ProDOS™ USER'S GUIDE

For The Thunderclock™

TABLE OF CONTENTS

1 INTRODUCTION
1 ABOUT ProDOS
1 YOUR THUNDERCLOCK & ProDOS
2 THE THUNDERCLOCK ProDOS DISK
2 ABOUT THE YEAR

3 THE THUNDERCLOCK TIME UTILITY (TUT)
3 INTRODUCTION
3 REQUIREMENTS
3 INSTALLING TUT
4 USING THE TUT COMMANDS
4 THE ALARM COMMAND
5 THE TIME COMMAND
6 THE MILLISECOND TIMER COMMANDS
7 THE DISPLAY COMMANDS—CDIS& TDIS
8 THE RATE COMMAND
8 THE SLOT COMMAND
8 THE UINT COMMAND
9 THE USER INTERRUPT ROUTINE
10 THE BSR COMMAND
11 THE NOTUT COMMAND
11 PROGRAMMING EXAMPLES & TIPS
12 MILLISECOND PRECISION
12 OVERHEAD
13 USING TUT FROM ASSEMBLY LANGUAGE

NOTICE

THUNDERWARE, INC. reserves the right to make improvements or changes in the product described in this manual at any time without notice.

INTRODUCTION

This manual is intended for those who are using a THUNDERCLOCK with Apple's ProDOS operating system, or ProDOS based applications software. If you have recently purchased an APPLE //e, then you received a ProDOS disk and some accompanying ProDOS documentation with your Disk II drive. If you plan to use your THUNDERCLOCK with Apple's older DOS 3.3 operating system, you should refer to the THUNDERCLOCK USER'S MANUAL for details. Instructions for installing your THUNDERCLOCK and more detailed technical information will also be found there.

The disk that came packaged with your THUNDERCLOCK has ProDOS on one side, and DOS 3.3 on the flip side. You should make copies of BOTH sides on separate disks and label them, and put the original away in a safe place. Now is a good time to do it!

If you plan to write your own software in BASIC, and want to do timing in milliseconds, or need access to the year, you should use ProDOS. The THUNDERCLOCK Time Utility (TUT) on the ProDOS—side of the THUNDERCLOCK disk has all the features you will need for these types of applications.

ABOUT ProDOS

ProDOS is Apple's new Professional Disk Operating System. It replaces the older DOS 3.3 and has many improvements and new features If you purchased your Apple //e after January 1984, you received a copy of ProDOS and the ProDOS User's Guide with your Disk II. Your THUNDERCLOCK disk contains the ProDOS operating system, the BASIC.SYSTEM, the ProDOS file utilities FILER and CONVERT, and the THUNDERCLOCK programs. If you are unfamiliar with ProDOS, you should read the ProDOS User's Guide that came with your system. Complete ProDOS documentation is available thru your local Apple Dealer. There is a good review of ProDOS in the February, 1984 issue of BYTE magazine.

YOUR THUNDERCLOCK & ProDOS

ProDOS is designed to automatically recognize and use your THUNDERCLOCK for time—and—date stamping of files on your disks. The THUNDERCLOCK driver is built into ProDOS. Many of the new ProDOS—based applications will take advantage of your THUNDERCLOCK's features as well. ProDOS requires an APPLE II+ with 64K or an APPLE //e. The THUNDERCLOCK Time Utility (TUT) on the ProDOS side of your

THUNDERCLOCK disk adds 13 new time—oriented commands to BASIC, making it easy to access time and date, measure intervals to milliseconds, and use interrupts. The section on "TUT" in this manual gives complete details about its features and use.

THE THUNDERCLOCK ProDOS DISK

When you boot your THUNDERCLOCK< ProDOS Disk, the STARTUP program will display the disk CATALOG and then run the CLOCK program. Press the <ESC> key to exit the CLOCK program. Your THUNDERCLOCK was set to Pacific Coast time during its final test at the factory. You can use the SET program to set it to your local time.

