)

Bit	State	Result
Ø	Ø	CA1 interrupt flag set by high to low transition on CA1
Ø	1	CA1 interrupt flag set by low to high transition on CA1
3,2,1	Ø,Ø,Ø	CA2 interrupt flag set by high to low transition on CA2 input; clear interrupt by read or write Port A.
3,2,1	Ø,Ø,1	Like Ø,Ø,Ø mode, but clear interrupt by writing logic 1 in interrupt register bit Ø
3,2,1	Ø,1,Ø	CA2 interrupt flag set by low to high transition on CA2 input; clear by reading or writing Port A
3,2,1	Ø,1,1	Like Ø,1,Ø mode, but clear by writing logic 1 in interrupt register bit Ø
3,2,1	1,Ø,Ø	Handshake mode. Set CA2 output low for one cycle following a read or write to Port A
3,2,1	1,1,Ø	Manual output mode. Hold CA2 output low
3,2,1	1,1,1	Manual mode; hold CA2 output high
4	Ø	CB1 interrupt flag set by high to low transition on CB1
4	1	CB1 interrupt flag set by low to high transition on CB1
7,6,5	Ø,Ø,Ø	CB2 interrupt flag set by high to low transition on CB2 input; clear by read or write Port B
7,6,5	Ø,Ø,1	Like Ø,Ø,Ø mode, but clear by writing logic 1 in interrupt register bit 3
7,6,5	Ø,1,Ø	CB2 interrupt flag set by low to high transition on CB2; clear by read or write Port B
7,6,5	Ø,1,1	Like Ø,1,Ø mode, but clear by writing logic 1 into interrupt register bit 3
7,6,5	1,Ø,Ø	Handshake mode. Set CB2 output low on write to Port B; reset CB2 high with active transition on CB1
7,6,5	1,Ø,1	Pulse output mode. Set CB2 output low for one cycle following a write to Port B
7,6,5	1,1,Ø	Manual output mode; hold CB2 output low
7,6,5	1,1,1	Manual mode; hold CB2 output high



For example, let us set up Port A as an input and Port B as an output, with both ports using the handshake mode of operation. We will assume that the Port B output data lines are connected directly to the Port A input data lines and that CA1 is connected to CB2, while CB1 is connected to CA2. In other words, this is exactly the way the self-test adapter is connected; it is also the way most instruments operate when using digital I/O.

```
LDA#$00
STA BASE2+$03      ;Port A data direction
LDA #$FF
STA BASE2+$02      ;Port B data direction
LDA BASE2+$0B      ;Read auxiliary control register
AND #$FC           ;Preserve bits 2 to 7
ORA #$01           ;Enable port A latch
STA BASE2+$0B      ;Save auxiliary control register
LDA #$88           ;Handshake on ports A and B
STA BASE2+$0C      ;Peripheral Control Register
```

Now, everything is initialized. Let's make a dry run to see how the handshaking works. First, we write data to Port B. This makes CB2 go low and, since CB2 is connected to CA1, CA1 also goes low. Because CA1 is low, the CA1 interrupt flag is set and CA2 goes high. This enables the input program to determine that data is ready. Next, the input program reads the data, which causes CA2 to go low again. Since CA2 is connected to CB1, this sets the CB1 interrupt flag and CB2 goes high. This tells the output program that the last data was received and thus, it is time to send new data. The following program could be used to output a group of 10 data points stored at address DATAOUT:

```
            LDX #$00          ;set up counter
OUTWAIT     LDA BASE2+$0D     ;read interrupt flag
                                register
            AND #$10          ;isolate bit 4 (CB1)
            BEQ OUTWAIT       ;off means not ready
            LDA DATAOUT,X     ;get data from memory
            STA BASE2         ;output to Port B
            INX               ;increment counter
            CPX #$0A          ;10 values done?
            BMI OUTWAIT       ;loop if not
```

To input 10 values and store them at address DATAIN, this program could be used:

```
            LDX #$00          ;set up counter
INWAIT      LDA BASE2+$0D     ;read interrupt flag
                                register
            AND #$02          ;isolate bit 1 (CA1)
            BEQ INWAIT        ;off means not ready
            LDA BASE2+$01     ;read input value
            STA DATAIN,X      ;store in memory
            INX               ;increment counter
            CPX #$0A          ;10 values done?
            BMI INWAIT        ;loop if not
```

ADALAB's hardware is capable of generating interrupts when the handshaking lines indicate that it is time to input or output digital data. Table IX lists the bit positions for the interrupt flags associated with various functions of the user 6522. One way to use the interrupt flags is the polling method, as used in the examples above. The other method is the true interrupt approach, whereby the interrupting condition causes the computer to stop what it is doing (the "main" program) and instead, the computer runs a subroutine called an "interrupt handler." The interrupt handler inputs or outputs some data and then returns to the main program. In order to enable interrupts, the appropriate bit in the interrupt enable register (IER) must be turned on. For example, to enable interrupts when CA1 is triggered:

```
LDA #$82          ;bit 7=1 means enable, bit 2 means CA1
STA BASE2+$0E     ;interrupt enable register
```

To disable interrupts, store $02 in the same place. The following simple interrupt routine inputs a byte of data when a CA1 interrupt occurs:

```
SETUP     LDA #$00            ;initialize pointer
          STA POINTER
          SEI                 ;disable interrupts during
                                setup
          LDA #<INTCA1        ;low byte service address
          STA $03FE           ;IRQ jump vector low byte
          LDA #>INTCA1        ;high byte service address
          STA $03FF           ;high byte of IRQ jump vector
          LDA #$82            ;allow interrupts on CA1
          STA BASE2+$0E       interrupt enable register
          CLI                 ;enable interrupts
          RTS

INTCA1    TXA                 ;save X register
          PHA                 ;on stack
          LDX POINTER         ;recover pointer in X
          LDA BASE2+$01       ;read input data
          STA DATAIN,X        ;store in memory
          INX                 ;increment pointer
          STX POINTER
          PLA                 ;unstack X
          TAX
          LDA $45             ;recover A at time of
                                interrupt
          RTI                 ;return from interrupt
```

Table IX: User 6522 Interrupt Control Flag & Enable Registers.

