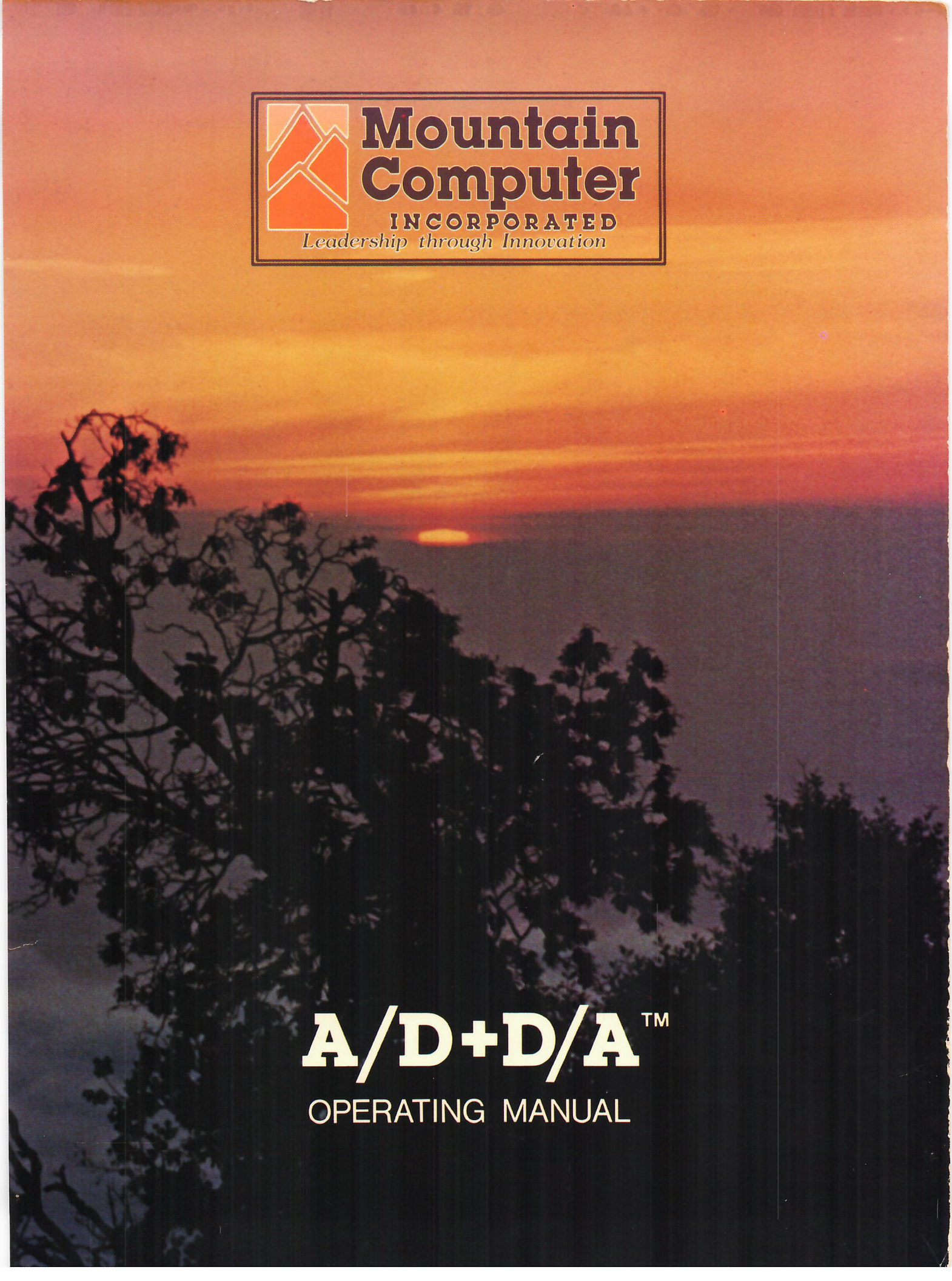

Mountain
Computer
INCORPORATED
Leadership through Innovation
A/D+D/A™
OPERATING MANUAL

S21

A/D + D/A

Operating Manual

Release 1.1

Copyright (C) 1981

Mountain Computer Inc.

Table of Contents

Introduction

Chapter 1 Installation

Chapter 2 Self Test

Chapter 3 Temp Test

Chapter 4 Lite Test

Chapter 5 Curve Tracer

Chapter 6 A Reference

Appendix A Specifications

Appendix B Programming Considerations

Appendix C Interface Information

Warranty

Product Evaluation Form

Introduction

The A/D + D/A board is an exciting new product that works with your Apple II* to convert data from analog to digital and digital to analog. The fast and efficient A/D + D/A board provides 16 channels of analog input to the Apple, and 16 channels of analog output from the Apple.

The A/D + D/A board was designed to operate in all Apple II and Apple II Plus computers. In fact, for applications requiring large amounts of analog data I/O, the A/D + D/A board installed in a minimum memory and component computer system is ideal.

You can use the A/D + D/A board in conjunction with a modem to make remote collection and transmission of data possible. Or you can monitor or transmit real-time data by using the Mountain Computer Apple Clock (part number MHP-X003) with the A/D + D/A board. Programming an interface between the A/D + D/A board and other peripheral devices is usually easy and can be accomplished by most dedicated hobbyists and professionals.

Hardware

The A/D + D/A board uses an eight bit digital to analog convertor (DAC) for analog output and an eight bit succesive approximation register (SAR) for analog to digital conversion (ADC). You can use these parts of the A/D + D/A board independently or combine them into an integrated bi-directional conversion system.

There are 16 channels of analog output from the DAC. Each channel can be programmed to output a voltage in the range of -5 to +5 volts.

There are also 16 channels of analog input to the ADC. Each channel of the ADC will accept voltages in the range of -5 to +5 volts and convert that voltage to a digital value. The ADC will convert the voltage from a selected channel to a digital value in 9 microseconds.

Programming the DAC channels, starting A/D conversions, and reading the ADC channels are all accomplished by reading or writing to one of 16 slot dependent addresses.

*Apple II is a trademark of Apple Computer Inc. of Cupertino, Ca.

Software

No special software is needed to use the A/D + D/A board. We have chosen not to provide "data acquisition" or "data transmission" programs. Instead we have included several skeletal programs that you can build on as you create your own applications for the A/D + D/A board. Each program is fully documented. The following programs are included:

Self Test -- a quick, practical way to test the performance of each of the 16 pairs of analog to digital convertors (ADC) and digital to analog convertors (DAC).

Temp Test -- a simple circuit built from easily obtainable parts senses the temperature; the A/D + D/A board converts the temperature reading into digital values and displays them on the screen.

Lite Test -- a photo-sensitive circuit monitors the amount of light present and produces "relative light value" readings, then displays the changing values on the screen.

Curve Tracer -- the performance of a transistor is measured and plotted on the screen. This may be used as a base program for automating test equipment.

These simple applications are designed to serve as skeleton programs on which similar applications can be based. They are perfect for the novice electronics hobbyist because they are free of extraneous programming options. These four programs are written in Applesoft BASIC.

Optional I/O Cable Assembly

Available as a separate option is the Mountain Computer I/O Cable Assembly (part number MHP-X027) for connecting the analog to digital convertor (ADC) and digital to analog convertor (DAC) to a wide variety of data collection and transmission devices. These cables are manufactured expressly for the A/D + D/A board and are available from all Mountain Computer Dealers. Generic substitutes are available, but we cannot guarantee their performance. The product performance specifications and warranty include the Mountain Computer I/O Cables.

Chapter 1 Installation

Installing the Mountain Computer A/D + D/A board is easy and quick. If you have ever installed a peripheral board in an Apple II computer, you should have no problems with the A/D + D/A board or the I/O cable assembly.

Hardware Requirements

The A/D + D/A board will work with almost every possible system configuration for the Apple II. However, if you plan to use the software provided on the Demonstration Diskette, your Apple must have the the following features:

- At least one disk drive.
- A set of I/O cables.
- 48K RAM memory.

NOTE: If you use I/O cables that are not wired like the Mountain Computer Inc. cables, you may have trouble with the Self Test program.

Software Requirements

The A/D + D/A board can be used in any Apple II or Apple II Plus computer. Pascal systems can use the A/D + D/A board, but no Pascal demonstration programs are included on the Demonstration Diskette.

Unpacking

The A/D + D/A board is packaged in a specially designed carton. Each unit is inspected and tested before it leaves our factory. As you unpack the board, observe how it is packed in case you wish to store or transport it.

As you unpack the A/D + D/A board, take a few moments to fill out the Warranty Registration Card. The Registration Card provides us with a way of sending you information about new products and A/D + D/A updates.

Parts List

Before installing your A/D + D/A board, check that the following parts are included in your package:

- A/D + D/A printed circuit board
- Demonstration Diskette
- Operating Manual
- Warranty Registration Card

If any part is missing, contact your dealer.

Installing

Follow the standard precautions for installing Apple peripherals:

- Make sure the Apple's power is OFF.
- Touch the power supply to discharge static electricity.
- Do not install the board in slot #0.

If you are using I/O Cables (Mountain Computer Inc. part number MHP-X027), attach them now. See Appendix C in this manual for instructions on attaching the cables.

When the cables are attached, carefully plug the A/D + D/A board into an available Apple peripheral slot. Make sure the A/D + D/A board is firmly seated in the slot.

Thread the I/O cables out the back of the Apple, and connect the desired circuits and interface units. (See chapters 2 through 5 for information on special attachments you may need for your application.)

That's all there is to installing the A/D + D/A board. Replace the Apple's cover and turn the power on before continuing.

The Demonstration Diskette

Before using the A/D + D/A Demonstration Diskette, be sure to make at least one backup copy of it. Store the original diskette in a safe place, and use the copy for a "working" diskette.

Boot the Demonstration Diskette. A title screen will be displayed for a few seconds, and then the diskette catalog will appear. The demonstration programs you will use are called Self Test, Temp Test, Lite Test and Curve Tracer. These programs demonstrate a variety of applications for the A/D + D/A board.

NOTE: Each test requires a slightly different hardware setup. The necessary hardware for each test is documented in chapters 2 through 5. Don't run the programs before reading the appropriate chapters.

Chapter 2 Self Test

This program provides a way to test whether the A/D + D/A board is properly converting digital signals to analog signals and vice versa. The program is simple to run and displays the conversion results in one of two formats.

The first type of display is a Transfer Chart which is convenient for viewing gross errors in the transfer function. The second type of display is a Monotonicity Chart which is more convenient for viewing errors of 4 units or less.

Purpose

The Self Test program tests all 16 DAC's and all 16 ADC's on the A/D + D/A board as a set of pairs by comparing the digital values written to the DAC with the digital values produced by the ADC. The results of the Self Test program indicate whether all parts of the A/D + D/A board are functional and properly installed. These results are printed on the video screen.

NOTE: If a channel is not working properly, you should avoid using it until the board has been repaired. Do not attempt to make modifications to the board yourself in the event of a malfunctioning DAC or ADC.

Introduction

Before you can use the Self Test program, the I/O cables must be correctly installed. The "header" ends of each cable must be attached to the ADC and DAC pins of the A/D + D/A board. The ends of the cables with DB-25 connectors must be attached to each other, creating a closed loop. The loop connects each channel of the ADC part of the board to the channel of the same number on the DAC part of the board. For more information on installing the I/O cables, see Appendix C.

When the cable is correctly assembled and the Self Test program is running, the Apple provides a series of 256 digital values to the DAC. The DAC then sends a series of analog signals through the cable to the ADC unit, and the analog signals are converted to digital values. If the DAC and ADC units are working properly, the digital values read at the ADC will equal the original digital values output to the DAC.

Here's how it works. As voltage is passed to the ADC, a digital value is produced and stored in an array in memory. The digital value output to the DAC is increased by one, converted from analog to digital, and stored in the next array location. When

all the digital values for that pair of channels have been converted and stored in an array in the Apple's memory, a display of the conversion results is plotted on the screen. You can choose which type of display, a Transfer Chart or a Monotonicity Chart, you wish to see.

1) The Transfer Chart -- on the X axis is the original value sent to the DAC; on the Y axis is the converted digital value from the ADC. The values are displayed in hexadecimal notation. A perfect transfer chart is a straight diagonal line from the lower left to upper right corners of the chart.

2) The Monotonicity Chart -- shows the deviation of the original value and the converted value. On the X axis is the original value sent to the DAC; on the Y axis is a scale from +4 to -4, indicating the decimal range of the deviation. A perfect result is a horizontal line from left to right starting at 0. However, the more general results of a "bar graph" ranging from +1 to -1 are quite acceptable and within the A/D + D/A board specification range.

When the test results are displayed, the system waits until you initiate the test of another channel or exit the Self Test program. If you want to see the other type of display, you must exit the program and re-run it, this time choosing the other display.

Components

I/O Cable assembly

Running the Program

Boot the A/D + D/A Demonstration Diskette. To run the Self Test program demonstration, type

```
RUN SELF TEST
```

and press the RETURN key. The screen will clear and the following display will appear:

```
MOUNTAIN COMPUTER
A/D + D/A
SELF TEST

WHICH SLOT IS THE A/D + D/A IN?
```

Enter the number of the slot in which the A/D + D/A board is plugged. If you enter an illegal value (anything but an integer from 1 to 7), the value will not be accepted, and the cursor will

continue flashing over the entered number. If you give the wrong slot number, the number will be accepted, and the program will seem to run fine, but the test results will not be valid.

When the slot number has been accepted, the following display will appear:

```
PLEASE MAKE SURE THE CABLES ARE IN PLACE

WHICH DISPLAY FORMAT WOULD YOU LIKE?

A - A/D VS D/A TRANSFER GRAPH
B - MONOTONICITY GRAPH

YOUR CHOICE (A OR B)?
```

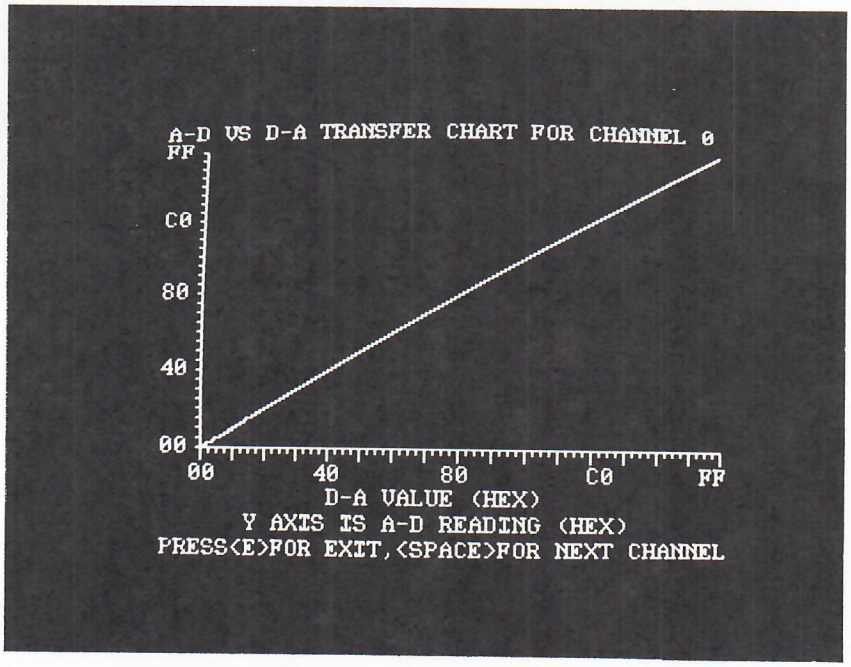
First let's look at display format A, the A/D + D/A Transfer Chart. Type

```
A
```

You are then prompted to

```
PRESS SPACE TO START TEST WITH CHANNEL 0
PRESS (0-9) OR (A-F) TO START
TEST WITH ANY CHANNEL
YOUR CHOICE ->
```

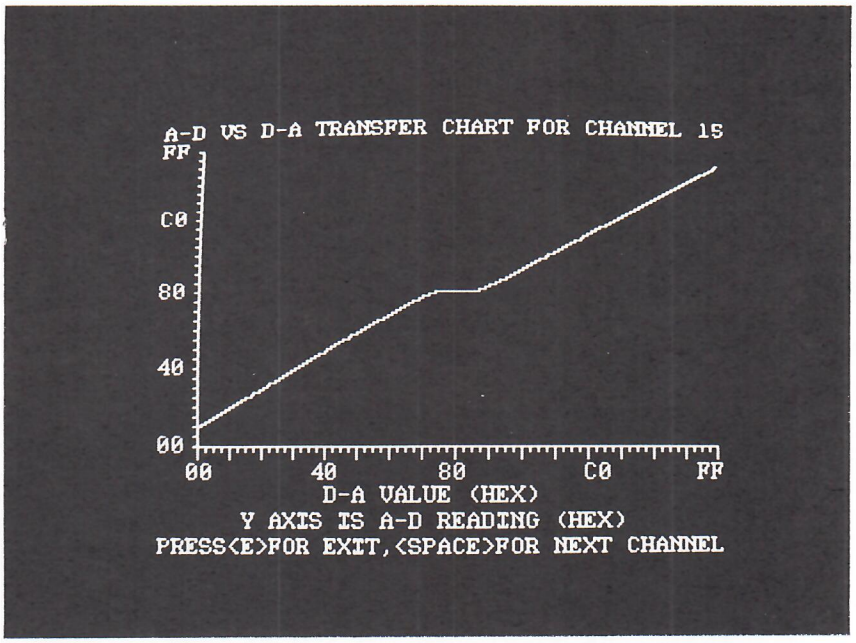
If you enter a hexadecimal digit from 0 to F, the pair of ADC and DAC channels associated with that number will begin converting signals. When all 256 digital values have been converted, the display for that channel appears on the screen:



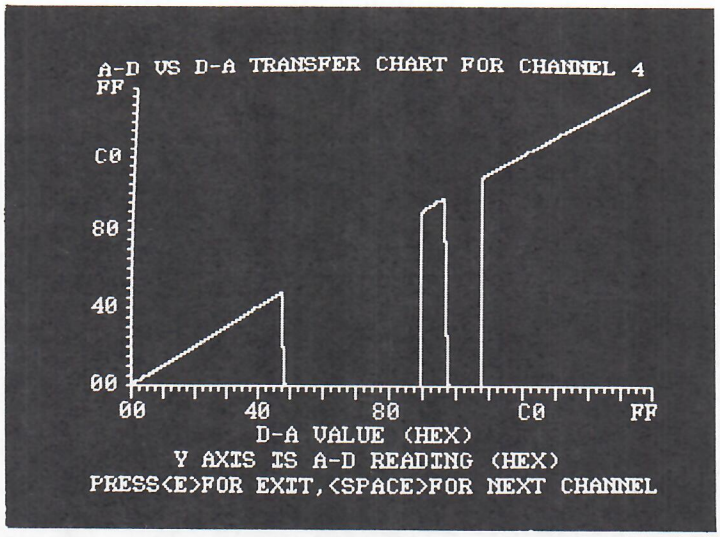
The line on the graph in the previous photograph is not perfectly straight. The small "gliches" in the line indicate variations of plus or minus one. A few variations of plus or minus one are allowable, and are within the A/D + D/A board specifications. A greater variation than that probably indicates a malfunction.