ABOUT THE YEAR

ProDOS makes the current year available, even though the THUNDERCLOCK does not actually keep track of the year. ProDOS calculates the year from the current month, date, and day—of—week. This means you that if you set your THUNDERCLOCK to the wrong month, date, or day—of—week, the year will probably be wrong, too. If the year appears to be wrong, make sure you've set the proper month, date, and day—of—week. Note that because the THUNDERCLOCK doesn't keep track of the year that you will need to reset the date on February 29th of leap years. The THUNDERCLOCK will always roll—over to March 1st at midnite of February 26th. During a leap year you will have to reset it to February 29th.. It will then roll—over to March 1st at midnite on February 29th.

THE THUNDERCLOCK TIME UTILITY (TUT)

INTRODUCTION

The THUNDERCLOCK Time Utility adds 13 new commands to BASIC under ProDOS. These commands make it easy for the BASIC programmer to access and display the time and date, to measure intervals to mill seconds, and to use interrupts. These new commands can be used directly from the keyboard, from EXEC files, or from within BASIC or assembly language programs.

REQUIREMENTS

THUNDERWARE's Time UTility (TUT) runs under ProDOS with the BASIC.SYSTEM. ProDOS requires an APPLE](+ with 841< or an APPLE //e. TUT installs itself above HIMEM and uses 4K of memory. TUT will automatically locate the THUNDERCLOCK, which can be in any slot 1 7 (if you have an APPLE //e with an 89-column card in the auxiliary connector, you cannot use slot 3).

INSTALLING TUT

Boot your copy of the 'THUNDERCLOCK PRODOS DISK" ·After ProDOS has finished booting and you have the BASIC prompt "3", just type "—TUT". After TUT is installed, try typing "TIME S" –the current date and time will be returned in the ProDOS system format. Now try typing "TIME F –the date and time will be returned in the "full" format including year and mill seconds. You can type "HELP" and get a summary of TUT's commands. Briefly the new commands are:

ALARM – Sets the "alarm clock" to remind you of appointments.
TIME – Returns the current date and time in a variety of formats.
START – Starts the elapsed millisecond timer.
STOP – Stops the elapsed millisecond timer.
RESET – Stops and resets the elapsed mill second timer.
TIMER – Returns elapsed time down to milliseconds.
CDIS – Continuously displays current date and time on the screen.
TDIS – Continuously displays the elapsed timer on the screen.
RATE – Returns or sets the THUNDERCLOCK interrupt rate.
SLOT – Returns the THUNDERCLOCK slot number.
BSR – Sends BSR X-18 commands (X—18 Interface Option required).
UINT – Sets user interrupt interval & interrupt routine address.
NOTUT – Removes TUT and frees the 4k of memory it uses.
HELP – Prints a summary of the TUT commands.

TUT remains installed until you use the NOTUT command, load some other BASIC utility (such as APA) or run some other ProDOS System program (such as FILER or CONVERT). You MUST remove TUT from memory with the NOTUT command before running another system program like FILER or CONVERT, or before re—booting your APPLE //with the "PR#6" command. TUT uses the interrupts from the THUNDERCLOCK, and you must use NOTUT to shut them off or other system programs may not work properly!

USING THE TUT COMMANDS

The TUT commands can be executed in either immediate (from the keyboard) or deferred (from within a program) mode. To use the new TUT commands from within a BASIC program, you must place them in a PRINT statement preceeded by a CONTROL/D character If a TUT command returns a value or a string (like the RATE or TIME commands), the PRINT statement must be followed by an INPUT statement. The INPUT statement variable list must agree with the type and number of values the TUT command expects to return. This works just like the ProDOS "PREFIX" command. See the following descriptions of the TUT commands for details and examples.

THE ALARM COMMAND

The ALARM command allows you to set the TUT "alarm clock'. When the time you specify arrives, your APPLE will "beep to remind you of whatever it was you wanted to be reminded of. You can set the alarm with an absolute time or a length of time relative to the current time. Here are some examples:

ALARM 3:45PM	—The alarm will go off at 3:45PM.
ALARM 10:00AM	—The alarm will go off at 10:00AM.
ALARM +11	—The alarm will go off 11 minutes from now.
ALARM +2:30	—The alarm will go off 2 hours and 38 minutes from now.
ALARM	—ALARM with no argument turns the alarm off.