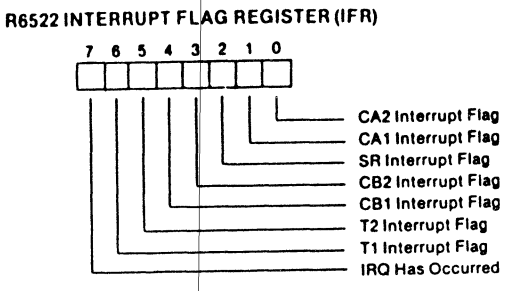
INTERRUPT FLAG REGISTER (BASE2+$ØD)

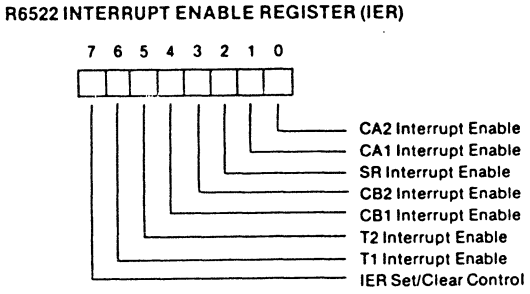
Bit	Set By	Cleared By
Ø	Active transition on CA2	Read or write Port A
1	Active transition on CA1	Read or write Port A
2	Completion of 8 shifts	Read or write shift register
3	Active transition on CB2	Read or write Port B
4	Active transition on CB1	Read or write Port B
5	Time-out of Timer 3*	Read low byte or write high byte
6	Time-out of Timer 2*	Read low byte or write high byte
7	Any of bits Ø-6 set and enabled	Clear bit in flag register or in enable register

INTERRUPT ENABLE REGISTER (BASE2+$ØE)

Bit	Enable	Action
Ø	CA2 Interrupt	Any bit=Ø disables interrupt
1	CA1 Interrupt	Any bit=1 enables interrupt
2	Shift Register Interrupt	
3	CB2 Interrupt	
4	CB1 Interrupt	
5	Timer 3* Interrupt	
6	Timer 2* Innterrupt	
7	Writing logic 1 in any bit while bit 7=Ø clears that enable bit Writing logic 1 in any bit while bit 7=1 sets that enable bit	

*Timers 2 and 3 here correspond to timers 1 and 2 in the VIA description sheets.





9.4 Digital I/O Connections and Interfacing

Port A and Port B have separate sockets on the ADALAB interface card, as indicated in Fig. 1. The pin assignments are detailed in Table I. Pin 1 on the board is on the top left side of each socket, and the cable should be plugged into the socket with the white mark closest to pin 1 (also marked white).

The current capability of the digital I/O ports is quite limited; they will source or sink only one TTL load. Also, input and output voltages must always be within the range of 0 to 5 volts. If your input or output requirements are different from these conditions, you will need to purchase or build a signal conditioning adapter to bring your signals within these specifications. ADA-BUFF is a digital buffer sold by Interactive Microware, Inc. that can be configured for 8 bits of input or 8 bits of output; all signals, including the handshake lines, are buffered to drive 10 low-power TTL loads or 5 standard TTL loads. IMI also offers a high-power isolated AC/DC relay adapter that permits input or output of DC signals up to 3 Amps at 60 volts or AC signals up to 3 Amps at 120 or 240 Volts. For even larger AC currents of up to 20 Amps at 120 or 240 Volts, our REMTROL interface can be used.

10.0 THE REAL-TIME CLOCK AND COUNTER/TIMER

10.1 Theory of Operation

The ADALAB interface card includes four 16 bit timers; two on each of the 6522 Versatile Interface Adapter (VIA) chips. The two timers on the dedicated 6522 chip (Timers 0 and 1) are ganged together to form a 32 bit timer that is used as the real time clock in QUICKI/O. If you have more than one ADALAB card, you can use the 32 bit timers on the second and subsequent boards as general-purpose timers. The two 16 bit timers located on the user 6522 chip (Timers 2 and 3) are completely available for use as a pulse generator, pulse counter, shift register or square wave generator. Please note that in the following descriptions, the timers are numbered as in QUICKI/O. That is, Timers 0 and 2 correspond to timer 1 in the VIA description sheets, whereas Timers 1 and 3 here correspond to timer 2 in the VIA description. Also, if you have more than one ADALAB card, Timers 4, 8 and 12 are identical to Timer 0, while Timers 5, 9 and 13 are the same as Timer 1.

ADALAB's real-time clock is implemented partly in hardware (the dedicated 6522 timers) and partly in software (the timer subroutines in QUICKI/O). Let us briefly consider how this works. During initialization of QUICKI/O, the preset count for timer 0 is set to $C7BE (51,134 decimal), and the auxiliary control register mode is set to $E0 (See Table VII). This mode makes timer 0 operate in the free-running mode, generating continuous interrupts at 50 millisecond intervals and inverting the state of bit 7 of Port B (PB7) at the same rate. The real-time clock is very accurate because timer 0 is driven by the quartz crystal oscillator (1.023 MHZ) in the APPLE computer. Timer 1 is set up to count pulses on bit 6 of Port B (PB6). Because PB7 is connected to PB6, the value in Timer 1 counts down at the rate of 10 counts per second. Since timer 1 is a 16 bit counter, it can count down from 32767 to -32768, a total of 65,536 counts. At 10 counts per second, a maximum time interval of 6553.6 seconds or 1.82 hours is possible before Timer 1 finishes counting and wraps around from -32768 back to 32767.

In QUICKI/O, the 50 millisecond interrupts from Timer 0 are used to update the hours, minutes, seconds and milliseconds counts. If the seconds count changes, the display at the upper right of the screen is also updated. The program checks to see whether the "software time" (as displayed on the screen) is the same as the "hardware time" (as measured by the count in Timer 1). If not, the software time is updated at a very fast pace until it catches up with the hardware time. The software time falls behind when interrupts are disabled, but the hardware time runs constantly, independent of interrupts. Thus, the software is able to detect when interrupts have been disabled and the software time can be corrected when interrupts are reenabled.

Bear in mind that interrupts are disabled by pressing RESET or by executing a SEI instruction. In addition, interrupts are temporarily disabled whenever the disk is reading or writing information.

The two 16 bit timers on the user 6522 are completely available for you to configure as you wish, as summarized in Table VII. You may use Timer 2 in the one shot mode or in the free-running mode. In either case, an interrupt flag is set when Timer 2 counts down to Ø, and an interrupt will occur if the interrupt enable register is set up properly. By appropriate initialization of the auxiliary control register, Timer 2 can be made to output a square wave on bit 7 of Port B. This method for producing a square wave signal is particularly convenient because after initialization, it runs automatically, without any further involvement of the microprocessor. The square wave frequency can range from about 8HZ to 167KHZ, with microsecond resolution between these extremes.