The number of the channel just tested is displayed at the top right of the screen. The X (horizontal) axis indicates, in hexadecimal format, the original value sent to the DAC. The Y (vertical) axis indicates, in hexadecimal format, the digital value converted from the analog signal by the ADC. The results of the transfer function are displayed as a line plotted on the graph. This line indicates which, if any, values were not properly transferred from digital to analog.

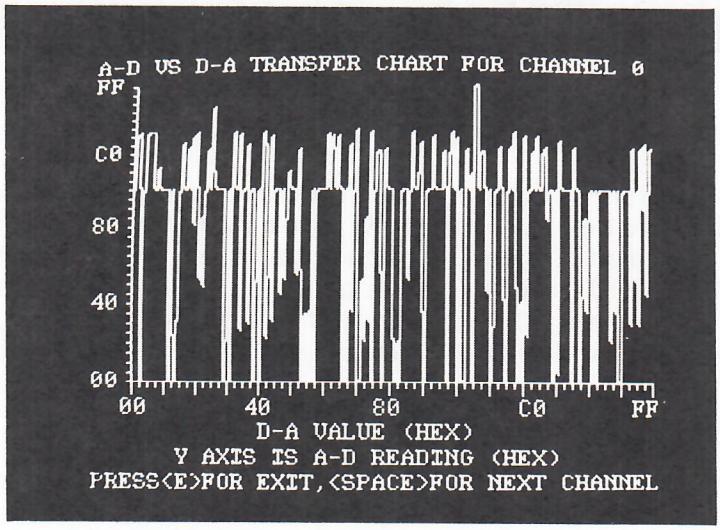
If the channel being tested is not functioning properly, the transfer function display will not be a straight line. The way the display looks depends on what is wrong with the channel. It could look like the following photograph if the problem is minor:



If the problem is more extensive, the display might look like the following photograph:



A display that is totally out of whack may indicate that you have entered the wrong slot number at the beginning of the test. If that is the case, the Self Test program will run normally, but the display might look like the following illustration:

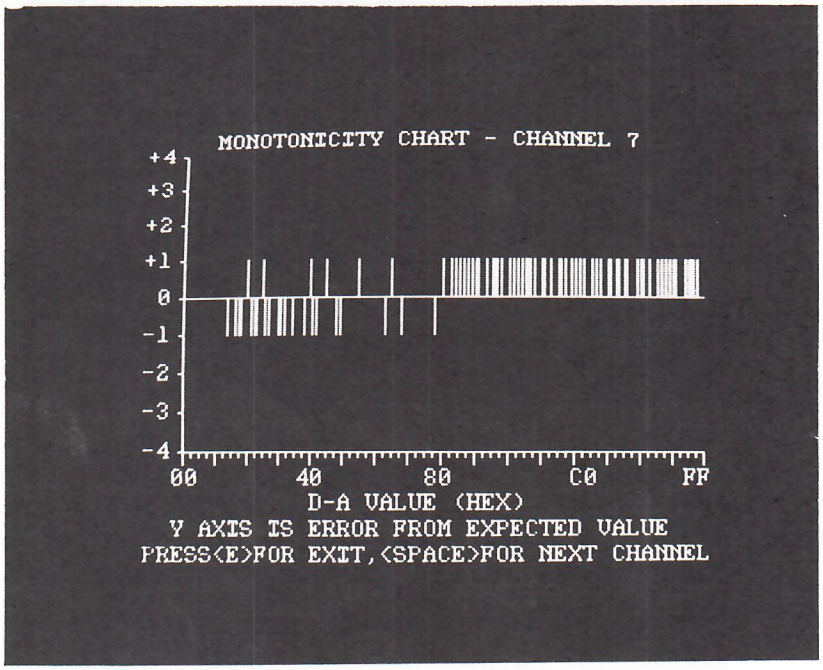


When you have finished looking at Transfer Charts, press E to exit the Self Test program, as indicated by the prompt line at the bottom of the screen.

To use display format B, the Monotonicity Chart, you must re-run the program. Type

RUN

This time, when you are prompted to choose the type of display choose B for the Monotonicity Chart. After you specify a channel, (or start with channel 0 by pressing the space bar), the Monotonicity Chart will appear:

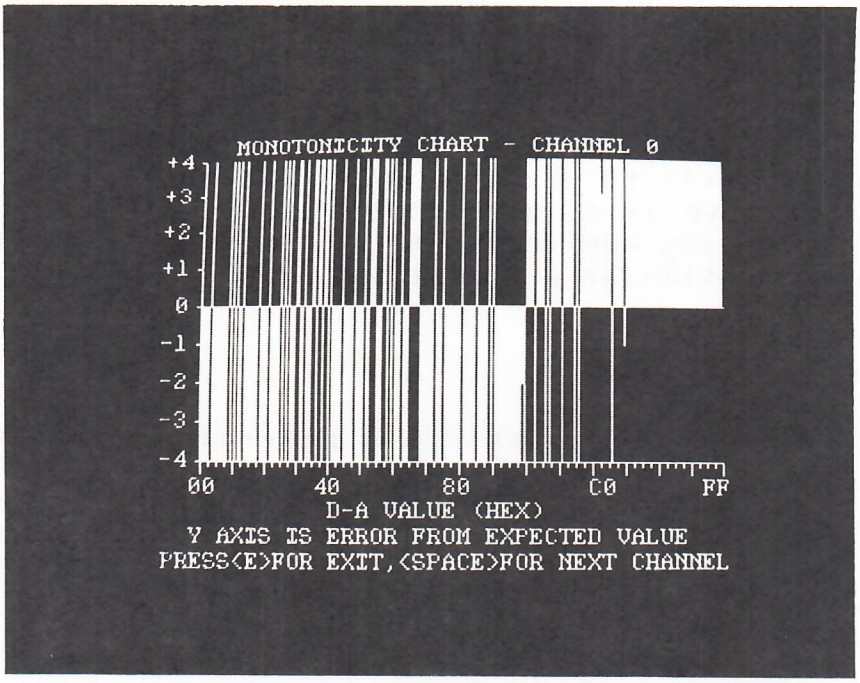


As with the Transfer Chart, a deviation of plus or minus one unit is normal.

The number of the channel being tested is indicated at the top right of the screen. The X (horizontal) axis indicates, in hexadecimal format, the digital values that have been converted. The Y (vertical) axis indicates, in decimal format, the deviations between the original values sent to the DAC and the digital values received from the ADC. The horizontal line that starts at 0 indicates which, if any, values include a deviation factor when converted from digital to analog.

If the Monotonicity Chart indicates a deviation greater than plus or minus one, the channel may not be fully functional. If there are just a few deviations, everything is probably alright.

If your response to the slot number query was incorrect, or if the cables are not properly assembled, the display may look like the following photograph:



When you have finished looking at Monotonicity Charts, press E to exit the Self Test program, as indicated at the bottom of the screen.

Problems and Solutions

If either the Transfer Chart or Monotonicity Chart display indicates a problem with all channels, check the following:

1) The right slot number was given.

2) The cables are properly connected (see Appendix C).

If the display indicates that a channel may not be functioning properly, avoid using that channel until you can return the A/D + D/A board for service.

Adapting the Program

If you wish to, you can modify the Self Test program to test a particular sequence of channels by changing the source code. For your convenience, the source code of the Self Test program is included on the Demonstration Diskette. No guarantee can be made for the useability of any modified programs, but the software is simple and should present no major problems.

Chapter 3 Temp Test

The Temp Test program provides a method for measuring and displaying temperature with the A/D + D/A board. This program is intended to serve as a guide to using the A/D + D/A board for temperature sensing applications. You can customize this program to fit your own similar applications.

Purpose

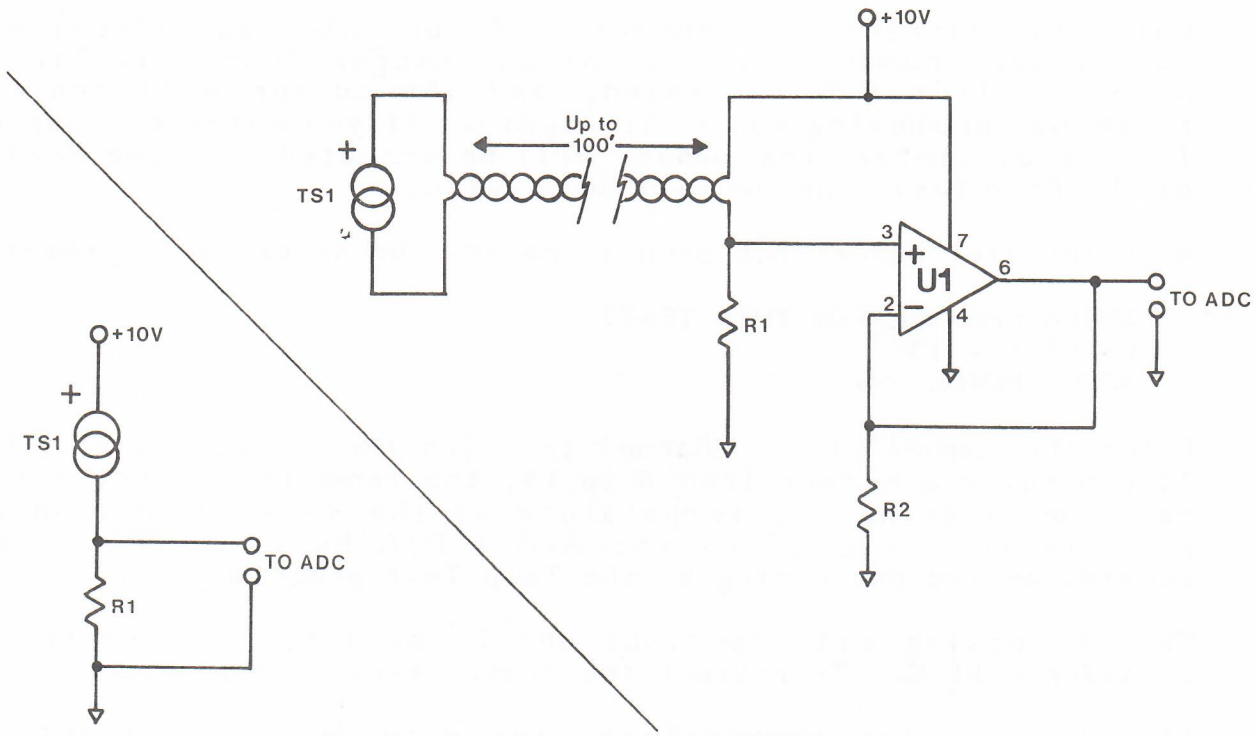
The Temp Test program provides a general purpose temperature sensing interface to the A/D + D/A board. It illustrates one of the many ways the board can be used to collect and compute analog data with an Apple II computer. For the sake of clarity and brevity, only the basic functions needed to demonstrate the use of A/D + D/A board as an analog collection and conversion device were included in the Temp Test program.

Introduction

Before running the Temp Test program, you must interface a simple temperature sensing device to the A/D + D/A board. A variety of temperature sensing devices are available. You must also provide a separate power source. One possible set-up will be discussed in this chapter. We have included manufacturer names and part numbers for the devices we recommend.

When the temperature sensing circuit has been attached to the A/D + D/A board, you will set up the interface between the Apple and the A/D + D/A board with the Temp Test program. When the interface has been set up, analog temperature data will be collected and converted to digital values. These values will be displayed on the video screen.

Schematics



Components

Following is a list of the components used in one version of the Temp Test demonstration. This component list is just one expample; you may choose your own components and create your own circuits.

ID #	Part# / Spec	Manufacturer / Source
R1	10K ohm 1/4W 5%	n/a
R2	1 k ohm 1/4W 5%	n/a
TS1	AD590	Analog Devices, Intersil
U1	NE535	Signetics (or other)
power supply	10 to 30 volts, DC	

Running the Program

Wire and assemble the circuit as described above. Then boot the A/D + D/A Demonstration Diskette. When the diskette catalog is displayed on the screen, type

RUN TEMP TEST

and press the RETURN key. The screen will clear, and you will be prompted for the number of the slot in which the A/D + D/A board is plugged.

Enter the correct slot number. If you enter an illegal slot number (any number that is not an integer from 1 to 7), the number will not be accepted, and the cursor will continue flashing, prompting you to try again. If you enter a wrong but legal slot number, the number will be accepted, but the results of the Temp Test program will be invalid.

When the slot number has been accepted, the screen will prompt

```
WHICH CHANNEL FOR THIS TEST?
ENTER 0 - 15
YOUR CHOICE -->
```

Enter the number of the channel to which the circuit is attached. If you enter a number from 0 to 15, the Temp Test circuit will begin monitoring the temperature at the sensor location and passing that signal to the A/D + D/A board for digital conversion and processing by the Temp Test program.

The processing will continue until you stop the program by pressing CTRL-C. To restart the test, re-run the program.

If you alter the temperature, the display will reflect the changes rapidly. Occasionally, the actual values displayed will fluctuate within plus or minus 3 degrees. This is to be expected with this type of circuit assembly and is within product specifications. More sophisticated circuitry will produce more exact conversions.

Problems and Solutions

If the display is very strange, for example, if the values change erratically, you may have entered the wrong slot or channel number at the beginning of the test. If so, the Temp Test program will run normally, but the values displayed will be unusable.

Here are some things to check for:

1) If the temperature display is not correct to within 3 degrees, check for the following:

- The correct slot and channel numbers were given.
- The circuit is properly assembled.
- The power supply hookup is correct, and the desired 10 to 30 volt output is constant.

If these conditions are met and the problem persists, check for faulty components.

2) If the Temp Test program indicates a failure on one channel, but runs properly on another channel, run the Self Test program to verify that the suspect channel is converting properly. Follow the Self Test procedures given in Chapter 2, The Self Test Program.

3) If no temperature display occurs even though the prompts were correctly answered,

- Check the power supply hookup, as above.
- Re-boot the Demonstration Diskette, and re-run the program. If the problem is still evident, there may be a serious problem with the Apple or its memory.
- Check for a bad channel. See problem 2, above.

Adapting the Program

You can modify the Temp Test program to test a series of sensors attached to a series of channels. This would provide an easy way of testing the comparative accuracy of several sensing devices. The Temp Test program can monitor up to sixteen environments or accept 16 circuits with no software changes.

The Temp Test program was designed to function as a base for more complicated programs. Unnecessary features were intentionally left out. For your convenience, the source code of the Temp Test program is included on the Demonstration Diskette.

NOTE: No guarantee can be made for the useability of any modified programs.

Chapter 4 Lite Test

This demonstration provides a model application for measuring and displaying relative light levels. It is not intended to serve as a "final and polished" application, but rather as a guide to using the A/D + D/A board for a particular purpose.

Purpose

The Lite Test program provides a general purpose light sensing interface to the A/D part of the A/D + D/A board. It illustrates one of the many ways the board can be used to collect and compute analog input with an Apple II computer. For the sake of clarity and brevity, no "bells and whistles" were added to the Lite Test program; just the basic functions needed to demonstrate the use of the A/D + D/A board as an analog collection and conversion device.

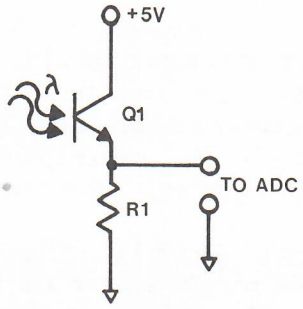
Introduction

Before running the Lite Test program, a simple hardware interface for sensing relative light levels must be attached to the A/D + D/A board. A separate power source must also be provided. A variety of light sensing devices are available. In this chapter we will discuss one possible set-up. Manufacturer names and part numbers have been included for the devices we recommend.

When the light sensing circuit has been built and attached to the A/D + D/A board, you will run the Lite Test program. The program will set up the interface between the Apple and the A/D + D/A board by means of a series of user prompts. When the promts have been answered, the program begins monitoring light levels and converting them to digital values. These values are displayed on the screen.

The relative light levels are indicated by a display of integers from 0 to 127. The actual light reading will depend on your choice of components. A reading of 0 indicates the absence of light. A reading of 127 is the upper limit that can be sensed by this particular circuit.

Schematic



Components

Following is a list of possible components for one Lite Test demonstration. This list is just an example. You may choose your own components and create your own circuits.