When the time you've specified arrives, your APPLE will beep for one minute or until you turn it off by typing "ALARM" with no argument. You can set only one alarm at a time. ALARM remembers only the last time you gave it. Thus you cannot say "ALARM 2:10PM" followed by "ALARM 2:30PM" and expect the alarm to go off at 2:10PM and again at 2:30PM — it will only remember the last time you gave it and go off

at 2:30PM. The set the alarm from a BASIC program you would do it like this:

```
220 PRINT CHR$(4)"ALARM 11:32AM"
```

The alarm would now be set for 11:32AM.

THE TIME COMMAND

The 'TIME' command lets you get the current date and time in a variety of different formats. You can chose the ones best suited to your particular needs. Note that three of the formats (F, M, and #) include mill seconds, and that three formats include the last two digits of the year (S, F, #). The 7 formats available are:

COMMAND	FORMAT NAME	TIME FORMAT RETURNED
TIME S	ProDOS system format	10—FEB—84 16:22
TIME F	Full time format	84 FRI FEB 18 16:22:35.827
TIME D	Screen display format	02/10 16:22:35
TIME M	Mountain Clock format	02/10 16:22:35.827
TIME %	THUNDERCLOCK AM/PM format	FRI FEB 10 4:22:35 PM
TIME &	THUNDERCLOCK 24—hour format	FRI FEB 10 16:22:35
TIME #	THUNDERCLOCK numeric format	84,02, 05,l0, I6, 22, 35.827

If you use the 'TIME command without a format argument, the time will be returned in the last format specified. The default format is the ProDOS system format. Note that all the formats except "TIME #" return a STRING value. The "TIME W" format returns 7 REAL numeric values. If you want to read the time as a string from a BASIC program, you would do it like this:

```
120 PRINT CHR$(4)"TIME S"
130 INPUT T$
```

Then the string variable T$ would contain the time in the ProDOS system format. You could read the time in one of the other string

formats by using one of the other format characters in place of the "'S". If you want to read the time as numeric values from a BASIC program you would use the "TIME $1" command like this:

```
120 PRINT CHR$(4)"TIME #"
130 INPUT YR,MO,DW,DT,HR,MN,SC
```

Then the 7 arguments would contain, in order, the current numeric

values of year, month, day—of—week, date, hour, minute, and second (to the millisecond). The day—of—week value goes from 9 (Sunday) thru 6 (Saturday). Note that all 7 of the arguments MUST be present in the INPUT statement.

You can also display the current time continuously on the screen by using the "CDIS command. See the section on "CDIS" for details.

THE MILLISECOND TIMER COMMANDS

The "TUT" utility has a built—in timer function that allows you to measure intervals down to milliseconds for an interval up to six days long. There are four command functions associated by the timer: RESET, START, STOP, and TIMER.

RESET	–	Stops the timer if it was running & resets it to zero.
START	–	Starts the timer.
STOP	–	Stops the timer.
TIMER	–	Returns the current value of the timer. (See below).

You can alternatedly START and STOP the timer to accumulate elapsed time. You can read the timer's current value anytime, or display it continuously on the screen (see the "TDIS" command). The TIMER command returns the timer value in either of two formats:

COMMAND	FORMAT NAME	FORMAT
TIMER	Timer String Format	0 00:02:36.492
TIMER #	Timer Numeric Format	0,00,02,36.492

The format and order of the timer values returned are days, hours, minutes, seconds and milliseconds. To read the timer value as a STRING from BASIC, you would do it like this:

```
400 PRINT CHR$(4)"TIMER"
410 INPUT T$
```

Then the string variable T$ would contain the current timer value. To read the timer as four numeric values from a BASIC program, you would use the "TIMER #" command like this:

```
559 PRINT CHR$(4)TIMER if"
569 INPUT ED,EH,EM,ES
```

Then the four arguments would contain in order the elapsed days,

hours, minutes, and seconds (including milliseconds) as numeric values. Note that all 4 of the arguments MUST be present in the INPUT statement.