Timer 3 may be used as a very precise interval timer. After an initial count is written to Timer 3, the count is decremented at the processor clock rate (1.Ø23MHZ) and when the count reaches Ø, the interrupt flag is set. If interrupts are enabled, an IRQ interrupt will occur (see Table IX). The count continues to decrement, so it is possible to determine how long it has been since the interrupt flag was set. In the second mode of Timer 3, it counts pulses on bit 6 of Port B. You will recall that we used this feature to create a 32 bit timer on the dedicated 6522 by connecting bits 6 and 7 of Port B together. However, you can use this timer mode to count TTL-level pulses coming from any external source. When the count reaches zero, the interrupt flag is set and, if the interrupt enable bit is also set, an IRQ interrupt will result. Timers 2 and 3 can also be used together as a frequency counter; Timer 2 could be used to count the time while Timer 3 is used to count a specified number of pulses coming from some external source. To use the frequency counter, read the time from Timer 2 and store a preset count in Timer 3. When an interrupt occurs for Timer 3, read the time from Timer 2 again and calculate the frequency from the ratio of the preset count (Timer 3) to the time elapsed (Timer 2).

The user 6522 chip also has an 8-bit shift register that can input or output serial information. The various modes of this shift register are listed in Table VII. The serial data is input or output on handshake line CB2, whereas the shift timing pulses are input or output on handshake line CBl. There are three possible sources of timing pulses: Timer 2, the processor clock (1.Ø23MHZ) or an external clock supplied by your instrument. The shifting operation is initiated by reading from or writing to the shift register. When shifting out, bit 7 is the first bit output and each successive bit is recirculated back into bit Ø. When shifting in, bits initially enter bit Ø and they are shifted towards bit 7. After 8 bits of data have be shifted in or out of the shift register, the interrupt flag is set and an IRQ interrupt will occur if the corresponding interrupt enable bit is

set (see Table IX). In all shift register modes except the free-running mode, the shifting operations stop after 8 bits. In the free-running output mode, the data are continuously shifted out at the Timer 2 rate. The latter feature could be used to generate repeating waveforms that are more complicated than a simple square wave.

10.2 Real Time Clock and Counter/Timers Specifications

Integrated Circuit:	Two MOS Technology 6522 Versatile Interface Adapters
Timers 0 and 2:	16 bit countdown timers can be used as: * one-shot interval timers with optional pulse output on PB7 * continuous frequency generator with optional square wave output on PB7
Timers 1 and 3:	16 bit countdown timers can be used as: * one-shot interval timers * frequency counter that counts a predetermined number of pulses on PB6 * shift register rate generator
Shift register:	Inputs or outputs 8-bit serial data with timing pulses supplied by Timers 1 or 3, the 1.023MHZ processor clock or an external clock.
Interrupt Control:	Interrupt flag and interrupt enable on all functions.
Signal Characteristics:	TTL compatible signals (one TTL load or drive)

Dedicated 6522 has bits 6 and 7 of Port B connected to allow operation of Timers 0 and 1 together as a 32 bit timer for use as a real-time clock.

User 6522 has all functions of Timers 2 and 3 available for user configuration.

10.3 Real Time Clock and Counter/Timers Programming Considerations

The QUICKI/O software provides the most convenient way to use the real time clock and counter/timers. However, there are some applications that require non-standard use of these features. For example, you might want a real-time clock that runs faster or slower than 20 ticks per second or you might want to use the shift register as a serial I/O port. This section of the manual provides detailed information about assembly language programming of the various clocks and timers included with ADALAB.

Tables III and VI list the addresses used for access to and control of the dedicated 6522 and the user 6522. (Recall that BASE1=$C000 + SLOT * $100 and that BASE2=BASE1+$30.) Table VII explains the function of each bit in the auxiliary control registers and Table IX contains information about the interrupt register and the interrupt enable register. Additional detailed information about the 6522 chip will be found in the VIA description sheet. Now, if all of this seems a bit complicated, you have come to the right conclusion. The 6522 chip has many features and capabilities, but we have to put up with this complexity in order to gain the versatility of its functions.

Timers 0 and 1 can be used as a 32 bit timer because bits 6 and 7 of Port B are connected together. The following code will initialize timer 0 to interrupt 20 times per second and timer 1 will count down at 10 counts per second:

```
SETUP  LDA #$8F        ;Bits 0-3 and 7 are outputs, 4-6 are
                         inputs
       STA BASE1+$02   ;Port B Data Direction Register
       LDA #$BE        ;50 milliseconds low byte
       STA BASE1+$04   ;Timer 0 low byte latch
       LDA #$C7        ;50 milliseconds high byte
       STA BASE1+$05   ;Timer 0 high byte latch
       LDA #$E0        ;Timer 0 free-running, Timer 1 counts
                         pulses
       STA BASE1+$0B   ;Auxiliary Control Register
       SEI             ;Disable interrupts
       LDA #<TIMINT    ;Low address of timer interrupt routine
       STA $03FE       ;IRQ jump vector low byte
       LDA #>TIMINT    ;High address of interrupt routine
       STA $03FF       ;IRQ jump vector high byte
       LDA #$C0        ;enable Timer 0 interrupts
       STA BASE1+$0E   ;Interrupt enable register
       CLI             ;reenable interrupts
       RTS
```

The following routine will intercept the Timer 0 interrupts and count time in hours (HOURS), minutes (MINS), seconds (SECS) and 20 millisecond units (UNITS):

```
TIMINT  TYA             ;stack Y register
        PHA
        LDA BASE1+$0D   ;Interrupt Flag Register
        AND #$40        ;Timer 0 bit on?
        BEQ OTHER       ;Not a timer interrupt
        LDA BASE1+$04   ;Clear interrupt flag
        INC UNITS       ;50 millisecond counter
        LDA UNITS
        CMP #$14        ;20 counts completed?
        BMI TIMOK       ;update seconds after 20 ticks
        LDA #$00        ;reset units to 0
        STA UNITS
        INC SECS        ;increment seconds
        LDY #$3C        ;compare to 60
```

```
          CPY SECS
          BNE TIMOK
          STA SECS        ;reset seconds to Ø
          INC MINS        ;increment minutes
          CPY MINS        ;compare to 6Ø
          BNE TIMOK
          STA MINS        ;reset minutes to Ø
          INC HOURS       ;increment hours
TIMOK     PLA             ;unstack Y register
          TAY
          LDA $45         ;recover A register
          RTI             ;return from interrupt
```

The crystal frequency of the Apple computer varies with temperature; thus, it may be necessary to fine tune the clock rate by changing the countdown value in BASE1+4 (low byte) and BASE1+5 (high byte). Try changing the low byte first, and only change the high byte if a large correction is needed. Bear in mind that it is possible to change the countdown value drastically in order to obtain a faster timebase. For example, if you set timer Ø to $2ØØ (BASE1+4=2 and BASE1+5=Ø), then timer 1 will decrement about once each millisecond. Note that timer Ø must count down to Ø twice for each time that timer 1 is decremented. The reason for this is that timer Ø simply reverses the state of PB7 each time it counts down to zero, while timer 1 requires a complete pulse (from low to high to low again) before it decrements by one.

