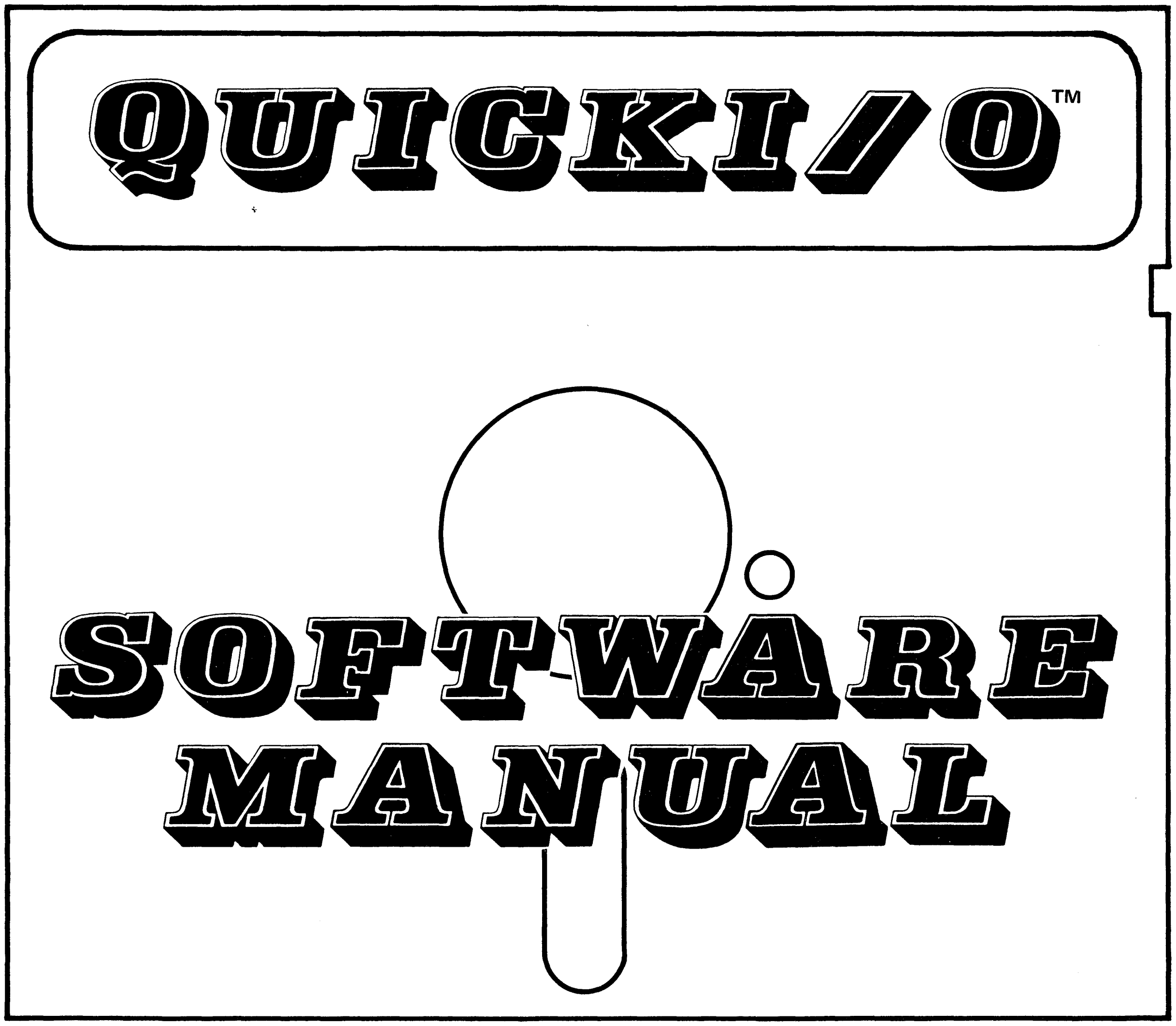
Interactive Microware, Inc.
P.O. Box 771
State College, Pa 16801
(814) 238-8294



QUICKI/O™ SOFTWARE MANUAL

By Paul K. Warme

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

Q U I C K I / O (tm) S O F T W A R E M A N U A L

By Paul K. Warme

PREVIEW OF QUICKI/O(tm) FEATURES

QUICKI/O(tm) provides a straightforward, easy-to-use method for controlling your instruments. In order to use this software, one or more ADALAB(tm) interface boards must be plugged into any slot of your APPLE computer. When you run QUICKI/O(tm), it automatically locates each ADALAB(tm) board and initializes the hardware. Thereafter, simple commands in BASIC enable you to control the ADALAB(tm) hardware, including the real-time clock and timers, digital Input/Output(I/O), serial I/O, parallel I/O and analog I/O. For example, &PIØ reads a value from Parallel Input channel Ø and stores the value in variable D%. &POØ sends the value in D% to Parallel Output channel Ø. Another form of command allows you to transfer values to and from an array called D%. For example, &PIØ,5 reads a value from Parallel Input channel Ø and stores it in D%(5). &POØ,2 sends the value in D%(2) to Parallel Output channel Ø. For ease of programming, the channel number and array element number may be replaced by any valid BASIC expression. For example, &PO(X/4),(Y+3) would transfer the value stored in D%(Y+3) to Parallel Output channel X/4. Commands such as these may be executed immediately after you type them or they may be included in any BASIC program for repeated execution under program control. Also, notice that these commands are very short, so that they are easy to remember and convenient to type.

QUICKI/O(tm) constantly displays the time in hours, minutes and seconds in the upper right corner of the screen. You may set the time or read the current time in your program, thus enabling you to start or stop external processes at particular times. In addition, a countdown timer may be set for any time interval. When the timer count reaches zero, the APPLE speaker will buzz to request your attention, but it continues to count, so that you can determine the amount of time that has elapsed since the alarm went off. The timer count may be read at any time, to an accuracy of Ø.1 second. Thus, you can time more than one event under program control.

QUICKI/O(tm) is written in machine language so that commands are executed as rapidly as possible, typically in less than Ø.ØØ2 seconds. In fact, there is no faster way to control your instruments, short of laboriously programming everything in machine language. QUICKI/O(tm) links your BASIC programs to the ADALAB(tm) hardware by a method that is very convenient and easy to learn. Also, since QUICKI/O(tm) uses only 2.25K of memory, you will have plenty of memory space left for your programs.