ID #	Part# / Spec	Manufacturer / Source
Q1	FPT100	Fairchild, Radio Shack #276-130
R1	1K ohm 1/4W 5%	n/a

NOTE: The power source may be obtained from the game I/O paddle connectors on the Apple mother board.

Running the Program

Wire and assemble a circuit as described above. Then boot the A/D + D/A Demonstration Diskette. When the diskette catalog is displayed, type

RUN LITE TEST

and press the RETURN key. The Lite Test program will announce itself and prompt you to enter the number of the slot in which the A/D + D/A board is plugged.

Enter the correct slot number. If you enter an illegal value (anything except an integer from 1 to 7), the number will not be accepted, and the cursor will continue flashing. If you enter a number that is legal, but the wrong slot number, the results of Lite Test program will not be valid.

When you have given an acceptable slot number, the screen will display

```
WHICH CHANNEL FOR THIS TEST?
ENTER 0 - 15
YOUR CHOICE -->
```

Enter the number of the channel to which you have attached the circuit. The Lite Test circuit will begin monitoring the light level at the sensor location and passing that signal to the A/D + D/A board for digital conversion and processing by the Lite Test program. The processing will continue until you stop the program by pressing CTRL-C. To restart the test, re-run the program.

If you alter the light level, the display will reflect the change rapidly. If you wish you can compare the results of the Lite Test program with the results given by a light meter. More sophisticated circuitry will produce more precise light level readings.

Problems and Solutions

A display that is completely wrong could indicate that you have entered the wrong slot or channel number at the beginning of the test. If so, the Lite Test program will run normally, but the displayed values will be useless.

1) If the display of light levels is erratic, check for the following problems:

- The right slot and channel number were given.
- The circuit is correctly assembled.
- The power supply hookup is correct, and the desired 5 volts are supplied.

If the above conditions are met, but the problem persists, check for faulty components.

2) If the Lite Test program indicates a failure on one channel, but another channel seems fine, run the Self Test program to check that the suspect channel is converting properly. Follow procedures for the Self Test program, described in Chapter 2, The Self Test Program.

3) If no light level display occurs, even though the prompts have all been answered,

- Check the power supply hookup, as described above.
- Re-boot the Demonstration Diskette, and re-run the program. If the problem persists, there may be a serious problem with your Apple or its memory.

- Check for a bad channel. See problem 2, above.

Adapting the Program

You can modify the Lite Test program to test a series of sensors attached to a series of channels. This would provide an easy way to test the comparative accuracy of several sensing devices. The Lite Test program can monitor up to sixteen environments or accept 16 circuits with no software changes.

The Lite Test program was designed to function as a possible base for more advanced programs. Unnecessary features were intentionally left out. For your convenience, the Lite Test program source code is included on the Demonstration Diskette.

NOTE: No guarantee can be made for the useability of any modified programs.

Chapter 5 Curve Tracer

This program lets you plot a transistor's characteristics on the Apple's high resolution graphics screen. The Curve Tracer is a complete and finished program that you can use as a tool for measuring and displaying transistor paramaters. This program also serves as an excellent example of the A/D and D/A parts of the A/D + D/A board working together in a single application.

Purpose

The purpose of the Curve Tracer program is to measure and display the characteristics of transistors. These characteristics are displayed as a graph of the voltage across the transisitor, versus the current through the transistor for a given base current.

In the Curve Tracer program, the A/D + D/A board is used to control part of the transistor curve tracer and to measure the actual parameters or characteristics of the transistor. Due to the nature of the test, the A/D + D/A board will both control the test and measure the test results.

Introduction

With the Curve Tracer program, you will use

- a DAC channel to control the current supplied to the transistor being tested. This is the base current, I_b.
- a DAC channel to control the the voltage applied across the transistor's collector-emitter junction. This is called V_{ce}.

You will also make measurements with two ADC circuits:

- An ADC to measure the actual voltage across the collector-emitter junction. The actual voltage is not always the same as the voltage that was set with the DAC (above). This voltage is called V_{ce}.
- An ADC to measure the amount of current flowing through the collector. Since the A/D + D/A board can only measure voltages applied to ADC channels, we will use a circuit to convert current to voltage. Measured current is called I_c.

Program Description

The Curve Tracer program is highly versatile and easy to use. In brief, with the Curve Tracer you can

- test NPN or PNP type transisitors with no wiring changes.

- run the Curve Tracer program in automatic or manual mode. In automatic mode, the computer generates 8 evenly spaced base currents (I_b)and plots them in succession. In manual mode, you select each of the base currents to be tested.

- display results in the format of a standard set of curves for a transistor.

- select a set of scales for each X or Y axis. Thus, with two choices of scale for each axis, 4 displays are available.

Following is a description of how the Curve Tracer demonstration will be done:

1) The first step is an analysis of the problem to be solved. This produces a series of tasks and questions.

2) What do we need to control with the program?
   - the current going to the base (I_b).
   - the voltage applied to collector emitter circuit (V_{ce}).

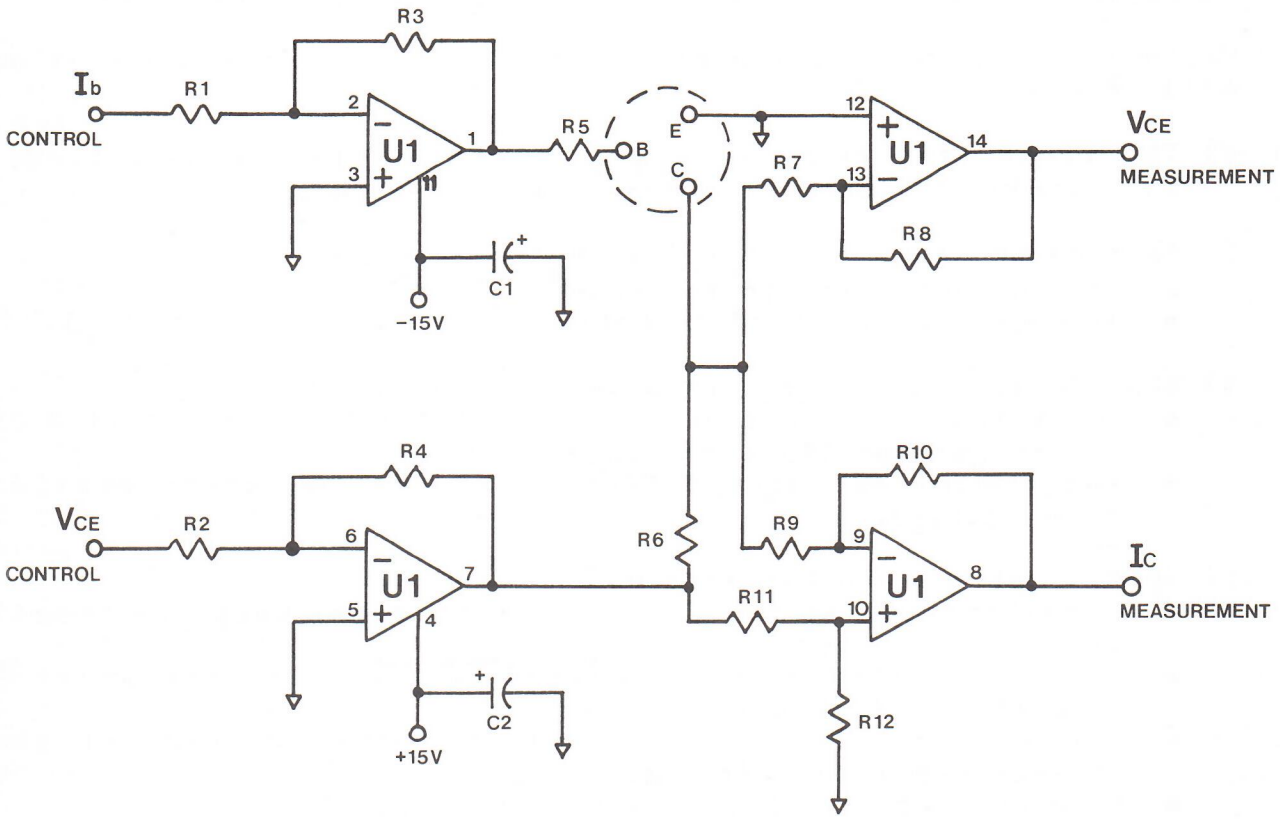
3) What do we need to measure with the program?
   - the actual voltage across the collector-emitter junction of the transistor (measured V_{ce}).
   - the amount of current flowing in the collector-emitter circuit (I_c).

4) Specifications and Limitations
   - limit testing to the class of transistors known as "small signal transistors".
   - maximum voltage to be applied to the collector-emitter circuit is 12.5 volts.
   - circuit applies a maximum of 30 mA of current for the collector-emitter circuit.
   - base current variable between 0 and 125 uA.
   - standard A/D + D/A board specifications.

5) Implementation Plan
   - create a circuit that meets the above specifications and interfaces with the DACs and the ADCs on the A/D + D/A board.

6) Reporting Plan
   - display the results on the high resolution graphics screen in an easily readable format.

Schematic



Theory of Operation

The schematic represents a solution to the problem of how to make a transistor curve tracer. This section discusses how the schematic works. A general working knowledge of electronics is assumed.

The A/D + D/A board outputs voltages in the range of -5 to +5 volts. The board also converts voltages in the same range to

digital values. Yet, in this example, we have specified that the maximum voltage applied to the transistor under test is 12.5 volts.

In this circuit, an op amp is used to increase the voltage from the A/D + D/A board from 0 to 5 volts up to the 0 to 12.5 volt range. Resistors R2 and R4, together with one of the op amps, form an amplifier stage with a gain of -2.5. The voltage appears on the output, pin 7 of the op amp. If the A/D + D/A board outputs +2 volts to this circuit via one of its DACs, the voltage on pin 7 will be +2 times -2.5 volts, or -5 volts. If the input to the circuit is -3.25 volts, then the output on pin 7 would be -3.25 times -2.5 volts, or 8.125 volts. Thus, the output of the A/D + D/A board may be adjusted to many different ranges.

The circuit made of resistors R1 and R3, together with an op amp, forms an amplifier with a gain of -2.5. The voltage output from this circuit is developed on pin 1 of the op amp. This voltage is fed through resistor R5 to the base of the transistor under test.

The known voltage from pin 1 fed to the resistor R5 will produce the base current, (I_b). This current I_b can be calculated by the computer by using Ohm's Law. The computer must know the value of the resistor R5, and the computer must know the voltage it is outputting on the DAC which controls the base current. This is how the computer controls the I_b variable.

The voltage on pin 7 of the op amp is controlled by the voltage applied to the V_{ce} control. In this way, the computer can control the other variable, the voltage applied across the collector-emitter junction. The voltage from pin 7 is fed through resistor R6 and then to the collector of the transistor under test. Now all that remains to be done is to take the measurements.

The measurements to be taken are the current flowing through the collector of the transistor I_c, and the actual voltage measured across the collector-emitter junction V_{ce}. The voltage applied across the transistor can be up to 12.5 volts, but the A/D + D/A board will only handle up to +5 volts.

The solution is to have an amplifier with a gain of less than one. Simply reduce a maximum of 12.5 volts to a maximum of 5 volts. We need an amplifier with a gain of 0.4. If a voltage of 12.5 is applied to the input, the output would be 12.5 times 0.4 volts, or 5 volts. The circuit made of resistors R7, R8 and an op amp make up an amplifier with a gain of -0.4. The output of this op amp appears on pin 14, and this voltage is applied to an ADC channel of the A/D + D/A board. Thus, we have a way to measure the V_{ce}.

To measure the current I_c, we again use Ohm's law. The voltage applied to the transistor goes through resistor R6. A circuit made up of resistors R9, R10, R11, and R12, and an op amp measure

the voltage drop across resistor R6. If the program knows the value of R6 and can measure the voltage drop across R6, then by using Ohm's law it can calculate the current flowing through R6 and hence the current flowing through the collector. This current is called I_C. The output of that circuit is a voltage on pin 8 of the op amp, and that voltage is applied directly to an ADC channel of the A/D + D/A board. The voltage on pin 8 is in proportion to the current I_C.

Several of the op amps are wired as inverters. That is, they have a negative gain. Because the A/D + D/A board has bipolar input and output, we simply supply a voltage of the correct polarity to the control circuits. When making measurements, the software makes the correct inversion of the voltage.

Components

ID #	Part# / Spec	Manufacturer / Source
C1,C2	10uF,35v	n/a
R1,R2,R8	300K ohm	n/a
R3,R4,R7	750K ohm	n/a
R5,R9,R10,R11,R12	100K ohm	n/a
R6	100 ohm	n/a
U1	LM324	National Semi-Conductor, TI, others
power supply	+15 to -15 volts	

In addition, be sure to attach the circuits as follows:

Circuit	Channel
I_b Control	DAC #14
V_{ce} Control	DAC #15
I_c Measurement	ADC #14
V_{ce} Measurement	ADC #15

NOTE: If you want to use other channels, you must change the Curve Tracer program source code.

Running the Program

Wire and assemble all circuits as specified in the schematic and the components list. When the circuits have been assembled, boot the A/D + D/A Demonstration Diskette. When the diskette catalog appears on the screen, type

RUN CURVE TRACER

The screen will clear and the following will be displayed:

MOUNTAIN COMPUTER

A/D + D/A
CURVE TRACER

WHICH SLOT IS THE A/D + D/A IN?

Enter the number of the slot in which the A/D + D/A board is installed. If you enter an illegal value (anything that is not an integer between 1 and 7), the number will be rejected, and the cursor will continue flashing, prompting you to try another number. If you enter a slot number that is legal but incorrect, the results of the Curve Tracer program will not be valid.

When the slot number has been accepted, the following display will appear:

```
WHAT TYPE OF TRANSISTOR ARE YOU TESTING?
A - NPN
B - PNP
YOUR CHOICE -->
```

Enter the appropriate type of transistor by typing A or B. You are then prompted

```
WHAT IS THE TRANSISTOR'S
PART NUMBER?
```

Enter any description, up to 7 characters or numbers. This information is displayed with the results.

At this point, you are prompted

```
YOU MAY SELECT ONE OF TWO
COLLECTOR CURRENT SCALES:
A - 20 MA MAXIMUM
B - 40 MA MAXIMUM
YOUR CHOICE -->
```

Enter either A or B, depending on your requirements. After you enter a letter, the screen clears and the next prompt appears at the top of a new screen:

```
WOULD YOU LIKE AN AUTOMATIC
TEST OR A MANUAL TEST?
A - AUTOMATIC TEST
B - MANUAL TEST
YOUR CHOICE -->
```

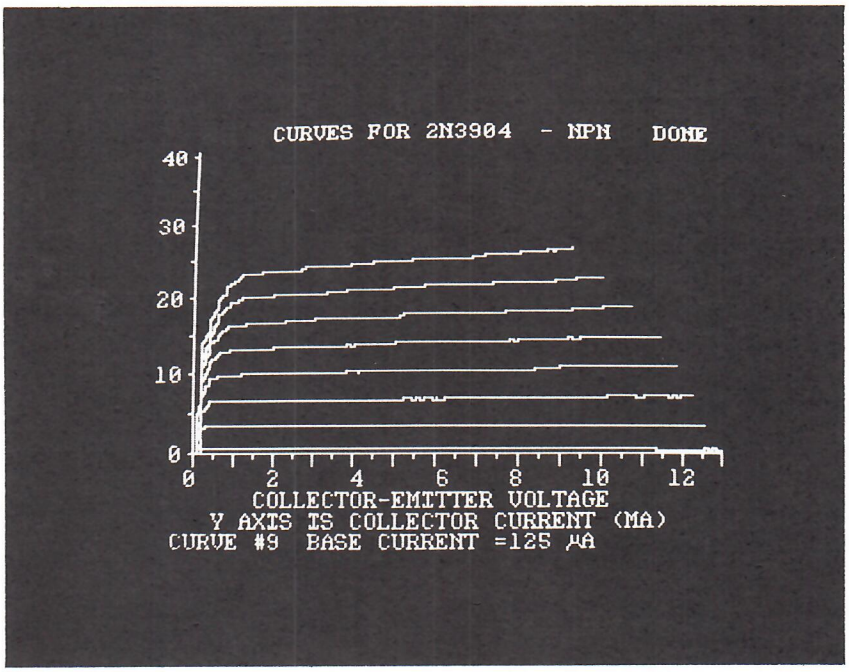
Enter A at this time. Later, we will explore Manual Mode. After pressing A, the next prompt appears:

```
YOU MAY SELECT ONE OF TWO SCALES FOR
THE COLLECTOR-EMITTER VOLTAGE:
A - 0 TO 13  VOLTS MAXIMUM
B - 0 TO 6.5 VOLTS MAXIMUM
YOUR CHOICE -->
```

Enter the appropriate response. As soon as you enter A or B, you are prompted

PRESS SPACE WHEN EVERYTHING IS READY.

This prompt allows you to get set for the display of the transistor's curves. The entire display takes about 4 minutes to calculate and plot eight transistor readings. When the display is complete, the screen will look something like this:



Note the following aspects of this particular display:

1. X Axis -- this is a scale of the collector-emitter voltage (V_{ce}), ranging from 0 to 12.5 volts.

2. Y Axis -- this is the collector current (I_C in MA).

3. Transistor Part Name -- as entered at the prompt.

4. Status Word -- either BUSY while calculating, PLOT while plotting the chart, or DONE when the program is finished.

5. CURVE # -- the sequential number of the curve being calculated or plotted.

6. BASE CURRENT -- the base current applied to the transistor for this particular curve.

When the status word is "DONE", the display stays on the screen until you press a key (any key except SHIFT, CTRL, or RESET). Then the screen will clear, and the Applesoft prompt will appear.