THE DISPLAY COMMANDS—CDIS & TDIS

TUT hag two commands that let you display the current clock time and the current timer value continuously on the screen. The 'CDIS" command controls the clock display, and the "TDIS controls the timer display. Both commands can be used with two arguments to specify the row (1—24) and column (1—40) position of the display on the screen. They also work in the 80—column display mode if you have an Apple //e and the auxiliary 80—column text card installed. In 80—column mode the column number you give as an argument must still be in the range 1—40, but will be automatically multiplied by 2. Here are some examples:

CDIS	–	Display the clock at it's default screen position.
CDIS 12,1	–	Display the clock at row 12, column 1.
CDIS OFF	–	Turn off the clock screen display.
TDIS	–	Display the timer at it's default screen position.
TDIS 2,20	–	Display the timer at row 2, column 20.
TDIS OFF	–	Turn off the timer screen display.

The default position for the timer display is row 1, column I (the upper left corner of the screen), and the default position for the clock display is row 1, column 26 (the upper right corner). Both displays are 14 characters long, so the the farthest right they can begin is column 26 (26+14=40). If you try to specify a greater value, it will be set to 26. The displays are updated on the screen 16 times/second under interrupt control. This means that when you scroll the screen, "images" of the display will scroll up from the actual display. This doesn't affect anything other than the screen – it looks messy. If your program doesn't scroll the screen, or if you position the displays in the top row (row 1), then you won't encounter this phenomenon.

Like all the other TUT commands, the display commands can be used directly from the keyboard, from an EXEC file, or from a BASIC program. For example, to turn on both displays at their default positions from BASIC, the following lines need to appear in your program:

```
470 PRINT CHR$(4)"CDIS" : REM TURN ON CLOCK DISPLAY
475 PRINT CHR$(4)"TDIS" : REM TURN ON TIMER DISPLAY
```

THE RATE COMMAND

The RATE command, when used without an argument, returns the THUNDERCLOCK's current interrupt rate (# of interrupts/second). The two interrupt rates available are 64 and 256. The THUNDERCLOCK hardware can generate a third interrupt rate of 2048, but ProDOS cannot handle such a high rate without unacceptable overhead. When TUT is loaded it checks the interrupt rate, and if it finds your THUNDERCLOCK set to the 2048 rate, it asks you to change it with the RATE command before it will work properly.

You can use the RATE command to select one of the two interrupt rates by using it with an argument:

RATE 1 – Set the interrupt rate to 64 times/second.
RATE 2 – Set the interrupt rate to 256 times/second.

Note that if you want to change the interrupt rate, you must manually place the two WRITE—PROTECT switches on your THUNDERCLOCK in the UN—PROTECTED position, just as if you were going to set the time. See the THUNDERCLOCK USER'S GUIDE for details on the WRITE—PROTECT switches.

The interrupt rate selected will determine the millisecond precision of both the TIME and TIMER commands. Be sure to read the section below on "MILLISECOND PRECISION" before changing the interrupt rate.

THE SLOT COMMAND

The SLOT command is used to return the slot number (1 thru 7) that your THUNDERCLOCK is in. Normally you won't care, since TUT automatically locates your THUNDERCLOCK and takes care of any slot—dependent stuff. But the slot number is accessible if you want it.

THE UINT COMMAND

The UINT command (User INTerrupt) gives you a simple and flexible way to use interrupts for more advanced programming applications. You specify how often a "user interrupt" should occur and the hexadecimal address of your assembly language interrupt routine. Then each time a user interrupt occurs, your assembly language routine is called automatically. Before you enable the user interrupt with the UINT command, you must be sure your interrupt routine is installed in memory and ready to be called. The general

form of the UINT command looks like this:

UINT hh:mm:ss,$address

where "hh:nu:ss" is the interrupt interval you want as hours, minutes, and seconds, and "address" is the hexadecimal address of your interrupt routine. Here are some examples:

UINT 00:00:0l,$300	— Call the subroutine at $300 once/second.
UINT 0l:00:00,$7000	— Call the subroutine at *7868 once/hour.
UINT 00:10:00,$7400	— Call the subroutine at $7468 once, every 10 minutes.
UINT 00::00:12,$300	— Call the subroutine at $398 once every 12 seconds.
UINT 03:12:46,$7000	— Call the subroutine at $7888 once every 3 hours, 12 minutes, and 46 seconds.