Included with QUICKI/O(tm) is a sample program that demonstrates all capabilities of the system. With this program, you can set or read the real time clock. Also, you may set the countdown timer to buzz after a specified period of time. Reading the timer will stop the buzzing. The digital I/O section of the sample program allows you to toggle output bits under keyboard control, while the state of each input and output bit is shown on the display screen. The general I/O section of the sample program permits you to input or output a series of values at a specified rate from the serial, parallel or analog I/O devices. You may also enter values from the keyboard, list them on the screen or plot a high-resolution graph. Most important, this sample program and other examples in this manual will show you how easy it is for you to write your own programs. Using the powerful capabilities of QUICKI/O(tm) and the ADALAB(tm) hardware, most scientists will find it quite easy to develop programs in BASIC that can control their scientific instruments.

HOW TO PRODUCE A BACKUP DISK

This program is copyrighted, which means that it is illegal for anyone to make copies of it or give copies to others. However, the original purchaser of this program is hereby granted permission to make a backup disk for his own exclusive use on a single computer system. For safety's sake, your first action should be to copy the master disk to another disk as follows:

1. Note that this program requires an APPLE II+ or APPLE II computer with Applesoft in ROM, 48K RAM and one or more ADALAB(tm) boards.

2. Mount the QUICKI/O(tm) disk, leaving the write protect tab in place. Then, type LOAD QUICKSAMPLE and press RETURN (from now on, you should always press RETURN after entering a command). After the disk stops, type BLOAD QUICKI/O.

3. Mount your copy disk (remove the write protect tab) and type SAVE QUICKSAMPLE. After the disk stops, type BSAVE QUICKI/O,A$8DØØ,L$8FØ.

HOW TO USE THE QUICKSAMPLE DEMONSTRATION PROGRAM

Before using QUICKSAMPLE, you should install the ADALAB(tm) board as described in the ADALAB(tm) hardware manual. The following explanation assumes that you have connected the cables to the self-test accessary board, although it is possible to use some parts of the QUICKSAMPLE program with other inputs and outputs from instruments, etc.

Initial Self-Test

First, type RUN QUICKSAMPLE. This program automatically loads QUICKI/O(tm), initializes the ADALAB(tm) hardware and runs the self-test on channel 0. The self-test program starts the real-time clock, verifies that the alarm (timer 1) is working, runs a test pattern through the digital I/O and parallel I/O and sends a series of values to the Digital to Analog (D/A) converter. The voltages output from the D/A converter are read by the Analog to Digital (A/D) converter and a running tally of the difference between the D/A output value and the A/D input value is printed on the screen. The D/A output value is printed in the column labelled VOUT=, the A/D input value is given under VIN=, the difference is printed in the column labelled ERROR= and the maximum difference between VOUT and VIN is given as MAX ERROR=. A separate number is given for the maximum positive and the maximum negative difference between VOUT and VIN. After these tests have been completed, you may change the output voltage by repeatedly pressing the left or right arrow keys. The left arrow key decreases the voltage, and the right arrow key increases the voltage. The first time you press one of the arrow keys, the step size is one; however, the step size increases by one each time you press the same arrow key. This will allow you to quickly scan to any possible output value and read the corresponding input value. You may press the REPT key, together with one of the arrow keys, in order to speed up the scanning. Whenever you switch from one arrow key to the other, the step size goes back down to one, so you can narrow down on a particular output voltage. To exit from the self-test on channel 0, press any key other than one of the arrow keys.

If you have plugged the ADALAB(tm) cables properly into the self-test board, all of these tests should proceed without any problems. The real-time clock test, the timer 1 test, the digital I/O test and the parallel I/O test should each print OK as the last thing on the corresponding line. The analog I/O test should display a maximum error of +10/-10. If ERROR is printed instead of OK after any test or if the analog error is greater than +10/-10, you should consult the hardware manual before proceeding. The hardware manual describes a procedure for calibrating the analog input and output values.

Setting and Reading the Real Time Clock

After the self-test has ended, the screen is erased and a list of QUICKSAMPLE options (the main "menu") appears on the screen. To start the real time clock, select the first option by typing a 1. Then, enter the current time as a 3- or 4-digit number. For example, if the time is 10:30, type 1030. You will notice that the time is now displayed in the upper right corner of the screen and the time is updated once each second.

After each program option, you will be asked to press RETURN to continue. This gives you time to read the results before they are erased. After you press RETURN, the screen will be erased and the menu will be redisplayed.

Next, you might want a report of the time. Of course, you could just look at your watch or look at the top of the screen, but please have patience and select option 2. This demonstrates that QUICKSAMPLE actually understands what time it is.

Setting and Reading the Alarm

Option 3 is used to set the countdown timer/alarm. If you have more than one ADALAB(tm) card, you must enter a CHANNEL number at this point; type 0 for the first card, 1 for the second, and so on. Then, you will be asked to enter a time interval in units of 0.1 second; for example, if you want the alarm to buzz after 3 seconds, type 30. After you set the timer alarm, you can read the time remaining by selecting option 4. If you repeatedly select option 4, you will find that it counts down at the rate of 10 per second, regardless of whatever else the program may be doing. Reading the timer/alarm disables the buzzer; thus, if you wish to hear the alarm, don't read the timer until it starts buzzing. You will note that the timer/alarm count is negative after the buzzer begins; this tells how long it has been since the alarm went off.

Digital Input and Output

Now, let's test the digital input/output by selecting option 5. The screen display shows the state of each input bit (blank for off and * for on) and it also shows the state of each output bit (0 for off and 1 for on). The bits are numbered from 7 (the most significant bit) to 0 (the least significant bit). On the Apple keyboard, keys 0 through 7 will reverse the state of the corresponding output bit. For example, if bit 2 is off (0), you can turn it on (1) by typing 2. If the ADALAB(tm) cables are connected to the self-test board, the digital outputs are connected to the digital inputs. In this case, the digital input states will always be the same as the digital output states.

General Input and Output

Option 6 selects the general purpose I/O test. First, you will be asked to select a source of input data. To input data from the keyboard, type 1. The input data is always stored in a data buffer (array D%), so after your first use of option 6, you may reuse the same input data by selecting input option 2. Options 3 through 5 select serial, parallel or analog input. Next, you will select the output device. Option 1 causes the input values to be printed on the screen, while options 3 through 5 send the values to the serial, parallel or analog outputs. Option 2 plots the input values using high-resolution graphics; in this case, you will be asked to enter a scale factor. Each input value will be multiplied by this scale factor before it is plotted. A scale factor less than 1.0 is needed if some of the input values are larger than 191, since this is the maximum Y value on the APPLE screen.

The next question asks how many values (1 to 280) you wish to input and output. Then, you will be asked to select the delay time, in units of 0.1 seconds. For example, if you want to input a value once each second, type 10. For fastest I/O, you may enter a delay time of 0, but this will be too fast for analog input, serial input and serial output.