Manual Test Mode

If you choose Manual Test Mode, before the calculation of the transistor's curves begins, you are prompted

```
                MANUAL TEST MODE

   CURVE NUMBER 1
   SELECT A BASE CURRENT
   BETWEEN 0 and 125 UA

   YOUR CHOICE -->
```

Enter a number in the indicated, range and press the RETURN key. Note that the Base Current value displayed on the high resolution graphics screen is not always exactly the same as entered. This small deviation is within the product specification. It is actually caused by a combination of Applesoft's math and the Curve Tracer program itself.

When the display is complete, the status word on the top of the screen changes to

```
   MORE?
```

Pressing the space bar will cause the test to continue. Pressing any other key (except SHIFT, CTRL, or RESET) will cancel the test.

Problems and Solutions

A Curve Tracer display that is totally out of whack or a display with no curves could indicate one of several problems. With the problems listed below, it is entirely possible that the Curve Tracer program will run normally, but the curves displayed will be erratic. Try the following:

1) Check that the circuits are properly wired and assembled according to the schematic and components list.

2) Re-run the program, being sure that the slot number is correct.

3) Check that the power supply is ON and functioning properly.

4) Check that your response to the TYPE OF TRANSISTOR prompt (NPN or PNP) is correct for the transistor you are measuring.

Adapting the Program

The Curve Tracer program is complete in itself. However, if your needs are not completely served by this version of the Curve Tracer program, you may modify the software to change the hardware limitations or optimize the processing speed.

Chapter 6 A Reference

This chapter provides technical details of the A/D + D/A board. In this chapter you will find information on voltages, reading and writing to a channel, and conversion and scan times.

Voltage Ranges

The Mountain Computer A/D + D/A board will function to specification within a specific voltage input range. For applications requiring voltages outside of that range, you must append circuitry to put the voltage within the specified range. The section called "Theory of Operation" in Chapter 5, The Curve Tracer Program, includes a brief statement about changing voltages for input to the ADC and amplifying output voltages from the DAC.

Analog Input Voltage -- from -5 volts to +5 volts

Digital Value to Voltage Conversion

The ADC converts voltages from -5 to + 5 volts. The DAC converts digital values from 0 to 255 to analog voltages within this range. The A/D + D/A board is designed to function as follows:

- Absolute +/- 3% FSR (Full Scale Reading)
- Relative +/- 1 LSB (Least Significant Bit)

The digital values generated by the ADC and the analog voltages created by the DAC are subject to the above performance factors. Thus, there will sometimes be a slight discrepancy in the values and voltages produced. As long as the difference is within the specified range, the board is functioning properly.

A/D + D/A Conversion Chart

(Output) DAC	Number	(Input) ADC
-5.00 V	0	-5.00 V
	.	
	.	
-2.50 V	64	-2.50 V
	.	
	.	
-0.00 V	128	-0.00 V
	.	
	.	
+2.50 V	192	+2.50 V
	.	
	.	
+5.00 V	255	+5.00 V

When the voltage changes 39 mV, the ADC will produce a change of 1 digital value. Likewise, when the digital value being converted changes by 1, the voltage will change 39 mV.

How to Read or Output Voltage to a Channel

The A/D + D/A board is read from, or written to with software commands. The board is not language dependent; it may be controlled by any language used by your Apple. For this discussion, we will refer to procedures for reading from, or writing to, a channel using Applesoft BASIC.

The statement PEEK (n) (where n is an address) is used to read the value at the desired ADC channel. The statement POKE n,m (where n is an address, and m is a value to be written) is used to write the value to the desired DAC channel. The actual addresses must be one of 16 slot dependent locations. The address depends on which slot the board is in and which of the channels on the board is being referenced.

You must PEEK an address twice to get a reliable reading of that address. Each PEEK first returns the value produced by the previous ADC conversion, and then starts another conversion. You must PEEK that address again to get the desired value. Unless you are absolutely sure of the address of a previous PEEK, always do two successive PEEKS to get an ADC generated value.

The following Applesoft BASIC formula can be used to determine the address of each channel:

```
// ADDR = 49280 + (slot# * 16) + channel #
```

The following conditions must be met in the formula:

- 49280 ($C080 in hexadecimal) is the starting address of all slot dependent locations.
- The slot# must be an integer from 1 to 7.
- The channel # must be an integer from 0 to 15.

The following chart may be helpful in determining channel addresses. The addresses are identical whether you are using the ADC or the DAC.

Slot #	Range
1	$C090 - $C09F
2	$C0A0 - $C0AF
3	$C0B0 - $C0BF
4	$C0C0 - $C0CF
5	$C0D0 - $C0DF
6	$C0E0 - $C0EF
7	$C0F0 - $C0FF

The address that the ADC uses must be within the ranges indicated above and depends on which slot is used.

Conversion Times

Conversion of analog data to a digital value requires 9 microseconds. While it is possible to write a program that reads the ADC faster than 9 microseconds, the program will not be able to do anything useful with the samples it reads. If your program does anything with the data, it won't be reading a channel more often than once in 9 microseconds. The 9 microsecond limitation should not cause any programming inconveniences.

Scan Times on Reading Channels

Scan time to convert a value at a channel of the DAC is 16 microseconds. This is fast enough for most applications. However, if you are generating voltages from all or most channels of the DAC at a fast rate, special considerations must be understood and followed for effective performance.

Due to the nature of the DAC hardware, if you write a digital value to the same channel of a DAC more than once in 16 microseconds, the output from some of the other channels is subject to a possible "droop". By this, we mean that the channel may begin to drift away from the designated output voltage.

This "drooping effect" will not occur if only a few channels are being used. If most or all of the channels are being used, it can be avoided by not updating any faster than once every 16 microseconds. This is usually not a limitation since most programs do not read or write that fast.

Appendix A Specifications

The following specifications describe the performance of the Mountain Computer A/D + D/A board.

Analog to Digital Conversion

Features

- 16 input analog channels, software selectable via READ.
- Resolution rated at 8 binary bits (including sign bit).
- Conversion method -- Successive Approximation.
- Conversion begins on software READ command.

Specifications

- Input Impedance for
  AC = 1K ohm
  DC = 1 Meg ohm
- Analog Input Voltage range -- from -5 to + 5 volts.
- Channel conversion time = 9 microseconds.
- Accuracy
  Absolute = +/- 3% FSR
  Relative = +/- 1 LSB

Digital to Analog Conversion

Features

- 16 output analog channels, software selectable via WRITE.

Specifications

- Monontonic outputs.
- Analog output voltage = +/- 5 volts maximum.
- Output Current (source or sink) -- 2 mA.
- Accuracy
  Absolute = +/- 3% FSR
  Relative = +/- 1 LSB
- Output slew rate = 10V/ms.
- Dynamic output impedance = 10 ohms.

Appendix B
Programming Considerations

Subroutines

We have included a series of special subroutines for use with the A/D + D/A board. Each subroutine is written in 6502 machine language code and can be used with an Integer BASIC or Applesoft BASIC program to perform a specific function. These subroutines are

1. FASTSAMPLE -- takes between 1 and 256 samples from a specified ADC channel and stores them in a buffer in memory as quickly as possible.

2. WAVEFORM -- outputs a waveform on a specified DAC channel, after you set up a waveform table in memory.

3. CLKREAD -- uses the AppleClock to generate interrupts. Upon each interrupt, a sample is taken from the specified ADC channel and stored in a buffer.

4. CLKWAVE -- uses the AppleCLock to generate interrupts. Upon each interrupt, a sample is taken from a wave form table and output to the specified DAC channel.

Each of these subroutines is described in the following pages of this appendix. Since the programs were written in 6502 assembly language, we have provided the assembled object code for you to use. Additionally, we have included listings of the source code for each subroutine in this appendix.

Foreground and Background Processing

The A/D + D/A board can be used in conjunction with the Mountain Computer Apple Clock to create some exciting real time applications as well as provide automated background processing of analog input or output. You can easily create programs that sample analog data at a specified time or specified intervals, record the time at which the sample was taken, and calculate the moment when the next sample should be taken. Similarly, a program can output voltage to any electronic circuit for use in an application based on the data passed by the A/D + D/A board.

Similarly, with some clever programming, you can use the A/D + D/A board to collect or output data while you are using the Apple for other purposes. Of course, you will have to design and create you own applications based on your particular needs, but once your background applications with the A/D + D/A board are running, you need not stop any foreground use of the system for data collection or output.

The subroutines provided with the A/D + D/A Demonstration Diskette and described in this appendix are designed to help you create real time and foreground-background applications. The subroutines are handy programming examples for helping you develop your own similar programs.

The FAST SAMPLE Subroutine

FASTSAMPLE collects between one and 256 samples from a particular ADC channel. The program is written in 6502 machine language. Once this program starts reading the ADC channel, it will collect the samples and store them into an array as fast as it can. It collects about one sample per 16 microseconds. The location of the buffer that receives the samples and the number of samples to collect are passed to the subroutine as parameters. The ADC channel number is also passed via a parameter.

Using This Subroutine

This program is intended to serve as a model. It is fully functional. In normal usage, the parameters would be POKEd into memory by a BASIC program. After the parameters are POKEd, the subroutine must be called. When the subroutine returns, the data buffer will contain the requested number of samples from the indicated ADC channel.

The program is very simple. It will enter a loop which reads a sample from the ADC and stores the sample into a buffer. The index into the buffer is incremented and then compared with the number of samples to collect. When the program has collected the samples, it returns control to the caller.

Remarks and Comments

CAUTION - THIS PROGRAM USES SELF-MODIFYING CODE!!!

The parameters are as follows:

CHNL - This parameter specifies the ADC channel number to be used for collecting the samples. A program must POKE the parameter into memory before the subroutine is called. The value of the parameter is determined by the following formula:

POKED VALUE = (SLOT NUMBER * 16) + CHANNEL NUMBER

BUFFER - This parameter is a 16-bit address which specifies the beginning memory location of the buffer. Since it is a 16-bit address, there is no limitation as to the buffer's location. The memory location, BUFFER, holds the low order byte of the address; the location, BUFFER+1, holds the high order byte of the address.

COUNT - This parameter specifies the number of samples to be read from the ADC channel. The program always reads at least one sample. If the value of this parameter is 1, then one sample is read. If the value of this parameter is 212, then 212 samples are read. There is one exception and that is if the parameter's value is zero. In that case, a total of 256 samples are read into the buffer. The minimum number of samples is 1 and the maximum number of samples is 256.

The Listing

```
K80N
SOURCE FILE: FASTSAMPLE
SOURCE FILE: FASTSAMPLE
                1 ;*************************************************************
0000:           2 ;
0000:           3 ; FASTSAMPLE - VER1.0 - 25 JUNE 80 -
0000:           4 ; COPYRIGHT 1980 MOUNTAIN COMPUTER INC.
0000:           5 ;
0000:           6 ; FASTSAMPLE IS A SUBROUTINE WHICH IS USED TO READ UP TO 256
0000:           7 ; SAMPLES FROM A PARTICULAR A-D CHANNEL. THIS SUBROUTINE
0000:           8 ; ALWAYS READS AT LEAST ONE SAMPLE. THIS SUBROUTINE MAY BE
0000:           9 ; CALLED FROM A BASIC PROGRAM. BEFORE A CALL IS MADE, THE
0000:          10 ; BASIC PROGRAM MUST POKE FOUR VALUES INTO MEMORY. THESE
0000:          11 ; VALUES ARE THE PARAMETERS FOR THE FASTSAMPLE SUBROUTINE.
0000:          12 ; THE LOCATION OF THESE PARAMETERS WILL DEPEND ON THE
0000:          13 ; LOCATION OF THE FASTSAMPLE SUBROUTINE. THE PARAMETERS ARE:
0000:          14 ;
0000:          15 ; CHANNEL - THIS VALUE SPECIFIES WHICH A-D CHANNEL IS TO BE
0000:          16 ; READ. THE BASIC PROGRAM MUST POKE THE CHANNEL PARAMETER
0000:          17 ; WITH A VALUE X WHERE X IS DEFINED AS:
0000:          18 ; X = SLOT * 16 + CHANNEL NUMBER
0000:          19 ;
0000:          20 ; BUFFER - THIS IS AN ADDRESS WHICH SPECIFIES THE BEGINNING
0000:          21 ; OF A BUFFER IN MEMORY. THE BUFFER IS FILLED WITH THE
0000:          22 ; SAMPLES FROM THE A-D CHANNEL BEING READ. THIS IS A SIXTEEN
0000:          23 ; BIT ADDRESS, WITH LOCATION BUFFER HOLDING THE LOWER EIGHT
0000:          24 ; BITS OF THE ADDRESS, AND THE LOCATION BUFFER+1 HOLDING THE
0000:          25 ; UPPER EIGHT BITS OF THE ADDRESS. THE BASIC PROGRAM MUST
0000:          26 ; POKE THE CORRECT ADDRESS INTO THE BUFFER PARAMETER.
0000:          27 ;
0000:          28 ; COUNT - THIS PARAMETER CONTAINS THE NUMBER OF SAMPLES TO
0000:          29 ; BE READ. A COUNT OF 1 WILL READ ONE SAMPLE, A COUNT OF 20
0000:          30 ; WILL READ TWENTY SAMPLES, ETC. A COUNT OF 0 WILL READ THE
0000:          31 ; MAXIMUM OF 256 SAMPLES. THEREFORE, THIS SUBROUTINE WILL
0000:          32 ; READ FROM 1 TO 256 SAMPLES.
0000:          33 ;
0000:          34 ; CAUTION - THIS SUBROUTINE USES SELF-MODIFYING CODE!!
0000:          35 ;
               36 ;*************************************************************
```

```
                37 ********************
----- NEXT OBJECT FILE NAME IS FASTSAMPLE.OBJ0
4000:           38         ORG  $4000      ;
                39 ********************
4000:           40 ;
0100:           41 BUF     EQU  $0100      ;BOGUS 16 BIT VALUE
4000:           42 ;
4000:4C 07 40   43         JMP  START      ;
4003:           44 ;
4003:           45 ;
4003:           46 CHNL    DS   1          ;CHANNEL# PARAMETER
4004:           47 BUFFER  DS   2          ;BUFFER ADDRESS PARAMETER
4006:           48 COUNT   DS   1          ;COUNT PARAMETER
4007:           49 ;
4007:           50 ;
4007:78         51 START   SEI              ;DISABLE INTERRUPTS
4008:48         52         PHA              ;SAVE THE
4009:8A         53         TXA              ;A, X, AND Y
400A:48         54         PHA              ;REGISTERS
400B:98         55         TYA              ;ON THE
400C:48         56         PHA              ;STACK.
400D:AD 04 40   57         LDA  BUFFER      ;GET FIRST 8 BITS AND
4010:8D 2F 40   58         STA  PATCH+1     ;PATCH INTO CODE.
4013:AD 05 40   59         LDA  BUFFER+1    ;GET NEXT 8 BITS AND
4016:8D 30 40   60         STA  PATCH+2     ;PATCH IT ALSO.
4019:AD 06 40   61         LDA  COUNT       ;GET THE COUNT OF SAMPLES
401C:8D 33 40   62         STA  PATCHX+1    ;AND PATCH INTO CPY INSTRUCTION.
401F:AE 03 40   63         LDX  CHNL        ;PUT CHANNEL # INTO X
4022:BD 80 C0   64         LDA  $C080,X     ;BOGUS READ - THROW AWAY.
4025:A0 00      65         LDY  #0          ;ZERO OUT INDEX
4027:EA         66         NOP              ;WASTE TIME BEFORE
4028:EA         67         NOP              ;THE NEXT READ
4029:EA         68         NOP              ;
402A:EA         69         NOP              ;
402B:BD 80 C0   70 LOOP    LDA  $C080,X     ;READ THE A-D CHANNEL
402E:99 00 01   71 PATCH   STA  BUF,Y       ;STORE INTO BUFFER
4031:C8         72         INY              ;INCREMENT INDEX
4032:C0 00      73 PATCHX  CPY  #00         ;COMPARE Y TO COUNT
4034:D0 F5      74         BNE  LOOP        ;NOT EQUAL-READ AGAIN
4036:68         75         PLA              ;DONE READING, SO RESTORE
4037:A8         76         TAY              ;THE REGISTERS
4038:68         77         PLA              ;AND GET READY
4039:AA         78         TAX              ;TO RETURN TO
403A:68         79         PLA              ;BASIC.
403B:58         80         CLI              ;ENABLE THOSE INTERRUPTS!
403C:60         81         RTS              ;BYE!

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

The WAVEFORM Subroutine

WAVEFORM is a subroutine which outputs the contents of a waveform table through a DAC channel. This program is written in 6502 machine language. The program is useful as it is written, but its main purpose is to serve as a model program for the user. Hopefully the user can look at this program, and modify it to suit his needs.

Using This Subroutine

This program expects a waveform to be stored in a table which is an integeral multiple of 256 bytes long. The waveform table **must** begin on a page boundary. This program is called by a BASIC program or any other program. The WAVEFORM program **will not return control.** Once this program starts to execute, the only way to stop it is to hit the RESET key. The program uses locations $20 - $24 on page zero. The previous contents are not saved because you must hit RESET to exit this program.

Remarks and Comments

There are three parameters used in this program. The parameters are set via POKEs in a BASIC program. The exact location of the POKEs depends on where you have loaded the program. This program must be reassembled if the location is to change. Here are the parameters:

CHNL - This parameter specifies which DAC channel is to output the waveform. The value POKEd into this location is defined as:

VALUE TO POKE = (SLOT NUMBER * 16) + CHANNEL NUMBER

For example, with the A/D + D/A board in slot 5 and a channel number of 11, the value to be POKEd is $5B (hexidecimal).

TPAGE - This parameter specifies the starting address of the waveform table in memory. Since all waveform tables **must** begin on page boundaries, only one byte is needed to specify the address. This value is POKEd by the BASIC program.

TSIZE - This parameter specifies the length of the waveform table in pages. Each page is 256 bytes long. **If this value is set to zero, then all 64K of memory will be treated as a waveform table.**

An easy way to place a waveform table into memory is to BLOAD it into the computer, if it already exists. If the BASIC program is calculating the values to store into memory, then the BASIC program will have to POKE each location in the table. If the calculation needs to be done one time only, then it would be a good idea to BSAVE it to the disk. The user must always beware

of the memory locations used for the WAVEFORM program, and the waveform table, and adjust the BASIC's HIMEM so that there are no memory conflicts.

The rate at which samples are fed to the DAC channel depends on the execution speed of the 6502 machine language program. The program has a minor loop and a major loop. The program executes the minor loop to output one page of the waveform table. When 256 samples (one page) have been output, the program enters the major loop and sets the pointers to the next page of the waveform table. Therefore, after every 256 bytes output to the DAC channel, there will be a small (approximatly 17 microseconds) **additional** delay between samples. This will cause a slight distortion of the waveform, but it will not affect most applications.

The Listing