If you need user interrupts more frequently than once/second, you should use the UINT command with the hours, minutes, and seconds all zero:

UINT 00:00:00,$4400 — Call the subroutine at $4488 at RATE times/second.

Now your interrupt routine will be called every time a THUNDERCLOCK interrupt occurs, either 64 or 256 times per second, depending on the current RATE the THUNDERCLOCK is set to.

Note that you can disable the user interrupts once they've been started by using the UINT command with no arguments. This stops user interrupts until you issue another UINT command. They can also be disabled by your interrupt routine as explained below.

THE USER INTERRUPT ROUTINE

Each time a user—interrupt occurs, TUT will call the address specified in the UINT command with a JSR instruction. The A, X, and Y registers have already been saved, and Page Zero locations $FA thru $FF are available for temporary storage by your routine. When your routine has completed its task, it should set or clear the CARRY flag, and then return with an RTS instruction. If you set the CARRY flag (SEC) before the RTS, then your user interrupt routine will be called again on the next user interrupt. If you clear the CARRY flag (CLC) before the RTS, TUT will discontinue the user interrupts.

Your user interrupt routine must conform to the following rules:

(1) It must be fast. Especially if it is being called on every THUNDERCLOCK interrupt. ProDOS, TUT, and your foreground program are need all sharing the same Apple.

(2) When your routine is called, you cannot assume that ROM or LANGUAGE card RAM are enabled. It can be either way. The MONITOR routines may not be banked—in. Remember than some peripheral cards' firmware makes calls to MONITOR routines. If you change anything in the Apple's environment, be sure to restore them exactly as you found them.

(3) If you want to make calls to the ProDOS MLI (the Machine Language Interface) you must follow the conventions as explained in the section on "Interrupt Handling Routines in " the "ProDOS TECHNICAL REFERENCE MANUAL".

THE BSR COMMAND

The BSR command allows you to send control commands to your BSR X—10 Home Control System thru Thunderware's X—10 Interface Option connector See your THUNDERCLOCK USER'S GUIDE for details on these commands. The BSR command sends the string of control characters following the command to the X—10 Ultrasonic Interface. Here's an example:

```
BSR "ABQR"
```

This sends the commands "Module 1, Module 2, ON, OFF".

NOTE: Since the BSR command simply passes characters to the THUNDERCLOCK, you can use it to set the time" as described in the section on "SETTING THE TIME FROM BASIC" in your THUNDERCLOCK USER'S GUIDE. For example, to set the time to Friday, February 13, 11:22:00 AM, the BSR command would look like this:

```
BSR !02 5 13 11 22 00
```

THE NOTUT COMMAND

The NOTUT command turns off interrupts, removes TUT from memory, and releases the memory TUT uses back to the system. You should always do this before calling another SYSTEM PROGRAM like FILER, CONVERT, or the ED.ASM system.

PROGRAMMING EXAMPLES & TIPS

Your THUNDERCLOCK ProDOS disk has some example programs that show the use of the timer function for measuring response time to a question (RESP.EXAMPLE), and for implementing a simple delay (DELAY.EXAMPLE). You should take a look at them.

You should exercise care in setting up comparisons in IF statements that involve time or timer values. Don't look for an exact comparison, but instead look for a "<=" (less or equal) or a ">=" (greater or equal). Because the clock or timer is running asynchronously with your program, you may miss the exact comparison you're looking for. For example, if you're waiting for the timer to reach exactly 10.000 seconds you should use a statement like:

```
700  PRINT CHR$(4)"TIMER"
710  INPUT TM$
720  IF ( TM$ )= "0 00:00:10.000" THEN GOTO 1200
730  GOTO 700
```

But if you used an IF statement like:

```
700   PRINT CHR$(4)"TIMER"
710   INPUT TM$
720   IF < TM$ =  "0 00:00:10.000"  >  THEN GOTO 1200
730   GOT0 700
```

there is a chance it would not always work properly. The exact 10.000 second point might pass by between reads of the timer and you'd never see the exact comparison.