As an example of using the shift register, we will set up the shift register on the user 6522 to continuously output a bit pattern at a rate governed by Timer 3:

```
LDA #$1Ø        ;free-running shift register mode
STA BASE2+$ØB   ;Auxiliary Control Register
LDA RATELO      ;low byte of Timer 3 rate
STA BASE2+$Ø8
LDA RATEHI      ;high byte of Timer 3 rate
STA BASE2+$Ø9
LDA PATTERN     ;shift register bit pattern
STA BASE2+$ØA   ;start shifting
```

This method could be used to produce musical tones. The pitch of the tone could be controlled by the rate of Timer 3, while the timbre (tone color) could be controlled by the pattern in the shift register. The output on handshake line CB2 could be attenuated to a Ø-1 volt range and played on a speaker coupled through an audio amplifier.

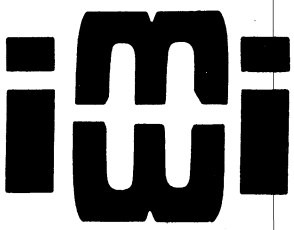
1Ø.4 Real-Time Clock and Counter/Timers

Connections and Interfacing

All of the signals pertaining to Timers 2 and 3 on the user 6522 are available on the two 16 pin DIP sockets used for Digital

I/O (see Table I). Timers Ø and 1 on the dedicated 6522 chip are not as versatile, because their handshake signals are not externally available. Moreover, bits 6 and 7 of port B on the dedicated 6522 chip are internally connected, and handshake lines CA2 and CB2 are committed for other purposes.

Bear in mind that the current and voltage capabilities of the 6522 chip are quite limited. No input should be outside the range of Ø to 5 volts and output current is limited to one TTL load.



Interactive Microware, Inc.
P.O. Box 771
State College, Pa 16801
(814) 238-8294

Using the ADALAB A/D Converter with Curve Fitter, VIDICHART and Other BASIC Programs

The machine language subroutine listed on the next page allows you to use the ADALAB(tm) A/D converter from a BASIC program. To use this program, enter the ROM Monitor (type CALL -151 in BASIC) and type 320: AD 03 60...etc., copying the hexadecimal numbers from the bottom of the column listing, with a space between each number. Then, return to BASIC (press RESET) and type BSAVE ADOBJ,A$320,L$50. Your BASIC program must begin with LOMEM:24576: D%=0 to ensure that the value of D% is stored at the right place. To read the A/D value, set D% to the slot number of your ADALAB card and CALL 800. On return, D% contains the current A/D value, sampled after the appropriate time delay. The most negative voltage reading is -4095 and the most positive voltage is 4095. Over-range is indicated by -8192 or 8192.

To modify Curve Fitter so that the Control Y command reads the ADALAB A/D converter, type LOAD CURFIT and make the following changes exactly. Then, type SAVE CURFITAD to save this new version.

```
10 HIMEM:38399: LOMEM:24576: D%=0: DIM IN$(50), D(1000), DD(5,2): MX=1000
2900 IF AD=0 THEN SLOT=2: AD=800: PRINT: PRINT CD$"BLOAD ADOBJ,A"AD
2910 D%=SLOT: CALL AD: V0=D%: RETURN
```

To modify VIDICHART(tm) so that the ADC command will access the ADALAB A/D converter, type LOAD VIDICHART and then type the following changes exactly. Then, type SAVE VIDICHARTAD to save this new version. When using the ADC command, the slot number of your ADALAB card must be typed for the channel number (#CHAN). The change in line 948 causes VIDICHART to return to the primary menu when you type Control X during a secondary menu entry.

```
1 HIMEM:35327: LOMEM: 24576: D%=0000: IF PEEK(35411)+PEEK(37232)
   <>128 THEN PRINT CHR$(4)"BLOAD VISIOBJ"
94 CD$ = CHR$(4)
948 IF N<1 OR N>N1-N0+1 THEN PRINT CHR$(7);: GOTO 932
1005 IF OP<1 OR OP>7 GOTO 72
1010 IF AD=0 THEN AD=800: PRINT CD$"BLOAD ADOBJ,A"AD
1015 FOR I=0 TO NS: D%=OP: CALL AD: D%(I,B0)=D%: U=USR(N):
     FOR J=0 TO DLY: NEXT: NEXT: U=USR(H): TEXT: RETURN
```

```
                0002  ;TITLE ADOBJ
                0004  ;ADALAB A/D ROUTINE
                0010  DPER    EQU 6002
                0020  BASE1   EQU C200
                0100          ORG 0320
                0105          OBJ 7000
0320  AD0360    0110  ADCONV  LDA DPER+01
0323  18        0120          CLC
0324  69C0      0130          ADC #$C0
0326  8D3903    0140          STA ADR1+02
0329  8D4603    0150          STA ADR2+02
032C  8D4C03    0160          STA ADR3+02
032F  8D3C03    0170          STA READ+02
0332  8D3F03    0180          STA WAIT+02
0335  A98A      0190          LDA #$8A
0337  8D0CC2    0200  ADR1    STA BASE1+0C
033A  8D00C2    0210  READ    STA BASE1
033D  AD0DC2    0220  WAIT    LDA BASE1+0D
0340  2910      0230          AND #$10
0342  F0F9      0240          BEQ WAIT
0344  AD10C2    0250  ADR2    LDA BASE1+10
0347  8D0360    0260          STA DPER+01
034A  AD20C2    0270  ADR3    LDA BASE1+20
034D  8D0260    0280          STA DPER
0350  2920      0290          AND #$20
  52  D013      0300          BNE PLUS
0354  AD0360    0310  MINUS   LDA DPER+01
0357  49FF      0320          EOR #$FF
0359  8D0360    0330          STA DPER+01
035C  AD0260    0340          LDA DPER
035F  49FF      0350          EOR #$FF
0361  09E0      0360          ORA #$E0
0363  8D0260    0370          STA DPER
0366  60        0380          RTS
0367  AD0260    0390  PLUS    LDA DPER
036A  291F      0400          AND #$1F
036C  8D0260    0410          STA DPER
036F  60        0420          RTS
```

```
LABEL  TABLE
DPER    6002
BASE1   C200
ADCONV  0320
ADR1    0337
READ    033A
WAIT    033D
ADR2    0344
ADR3    034A
MINUS   0354
  US    0367
```

```
0320- AD 03 60 18 69 C0 8D 39
0328- 03 8D 46 03 8D 4C 03 8D
0330- 3C 03 8D 3F 03 A9 8A 8D
0338- 0C C2 8D 00 C2 AD 0D C2
0340- 29 10 F0 F9 AD 10 C2 8D
0348- 03 60 AD 20 C2 8D 02 60
0350- 29 20 D0 13 AD 03 60 49
0358- FF 8D 03 60 AD 02 60 49
0360- FF 09 E0 8D 02 60 60 AD
0368- 02 60 29 1F 8D 02 60 60
```