As an example, let's input some values from the keyboard and output them to the Analog Output. First, select option 6 (GENERAL I/O) on the main menu. Then, type 1 to select the keyboard for input. Next, type 5 to select Analog Output. If you have more than one ADALAB(tm) card, you will be asked to enter a CHANNEL number. When it asks HOW MANY VALUES?, type 10 or any positive number. Since our typing speed is much slower than the Analog Output can handle, you may type 0 when it asks for the DELAY TIME. Now the program will ask you to enter VALUE 1, VALUE 2 and so on. After you type each value, it is sent immediately to the Analog Output channel selected. You may connect a voltmeter to the voltage test points on the self-test adapter in order to verify that the Analog Output is working properly. If you type the value -2047, the Analog Output will give the most negative voltage for the range you have selected (see the ADALAB(tm) hardware manual for more information about jumper options). If you type 0, the output will read zero volts. If you type 2047, the voltage will be the most positive value for the selected range. This approach may be used for calibrating the Analog Output (see the instructions in the ADALAB(tm) hardware manual).

Our second example will collect samples from Analog Input channel 0 and plot a graph. First, select option 6 (GENERAL I/O) on the main menu. Then type 5 to select Analog Input. If you have more than one ADALAB(tm) card, you must enter a CHANNEL number. Next, select option 2 for GRAPHic Output. When you are asked to enter the scale factor, type 0.0465 (this will plot the maximum negative voltage at the bottom of the screen, 0 volts in the center and the maximum positive voltage at the top of the screen). Feel free to experiment wih different scale factors, if you wish. Next, enter 50 or more for the number of VALUES and enter 1 to select a DELAY TIME interval of 0.1 seconds between samples. After you type the delay value and press RETURN, the screen will be erased and a series of points will be plotted. After the plotting ends, press RETURN to display the main menu. If the analog cable is plugged into the self-test adapter, the input voltage will be constant (whatever you last sent out on the Analog Output), so the plot should be a straight line. In fact, this is a good way to check the stability of the analog circuits on the ADALAB(tm) board. Since the plot is scaled down quite a bit, you might want to list the values stores in the data buffer in order to detect small changes in the Analog Input values that are not visible on the graph. To do this, select option 2 (DATA BUFFER) for input and select option 1 (SCREEN LIST) for output. If you wish, you may connect some other voltage source to Analog Input channel 0 and in this case, the plot will generally be a curved line.

Analog Linearity Test

Option 7 on the main menu tests the Analog Input and Analog Output circuits to make sure that the signal response is linear over their entire voltage range. Before running this test, plug the analog I/O cable into the self-test adapter board, so that the Analog Output is connected to the Analog Input. Also, make sure that the voltage range jumper is on the same range for both Analog Input and Analog Output. If you have more than one ADALAB(tm) card, the program asks you to enter a CHANNEL #. During the analog linearity test, a graph of the input voltage is plotted; it should be a straight line. After the last point is plotted, a straight line will be drawn from the last point to the first point for comparison with the plotted points. Then, the maximum differential nonlinearity is calculated by taking the difference between each successive pair of input values. Since the output value changes in steps of 15, the difference between successive input values should also be 15. However, a maximum differential nonlinearity of +2/-2 ($\pm$0.05% of full scale range) is considered to be within specifications for the +4V range. Somewhat greater nonlinearity may be observed at higher gain values. The maximum integral nonlinearity is also reported. This is calculated by comparing each input value with the predicted value, based on the average step size for the entire voltage range. Here, a maximum integral nonlinearity of +4/-4 ($\pm$0.1% of full scale range) is considered to be within specifications for the +4V range. Note that these nonlinearity figures include errors from four possible sources: the D/A converter, the output amplifier, the input amplifiers for the A/D converter and the A/D converter itself. By testing the analog I/O in this way, we tend to exaggerate the nonlinearity error.

Analog Stability Test

Option 8 on the main menu permits you to measure the short term and long term stability of the analog I/O subsystem. Before running this test, plug the analog I/O cable into the self-test adapter board. Normally, the voltage range jumper should be on the same range for the D/A as for the A/D, although this test will operate successfully with different ranges, provided that the D/A output voltage is within the range of the A/D input.

At the beginning of this test, you will be asked to enter the CHANNEL number, unless you have only one ADALAB(tm) board. Next, you will select the output voltage, which should be a value between -2047 and 2047. After this, the screen will be erased and a scale from -5 to +5 will be printed along the bottom. The channel number, output voltage (VOUT), initial input voltage (VØ) and the initial time (TØ) are printed on the second line. On the third line are printed the SCALE factor for the histogram graph, the percentage of input values which fall outside the ideal range of ±1 (ERROR%), the current input voltage (V1) and the current time (T1). A continuous stream of input values is read by the A/D converter, running at its maximum rate and a histogram of the deviations from the average input value (V1) is plotted. For example, if an input value of 1ØØ2 is read from the A/D while the average value (V1) is 1ØØØ, the +2 column on the histogram is incremented. If one of the columns runs off the top of the screen, the SCALE factor is incremented and the histogram is replotted according to the new SCALE factor.

Short term stability of the analog I/O subsystem is indicated by the distribution of values on the histogram and by the ERROR% figure on line 3. Bear in mind that V1 is the average of the most recent batch of 15 A/D input values and therefore, the histogram reflects only the short term stability. Long term stability (drift) is indicated by the difference between VØ and V1, over the time interval TØ to T1. You will note that the ERROR% value generally starts out high, because it takes a few readings before the A/D input settles to the exact value. Thereafter, the ERROR% value decreases rapidly. Also, note that ERROR% is not a percentage of full scale voltage; it is actually the percentage of values falling outside the ideal of ±1 least significant bit. This is a very stringent measure of accuracy.

Long term drift is primarily a function of temperature. If you have just turned on the computer, you may expect a long term drift of as much as 1Ø over the first hour of warmup, but thereafter, the drift will be much less. Removing the cover of the computer will also cause drift because drafts of air will change the temperature of the interface card. In many applications, long term drift is of little significance because measurements are made over a short time interval or else the application program can measure and compensate for drift.

Self-Test Program

Option 9 on the main menu selects the self-test program. As discussed earlier (see Initial Self-Test), the self-test program checks all parts of the standard ADALAB(tm) hardware. The only difference here is that you will be asked to enter a CHANNEL number, if you have more than one ADALAB(tm) interface card. Consult the earlier discussion of the initial self-test procedure for further details about interpretation. Also, you should consult the hardware manual about calibration procedures.