```
K8ON
SOURCE FILE: WAVEFORM
SOURCE FILE: WAVEFORM
                1 ;*****************************************************************
0000:           2 ;
0000:           3 ; WAVEFORM - VER1.0 - 26 JUNE 80 -
0000:           4 ; COPYRIGHT 1980 MOUNTAIN COMPUTER INC.
0000:           5 ;
0000:           6 ; WAVEFORM IS A 6502 PROGRAM WHICH GENERATES A WAVEFORM OF
0000:           7 ; ARBITRARY SHAPE ON ONE OF THE D-A CHANNELS. SOMEWHERE IN
0000:           8 ; MEMORY, THERE EXISTS A WAVE-TABLE WHICH CONTAINS THE DATA
0000:           9 ; POINTS WHICH DEFINE THE WAVEFORM. THIS PROGRAM TAKES AN
0000:          10 ; ENTRY FROM THE WAVE-TABLE AND OUTPUTS THAT VALUE TO THE
0000:          11 ; D-A CHANNEL SPECIFIED. A POINTER TO THE TABLE ENTRY IS
0000:          12 ; INCREMENTED AND THE OUTPUT CONTINUES. WHEN THE END OF THE
0000:          13 ; TABLE IS REACHED, OUTPUT CONTINUES WITH THE BEGINNING OF
0000:          14 ; THE TABLE. THIS PROGRAM NEVER RETURNS CONTROL. RESET IS
0000:          15 ; THE ONLY WAY TO EXIT!
0000:          16 ;
0000:          17 ; THE WAVE-TABLES' LENGTH MUST BE INTEGRAL MULTIPLES OF 256.
0000:          18 ; WAVE-TABLES MUST BEGIN ON PAGE BOUNDARIES. THE WAVE-TABLE
0000:          19 ; IS SETUP BY THE BASIC PROGRAM. THE BASIC PROGRAM MUST ALSO
0000:          20 ; POKE THE PARAMETERS INTO MEMORY. THE PARTICULAR LOCATIONS
0000:          21 ; DEPEND ON THE LOCATION OF THE PROGRAM. AFTER THE
0000:          22 ; WAVE-TABLE IS SETUP AND THE PARAMETERS POKED, THE BASIC
0000:          23 ; PROGRAM CALLS THIS PROGRAM. NOTE THAT THIS PROGRAM NEVER
0000:          24 ; RETURNS!! THE PARAMETERS ARE AS FOLLOWS:
0000:          25 ;
0000:          26 ; CHNL - THIS VALUE SPECIFIES WHICH D-A CHANNEL IS TO
0000:          27 ; OUTPUT THE WAVEFORM. THE BASIC PROGRAM MUST POKE THIS
0000:          28 ; PARAMETER WITH A VALUE X WHERE X IS DEFINED AS:
0000:          29 ; X = SLOT * 16 + CHANNEL NUMBER
0000:          30 ;
0000:          31 ; TPAGE - THIS PARAMETER SPECIFIES THE ADDRESS OF THE
0000:          32 ; BEGINNING OF THE WAVE-TABLE. SINCE ALL WAVE-TABLE BEGIN
0000:          33 ; ON PAGE BOUNDARIES, ONLY A PAGE NUMBER IN NECESSARY. THUS
0000:          34 ; TBLPAGE IS A SINGLE BYTE. THIS VALUE MUST BE POKED BY THE
0000:          35 ; BASIC PROGRAM.
0000:          36 ;
0000:          37 ; TSIZE - THIS PARAMETER SPECIFIES THE WAVE-TABLES'
0000:          38 ; LENGTH IN PAGES. IF THIS IS SET TO ZERO, THEN ALL 64K
0000:          39 ; WILL BE TREATED AS A WAVE-TABLE! THE BASIC PROGRAM MUST
0000:          40 ; POKE THIS PARAMETER WITH CAUTION (OR A NUMBER > 0).
0000:          41 ;
0000:          42 ; THIS PROGRAM USES PAGE ZERO LOCATIONS $20 - $24.
0000:          43 ; THEY ARE NOT SAVED, BUT DON'T WORRY.
0000:          44 ;
               45 ;*****************************************************************
```

```
0000:             46 ;
0000:             47 ;
001B:             48 TABLE  EQU  $1B        ;DEFINE THE
001D:             49 TBLLOC EQU  $1D        ;ZERO PAGE
001E:             50 TBLSIZ EQU  $1E        ;LOCATIONS
001F:             51 PAGCNT EQU  $1F        ;USED.
0000:             52 ;
0000:             53 ;
                  54 *********************
----- NEXT OBJECT FILE NAME IS WAVEFORM.OBJ0
4000:             55        ORG  $4000      ;
                  56 *********************
4000:             57 ;
4000:             58 ;
4000:4C 06 40     59        JMP  START      ;
4003:             60 ;
4003:             61 ;
4003:             62 CHNL   DS   1          ;CHANNEL # PARAMETER
4004:             63 TPAGE  DS   1          ;TABLE PAGE ADDRESS PARAMETER
4005:             64 TSIZE  DS   1          ;TABLE SIZE IN PAGES PARAMETER
4006:             65 ;
4006:             66 ;
4006:A9 00        67 START  LDA  #00        ;INITIALIZE THE
4008:85 1B        68        STA  TABLE      ;TABLE POINTER
400A:AD 04 40     69        LDA  TPAGE      ;
400D:85 1C        70        STA  TABLE+1    ;
400F:85 1D        71        STA  TBLLOC     ;
4011:AD 05 40     72        LDA  TSIZE      ;INITIALIZE PAGE COUNT
4014:85 1E        73        STA  TBLSIZ     ;
4016:AE 03 40     74        LDX  CHNL       ;PUT CHANNEL # INTO X
4019:A5 1E        75 MJRLP  LDA  TBLSIZ     ;GET # OF PAGES
401B:85 1F        76        STA  PAGCNT     ;AND STORE INTO COUNT
401D:A5 1D        77        LDA  TBLLOC     ;GET TABLE ADDRESS
401F:85 1C        78        STA  TABLE+1    ;AND STORE INTO POINTER
4021:A0 00        79        LDY  #$00       ;SET THE INDEX TO ZERO
4023:B1 1B        80 MINLP  LDA  (TABLE),Y  ;GET A TABLE VALUE
4025:9D 80 C0     81        STA  $C080,X    ;AND OUTPUT IT TO D-A
4028:C8           82        INY             ;INCREMENT INDEX
4029:D0 F8        83        BNE  MINLP      ;IF NOT ZERO, MINOR LOOP
402B:E6 1C        84        INC  TABLE+1    ;OTHERWISE INCREMENT PAGE # OF POINTER
402D:C6 1F        85        DEC  PAGCNT     ;DECREMENT PAGE COUNT
402F:D0 F2        86        BNE  MINLP      ;IF NOT ZERO, TABLE IS NOT FINISHED
4031:4C 19 40     87        JMP  MJRLP      ;OTHERWISE, START TABLE OVER.

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

The CLKREAD Subroutine

CLKREAD is an interrupt driven subroutine which samples a particular ADC channel once per interrupt, and stores the sample in a buffer. This program is useful for sampling an analog input at a period defined by the Apple Clock's interrupt rate. This program is written in 6502 machine language. It is a subroutine which may be called by a user's program, BASIC or otherwise.

Using This Subroutine

The CLKREAD program uses a buffer which is 512 bytes long. This buffer is divided into two logical buffers which are each 256 bytes in length. The two logical buffers are named BUFFER1 and BUFFER2. When the CLKREAD program starts to run, it will wait for an interrupt. When an interrupt occurs, the CLKREAD program will do two reads from the ADC channel. The first read is bogus and is thrown away. The second read is the valid read and the sample is stored into a temporary location. The CLKREAD program then checks to see if there is room in either of the two buffers. If both of the buffers are full, an error condition exists, and the sample that was just read is thrown away. If there is room in at least one of the buffers, the sample is stored into the buffer, and the buffer pointer is updated. If the sample that was just stored into a buffer causes that buffer to be full, then the buffer pointer is adjusted to point to the next buffer. Whenever a buffer becomes full, the STATUS byte is updated to indicate that a buffer is full. It is this same STATUS byte which is used to determine if both buffers are full. After all this stuff is figured out, the COUNT is decremented, and if it is zero then all the samples have been read and the program will disable the Apple Clock interrupts, clean things up and return.

The user's program must monitor the STATUS byte to determine when a buffer is filled. When a buffer becomes full, the user's program must do something with that data. After the data in the buffer has been processed, the user's program **must update the STATUS byte to indicate that the buffer is now empty.** If both buffers are full, then an error condition exists. Samples will be thrown away until the next buffer is emptied. The user's program must keep track of which buffer is to be emptied next. For example, suppose that both buffers have been filled. The CLKREAD program will then be ready to fill BUFFER1 next because it alternates between buffers. Now suppose that the user's program has **not** processed either of the buffers. If the user's program then sets the STATUS byte so that it indicates that BUFFER2 has been processed (which is the wrong order) then the CLKREAD program will begin to fill BUFFER1 and **not** BUFFER2. This is why it is important that the user process the buffers in the correct order.

The COUNT of the number of samples to read is a 16-bit quantity,

so that up to 65,536 samples may be read. The CLKREAD program **always reads at least one sample.** The BUFFER parameter specifies the starting page of the buffer. The buffer is two pages long and must start on a page boundary. The STATUS byte is used to indicate when a buffer is filled. This byte is used for communication between the CLKREAD program and the user's program. The MSB (bit 7) is set when BUFFER2 is filled, and bit 3 is set when BUFFER1 is filled. Because of the asynchronous nature of the communication between the user and the CLKREAD program, the user should use the special entry points provided in this program to clear the status of BUFFER1 or BUFFER2.

Remarks and Comments

CAUTION - THIS PROGRAM USES SELF MODIFYING CODE!!!

CAUTION - DOS COMMANDS MAY NOT BE USED WHILE INTERRUPTS ARE ENABLED!!!

Here are details on the parameters:

CHNL - This parameter specifies which ADC channel is used for the samples. The parameter is POKEd by a BASIC program before the routine is called. The value of the POKE is determined by this formula:

POKED VALUE = (SLOT NUMBER * 16) + CHANNEL NUMBER

BUFFER - This parameter specifies the starting page of the buffer. The buffer must begin on a page boundary, and is 512 bytes long.

COUNT - This parameter specifies the number of samples to be read. It is a 16-bit quantity, and up to 65,536 samples may be read. This parameter is forced to a value of one if it is zero. Therefore one sample is always read. This parameter may be PEEKed at to see how many samples remain to be read. The low order byte is at COUNT, and the high order byte is at COUNT+1.

STATUS - This byte is not a parameter, but a communication byte. Both the user's program and the CLKREAD program use this byte to control the buffers. The MSB, which is bit 7, is set to 1 when BUFFER2 is full and set to 0 when BUFFER2 is empty. Bit 3 is set to 1 when BUFFER1 is full and set to 0 when BUFFER1 is empty. When resetting a bit to zero, the other bit must not be changed. Special calls to clear the STATUS byte for either buffer have been provided. These calls are used to prevent the asynchronous programs from affecting each other.

In normal use, the user will set the parameters up in memory, and when the user is ready to begin sampling, a call is made to START. Then once each interrupt, a sample is taken until the buffers are both filled. If both buffers are full, samples are thrown away and the COUNT is not decremented. The user's program

will monitor the STATUS byte and process the buffers in the correct order. When the buffer has been processed, the user's program calls the appropriate STATUS buffer clear routine and then waits for the next buffer to fill. The whole process can be aborted early by calling the STOP entry point. This will disable the Apple Clock's interrupts and restore the interrupt vector. The COUNT parameter may be read to see how many samples remain to be read. The INTON and INTOFF calls may be used to temporarily disable the interrupts. This is useful if you must use DOS. Be aware that you can lose some samples if interrupts are disabled long enough.