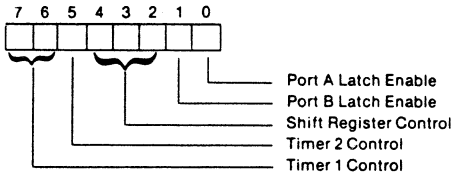
```
JLIST 1000-

1000  GOSUB 80:U =  USR (B0 + 1) +
      USR ( - 3071): POKE XC,0:N =
     78
1005  IF OP < 1 OR OP > 7 GOTO 72

1010  IF AD = 0 THEN AD = 800: PRINT
     CD$"BLOAD ADOBJ,A"AD
1015  FOR I = 0 TO NS:D% = OP: CALL
     AD:D%(I,B0) = D%:U =  USR (N
     ): FOR J = 0 TO DLY: NEXT : NEXT
     :U =  USR (H): TEXT : RETURN
```

USER R6522 VERSATILE INTERFACE ADAPTER (VIA)

R6522 MEMORY ASSIGNMENTS

Location	Function		
0	Port B Output Data Register (ORB)		
1	Port A Output Data Register (ORA)	**Controls handshake**	
2	Port B Data Direction Register (DDRB)	} 0 = Input	
3	Port A Data Direction Register (DDRA)	} 1 = Output	
	Timer	**R/$\overline{W}$ = L**	**R/$\overline{W}$ = H**
4	T1	Write T1L-L	Read T1C-L Clear T1 Interrupt Flag
5	T1	Write T1L-H & T1C-H T1L-L → T1C-L Clear T1 Interrupt Flag	Read T1C-H
6	T1	Write T1L-L	Read T1L-L
7	T1	Write T1L-H Clear T1 Interrupt Flag	Read T1L-H
8	T2	Write T2L-L	Read T2C-L Clear T2 Interrupt Flag
9	T2	Write T2C-H T2L-L → T2C-L Clear T2 Interrupt Flag	Read T2C-H
A	Shift Register (SR)		
B	Auxiliary Control Register (ACR)		
C	Peripheral Control Register (PCR)		
D	Interrupt Flag Register (IFR)		
E	Interrupt Enable Register (IER)		
F	Port A Output Data Register (ORA)	*No effect on handshake*	

R6522 AUXILIARY CONTROL REGISTER (ACR)



PORT A LATCH ENABLE

ACR0 = 1 Port A latch is enabled to latch input data when CA1 Interrupt Flag (IFR1) is set.
= 0 Port A latch is disabled, reflects current data on PA pins.

PORT B LATCH ENABLE

ACR1 = 1 Port B latch is enabled to latch the voltage on the pins for the input lines or the ORB contents for the output lines when CB1 Interrupt Flag (IFR4) is set.
= 0 Port B latch is disabled, reflects current data on PB pins.

SHIFT REGISTER CONTROL

ACR4	ACR3	ACR2	Mode
0	0	0	Shift Register Disabled.
0	0	1	Shift in under control of Timer 2.
0	1	0	Shift in under control of ø2.
0	1	1	Shift in under control of external clock.
1	0	0	Free-running output at rate determined by Timer 2.
1	0	1	Shift out under control of Timer 2.
1	1	0	Shift out under control of ø2.
1	1	1	Shift out under control of external clock.

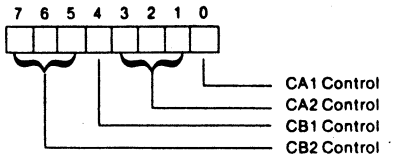
TIMER 2 CONTROL

ACR5 = 0 T2 acts as an interval timer in the one-shot mode.
= 1 T2 counts a predetermined number of pulses on PB6.

TIMER 1 CONTROL

ACR7	ACR6	Mode
0	0	T1 one-shot mode — Generate a single time-out interrupt each time T1 is loaded. Output to PB7 disabled.
0	1	T1 free-running mode — Generate continuous interrupts. Output to PB7 disabled.
1	0	T1 one-shot mode — Generate a single time-out interrupt and an output pulse on PB7 each time T1 is loaded.
1	1	T1 free-running mode — Generate continuous interrupts and a square wave output on PB7.

R6522 PERIPHERAL CONTROL REGISTER (PCR)



CA1 CONTROL

PCR0 = 0 The CA1 Interrupt Flag (IFR1) will be set by a negative transition (high to low) on the CA1 pin.
= 1 The CA1 Interrupt Flag (IFR1) will be set by a positive transition (low to high) on the CA1 pin.