BACKGROUND INFORMATION ABOUT QUICKI/O(tm) PROGRAMMING

Initialization of QUICKI/O(tm)

QUICKI/O(tm) is a machine language program that quickly executes commands that you include in your BASIC programs. The easiest way to activate QUICKI/O(tm) for use in immediate mode is to type BRUN QUICKI/O. This initializes the ADALAB(tm) hardware and then returns to BASIC. Thereafter, you may type QUICKI/O(tm) commands (see below) in the immediate mode and obtain your results immediately. If you have already used BASIC to run another program, type NEW or CLEAR to clear all variables and arrays.

To use QUICKI/O(tm) in deferred execution mode, begin your BASIC program with the following statement:

```
1 HIMEM:36095: D%=Ø: DIM C%(5),Q%(5),D%(desired size):
    PRINT CHR$(4)"BRUN QUICKI/O"
```

When this statement is executed, QUICKI/O(tm) is loaded at address $8DØØ through $95FF and the ADALAB(tm) hardware is initialized. QUICKI/O(tm) automatically determines which slots contain an ADALAB(tm) card. The card in the lowest-numbered slot will be called channel Ø, the next card is channel 1, and so on. Memory location 36221 ($8D7D) contains the number of ADALAB(tm) cards found. Memory locations 36222 to 36225 tell which slots are occupied by ADALAB(tm) cards.

QUICKI/O(tm) Command Format

QUICKI/O(tm) commands are very short, easy to type and easy to remember. The first letter of every command is the ampersand (&); this tells BASIC that a QUICKI/O(tm) command is to follow. The second letter of each command selects the type of device; T selects the Real-Time clock or a Timer, D means Digital, S means Serial, P means Parallel and A means Analog. The third letter of a QUICKI/O(tm) command is either I for Input or O for Output. The fourth letter selects the channel number; this may be a number, a variable name or an expression that represents the channel number. Let's consider a few examples: &PIØ means Parallel Input on channel Ø. &DO1 means Digital Output on channel 1. What does &AIØ mean? What does &TO2 mean? Now, suppose that the variable X has a value of 2. What does &DOX mean? What channel would be read by the command &PIX/2? [&AIØ means Analog Input channel Ø, &TO2 means Timer Output channel 2, &DOX means Digital Output channel 2 and &PIX/2 means Parallel Input channel 1].

By now, you're probably wondering where the values for input and output are placed. Variable D% is used for both input and output. In other words, the result from an Input command is returned in D% and you should place a value in D% before issuing an Output command. Another form of command allows you to input and output values from array D%(). Any QUICKI/O(tm) command may be followed by a comma and a number, variable name or expression that represents the element number in D%() that contains the input or output value. For example, the command &AI0,10 returns a value in D%(10) and &PO0,10 outputs the value in D%(10). If the value of X is 10, &AI0,X and &PO0,X would give the same results.

Other Things You Should Know About QUICKI/O(tm)

After QUICKI/O(tm) has been initialized by executing statement 1 (above), commands may be executed in deferred mode (while your program is running) or in immediate mode (when your program is stopped). Bear in mind that input values are returned in variable D% or array D% and also, that values for output must be placed in variable D% or array D%. D% must be the first variable declared in your program. If you want to use array D%, you must DIMension C%(5), Q%(5), D%(desired size) as the first arrays declared in your program. Arrays C% and Q% are not actually used by QUICKI/O(tm), but they must be included to allow for upward compatibility with the ADALAB(tm) Real-Time Operating System (available soon).

After you start the clock, it continues to tick along, even if you stop your program. You can even load and run a different program without stopping the clock, but in this case you must make sure that the new program doesn't erase any part of QUICKI/O(tm) (set HIMEM:36095). If you press RESET, the clock stops and the ADALAB(tm) hardware is partially disabled. To reinitialize the hardware, type CALL 36096.

Since QUICKI/O(tm) uses memory locations 36096 to 38399, you should press RESET before running any program that uses memory above 36095. To reload and initialize QUICKI/O(tm) later, type BRUN QUICKI/O.

QUICKI/O(tm) Error Conditions

If an error is detected in any QUICKI/O(tm) command, a SYNTAX ERROR will result. Any of these conditions will cause an error:

1. The second letter is not T,D,S,P or A.
2. The third letter is not I or O.
3. The channel number is larger than the number of ADALAB(tm) boards in your computer.
4. The element number is larger than the DIMension of D%().
5. An invalid expression is given for the channel number or element number.

If such a SYNTAX ERROR occurs in a running program, BASIC will print the line number in which the error occurred. In immediate mode, no line number will be printed.

DETAILS AND EXAMPLES OF QUICKI/O(tm) FUNCTIONS

One very good way to learn how to use QUICKI/O(tm) is to study the QUICKSAMPLE source program. To obtain a listing of this program, just LOAD QUICKSAMPLE, activate your printer (using the PR# command) and then type LIST. The code for options 1 through 9 will be found at the subroutine addresses following the ON . . . GOSUB command in line 114. The examples given in the remainder of this manual will also help you to learn how to use all features of QUICKI/O(tm).

Details About the Real-Time Clock

Timer Ø is used as the real-time clock. It is very accurate because it is controlled by the quartz crystal oscillator deep in the heart of your APPLE. To set the time, set D% equal to the hour times 1ØØ plus the minutes. For example, if the time is 1Ø:3Ø, set D% to 1Ø3Ø. Then, type &TOØ. When you set the time, the seconds count is always set to zero. Thus, if you want the time to be accurate to the nearest second, wait until the seconds reading on your watch reaches zero before issuing the &TOØ command. After you set the real-time clock, the time is displayed in hours, minutes and seconds at the top right corner of your display screen. The display is updated each second. If interrupts are disabled, the clock will fall behind; however, the clock will catch up as soon as interrupts are reenabled. Since the disk operating system (DOS) disables interrupts while reading or writing, you may notice the clock falling behind temporarily during disk operations.

The command to read the real-time clock is &TIØ. This returns the time in D%, in the same format as you used to set the clock. In other words, if the time is 1Ø:3Ø, D% will be set to 1Ø3Ø. After the hour reaches 12, it is not reset to zero, but continues counting up to 13, etc. as they do in the military.

Examples Using the Real-Time Clock

This example illustrates setting and reading the clock:

```
1Ø INPUT "WHAT TIME IS IT?";D%: &TOØ          (Set the time)
2Ø INPUT "WHEN DO YOU WANT TO STOP?";T        (Enter stop time)
3Ø &TIØ: IF D%<T GOTO 3Ø                      (Wait for stop time)
4Ø PRINT "IT IS " D% "!  TIME TO STOP"        (Print time)
5Ø STOP
```

Details About the Countdown Timer/Alarm

Timer 1 is a 16-bit timer/alarm. To set the timer, place a value in D% and then type &TO1. To read the alarm, issue the command &TI1. This returns the timer value in D%. The timer continuously counts down at the rate of 10 counts per second, regardless of whether you set the timer or not. However, if you have set the alarm, the speaker will start to buzz when the count reaches zero and will continue to buzz as long as the count is negative. The buzzing will stop when you read the alarm. Note that if you read the timer while the count is still positive, the buzzing will not occur, although the timer will continue to count down.