The Listing

```
K80N
SOURCE FILE: CLKREAD
SOURCE FILE: CLKREAD
                    1 ************************************************************
0000:               2 ;
0000:               3 ; CLKREAD - VER1.0 - 2 JULY 80 -
0000:               4 ; COPYRIGHT 1980 MOUNTAIN COMPUTER INC.
0000:               5 ;
0000:               6 ; CHANGES--------------------
0000:               7 ; 22 AUGUST 80 - THE ENTRY POINTS CLRS1 AND CLRS2
0000:               8 ; WERE ADDED TO EASE USER INTERFACING.
0000:               9 ; CHANGES--------------------
0000:              10 ;
0000:              11 ; CLKREAD IS AN INTERRUPT DRIVEN PROGRAM WHICH WILL SAMPLE
0000:              12 ; A PARTICULAR A-D CHANNEL ONCE PER INTERRUPT. AN APPLE
0000:              13 ; CLOCK IS USED TO GENERATE THE INTERRUPTS. ONCE AN
0000:              14 ; INTERRUPT OCCURS, THE A-D CHANNEL SPECIFIED IS READ TWICE.
0000:              15 ; THE FIRST IS A BOGUS READ WHICH STARTS THE CONVERSION FOR
0000:              16 ; THE CHANNEL OF INTEREST. AFTER A BRIEF DELAY, THE CHANNEL
0000:              17 ; IS READ A SECOND TIME, AND THE SAMPLE IS STORED IN A
0000:              18 ; TEMPORARY LOCATION. NEXT, THE INTERRUPT HANDLER DETERMINES
0000:              19 ; IF BOTH 256 BYTE BUFFERS ARE FULL. IF SO, THE SAMPLE IS
0000:              20 ; THROWN AWAY AND THE INTERRUPT HANDLER RETURNS. TWO FULL
0000:              21 ; BUFFERS IS CONSIDERED AN ERROR CONDITION. IF THERE IS ROOM
0000:              22 ; IN THE BUFFERS, THE SAMPLE IS PLACED INTO THE BUFFER.
0000:              23 ; THE BUFFER POINTER IS THEN INCREMENTED, AND IF AT THIS
0000:              24 ; TIME A BUFFER BECOMES FULL, THE POINTER IS SET UP TO POINT
0000:              25 ; TO THE NEXT BUFFER. THE STATUS IS UPDATED TO INDICATE
0000:              26 ; WHICH BUFFER WAS JUST FILLED. NEXT, THE COUNT OF SAMPLES
0000:              27 ; TO BE READ IS DECREMENTED. IF THE LOW ORDER BYTE OF THIS
0000:              28 ; 16 BIT VALUE GOES TO ZERO, THEN THE HIGH ORDER BYTE IS
0000:              29 ; CHECKED FOR ZERO. IF THE HIGH ORDER BYTE IS ZERO, ALL OF
0000:              30 ; THE SAMPLES HAVE BEEN READ. OTHERWISE, THE HIGH ORDER BYTE
0000:              31 ; IS DECREMENTED. THE APPLE CLOCK IS THEN RESET FOR THE
0000:              32 ; NEXT INTERRUPT, THE REGISTERS ARE RESTORED AND THE
0000:              33 ; HANDLER RETURNS FROM THE INTERRUPT.
0000:              34 ;
0000:              35 ; CLKREAD CONTAINS AN ENTRY CALLED START WHICH SETS UP THE
0000:              36 ; PARAMETERS AND TURNS THE APPLE CLOCK INTERRUPTS ON.
0000:              37 ; IT ALSO CHECKS THE COUNT PARAMETER TO SEE IF IT IS ZERO.
0000:              38 ; IF SO, COUNT IS FORCED TO ONE. THEREFORE, ONE SAMPLE IS
0000:              39 ; ALWAYS READ. THIS SUBROUTINE ALSO SETS UP THE INTERRUPT
0000:              40 ; VECTOR AT LOCATION $3FE. THIS SUBROUTINE IS CALLED BY A
0000:              41 ; BASIC PROGRAM. THE PARAMETERS MUST BE POKED INTO MEMORY
0000:              42 ; BEFORE THE START SUBROUTINE IS CALLED. THESE ARE THE
0000:              43 ; PARAMETERS:
0000:              44 ;
0000:              45 ; CHNL - THIS PARAMETER SPECIFIES WHICH CHANNEL IS TO BE
0000:              46 ; READ. THE VALUE POKED INTO THE MEMORY LOCATION IS A
0000:              47 ; VALUE X SPECIFIED BY THIS FORMULA:
0000:              48 ; X = SLOT * 16 + CHANNEL NUMBER
0000:              49 ;
```

```
0000:            50 ; BUFFER - THIS PARAMETER SPECIFIES A PAGE NUMBER FOR THE
0000:            51 ; BUFFER. ALL BUFFERS MUST BEGIN ON A PAGE BOUNDARY AND
0000:            52 ; THEREFORE, THE BUFFER PARAMETER IS THE HIGH ORDER BYTE
0000:            53 ; OF THE BUFFER'S ADDRESS IN MEMORY. A BUFFER OF 512 BYTES
0000:            54 ; IS USED IN THIS PROGRAM. THE BUFFER IS SPLIT INTO TWO 256
0000:            55 ; BYTE BUFFERS, CALLED BUFFER1 AND BUFFER2. WHILE THE
0000:            56 ; CLKREAD READ PROGRAM FILLS ONE BUFFER, THE APPLICATION
0000:            57 ; PROGRAM CAN EMPTY THE OTHER BUFFER. IF BOTH BUFFERS
0000:            58 ; BECOME FULL, AN ERROR CONDITION EXISTS, AND ALL
0000:            59 ; SUBSEQUENT READS FROM THE CHANNEL ARE THROWN AWAY. IF A
0000:            60 ; READ IS THROWN AWAY, THE COUNT IS NOT DECREMENTED. IT IS
0000:            61 ; THE USERS RESPONSIBILITY TO EMPTY THE BUFFERS IN A TIMELY
0000:            62 ; MANNER AND TO KEEP TRACK OF WHICH BUFFER IS TO BE EMPTIED
0000:            63 ; NEXT. A STATUS OF THE BUFFERS IS OBTAINED BY PEEKING AT
0000:            64 ; PARTICULAR MEMORY LOCATIONS. THE ACTUAL LOCATION DEPENDS
0000:            65 ; ON THE ORIGIN ADDRESS OF CLKREAD.
0000:            66 ;
0000:            67 ; COUNT - THIS PARAMETER SPECIFIES HOW MANY SAMPLES ARE TO
0000:            68 ; READ FROM A CHANNEL INTO THE BUFFER. THE PARAMETER IS TWO
0000:            69 ; BYTES LONG, AND MAY BE READ (VIA PEEKS) TO DETERMINE HOW
0000:            70 ; MANY SAMPLES REMAIN TO BE READ. A COUNT OF ZERO IS NOT
0000:            71 ; ALLOWED, AND IF THIS PARAMETER IS SET TO ZERO, THE START
0000:            72 ; SUBROUTINE FORCES IT TO ONE. THERE MUST BE AT LEAST ONE
0000:            73 ; SAMPLE READ! IF BOTH BUFFERS BECOME FULL, SAMPLES WILL
0000:            74 ; BE THROWN AWAY AND THE COUNT WILL NOT BE DECREMENTED. AS
0000:            75 ; SAMPLES ARE PLACED INTO THE BUFFER, THE COUNT IS
0000:            76 ; DECREMENTED.
0000:            77 ;
0000:            78 ; STATUS - THIS IS NOT A PARAMETER, BUT IT IS A BYTE USED
0000:            79 ; COMMUNICATE TO THE BASIC PROGRAM THE STATUS OF THE TWO
0000:            80 ; BUFFERS. BIT NUMBER 3 IS SET TO A ONE WHEN BUFFER ONE
0000:            81 ; IS FILLED. BIT NUMBER 7 (MSB) IS SET TO A ONE WHEN
0000:            82 ; BUFFER TWO IS FULL. THE BASIC PROGRAM MUST POLL THE
0000:            83 ; STATUS BYTE AND UPON FINDING A FULL BUFFER, IT MUST
0000:            84 ; PROCESS IT. AFTER THE BASIC PROGRAM HAS PROCESSED THE
0000:            85 ; BUFFER, IT MUST SET THE CORRECT BIT TO ZERO. CAUTION -
0000:            86 ; THE OTHER BIT MUST NOT BE DISTURBED.
0000:            87 ;
0000:            88 ; CAUTION - THIS CODE IS SELF-MODIFYING!!
0000:            89 ;
0000:            90 ; GENERALLY, YOU MUST POKE THE MEMORY WITH THE
0000:            91 ; PARAMETERS AND THEN CALL START FROM BASIC. THE BASIC
0000:            92 ; SHOULD MONITOR THE STATUS LOCATION (PERHAPS WITH A WAIT)
0000:            93 ; AND WHEN A BUFFER BECOMES FULL, IT SHOULD PROCESS THE
0000:            94 ; DATA. WHEN THE CORRECT NUMBER OF SAMPLES HAVE BEEN READ,
0000:            95 ; THE CLKREAD PROGRAM WILL DEACTIVATE THE APPLE CLOCK
0000:            96 ; INTERRUPTS AND RESTORE THE INTERRUPT VECTOR TO $FF65.
0000:            97 ; THE SAMPLING MAY BE ABORTED BY CALLING THE STOP SUBROUTINE
0000:            98 ; AT ANY TIME. THE LOCATION COUNT AND COUNT+1 MAY BE PEEKED
0000:            99 ; AT ANY TIME TO DETERMINE THE NUMBER OF SAMPLES REMAINING
0000:           100 ; TO BE READ.
0000:           101 ;
0000:           102 ; CAUTION - DOS 3.2 COMMANDS CANNOT, I REPEAT, CANNOT BE
0000:           103 ; EXECUTED WHILE INTERRUPTS ARE ENABLED. FOR CONVENIENCE,
0000:           104 ; TWO SUBROUTINE CALLS HAVE BEEN INCLUDED WHICH TURN THE
0000:           105 ; INTERRUPTS ON OR OFF. CALL INTON OR INTOFF FROM BASIC
0000:           106 ; TO TURN INTERRUPTS ON OR OFF.
0000:           107 ;
0000:           108 ; ONE LAST NOTE - THIS PROGRAM ASSUMES THAT THE APPLE
0000:           109 ; CLOCK IS IN SLOT FOUR. IF THE CLOCK IS IN A DIFFERENT
0000:           110 ; SLOT, YOU NEED TO CHANGE THE "CLKSLT" EQUATE ONLY TO
0000:           111 ; THE CORRECT SLOT. ALL ADDRESSES ON THE CLOCK DEPEND ON
0000:           112 ; THIS EQUATE.
0000:           113 ;
                114 ************************************************************
                115 ********************
----- NEXT OBJECT FILE NAME IS CLKREAD.OBJ0
4000:           116         ORG  $4000     ;
                117 ********************
4000:           118 ;
4000:           119 ;
4000:           120 ; ENTRY POINT JUMPS
4000:           121 ;
4000:           122 ;
```

```
16384 4000:78          123 START  SEI              ;
      4001:4C 96 40    124        JMP  INIT       ;JUMP TO SET-UP ROUTINE
      4004:78          125 STOP   SEI              ;
      4005:4C D1 40    126        JMP  FINISH     ;JUMP TO DISABLE ROUTINE
      4008:4C 22 40    127 TICK   JMP  IHNDLR     ;JUMP TO INTERRUPT HANDLER
      400B:4C 07 41    128 INTON  JMP  ION        ;JUMP TO INTERRUPTS ON
      400E:4C 09 41    129 INTOFF JMP  IOFF       ;JUMP TO INTERRUPTS OFF
16401 4011:78          130 CLRS1  SEI              ;
      4012:4C 0B 41    131        JMP  CLR1       ;JUMP TO STATUS CLEAR 1
16405 4015:78          132 CLRS2  SEI              ;
      4016:4C 15 41    133        JMP  CLR2       ;JUMP TO STATUS CLEAR 2
      4019:            134 ;
      4019:            135 ;
      4019:            136 ; DATA STORAGE ALLOCATIONS
      4019:            137 ;
      4019:            138 ;
      4019:            139 ;
      4019:            140 ;PARAMETERS FIRST
      4019:            141 ;
16409 4019:            142 CHNL   DS   1          ;CHANNEL# PARAMETER
  410 401A:            143 BUFFER DS   1          ;BUFFER PAGE NUMBER
411/412 401B:          144 COUNT  DS   2          ;NUMBER OF SAMPLES TO READ
      401D:            145 ;
      401D:            146 ;
      401D:            147 ;INTERNAL VARIABLES NEXT
      401D:            148 ;
      401D:            149 ;
      401D:00          150 FLAG   DFB  $00        ;SET THE FLAG TO ZERO
16414 401E:00          151 STATUS DFB  $00        ;SET STATUS TO ZERO
      401F:00          152 BUFL   DFB  $00        ;BUFFER POINTER (LOW)
      4020:            153 BUFH   DS   1          ;BUFFER POINTER (HIGH)
      4021:            154 SAMPLE DS   1          ;A-D SAMPLE-TEMPORARY
      4022:            155 ;
      4022:            156 ;
      4022:            157 ;EQUATES
      4022:            158 ;
      4022:            159 ;
      0004:            160 CLKSLT EQU  4          ;APPLE CLOCK IN SLOT 4
      0049:            161 SETINT EQU  CLKSLT*16+9 ;INTERRUPT ENABLE ADDR.
      0047:            162 CLRIRQ EQU  CLKSLT*16+7 ;CLOCK IRQ LINE
      0048:            163 CLRINT EQU  CLKSLT*16+8 ;CLOCK INT-OUT LINE
      401B:            164 CNTL   EQU  COUNT      ;RENAME COUNT LOW
      401C:            165 CNTH   EQU  CNTL+1     ;RENAME COUNT HIGH
      4022:            166 ;
      4022:            167 ;
      4022:            168 ;INTERRUPT HANDLER
      4022:            169 ;
      4022:            170 ;
      4022:A5 45       171 IHNDLR LDA  $45        ;GET ACCUMULATOR
      4024:48          172        PHA             ;AND SAVE ON STACK
      4025:8A          173        TXA             ;GET X AND Y REGISTERS
      4026:48          174        PHA             ;AND SAVE
      4027:98          175        TYA             ;ON THE
      4028:48          176        PHA             ;STACK.
      4029:            177 ;
      4029:AE 19 40    178        LDX  CHNL       ;PUT CHANNEL# INTO X
      402C:BD 80 C0    179        LDA  $C080,X    ;DO A BOGUS READ
      402F:EA          180        NOP             ;AND
      4030:EA          181        NOP             ;WASTE
      4031:EA          182        NOP             ;ABOUT
      4032:EA          183        NOP             ;TEN
      4033:EA          184        NOP             ;MICROSECONDS.
      4034:            185 ;
```

```
4034:BD 80 C0  186          LDA  $C080,X    ;NOW GET THE SAMPLE
4037:8D 21 40  187          STA  SAMPLE     ;AND SAVE FOR LATER
403A:          188 ;
403A:AD 1E 40  189          LDA  STATUS     ;GET THE STATUS AND
403D:C9 88     190          CMP  #$88       ;IF BOTH BUFFERS ARE FULL
403F:F0 4F     191          BEQ  LEAVE      ;THROW AWAY SAMPLE.
4041:          192 ;
4041:AD 1F 40  193          LDA  BUFL       ;IF THERE IS ROOM IN THE
4044:8D 51 40  194          STA  PATCH+1    ;BUFFER, THEN PATCH THE
4047:AD 20 40  195          LDA  BUFH       ;ADDRESS AND
404A:8D 52 40  196          STA  PATCH+2    ;SAVE THE
404D:AD 21 40  197          LDA  SAMPLE     ;SAMPLE INTO
4050:8D 00 01  198 PATCH    STA  $0100      ;THE BUFFER.
4053:          199 ;
4053:EE 1F 40  200          INC  BUFL       ;INCREMENT BUFFER POINTER (LOW)
4056:D0 21     201          BNE  DECCNT     ;IF NOT ZERO, DECREMENT COUNT
4058:AD 1A 40  202          LDA  BUFFER     ;OTHERWISE, WHICH BUFFER IS FULL?
405B:CD 20 40  203          CMP  BUFH       ;IF BUFFER=BUFH, BUFFER 1 IS FULL.
405E:F0 0E     204          BEQ  BUF2       ;SO GET BUFFER 2 READY.
4060:          205 ;
4060:8D 20 40  206 BUF1     STA  BUFH       ;BUFFER 2 FULL-GET BUFFER 1 READY.
4063:AD 1E 40  207          LDA  STATUS     ;GET BUFFER STATUS
4066:09 80     208          ORA  #$80       ;AND SET BUFFER 2 FULL.
4068:8D 1E 40  209          STA  STATUS     ;RESTORE NEW STATUS.
406B:4C 79 40  210          JMP  DECCNT     ;NOW DECREMENT COUNT
406E:          211 ;
406E:EE 20 40  212 BUF2     INC  BUFH       ;BUFFER 1 FULL-GET BUFFER 2 READY.
4071:AD 1E 40  213          LDA  STATUS     ;GET BUFFER STATUS
4074:09 08     214          ORA  #$08       ;AND SET BUFFER 1 FULL.
4076:8D 1E 40  215          STA  STATUS     ;RESTORE NEW STATUS.
4079:          216 ;
4079:CE 1B 40  217 DECCNT   DEC  CNTL       ;DECREMENT COUNT LOW
407C:D0 08     218          BNE  RSTCLK     ;IF <> 0, THEN RESET CLOCK
407E:AD 1C 40  219          LDA  CNTH       ;OTHERWISE, CHECK COUNT HIGH
4081:F0 58     220          BEQ  EXIT       ;IF ZERO ALSO, EXIT!
4083:CE 1C 40  221          DEC  CNTH       ;OTHERWISE, DECREMENT COUNT HIGH
4086:          222 ;
4086:A2 47     223 RSTCLK   LDX  #CLRIRQ    ;
4088:BD 80 C0  224          LDA  $C080,X    ;THIS CLEARS CLOCK
408B:A2 48     225          LDX  #CLRINT    ;
408D:BD 80 C0  226          LDA  $C080,X    ;FOR NEXT INTERRUPT
4090:          227 ;
4090:68        228 LEAVE    PLA             ;ALL DONE SO
4091:A8        229          TAY             ;RESTORE THE REGISTERS
4092:68        230          PLA             ;
4093:AA        231          TAX             ;
4094:68        232          PLA             ;
4095:40        233          RTI             ;RETURN FROM INTERRUPT
4096:          234 ;
4096:          235 ;
4096:          236 ;INITIALIZE THINGS AT START
4096:          237 ;
4096:          238 ;
4096:48        239 INIT     PHA             ;SAVE THE REGISTERS
4097:8A        240          TXA             ;
4098:48        241          PHA             ;
4099:98        242          TYA             ;
409A:48        243          PHA             ;
409B:          244 ;
409B:AD 1C 40  245          LDA  CNTH       ;CHECK FOR COUNT
409E:D0 08     246          BNE  OKAY       ;OF ZERO. IF FOUND,
40A0:AD 1B 40  247          LDA  CNTL       ;FORCE COUNT TO
40A3:D0 03     248          BNE  OKAY       ;EQUAL ONE
```

```
40A5:EE 1B 40  249         INC  CNTL       ;COUNT EQUALS 1
40A8:          250 ;
40A8:AD 1A 40  251 OKAY    LDA  BUFFER     ;INITIALIZE BUFFER
40AB:8D 20 40  252         STA  BUFH       ;POINTER
40AE:A9 00     253         LDA  #$00       ;
40B0:8D 1F 40  254         STA  BUFL       ;
40B3:8D 1D 40  255         STA  FLAG       ;SET FLAG TO ZERO
40B6:8D 1E 40  256         STA  STATUS     ;SET STATUS TO EMPTY
40B9:          257 ;
40B9:A9 22     258         LDA  #>IHNDLR   ;SET UP THE
40BB:8D FE 03  259         STA  $3FE       ;INTERRUPT VECTOR
40BE:A9 40     260         LDA  #<IHNDLR ;TO POINT TO THE
40C0:8D FF 03  261         STA  $3FF       ;INTERRUPT ROUTINE
40C3:          262 ;
40C3:A9 01     263         LDA  #$01       ;TURN APPLE CLOCK
40C5:A2 49     264         LDX  #SETINT    ;
40C7:9D 80 C0  265         STA  $C080,X    ;INTERRUPTS ON.
40CA:          266 ;
40CA:68        267         PLA             ;NOW RESTORE THE
40CB:A8        268         TAY             ;REGISTERS AND
40CC:68        269         PLA             ;RETURN TO CALLING
40CD:AA        270         TAX             ;BASIC PROGRAM
40CE:68        271         PLA             ;
40CF:58        272         CLI             ;ENABLE THOSE INTERRUPTS!
40D0:60        273         RTS             ;AND RETURN
40D1:          274 ;
40D1:          275 ;
40D1:          276 ;FINISH - TURN OFF INTERRUPTS
40D1:          277 ;
40D1:          278 ;
40D1:48        279 FINISH PHA             ;SAVE THE REGISTERS
40D2:8A        280         TXA             ;
40D3:48        281         PHA             ;
40D4:98        282         TYA             ;
40D5:48        283         PHA             ;
40D6:          284 ;
40D6:A9 FF     285         LDA  #$FF       ;FLAG SAYS BASIC CALLED US
40D8:8D 1D 40  286         STA  FLAG       ;AND WE SHOULD RTS
40DB:          287 ;
40DB:A9 00     288 EXIT    LDA  #$00       ;TURN THE APPLE CLOCK
40DD:A2 49     289         LDX  #SETINT    ;
40DF:9D 80 C0  290         STA  $C080,X    ;INTERRUPTS OFF.
40E2:A2 47     291         LDX  #CLRIRQ    ;CLEAR THE CLOCK
40E4:BD 80 C0  292         LDA  $C080,X    ;JUST FOR
40E7:A2 48     293         LDX  #CLRINT    ;GOOD HOUSEKEEPING
40E9:BD 80 C0  294         LDA  $C080,X    ;
40EC:          295 ;
40EC:A9 65     296         LDA  #$65       ;NOW RESTORE THE
40EE:8D FE 03  297         STA  $3FE       ;ORIGINAL INTERRUPT
40F1:A9 FF     298         LDA  #$FF       ;VECTOR
40F3:8D FF 03  299         STA  $3FF       ;WHICH IS A RESET
40F6:          300 ;
40F6:AD 1D 40  301         LDA  FLAG       ;WHERE DID I COME FROM?
40F9:F0 95     302         BEQ  LEAVE      ;IF ZERO, I AM AN INTERRUPT
40FB:A9 00     303         LDA  #$00       ;OTHERWISE, I AM A BASIC CALL &
40FD:8D 1D 40  304         STA  FLAG       ;I SHOULD RESET THE FLAG
4100:68        305         PLA             ;EVERYTHING IS OKAY,
4101:A8        306         TAY             ;SO RESTORE THE
4102:68        307         PLA             ;REGISTERS
4103:AA        308         TAX             ;
4104:68        309         PLA             ;
4105:58        310         CLI             ;ENABLE INTERRUPTS &
4106:60        311         RTS             ;RETURN.
```

```
4107:            312 ;
4107:            313 ;
4107:            314 ;INTON ROUTINE
4107:            315 ;
4107:            316 ;
4107:58          317 ION     CLI             ;ENABLE INTERRUPTS
4108:60          318         RTS             ;AND RETURN
4109:            319 ;
4109:            320 ;
4109:            321 ;INTOFF ROUTINE
4109:            322 ;
4109:            323 ;
4109:78          324 IOFF    SEI             ;DISABLE INTERRUPTS
410A:60          325         RTS             ;AND RETURN
410B:            326 ;
410B:            327 ;
410B:            328 ; CLEAR STATUS FOR BUFFER 1
410B:            329 ;
410B:            330 ;
410B:AD 1E 40    331 CLR1    LDA  STATUS     ;GET STATUS BYTE
410E:29 80       332         AND  #$80       ;CLEAR ALL BITS EXCEPT 7
4110:8D 1E 40    333         STA  STATUS     ;RESTORE STATUS
4113:58          334         CLI             ;ENABLE INTERRUPTS
4114:60          335         RTS             ;RETURN
4115:            336 ;
4115:            337 ;
4115:            338 ; CLEAR STATUS FOR BUFFER 2
4115:            339 ;
4115:            340 ;
4115:AD 1E 40    341 CLR2    LDA  STATUS     ;GET STATUS BYTE
4118:29 08       342         AND  #$08       ;CLEAR ALL BITS EXCEPT 3
411A:8D 1E 40    343         STA  STATUS     ;RESTORE STATUS
411D:58          344         CLI             ;ENABLE INTERRUPTS
16670 411E:60    345         RTS             ;RETURN

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

The CLKWAVE Subroutine

CLKWAVE is a subroutine which may be called by other programs. It is written in 6502 machine language. CLKWAVE is an interrupt driven program which will output one sample from the waveform table for each interrupt. This program was written to use the Apple Clock to generate the interrupts. The sample is output to a particular DAC channel. The main part of the program is the interrupt handler. There are also entry points so that the BASIC program may control the operation of the CLKWAVE program.

Using This Subroutine

The waveform table must be an integeral multiple of 256 bytes in length. The waveform table must be at least one page long. There is a parameter which controls the size of the increment of the pointer into the waveform table. This is the same as a crude frequency control. During normal operation, an interrupt occurs and control is passed to the interrupt handler. The interrupt handler is called, and it fetches the value in the waveform table that is pointed to by the pointer. The value is sent to the specified DAC channel. The pointer into the waveform table is then incremented by the value of the increment parameter so that it will point to the next value to be fetched from the table. This pointer value is then checked against the end of the table, and if the pointer is past the end of the table, the pointer will wrap-around to the beginning of the table. (The pointer value is the same as MOD table length.) The interrupt handler then returns.

There are a few other entry points into the code. One entry will enable interrupts for the whole Apple. Another entry will disable interrupts for the Apple. Together these may be used to turn interrupts ON and OFF momentarily. An example of their use would be if you wanted to write something out to the disk. In that case you must disable interrupts while doing the disk write. After the disk write was finished, you would re-enable the interrupts.

There is an initial entry point called START which checks the parameters for legal values, sets the Apple Clock up for generating interrupts, and stores the interrupt handler address at the interrupt vector. The parameters must have been POKEd into memory before this routine is called. Another entry point called STOP will disable interrupts, and reset the Apple Clock, and restore the interrupt vector to the monitor's RESET routine.

Remarks and Comments

CAUTION - THIS PROGRAM USES SELF-MODIFYING CODE!!!

CAUTION - DOS 3.2 CANNOT BE USED WHILE INTERRUPTS ARE ENABLED!!! Because of the way that DOS 3.2 was written, **interrupts may not be used.** This caution applies to all earlier DOS versions.

Here is a breakdown on the parameters:

CHNL - This parameter specifies which DAC channel will be used to output the waveform. Like all parameters, it is POKEd into memory before the subroutine is called. The value is determined by this formula:

POKED VALUE = (SLOT NUMBER * 16) + CHANNEL NUMBER

WAVEPG - This parameter specifies the starting page address of the waveform table. All waveform tables must be an integeral multiple of 256 bytes long, and all waveform tables must be at least one page long.

WAVESZ - This parameter specifies the size of the waveform table in pages. If this parameter is POKEd with a value of zero, the START subroutine will force it to a value of one. Therefore, all waveform tables **must** be at least one page long.

INCR - This parameter specifies the increment for the pointer into the waveform table. Changing this parameter will change the frequency of the waveform as it appears on the output. This parameter must be set to a reasonable value, i.e., between 1 and 127.

The parameters are POKEd into memory and the START subroutine is called. At that point, the program will output a sample on every interrupt and continue to do so until the STOP subroutine is called. When the STOP subroutine is called, the program cleans up the environment and returns. The operation of the program may be temporarily suspended by use of the INTON and INTOFF entry points.

The location of the entry points and the parameters will depend on where the program is assembled.

The Listing