CA2 CONTROL

PCR3	PCR2	PCR1	Mode
0	0	0	CA2 negative edge interrupt (IFR0/ORA clear) mode — Set CA2 interrupt flag (IFR0) on a negative transition of the CA2 input signal. Clear IFR0 on a read or write of the ORA or by writing logic 1 into IFR0.
0	0	1	CA2 negative edge interrupt (IFR0 clear) mode — Set IFR0 on a negative transition of the CA2 input signal. Clear IFR0 by writing logic 1 into IFR0.
0	1	0	CA2 positive edge interrupt (IFR0/ORA clear) mode — Set CA2 interrupt flag (IFR0) on a positive transition of the CA2 input signal. Clear IFR0 on a read or write of the ORA or by writing logic 1 into IFR0.
0	1	1	CA2 positive edge interrupt (IFR0 clear) mode — Set IFR0 on a positive transition of the CA2 input signal. Clear IFR0 by writing logic 1 into IFR0.
1	0	0	CA2 handshake output mode — Set CA2 output low on a read or write of the Peripheral A Output Register. Reset CA2 high with an active transition on CA1.
1	0	1	CA2 pulse output mode — CA2 goes low for one cycle following a read or write of the Peripheral A Output Register.
1	1	0	CA2 low output mode — The CA2 output is held low in this mode.
1	1	1	CA2 high output mode — The CA2 output is held high in this mode.

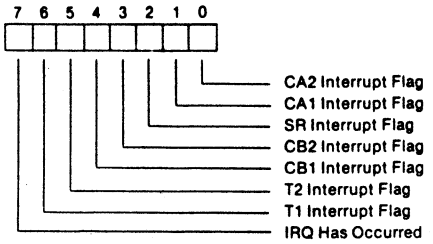
CB1 CONTROL

PCR4 = 0 The CB1 Interrupt Flag (IFR4) will be set by a negative transition (high to low) on the CB1 pin.
= 1 The CB1 Interrupt Flag (IFR4) will be set by a positive transition (low to high) on the CB1 pin.

CB2 CONTROL

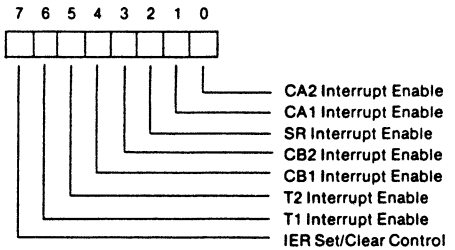
PCR7	PCR6	PCR5	Mode
0	0	0	CB2 negative edge interrupt (IFR3/ORB clear) mode — Set CB2 interrupt flag (IFR3) on a negative transition of the CB2 input signal. Clear IFR3 on a read or write of the ORB or by writing logic 1 into IFR3.
0	0	1	CB2 negative edge interrupt (IFR3 clear) mode — Set IFR3 on a negative transition of the CB2 input signal. Clear IFR3 by writing logic 1 into IFR3.
0	1	0	CB2 positive edge interrupt (IFR3/ORB clear) mode — Set CB2 interrupt flag (IFR3) on a positive transition of the CB2 input signal. Clear IFR3 on a read or write of the ORB or by writing logic 1 into IFR3.
0	1	1	CB2 positive edge interrupt (IFR3 clear) mode — Set IFR3 on a positive transition of the CB2 input signal. Clear IFR3 by writing logic 1 into IFR3.
1	0	0	CB2 handshake output mode — Set CB2 output low on a write of the Peripheral ß Output Register. Reset CB2 high with an active transition on CB1.
1	0	1	CB2 pulse output mode — CB2 goes low for one cycle following a write of the Peripheral ß Output Register.
1	1	0	CB2 low output mode — The CB2 output is held low in this mode.
1	1	1	CB2 high output mode — The CB2 output is held high in this mode.

R6522 INTERRUPT FLAG REGISTER (IFR)



IFR Bit	Set By	Cleared By
0	Active transition on CA2	Reading or writing the ORA ($A001 or $A00F)
1	Active transition on CA1	Reading or writing the ORA ($A001 or $A00F)
2	Completion of eight shifts	Reading or writing the SR ($A00A)
3	Active transition on CB2	Reading or writing the ORB ($A000)
4	Active transition on CB1	Reading or writing the ORB ($A000)
5	Time-out of Timer 2	Reading T2C-L ($A008) or writing T2C-H ($A009)
6	Time-out of Timer 1	Reading T1C-L ($A004) or writing T1C-H ($A005 or $A007)
7	Any IFR bit set with its corresponding IER bit also set	Clearing IFR0-IFR6 ($A00D) or IER0-IER6 ($A00E)

R6522 INTERRUPT ENABLE REGISTER (IER)



INTERRUPT ENABLE BITS (IER0-6)

IERn = 0 Disable interrupt
= 1 Enable interrupt

IER SET/CLEAR CONTROL (IER7)

IER7 = 0 For each data bus bit set to logic 1, clear corresponding IER bit
= 1 For each data bus bit set to logic 1, set corresponding IER bit.

Note: IER7 is active only when R/$\overline{W}$ = L; when R/$\overline{W}$ = H, IER7 will read logic 1.

MCS6522
Versatile Interface Adapter (VIA)

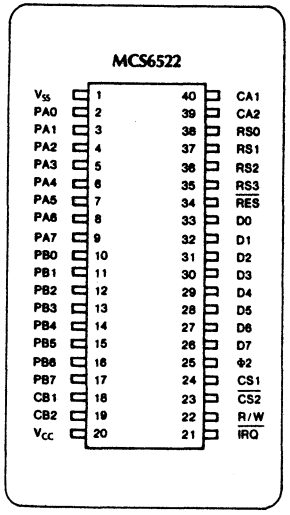
- **Completely Static**
- **Fully TTL Compatible**
- **CMOS Compatible Peripheral Control Lines**
- **8-Bit Bidirectional Data/Control Transfer**
- **2 Powerful Interval Timers**
- **Shift Register for Serial/Parallel and Parallel/Serial Transfers**
- **Input Data Latching on Peripheral Ports**
- **Fully Automatic Handshake**
- **Independent Interrupt Control**
- **Single +5V Supply**

DESCRIPTION

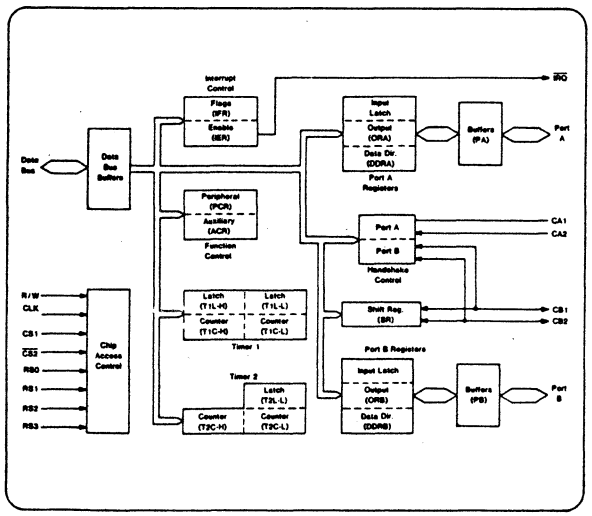
The MCS6522 Versatile Interface Adapter (VIA) provides all of the capability of the MCS6520 Peripheral Adapter. In addition, it offers a pair of powerful interval timers, a serial-to-parallel/parallel-to-serial shift register and input data latching on the peripheral ports. Expanded handshaking capability over that of the MCS6520 allows control of bidirectional data transfers between VIAs in a multiple processor system.