The countdown timer is very useful for timing various activities and for starting or stopping something after a certain delay.

Examples Using the Countdown Timer/Alarm

The following program measures the length of time that Digital Input bit 1 is on.

```
10 D%=32767: &TO1                        (Set to maximum + value)
20 &DI1: IF D%=0 GOTO 20                 (Wait for bit=1 to start)
30 &TI1: T=D%                            (Read start time, T)
40 &DI1: IF D%=2 GOTO 40                 (Wait for bit=0 to stop)
50 &TI1: PRINT "DURATION = " (T-D%)/10 " SECONDS"
```

This program will work for time intervals as long as 6553.5 seconds (1.82 hours), but beyond this time, the duration will start over at 0.

In the next example, we wish to collect 10 samples from Analog Input channel 0 with a certain time interval (DELAY) between each sample. The sample values will be stored in array D%().

```
10 &TI1: T=D%: &AI0              (Read initial count & start A/D)
20 FOR SAMPLE=1 TO 10            (Loop for 10 samples)
30 T=T-DLAY                      (Subtract delay)
40 &TI1: IF D%>T GOTO 40         (Wait for delay time)
50 &AI0,SAMPLE: NEXT SAMPLE      (Read value and loop)
```

Note that DELAY is a precise time interval measured in tenths of seconds; this is much more accurate than a counting loop written in BASIC. Also note that the loop counter, SAMPLE, is used to index the values stored in D%().

Details About Timers 2 and 3

Timers 2 and 3 are each 16-bit timers that are available for your use as timers, frequency generators, frequency counters, shift registers, etc. QUICKI/O(tm) allows you to set these timers by the &TO2 and &TO3 commands. You can read these timers via the &TI2 and &TI3 commands. Timers 2 and 3 are actually timers 1 and 2, respectively, on the user 6522 chip. Initially, both are configured as one shot interval timers that count at a rate of 1.Ø23 MHz. However, you can change their mode of operation by POKEing the desired mode value in location $CN3B (or 49211+256*N), where N stands for the slot number of your ADALAB(tm) board. See the ADALAB(tm) hardware manual for more information about modes of these timers.

Examples Using Timers 2 and 3

As an example, we'll set up Timer 2 to output a continuous square wave on bit 7 of parallel output channel Ø and Timer 3 will count pulses on bit 6 of parallel output channel Ø. We will assume that your ADALAB(tm) board is plugged into slot 2 (N=2). Before running this program, bit 6 of Parallel Output channel Ø must be changed to an input. To do this, POKE the value 191 into location $CN32(for slot N, type POKE 492Ø2+256*N,191). After doing this, you may jumper bit 7 to bit 6 on Parallel Output channel Ø (connect pin 4 to pin 13 on the 16 pin DIP output cable), so that the pulses output on bit 7 will be input on bit 6. Next, RUN this program:

```
1Ø INPUT "SLOT #?";N: POKE 49211+256*N,224   (Set mode $EØ=224)
2Ø INPUT "TIMER 2 RATE?";D%: &TO2            (Set pulse rate)
3Ø &TI3: HTAB 1: PRINT D%;                   (Read & display count)
4Ø IF PEEK(-16384)<128 GOTO 3Ø               (Loop until key press)
5Ø GOTO 2Ø                                   (Enter new pulse rate)
```

Details About Timers on Extra ADALAB(tm) Boards

If you have more than one ADALAB(tm) board, 4 additional timers per board are available for your use. These timers are numbered 4 to 7 for the second board, 8 to 11 for the third board and 12 to 15 for the fourth board. Timers 4, 8 and 12 are initially set up to continuously output pulses at the rate of 1Ø per second and these pulses are counted by timers 5, 9 and 13, respectively. Thus, timers 5, 9 and 13 may be used exactly the same way as timer 1; that is, as alarm timers. You may change the frequency of timers 4, 8 and 12 with an &TO command. All of these timers count down at the 1.Ø23 MHz clock rate and output one-half pulse to timers 5, 9 or 13 each time the count reaches Ø. If you set timers 5, 9 or 13 with an &TO command, the speaker will buzz when their count becomes negative (providing you don't read them with an &TI command before the alarm begins).

Timers 6, 1Ø and 14 are exactly analogous to timer 2, while timers 7, 11 and 15 are just like timer 3. The modes of these timers are entirely up to you, as described above for timers 2 and 3. Consult the ADALAB(tm) hardware manual for further information.

Details About Digital Input and Output

Many people think that Digital I/O is the same as Parallel I/O. However, QUICKI/O(tm) makes the distinction that Digital I/O refers to a single bit, whereas Parallel I/O refers to a group of 8 bits. If you have a single ADALAB(tm) board, you may use Digital bits Ø to 7. A second board adds bits 8 to 15, a third board adds bits 16 to 23 and a fourth board adds bits 24 to 31. Actually, Digital bits Ø to 7 are the same as Parallel channel Ø. Likewise, Digital bits 8 to 15 are the same as Parallel channel 1, bits 16 to 23 are the same as channel 2 and bits 24 to 31 are the same as channel 3. In spite of this overlap in function, our distinction between Digital and Parallel I/O makes it much easier to use ADALAB(tm) for turning switches on or off and for reading switch states. This feature is included in QUICKI/O(tm) because Applesoft BASIC is poorly suited for extracting digital (bitwise) information from parallel I/O.

To read a Digital Input bit, type &DI, followed by the bit number. The value returned in D% is Ø if that bit is off. If that bit is on, D% will contain two raised to the power N-8*C, where N is the bit number and C is the channel number of the corresponding Parallel Input channel. For example, if bit 9 is on, the value returned in D% will be 2 raised to the 9-8*1 power (that is, D%=2) because N is 9 and C is 1. Normally, we don't care about the exact value of D%; we only need to determine whether D% is zero or not zero.

To write a Digital Output bit, set D% to Ø if you want to turn it off or set D% to 1 (or any other nonzero value) to turn it on. Then, issue the &DO command, followed by the desired bit number.