```
K80N
SOURCE FILE: CLKWAVE
SOURCE FILE: CLKWAVE
                    1 ****************************************************************
0000:               2 ;
0000:               3 ; CLKWAVE - VER1.1 - 3 JULY 80 -
0000:               4 ; COPYRIGHT 1980 MOUNTAIN COMPUTER INC.
0000:               5 ;
0000:               6 ; CHANGES-----------------------
0000:               7 ; 11 JULY 80 - INCR PARAMETER ADDED. CODE FOR INCREMENTING
0000:               8 ; THE WAVE TABLE POINTER BY A PARAMETER'S VALUE WAS ADDED.
0000:               9 ; THIS IS TO ALLOW A FREQUENCY CONTROL TO THE WAVE.
0000:              10 ;
0000:              11 ; 11 JULY 80 - WPTRL & WPTRH ARE NO LONGER SEPERATE
0000:              12 ; VARIABLES. THEY ARE NOW STORED AT THE PATCH LOCATIONS.
0000:              13 ; THIS IS FOR THE MICROSECOND NIGGLER. YOW! I SAVED 16US!
0000:              14 ; -----------------------------
0000:              15 ;
0000:              16 ; CLKWAVE IS AN INTERRUPT DRIVEN PROGRAM WHICH WILL OUTPUT
0000:              17 ; A SAMPLE TO A PARTICULAR D-A CHANNEL ONCE PER INTERRUPT.
0000:              18 ; AN APPLE CLOCK IS USED TO GENERATE THE INTERRUPTS. WHEN
0000:              19 ; AN INTERRUPT OCCURS, THE SAMPLE IN THE WAVEFORM TABLE
0000:              20 ; IS FETCHED AND OUTPUT TO THE PARTICULAR D-A CHANNEL. A
0000:              21 ; POINTER KEEPS TRACK OF THE NEXT SAMPLE IN THE WAVEFORM
0000:              22 ; TABLE. AFTER THE SAMPLE IS OUTPUT, THE WAVE POINTER IS
0000:              23 ; INCREMENTED. WHEN THE POINTER LOW IS ZERO, THE POINTER
0000:              24 ; HIGH IS INCREMENTED. A CHECK OF THE NEW POINTER AGAINST
0000:              25 ; THE END OF THE WAVE TABLE IS MADE. IF THE POINTER IS
0000:              26 ; PAST THE END OF THE TABLE, IT IS RESET TO THE BEGINNING
0000:              27 ; OF THE TABLE. ONCE THE POINTER IS READY FOR THE NEXT
0000:              28 ; OUTPUT, THE INTERRUPT HANDLER RESETS THE APPLE CLOCK
0000:              29 ; AND RETURNS FROM THE INTERRUPT.
0000:              30 ;
0000:              31 ; CLKWAVE CONTAINS A SUBROUTINE CALLED START WHICH SETS
0000:              32 ; UP THE VARIABLES AND PROCESSES THE PARAMETERS. A CHECK
0000:              33 ; OF THE VALUE OF WAVESZ IS MADE, AND IF IT IS ZERO, IT
0000:              34 ; IS FORCED TO ONE. ALL WAVE TABLES MUST BE AT LEAST ONE
0000:              35 ; PAGE LONG!! THE START SUBROUTINE ALSO TURNS THE APPLE
0000:              36 ; CLOCK'S INTERRUPTS ON, SETS THE INTERRUPT VECTOR TO
0000:              37 ; POINT TO THE INTERRUPT HANDLER, ENABLES INTERRUPTS AND
0000:              38 ; RETURNS TO THE CALLING PROGRAM. BEFORE START IS CALLED,
0000:              39 ; THE BASIC PROGRAM MUST POKE THE PARAMETERS INTO MEMORY.
0000:              40 ; THE LOCATION OF THE PARAMETERS DEPENDS ON THE LOCATION
0000:              41 ; OF THIS PROGRAM IN MEMORY. MORE ON THE PARAMETERS LATER.
0000:              42 ;
0000:              43 ; CLKWAVE CONTAINS A SUBROUTINE CALLED STOP WHICH IS USED
0000:              44 ; TO ABORT THE OUTPUT OF THE CLKWAVE PROGRAM. THE CLKWAVE
0000:              45 ; PROGRAM WILL CONTINUE LOOPING THROUGH THE WAVE TABLE
0000:              46 ; UNTIL THIS SUBROUTINE IS CALLED. THIS SUBROUTINE DISABLES
0000:              47 ; THE APPLE CLOCK INTERRUPTS AND CHANGES THE INTERRUPT
0000:              48 ; VECTOR BACK TO $FF65. INTERRUPTS ARE ENABLED ON THE CPU,
0000:              49 ; AND THE SUBROUTINE RETURNS.
0000:              50 ;
0000:              51 ; TWO ADDITIONAL SUBROUTINES ARE PROVIDED FOR CONVENIENCE.
0000:              52 ; INTON ENABLES INTERRUPTS AND INTOFF DISABLES INTERRUPTS.
0000:              53 ;
0000:              54 ; THE PARAMETERS ARE:
0000:              55 ;
0000:              56 ; CHNL - THIS PARAMETER SPECIFIES WHICH CHANNEL IS TO BE
0000:              57 ; USED FOR OUTPUT. THE VALUE POKED INTO THE MEMORY
0000:              58 ; LOCATION IS A VALUE X SPECIFIED BY THIS FORMULA:
0000:              59 ; X = SLOT * 16 + CHANNEL NUMBER
0000:              60 ;
```

```
0000:             61 ; WAVEPG - THIS PARAMETER SPECIFIES THE PAGE NUMBER OF
0000:             62 ; THE BEGINNING OF THE WAVE TABLE. ALL BUFFERS MUST BE
0000:             63 ; INTEGRAL MULTIPLES OF 256 BYTES. THE WAVE TABLE MUST
0000:             64 ; BE AT LEAST ONE PAGE LONG. FOR EXAMPLE, IF THE WAVE
0000:             65 ; TABLE STARTS AT $8000, THEN THE VALUE POKED INTO THIS
0000:             66 ; PARAMETER IS $80 ( OR 128 ).
0000:             67 ;
0000:             68 ; WAVESZ - THIS PARAMETER SPECIFIES THE LENGTH OF THE
0000:             69 ; WAVE TABLE IN PAGES. ALL WAVE TABLES MUST BE AT LEAST
0000:             70 ; ONE PAGE LONG.
0000:             71 ;
0000:             72 ; INCR - THIS PARAMETER SPECIFIES THE INCREMENT FOR THE
0000:             73 ; POINTER USED TO STEP THRU THE WAVE TABLE. BY CHANGING
0000:             74 ; THIS PARAMETER (WITHIN REASON) THE FREQUENCY MAY BE
0000:             75 ; CONTROLLED. TYPICAL VALUES ARE 1 THRU 127.
0000:             76 ;
0000:             77 ;
0000:             78 ; CAUTION - THIS PROGRAM USES SELF-MODIFYING CODE!!
0000:             79 ;
0000:             80 ;
0000:             81 ; CAUTION - DOS 3.2 COMMANDS CANNOT, I REPEAT, CANNOT BE
0000:             82 ; EXECUTED WHILE INTERRUPTS ARE ENABLED. THE RISK OF
0000:             83 ; MUNGING THE DISK IS GREAT! FOR YOUR CONVENIENCE, INTON
0000:             84 ; AND INTOFF WILL ENABLE AND DISENABLE INTERRUPTS. WHILE
0000:             85 ; INTERRUPTS ARE DISABLED, THE OUTPUT ON THE D-A CHANNEL
0000:             86 ; WILL NOT CHANGE.
0000:             87 ;
0000:             88 ; GENERALLY SPEAKING, THE BASIC PROGRAM MUST POKE THE
0000:             89 ; CORRECT PARAMETERS INTO MEMORY AND THEN CALL THE START
0000:             90 ; SUBROUTINE TO INITIALIZE THINGS. OUTPUT CONTINUES FROM
0000:             91 ; THE D-A CHANNEL UNTIL THE STOP SUBROUTINE IS CALLED.
0000:             92 ;
                  93 ************************************************************
                  94 ********************
----- NEXT OBJECT FILE NAME IS CLKWAVE.OBJ0
4000:             95         ORG  $4000     ;
                  96 ********************
4000:             97 ;
4000:             98 ;
4000:             99 ;ENTRY POINT JUMPS
4000:            100 ;
4000:            101 ;
4000:78          102 START  SEI              ;DISABLE INTERRUPTS &
4001:4C 4F 40    103        JMP  INIT       ;JUMP TO INITIALIZATION
4004:78          104 STOP   SEI              ;DISABLE INTERRUPTS &
4005:4C 93 40    105        JMP  FINISH     ;JUMP TO PROGRAM EXIT ROUTINE
4008:4C BA 40    106 INTON  JMP  ION        ;JUMP TO INTERRUPT ENABLE
400B:4C BC 40    107 INTOFF JMP  IOFF       ;JUMP TO INTERRUPT DISABLE
400E:            108 ;
400E:            109 ;
400E:            110 ;DATA STORAGE ALLOCATION
400E:            111 ;
400E:            112 ;
400E:            113 ;PARAMETERS FIRST
400E:            114 ;
400E:            115 CHNL   DS   1          ;CHANNEL# PARAMETER
400F:            116 WAVEPG DS   1          ;WAVE TABLE STARTING PAGE
4010:            117 WAVESZ DS   1          ;WAVE TABLE SIZE IN PAGES
4011:            118 INCR   DS   1          ;WAVE TABLE INCREMENT (FREQUENCY)
4012:            119 ;
4012:            120 ;
4012:            121 ;INTERNAL VARIABLES NEXT
4012:            122 ;
```

```
4012:           123 ;
4012:           124 WAVLIM DS   1           ;WAVE TABLE LIMIT
4013:           125 ;
4013:           126 ;
4013:           127 ;EQUATES
4013:           128 ;
4013:           129 ;
0004:           130 CLKSLT EQU  4           ;APPLE CLOCK IN SLOT 4
0049:           131 SETINT EQU  CLKSLT*16+9 ;INTERRUPT ENABLE ADDRESS
0047:           132 CLRIRQ EQU  CLKSLT*16+7 ;CLOCK IRQ LINE
0048:           133 CLRINT EQU  CLKSLT*16+8 ;CLOCK INT LINE
03FE:           134 IRQVEC EQU  $03FE       ;INTERRUPT VECTOR
4013:           135 ;
4013:           136 ;
4013:           137 ;INTERRUPT HANDLER
4013:           138 ;
4013:           139 ;
4013:A5 45      140 IHNDLR LDA  $45         ;GET THE ACCUMULATOR
4015:48         141        PHA              ;AND SAVE ON THE STACK
4016:8A         142        TXA              ;GET THE X AND Y REGISTERS
4017:48         143        PHA              ;AND
4018:98         144        TYA              ;SAVE
4019:48         145        PHA              ;ON THE STACK
401A:           146 ;
401A:AE 0E 40   147        LDX  CHNL        ;LOAD CHANNEL NUMBER INTO X
401D:AD 00 01   148 PATCH  LDA  $0100       ;GET SAMPLE FROM WAVE TABLE
4020:9D 80 C0   149        STA  $C080,X     ;AND OUTPUT SAMPLE TO D-A
4023:           150 ;
4023:18         151        CLC              ;CLEAR THE CARRY
4024:AD 1E 40   152        LDA  WPTRL       ;GET THE POINTER LOW
4027:6D 11 40   153        ADC  INCR        ;ADD IN THE INCREMENT
402A:8D 1E 40   154        STA  WPTRL       ;AND STORE IN POINTER LOW
402D:90 10      155        BCC  RSTCLK      ;IF CARRY NOT SET THEN RESET CLOCK
402F:           156 ;
402F:AD 1F 40   157        LDA  WPTRH       ;GET THE POINTER HIGH
4032:69 00      158        ADC  #$00        ;AND ADD THE CARRY
4034:CD 12 40   159        CMP  WAVLIM      ;COMPARE TO END OF TABLE
4037:D0 06      160        BNE  RSTCLK      ;IF NOT END OF TABLE, RESET CLOCK
4039:           161 ;
4039:AD 0F 40   162        LDA  WAVEPG      ;YES - MOVE POINTER TO
403C:8D 1F 40   163        STA  WPTRH       ;BEGINNING OF WAVE TABLE
403F:           164 ;
403F:A2 47      165 RSTCLK LDX  #CLRIRQ     ;RESET THE APPLE
4041:BD 80 C0   166        LDA  $C080,X     ;FOR THE NEXT
4044:A2 48      167        LDX  #CLRINT     ;INTERRUPT
4046:BD 80 C0   168        LDA  $C080,X     ;
4049:           169 ;
4049:68         170        PLA              ;NOW RESTORE
404A:A8         171        TAY              ;THE REGISTERS
404B:68         172        PLA              ;THAT WE SAVED
404C:AA         173        TAX              ;
404D:68         174        PLA              ;
404E:40         175        RTI              ;AND RETURN FROM INTERRUPT
404F:           176 ;
404F:           177 ;
404F:           178 ;INITIALIZATION SUBROUTINE
404F:           179 ;
404F:           180 ;
404F:48         181 INIT   PHA              ;SAVE THE
4050:8A         182        TXA              ;REGISTERS FOR
4051:48         183        PHA              ;A RAINY DAY
4052:98         184        TYA              ;
4053:48         185        PHA              ;
```