Control of peripheral devices is handled primarily through two 8-bit bidirectional ports. Each line in these ports can be programmed to act as either an input or an output. Serveral peripheral I/O lines can also be controlled directly from the MCS6522's internal interval timer, permitting the generation of programmable-frequency square waves and for counting pulses generated externally. Internal registers are organized into an interrupt flag register, an interrupt enable register and a pair of function control registers. This permits easy control of the many features of the device.

PIN CONFIGURATION



BLOCK DIAGRAM



4

INTERFACE TO THE PROCESSOR

This section contains a description of the buses and control lines which are used to interface the MCS6522 to the system processor.

Phase Two Clock (Φ2). Data transfers between the MCS6522 and the system processor take place only while the Phase Two Clock is high. In addition, Φ2 acts as the time base for the various timers and shift registers on the chip.

Chip Select Lines (CS1, $\overline{CS2}$). The two chip select inputs are normally connected to processor address lines either directly or through decoding. The selected MCS6522 register will be accessed when CS1 is high and $\overline{CS2}$ is low.

Register Select Lines (RS0, RS1, RS2, RS3). The four Register select lines are normally connected to the processor address bus lines to allow the processor to select the internal MCS6522 register which is to be accessed. The sixteen possible combinations access the registers shown in Table 1.

Table 1. Register Select Line Definitions

RS3	RS2	RS1	RS0	Register	Remarks
L	L	L	L	ORB	
L	L	L	H	ORA	Controls Handshake
L	L	H	L	DDRB	
L	L	H	H	DDRA	
L	H	L	L	T1L-L T1C-L	Write Latch Read Counter
L	H	L	H	T1C-H	Trigger T1L-L/ T1C-L Transf.
L	H	H	L	T1L-L	
L	H	H	H	T1L-H	
H	L	L	L	T2L-L T2C-L	Write Latch Read Counter
H	L	L	H	T2C-H	Triggers T2L-L/ T2C-L Transfer
H	L	H	L	SR	
H	L	H	H	ACR	
H	H	L	L	PCR	
H	H	L	H	IFR	
H	H	H	L	IER	
H	H	H	H	ORA	No Effect on Handshake

Read/Write Line (R/W). The direction of data transfers between the MCS6522 and the system processor is controlled by the R/W line. If R/W is low, data will be transferred out of the processor into the selected MCS6522 register (write operation). If R/W is high and the chip is selected, data will be transferred out of the MCS6522 to the data bus (read operation).

Data Bus (DB0 - DB7). The 8 bi-directional data bus lines are used to transfer data between the MCS6522 and the system processor. The internal drivers will remain in the high-impedance state except when the chip is selected (CS1 = 1, $\overline{CS2}$ = 0), Read/Write is high and the Phase Two Clock is high. At this time, the contents of the selected register are placed on the data bus. When the chip is selected, with Read/Write low and Φ2 = 1, the data on the data bus will be transferred into the selected MCS6522 register.

Reset ($\overline{RES}$). The Reset input clears all internal registers (except T1, T2, and SR) to logic 0. This places all peripheral interface lines in the input state, disables the timers, shift register, and interrupts from the chip.

Interrupt Request ($\overline{IRQ}$). The Interrupt Request output goes low whenever an internal interrupt flag is set and the correspondeing interrupt enable bit is a logic 1. This output is "open drain" to allow the interrupt request signal to be wire-ORed with other equivalent signals in the system.

INTERFACE TO THE PERIPHERAL. This section contains a brief description of the buses and control lines used to drive peripheral devices under control of the MCS6522 registers.

Peripheral A Port (PA0 - PA7). The Peripheral A port consists of 8 lines which can be individually programmed to act as input or output under control of a Data Direction Register. The polarity of output pins is controlled by an Output Register and input data can be latched into an internal register under control of the CA1 line. All of these modes of operation are controlled by the system processor through the internal control registers. These lines represent one standard TTL load in the input mode and will drive one standard TTL load in the output mode.

Peripheral A Control Lines (CA1, CA2). The two peripheral A control lines act as interrupt inputs or as handshake outputs. Each line controls an internal interrupt flag with a corresponding interrupt enable bit. In addition, CA1 controls the latching of data on Peripheral A Port input lines. The various modes of operation are controlled by the system processor through the internal control registers. CA1 is a high-impedance input only while CA2 represents one standard TTL load in the input mode. CA2 will drive one standard TTL load in the output mode.

Peripheral B Port (PB0 - PB7). The Peripheral B port consists of 8 bi-directional lines controlled by an output register and a Data Direction Register in much the same manner as the PA port. In addition, the polarity of the PB7 output signal can be controlled by one of the interval timers while the second timer can be programmed to count pulses on the PB6 pin. These lines represent one standard TTL load in the input mode and will drive one standard TTL load in the output mode.

Peripheral B Control Lines (CB1, CB2). The Peripheral B control lines act as interrupt inputs or as handshake outputs. As with CA1 and CA2, each line controls an interrupt flag with a corresponding interrupt enable bit. In addition, these lines act as a serial port under control of the Shift Register. These lines represent one standard TTL load in the input mode and will drive one standard TTL load in the output mode.

OPERATION

This section contain a discussion of the various blocks of logic shown in the block diagram. In addition, the internal operation of the MCS6522 is described in detail.

Chip Access Control.
The Chip Access Control contains the necessary logic to detect the chip select condition and to decode the Register Select inputs to allow access to the desired register. In addition, the R/W and Φ2 signals are utilized to control the direction and timing of data transfers. When writing into the MCS6522, data is first latched into a data input register during Φ2. Data is then transferred into the desired internal register during $\overline{\Phi 2}$ - $\overline{\text{Chip Select}}$. This allows the peripheral I/O line to change without "glitching." When the processor reads the MCS6522, data is transferred from the desired internal register directly onto the Data Bus during Φ2.