Examples Using Digital Input/Output

This program will familiarize you with the &DI and &DO commands:

```
1Ø INPUT "INPUT(I) OR OUTPUT(O)?"; IO$              (Select IN or OUT)
2Ø INPUT "BIT NUMBER?";B                            (Select Bit)
3Ø IF IO$="I" THEN &DIB: PRINT "VALUE="D%           (Input)
4Ø IF IO$="O" THEN INPUT"VALUE?";D%: &DOB           (Output)
5Ø GOTO 1Ø                                          (Try again)
```

Details About Parallel Input/Output

Parallel I/O means simultaneous Input or Output of 8 bits of information. This is the fastest method for communicating with instruments. A single ADALAB(tm) card enables channel Ø; a second card adds channel 1, a third card adds channel 2 and a fourth card adds channel 3.

To read a Parallel Input value, type &PI, followed by the channel number. The result returned in D% will have a value between 0 and 255. To write a Parallel Output value, put a value in the range of 0 to 255 in D% and then issue the &PO command, followed by the channel number. Often, it is convenient to use the D%() array to contain the I/O values. In this case, add a comma and the next element number at the end of the command.

In order to simplify communication between instruments and the ADALAB(tm) board, handshaking signals are provided. These signals help the computer and the instruments to synchronize their information transfer. After sending data, the sender has to make sure that the receiver has received the data. Similarly, a receiver of data has to make sure that the data is valid before accepting it. Initially, QUICKI/O(tm) assumes that you don't care about handshaking. In order to make QUICKI/O(tm) pay attention to the handshaking signals, you must POKE 36257,1 (for Parallel Input) or POKE 36273,1 (for Parallel Output). To disable handshaking mode, POKE zero in the same locations. If something disturbs the handshaking process (such as disconnecting the cable), the computer may have to wait around forever for data. QUICKI/O(tm) contains a safety measure to prevent this sort of deadlock; if you type any key during Parallel I/O, the handshaking signal will be ignored until the next time your program reads the keyboard.

Several different types of handshaking can be used with QUICKI/O(tm) (see the ADALAB(tm) hardware manual for details). To change the handshaking mode, you must POKE the desired MODE value in location $CN3C, where N is the slot number. In BASIC, use POKE 49212+256*N,MODE to do this.

Examples Using Parallel Input/Output

This program allows input or output on any parallel channel:

```
10 INPUT "INPUT(I) OR OUTPUT(O)?";IO$            (Select type)
20 INPUT "CHANNEL #?";CH                         (Select channel)
30 IF IO$="I" THEN &PICH: PRINT "VALUE="D%       (Input)
40 IF IO$="O" THEN INPUT "VALUE?";D%: &POCH      (Output)
50 GOTO 10                                       (Try again)
```

This program inputs 100 values from Parallel Input channel 1 (in slot 2) and stores the values in the D% array:

```
10 POKE 49212+256*2,8                    (Handshaking mode 8)
20 POKE 36257,1                          (Enable handshake)
30 FOR I=1 TO 100                        (Take 100 samples)
40 &PI1,I: NEXT                          (Input and loop)
```

Details About Analog Input/Output

An analog signal is a positive or negative voltage. You may select the voltage range for input and output by connecting certain pins on the ADALAB(tm) interface card (see the ADALAB(tm) hardware manual). Voltage ranges ±0.5V, ±1V, ±2V and ±4V are available. Regardless of which voltage range you select, the value -2047 corresponds to the most negative voltage, the value 0 means zero volts and the value 2047 is the most positive voltage. The channel numbers for analog input and output range from 0 to 3, depending on the number of ADALAB(tm) boards in your system.

To read an analog voltage, use the &AI command, followed by the channel number. Variable D% then contains a value between -2047 and 2047, which denotes the most negative or the most positive voltage, respectively, for the selected range. If D% is -4095 or 4095, this means that the voltage exceeds the limits for accurate voltage conversion. It is important to realize that the analog to digital conversion process takes slightly less than 0.05 seconds and therefore, it is necessary to wait at least this long before asking for a new value. Each time you use the &AI command, the previous reading is returned in D% and a new conversion is started. This approach allows your program to perform some calculations while the next sample is being taken. It also means that the first sample value is inaccurate and should be discarded. Thus, if you want to read a single value, use the &AI command once to start the conversion and then use the &AI command again after 0.05 seconds to read the actual value. If you wish to read more than one value, it is not necessary to issue the &AI command twice for each sample; just bear in mind that the value in D% was sampled immediately following the previous &AI command.

Initially, QUICKI/O(tm) is set up to ignore the signal from the A/D converter that tells when a conversion is done. However, if you POKE the value 1 into location $8DA3 (decimal 36259), a value will not be returned until the previous conversion is done. This conveniently avoids the need to write a timing loop in BASIC to wait 0.05 seconds between each reading. To return to normal mode, POKE 36259,0.

In order to write a value to the Digital to Analog converter, place a value between -2047 and 2047 in D% and use the &AO command, followed by the appropriate channel number. The value -2047 outputs the most negative voltage for the selected range, while the value 2047 outputs the most positive voltage. When the computer is first turned on, the output voltage is set to the most negative voltage for the selected range; thus, if this negative voltage could harm your instrument, be sure to disconnect the ADALAB(tm) cable before turning the computer on. When QUICKI/O(tm) is initialized (by BRUN QUICKI/O), the output voltage is changed to 0 volts. Thereafter, the output voltage will be determined by your &AO commands and the voltage will remain constant until the next &AO command. One feature of the Digital to Analog converter that you should know about is that the &AO command automatically triggers an Analog to Digital conversion on the board with the same channel number. This will affect you only if you have connected the Analog Output to the Analog Input on the same board. In this case, it is necessary to discard the first two Analog Input values after changing the Analog Output value, due to the delayed response of the Analog to Digital converter. The Digital to Analog converter works much faster than BASIC, so it is not necessary to test for conversion done. The maximum D/A conversion time is 2Ø microseconds.