MILLISECOND PRECISION

TUT keeps track of milliseconds in both the clock and the timer by using the THUNDERCLOCK's interrupts. This means the precision of the milliseconds depends on the interrupt rate selected. When your THUNDERCLOCK is set to 64 interrupts/second (RATE 1), the resolution of the milliseconds is 1/64 of a second (0.0156). When it is set to 256 interrupts/second (RATE 2), the resolution of milliseconds is 1/256 of a second (0.0039).

Because keeping track of milliseconds is done using interrupts, there are certain conditions under which the milliseconds will not be accurate. Any time ProDOS has to do I/O (Input or Output) to a peripheral device such as a disk or printer or the screen, ProDOS disables interrupts for a short period of time. During the time ProDOS has interrupts disabled temporarily, the milliseconds will fall "out—of—synchronization" and will be in error, The amount of this error will never exceed 8.999, and the milliseconds will be accurately "re—synchronized within no more than I second after the I/O operation is completed. Thus if you are using the timer or reading the clock to milliseconds, you should wait a minimum of 1 second after an I/O operation completes to get true milliseconds.

OVERHEAD

TUT uses 4k of memory and the interrupts will make your Apple run a bit slower, depending on the interrupt rate set. When set at "RATE 1" (64 HZ), your Apple will run at about 90% normal speed. When set at "RATE 2" (256 HZ), it will run at about 67% normal speed. This reduction in apparent speed is due to the time required by ProDOS and TUT to process the interrupts, update milliseconds, keep track of the timer, and refresh the screen displays if they're being used. You should use the 256 HZ (RATE 2) interrupt rate only if your timing requires accuracy higher than the plus or minus 0.015 seconds provided by the 64 HZ rate.

USING TUT FROM ASSEMBLY LANGUAGE

You can use all of the TUT commands from assembly language as long as you are running under BASIC.SYSTEM. The TUT assembly language interface is set up much like the ProDOS MLI interface. You need to setup some parameters and then do a JSR to the TUT assembly language entry point at $8A00:

```
        SEC/CLC             ;CLC=immediate / SEC=deferred
        JSR     $8A00       ;TUT assembly language entry point
        DW      CMDSTR      ;Address of command string
        BNE     ERROR       ;Branch on TUT error
        ...
        ...
        ...
CMDSTR  ASC     "TIME F"    ;TUT command, terminated with
        DB      $8D         ; a Carriage Return character
```

You specify whether a TUT command is to be executed in the immediate or deferred mode by doing a SEC (Set Carry) or CLC (Clear Carry) before the JSR to the entry point. In the immediate mode any output returned by the TUT command will go to the screen via COUT. In the deferred mode the output will be placed in the GETLN buffer at $200, terminated with a carriage return ($8D) The address of the TUT command string must follow the JSR. The command string must be terminated with a carriage return character. After executing the TUT command, control is returned to the instruction following the command string address parameter. A Branch Not Equal instruction (BNE) at this point will cause a branch if TUT has detected an error during the call, and the A—Register will contain an error code:

ERROR CODE	MEANING
$01	Unrecognizable command
$02	Command string too long (> 40 chars)
$10	Syntax error in command or arguments

The current time can also be accessed from assembly language by doing a JSR to $8A03. This call places the curren.t date and time in the locations and in' the format shown. Values are in binary, except milliseconds as noted below:

$8A09	Year (00-99)
$8A0A	Month (1-12)
$8A0B	Day-of-week (0-6)
$8A0C	Date (1-31)
$8A0D	Hour (0-23)
$8A0E	Minute (0-59)
$8A0F	Second (0-59)
$8A10	Milliseconds (10ths, l00ths as two 4-bit BCD nibbles)
$8A11	Milliseconds (1000ths as one 4—bit BCD nibble, bits 4-7)