Port A Registers, Port B Registers
Three registers are used in accessing each of the 8-bit peripheral ports. Each port has a Data Direction Register (DDRA, DDRB) for specifying whether the peripheral pins are to act as inputs or outputs. A 0 in a bit of the Data Direction Register causes the corresponding peripheral pin to act as an input. A 1 causes the pin to act as an output.

Each peripheral pin is also controlled by a bit in the Output Register (ORA, ORB) and an Input Register (IRA,IRB). When the pin is programmed to act as an output, the voltage on the pin is controlled by the corresponding bit of the Output Register. A 1 in the Output Register causes the pin to go high, and a 0 causes the pin to go low.

Reading a peripheral port causes the contents of the Input Register (IRA, IRB) to be transferred onto the Data Bus. With input latching disabled, IRA will always reflect the data on the PA pins. With input latching enabled, IRA will reflect the contents of the Port A prior to setting the CA1 Interrupt Flag (IFR1) by an active transition on CA1.

The IRB register operates in a similar manner. However, for output pins, the corresponding IRB bit will reflect the contents of the Output Register bit instead of the actual pin. This allows proper data to be read into the processor if the output pin is not allowed to go to full voltage. With input latching enabled on Port B, setting CB1 interrupt flag will cause IRB to latch this combination of input data and ORB data until the interrupt flag is cleared.

Handshake Control
Th MCS6522 allows positive control of data transfers between the system processor and peripheral devices through the operation of "handshake" lines. Port A lines (CA1, CA2) handshake data on both a read and a write operation while the Port B lines (CB1, CB2) handshake on a write operation only.

Read Handshake. Positive control of data transfers from peripheral devices into the system processor can be accomplished using "Read" handshaking. In this case, the peripheral device must generate "Data Ready" to signal the processor that valid data is present on the peripheral port. This signal normally interrupts the processor, which then reads the data, causing generation of a "Data Taken" signal. The peripheral device responds by making new data available. This process continues until the data transfer is complete.

In the MCS6522, automatic "Read" handshaking is possible on the Peripheral A port only. The CA1 interrupt input pin accepts the "Data Ready" signal and CA2 generates the "Data Taken" signal. The Data Ready signal will set an internal flag which may either interrupt the processor or be polled by software. The Data Taken signal can be either a pulse or a DC level which is set low by the system processor and cleared by the Data Ready signal. These options are shown in Figure 1 which illustrates the normal Read handshaking sequence.

Write Handshake. The sequence of operations which allows handshaking data from the system processor to a peripheral device is very similar to that described for Read Handshaking. However, for "Write" handshaking, the processor must generate the "Data Ready"signal (through the MCS6522) and the peripheral device must respond with the "Data Taken" signal. This can be accomplished on both the PA port and the PB port on the MCS6522. CA2 or CB2 acts as a Data Ready output in either the DC level or pulse mode and CA1 or CB1 accepts the "Data Taken" signal from the peripheral device, setting the interrupt flag and clearing the "Data Ready" output. This sequence is shown in Figure 2.

4

6522

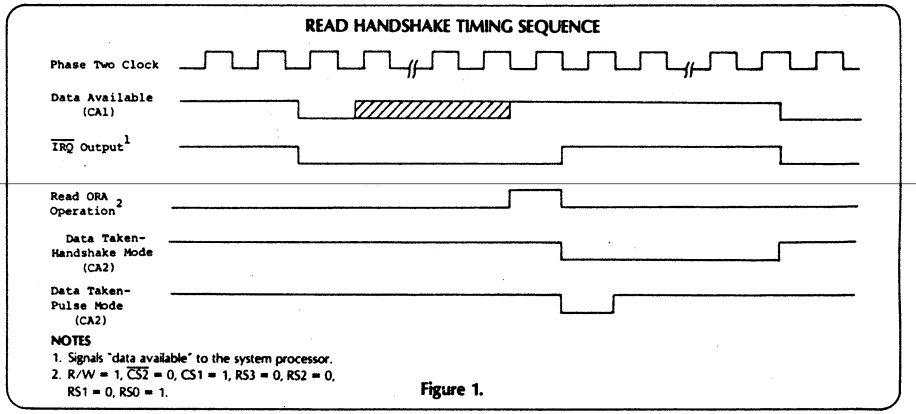


Figure 1.

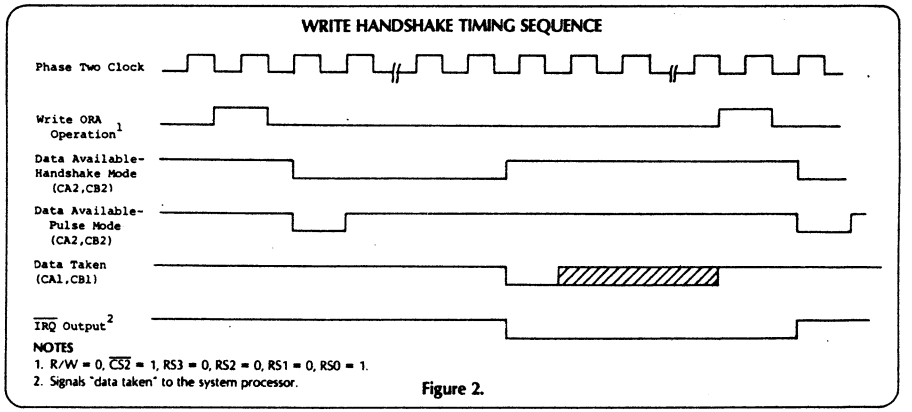


Figure 2.

Timer 1 (T1)
Interval Timer T1 consists of two 8-bit latches and a 16-bit counter. The latches are used to store data to be loaded into the counter. After loading, the counter decrements at system clock rate. Upon reaching zero, an interrupt flag will be set, and $\overline{\text{IRQ}}$ will go low. The timer will then disable any further interrupts, or automatically transfer the contents of the latches into the counter and continue to decrement. In addition, the timer can be instructed to invert the output signal on a peripheral pin each time it "times-out". Each of these modes is discussed separately below.

Writing T1. Operations which take place when writing to each of the four T1 addresses are shown in Table 2.

Table 2. Writing to T1 Registers

RS3	RS2	RS1	RS0	Operation (R/W = L)
L	H	L	L	Write into low order latch.
L	H	L	H	Write into high order latch. Write into high order counter. Transfer low order latch into low order counter. Reset T1 interrupt flag.
L	H	H	L	Write low order latch.
L	H	H	H	Write high order latch. Reset T1 interrupt flag.