Examples Using Analog Input and Output

Our first example allows input or output of a value on any valid analog channel:

```
1Ø INPUT "INPUT(I) OR OUTPUT(O)?";IO$              (Select type)
2Ø INPUT "CHANNEL #?";CH                           (Select channel)
3Ø IF IO$="I" THEN &AICH: FOR I=1 TO 5Ø: NEXT      (Wait 5Ø msec)
   : &AICH: PRINT "VALUE="D%                       (Input and print)
4Ø IF IO$="O" THEN INPUT "VALUE";D%: &AOCH         (Get value & output)
5Ø GOTO 1Ø                                         (Try again)
```

The next program takes 28Ø Analog Input samples on channel Ø at the maximum rate permitted by the conversion done signal. Each reading is scaled for plotting on the high resolution screen, with the value 2047 at the top of the screen and -2Ø47 at the bottom of the screen. At the end, the graph remains on the screen until you press any key.

```
1Ø &AIØ: POKE 36259,1: HGR              (Initialize, set graphics mode)
2Ø FOR SAMPLE=Ø TO 279                  (Take 28Ø samples)
3Ø &AIØ: Y=96-D%/21.5                   (Input and scale)
4Ø HPLOT SAMPLE,Y: NEXT SAMPLE          (Plot a point)
5Ø IF PEEK(-16384)<128 GOTO 5Ø          (Wait for key press)
6Ø TEXT                                 (Revert to text mode)
```

This program outputs a sawtooth wave form on Analog Output channel Ø. You select the frequency from Ø.Ø1 to 1ØØ waves per second (HZ). In each case, the output voltage ranges from Ø volts to the maximum for the selected voltage range. With the faster speeds, an oscilloscope trace will show a more jagged pattern, due to the large step size.

```
1Ø MN=Ø: MAX=2Ø47: SCAL=2Ø                    (Set voltage range)
2Ø INPUT "FREQUENCY (Ø.Ø1 TO 1ØØ HZ)?";FRQ    (Get frequency)
3Ø DV=FRQ *SCAL                               (Calculate step size)
4Ø FOR V=MN  TO MAX STEP DV                   (Loop for one wave)
5Ø D%=V: &AOØ: NEXT V                         (Output voltage)
6Ø IF PEEK(-16384)<128 GOTO 4Ø                (Continue until key pressed)
```

Details About Serial Input and Output (optional)

QUICKI/O(tm) includes provisions for Serial I/O using an optional APPLE Communications Card plugged in slot number 7.

This enables communication of data via a telephone modem or any other serial terminal. Serial communication means that an 8 bit unit of data is broken down into a timed series of on and off signals that reflect the state of each successive bit of the data unit. A start signal precedes the first data bit and a stop signal is transmitted following the last data bit. Although serial I/O is slower than parallel I/O, it has the advantage of requiring only 2 connecting wires.

To read a value from the Serial Input, use the &SI0 command. The value is returned in D%. To send a value to the Serial Output, place the value in D% and issue the command, &SO0. Note that only channel 0 is allowed in each of these cases. Since the data is transferred directly to and from the serial interface chip, none of the special characters described in the APPLE Communication Card Manual have any effect.

Initially, QUICKI/O(tm) is set up to ignore the handshaking signals that tell whether the serial interface chip has completed its previous operation. In this condition, you must ensure that serial data is not read or written at too fast a rate (11 values per second for 110 baud or 30 values per second for 300 baud). To make QUICKI/O(tm) pay attention to the handshaking signals, you should POKE 36255,1 for serial input or POKE 36271,1 for serial output. If you do this, QUICKI/O(tm) will wait until the serial interface chip is ready before transferring data. As a safety measure, in case no data is available for a long time, you may press any key to make QUICKI/O(tm) temporarily ignore the handshaking signals. To return to the non-handshaking mode permanently, POKE 0 into the locations indicated above.

Examples Using Serial Input and Output

This program echoes Serial Input values to the Serial Output channel in handshaking mode. The program stops when you press any key on the APPLE keyboard.

```
10 POKE 36255,1: POKE 36271,1        (Invoke handshaking)
20 &SI0: &SO0                        (Input and Echo Output)
30 IF PEEK(-16384)<128 GOTO 20       (Loop until key is pressed)
```

To store successive Serial Input values in an array, just append the element number to the command. This program reads 10 values into D%():

```
10 POKE 36255,1                      (Invoke handshaking)
20 FOR SAMPLE=1 TO 10                (Take 10 samples)
30 &SI0,SAMPLE: NEXT                 (Input and store)
```

Often, serial I/O is used for transmitting strings of letters. This program shows you how to assemble and disassemble strings for I/O. After 10 letters are input, the string is printed and then the string is output serially.

```
10 S$="": FOR L=1 TO 10            (Input 10 letters)
20 &SI0: S$=S$+CHR$(D%)            (Assemble string)
30 NEXT: PRINT S$                  (Print string)
40 FOR L=1 TO 10                   (Output 10 letters)
50 D%=ASC(MID$(S$,L,1)): &SO0      (Disassemble and output)
60 NEXT
```

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

QUICKI/O UPDATE #1

When the real-time clock is running and/or whenever interrupts are enabled for any other device, conflicts with the Disk Operating System (DOS) may occur at random times. DOS sometimes uses location $45, which is also used to store the A register contents when an interrupt occurs. The only way to prevent this conflict is to disable interrupts before executing any DOS command, including the "PR#" command. After the DOS command is finished, interrupts may be reenabled. The real-time clock will recalculate the correct time after interrupts are disabled for as long as 6553 seconds (1.82 hours). However, no data will be input or output from other devices that depend on interrupt servicing as long as interrupts are disabled.

The following short machine language subroutines will disable or enable interrupts:

```
DISABLE  SEI   ;$78=120 decimal
         RTS   ;$60=96 decimal
ENABLE   CLI   ;$58=88 decimal
         RTS   ;$60=96 decimal
```

These subroutines may be placed anywhere in memory. For example, if we put them at location 0, which is safe for BASIC, you could use the following BASIC statements to set up the subroutines:

```
POKE 0,120: POKE 1,96: POKE 2,88: POKE 3,96
```

and use them as follows:

```
CALL 0: PRINT CHR$(4)"PR#0": CALL 2
```

In order to make the DISABLE and ENABLE subroutines a permanent part of QUICKI/O, type BLOAD QUICKI/O and then POKE 38300,120: POKE 38301,96: POKE 38302,88: POKE 38303,96. Now, type BSAVE QUICKI/O,A$8D00,L$8F0. In subsequent QUICKI/O programs, you should CALL 38300 to disable interrupts and CALL 38302 to reenable interrupts.

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

USING THE ADALAB REAL-TIME CLOCK WITH VIDICHART AND OTHER BASIC PROGRAMS

Normally, the &T commands in QUICKI/O are ideally suited for setting and reading ADALAB's real-time clock and timers. However, if QUICKI/O conflicts with the memory space occupied by another program (as it does with VIDICHART) or if you want to minimize the amount of memory space devoted to ADALAB routines, you may use one of the programs described here.

The first and simplest program (ADALAB TIMER I/O #1) uses Timer 1 to count tenths of seconds and does not use interrupts. To set the time, set D% to a number (tenths of seconds) and CALL 88Ø ($37Ø). To read the time, CALL 9Ø8 ($38C), and the current time is returned in D% (in tenths of seconds). To wait for a given time, set D% to the desired time (in tenths of seconds) and CALL 921 ($399). Your program will resume when Timer 1 counts down to that value. In case of error (such as setting the desired time greater than the current time), your program may halt for as long as 1.82 hours. In this unfortunate case, you may terminate the request by pressing any key on the keyboard. The accompanying BASIC program is an example of how you can use this program to SET, GET, or WAIT for a given time and convert readings from timer 1 into hours, minutes, and seconds.