```
4054:           186 ;
4054:AD 10 40   187         LDA  WAVESZ     ;FIRST - CHECK FOR A
4057:D0 03      188         BNE  OKAY       ;WAVE TABLE SIZE <> 0
4059:EE 10 40   189         INC  WAVESZ     ;FORCE SIZE TO ONE PAGE
405C:           190 ;
405C:AD 0F 40   191 OKAY    LDA  WAVEPG     ;GET STARTING PAGE &
405F:8D 1F 40   192         STA  WPTRH      ;STORE INTO POINTER HIGH
4062:A9 00      193         LDA  #$00       ;LOAD A ZERO INTO
4064:8D 1E 40   194         STA  WPTRL      ;POINTER LOW
4067:18         195         CLC              ;CLEAR THE CARRY
4068:AD 0F 40   196         LDA  WAVEPG     ;NOW CALCULATE THE
406B:6D 10 40   197         ADC  WAVESZ     ;WAVE TABLE LIMIT
406E:8D 12 40   198         STA  WAVLIM     ;AND STORE FOR LATER.
4071:           199 ;
4071:A9 13      200         LDA  #>IHNDLR   ;STORE THE INTERRUPT
4073:8D FE 03   201         STA  IRQVEC     ;HANDLER ADDRESS INTO
4076:A9 40      202         LDA  #<IHNDLR   ;THE INTERRUPT VECTOR
4078:8D FF 03   203         STA  IRQVEC+1   ;
407B:           204 ;
407B:A2 47      205         LDX  #CLRIRQ    ;CLEAR THE APPLE CLOCK
407D:BD 80 C0   206         LDA  $C080,X    ;FOR INTERRUPTS
4080:A2 48      207         LDX  #CLRINT    ;
4082:BD 80 C0   208         LDA  $C080,X    ;
4085:A9 01      209         LDA  #$01       ;TURN THE APPLE CLOCK
4087:A2 49      210         LDX  #SETINT    ;INTERRUPTS ON
4089:9D 80 C0   211         STA  $C080,X    ;
408C:           212 ;
408C:68         213         PLA              ;REMEMBER THOSE REGISTERS
408D:A8         214         TAY              ;THAT WE SAVED?
408E:68         215         PLA              ;NOW WE WILL RESTORE
408F:AA         216         TAX              ;THOSE REGISTERS
4090:68         217         PLA              ;AND
4091:58         218         CLI              ;ENABLE INTERRUPTS.
4092:60         219         RTS              ;RETURN TO CALLER
4093:           220 ;
4093:           221 ;
4093:           222 ;FINISH SUBROUTINE
4093:           223 ;
4093:           224 ;
4093:48         225 FINISH  PHA              ;SAVE THOSE
4094:8A         226         TXA              ;REGISTERS
4095:48         227         PHA              ;
4096:98         228         TYA              ;
4097:48         229         PHA              ;
4098:           230 ;
4098:A9 00      231         LDA  #$00       ;TURN THE APPLE CLOCK
409A:A2 49      232         LDX  #SETINT    ;INTERRUPTS OFF
409C:9D 80 C0   233         STA  $C080,X    ;
409F:A2 47      234         LDX  #CLRIRQ    ;FOR GOOD HOUSEKEEPING
40A1:BD 80 C0   235         LDA  $C080,X    ;AWARD, YOU MUST BE
40A4:A2 48      236         LDX  #CLRINT    ;CLEAN AND NEAT
40A6:BD 80 C0   237         LDA  $C080,X    ;
40A9:           238 ;
40A9:A9 65      239         LDA  #$65       ;RESET THE INTERRUPT
40AB:8D FE 03   240         STA  IRQVEC     ;VECTOR BACK
40AE:A9 FF      241         LDA  #$FF       ;TO APPLE
40B0:8D FF 03   242         STA  IRQVEC+1   ;RESET
40B3:           243 ;
40B3:68         244         PLA              ;
40B4:A8         245         TAY              ;RESTORE THE
40B5:68         246         PLA              ;REGISTERS
40B6:AA         247         TAX              ;
40B7:68         248         PLA              ;
```

```
40B8:58          249          CLI              ;ENABLE THE INTERRUPTS
40B9:60          250          RTS              ;RETURN TO CALLER
40BA:            251 ;
40BA:            252 ;
40BA:            253 ;TURN INTERRUPTS ON
40BA:            254 ;
40BA:            255 ;
40BA:58          256 ION      CLI              ;ENABLE INTERRUPTS
40BB:60          257          RTS              ;AND RETURN
40BC:            258 ;
40BC:            259 ;
40BC:            260 ;TURN INTERRUPTS OFF
40BC:            261 ;
40BC:            262 ;
40BC:78          263 IOFF     SEI              ;DISABLE INTERRUPTS
40BD:60          264          RTS              ;AND RETURN
40BE:            265 ;
40BE:            266 ;
40BE:            267 ; MORE EQUATES
40BE:            268 ; LOCATED HERE BECAUSE OF THE
40BE:            269 ; ASSEMBLER WHICH
40BE:            270 ; EVALUATES EXPRESSIONS DURING
40BE:            271 ; PASS ONE AND NOT AT THE END
40BE:            272 ; OF THE PASS.
40BE:            273 ;
40BE:            274 ;
40BE:            275 ;
401E:            276 WPTRL    EQU  PATCH+1     ;
401F:            277 WPTRH    EQU  PATCH+2     ;
40BE:            278 ;
40BE:            279 ;

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

```
400E CHNL        04 CLKSLT      48 CLRINT      47 CLRIRQ
4093 FINISH    4013 IHNDLR    4011 INCR      404F INIT
?400B INTOFF   ?4008 INTON    40BC IOFF      40BA ION
03FE IRQVEC     405C OKAY     401D PATCH     403F RSTCLK
  49 SETINT    ?4000 START    ?4004 STOP     400F WAVEPG
4010 WAVESZ    4012 WAVLIM    401F WPTRH     401E WPTRL
```

```
  04 CLKSLT      47 CLRIRQ      48 CLRINT      49 SETINT
03FE IRQVEC    ?4000 START    ?4004 STOP     ?4008 INTON
?400B INTOFF     400E CHNL      400F WAVEPG    4010 WAVESZ
4011 INCR      4012 WAVLIM    4013 IHNDLR    401D PATCH
401E WPTRL     401F WPTRH     403F RSTCLK    404F INIT
405C OKAY      4093 FINISH    40BA ION       40BC IOFF
```

Appendix C Interface Information

The I/O Cable Assembly

The Mountain Computer I/O Cable Assembly (part number MHP-X027) is specially designed for use with the A/D + D/A board. The assembly consists of two 3 foot long cables. Each cable plugs into the A/D + D/A board. The free ends of the cable have mating DB-25 connectors. These are all-purpose cables that can be used with the A/D + D/A board applications as well as for the Self Test program that comes on the Demonstration Diskette.

At the top edge of the A/D + D/A board are two 25-pin connectors. These connectors are labeled "J1" and "J2". Each of the cables has a plastic plug on one end that will be attached to the A/D + D/A board at pin J1 and J2.

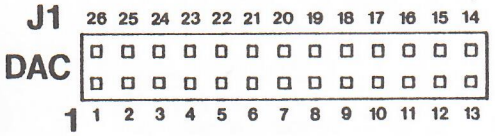
On each plug is a small marking below one of the sockets on the far right end. This is the "pin 1" locator. Be sure that on each cable this pin is attached to the left-most pin on the connector. It does not matter which cable is attached to which set of pins on the board. Attach each of the plugs to the J1 and J2 connectors.

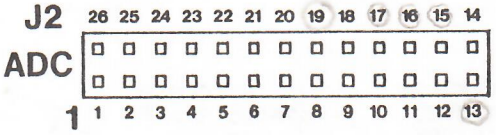
The other ends of the cables have mating DB-25 connectors with pin contacts (male) or socket contacts (female). Use these to hook up the circuits for your applications.

If you are running the Self Test program, attach the two DB-25 connectors to each other. With these connected and attached to the A/D + D/A board, you are ready to run the Self Test program.

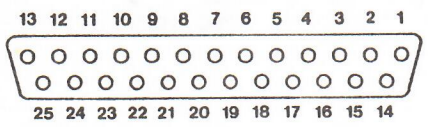
Pin-Outs

The A/D + D/A board pinouts may be identified as follows:

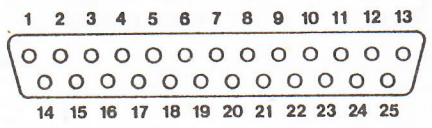




The I/O Cable pinouts may be identified as follows:



DB 25 Female



DB 25 Male

The following chart shows the relationship of board pins, cable pins and the signal on each pin:

DAC J1 Pin #	ADC J2 Pin #	DB 25 Pin #	Signal
01	01	01	channel 15
02	02	02	channel 14
03	03	03	channel 13
04	04	04	channel 12
05	05	05	channel 11
06	06	06	channel 10
07	07	07	channel 09
08	08	08	channel 08
09	09	09	channel 07
10	10	10	channel 06
11	11	11	channel 05
12	12	12	channel 04
13	13	13	channel 03
14	14	no connection	no connection
15	15	25	channel 02
16	16	24	channel 01
17	17	23	channel 00
18	18	22	See Note 1
19	19	21	Ground
20	20	20	Ground
21	21	19	Ground
22	22	18	Ground
23	23	17	Ground
24	24	16	Ground
25	25	15	Ground
26	26	14	Ground

NOTE: On DAC Pin #18, the signal is no connection; on ADC Pin #18, the signal is -5 Volt Reference.

Sample and Hold Circuitry

If you wish to use sample and hold circuitry, you should be aware that the IC numbered U9 on the A/D + D/A board is a logic gate. This logic gate contains a signal that may be used to aid in triggering sample and hold circuitry.

Pin 10 of the U9 IC emits the signal to begin a conversion. This signal is present about 500 nanoseconds prior to the beginning of the conversion. The sample and hold circuit should be stable for a full 9.5 microseconds to allow for conversion time. Another pin on the U9 IC, pin 2, signals the completion of a conversion.

Warranty

Limited Warranty for Mountain Computer Inc. Hardware

Your factory-built Mountain Computer Inc. product is warranted against defects in materials and workmanship for a period of one year from the date of delivery. We will repair or replace products that prove to be defective during the warranty period, provided they are returned to Mountain Computer Inc. No other warranty is expressed or implied. We reserve the right to refuse to repair any product that, in our opinion, has been subjected to abnormal electrical or mechanical abuse. Products less than two years out of warranty will be repaired for a nominal flat fee. Before sending your Mountain Computer Inc. unit in for repair, contact our Customer Service Representative for a Return Authorization Number.

Limited Warranty for Mountain Computer Inc. Software

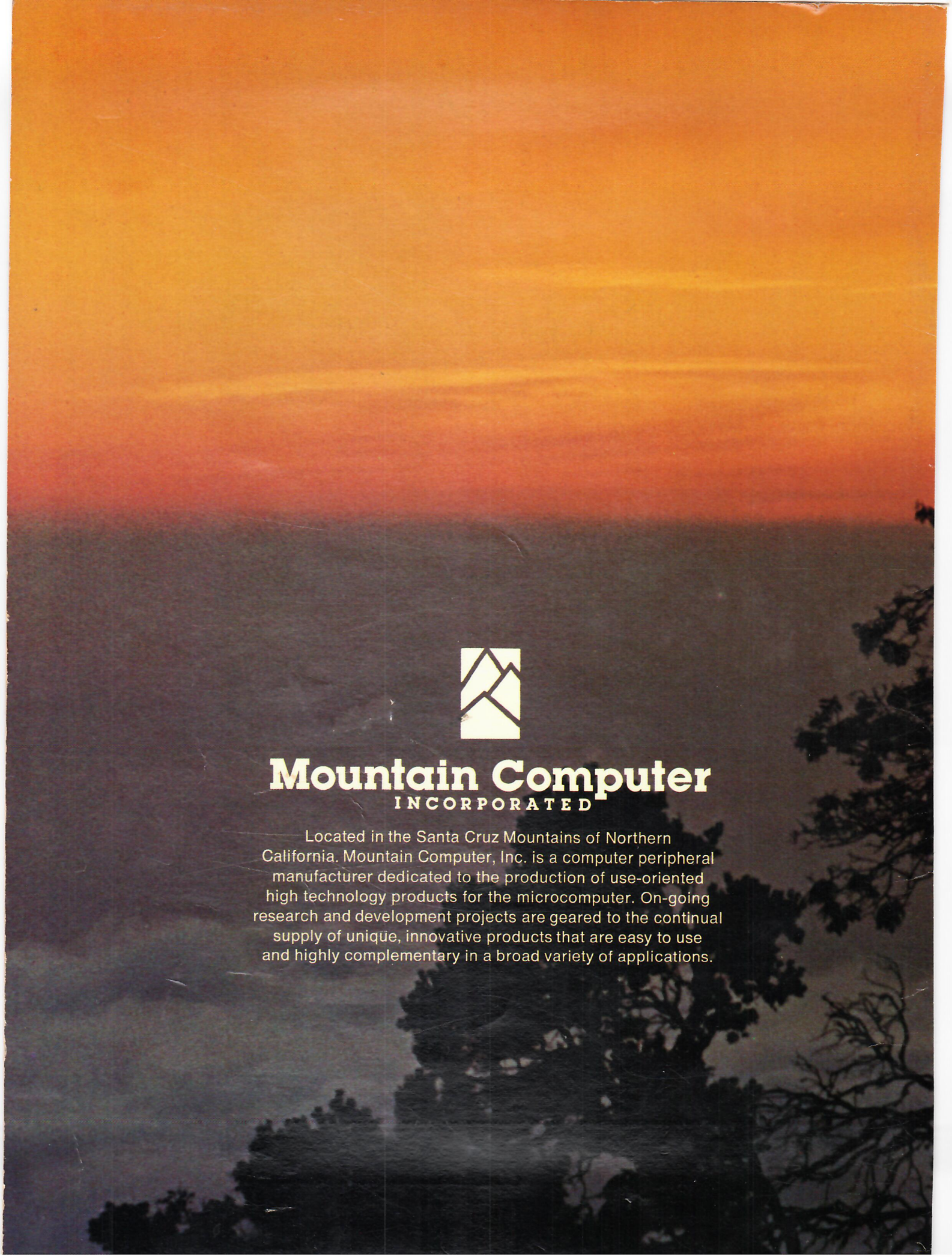
Computer software programs cannot replace your sound business judgment or make decisions for you. You, therefore, assume complete responsibility for any decisions made or actions taken based on information obtained using Mountain Computer Inc. software programs and instructional materials.

Mountain Computer Inc. software and the attached instructional material are sold "AS IS", without warranty as to their performance. The entire risk as to the quality and performance of the computer software is assumed by you.

However, to the original purchaser only, Mountain Computer Inc. warrants the software diskette or cassette to be free from defects in materials and faulty workmanship under normal use and service for a period of one (1) year from the date of purchase. If, during this one year period, a defect in the diskette or cassette should occur, it may be returned to Mountain Computer Inc. or an authorized Mountain Computer Inc. dealer for replacement of the cassette or diskette without charge to you. Your sole and exclusive remedy in the event of a defect is expressly limited to replacement of the diskette or cassette as provided above. To provide proof that you are the original purchaser, please complete and mail the attached Registration/Warranty Card to Mountain Computer, Inc. within thirty (30) days of the date of purchase.

If the failure of the diskette or cassette, in the judgment of Mountain Computer Inc. resulted from accident, abuse or misapplication of the diskette or cassette, then Mountain Computer Inc. shall have no responsibility to replace the diskette or cassette under the terms of this warranty. In such an event, replacement of the diskette or cassette is available to the original purchaser at a nominal charge.

The above warranties for goods are in lieu of all warranties, express, implied or statutory, including, but not limited to, any implied warranties of merchantability and fitness for a particular purpose, and of any other warranty obligation on the part of Mountain Computer Inc. In no event shall Mountain Computer Inc. or anyone else who has been involved in the creation and production of this product be liable for indirect, special or consequential damages, such as, but not limited to, loss of anticipated profits or benefits resulting from the use of this product, or arising out of any breach of this warranty. Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation may not apply to you.



Mountain Computer
INCORPORATED

Located in the Santa Cruz Mountains of Northern California. Mountain Computer, Inc. is a computer peripheral manufacturer dedicated to the production of use-oriented high technology products for the microcomputer. On-going research and development projects are geared to the continual supply of unique, innovative products that are easy to use and highly complementary in a broad variety of applications.