Current versions of QUICKI/O contain this program as file TIMOBJ1, and the BASIC test program is called TIMTEST1. If these programs are not on your QUICKI/O disk, enter the Monitor (type CALL -151) and type 37Ø:A9 EØ 8D, etc., copying the hex numbers in column 2 of the listing. Then, return to BASIC (press RESET) and type BSAVE TIMOBJ1, A$37Ø, L$4Ø.

You will note that TIMOBJ1 resides just above the ADOBJ machine language routine for reading ADALAB's A/D converter. Thus, you can use both the A/D routine and the timer routine in the same program. To combine the ADOBJ and TIMOBJ1 programs, type BLOAD ADOBJ,A$320 and BLOAD TIMOBJ1,A$370 and then type BSAVE ADTIM,A$320,L$90. This combined program is used for a new version of VIDICHARTAD which uses Timer 1 to accurately control the delay (DLY) between samples. Since the timer runs independently of your BASIC program, you can perform some calulations between samples in the BASIC program and still maintain an accurate time interval between samples. To use this new version of VIDICHARTAD, type LOAD VIDICHARTAD and enter the changes in lines 1010-1015 as listed. Then, type SAVE VIDICHARTADTIM to save this new version. When you use the ADC command, enter the #DELAY value in units of tenths of seconds.

The second time-keeping program (ADALAB TIMER I/O #2) uses 50 millisecond interrupts to update the time in hours, minutes, and seconds. To initialize the real-time clock, CALL 880 ($370). To set the time, POKE HOURS at 972 ($3CC), POKE MINUTES at 971 ($3CB), POKE SECONDS at 970 ($3CA), and POKE UNITS at 969 ($3C9) where UNITS are measured as 50 millisecond intervals. To read the time, read HOURS, MINUTES, SECONDS, and UNITS at the same corresponding locations. Here, the time is automatically updated, so you don't need to calculate the time from the 50 millisecond down-counter value. The accompanying BASIC program gives an example of how to SET or GET the time and WAIT for a particular time to arrive. It also shows how to display a continuously updated time on the video screen.

Current versions of QUICKI/O contain this program as file TIMOBJ2 and the BASIC test program is called TIMTEST2. If these programs are not on your QUICKI/O disk, use the monitor to enter 370:A9 E0 8D, etc., copying the second column from the listing. Then, press RESET and BSAVE TIMOBJ2,A$370,L$5F.

Both TIMOBJ1 and TIMOBJ2 assume that your ADALAB interface card is in slot 2. If this is not the case, you should change all of the $C2's in the 3-byte instructions to $C0+N, where N is the slot containing the ADALAB card. Also, note that both TIMOBJ1 and TIMOBJ2 are relocatable to any address in memory, although if you move TIMOBJ2, you must update INTADR (location $3C7) to contain the new address of the TIMINT interrupt routine.

```
                ;ADALAB TIMER I/O#1
                DPER    EQU 6002
                BASE1   EQU C200
                        ORG 0370
                        OBJ 8000
                ;SET TIME IN D%
0370  A9E0      SETIM   LDA #$E0
0372  8D0BC2            STA BASE1+0B
0375  A9BE              LDA #$BE
0377  8D04C2            STA BASE1+04
037A  A9C7              LDA #$C7
037C  8D05C2            STA BASE1+05
037F  AD0360            LDA DPER+01
0382  8D08C2            STA BASE1+08
0385  AD0260            LDA DPER
0388  8D09C2            STA BASE1+09
038B  60                RTS
                ;GET TIME IN D%
038C  AD08C2    GETIM   LDA BASE1+08
038F  AC09C2            LDY BASE1+09
0392  8D0360            STA DPER+01
0395  8C0260            STY DPER
0398  60                RTS
                ;WAIT TIME IN D%
0399  2C00C0    WATIM   BIT $C000
                ;TIMEUP ON KEYPRESS
039C  3010              BMI OUT
039E  AD08C2            LDA BASE1+08
03A1  CD0360            CMP DPER+01
03A4  D0F3              BNE WATIM
03A6  AD09C2            LDA BASE1+09
03A9  CD0260            CMP DPER
03AC  D0EB              BNE WATIM
03AE  60        OUT     RTS


LABEL TABLE
DPER    6002
BASE1   C200
SETIM   0370
GETIM   038C
WATIM   0399
OUT     03AE
```

```
]LIST

1  LOMEM: 24576:D% = 0: PRINT  CHR$
     (4)"BLOAD TIMOBJ1,A$370"
100  INPUT "SET(S), GET(G) OR WAI
     T(W) ?";A$
110  IF A$ = "S" THEN  GOSUB 9000

120  IF A$ = "G" THEN  GOSUB 9100

130  IF A$ = "W" THEN  GOSUB 9200

140  GOTO 100
9000 SETIM = 880:GTIM = 908:WTIM =
     921:D% = 32767: CALL SETIM
9010  INPUT "TIME (HOURS,MIN,SEC)
      ?";HR,MN,SC: CALL GTIM:T0 =
     D%: RETURN
9100  CALL GTIM:T1 = D%:DT = T0 -
     T1: IF DT < 0 THEN DT = DT +
     65536
9110 D% = DT / 36000:DT = DT - D%
      * 36000:HR = HR + D%
9120 D% = DT / 600:DT = DT - D% *
     600:MN = MN + D%
9130 D% = DT / 10:DT = DT - D% *
     10:T0 = T1 + DT:SC = SC + D%

9140  IF SC >  = 60 THEN SC = SC -
     60:MN = MN + 1
9150  IF MN >  = 60 THEN MN = MN -
     60:HR = HR + 1
9160  PRINT "TIME IS "HR":"MN":"S
     C"."DT: RETURN
9200  INPUT "DELAY TIME (TENTHS O
     F SECONDS) ?";DLY
9210  CALL GTIM:D% = D% - DLY: CALL
     WTIM: RETURN
```

```
]LIST 1010-

1010  IF AD = 0 THEN AD = 800: PRINT
     CD$"BLOAD ADTIM,A"AD
1012 T = 32767:D% = T: CALL 880:W
     T = 921
1015  FOR I = 0 TO NS:D% = OP: CALL
     AD:D%(I,B0) = D%:U =  USR (N
     ):T = T - DLY:D% = T: CALL W
     T: NEXT :U =  USR (H): TEXT
     : RETURN
```